CHEM 436 / CHEM 630: Molecular Modelling of Proteins
TUTORIAL #2a: Homology Modelling 1

INTRODUCTION

The goal of this tutorial is to learn how to perform comparative protein modeling using the `automodel` command of the program MODELLER.

To carry out this tutorial, you will need a working installation of MODELLER (available for free at http://salilab.org/modeller/download_installation.html) and a text editor such as `gedit`, `emacs` or `vi`. All software are installed on the Linux Virtual Machine you will be using.

MODELLER is a "command line interface" program, which means that it interacts with the user only through text files. The user writes specific instructions into a text file and "feeds" them to the program, which then writes the results into another text file:

Instructions ("`model.py`") → MODELLER ("`mod9.19`") → Results ("`model.log`")

These instructions are written in Python, a computer language that has very strict syntactic rules, and for which even whitespace matters. The "`.py`" files themselves are called "Python scripts". So be extremely careful when you edit a Python script.

In Steps 1, 2, 3, and 4 of the Tutorial, you will go through the procedure using the example scripts provided by the developers of MODELLER. You will then use the same techniques in Step 5 to produce a homology model of your own sequence and evaluate it using the DOPE score.

REQUIRED PRE-LAB READING

The first two sections of the MODELLER manual:
("Introduction" and "Automated comparative modeling with `automodel`")

http://salilab.org/modeller/manual

The description of the PIR format:

https://salilab.org/modeller/manual/node496.html

PRE-LAB REPORT

- ✦ Perform a BLAST alignment of your query sequence with the template sequence you have chosen from Tutorial #1 (see example next page).
- ✦ Prepare a "PIR" text file describing the alignment of your query sequence with the template sequence (see MODELLER manual and example next page).
- ✦ This PIR file should be plain text. Make sure you prepare it using either a text editor (example: "Notepad" on Windows or "TextEdit" on Mac OS) or a word processor that you will use as a text editor—by saving the file as "plain text"
- ✦ Your pre-lab "report" should consist of two files, to be emailed to the instructor:
  - The BLAST alignment report (see example below), either in PDF format or in plain text format
  - The PIR file itself
- ✦ Email the PIR file to yourself as well, so that you can retrieve it from a web browser. (You will need it for STEP 5 of the tutorial.)

Example

Using BLOSUM62 and a gap cost of 11 (the default value), the query sequence

`AYVINDSCIACGACKPECPVNIIQGSIYAIDADSCIDCGSCASVCPVGAPNPED`

aligns with template sequence "`pdb|5fd1`" as follows:

```
>pdb|5FD1|  Chain  , Crystal Structures Of Oxidized And Reduced Azotobacter
Vinelandii Ferredoxin At Ph 8 And Ph 6
Length=106

 Score = 33.1 bits (74),  Expect = 5e-07, Method: Compositional matrix adjust.
 Identities = 23/58 (39%), Positives = 29/58 (50%), Gaps = 4/58 (6%)

Query  1   AYVINDSCIACGA--CKPECPVN-IIQG-SIYAIDADSCIDCGSCASVCPVGAPNPED  54
           A+V+ D+CI C    C   CPV+   +G +   I  D CIDC  C   CP  A   ED
Sbjct  1   AFVVTDNCIKCKYTDCVEVCPVDCFYEGPNFLVIHPDECIDCALCEPECPAQAIFSED  58
```

The PIR file corresponding to this alignment would be:

```
>P1;5fd1
structureX:5fd1:1    :A:58   :A::::
AFVVTDNCIKCKYTDCVEVCPVDCFYEGPNFLVIHPDECIDCALCEPECPAQAIFSED*
>P1;query
sequence:query:1    : :54   : ::::
AYVINDSCIACGA--CKPECPVN-IIQG-SIYAIDADSCIDCGSCASVCPVGAPNPED*
```

READING

On homology modeling:
Chapter 7 of Tramontano ("Homology Modeling"): All sections.
Chapter 13 of Zvelebil & Baum ("Modeling Protein Structure"): Sections 13.3 to 13.6.

ABOUT THE COMPUTER WORKSTATIONS

To start the Linux Virtual Machine

Copy the "virtual machine" file from the USB key to your local personal directory (the folder icon on the desktop, called by your name).

