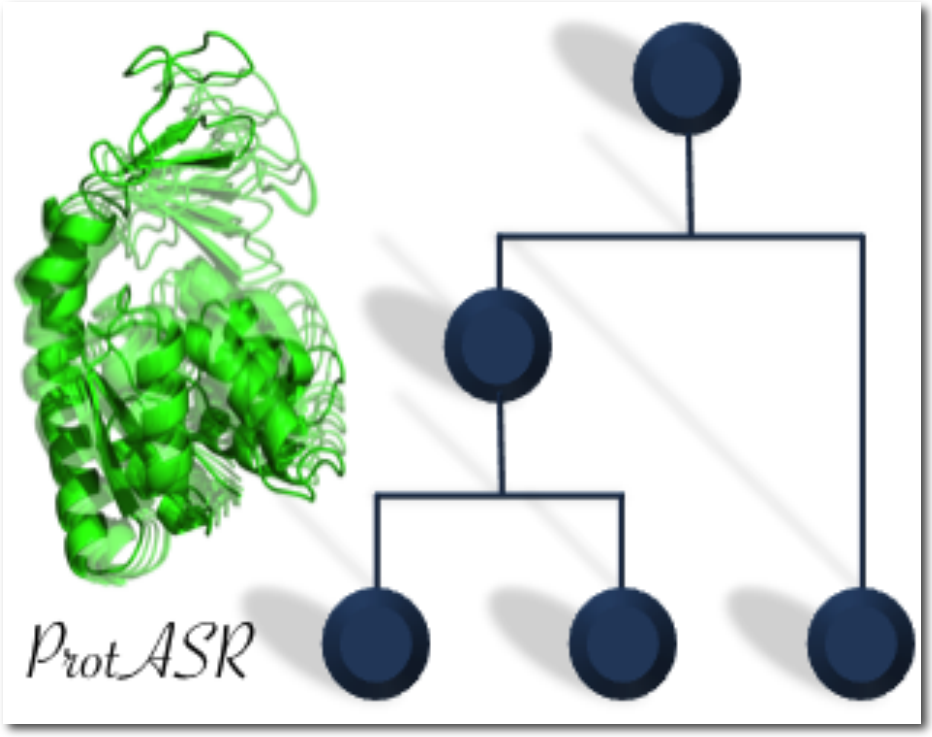
Documentation for **ProtASR2.2**

Ancestral protein reconstruction accounting for structural constrains

Current version is 2.2.0.

© 2013-2018
Miguel Arenas (miguelmmmab@gmail.com, marenas@uvigo.es) and Ugo Bastolla (ubastolla@cbm.csic.es).

April 12, 2018



Contents

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, Vigo, Spain. *Author/s: MA.*
- Centre for Molecular Biology "Severo Ochoa", Consejo Superior de Investigaciones Científicas (CSIC), Madrid, Spain. *Author/s: UB.*

1. Purpose

ProtASR is an evolutionary framework to reconstruct ancestral protein sequences accounting for structural constrains. *ProtASR* is designed to run on the command line of Linux OS or Mac OS. The program is freely available from https://github.com/miguelarenas/protasr and the first version of the framework was published in [1].

It is known that protein evolution is strongly influenced by the protein structure [e.g., 2, 3-11]. However, most of current phylogenetic frameworks do not implement structurally constrained substitution models of evolution mainly due to their mathematical complexity. We have recently developed a substitution model of protein evolution called *MeanField* (MF) [12] that accounts for the protein structure including positive and negative design (including unfolding and misfolding states) and that can generate site-specific evolutionary parameters that can be introduced into a maximum likelihood function. We already found that this model can better fit real protein evolution by maximum likelihood estimates (comparisons with the AIC criterion) and amino acid distribution across sites.
On the other hand, the reconstruction of ancestral proteins is commonly applied for a variety of purposes such as the inference of centralized proteins (centralized vaccines) [e.g., 13, 14-16] and ancestral enzymes (paleoenzymology) [e.g., 17, 18-20]. For additional applications of ancestral sequence reconstruction see [21].
Importantly, the inferred proteins should be as realistic as possible.
ProtASR arises from these aims and constitutes a fast and accurate evolutionary framework to infer ancestral protein sequences accounting for structural constrains.

The first version of *ProtASR* outperforms the traditional empirical substitution models (through MF, the framework can generate ancestral proteins that are more stable than ancestral proteins generated under the empirical substitution models). However, the new

version *ProtASR2* even outperforms the first version of the framework by incorporating an improved MF model that includes an adaptation of the Halpern-Bruno models [22] and the consideration of secondary structures. Additionally, *ProtASR2* implements a new model called *Wild Type* (WT) that considers the effect on stability and fitness of mutations of the wild-type sequence and so it is more suited for datasets with short evolutionary divergences.

The user has to specify an amino acid sequence alignment, the corresponding rooted phylogenetic tree and evolutionary parameters for the substitution model. *ProtASR* implements several structurally constrained substitution (SCS) models (see next sections) that require to specify configurational entropies, thermodynamic temperature and a Protein Data Bank (http://www.rcsb.org/PDB/home/home.do) (PDB) file, but also the framework implements a number of classical empirical amino acid substitution models such as WAG [23] and JTT [24].

Basically, *ProtASR* consists of two steps. The generation of site-specific evolutionary parameters by applying a SCS substitution model and the inference of the ancestral sequences with an adapted version of *PAML* [25] (included with the package) according to the evolutionary parameters previously generated. Therefore, the ancestral sequence reconstruction can be performed with both marginal and joint maximum likelihood (ML) approaches.
ProtASR provides ML estimates and the protein sequence for every ancestral node. Indeed, additional parameters related with the protein evolution under structural constrains (i.e., site-specific fitness, rate of change, entropy, etc) can also be provided.

2. ProtASR version A and B

Two *ProtASR* versions have been developed to accommodate the analysis of different data.

- The version A is oriented to ancestral sequence reconstruction (ASR) from an alignment with length equal to the length of the sequence of the PDB file. So here every site of the alignment is evolved with the structural constraints imposed by the corresponding site of the PDB file (e.g., site #4 of the alignment evolves with the constraints of site #4 of the PDB file).

- The version B is oriented to ASR from an alignment with length different to the length of the sequence of the PDB file, but a sequence of the alignment must be the sequence of the PDB file (and importantly gaps are allowed). So here the PDB sequence must be included in the alignment and properly identified (see “Target alignment” section), then only sites with states (different to gaps) in such a sequence are modelled with the structurally constrained model. Sites with gaps in such a sequence are allowed but are modelled with an empirical substitution model since only sites existing in the PDB file present structural information.

3. Installation and execution ProtASR

3.1. Software installation

The *ProtASR* framework runs on the command line and requires *Perl* is installed. *Perl* can be downloaded from http://www.perl.org/.

The folder "src" (of any *ProtASR* version) contains the source codes of *Prot_evol* and the adapted version of *PAML* [25] to *ProtASR*, included in the folders "Prot_evol_src" and "PAML_src", respectively. This provided PAML code should be applied (other PAML codes could lead to incorrect reconstructions), see next sections. Both programs can be compiled with the compiler *gcc*.
The folder "scripts" contains several scripts required to execute *ProtASR*.
These folders must not be modified by the user as they are required to run the framework.

3.2. Run ProtASR

First, *ProtASR* must be compiled for the OS through a Makefile that is placed on the main directory,

```
make all
```

With this specification both *Prot_evol* and *PAML* will be compiled and the corresponding executable files will be placed in the folder "*src*".

To execute *ProtASR*, all the compiled files (previous phase) together with a "Settings.txt" file and other input files must be present in the same directory and then just type,

```
perl ProtASR_main.pl Settings.txt
```

Information will be printed on the screen (see next section for details).

Note that "Settings.txt" requires other input files placed in the working directory (see next section and "examples"). Particularly, an alignment of protein sequences in Nexus format that must include a rooted tree in Newick format (see next section and "examples"). In addition, SCS models also require a PDB file (see next section and "examples").

After using *ProtASR*, executable files can be removed by,

```
make clean
```

This will free up hard disk space.

