CHEM 498Q / CHEM 630Q: Molecular Modelling of Proteins
TUTORIAL #2b: Homology Modelling 2

INTRODUCTION

The main goal of tutorial #2b is to learn how to improve homology models by including structurally important ligands and by changing the optimization protocol.

This tutorial generates a lot of files, so make sure you always assign the correct values of "`starting_model`" and "`ending_model`" in all your scripts.

First, make sure you keep a copy of the files you have created in STEP 5 in a separate directory.

PRE-LAB REPORT

None

PROCEDURE

STEP 6: Include cofactors, metal atoms, and buried water molecules in the model

This step is necessary for those who have a protein that binds cofactors or metal atoms, or that is modelled after a template structure that contains buried water molecules. If your template structure contains no ligands, no metal atoms, and no buried water molecules, go directly to STEP 7.

Note:
Consider a "buried" water molecule to be any molecule you don't see from the surface of the protein if you render it as "spheres".

To decide whether or not to include a buried water molecule in the model:

- In PyMOL, zoom in on each buried water molecule and list the proteins side chains surrounding it (and likely forming hydrogen bonds with it).
- Include the water if those side chains are the same in your sequence, or if they are replaced by smaller ones. Do not include the water if they are replaced by larger residues.

If your template structure has many water molecules at the surface of the protein, you can delete most of them by using the PyMOL console as following:

```
> hide
> show surface
> set surface_color, white
> set transparency, 0.5
> show sphere, hoh/
> set dot_solvent, on          (to calculate the solvent-accessible surface area (SASA))
> get_area hoh/, load_b=1      (calculate the SASA of each water molecule and store the results
                                in the "B-factor" column)
> select water, hoh/           (create a new object called "water", which you can then color
                                according to the "B-factor")
> remove hoh/ and b>30         (remove all water molecules with a SASA > 30 Å²)
> remove hoh/ and b>28         (remove all water molecules with a SASA > 28 Å²)
> ...                          (keep reducing the SASA cutoff until you have a manageable
                                number of water molecules to examine and delete by hand)
```

To read in the cofactors, metal atoms, and water molecules in your model, do the following:

- Add the lines "`env.io.hetatm = True`" and "`env.io.water = True`" to all your MODELLER Python scripts (see example below).
- Modify the template sequence of your PIR file by adding as many dots ("`.`") as you have residues following the last amino acid needed for your protein sequence (whether they are ATOM or HETATM residues). Modify the target sequence by adding either dashes ("`-`"), dots ("`.`"), or "`w`" depending on whether you want to ignore the residue, include it in your model, or include it as a water molecule. Change the "*`end_res`*" fields accordingly. Make sure the residues you are reading from the PDB file are numbered contiguously.

Note:
It is easy to make mistakes in preparing the PIR file, and to end up selecting the wrong HETATM residues (especially if the template structure contains many crystal water molecules but only a few buried ones). Make sure your final model has exactly the ligands you want.

Note:
For more information on how to prepare the PIR file, see http://salilab.org/modeller/manual/node18.html.

STEP 7: Produce multiple models

Modify the Python script from STEP 5 (file "`model.py`") so that "`a.ending_model = 10`". Change the "`env.io.hetatm`" and "`env.io.water`" variables if needed. Your script file should look like this (with the questions marks replaced by what is appropriate for your system):

`model.py`

```
# Homology modeling by the automodel class
from modeller import *              # Load standard Modeller classes
from modeller.automodel import *    # Load the automodel class

log.verbose()    # request verbose output
env = environ()  # create a new MODELLER environment to build this model in

# directories for input atom files
env.io.atom_files_directory = ['.', '../atom_files']

# Read in HETATM records from template PDBs (only if needed)
env.io.hetatm = True

# Read in water molecules from template PDBs (only if needed)
env.io.water = True

a = automodel(env,
              alnfile  = '????.ali',     # alignment filename
              knowns   = '????',         # codes of the templates
              sequence = '????',         # code of the target
              assess_methods = (assess.DOPE))
a.starting_model= 1                 # index of the first model
a.ending_model  = 10                # index of the last model
                                    # (determines how many models to calculate)
a.make()                            # do the actual homology modeling
```

From the MODELLER command prompt, type:

```
mod9.15 model.py
```

This will produce 10 models of your sequence based on the alignment with the template you provided in the "`.ali`" file. These files will be called "`.B99990001.pdb`" to "`.B99990010.pdb`".

- Inspect the "`model.log`" file and report the sums of violations and the DOPE scores of all 10 models. (In the "`.log`" file, the sum of violations is called "`molpdf`".) How do the two numbers correlate?
- Open all 10 PDB files in PyMOL and align their backbone atoms ("`n+ca+c+o`") to those of the template structure. Report the 10 RMS distances. (Note that you don't have to re-type your "align" command every time: you can just press the "up" arrow and modify the old one.)
- How do these models compare to the one obtained in STEP 5? Has the presence of ligands (if any) affected the structure?
- Prepare a raytraced figure similar to the one from STEP 2, showing the template structure and the 10 aligned models colored according to their total "violation". Show the template as a black "ribbon" and the 10 models as "cartoons". Render metal atoms and buried water molecules as "spheres", and molecular cofactors as "sticks" (if you have any).

