Comparative Modeling: Multi-template modeling with RosettaCM

This tutorial will guide you through modeling the dopamine 3 receptor (D3R), a class A GPCR using comparative modeling with multiple templates. Multi-template comparative modeling will be performed with the RosettaCM protocol and five class A GPCR templates. Final results can be compared to the actual crystal structure of rhodopsin (PDB: 3pbl) 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 within the rosetta_cm directory & move into it by typing:

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

Copy the rosetta_cm folder into your my_model/ folder. Prepared files can be copied from the demo directory into your working directory at any step if you wish to skip creating a particular file yourself. It is recommended to maintain the same directory architecture (i.e. /rosetta_cm/input_files/) both to ensure the scripts run correctly without modification and as a good way to separate different types of files. You can run the command

```
cp -R ../* .
```

to copy the rosetta_cm folder into your my_model folder.

a. Target Sequence

Your target protein is D3R. For our purposes we will be modeling the sequence from PDB 3pbl. However, with most comparative modeling applications, you will only have your target's amino acid sequence to start with and will retrieve this from NCBI.

```
    cd input_files/
```

OBTAIN D3R SEQUENCE FROM PDB:

1. Go to http://www.rcsb.org/pdb/
2. Type 3pbl in search bar.
3. Click the *Display Files* -> *FASTA sequence* link in upper right corner
4. Copy the fasta text into a file called **3pbl.fasta** In the command line type:

   ```
   gedit 3pbl.fasta
   ```

5. Replace the header with ">3pbl" as the first line of the file
6. Make a copy of the full fasta before we alter the sequence.

   ```
   cp 3pbl.fasta 3pbl_full.fasta
   ```

7. In 3pbl.fasta delete the fusion protein in intracellular loop 3:

    a. Delete the first 40 residues and the last 9 residues

    b. Replace the sequence

    ```
    NIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAKSELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGILRN
    AKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGTWDAY
    ```

    from ICL3 with

    ```
    AAAAAAAA
    ```

    c. Save the changes to **3pbl.fasta**.

OR; OBTAIN THE SEQUENCE FOR D3R FROM NCBI:

1. Go to https://www.ncbi.nlm.nih.gov/protein/
2. Type *DRD3* in the search bar.
3. Find the full human sequence
4. Click the *FASTA* link to see the protein sequence in FASTA format.
5. Copy all the sequence information including the line beginning with ">", into a file called **3pbl.fasta**. In the command line type:

    ```
    gedit 3pbl.fasta
    ```

6. Replace the first line of the file with `>3pbl`
7. Make a copy of the full fasta before we alter the sequence.

    ```
    cp 3pbl.fasta 3pbl_full.fasta
    ```

8. Remove the N-terminal region as this region is expected to be disordered and has no template information. Additionally, ICL3 is quite long and likely disordered. We will remove this:

    a. Delete `MASLSQLSSHLNYTCGAENSTGASQARPHAY` from the beginning of the sequence.

    b. Delete

    ```
    RILTRQNSQCNSVRPGFPQQTLSPDPAHLELKRYYSICQDTALGGPGFQ
    ERGGELKREEKTRNSLSPTIAPKLSLEVRKLSNGRLSTSLKLGPPQPR
    ```

    from ICL3.

    c. Add a short connector `AAAAAAAA` in the place of the removed ICL3 sequence.

    d. Save the changes to **3pbl.fasta**.

FINAL D3R SEQUENCE

3pbl.fasta should look like this:

```
>3pbl
YALSYCALILAIVFGNGLVCMAVLKERALQTTTNYLVVSLAVADLLVATLVMPWVVYLEVTGGVWNFSRICCDVFVTLDVM
MCTASIWNLCAISIDRYTAVVMPVHYQHGTGQSSCRRVALMITAVWVLAFAVSCPLLFGFNTTGDPTVCSISNPDFV
IYSSVVSFYLPFGVTVLVYARIYVVLKQRRRKAAAAAAAAGVPLREKKATQMVAIVLGAFIVCWLPFFLTHVLNTHC
QTCHVSPELYSATTWLGYVNSALNPVIYTTFNIEFRKAFLKILSC
```

The prepared **3pbl.fasta** can be found in ~**/rosetta_workshop/tutorials/rosetta_cm/demo/input_files/**

If your **3pbl.fasta** file matches with the file that we prepared, the command line `diff 3pbl.fasta ../demo/input_files/3pbl.fasta` should return nothing.

