Protein Structure Modeling via SWISS-MODEL, MODELLER, and AlphaFold

Modeller Tutorial

https://salilab.org/modeller/tutorial/ 20210922

MODELLER is used for homology or comparative modeling of protein three-dimensional structures. The user provides an alignment of a sequence to be modeled with known related structures and MODELLER automatically calculates a model containing all non-hydrogen atoms.
This web site presents a tutorial for the use of MODELLER 10.0 or newer (for older versions of MODELLER, use the old MODELLER 9v4 tutorial). There are 5 modeling examples that the user can follow:

1. **Basic Modeling**. Model a sequence with high identity to a template. This exercise introduces the use of MODELLER in a simple case where the template selection and target-template alignments are not a problem.
2. **Advanced Modeling**. Model a sequence based on multiple templates and bound to a ligand. This exercise introduces the use of multiple templates, ligands and loop refinement in the process of model building with MODELLER.
3. **Iterative Modeling**. Increase the accuracy of the modeling exercise by iterating the 4 step process. This exercise introduces the concept of MOULDING to improve the accuracy of comparative models.
4. **Difficult Modeling**. Model a sequence based on a low identity to a template. This exercise uses resources external to MODELLER in order to select a template for a difficult case of protein structure prediction.
5. **Modeling with cryo-EM**. Model a sequence using both template and cryo-EM data. This exercise assesses the quality of generated models and loops by rigid fitting into cryo-EM maps, and improves them with flexible EM fitting.

Basic example: Modeling lactate dehydrogenase from *Trichomonas vaginalis* based on a single template.

All input and output files for this example are available to download, in either .zip format (for Windows) or .tar.gz format (for Unix/Linux).

A novel gene for lactate dehydrogenase was identified from the genomic sequence of.Trichomonas vaginalis.(TvLDH). The corresponding protein had a higher similarity to the malate dehydrogenase of the same species (TvMDH) than to any other LDH. We hypothesized that TvLDH arose from TvMDH by convergent evolution relatively recently. Comparative models were constructed for TvLDH and TvMDH to study the sequences in the structural context and to suggest site-directed mutagenesis experiments for elucidating specificity changes in this apparent case of convergent evolution of enzymatic specificity. The native and mutated enzymes were expressed and their activities were compared.

The individual modeling steps of this example are explained below. Note that we go through every step in this tutorial to build a model knowing only the amino acid sequence. In practice you may already know the related structures, and may even have an alignment from another program, so you can skip one or more steps. Alternatively, for very simple applications you may be able to use the ModWeb web server rather than Modeller itself.

1. Searching for structures related to TvLDH

First, it is necessary to put the target TvLDH sequence into the PIR format readable by MODELLER (file "TvLDH.ali").

```
>P1;TvLDH
sequence:TvLDH:::::::0.00: 0.00
MSEAAHVLITGAAGQIGYILSHWIASGELYGDRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPKA
AFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPEN
FSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKKI
GHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHVV
EGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG*
```

File: TvLDH.ali

The first line contains the sequence code, in the format ">P1;code". The second line with ten fields separated by colons generally contains information about the structure file, if applicable. Only two of these fields are used for sequences, "sequence" (indicating that the file contains a sequence without known structure) and "TvLDH" (the model file name). The rest of the file contains the sequence of TvLDH, with "*" marking its end. The standard one-letter amino acid codes are used. (Note that they must be upper case; some lower case letters are used for non-standard residues. See the file modlib/restyp.lib in the Modeller distribution for more information.)

A search for potentially related sequences of known structure can be performed by the Profile.build() command of MODELLER. The following script, taken line by line, does the following (see file "build_profile.py"):