4. *ProtASR* Usage

The main input file of the program is the Settings file, where the user must specify the substitution models and additional files to be used for ASR.

A standard run of *ProtASR* framework requires the following files from the user:

- **Settings file**. This input file contains the information for the inference. It should be modified by the user with the desired model and parameters (see details below). Note that the name of this file must be specified to run *ProtASR* (e.g., *perl ProtASR_main.pl Settings.txt*).
- **Target coding alignment and rooted tree**. The protein sequence alignment (Nexus format) and rooted tree (in Newick format) that will be analyzed must be provided by the user. **This file should be in the working directory and its name should be indicated in the Settings file.** See details in section 3.2.

- **PDB file**. The ASR under any SCS model requires a PDB file that must be placed in the working directory and specified in the Settings file.

Examples of a Settings and target protein alignment files are provided in the package (folder "examples").

After concluding a full run of *ProtASR* a folder called "RESULTS" is created in the working directory (see details in next sections).

4.1. The Settings input file

The Settings input file must contain all the information required to perform the ASR and should be carefully checked by the user because mistakes can alter the results (error messages can also be displayed on the screen and will suggest stop the execution by CTRL+C).
The Settings file includes three main blocks: (1) the alignment and rooted tree, (2) the general settings for the ASR and, (3) the specific settings for SCS models.

Important notes when dealing 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 to identify the parameter).
- Some parameters are mandatory and must be specified, these parameters contain an "*" (see below).
- Optional parameters can be "turned-off" by an "#" at the beginning of the line.

4.1.1. Settings

Alignment and tree

- **Name of the file with the target alignment of protein sequences and the corresponding rooted tree.** This parameter is mandatory. Only specify the filename (i.e. no pathway) since the alignment file needs to be placed in the same directory of the Settings file. For example,

```
 ### Target alignment file with a rooted tree ### # nexus format with a rooted tree, see documentation and examples
*NameOfNexusFile=CytochromeP450_Aln_Tree.nex
```

General settings for the ASR

- **Substitution model.** This parameter is mandatory. A substitution model must be introduced. For SCS models type "MEAN-FIELD". Indeed, a variety of empirical amino acid models are available: Blosum62 [26], cpREV64 [27], Dayhoff [28], DayhoffDCMUT [29], G1974a [25], G1974c [25], G1974p [25], G1974v [25], Grantham, HIVb [30], HIVw [30], JTT [24], JonesDCMUT [29], LG [31], Miyata, MtArt [32], MtMam [33], MtRev24 [34], MtZoa, RtRev [35], VT [36], WAG [23]. For example,

```
# Substitution model: MEANFIELD (requires specification of settings in the following section), Blosum62, cpREV64, Dayhoff, DayhoffDCMUT, G1974a, G1974c, G1974p, G1974v, Grantham, HIVb, HIVw, JTT, JonesDCMUT, LG, Miyata, MtArt, MtMam, MtRev24, MtZoa, RtRev, VT, WAG
*SubsModel=MEANFIELD
```

- **Apply frequencies of the model.** This parameter is mandatory. SCS models are not influenced by this parameter but empirical models are. Note that if this parameter is specified for an empirical model, the number of free parameters of the model increases in 19 (20 amino acid frequencies - 1 (normalization)). For example,

```
# Consider frequencies from the model (+F) (0: No, 1: Yes)
*ModelFreqs=0
```

- **Type of gamma shape (+G) parameter.** This parameter is relevant for empirical models. If 0, it is estimated. If 1, it is fixed. The value of alpha can be provided in the following setting. For example,

```
# Estimate (0) or fix (1) gamma shape parameter
*TypeG=1
```

- **Gamma shape (+G) parameter.** This parameter is relevant for empirical models. If 0, it is not applied. Otherwise introduce the alpha value (if in the previous setting +G is estimated, now introduce the initial value for the estimation; if in the previous setting +G is fixed, now introduce the value). For example,

```
# Estimate (0) or fix (1) gamma shape parameter
*TypeG=1
```

- **Heterogeneous +G for genes.** This parameter is relevant for empirical models. If 0, alpha is constant along the sequence. Otherwise, alpha can vary under a number of categories that must be here specified. For example,

```
# Different alphas for genes, introduce a number
*Malpha=0
```

- **Type of correlation parameter ρ.** This parameter is relevant for empirical models. If 0, it is estimated, if 1 it is fixed (see next setting). Additional details about this parameter can be found in [25]. For example,

```
# Estimate (0) or fix (1) rho (correlation parameter)
*TypeRho=1
```

- **Correlation parameter ρ.** This parameter is relevant for empirical models. If 0, it is not applied. Otherwise introduce the alpha value (if in the previous setting it is estimated, now introduce the initial value for the estimation; if in the previous setting it is fixed, now introduce the value). For example,

```
# Rho (correlation parameter). initial or fixed rho, 0:no correlation
*RhoCorr=0
```

SCS models

- **PDB file.** Name of the PDB file. For example,

```
# PDB file (must be placed in the current directory)
*PDBfile=1Z10.pdb
```

- **Chain of the PDB file.** Indicate chain of the PDB file. For example,

```
# Chain of the PDB file
*CHAIN=A
```

- **Temperature.** Thermodynamic temperature. We recommend the default value (0.5) but values between 0.1 and 1 could be also applied. For further details see [2]. For example,

```
# Temperature
*TEMP=0.5
```

- **Configurational entropy per residue (unfolded).** We recommend the default value (0.065) but see other values in [2]. For example,

```
# Configurational entropy per residue (unfolded)
*SU1=0.065
```

- **Configurational entropy per residue (misfolded).** We recommend the default value (0.065) but see other values in [2]. For example,

```
# Configurational entropy per residue (misfolded)
*SC1=0.065
```

- **Configurational entropy offset (misfolded).** We recommend the default value (0) but see other values in [2]. For example,

```
# Configurational entropy offset (misfolded)
*SC0=0.0
```

- **Random Energy Model (REM) moment for misfolding.** We recommend the Random Energy Model [37] with moment 2 (see [2]). For example,

```
# Use up to 1,2,3 moments of misfolding energy?
*REM=2
```

- **Coefficient of local interactions.** This parameter is available form the second version of *ProtASR* and refers to the consideration (1) or not consideration (0) of constraints from the secondary protein structure. We recommend explore both options and finally apply that one that provides the highest likelihood. For example,

```
# Coefficient of local interactions
*A_LOC=0
```

- **Contacts Map file.** A file with a list of contact matrices must be specified. However, the folder "src/Prot_evol_src" already includes a file "*structures.in*" with a huge list of contact matrices downloaded from the PDB and that can be applied to compute energies from any PDB protein structure. Thus, *ProtASR_main.pl* will automatically read "*structures.in*" but the file must be in the working directory. For example,

```
# Contacts map (must be placed in the ProtEvol directory)
*FILE_STR=structures.in
```

Optimization of the SCS models

- **Number of substitutions to simulate data.** This parameter is dedicated to perform computer simulations (could be useful for other experimtents). In *ProtASR* we recommend set it to 0 by default. For example,

```
# Number of substitutions to simulate data (0 by default, not required for
ASR)
*TMAX=0
```

- **Lambda.** This parameter is required for the internal optimization of SCS models, which is improved in the second version of the framework (*ProtASR2*). By default we recommend its value 0.9, but values between 0.1 and 2 can be explored. In any case, see next three settings. For example,

```
# LAMBDA~ NPOP*exp(-DELTA G/TEMP)
*LAMBDA_par=0.90
```

- **Optimize Lambda.** If 0, Lambda is not optimized (applying the value introduced in the previous setting), otherwise it is optimized by ML computations. See also next settings. For example,

```
# Optimize LAMBDA? (0: No, 1: Yes, default)
OPT_LAMBDA=1
```

- **Target Lambda.** If Lambda is optimized (previous setting), the target Lambda can be introduced. We recommend -1 by default, negative values are recommended to search for stable proteins. For example,

```
# Target value of DeltaG if OPT_LAMBDA
DG_OPT=-1
```