Start "Oracle VM VirtualBox" from the Windows environment. Create the virtual machine from the file in your local personal directory using "File > Import Appliance…", then start the "MMVM" virtual machine by double-clicking on its icon.

If the Linux window manager does not start automatically and you are presented with a text prompt, do the following:

```
localhost login: mmvm
[mmvm@localhost ~]$ systemctl start lightdm
```

To avoid inadvertently switching between the Windows and the Linux environments, it is recommended to use the full-screen mode, by selecting "View > Full-screen Mode" from the menu or by pressing "Right Ctrl"+F. ("Right Ctrl" is the "Host" key on Windows.)

To store files on the Linux VM

Create your own work folder on the Linux desktop by right-clicking on the desktop background. Give this folder a name that can clearly be identified to the course, containing no special characters and no spaces (example: "`chem436`" or "`chem630`"). Put all your files in that folder. Do not expect this folder to be safe from one week to the other and back up everything at the end of every tutorial.

To back up your files on the Linux VM

Right-click on the icon of the folder you have created, and use "Create Archive…" to create a ".tar.gz" file on the desktop (example: "`chem436.tar.gz`"), which you can then email to yourself or upload to DropBox or Google Drive.

Alternatively, you can create the archive directly from the terminal (accessible from the "Applications > Terminal Emulator" menu, at the top left of the screen). Once you have opened a terminal window, go to the Desktop directory by typing "`cd ~/Desktop`" and create the archive by typing the command "`tar -z -cvf chem436.tar.gz chem436/`".

To shut down the Linux VM

You can do a proper shutdown from the Linux VM using the "Applications > Log Out > Shut Down" menu. Alternatively, you can exit the full-screen mode by pressing "Right Ctrl"+F, then select "Machine > Close > ACPI Shutdown" from the Oracle VM VirtualBox menu. Any change you have made to the Linux system will be saved and available the next time you start the virtual machine.

PROCEDURE

HELP

Some useful Linux commands for STEP 1, STEP 2, and STEP 3:

From the terminal:

Type "`cd`" to get back to the home directory.

Type "`pwd`" to see what directory you are currently in.

Type "`ls`" ("list") to see the list of files/directories in the current directory.

Use "mkdir" to create a new directory and "cd" to move into it. Example:

```
$ cd Desktop
$ mkdir chem436
$ cd chem436
$ mkdir tutorial2
$ cd tutorial2
```

Use "`wget`" to download a file from the internet. Example:

```
$ wget http://salilab.org/modeller/examples/automodel/model-default.py
```

Use "`more`" to view the content of a text file. Example:

```
$ more model-default.log
```

Call PyMOL directly from the command line. Example:

```
$ pymol 5fd1.pdb 1fdx.B99990001.pdb
```

Use "`gedit`" to edit a text file. Example:

```
$ gedit model-default.py
```

Use "`mv`" to rename a file. Example:

```
$ mv old_name.log new_name.log
```

Use "`cp`" to create a copy of a file. Example:

```
$ cp model-default.log model-default-ali.log
```

STEP 1: Run the "`model-default.py`" example

Create a directory "`tutorial2`" in your work folder (example: "`chem436/tutorial2`"), and download the Python script "`model-default.py`" (http://salilab.org/modeller/examples/automodel/model-default.py) and the file "`alignment.ali`" (http://salilab.org/modeller/examples/automodel/alignment.ali). The two files should be put in that directory.

Download the PDB file of the template sequence ("`5fd1.pdb`") and put it in the same directory as the other two files.

From the MODELLER command prompt, type:

```
mod9.19 model-default.py
```

MODELLER will read the Python script and write the following files (among other):

`model-default.log`	"log" file (contains the summary of the results)
`1fdx.B99990001.pdb`	3D model (can be viewed in PyMOL)
`1fdx.V99990001`	Violation profiles for the model

Inspect the log file ("`model-default.log`") and signal any error or warning message to the instructor (search for "`_E>`" and "`_W>`").

