Comparative Modeling: Multi-template modeling with RosettaCM

This tutorial will guide you through modeling the rhodopsin, a class A GPCR using comparative modeling with multiple templates. This protocol also includes additional relax to incorporate a disulfide bond. Multi-template comparative modeling will be performed with the RosettaCM protocol and three class A GPCR templates (beta-2 adrenergic receptor, PDB: 2RH1; Adenosine A2A receptor, PDB: 3EML; and CXCR4, PDB: 3ODU). Final results can be compared to the actual crystal structure of rhodopsin (PDB: 1U19) for accuracy.

1. Setup

Comparative modeling requires various input files that are either generated manually or downloaded from the internet. These files have already been created and are available in their appropriate directories but it is recommended that you try to gather/generate these files yourself. **Boldface** indicates specific filenames. *Italics* indicates webpage entries such as query terms or menu selections.

To start, create your own working directory & move into it by typing:

```
mkdir my_model/
cd my_model/
```

Prepared files can be copied from the indicated directories into your working directory at any step if you wish to skip creating a particular file yourself.

a. Target Sequence

Your target protein is rhodopsin. With most comparative modeling applications, you will only have your target's amino acid sequence to start with.

DOWNLOAD THE SEQUENCE FOR RHODOPSIN IN FASTA FORMAT:

1. Go to www.ncbi.nlm.nih.gov/protein/
2. Type *1u19A* in the search bar.
3. Click the *Display Files* -> *FASTA* link to see the protein sequence in FASTA format.
4. Copy all the sequence information for chain A, including the line beginning with ">", into a file called **1u19.fasta**.

   gedit 1u19.fasta
5. Replace the first line of the file with ">1u19"
6. Remove the N- and C-terminal regions to reduce the time and difficulty of this comparative modeling project:

   a. Delete `XMNGTEGPNFYVPFSNKTGVVRSPFEAPQYYLAE` from the beginning of the sequence.
   b. Delete `KNPLGDDEASTTVSKTETSQVAPA`from the end of the sequence.
   c. Save the changes to **1u19.fasta**. **1u19.fasta** should look like this:

   1u19 PWQFSMLAAYMFLLIMLGFPINFLTLYVTVQHKKLRTPLNYILLNLAVADLFMVFG-GFTTTLYTSLHGYFVF GPTGCNLEGFFATLGGEIALWSLVVLAIERYVVVCKPMSNFR-FGENHAIMGVAFTVMALACAAPPLVGWS RYIPEGMQCSCGIDYYTPHEETNNES-FVIYMFVVHFIIPLIVIFFCYGQLVFTVKEAAAQQQESATTQKAEKE VTRMVIIMVI-AFLICWLPYAGVAFYIFTHQGSDFGPIFMTIPAFFAKTSAVYNPVIYIMMNKQFRNCMVTTLCCG

The prepared **1u19.fasta** can be found in

```
 ~/rosetta_workshop/tutorials/rosetta_cm/1_setup/
```

b. Template structures

Comparative modeling requires template structures to guide the target sequence folding. Since rhodopsin is a class A GPCR and all class A GPCR's have the same basic structure profile (7 transmembrane helices, 3 intracellular loops, 3 extracellular loops), we will use three class A GPCR crystal structures as templates. These structures are available on the RCSB Protein Data Bank (PDB). The raw structures from the PDB often contain information not necessary for comparative modeling such as attached T4 lysozyme and/or specific ligands. Once a PDB is downloaded for use as a template, this extra information must be removed before it can be used for comparative modeling with RosettaCM.

DOWNLOAD TEMPLATE PDB's:

1. Go to www.rcsb.org.
2. Search for *2RH1*.
3. Click *Download Files* -> *PDB File (text)*.
4. Remove the T4 lysozyme residues from chain A. These residues appear within the chain A sequence and are numbered 1002-1161. Simply delete any line for chain A residues 1002-1161 (between residues 230 and 263) from **2RH1.pdb**

   gedit 2RH1.pdb ->manually delete lines by highlighting and clicking *delete*
5. Save changes to **2RH1_ISOLATED.pdb**
6. Repeat steps 1 - 5 for *3EML* and *3ODU*. The extra residues to be removed from **3EML.pdb** include chain A residues numbered 1002-1161 (between residues 208 and 222). The extra residues to be removed from **3ODU.pdb** include the first 7 residues of chain A (27-33), the end of chain A (residues 303-328) and the lysozyme residues including those numbered 900-901 and 1002-1161 and 1200-1201 (between residues 229 and 230).
7. In addition to extra residues, these PDB's contain additional information that is not useful for Rosetta and may cause problems during the modeling. A script has been prepared to remove all of this extraneous information. This script has the following usage: clean_pdb.py <pdb file> <chain letter>