- **Optimization criterion for Lambda.** If Lambda is optimized (previous settings), the optimization can be performed under different models. "NAT": consideration of the native PDB structure, "DG": searching for the lowest energy, "ALL": accounting for both "NAT" and "DG". We found that "ALL" presented the best performance and we recommend its application. For example,

```
# Optimization criterion. Allowed: NAT ALL DG
*MODEL=ALL
```

- **MF model or WT model.** This option is available from *ProtASR2*. Here the user can specify the MF model (0) or the WT model (1). We found that WT better fits with most of our studied proteins but still we recommend explore both options and finally apply that one that provides the highest likelihood. For example,

```
# WildType model: No (0) or yes (1)
*WTmodel=1
```

Mutation in the SCS models

- **Frequencies in the global model.** Site-specific amino acid frequencies can be considered for the global estimation by applying just the mean or applying weighted frequencies. However we found that both strategies present similar results. For example,

```
# Global matrix. Mean (0) or mean weighted by frequencies (1)
*GLOBALMATRIX=0
```

- **Exchangeability process.** This parameter is very important for SCS models. Four models are allowed: "MUT" (based on a DNA mutation process for which DNA evolutionary parameters must be specified, see settings below for this specific model), "EXCH" a model based on exchangeability where the fitness of the structural constrains can be over represented (by the empirical process and the PDB structure), "FLUX" (like EXCH but improve it to remove a fitness effect), "RATE" (based on the rates of change across sites). Our results indicated that the "FLUX" model is the best in terms of ML. For example,

```
# Exchangeability. Allowed: MUT, EXCH, FLUX, RATE
*EXCHANGE=FLUX
```

- **Empirical rate matrix.** SCS models can consider empirical matrices for sampling (not evaluating) potential mutations and we found that this can improve the accuracy of the structural model. The current version only implements the models JTT

and WAG (in general, empirical models can be selected with tools such as *ProtTest* [38]). If the version B of *ProtASR* is applied, sites not included in the PDB file will be evolved under the empirical model here specified. For example,

```
# Rate matrix. Allowed: JTT, WAG
*MATRIX=JTT
```

- **Data considered for frequencies.** This parameter is fundamental for *SCS* models. If 0: DNA frequencies are user-specified (see next settings), can be important for "MUT" model. If 1: DNA frequencies are fitted with the protein sequences. If 2: Amino acid frequencies are estimated from the alignment of protein sequences. The option 2 presented the highest ML estimates and it is recommended for ASR. For example,

```
# Get nucleotide frequencies from sequence? (0: Use input nucleotide frequencies, 1: Fit nucleotide frequencies from prot sequences, 2: Fit amino acid frequencies from prot sequences)
*GET_FREQ=2
```

DNA evolutionary parameters for the "MUT" SCS model

- **Frequency for A.** Frequencies for nucleotide A. For example,

```
# Frequency for A
*fA=0.25
```

- **Frequency for T.** Frequencies for nucleotide T. For example,

```
# Frequency for T
*fT=0.25
```

- **Frequency for C.** Frequencies for nucleotide C. For example,

```
# Frequency for C
*fC=0.25
```

- **Frequency for G.** Frequencies for nucleotide G. For example,

```
# Frequency for G
*fG=0.25
```

- **CpG transition ratio.** GC content. The parameter refers to the CG transition ratio. For example,

```
# CpG transition ratio
*kCpG=2
```

- **Transition transversion ratio rate (Kappa)**. For example,

```
# Transition transversion ratio (Kappa, >1)
*TT_RATIO=1.3
```

- **Ratio between 1-nuc and 2-nuc mutations.** This parameter allows single and double mutations per mutation event. If 0, only single mutations are allowed. If 1, the number of single mutations is equal to the number of double mutations. We found best results when this parameter is very close to 0. For example,

```
# Ratio between 1-nuc and 2-nuc mutations (0-1)
*TWONUCMUT=0.25
```

4.2. Target alignment and rooted tree

The name of the file with the alignment of protein sequences and the rooted tree must be indicated in the Settings file (specified in the parameter *NameOfNexusFile*). Importantly, the alignment file must be placed in the working directory.
The alignment must be presented in *nexus* format and at the end of the file a rooted tree must be included. A variety of examples are provided in the *ProtASR* package (in the folder "examples").

Illustrative example for version A of *ProtASR*,

```
#NEXUS

[
Real data set from Pfam
]

Begin data;
Dimensions ntax=5 nchar=15;
        Format datatype=protein gap=- missing=? matchchar=.;
        Matrix
1Z10_A  ASWLDVMIPKNNPST
CP2A6_H  ANWLDVMIPKNNPST
Q307K8  ANWLDVMIPKNNPST
Q8SQ68  ASTLVVMILKLNPST
CP2A5_M  ASWLDVMIPKNNTST
        ;
End;


BEGIN TREES;
       TREE                                part_1                                =
((CP2A6_H:0,((Q8SQ68:0.04,CP2A5_M:0.06):0.01,Q307K8:0.03):0.07):0.001,1Z10_A:0
.001);

END;

BEGIN ASSUMPTIONS;
       CHARSET span_1 = 1-15;

END;
```

The version B of *ProtASR* must include a line with the name of the taxa of the sequence of the PDB file. This taxa must be specified in "*PDBtaxa=*" and must exist in the alignment. For example,
PDBtaxa=Taxa_PDBsequence
Illustrative example for version B of *ProtASR*,

```
#NEXUS

[
Real data set from Pfam

PDBtaxa=1Z10_A
]

Begin data;
Dimensions ntax=5 nchar=17;
        Format datatype=protein gap=- missing=? matchchar=.;
        Matrix
1Z10_A  ASWLDVMIPKNNPST--
CP2A6_H  ANWLDVMIPKNNPST-S
Q307K8  ANWLDVMIPKNNPST--
Q8SQ68  ASTLVVMILKLNPSTNN
CP2A5_M  ASWLDVMIPKNNTSTA-
```

```
        ;
End;


BEGIN TREES;
       TREE                                  part_1                                  =
((CP2A6_H:0,((Q8SQ68:0.04,CP2A5_M:0.06):0.01,Q307K8:0.03):0.07):0.001,1Z10_A:0
.001);

END;

BEGIN ASSUMPTIONS;
       CHARSET span_1 = 1-15;

END;
```

Taxa names should be separated to the corresponding sequence by any space/s.

Importantly, under SCS models, the alignment length must be equal to the length of the protein of the PDB file and, every site of the alignment must be homologous with every site of the protein of the PDB file. Otherwise, incorrect results may be generated. This is important because every site must be present in the protein structure to be evolved under structural constrains.
A recommended strategy is to obtain the sequence of the PDB file and then, align this sequence with all the sequences of the nexus alignment. Then remove those sites that are not included in the PDB sequence (to keep for homology, keep gaps that the PDB sequence may induce in the other sequences). Finally, if desired, remove the PDB sequence from the alignment (otherwise note that this sequence must also be included in the input phylogenetic tree).

4.3. Screen information

The information printed on the screen during an execution example is shown below.

-> As an illustrative example we next run the example “Cytochrome_P450_Input” of the *ProtASR2* package (version A).

