

Comparative Modeling: Multi-template modeling with RosettaCM

This tutorial will guide you through modeling a dopamine 3 (D3) and dopamine 2 (D2) chimeric receptor, a class A GPCR using comparative modeling with multiple templates. Multi-template comparative modeling will be performed with the RosettaCM protocol and three class A GPCR templates. Final results can be compared to the actual crystal structure of the ‘normal’ dopamine 3 receptor (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 & 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. rosetta_cm_tutorial.md

a. Target Sequence

Your target protein is the chimeric D3/D2 receptor (replaces D3R extracellular loop 2 (ECL2) with D2R ECL2 sequence). For our purposes we will be modeling the sequence from PDB 3pbl and make changes to produce the d3d2 chimeric sequence. 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.

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 sequence for chain A into a file called **d3d2_chimera.fasta** In the command line type: `gedit d3d2_chimera.fasta`
5. Add “>d3d2_chimera” as the first line of the file
6. Be sure to delete the fusion protein in intracellular loop 3:
 - a. Delete the first 40 residues and the last 9 residues
 - b. Replace the sequence

```
NIFEMLRIDEGRLRLKIYKDTEGYTIGHLLTKSPSLNAAKSELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGILRN  
AKLKPVYDSLDAVRRRAALINMVFMGETGVAGFTNSLRMLQQRWDEAAVNLAKSRYNQTPNRAKRVITTFRTGTWDAY
```

from ICL3 with

```
AAAAAAA
```

- c. Replace the sequence

LLFGFNTTGDPTVCSISNPFD

from ECL2 with

LLFGLNADQNECIIANAPAF

- d. Save the changes to **d3d2_chimera.fasta**.

OR; OBTAIN THE SEQUENCE FOR D3R FROM NCBI:

1. Go to <http://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 **d3d2_chimera.fasta**.
In the command line type: `gedit d3d2_chimera.fasta`
6. Replace the first line of the file with `>d3d2_chimera`
7. 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. Replace the sequence

LLFGFNTTGDPTVCSISNPFD

from ECL2 with

LLFGLNADQNECIIANAPAF

- e. Save the changes to **d3d2_chimera.fasta**.

FINAL D3/D2 CHIMERA RECEPTOR SEQUENCE

d3d2_chimera.fasta should look like this:

```
>d3d2_chimera
YALSYCALILAIVFGNGLVCMVAVLKERALQTTTNYLVVSLAVADLLVATLVMPWVVYLEVTGGVWNFSRICCDVFVTLDVM
MCTASIWNLCAISIDRYTAVVMPVHYQHGTGQSSCRRVALMITAVVWLAFAVSCPLLFGLNADQNECIIANAPAFV
IYSSVVSFYLPFGVTVLVYARIYVVLKQRRRKAAGVPLREKKATQMVAIVLGAFIVCWLPFFLTHVLNTHC
QTCHVSPELYSATTWLGyvNSALNPVIYTTFNIEFRKAFKILSC
```

The prepared **d3d2_chimera.fasta** can be found in

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

If your **d3d2_chimera.fasta** file matches with the file that we prepared, the command line `diff d3d2_chimera.fasta ./demo/input_files/d3d2_chimera.fasta` should return nothing.

b. Template structures

Comparative modeling requires template structures to guide the target sequence folding. The D3/D2 chimera receptor would be classified as 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.

IDENTIFY TEMPLATES PDBs:

1. Go to Blastp <http://blast.ncbi.nlm.nih.gov/>
2. Copy sequence from **d3d2_chimera.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 D3/D2 chimera receptor include other bioamine receptors such as the dopamine, 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 the TM and extracellular loop regions. The amount of sequence identity of the templates will determine how many are necessary for. For this tutorial we have three templates available from the dopamine receptor family with high sequence identity. The three templates including the following:

- a. 3pbl: D3R
- b. 6cm4: D2R
- c. 5wiu: D4R

DOWNLOAD TEMPLATE PDBs:

1. Go to <http://www.rcsb.org/pdb/>.
2. Search for *3pbl*.
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 from **3pbl.pdb**
`gedit 3pbl.pdb -> manually delete lines by highlighting and clicking delete`
5. Save changes to **3pbl_isolated.pdb**
6. Repeat steps 1 - 5 for *6cm4* and *5wiu*.
The extra residues to be removed from **6cm4.pdb** include chain A residues numbered 1002-1161.
The extra residues to be removed from **5wiu.pdb** include chain A residues numbered 1001-1106.