You should also have **3pbl_full.fasta** with the unaltered sequence, for one of the next steps. This file can be found in **/rosetta_workshop/tutorials/rosetta_cm/demo/input_files/**

b. Template structures

Comparative modeling requires template structures to guide the target sequence folding. D3R 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 templates from other Class A GPCRs identified from a sequence similarity search. 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.

From your my_model directory: cd template_pdbs

IDENTIFY TEMPLATES PDBs:

1. Go to Blastp http://blast.ncbi.nlm.nih.gov/ and select protein BLAST
2. Copy sequence from **3pbl.fasta** into search query
3. Select 'Protein Data Bank proteins(pdb)' in Database search set
4. Start search by clicking the 'Blast' button in bottom of page

Top hits for D3R include other bioamine receptors such as the adrenergic, serotonin, and muscarinic receptors. As multiple structures have been determined for redundant receptors, we select the receptor templates that have the best resolution and completeness in tn TM and extracellular loop regions. For this tutorial we will use five templates including the following:

a. 4iar: 5HT-1B
b. 4bvn: B1AR
c. 2rh1: B2AR
d. 5dsg: M4R
e. 5cxv: M1R

DOWNLOAD TEMPLATE PDBs:

1. Go to https://www.rcsb.org/pdb/.

2. Search for *2rh1*.

3. Click *Download Files* -> *PDB File (text)*.

4. Remove the fusion protein 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** in you prefered text editor

   `gedit 2rh1.pdb` -> manually delete lines by highlighting and clicking *delete*

   This can be dome in vi visual mode as well if you prefer a command line oriented text editor: vi 2rh1.pdb Move cursor (using arrow keys) to start of selection and press v for visual selection. You can use your cursor to choose your selection or type 100j or 200j to rapidly select 100 or 200 lines. To delete the selected lines, press x. Be sure that your cursor is at the end of the last line you want to delete before pressing x. To exit vi, type :wq (for write and quit - :w will save changes but not exit vi) Or :q (to quit) Or :q! (to quit and discard changes) Type u to undo changes Press v to exit visual mode as well.

5. Save changes to **2rh1_isolated.pdb**

6. Repeat steps 1 - 5 for *4bvn*, *4iar*, *5dsg*, and *5cxv*.
   No extra residues need to be removed from **4bvn.pdb**.
   The extra residues to be removed from **4iar.pdb** include chain A residues numbered 1001-1106.
   The extra residues to be removed from **5dsg.pdb** include chain A residues numbered 1002-1118.
   The extra residues to be removed from **5cxv.pdb** include chain A residues numbered 1001-1160.

   All files (direct from PDB and isolated) are located in directory:

   ~/rosetta_workshop/tutorials/rosetta_cm/demo/template_pdbs/original_files/

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>`

   List all isolated pdbs into a file and remove the .pdb file extension in the list

   ```
       ls *isolated.pdb | awk -F. '{print $1}' > list_of_pdbs.txt
   ```

   Run this command to clean the PDBs and generate a cleaned FASTA file for each (You may need to run this script with python2 for the copy of clean_pdb.py in template_pdbs/scripts/clean_pdb.py. Remember that using denotes that a command is being split between multiple lines.)