First we move all the files and folders present in the directory “src” to the directory “Cytochrome_P450_Input” (see also Compilation section), the latter is now our working directory:

```
Cytochrome_P450_Input Miguel$ ls
1Z10.pdb     Makefile     ProtASR_main.pl     Scripts
       CytochromeP450_Aln_Tree.nex      PAML_src     Prot_evol_src
       Settings.txt
```

-> **Compile *ProtASR*** (it will compile *Prot_evol* and *PAML*, and then it will move the executable files to the working directory).

```
Cytochrome_P450_Input Miguel$ make all

Compiling Prot_evol ..
/Applications/Xcode.app/Contents/Developer/usr/bin/make -C Prot_evol_src clean
rm -fr Prot_evol_mean_field.o REM.o Mean-field.o meta_Mean-field.o random3.o
Fit_mutation.o Get_pars_pop_dyn.o Codes.o Input.o gen_code.o mutation.o
allocate.o read_pdb.o read.o Print_Mean-field.o output.o fits.o
optimization.o Prot_evol
/Applications/Xcode.app/Contents/Developer/usr/bin/make -C Prot_evol_src all
```

```
gcc -O2   -c -o Prot_evol_mean_field.o Prot_evol_mean_field.c
gcc -O2   -c -o REM.o REM.c
gcc -O2   -c -o Mean-field.o Mean-field.c
gcc -O2   -c -o meta_Mean-field.o meta_Mean-field.c
gcc -O2   -c -o random3.o random3.c
gcc -O2   -c -o Fit_mutation.o Fit_mutation.c
gcc -O2   -c -o Get_pars_pop_dyn.o Get_pars_pop_dyn.c
gcc -O2   -c -o Codes.o Codes.c
Codes.c:102:54: warning: array index 4 is past the end of the array (which
contains 4 elements) [-Warray-bounds]
  sscanf(string, "%s%s%s%s", nuc[0], nuc[1], nuc[3], nuc[4]);
                                                     ^   ~
Codes.c:98:3: note: array 'nuc' declared here
  char string[200], nuc[4][3];
  ^
1 warning generated.
gcc -O2   -c -o Input.o Input.c
gcc -O2   -c -o gen_code.o gen_code.c
gcc -O2   -c -o mutation.o mutation.c
gcc -O2   -c -o allocate.o allocate.c
gcc -O2   -c -o read_pdb.o read_pdb.c
gcc -O2   -c -o read.o read.c
gcc -O2   -c -o Print_Mean-field.o Print_Mean-field.c
gcc -O2   -c -o output.o output.c
gcc -O2   -c -o fits.o fits.c
gcc -O2   -c -o optimization.o optimization.c
gcc  Prot_evol_mean_field.o  REM.o  Mean-field.o  meta_Mean-field.o  random3.o
Fit_mutation.o  Get_pars_pop_dyn.o  Codes.o  Input.o  gen_code.o  mutation.o
allocate.o   read_pdb.o   read.o      Print_Mean-field.o   output.o   fits.o
optimization.o -o Prot_evol -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 codeml (PAML) ..
/Applications/Xcode.app/Contents/Developer/usr/bin/make -C PAML_src
cc    -O4  -funroll-loops  -fomit-frame-pointer  -finline-functions  -o  codeml
codeml.c tools.c -lm
clang: warning: -O4 is equivalent to -O3
clang: warning: -O4 is equivalent to -O3

… Text …

Done!

Cytochrome_P450_Input Miguel$ ls
1Z10.pdb     Makefile     ProtASR_main.pl     Prot_evol_src       Settings.txt
CytochromeP450_Aln_Tree.nex      PAML_src     Prot_evol    Scripts      codeml
```

Notice that now the executable files *Prot_evol* and *codeml* are present in the directory.

Alternatively, executable files for Linux Debian and Mac OS X 10.10.5 (Yosemite) are placed in the folder "*bin*". To apply them, just move them to the working directory.

-> Before running *ProtASR* check that **all the required files and folders are present**, including the Settings file, PDB file and alignment with the tree.

```
 Cytochrome_P450_Input Miguel$ ls
1Z10.pdb     Makefile     ProtASR_main.pl     Prot_evol_src       Settings.txt
CytochromeP450_Aln_Tree.nex      PAML_src     Prot_evol    Scripts      codeml
```

-> Run *ProtASR*.

```
Cytochrome_P450_Input Miguel$ perl ProtASR_main.pl Settings.txt
```

```
*********************************************************************************
*********************************
- ProtASR.pl 2.2 -
Miguel Arenas & Ugo Bastolla
Centre for Molecular Biology "Severo Ochoa" - CSIC
Madrid
2014-2018

ProtASR does:
- Read Settings file
- Read nexus alignment file (should include tree/s, see examples)
- Generate a matrix per site with Prot_evol
- Run Prot_evol
- Generate input files for PAML
- Optimize a global tree with PAML
- Run ASR per site with PAML
- Generate ancestral sequences
ProtASR requires PAML installed to run on the command line (executable file
must be in the "source" folder)
*********************************************************************************
*********************************


>>> Input file uploaded: Settings.txt

>>> ProtASR directory detected: .

>>> Settings directory detected: .

>>> OS detected: darwin (Mac OS X):
  darwin
  darwin-thread-multi-2level

>>> Checking executable files ...

   ./PAML_src/codeml was detected ... Ok!
   ./Prot_evol_src/Prot_evol was detected ... Ok!

>>> Moving executables ...


>>> Creating a RESULTS folder (previous "RESULTS" folder is removed) ...
```

Reading the input file (this should be fast) ..

```
>>> Reading Settings from input file ...

 > Dataset and tree/s
   Target      alignment      file:      CytochromeP450_Aln_Tree.nex      ...
./CytochromeP450_Aln_Tree.nex was detected ... Ok!
   Substitution model: MEANFIELD
   Apply frequencies from the model (0: No, 1: Yes): 0
   Estimate (0) or fix (1) gamma shape parameter: 1
   Gamma shape parameter value. Initial or fixed alpha, 0:infinity (constant
rate): 0
   Different alphas for genes, introduce a number: 0
   Estimate (0) or fix (1) rho (correlation parameter): 1
   Rho (correlation parameter). Initial or fixed rho, 0:no correlation: 0


 > Settings for the "Meanfield" substitution model
  > Defining the protein
   PDB: 1Z10.pdb ...    ./1Z10.pdb was detected ... Ok!
   CHAIN: A
  > Thermodynamics
   TEMP: 0.5
```

```
   Configurational entropy per residue (unfolded): 0.065
   Configurational entropy per residue (misfolded): 0.065
   Configurational entropy offset (misfolded): 0.0
   Moment of misfolding energy (REM): 2
   Coefficient of local interactions: 0
   Contacts map file: structures.in ...   ./structures.in was detected ... Ok!
  > Meanfield model
   Number of substitutions to evaluate the model (TMAX): 0
   LAMBDA: 0.90
   Optimize LAMBDA: 1
   Target value of DeltaG (if OPT_LAMBDA): -1
   MODEL: ALL
  > Wild Type model
   Wild Type model: 1
  > Mutation model
   Global matrix: 0
   Matrix: 0
   Exchangeability: FLUX
   Matrix: JTT
   Get nucleotide frequencies from sequence: 2
   Improve mutation parameters after selection?: 0
   Frequency for A: 0.25
   Frequency for T: 0.25
   Frequency for C: 0.25
   Frequency for G: 0.25
   CpG transition ratio: 2
   Transition transversion ratio (Kappa): 1.3
   Ratio between 1-nuc and 2-nuc mutations: 0.25
```

Reading the alignment/tree file (this should be fast) ..

```
>>> Reading Settings from data file "./CytochromeP450_Aln_Tree.nex" ...
   CytochromeP450_Aln_Tree.nex with sample size 5, length 465 amino acids and
1 partitions (trees)

>>> Creating a fasta alignment from the nexus alignment ...

-          NexusToFasta_MA.pl          ->          Directory          detected:
/Users/Miguel/Desktop/Cytochrome_P450_Input
> Input file uploaded: CytochromeP450_Aln_Tree.nex
> Reading input file "CytochromeP450_Aln_Tree.nex" ...


>>> Extracting tree from the nexus alignment ...

-          NexusToTree_MA.pl          ->          Directory          detected:
/Users/Miguel/Desktop/Cytochrome_P450_Input
> Input file uploaded: CytochromeP450_Aln_Tree.nex
> Reading input file "CytochromeP450_Aln_Tree.nex" ...


>>> Generating input file for Prot_Evol "./Prot_Evol.in" ...
```

Running *Prot_evol* (this should go fast) ..

```
>>> Running Prot_evol ...

Reading parameters in Prot_Evol.in
A 1 chains to read
A 1 chains to read
PDB 1Z10 chain A nres=465
Contact type c, thr= 4.50, 1187 native contacts
Sequence:
KGKLPPGPTPLPFIGNYLQLNTEQMYNSLMKISERYGPVFTIHLGPRRVVVLCGHDAVREALVDQAEEFSGRGEQATF
DWVFKGYGVVFSNGERAKQLRRFSIATLRDFGVGKRGIEERIQEEAGFLIDALRGTGGANIDPTFFLSRTVSNVISSI
VFGDRFDYKDKEFLSLLRMMLGIFQFTSTSTGQLYEMFSSVMKHLPGPQQQAFQLLQGLEDFIAKKVEHNQRTLDPNS
PRDFIDSFLIRMQEEEKNPNTEFYLKNLVMTTLNLFIGGTETVSTTLRYGFLLLMKHPEVEAKVHEEIDRVIGKNRQP
```

```
KFEDRAKMPYMEAVIHEIQRFGDVIPMSLARRVKKDTKFRDFFLPKGTEVYPMLGSVLRDPSFFSNPQDFNPQHFLNE
KGQFKKSDAFVPFSIGKRNCFGEGLARMELFLFFTTVMQNFRLKSSQSPKDIDVSPKHVGFATIPRNYTMSFLPR
```

The program printed above the sequence of the PDB file!