   ```
   ~/rosetta_workshop/tutorials/rosetta_cm/scripts/clean_pdb.py 2RH1_ISOLATED A
   ~/rosetta_workshop/tutorials/rosetta_cm/scripts/clean_pdb.py 3EML_ISOLATED A
   ~/rosetta_workshop/tutorials/rosetta_cm/scripts/clean_pdb.py 3ODU_ISOLATED A
   ```

Running these three commands should yield the files: **2RH1_ISOLATED_A.pdb, 2RH1_A.fasta, 3EML_ISOLATED_A.pdb, 3EML_A.fasta, 3ODU_ISOLATED_A.pdb**, and **3ODU_A.fasta**.

To make sure that you removed all necessary residues from chain A, compare your **2RH1_A.fasta, 3EML_A.fasta**, and **3ODU_A.fasta** to those that have already been prepared. They should be identical.

RENAME CLEANED TEMPLATES:

```
    mv 2RH1_ISOLATED_A.pdb 2rh1.pdb
    mv 3EML_ISOLATED_A.pdb 3eml.pdb
    mv 3ODU_ISOLATED_A.pdb 3odu.pdb
```

Note: Rosetta's threading is very particular with its interpretation of filenames so renaming them is necessary for it to function properly. All prepared files for this step can be found in

```
~/rosetta_workshop/tutorials/rosetta_cm/1_setup/
```

c. Align target sequence to templates

Comparative modeling uses template structures to guide initial placement of target amino acids in three-dimensional space. This is done according to the sequence alignment of target and template. Residues in the target sequence will be assigned the coordinates of those residues they align with in the template structure. Residues in the target sequence that do not have an alignment partner in any template will be filled in during the hybridize step.

SIMULTANEOUSLY ALIGN TARGET AND ALL TEMPLATE SEQUENCES:

1. Go to http://www.ebi.ac.uk/Tools/msa/clustalo/
2. Copy/paste all sequence information from your four fasta files (**1u19.fasta, 2RH1__A.fasta, 3EML__A.fasta**, and **3ODU__A.fasta** including the ">" header line into clustal.

   ```
   cat *.fasta
   ```

   ->highlight sequences in terminal and paste them into webserver by clicking the middle mouse button
3. Leave default settings and click *Submit*
4. *Download* the *alignment* to a file called **1u19__2rh1__3eml__3odu.aln**

The prepared alignment can be found in `~/rosetta_workshop/tutorials/rosetta_cm/2_threading/`

5. Copy the adjusted alignment file **1u19__2rh1__3eml__3odu__adjusted.aln** to your directory. Don't forget to include the "." at the end of the line:

   ```
   cp ~/rosetta_workshop/tutorials/rosetta_cm/2_threading/1u19_2rh1_3eml_3odu_adjusted.aln .
   ```

It is recommended that you skip the following step during this tutorial and use the prepared "adjusted alignment" file that you just copied and return to this step while either the hybridize or relax processes are running.

Because comparative modeling uses alignments to assign initial coordinates to the target sequence, it is sometimes necessary to adjust the alignments before threading is performed. This is not an absolute requirement and may vary depending on the target and templates. In this example, we are modeling a class A GPCR that contains 7 transmembrane helices, each of which contains one or more highly conserved residues between class A GPCR's. The accuracy of our comparative models can be improved if we ensure that our sequence alignment follows certain structural expectations. Our expectations include alignment of the highly conserved residues within each transmembrane helix and helix continuity. In other words, we want to remove any gaps in the transmembrane regions of these alignments. Alignment gaps represent regions in which Rosetta must either insert missing target residues or skip template residues. This may inappropriately disrupt helix regions during the threading process, making Rosetta's subsequent relaxation steps more difficult.

CLUSTAL format alignments can be edited with a number of sequence alignment editors, or they can be (carefully) adjusted using a text editor.

6. Manually adjust the alignment using a a sequence alignment editor or text editor.

To get an idea of how these alignments were adjusted to align conserved residues and remove trans-membrane gaps, color-coded alignments before and after adjustments were made are provided as open office spreadsheet files. To open the files type *ooffice* and use the *file toolbar* to find and open the files.