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

```
~/rosetta_workshop/tutorials/rosetta_cm/demo/template_pdb/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>`
Make a list with the names of all the isolated pdbs. Remove the file extension (.pdb) of each of the isolated pdb names.

```
ls original_files/*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

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

Running these commands should yield the files: **3pbl_isolated_A.pdb**, **3pbl_isolated_A.fasta**, **6cm4_isolated_A.pdb**, **6cm4_isolated_A.fasta**, **5wiu_isolated_A.fasta**, **5wiu_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:

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 /demo/input_files/d3d2_chimera.fasta  
cat /demo/template_pdbs/*.fasta
```

3. In Output Format, select Pearson/FASTA
4. Download the alignment to a file called **d3d2_chimera_3pbl_6cm4_5wiu.aln** (Make sure the receptor sequences are on one line)
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/d3d2_chimera_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: d3d2_chimera.span

D3/D2 chimera receptor 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 **d3d2_chimera.fasta**

```
cat ~/rosetta_workshop/tutorials/rosetta_cm/demo/input_files/d3d2_chimera.fasta
```

3. *Save* the OCTOPUS *topology file* as **d3d2_chimera.octopus**
4. Convert the OCTOPUS file to a span file using the script:

```
~/rosetta_workshop/rosetta/main/source/src/apps/public/membrane_abinitio/octopus2span.pl \  
d3d2_chimera.octopus > d3d2_chimera.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 103 and 182. 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 at:

```
~/rosetta_workshop/tutorials/rosetta_cm/demo/input_files/d3d2_chimera.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.

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

```
cd ~/rosetta_workshop/tutorials/rosetta_cm/demo/alignment_files/
```

To manually convert your alignments into grishin format, 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.

The target sequence is threaded onto the template structures to generate a set of partial threads. A partial thread is randomly selected as the base model to create the global superposition in which all models are generated from. The user can also specify which partial thread is used for the base model of the global superposition as well. 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.

```
cd /demo/threaded_pdbs/
```

Run the script below to thread the target sequence over the template sequences. This script will read a list of template names from the file template.txt, and then run the command on each one in turn.

```
cat templates.txt | xargs -n1 -I@ \
~/rosetta_workshop/rosetta/main/source/bin/partial_thread.linuxgccrelease \
-in:file:fasta ../demo/input_files/d3d2_chimera.fasta \
-in:file:alignment ../demo/alignment_files/@.aln \
-in:file:template_pdb ../demo/template_pdbs/@_isolated_A.pdb
```

The output files should be named **3pbl_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/3pbl_out.pdb** (generated during threading)
- **threaded_pdbs/6cm4_out.pdb** (generated during threading)
- **threaded_pdbs/5wiu_out.pdb** (generated during threading)
- **input_files/d3d2_chimera.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/d3d2_chimera.span** (generated during setup)
- **input_files/d3d2_chimera.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 soluble protein versions of these weight files are also included in those files in case you wish to try RosettaCM with soluble proteins. For now, just copy the `_membrane.wts` files to your working directory.

The weight files are found in the `input_files` directory

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

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 `input_files` directory

```
~/rosetta_workshop/tutorials/rosetta_cm/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

```
~/rosetta_workshop/tutorials/rosetta_cm/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.

The following command launches the RosettaCM run. Be sure to run this from the parent directory
~/rosetta_workshop/tutorials/rosetta_cm/

```
~/rosetta_workshop/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 model. 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. A classic example of analyzing your results is to plot score versus rmsd to the target native and obtain a “folding funnel” where large rmsds correspond to poor scores and poor accuracy. The “folding funnel” represents convergence to an accurate model with respect to the score function being used.

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.