And next it can print information about the secondary structure,

```
Secondary                                                                structure:
CCCCCCCCCCCCCCCHHHHHHHHHHHHHHHHHHHHHHHCEEEEEECCEEEEEECHHHHHHHHHHCCCCCCCEECCHHHH
HHHHHCCCCCCCHHHHHHHHHHHHHHHHHHHHCCCHHHHHHHHHHHHHHHHHHHHHCCCCCCHHHHHHHHHHHHHHH
HHHCCCCCCHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHCCCHHHHHHHHHHHHHHHHHHHHHHHCCCCCC
CCHHHHHHHHHHHHCCCCCCCCHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHCCCCCC
HHHHHHHHHHHHHHHHHHHHHHHCCCCCCCEECCCCEECCCCEECCCCEEEEHHHHHHHCCCCCCCCCCCHHHHHCC
CCCCCCCCCCCCCCCCCCCCCHHHHHHHHHHHHHHHHHHHHEEEEECCHHHHCCCEEEECCCEEECCCCEEEEEC
16                                                                         states:
CCCCCCCCCCCCCCC01234HHHHHHHHHHHHHHHHHXABEEEEYCABEEEEYC01234HHHHXCCCCCCABYCC0123
4HHHXCCCCCCC01234HHHHHHHHHHHHHXCCC01234HHHHHHHHHHHHHHHHHXCCCCCC01234HHHHHHHHHH
HHXCCCCCC01234HHHHHHHHHHHHHHHHHHHHHHHHHHHHXCCC01234HHHHHHHHHHHHHHHHHHHXCCCCCC
CC01234HHHHHHHXCCCCCCCC01234HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHXCCCCCC
01234HHHHHHHHHHHHHHHHHXCCCCCCABYCCCABYCCCABYCCCABEEE01234HXCCCCCCCCCCC01234XCC
CCCCCCCCCCCCCCCCCCCCC01234HHHHHHHHHHHHHHHBEEEYCC01234CCABEEYCCABEYCCCABEEEYC
Allocating native contact matrix, L=465 Nc=1187
Writing contact matrix in file 1Z10A_Contact_matrix.cm
Extracting DNA sequence from AA sequence
```

And again it prints the sequence of PDB file,

```
# KGKLPPGPTPLPFIGNYLQLNTEQMYNSLMKISERYGPVFTIHLGPRRVVVLCGHDAVRE
# ALVDQAEEFSGRGEQATFDWVFKGYGVVFSNGERAKQLRRFSIATLRDFGVGKRGIEERI
# QEEAGFLIDALRGTGGANIDPTFFLSRTVSNVISSIVFGDRFDYKDKEFLSLLRMMLGIF
# QFTSTSTGQLYEMFSSVMKHLPGPQQQAFQLLQGLEDFIAKKVEHNQRTLDPNSPRDFID
# SFLIRMQEEEKNPNTEFYLKNLVMTTLNLFIGGTETVSTTLRYGFLLLMKHPEVEAKVHE
# EIDRVIGKNRQPKFEDRAKMPYMEAVIHEIQRFGDVIPMSLARRVKKDTKFRDFFLPKGT
# EVYPMLGSVLRDPSFFSNPQDFNPQHFLNEKGQFKKSDAFVPFSIGKRNCFGEGLARMEL
# FLFFTTVMQNFRLKSSQSPKDIDVSPKHVGFATIPRNYTMSFLPR
```

Next the SCS model is going to run,

```
#   5   sequences   with   on   average   465   residues   each   read   in   file
CytochromeP450_Aln_Tree.nex.fas
A 115.000  E 187.000  Q 136.000  D 141.000  N 107.000  L 257.000  G 207.000  K
171.000  S 155.000  V 162.000  R 189.000  T 158.000  P 164.000  I 138.000  M
96.000  F 263.000  Y 72.000  C 12.000  W 7.000  H 53.000

Writing 1Z10A_mutation.dat
Initial Mut_par:  0.25 0.25 0.25 0.25 2 1.3 0.25
Likelihood= -419.77 Npar=7

Maximize ML by steepest ascent, starting: 0.25 0.25 0.25 0.25 1 1 0
### Free par.= 3
it= 0 lik= -434.254  Mut:  0.25 0.25 0.25 0.25 1 1 0
it= 1 lik= -420.558  Mut:  0.253 0.253 0.253 0.241 1 1 0
it= 2 lik= -408.540  Mut:  0.256 0.256 0.256 0.232 1 1 0
it= 3 lik= -398.470  Mut:  0.259 0.259 0.259 0.223 1 1 0
it= 4 lik= -390.338  Mut:  0.262 0.262 0.259 0.216 1 1 0
it= 5 lik= -383.080  Mut:  0.265 0.265 0.256 0.213 1 1 0
… Text (more iterations) …
it= 7 lik= -343.721  Mut:  0.306 0.265 0.238 0.191 2.17 1 0.05
it= 7 lik= -343.751  Mut:  0.306 0.265 0.238 0.191 2.17 1 0.05#AA:   A  E
Likelihood computed from amino acid sequence
Obtaining mutational parameters from ML fit plus amino acid sequence
#AA:   A     E     Q     D     N     L     G     K     S     V     R
       T     P     I     M     F     Y     C     W     H
#Pobs: 0.041 0.067 0.049 0.051 0.038 0.092 0.074 0.061 0.056 0.058 0.068
       0.057 0.059 0.049 0.034 0.094 0.026 0.004 0.003 0.019
```

```
#Pmut: 0.042 0.064  0.047  0.049  0.039  0.093  0.072  0.061  0.059  0.060  0.069
       0.057  0.056  0.052  0.033  0.087  0.028  0.008  0.004  0.021
#DP:   -0.0010.003  0.001  0.002  -0.001 -0.001 0.002  0.001  -0.004 -0.002 -0.001
       -0.000 0.003  -0.002 0.002  0.007  -0.002 -0.004 -0.002 -0.002

Mut_par:  0.306 0.265 0.238 0.191 2.33 1 0
Likelihood= -342.92

###
### End of computation of background distributions
###

Dividing contact energies by T= 0.50
Writing E_loc_0.txt
### Contact statistics
1081 contact matrices found in file structures.in
Selecting fragments of 465 residues with 910 < NC < 1840
(NC/L ~ 4.00 - 8.07^(-1/3) +/- 1.00)
9488 fragments used for statistics
Fraction of discarded contact matrices: 0.084
Minimum contact distance IJ_MIN= 4
<NC>= 1098 M2(NC)/<NC>^2= 0.0067 M3(NC)/<NC>^3= 0.000162
Writing Contact_statistics_465.dat
T= 0.5 sU1= 0.06 sC1= 0.06 sC0= 0.00 L=465
DeltaG/T= -123.24
Writing 1Z10A_DeltaG.dat
Temperature with smallest DG: T=0.10
### Contact statistics
1081 contact matrices found in file structures.in
Selecting fragments of 465 residues with 910 < NC < 1840
(NC/L ~ 4.00 - 8.07^(-1/3) +/- 1.00)
T= 0.50 L= 465 S_U=30.22 S_C=30.22
THREADING results:
<U^2>= 146
<EC>= -37.9           /L=  -0.0815
<(EC-<EC>)^2>/2=  89.56  /L=    0.193
<(EC-<EC>)^3>/6= -15.68  /L=  -0.0337
CONTFREQ results:
<U^2>= 146
E1= -37.9
<(EC-<EC>)^2>/2=  98.18  E22= 65 E23= 26 E24= 7
<(EC-<EC>)^3>/6=   0.00  E321= 0 E332= 0
       E342= 0 E343_2= -20 E343_3= 0    E353=0 E363= 0
E36 used: YES
Writing 1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_Threading.dat
Writing 1Z10A_summary.dat
Wild type mutants. DG_wt=-123.24 REM=2
8835 mutants computed
#Lambda log_likelihood
0.000 -2.853
0.100 -2.797
0.200 -2.749
0.300 -2.711
0.400 -2.681
0.500 -2.660
0.600 -2.646
0.700 -2.638
0.800 -2.636
0.900 -2.640
1.000 -2.648
1.100 -2.660
Optimal likelihood= -2.636 Lambda=0.783

### Mean-field computations
Mean field distribution, native state Lambda=0.000
#iteration   DeltaG   Tfreeze   factor   Kul-Leib   KL+Lambda*E_nat   convergence
(Lambda=0.00 T=0.50 SC=30.22 SU=30.22)
 0     74.05  1.56   1.00   0.00   0.00   0
```

```
 1     74.05  1.56   1.00   0.00   0.00   0
Mean field distribution, native state Lambda=0.050
#iteration   DeltaG   Tfreeze   factor   Kul-Leib   KL+Lambda*E_nat   convergence
(Lambda=0.05 T=0.50 SC=30.22 SU=30.22)
 0     74.05  1.56   1.00   0.00   -2.24  0.0031
 1     70.96  1.71   1.00   0.66   -2.93  0.00019
 2     70.81  1.71   1.00   0.73   -2.93  1.4e-05
 3     70.80  1.71   1.00   0.73   -2.93  1.1e-06
 4     70.80  1.71   1.00   0.73   -2.93  8.5e-08
Mean field distribution, native state Lambda=0.100
#iteration   DeltaG   Tfreeze   factor   Kul-Leib   KL+Lambda*E_nat   convergence
(Lambda=0.10 T=0.50 SC=30.22 SU=30.22)
 0     70.80  1.71   1.00   0.73   -6.59  0.0033
 1     67.94  1.90   1.00   2.94   -7.40  0.00043
 2     67.58  1.93   1.00   3.25   -7.42  6.6e-05
 3     67.51  1.93   1.00   3.30   -7.42  1e-05
 4     67.50  1.93   1.00   3.30   -7.42  1.7e-06
 5     67.50  1.93   1.00   3.31   -7.42  2.7e-07
Mean field distribution, native state Lambda=0.150
#iteration   DeltaG   Tfreeze   factor   Kul-Leib   KL+Lambda*E_nat   convergence
(Lambda=0.15 T=0.50 SC=30.22 SU=30.22)
 0     67.50  1.93   1.00   3.31   -12.78 0.0035
 1     64.67  2.17   1.00   7.44   -13.74 0.00073
 2     64.00  2.21   1.00   8.28   -13.78 0.00017
 3     63.81  2.22   1.00   8.49   -13.78 4.1e-05
 4     63.76  2.22   1.00   8.54   -13.78 9.9e-06
 5     63.74  2.23   1.00   8.55   -13.78 2.4e-06
 6     63.74  2.23   1.00   8.55   -13.78 5.9e-07
… Text (more iterations to optimize Lambda by ML until reach convergence) …
23     -58.98 1.48   1.00   56.24  14.01  0.00084
24     -59.03 1.50   1.00   55.65  13.38  0.00072
Using distribution with minimum convergence error 5.15e-07
DG= -59.029
Maximum likelihood complete model: -2.71 optimal Lambda=0.716

A warning could appear if Lambda does not reach the best convergence. This
situation will not affect the ASR because an optimized Lambda after a number
of steps is enough (even if it does not reach the best convergence)
```