STEP 2: Visualize the template and the model produced by MODELLER

Open the two PDB files ("`5fd1.pdb`" and "`1fdx.B99990001.pdb`") in PyMOL and render them as cartoons. Display the amino acid sequences ("Display > Sequence On" menu) and color **in blue** the amino acids of the template that participate in the alignment, and **in red** the amino acids of the model that participate in the alignment. Leave all other amino acids (including those in gaps) their original colors.

Align the two structures using the backbone atoms of the matching residues:

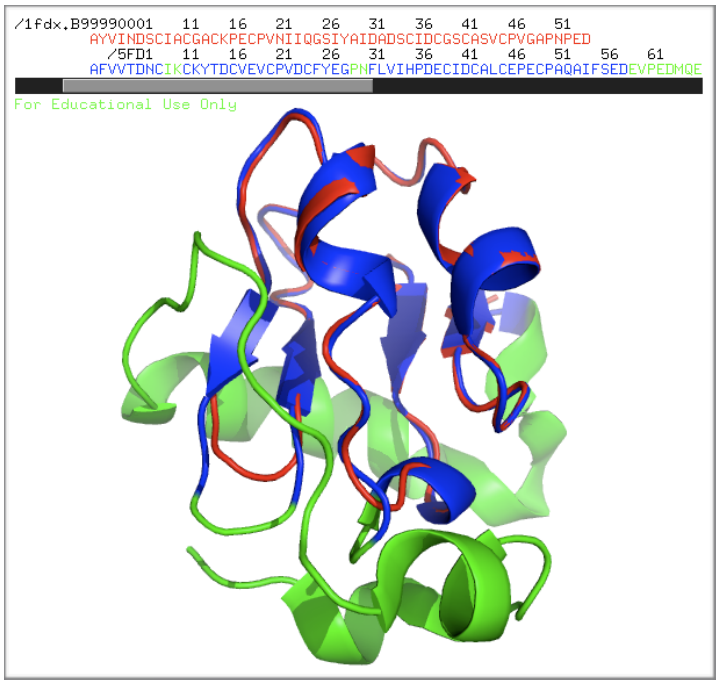
```
PyMOL> align 1fdx.B99990001 & resi 1-54 and name n+ca+c+o, 5fd1 &
       resi 1-8+11-28+31-58 and name n+ca+c+o
```

Note that the model ("`1fdx`") is selected from residues 1 to 54 and the template ("`5fd1`") is selected from residues 1 to 58 but excluding residues 9, 10, 29, and 30 (because they are gaps).

The PyMOL log should contain something like this:

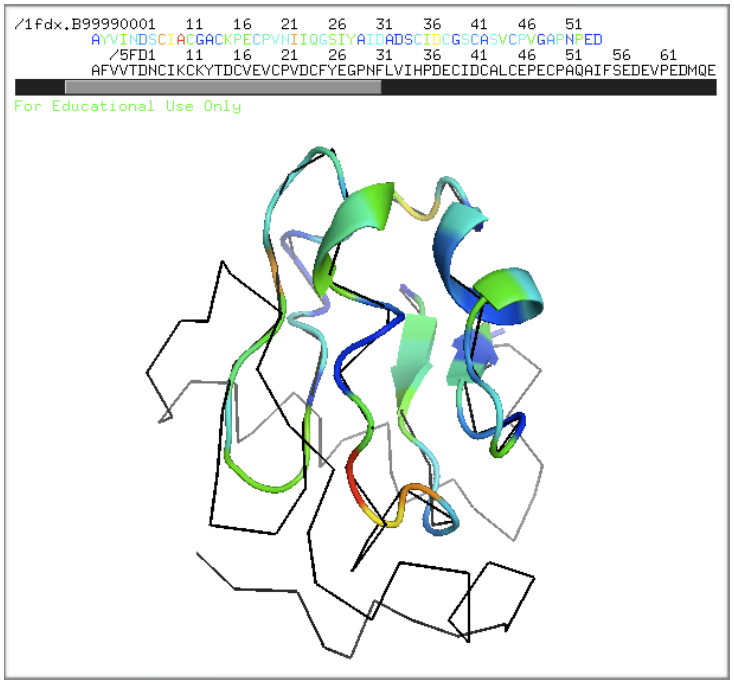
```
Match: read scoring matrix.
Match: assigning 54 x 54 pairwise scores.
MatchAlign: aligning residues (54 vs 54)…
MatchAlign: score 103.000
ExecutiveAlign: 216 atoms aligned.
ExecutiveRMS: 13 atoms rejected during cycle 1 (RMS=0.67).
ExecutiveRMS: 10 atoms rejected during cycle 2 (RMS=0.35).
ExecutiveRMS: 8 atoms rejected during cycle 3 (RMS=0.28).
ExecutiveRMS: 3 atoms rejected during cycle 4 (RMS=0.25).
ExecutiveRMS: 2 atoms rejected during cycle 5 (RMS=0.24).
Executive: RMS =    0.235 (180 to 180 atoms)
```