These files can be found at

```
~/rosetta_workshop/tutorials/rosetta_cm/2_threading/alignment_analysis.ods
~/rosetta_workshop/tutorials/rosetta_cm/2_threading/alignment_analysis_adjusted.ods
```

d. Fragment files

RosettaCM will use fragments to fill in missing residues that didn't align with any template sequences and to connect the different pieces during hybridization.

GENERATE 3MER AND 9MER FRAGMENT LIBRARIES:

1. Go to robetta.bakerlab.org and register as an academic or non-profit user.
2. Go to robetta.bakerlab.org/fragmentsubmit.jsp
3. Fill in your *username*.
4. Put *1u19* under "Target name"
5. Copy/paste all text from **1u19.fasta** into the provided field.

   ```
   cat ~/rosetta_workshop/tutorials/rosetta_cm/my_model/1u19.fasta
   ```

   ->highlight sequence shown in terminal window. Center the center button of the mouse to paste the sequence into the Robetta *Paste Fasta* window
6. Click "*Submit*".
7. See your position in the queue by clicking *Queue* under *Fragment Libraries*. Click on the *job ID* link for your job and refresh to monitor progress until completed.
8. Generating fragment files may take time. While you are waiting for the fragment files, you may continue on through 1.e. and all of step 2.
9. Once completed, fragment files can be downloaded and should be saved as **1u19__3.frags** and **1u19__9.frags**.

Prepared fragment files can be found in

```
~/rosetta_workshop/tutorials/rosetta_cm/1_setup/
```

e. Define the membrane region: 1u19.span

Rhodopsin is a membrane protein but we may not know which residues are within the membrane region. This information can be predicted based on the amino acid sequence.

A "span file" informs Rosetta which portions of the protein exist within the membrane. Rosetta uses this information to apply different scoring terms to soluble residues versus those in the membrane.

There are various topology prediction algorithms available. OCTOPUS makes predictions based on artificial neural networks trained with many protein sequences and structures to identify residues at the sites of membrane entry, reentry, membrane dip, TM hairpin regions, and membrane exit. OCTOPUS predictions have been shown to be effective approximately 96% of the time. We will be using the predictions directly from OCTOPUS in this example. However, it may be necessary to adjust your own predictions to reflect any experimental evidence you may have that is not reflected in the OCTOPUS prediction.

CREATE SPAN FILE USING OCTOPUS PREDICTIONS

1. Go to http://octopus.cbr.su.se
2. Submit the sequence from **1u19.fasta**

```
cat ~/rosetta_workshop/tutorials/rosetta_cm/my_model/1u19.fasta
```

->highlight sequence shown in terminal window. Center the center button of the mouse to paste the sequence into the Octopus *Fasta* window

3. *Save* the OCTOPUS *topology file* as **1u19.octopus**
4. Convert the OCTOPUS file to a span file using the script:

```
~/rosetta_workshop/tutorials/rosetta_cm/scripts/octopus2span.pl 1u19.octopus > 1u19.span
```

2. Threading

a. Convert alignments to Grishin format.

Rosetta's threading requires alignments to be supplied in Grishin format. This is an uncommon alignment format and we will manually prepare the Grishin alignment files from our multiple sequence alignment. With the Grishin format, each template-target alignment gets its own alignment file.

It is recommended that you copy the prepared Grishin files into your working directly rather than converting the alignments manually.

The prepared Grishin alignment files (**1u19__2rh1.grishin, 1u19__3eml.grishin, 1u19__3odu.grishin**) can be copied into your working directory from

```
~/rosetta_workshop/tutorials/rosetta_cm/2_threading/

cp  ~/rosetta_workshop/tutorials/rosetta_cm/2_threading/1u19_2rh1.grishin .
cp  ~/rosetta_workshop/tutorials/rosetta_cm/2_threading/1u19_3eml.grishin .
cp  ~/rosetta_workshop/tutorials/rosetta_cm/2_threading/1u19_3odu.grishin .
```

To manually convert your alignments, you can follow the format specifications to generate three individual alignment files yourself using a linux editing tool such as `gedit`. Grishin format specifications:

```
## Target_name template_pdb_file
#
scores from program: 0
0 target sequence copied from the alignment file (continuous)
0 template sequence copied from the alignment file (continuous)
```

Notice that, unlike clustalO, each file contains two sequences, the target and template and appear one after another in their entirety, rather than broken up over several lines. Both sequences are preceded on the same line by a "0" and a single space.

b. Thread target sequence over three individual template sequences.