Example:

STEP 8: Improve the optimization protocol

Copy file "`model.py`" under the name "`model2.py`" and do the following modifications to it (in red):

`model2.py`

```
# Homology modeling by the automodel class
from modeller import *              # Load standard Modeller classes
from modeller.automodel import *    # Load the automodel class

log.verbose()    # request verbose output
env = environ()  # create a new MODELLER environment to build this model in
```

```
# directories for input atom files
env.io.atom_files_directory = ['.', '../atom_files']

# Read in HETATM records from template PDBs (only if needed)
env.io.hetatm = True

# Read in water molecules from template PDBs (only if needed)
env.io.water = True

a = automodel(env,
              alnfile  = '????.ali',     # alignment filename
              knowns   = '????',         # codes of the templates
              sequence = '????',         # code of the target
              assess_methods = (assess.DOPE))
a.starting_model= 11                 # index of the first model
a.ending_model  = 20                 # index of the last model
                                     # (determines how many models to calculate)

# Thorough VTFM optimization:
a.library_schedule = autosched.slow
a.max_var_iterations = 300

# Thorough MD optimization:
a.md_level = refine.slow

a.make()                             # do the actual homology modeling
```

Run the script "`model2.py`". This will produce 10 more models (files "`.B99990011.pdb`" to "`.B99990020.pdb`"), optimized and refined using a more thorough protocol.

✦ Report the sum of violations and the DOPE scores of these new models. How do they compare with the models obtained in STEP 7?

STEP 9: Build an "all-hydrogen" model from the best model

Copy file "`model2.py`" under the name "`model3.py`" and modify this new script as following:

- Replace the keyword "`automodel`" by "`allhmodel`".
- Use "`a.starting_model = 21`" and "`a.ending_model = 21`".
- The same way you added the argument "`assess_methods`" in STEP 3, add yet another argument "`inifile = '????.B999900??'`", where the questions marks should be replaced to match the name of the model that has the lowest DOPE score among the 20 you have produced so far.

Your "`model3.py`" script file should look like this (changes in red):

`model3.py`

```
# Homology modeling by the allhmodel class
from modeller import *               # Load standard Modeller classes
from modeller.automodel import *     # Load the automodel class

log.verbose()    # request verbose output
```

```
env = environ()  # create a new MODELLER environment to build this model in

# directories for input atom files
env.io.atom_files_directory = ['.', '../atom_files']

# Read in HETATM records from template PDBs (only if needed)
env.io.hetatm = True

# Read in water molecules from template PDBs (only if needed)
env.io.water = True

a = allhmodel(env,
              alnfile  = '????.ali',     # alignment filename
              knowns   = '????',         # codes of the templates
              sequence = '????',         # code of the target
              inifile  = '????.B999900??',
              assess_methods = (assess.DOPE))
a.starting_model= 21                 # index of the first model
a.ending_model  = 21                 # index of the last model
                                     # (determines how many models to calculate)
a.make()                             # do the actual homology modeling
```

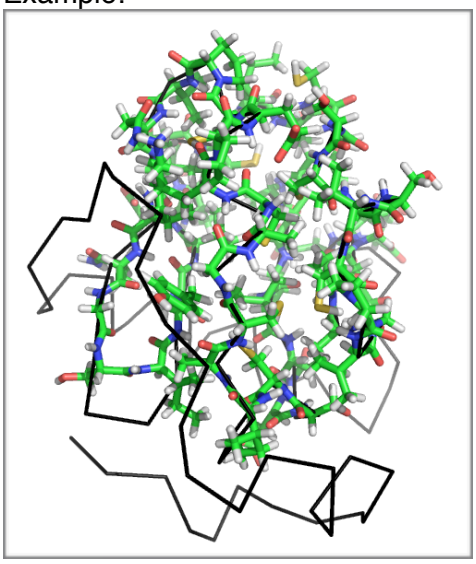
Note:
The alignment (PIR) file may have to be modified so that the residue numbering for the target sequence matches that of the file being used as "ini" file.

Run the new script. This will produce a PDB file "`.B99990021.pdb`" that contains a model in which hydrogen atoms have been added.

Inspect file "`model3.log`" to make sure the "inifile" you provided in the script was correctly read (search for the name of the file). Compare the content of that "inifile" to the content of the "`????.ini`" file just created. The XYZ coordinates of heavy atoms should be the same in both.

✦ Prepare a raytraced figure of that "all-atom" model, aligned on the structural template. Render the template as a black "ribbon" and the model as "sticks". Render metal atoms and buried water molecules as "spheres", and molecular cofactors as "sticks" (if you have any).

Example:



STEP 10: Examine the structure of your "all-hydrogen" model

- ✦ Examine the structures of the N-terminus and C-terminus residues of your model (that is, the first and last amino acids of your target sequence). How different are they from those of equivalent amino acids in the middle of the sequence?
- ✦ Find some of the polar contacts predicted by your model by clicking on the "A" button of the object, and selecting "find > polar contacts > involving side chains". Inspect visually at least 10 of these contacts and list the pairs of amino acids involved. (Indicate which of the two is the proton donor.) Start with the polar contacts involving functionally important amino acids.

- ✦ Examine *one by one* all histidine residues (H) of your sequence and consider the possibility that their protonation states have been incorrectly assigned by MODELLER. Note that histidines have two neutral protonation states: Would the alternate neutral protonation state create better polar contacts or remove a hydrogen atom clash?
- ✦ Examine the protonation states of all residues in contact with cofactors, metals atoms, or water molecules. Comment on any protonation state assignment likely to be incorrect.

Note:
When they are exposed to a solvent at physiological pH, lysine (K) and arginine (R) are protonated, while aspartic acid (D) and glutamic acid (E) are deprotonated. This is the assumption MODELLER is making for all K, R, D, and E residues in the protein, regardless of their environment. We will leave this assumption unexamined for now.