Prot_evol prints output files (see next section, this should be fast) ..

```
Writing 1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_fitness.dat
### End of mean-field computations
Writing 1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_likelihood.dat
Meanfield model, type ALL
Writing
1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_MF_AA_profiles.txt
Writing
1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_MF_AA_profile_global.tx
t
Writing 1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_rate_mut.dat
Writing
1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_MF_exchangeability_HB_J
TT_FLUX.txt
Writing
1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_MF_rate_profile.dat
Writing
1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_MF_exchangeability_JTT_
FLUX.txt
Writing 1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_AArates.dat
Writing
1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_WT_AA_profiles.txt
Writing
1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_WT_AA_profile_global.tx
t
Writing 1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_rate_mut.dat
```

```
Writing
1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_WT_exchangeability_HB_J
TT_FLUX.txt
Writing
1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_WT_rate_profile.dat
Writing
1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_WT_exchangeability_JTT_
FLUX.txt
Writing 1Z10A_MF_LAMBDA_OPT_ALL_REM2_T0.50_SU0.065_GC0.43_CpG2_AArates.dat
```

ProtASR adapts global evolutionary parameters generated by *Prot_evol* to the *PAML* format,

```
>>> Copying matrix per site into file "MatricesFileFromProt_Evol.txt" ...

> Matrices file uploaded: GlobalWeightedMatrixFileFromProt_Evol.txt
> Reading ... Ok!

> Matrices file uploaded: MatricesFileFromProt_Evol.txt
> Reading ... Ok!
   ./MatricesFileFromProt_Evol.txt       was       detected       ...       Ok!
./GlobalWeightedMatrixFileFromProt_Evol.txt was detected ... Ok!
>>> Creating a RESULTS/Mean-field folder with the output files of Mean-field
...

>>> Adapting matrices file per site ...
- MatricesFile_FromProtEvol_to_PAML.pl -

> Adapting matrices file generated with Prot_Evol
> Matrices file uploaded: MatricesFileFromProt_Evol.txt
> Reading matrix per site ...
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

… Text …

448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465
Detected 465 matrices ...
> Adapting matrix per site ...
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

… Text …

448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465

>>> Adapting a global matrix file ...
Mean matrix will be printed to GlobalMatrixProt_Evol.txt ...

>>> Generating mean global matrix for PAML ..
> File created: GlobalSimpleMatrixFileFromProt_Evol.txt
> File uploaded: GlobalWeightedMatrixFileFromProt_Evol.txt
Matrices file uploaded: GlobalMatrixProt_Evol.dat
Reading frequencies ...
Normalizing frequencies ...
Saving frequencies ...


>>> Writing global input file for maximum likelihood computations  ...
```

Running *PAML* for the global (all sites) alignment ..

```
>>> Computing maximum likelihood for the global alignment ... (this may take
some time)
Maximum likelihood lnL: -1968.399101
```

ProtASR adapts site-specific evolutionary parameters generated by *Prot_evol* to the *PAML* format,

```
>>> Extracting optimized tree ...
Optimized tree: ((CP2A6_H: 0.000000, ((Q8SQ68: 0.038538, CP2A5_M: 0.065489):
0.021289, Q307K8: 0.024340): 0.081534): 0.000000, 1Z10_A: 0.002092);

>>> Saving the optimized tree ...
```

The optimized tree generated by *PAML* from the global alignment is applied to each site without further optimizations. This is because optimize a tree per site can lead to incorrect trees derived from a potential lack of information per site to infer a phylogeny.