Rosetta's partial thread application will generate .pdb files for each target-template alignment by assigning coordinates from the template pdb onto the aligned residues in the target sequence. This will be run once for each target-template alignment and will result in three threaded .pdb files.

1. Run the following commands:

```
~/rosetta_workshop/rosetta/main/source/bin/partial_thread.default.linuxgccrelease \
```

-in:file:fasta 1u19.fasta -in:file:alignment 1u19_2rh1.grishin -in:file:template_pdb 2rh1.pdb

```
~/rosetta_workshop/rosetta/main/source/bin/partial_thread.default.linuxgccrelease \
-in:file:fasta 1u19.fasta -in:file:alignment 1u19_3eml.grishin -in:file:template_pdb 3eml.pdb

~/rosetta_workshop/rosetta/main/source/bin/partial_thread.default.linuxgccrelease \
-in:file:fasta 1u19.fasta -in:file:alignment 1u19_3odu.grishin -in:file:template_pdb 3odu.pdb
```

2. After running these commands, you should have three new pdbs: **2rh1.pdb.pdb, 3eml.pdb.pdb**, and **3odu.pdb.pdb**. For clarity, rename these files appropriately:

```
mv 2rh1.pdb.pdb 1u19_on_2rh1.pdb
mv 3eml.pdb.pdb 1u19_on_3eml.pdb
mv 3odu.pdb.pdb 1u19_on_3odu.pdb
```

Prepared threaded pdbs can be found in

```
~/rosetta_workshop/tutorials/rosetta_cm/2_threading/
```

3. RosettaCM Hybridize

RosettaCM is capable of breaking up multiple templates and generating hybridized structures that contain pieces from different templates. This provides a more accurate comparative model by including different pieces from each of the threaded structures to include those that are most energetically favorable given the residues in the target sequence. Additionally, this application uses fragments and minor ab initio folding to fill in residues not previously assigned coordinates during the threading process.

This step requires the following files to be in the same directory:

- **1u19_3.frags** (downloaded during setup)
- **1u19_9.frags** (downloaded during setup)
- **1u19_on_2rh1.pdb** (generated during threading)
- **1u19_on_3eml.pdb** (generated during threading)
- **1u19_on_3odu.pdb** (generated during threading)
- **1u19.fasta** (downloaded and altered during setup)
- **stage1_membrane.wts** (will be created in this step)
- **stage2_membrane.wts** (will be created in this step)
- **stage3_rlx_membrane.wts** (will be created in this step)
- **rosetta_cm.xml** (will be created in this step)
- **rosetta_cm.options** (will be created in this step)
- **1u19.span** (generated during setup)

a. Generate the weights files

RosettaCM uses individual scoring weights for each stage. Since this is a membrane protein, we will be using weights that include membrane-specific scoring terms: **stage1_membrane.wts, stage2_membrane.wts**, and **stage3_rlx_membrane.wts**.

Note that non-membrane versions of these weight files are also included in this directory in case you wish to try RosettaCM with non-membrane proteins. For now, just copy the _membrane.wts files to your working directory. The weight files can be copied into your working directory from `~/rosetta_workshop/tutorials/rosetta_cm/3_hybridize/`

```
cp ~/rosetta_workshop/tutorials/rosetta_cm/3_hybridize/stage1_membrane.wts .
cp ~/rosetta_workshop/tutorials/rosetta_cm/3_hybridize/stage2_membrane.wts .
cp ~/rosetta_workshop/tutorials/rosetta_cm/3_hybridize/stage3_rlx_membrane.wts .
```

b. Define hybridize script: rosetta_cm.xml

RosettaCM hybridize is run as a Rosetta scripts mover. Therefore, **rosetta_cm.xml** will define the hybridize mover, assign the different weight files to each stage, and list all threaded pdbs. In this example, all threaded files are given identical weights. However, one can adjust individual weights for each threaded pdb, increasing or decreasing the likelihood that fragments from that particular template-threading will appear in the hybrid model. The Rosetta scripts file **rosetta_cm.xml** can be copied into your working directory from `~/rosetta_workshop/tutorials/rosetta_cm/3_hybridize/`

```
cp  ~/rosetta_workshop/tutorials/rosetta_cm/3_hybridize/rosetta_cm.xml .
```