   ```
       cat list_of_pdbs.txt | xargs -n1 -I@ \
          python ~/rosetta_workshop/rosetta/main/tools/protein_tools/scripts/clean_pdb.py @ A
   ```

Running these commands should yield the files: **2rh1_isolated_A.pdb**, **2rh1_isolated_A.fasta**, **4bvn_isolated_A.pdb**, **4bvn_isolated_A.fasta**, **4iar_isolated_A.pdb**, **4iar_isolated_A.fasta**, **5cxv_isolated_A.pdb**, **5cxv_isolated_A.fasta**, **5dsg_isolated_A.fasta**, **5dsg_isolated_A.fasta**.

To make sure that you removed all necessary residues from chain A, compare your fastas to those that have already been prepared (hint: use the command `diff`). They should be identical.

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/demo/template_pdbs/**

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:

Move from your template_pdb directory to your alignment_files directory: cd ../alignment_files/

1. Go to http://www.ebi.ac.uk/Tools/msa/clustalo/
2. Copy/paste all sequence information from your fasta files including the ">" header line into clustal.

   ```
   cat ../input_files/3pbl.fasta
   cat ../template_pdbs/*.fasta
   ```

3. In Output Format, select Pearson/FASTA
4. *Download* the *alignment* to a file called **3pbl_2rh1_4bvn_4iar_5cxv_5dsg.aln**
5. This is an initial alignment. It is important to inspect the alignment to ensure conserved residues, helical spans, and loop regions are in agreement between the targets and templates. A prepared alignment can be found in ~**/rosetta_workshop/tutorials/rosetta_cm/demo/alignment_files/3pb_alignments.txt**

   ```
   cp ../demo/alignment_files/3pbl_alignments.txt .
   ```

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.

d. Define the membrane region: 3pbl.span

D3R 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 from OCTOPUS in this example. However, OCTOPUS is more accurate given the whole sequence. We will use the full sequence of the receptor as input for OCTOPUS, and then adjust the numbers to reflect our sequence. Be aware that 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. For example, OCTOPUS will predict only six transmembrane helices given the altered sequence we are modeling, but since it is a GPCR we know it should have seven transmembrane helices.

CREATE SPAN FILE USING OCTOPUS PREDICTIONS

From your alignment_files directory move to your input_files directory: cd ../input_files/

1. Go to http://octopus.cbr.su.se This will redirect you to https://topcons.net. Giving your sequence to Topcons will make octopus predictions as well as predictions from other membrane protein topology predictors. Topcons is a consensus method that uses predicted topologies from many methods such as octopus.

2. Submit the sequence from **3pbl.fasta**

   ```
   cat 3pbl_full.fasta
   ```

3. *Save* the OCTOPUS *topology file* as **3pbl.octopus** Since we will be using topcons to get octopus predictions, you should download the query.result.txt file and convert that to an octopus file.

   ```
   sed -n '1,11p; 17,18p' < template_pdbs/query.result.txt > 3pbl.octopus
   ```

4. Convert the OCTOPUS file to a span file using the script, be sure to replace /PATH/TO/ROSETTA/ with the correct path on your computer.

   ```
   /PATH/TO/ROSETTA/rosetta/main/source/src/apps/public/membrane_abinitio/octopus2span.pl \
     3pbl.octopus > 3pbl.span
   ```

5. You will then need to edit your span file to correct for the different residue numbers in our altered sequence. This could be time consuming. An already edited span file has been provided for you in the demo directory. You can copy it into your input_files directory.

   ```
   cp ../../demo/input_files/3pbl.span .
   ```

e. Define disulfide bond.

The conserved disulfide bond between TM3 and ECL2 needs to be predefined to ensure its formation during RosettaCM hybridization. Based on sequence position we have identified these cysteines as residues 72 and 150. A disulfide file is created which is a space-separated list of cysteine pairs. If multiple disulfides are present, they are listed on subsequent lines.

The prepared disulfide file can be found in the demo folder:

```
cp ../demo/input_files/3pbl.disulfide .
```

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.

From your input_files directory move to your alignment directory:

```
    cd ../alignment_files/
```

A script has been provided **make_alignment_files.sh** that converts the optimized alingnment to individual grishin files using an additional list_of_fastas.txt file. Run this command to generate grishin files.

```
    ./make_alignment_files.sh
```

To manually convert your alignments, you can follow the format specifications to generate 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 the template structures.