1. Initializes the 'environment' for this modeling run, by creating a new 'Environ' object. Almost all MODELLER scripts require this step, as the new object (which we call here 'env', but you can call it anything you like) is needed to build most other useful objects.
2. Creates a new 'SequenceDB' object, calling it 'sdb'. 'SequenceDB' objects are used to contain large databases of protein sequences.
3. Reads a text format file containing non-redundant PDB sequences at 95% sequence identity into the.sdb.database. The sequences can be found in the file "pdb_95.pir" (which can be downloaded using the link at the top of this page). Like the previously-created alignment, this file is in PIR format. Sequences which have fewer than 30 or more than 4000 residues are discarded, and non-standard residues are removed.
4. Writes a binary machine-specific file containing all sequences read in the previous step.
5. Reads the binary format file back in. Note that if you plan to use the same database several times, you should use the previous two steps only the first time, to produce the binary database. On subsequent runs, you can omit those two steps and use the binary file directly, since reading the binary file is a lot faster than reading the PIR file.
6. Creates a new 'Alignment' object, calling it 'aln', reads our query sequence "TvLDH" from the file "TvLDH.ali", and converts it to a profile 'prf'. Profiles contain similar information to alignments, but are more compact and better for sequence database searching.
7. Searches the sequence database 'sdb' for our query profile 'prf'. Matches from the sequence database are added to the profile.
8. Writes a profile of the query sequence and its homologs (see file "build_profile.prf"). The equivalent information is also written out in standard alignment format.

```
from modeller import *

log.verbose()
env = Environ()

#-- Prepare the input files

#-- Read in the sequence database
sdb = SequenceDB(env)
sdb.read(seq_database_file='pdb_95.pir', seq_database_format='PIR',
        chains_list='ALL', minmax_db_seq_len=(30, 4000), clean_sequences=True)

#-- Write the sequence database in binary form
sdb.write(seq_database_file='pdb_95.bin', seq_database_format='BINARY',
         chains_list='ALL')

#-- Now, read in the binary database
sdb.read(seq_database_file='pdb_95.bin', seq_database_format='BINARY',
        chains_list='ALL')

#-- Read in the target sequence/alignment
aln = Alignment(env)
aln.append(file='TvLDH.ali', alignment_format='PIR', align_codes='ALL')

#-- Convert the input sequence/alignment into
#   profile format
prf = aln.to_profile()

#-- Scan sequence database to pick up homologous sequences
prf.build(sdb, matrix_offset=-450, rr_file='${LIB}/blosum62.sim.mat',
         gap_penalties_1d=(-500, -50), n_prof_iterations=1,
         check_profile=False, max_aln_evalue=0.01)

#-- Write out the profile in text format
prf.write(file='build_profile.prf', profile_format='TEXT')

#-- Convert the profile back to alignment format
aln = prf.to_alignment()

#-- Write out the alignment file
aln.write(file='build_profile.ali', alignment_format='PIR')
```

File: build_profile.py

This is a regular Python script, and so can be run with a command similar to the following at your command line:

```
python3 build_profile.py > build_profile.log
```

Note that on some systems the Python interpreter is called python2 or python rather than python3. The full path to the Python interpreter may also be necessary, such as. /usr/bin/python on a Linux or Mac machine or C:\python27\python.exe on a Windows system. If Python is not installed on your machine, Modeller also includes a basic Python 2.3 interpeter as mod<version>. For example, to run this script using Modeller 10.0's own interpreter, use mod10.0 build_profile.py. Note that mod10.0 automatically creates a build_profile.log logfile.
(You can get a command line using xterm or GNOME Terminal in Linux, Terminal in Mac OS X, or the 'Modeller' link from your Start Menu in Windows. For more information on running Modeller, see the release notes. For more information on using Python, see the Python web site. Note that you can use other Python modules within your Modeller scripts, if Python is correctly installed on your system.)

The Profile.build() command has many options. In this example rr_file is set to use the BLOSUM62 similarity matrix (file "blosum62.sim.mat" provided in the MODELLER distribution). Accordingly, the parameters matrix_offset and gap_penalties_1d are set to the appropriate values for the BLOSUM62 matrix. For this example, we will run only one search iteration by setting the parameter n_prof_iterations equal to 1. Thus, there is no need for checking the profile for deviation (check_profile set to False). Finally, the parameter max_aln_evalue is set to 0.01, indicating that only sequences with e-values smaller than or equal to 0.01 will be included in the final profile.