*S*ite-specific (local) alignments are generated with *ProtASR* ..

```
>>> Generating local alignment files ...

- NexusToNexus_localAln.pl -
> Directory detected: /Users/Miguel/Desktop/Cytochrome_P450_Output
> Input file uploaded: CytochromeP450_Aln_Tree.nex
> Reading input file "CytochromeP450_Aln_Tree.nex" ...

NumberSites: 465, NumberSeqs: 5
 Working on site: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

… Text …

443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461
462 463 464 465
```

Local input files for *PAML* are generated with *ProtASR* ..

```
>>> Generating local input files for maximum likelihood computations ...

> Site: 1
Matrices file uploaded: MyModel_1.dat
Reading frequencies ...
Normalizing frequencies ...
Saving frequencies ...

… Text …

> Site: 465
Matrices file uploaded: MyModel_465.dat
Reading frequencies ...
Normalizing frequencies ...
Saving frequencies ...
```

PAML is executed for each site under the corresponding evolutionary parameters generated by *Prot_evol* ..

```
>>> Computing site-specific maximum likelihood ...
1(465) 2(465) 3(465) 4(465) 5(465) 6(465) 7(465) 8(465) 9(465) 10(465) 11(465)

… Text …

454(465) 455(465) 456(465) 457(465) 458(465) 459(465) 460(465) 461(465)
462(465) 463(465) 464(465) 465(465)
```

The ML per site is printed to the screen (also to an output file, see next section) ..

```
>>> Extracting likelihoods per site ...

Working file: LocalOutPAML_1.txt ... lnL: -18.053969
Working file: LocalOutPAML_2.txt ... lnL: -2.330735

… Text …

Working file: LocalOutPAML_465.txt ... lnL: -2.723852
```

```
Total likelihood: -1864.694481. Average among sites: -4.01009565806452, 95%CI:
0.251778254159285
```

Moving output files to the output folder "*RESULTS*" ..

```
>>> Moving output files to the RESULTS folder ...


>>> Printing ancestral protein sequences ...
```

Printing ancestral protein sequences to an output file that will be placed in the output folder "*RESULTS*" ..

```
Working on site 1(465) 2(465) 3(465) 4(465) 5(465) 6(465) 7(465) 8(465) 9(465)

… Text …

460(465) 461(465) 462(465) 463(465) 464(465) 465(465)

>>> Cleaning ...

>>> ProtASR done!
```

4.4. Output Files

Several output files are generated by *ProtASR*. These files are saved in a folder "*RESULTS*" that will be placed at the working directory.

4.4.1. Maximum likelihood and ancestral protein sequences

In the output folder "*RESULTS*":

- The file "*InferredAncestralProteins.txt*" shows the ancestral protein sequences for all (tip and internal) nodes under both *marginal* and *joint* ML.
- The file "*LocalResultsLikelihood.txt*" shows the likelihood (lnL) estimates per site and at the global level (several statistics are also included).
- The input nexus file can also be found in the folder "*RESULTS*" in two files, an alignment of the protein sequences in *fasta* format and the tree in *Newick* format.

4.4.2. Inferences at the global (entire sequence) level

The folder "*Global_ASR_ML*" placed in the folder "*RESULTS*" includes output files with detailed information about the ancestral reconstruction of the entire sequences assuming that all sites evolve under a same evolutionary process. This should be therefore carefully considered, see sections 3.4.1 and 3.4.2 for results accounting for heterogeneous evolutionary processes across sites.

4.4.3. Inferences at the local (site-specific) level

The folder "*Local_ASR_ML*" placed in the folder "*RESULTS*" includes output files with detailed information about the ancestral reconstruction at each particular site. The summary of all sites is placed in the files included in the section 3.4.1. Here we recommend to see the files "*rst_#*". Other files might not provide relevant information.

4.4.4. Other output files from structurally constrained substitution models

The folder "*Meanfield*" placed in the folder "*RESULTS*" includes output files (for MF and for WT) with detailed information generated by the structural constrained substitution model:

- The file "*PDB_DeltaG.dat*" provides DeltaG values for different temperatures.
- The file "**_profile_global.txt*" shows a global overview of the abundance of the 20 amino acid states at the native and mutated sequences.
- The file "**_profiles.txt*" shows the site-specific amino acid frequencies estimated with the *MF* model. The site-specific frequencies are normalized in the file "*MatricesFileFromProt_Evol.txt*". These frequencies are key to perform the ASR. The estimated site-specific entropy is also shown.
- The file "**_allsites.txt*" includes the user-specified empirical matrix that is applied for sampling in SCS models. The frequencies are normalized in the files "*GlobalMatrixProt_Evol.dat*", "*GlobalSimpleMatrixFileFromProt_Evol.txt*" and "*GlobalWeightedMatrixFileFromProt_Evol.txt*", which are key to perform the ASR.
- The file *"*_FLUX.txt*" indicates the global and site-specified matrix and amino acid frequencies generated by the model under the HB approach. This material is key to perform the ASR.
- The file "**_likelihood.dat*" includes information (for each tested Lambda it shows DeltaG all, DeltaG native, likelihood all, likelihood native, freezing temperatures, etc) about the optimization of the Lambda parameter by ML.
- The files "**_profiles.dat*" include information for each site: number of contacts, rate of change, entropy, hydrophobicity in the native state and hydrophobicity induced by the model.
- The file "**_summary.dat*" presents a variety of parameters such as protein length, estimated DNA frequencies, Lambda, likelihoods, freezing temperatures, DeltaGs, PDB code, etc.
- The file "**_Threading.dat*" presents parameters about the threading of the protein structure (i.e., energies).
- The file "*_AArates.dat" presents rates of change and other parameters like hydrophobicity for every amino acid state.
- The file "*_fitness.dat" presents free energy and likelihood of the representative protein structure.
- The file "*rate_mut.dat" presents rate of change among pairs of amino acid states in both directions.
- The file "Local_interactions.dat" presents the observed local interactions in secondary structure.
- The file "*Contact_statistics_#.dat*" presents the frequency of contacts for each site.
- The file "*Prot_Evol.in*" is the input file generated by *ProtASR* that is applied to *Prot_evol* (MF model).

4.5. Re-analysing data

The same data can be analysed in the working directory (i.e., without any modification of the Settings file). The user has only to consider that after running *ProtASR*, the

"*RESULTS*" folder will be overwritten. So we recommend rename the "*RESULTS*" folder after every analysis.

4.6. Practical examples

The package includes several different examples in the folder "examples". Each example is a protein family downloaded from the *Pfam* database (http://pfam.xfam.org/), seed alignment of protein sequences and the corresponding tree, and includes a PDB file that belongs to a protein of such a protein family.
For each example two folders are present:

- Input: Only the input data files (nexus file, DB file and Settings file).
- Output: Input data files and the corresponding "*RESULTS*" folder with the output files.

To run any of these examples just move into the output folder, remove the "*RESULTS*" folder if desired, compile *ProtASR* (move all the material of the "*src*" directory to this directory, compile, see previous sections) and run it (see previous sections).

4.7. Message errors

Errors generated from incorrect settings are usually shown on the screen and the program will suggest abort the execution by typing CTRL+C. **For beginners, we recommend copying the provided input files "Settings.txt" in the "examples" folder and editing them with the desired settings.**

Input nexus alignment: Check that the PDB sequence (the PDB sequence can be provided with *Prot_Evol* and *ProtASR*, information printed on the screen at the beginning of a run) presents the same length that the alignment length (see section "Target alignment and rooted tree"). In the nexus file, taxa names should be separated from the corresponding sequence by any space/s. Indeed, note that every taxa included in the alignment must be also in the tree. Check that the tree is rooted.

SCS models: If MF models do not converge after 10 automatic attempts (this is unexpected but in case it appears will be shown on the console), stop the execution (CTRL+C several times) and run again *ProtASR*. Large datasets are expected to generate more convergence problems and so reduce the sample size can be a good idea, if possible. If problems persists, contact us.

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

5. Models and Methods

ProtASR is an analytical framework to infer ancestral protein sequences while accounting for structural constrains. The program includes two main stages, the generation of evolutionary parameter values with MF or WT SCS models and the ASR with an adapted version of the *PAML* package. Details about MF and WT models are provided in the article, and details about previous versions of these programs can be found in [1, 12].
ProtASR should work in seconds/minutes depending on the length of the PDB protein

and the alignment size (number of sequences and sequence length).

6. ProtASR performance

Previous versions of MF and *ProtASR* have presented a better fit (in terms of ML) to real data than empirical substitution models and other structurally constrained substitution models [1, 12]. In addition, ancestral proteins generated with *ProtASR* are more stable (in terms of DOPE energy [39]) than ancestral proteins generated by empirical substitution models.

The new version of *ProtASR* (*ProtASR2*) outperforms the previous version by implementing more realistic SCS models (see paper on *ProtASR2*).