You should get something like this:



Change the rendering of structure “5fd1” to “ribbon” and make it black. Change the color of the “1fdx” model by clicking on the “C” button and by selecting “spectrum > b-factors”. This colors each residue according to the “B-factor” information written into file “`1fdx.B99990001.pdb`” by MODELLER.

You should get the following image:



The number written in place of the “B-factors” actually correspond to the total “violation” of the model at each position along the sequence: blue means a lower level of violation and red means a higher level of violation.

STEP 3: Assess the model using the DOPE score

Modify the Python script as follows:

```
a = automodel(env,
              alnfile  = 'alignment.ali',     # alignment filename
              knowns   = '5fd1',              # codes of the templates
              sequence = '1fdx',              # code of the target
              assess_methods = (assess.DOPE))
```

- Run the modified script (by typing "`mod9.19 model-default.py`") and inspect the new log file. What is the value of the DOPE score for the final model? (Note that a lower DOPE score means a better model.)
- Keep a copy of the file "`model-default.log`" by renaming it "`model-default-ali.log`". In particular, make sure you keep a copy of the "Summary of the restraint violations":

```
Summary of the restraint violations:

   NUM     ... number of restraints.
   NUMVI   ... number of restraints with RVIOL > VIOL_REPORT_CUT[i].
   RVIOL   ... relative difference from the best value.
   NUMVP   ... number of restraints with -Ln(pdf) > VIOL_REPORT_CUT2[i].
   RMS_1   ... RMS(feature, minimally_violated_basis_restraint, NUMB).
   RMS_2   ... RMS(feature, best_value, NUMB).
   MOL.PDF ... scaled contribution to -Ln(Molecular pdf).

 #                    RESTRAINT_GROUP      NUM   NUMVI  NUMVP   RMS_1   RMS_2         MOL.PDF     S_i
-------------------------------------------------------------------------------------------------------
 1 Bond length potential              :     379       0      0   0.005   0.005       2.8674       1.000
 2 Bond angle potential               :     520       0      0   2.409   2.409       58.247       1.000
 3 Stereochemical cosine torsion poten:     244       0     11  54.040  54.040       100.17       1.000
 4 Stereochemical improper torsion pot:     146       0      0   1.402   1.402       5.7486       1.000
 5 Soft-sphere overlap restraints     :     637       0      0   0.003   0.003     0.47098       1.000
 6 Lennard-Jones 6-12 potential       :       0       0      0   0.000   0.000       0.0000       1.000
 7 Coulomb point-point electrostatic p:       0       0      0   0.000   0.000       0.0000       1.000
 8 H-bonding potential                :       0       0      0   0.000   0.000       0.0000       1.000
 9 Distance restraints 1 (CA-CA)      :     804       0      0   0.324   0.324       24.673       1.000
10 Distance restraints 2 (N-O)        :     837       0      1   0.468   0.468       38.091       1.000
11 Mainchain Phi dihedral restraints  :       0       0      0   0.000   0.000       0.0000       1.000
12 Mainchain Psi dihedral restraints  :       0       0      0   0.000   0.000       0.0000       1.000
13 Mainchain Omega dihedral restraints:      53       0      0   4.703   4.703       13.824       1.000
14 Sidechain Chi_1 dihedral restraints:      43       0      0  76.876  76.876       11.203       1.000
15 Sidechain Chi_2 dihedral restraints:      26       0      0  73.033  73.033       11.449       1.000
16 Sidechain Chi_3 dihedral restraints:       9       0      0  46.929  46.929       4.0291       1.000
17 Sidechain Chi_4 dihedral restraints:       1       0      0   9.151   9.151      0.27906       1.000
18 Disulfide distance restraints      :       0       0      0   0.000   0.000       0.0000       1.000
19 Disulfide angle restraints         :       0       0      0   0.000   0.000       0.0000       1.000
20 Disulfide dihedral angle restraints:       0       0      0   0.000   0.000       0.0000       1.000
21 Lower bound distance restraints    :       0       0      0   0.000   0.000       0.0000       1.000
22 Upper bound distance restraints    :       0       0      0   0.000   0.000       0.0000       1.000
23 Distance restraints 3 (SDCH-MNCH)  :     553       0      0   0.486   0.486       11.315       1.000
24 Sidechain Chi_5 dihedral restraints:       0       0      0   0.000   0.000       0.0000       1.000
25 Phi/Psi pair of dihedral restraints:      52       5      5  28.044  60.583       24.070       1.000
26 Distance restraints 4 (SDCH-SDCH)  :     100       0      0   0.870   0.870       7.4694       1.000
27 Distance restraints 5 (X-Y)        :       0       0      0   0.000   0.000       0.0000       1.000
...
38 SAXS restraints                    :       0       0      0   0.000   0.000       0.0000       1.000
39 Symmetry restraints                :       0       0      0   0.000   0.000       0.0000       1.000
```