2. Selecting a template

The output of the "build_profile.py" script is written to the "build_profile.log" file. MODELLER always produces a log file. Errors and warnings in log files can be found by searching for the "_E>" and "_W>" strings, respectively. MODELLER also writes the profile in text format to the "build_profile.prf" file. An extract (omitting the aligned sequences) of the output file can be seen next. The first 6 commented lines indicate the input parameters used in MODELLER to build the profile. Subsequent lines correspond to the detected similarities by.Profile.build().

```
# Number of sequences:     30
# Length of profile  :    335
# N_PROF_ITERATIONS  :      1
# GAP_PENALTIES_1D   :   -900.0   -50.0
# MATRIX_OFFSET      :    0.0
# RR_FILE           : ${MODINSTALL8v1}/modlib//as1.sim.mat
    1 TvLDH S     0   335     1   335     0     0     0    0.    0.0
    2 1a5z  X     1   312    75   242    63   229   164   28.   0.83E-08
    3 1b8pA X     1   327     7   331     6   325   316   42.    0.0
    4 1bdmA X     1   318     1   325     1   310   309   45.    0.0
    5 1t2dA X     1   315     5   256     4   250   238   25.   0.66E-04
    6 1civA X     1   374     6   334    33   358   325   35.    0.0
    7 2cmd  X     1   312     7   320     3   303   289   27.   0.16E-05
    8 1o6zA X     1   303     7   320     3   287   278   26.   0.27E-05
    9 1ur5A X     1   299    13   191     9   171   158   31.   0.25E-02
   10 1guzA X     1   305    13   301     8   280   265   25.   0.28E-08
   11 1gv0A X     1   301    13   323     8   289   274   26.   0.28E-04
   12 1hyeA X     1   307     7   191     3   183   173   29.   0.14E-07
```

```
   13 1i0zA X     1   332    85   300    94   304   207   25.   0.66E-05
   14 1i10A X     1   331    85   295    93   298   196   26.   0.86E-05
   15 1ldnA X     1   316    78   298    73   301   214   26.   0.19E-03
   16 6ldh  X     1   329    47   301    56   302   244   23.   0.17E-02
   17 2ldx  X     1   331    66   306    67   306   227   26.   0.25E-04
   18 5ldh  X     1   333    85   300    94   304   207   26.   0.30E-05
   19 9ldtA X     1   331    85   301    93   304   207   26.   0.10E-05
   20 1llc  X     1   321    64   239    53   234   164   26.   0.20E-03
   21 1lldA X     1   313    13   242     9   233   216   31.   0.31E-07
   22 5mdhA X     1   333     2   332     1   331   328   44.    0.0
   23 7mdhA X     1   351     6   334    14   339   325   34.    0.0
   24 1mldA X     1   313     5   198     1   189   183   26.   0.13E-05
   25 1oc4A X     1   315     5   191     4   186   174   28.   0.18E-04
   26 1ojuA X     1   294    78   320    68   285   218   28.   0.43E-05
   27 1pzgA X     1   327    74   191    71   190   114   30.   0.16E-06
   28 1smkA X     1   313     7   202     4   198   188   34.    0.0
   29 1sovA X     1   316    81   256    76   248   160   27.   0.93E-03
   30 1y6jA X     1   289    77   191    58   167   109   33.   0.32E-05
```

File: build_profile.prf

The most important columns in the **Profile.build()** output are the second, tenth, eleventh and twelfth columns. The second column reports the code of the PDB sequence that was compared with the target sequence. The PDB code in each line is the representative of a group of PDB sequences that share 95% or more sequence identity to each other and have less than 30 residues or 30% sequence length difference. The eleventh column reports the percentage sequence identities between TvLDH and a PDB sequence normalized by the lengths of the alignment (indicated in the tenth column). In general, a sequence identity value above approximately 25% indicates a potential template unless the alignment is short (i.e., less than 100 residues). A better measure of the significance of the alignment is given in the twelfth column by the e-value of the alignment. In this example, six PDB sequences show very significant similarities to the query sequence with e-values equal to 0. As expected, all the hits correspond to malate dehydrogenases (1bdm:A, 5mdh:A, 1b8p:A, 1civ:A, 7mdh:A, and 1smk:A). To select the most appropriate template for our query sequence over the six similar structures, we will use the **Alignment.compare_structures()** command to assess the structural and sequence similarity between the possible templates (file "compare.py").

```
from modeller import *
env = Environ()
aln = Alignment(env)
for (pdb, chain) in (('1b8p', 'A'), ('1bdm', 'A'), ('1civ', 'A'),
                     ('5mdh', 'A'), ('7mdh', 'A'), ('1smk', 'A')):
    m = Model(env, file=pdb, model_segment=('FIRST:'+chain, 'LAST:'+chain))
    aln.append_model(m, atom_files=pdb, align_codes=pdb+chain)
aln.malign()
aln.malign3d()
aln.compare_structures()
aln.id_table(matrix_file='family.mat')
env.dendrogram(matrix_file='family.mat', cluster_cut=-1.0)
```