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 five threaded .pdb files, one for each template.

From your alignment directory, move back to your my_model directory:

```
    cd ../
```

Create a threaded_pdbs directory and move to it:

```
mkdir threaded_pdbs
    cd threaded_pdbs
```

You have multiple files to thread. You will need to run the command below for each template pdb, replacing PDB with the template PDB ID and updating the path to rosetta.

```
    /PATH/TO/ROSETTA/rosetta/main/source/bin/partial_thread.linuxgccrelease \
      -in:file:fasta ../input_files/3pbl.fasta -in:file:alignment ../alignment_files/PDB.aln \
      -in:file:template_pdb ../template_pdbs/PDB_isolated_A.pdb
```

The output files should be named **2rh1_out.pdb** and so on.

Prepared threaded pdbs can be found in ~**/rosetta_workshop/tutorials/rosetta_cm/demo/threaded_pdbs/**

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:

- **threaded_pdbs/2rh1_out.pdb** (generated during threading)
- **threaded_pdbs/4bvn_out.pdb** (generated during threading)
- **threaded_pdbs/4iar_out.pdb** (generated during threading)
- **threaded_pdbs/5cxv_out.pdb** (generated during threading)
- **threaded_pdbs/5dsg_out.pdb** (generated during threading)
- **input_files/3pbl.fasta** (downloaded and altered during setup)
- **input_files/stage1_membrane.wts** (will be created in this step)
- **input_files/stage2_membrane.wts** (will be created in this step)
- **input_files/stage3_rlx_membrane.wts** (will be created in this step)
- **input_files/rosetta_cm.xml** (will be created in this step)
- **input_files/rosetta_cm.options** (will be created in this step)
- **input_files/3pbl.span** (generated during setup)
- **input_files/3pbl.disulfide** (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 adjustment for non-membrane versions of these weight files are also included in those files in case you wish to try RosettaCM with non-membrane proteins. For now, just copy the _membrane.wts files into your input_files folder.

The weight files are found in the input_files directory

~/rosetta_workshop/tutorials/rosetta_cm/demo/input_files/

Move to input_files directory and copy the membrane weights files:

```
    cd ../input_files
    cp ../demo/input_files/*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.

In addition to the hybridize mover, we perform an additional relax step to ensure diversity in the output backbones and to energetically minimize the hybridized structures.

The Rosetta scripts file **rosetta_cm.xml** can be found in the demo/input_files directory. Copy it into your input_files folder:

```
cp ../demo/input_files/rosetta_cm.xml .
```

c. Define options: rosetta_cm.options

The options file is a means to clean up the command line. Many Rosetta protocols can take additional options to modify the output or direct Rosetta to the input files. Here we define input/output, the number of models to be made, membrane options, relax options, and additional options to aid in the computation.

The prepared **rosetta_cm.options** can be found in the input_files directory. Copy it into your input_files folder:

```
cp ../demo/input_files/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 1000-5000 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 one model.

Make a directory for your output files in your my_model directory:

```
    cd ../
    mkdir output_files
```

The following command launches the RosettaCM run. Be sure to run this from the my_models directory you created at the beginning

```
/PATH/TO/ROSETTA/rosetta/main/source/bin/rosetta_scripts.linuxgccrelease  \
    @ input_files/rosetta_cm.options
```

This will generate 1 model (as defined in the rosetta_cm.options file) in ~30 minutes: **S_0001.pdb**.

Additional models have already been generated using hybridize and can be found in

~/rosetta_workshop/tutorials/rosetta_cm/demo/output_files/

5. Final model selection

Due to time constraints, we generated only one models. Ideally, you will generate 1000 to 5000 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.

We suggest clustering models and selecting representatives from the largest, best scoring clusters.

This has previously been done and the top five models by cluster and energy have been deposited into the `demo/final_models/` directory.

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