STEP 4: Try to model the sequence based on an alternative alignment

Create a copy of the file "`alignment.ali`", called "`alignment2.ali`", that corresponds exactly to the alignment presented in the "PRE-LAB REPORT" section, containing the sequences:

AFVVTDNC**IKCKYTDCVEVCPVDCFYEGPN**FLVIHPDECIDCALCEPECPAQAIFSEDEVP...
AYVINDSC**IACGA--CKPECPVN-IIQG-S**IYAIDADSCIDCGSCASVCPVGAPNPED------

instead of:

AFVVTDNC**IKCKYTDCVEVCPVDCFYEGPN**FLVIHPDECIDCALCEPECPAQAIFSEDEVP...
AYVINDSC**--IACGACKPECPVNIIQGS--**IYAIDADSCIDCGSCASVCPVGAPNPED------

- ✦ Modify the Python script so that it reads file "`alignment2.ali`" instead of "`alignment.ali`" and run the script. How does the DOPE score for "`ali2`" compare to the one for "`ali`"?
- ✦ Keep a copy of the file "`model-default.log`" by renaming it "`model-default-ali2.log`". Based on the "Summary of the restraint violations", how do the violations for "`ali2`" compare to those for "`ali`"?

STEP 5: Redo STEP 1, STEP 2, and STEP 3 for your sequence

Save the Python script used in previous step (including the "DOPE" assessment) under the name "`model.py`". Modify that script file to use your sequence and structure:

- Change the value of variable "`alnfile`" from "`alignment.ali`" to whatever name you gave your PIR file.
- Change the value of variable "`knowns`" from "`5fd1`" to the four-character PDB ID of your template protein and change the value of variable "`sequence`" from "`1fdx`" to "`query`" (or whatever you have put in the PIR file).

- ✦ Run the script (by typing "`mod9.19 model.py`") and inspect the log file "`model.log`" for error and warning messages. You are now using your own PIR file, so pay special attention to the output from the alignment checking command (search for "`check_ali___>`") and correct your PIR file if necessary.
- ✦ What is the DOPE score of your model?
- ✦ Produce figures similar to those of STEP 2 for your model. Based on the alignment of the two structures, what is the RMS distance for the backbone atoms ("N+CA+C+O")? (Note that the RMS distances are given in Angstroms.)
- ✦ Perform the same structure alignment using only the alpha-carbons (using "`name ca`" instead of "`name n+ca+c+o`" in the "`align`" command). What is the alpha-carbon RMS distance?
- ✦ How well are the two structures overlapping? Discuss any significant deviation of the model with respect to the template.
- ✦ Inspect the "`.V99990001`" file for your model. Which amino acids have the largest total violation, and which "restraints groups" are they violating the most? (Note: There are 39 columns, which correspond to the 39 "restraint groups" from the "Summary of the restraint violations" in the log file.)

Back up your files

Back up your files as explained above.