6.1. Assumptions and Limitations

ProtASR assumes that the structural constrains are constant with time, this is, the protein structure remains constant with time. Indeed, it assumes that all proteins of the dataset can fit with the user-specified PDB structure. Therefore, *ProtASR* is oriented to the analysis of protein families and, in general, datasets of proteins with similar structure. *ProtASR2* tried to address this final issue by implementing SCS models able to outperform empirical models also at low levels of sequence identity of the input multiple sequence alignment.

Future versions of *ProtASR* will try to incorporate branch-specific models where different lineages could evolve under different protein structures. We also expect to improve the current set of available SCS models.

7. Acknowledgments

MA wants to thank the Spanish Government through the grant “Ramón y Cajal” RYC-2015-18241.

8. References

1. Arenas M, Weber CC, Liberles DA, Bastolla U: **ProtASR: An Evolutionary Framework for Ancestral Protein Reconstruction with Selection on Folding Stability**. *Syst Biol* 2017, **66**(6):1054-1064.
2. Arenas M, Dos Santos HG, Posada D, Bastolla U: **Protein evolution along phylogenetic histories under structurally constrained substitution models**. *Bioinformatics* 2013, **29**(23):3020-3028.
3. Bastolla U, Demetrius L: **Stability constraints and protein evolution: the role of chain length, composition and disulfide bonds**. *Protein Eng Des Sel* 2005, **18**(9):405-415.
4. Grahnen JA, Liberles DA: **CASS: Protein sequence simulation with explicit genotype-phenotype mapping**. *Trends in Evolutionary Biology* 2012, **4**:1.
5. Grahnen JA, Nandakumar P, Kubelka J, Liberles DA: **Biophysical and structural considerations for protein sequence evolution**. *BMC Evol Biol* 2011, **11**:361.

6. Wilke CO: **Bringing molecules back into molecular evolution**. *PLoS Comput Biol* 2012, **8**(6):e1002572.
7. Goldstein RA: **The evolution and evolutionary consequences of protein marginal stability**. *Proteins* 2011, **In press**.
8. Goldstein RA: **The evolution and evolutionary consequences of marginal thermostability in proteins**. *Proteins* 2011, **79**(5):1396-1407.
9. Parisi G, Echave J: **Structural constraints and emergence of sequence patterns in protein evolution**. *Mol Biol Evol* 2001, **18**(5):750-756.
10. Parisi G, Echave J: **The structurally constrained protein evolution model accounts for sequence patterns of the LbetaH superfamily**. *BMC Evol Biol* 2004, **4**:41.
11. Parisi G, Echave J: **Generality of the structurally constrained protein evolution model: assessment on representatives of the four main fold classes**. *Gene* 2005, **345**(1):45-53.
12. Arenas M, Sanchez-Cobos A, Bastolla U: **Maximum likelihood phylogenetic inference with selection on protein folding stability**. *Mol Biol Evol* 2015, **32**(8):2195-2207.
13. Gao F, Bhattacharya T, Gaschen B, Taylor J, Moore JP, Novitsky V, Yusim K, Lang D, Foley B, Beddows S *et al*: **Consensus and ancestral state HIV vaccines**. *Science* 2003, **299**(5612):1515-1518.
14. Arenas M, Posada D: **Computational Design of Centralized HIV-1 Genes**. *Curr HIV Res* 2010, **8**(8):613-621.
15. Kothe DL, Li Y, Decker JM, Bibollet-Ruche F, Zammit KP, Salazar MG, Chen Y, Weng Z, Weaver EA, Gao F *et al*: **Ancestral and consensus envelope immunogens for HIV-1 subtype C**. *Virology* 2006, **352**(2):438-449.
16. Doria-Rose NA, Learn GH, Rodrigo AG, Nickle DC, Li F, Mahalanabis M, Hensel MT, McLaughlin S, Edmonson PF, Montefiori D *et al*: **Human immunodeficiency virus type 1 subtype B ancestral envelope protein is functional and elicits neutralizing antibodies in rabbits similar to those elicited by a circulating subtype B envelope**. *J Virol* 2005, **79**(17):11214-11224.
17. Perez-Jimenez R, Ingles-Prieto A, Zhao ZM, Sanchez-Romero I, Alegre-Cebollada J, Kosuri P, Garcia-Manyes S, Kappock TJ, Tanokura M, Holmgren A *et al*: **Single-molecule paleoenzymology probes the chemistry of resurrected enzymes**. *Nat Struct Mol Biol* 2011, **18**(5):592-596.
18. Thornton JW: **Resurrecting ancient genes: experimental analysis of extinct molecules**. *Nat Rev Genet* 2004, **5**(5):366-375.
19. Thornton JW, Need E, Crews D: **Resurrecting the ancestral steroid receptor: ancient origin of estrogen signaling**. *Science* 2003, **301**(5640):1714-1717.
20. Thomson JM, Gaucher EA, Burgan MF, De Kee DW, Li T, Aris JP, Benner SA: **Resurrecting ancestral alcohol dehydrogenases from yeast**. *Nat Genet* 2005, **37**(6):630-635.
21. Liberles DA: **Ancestral Sequence Reconstruction**: Oxford University Press; 2007.
22. Halpern AL, Bruno WJ: **Evolutionary distances for protein-coding sequences: modeling site-specific residue frequencies**. *Mol Biol Evol* 1998, **15**(7):910-917.
23. Whelan S, Goldman N: **A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach**. *Mol Biol Evol* 2001, **18**(5):691-699.

24. Jones DT, Taylor WR, Thornton JM: **The rapid generation of mutation data matrices from protein sequences**. *Comput Appl Biosci* 1992, **8**(3):275-282.
25. Yang Z: **PAML 4: phylogenetic analysis by maximum likelihood**. *Mol Biol Evol* 2007, **24**(8):1586-1591.
26. Henikoff S, Henikoff JG: **Amino acid substitution matrices from protein blocks**. *Proc Natl Acad Sci U S A* 1992, **89**(22):10915-10919.
27. Adachi J, Waddell PJ, Martin W, Hasegawa M: **Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast DNA**. *J Mol Evol* 2000, **50**(4):348-358.
28. Dayhoff MO, Schwartz RM, Orcutt BC: **A model of evolutionary change in proteins**. In: *Atlas of protein sequence and structure.* Edited by Dayhoff MO, vol. 5, Suppl. 3. Washington D. C.; 1978: 345-352.
29. Kosiol C, Goldman N: **Different versions of the Dayhoff rate matrix**. *Mol Biol Evol* 2005, **22**(2):193-199.
30. Nickle DC, Heath L, Jensen MA, Gilbert PB, Mullins JI, Kosakovsky Pond SL: **HIV-specific probabilistic models of protein evolution**. *PLoS One* 2007, **2**(6):e503.
31. Le SQ, Gascuel O: **An improved general amino acid replacement matrix**. *Mol Biol Evol* 2008, **25**(7):1307-1320.
32. Abascal F, Posada D, Zardoya R: **MtArt: A New Model of Amino Acid Replacement for Arthropoda**. *Mol Biol Evol* 2007, **24**(1):1-5.
33. Yang Z, Nielsen R, Masami H: **Models of amino acid substitution and applications to mitochondrial protein evolution**. *Mol Biol Evol* 1998, **15**(12):1600-1611.
34. Adachi J, Hasegawa M: **MOLPHY version 2.3: programs for molecular phylogenetics based in maximum likelihood**. *Comput Sci Monogr* 1996, **28**:1-150.
35. Dimmic MW, Rest JS, Mindell DP, Goldstein RA: **rtREV: an amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny**. *J Mol Evol* 2002, **55**(1):65-73.
36. Muller T, Vingron M: **Modeling amino acid replacement**. *J Comput Biol* 2000, **7**(6):761-776.
37. Derrida B: **Random Energy Model: An exactly solvable model of disordered systems**. *Phys Rev B* 1981, **24**:2613-2626.
38. Abascal F, Zardoya R, Posada D: **ProtTest: selection of best-fit models of protein evolution**. *Bioinformatics* 2005, **21**(9):2104-2105.
39. Shen MY, Sali A: **Statistical potential for assessment and prediction of protein structures**. *Protein Sci* 2006, **15**(11):2507-2524.