c. Define options: rosetta_cm.options

You can copy the prepared rosetta_cm.options into your working directory from

`~/rosetta_workshop/tutorials/rosetta_cm/3_hybridize/`

```
cp ~/rosetta_workshop/tutorials/rosetta_cm/3_hybridize/rosetta_cm.options .
```

d. Run RosettaCM hybridize

Run the RosettaCM hybridize protocol using Rosetta Scripts. As mentioned before, make sure all of the appropriate files are in the same directory. For production runs, at least 10000-15000 models should be created. However, note that the number of templates used, the length of the protein, the type of protein etc. all affect sampling size. For the purposes of this tutorial, you will only create three models.

```
~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
    @rosetta_cm.options -nstruct 3
```

This will generate 3 models: **S_0001.pdb, S_0002.pdb, S_0003.pdb**.

Models have already been generated using hybridize and can be found in `~/rosetta_workshop/tutorials/rosetta_cm/3_hybr`

4. Final relax

Rhodopsin is a membrane protein that contains three intracellular loops, three extracellular loops, and one disulfide bond. Due to the nature of RosettaCM hybridize, initial models are generated without constraint information such as disulfide bonds, or other experimentally-derived constraints that you might want to include such as known atom-pair distances, etc. To refine our models with this additional information we will finalize our models with a relaxation step in the presence of a membrane. This step requires the following files to be in the same directory:

- **1u19.span** (generated during setup)
- **1u19.disulfide** (will be created in this step)
- **relax.options** (will be created in this step)

a. Define disulfide bond: 1u19.disulfide

Experimental evidence may suggest specific constraints you want to impose during your comparative modeling. In this example, we know that there exists a disulfide bond between residues 77 and 154.

Disulfide constraint files contain one disulfide bond per line and include only the pair of residue position numbers involved for each line.

The prepared 1u19.disulfide file can be copied into your working directory from `~/rosetta_workshop/tutorials/dosetta_cm/`

```
cp ~/rosetta_workshop/tutorials/rosetta_cm/4_relax/1u19.disulfide .
```

b. Define options: relax.options

This options file will cause Rosetta to set up a membrane environment based on **1u19.span**, fix a disulfide bond between residues 77 and 154, and relax the models using a dualspace Cartesian/fast-relax algorithm that has been shown to provide better results than fast-relax on its own.

The prepared options file **relax.options** can be copied into your working directory from

`~/rosetta_workshop/tutorials/rosetta_cm/4_relax/`

```
cp ~/rosetta_workshop/tutorials/rosetta_cm/4_relax/relax.options .
```

c. Run relax

Perform this step once per each of the three models generated from hybridize (**S_0001.pdb** through **S_0003.pdb**): Note that for the purpose of this tutorial, the "nstruct" value has been placed at one. Additional relax runs may be necessary for individual cases.

```
~/rosetta_workshop/rosetta/main/source/bin/relax.default.linuxgccrelease \
    @relax.options -s S_0001.pdb -out:prefix S_0001_relax_ -nstruct 1

~/rosetta_workshop/rosetta/main/source/bin/relax.default.linuxgccrelease \
    @relax.options -s S_0002.pdb -out:prefix S_0002_relax_ -nstruct 1

~/rosetta_workshop/rosetta/main/source/bin/relax.default.linuxgccrelease \
    @relax.options -s S_0003.pdb -out:prefix S_0003_relax_ -nstruct 1
```

Final models relax can be found in:

```
~/rosetta_workshop/tutorials/rosetta_cm/4_relax/
```

5. Final model selection

Due to time constraints, we generated only three models. Ideally, you will generate 10,000 to 15,000 comparative models. From this collection of models you can then select a single comparative model or an ensemble of models. In this example, we will select the top scoring pose as our final comparative model. It is important to visually inspect your final models for any chain-breaks or violations such as broken disulfide bonds or failure to reflect any experimental expectations you may have regarding the structure of your target protein.

The previous step generated a score file for each model, all of which end in .fasc.

COMBINE SCORE FILES AND SORT BY TOTAL POSE SCORE:

```
cat *.fasc | sort -nk2 > scores_sorted.txt
```

This generates **scores_sorted.txt** and the first line is your best model by total pose score.

You can visually inspect this model using pymol or whichever visualization tool you prefer.