

#Protein-Protein Docking with Membrane Proteins

Bold text indicates that the files and/or information has been provided

Italicized text indicates that this material will NOT be conducted during the workshop. However instructions for these steps can be found in the appendix of the tutorial.

Fixed width text indicates the command which should be typed into your terminal exactly as displayed

Tutorial:

This tutorial presents a membrane protein-docking experiment. The β -2 Adrenergic Receptor binds to the $G\alpha/s$ subunit of heterotrimeric G proteins. It has been crystallized in the receptor-bound form in the presence of a nanobody and polyamine fragment (PDBID: 3SN6). The alpha subunit will be docked into the receptor and compared to the crystal structure. This type of experiment is useful for protocol optimization and development.

The beginning steps for structure preparation are covered in detail in several other tutorials. We will be using pre-created input structures to save time. This includes structures that have been cleaned and structures with chain breaks closed and. It is advised that you relax/repack your structures to remove clashes, but for this tutorial, it can be skipped.

1. Prepare the Input Template Proteins for Docking:

1. Download the PDB files.

Download 3SN6.pdb from the Protein Data Bank website www.rcsb.org. This file is also provided in the **input_files** directory.

1. Create a new directory in the protein_protein_docking directory called 'my_files' and move into that directory. We will work from that directory for the rest of the tutorial

```
cd ~/rosetta_workshop/tutorials/membrane/protein_protein_docking
mkdir my_files
cd my_files
```

2. Go to the rcsb.org website and type '3SN6' into the search bar.
3. Click 'Download Files' on the right side of the page.
4. Select 'PDB file (Text)'.
5. Save the PDB file in the 'my_files' directory.

2. Clean the PDBs.

The process to clean PDB files is covered in a separate section and should be followed when time is not limited; Cleaned PDB files are provided in the **input_files** directory as **3SN6_A_cleaned.pdb** and **3SN6_R_renumbered.pdb**.

1. Copy the cleaned PDB files into your 'my_files' directory. Make sure to include the '.' at the end of the command line.

```
cp ~/rosetta_workshop/tutorials/input_files/3SN6_A_cleaned.pdb .
cp ~/rosetta_workshop/tutorials/input_files/3SN6_R_renumbered.pdb .
```

3. Close the chain breaks found in the receptor and alpha subunit.

When preparing protein files for docking, chain breaks can cause unexpected behavior. As the chain break is far from the expected interface, the goal is to simply close the chain break within the secondary structure element and not to rigorously build this loop. The appendix provides an example protocol for fixing small chain breaks; here we will copy a file with the break already fixed to continue the tutorial.

- Note: full loop building protocols will be covered in the comparative modeling tutorial.

Copy the best scoring model (the first line of the sorted output) to model_a_1.pdb or model_r_1.pdb respectfully. Which file this is will depend on your particular run. For the provided examples in the output_files directory, the best scoring model is **3SN6_A_cleaned_occupancy_cor_0057.pdb** for the alpha subunit and for the receptor protein **3SN6_R_renumbered_occupancy_cor_0111.pdb**

```
cp 3SN6_A_cleaned_occupancy_cor_0057.pdb model_a_1.pdb
cp 3SN6_R_renumbered_occupancy_cor_0111.pdb model_r_1.pdb
```

4. Repack or relax the template structures.

Repacking is often necessary to remove small clashes, identified by the score function, that are present in the crystal structure. Certain amino acids within the interface are strictly conserved and their conformation has been shown to be critical for successful docking. RosettaScripts allows for fine control of these details using TaskOperations. For brevity, repacking has not been included in this protocol but is present in other tutorials.

*Pre-generating backbone conformational diversity prior to docking can be useful for obtaining well-docked models, particularly when the partners are crystallized separately. The Rosetta FastRelax algorithm can be accessed through RosettaScripts. Backbone conformational diversity will not be explored in this tutorial due to time constraints. However, example command lines along with XML scripts, input files, and options files are available in the **input_files** directory.*

5. Orient the alpha subunit in a proper starting conformation for the receptor.

Use available information on the participating interface residues to decrease the global conformational search space. This improves the efficiency of the docking process and the quality of the final model. For this tutorial, we will use the ideal starting conformation found in the **input_files** directory named **3SN6_r_a_aligned.pdb**.

1. Copy native structure into your working directory

```
cp ../input_files/3SN6_A_R_cleaned.pdb .
```

2. Open the files with pymol

```
pymol model_r_1.pdb model_a_1.pdb 3SN6_A_R_cleaned.pdb
```

3. Align the structures with the following pymol commands:

```
align model_r_1, 3SN6_A_R_cleaned
align model_a_1, 3SN6_A_R_cleaned
save model_r_a_aligned.pdb, model_r_1 + model_a_1
```

4. Renumber the PDB file from 1 to the end without restarting.

```
python2.7 ~/rosetta_workshop/Rosetta/tools/protein_tools/scripts/pdb_renumber.py \
--norestart model_r_a_aligned.pdb model_r_a_renumbered.pdb
```

6. Create the span file for the receptor.

When it is unknown what residues are in the membrane, it is best to use prediction software to identify transmembrane regions. To do this, we will use octopus.

1. Create a fasta file for your complex

```
~/rosetta_workshop/Rosetta/main/source/src/python/apps/public/pdb2fasta.py \
model_r_a_aligned.pdb
```

2. Go to <http://octopus.cbr.su.se/index.php>

Paste the fasta into the box and hit the button Submit OCTOPUS

Download the OCTOPUS topology file and save it to your input_files directory

3. Run the converter script to convert octopus topology files to span files

```
~/rosetta_workshop/Rosetta/main/source/src/apps/public/membrane_abinitio/octopus2span.pl \
octopus.topo
```

You will notice that it has a line, 482-502, which corresponds to the soluble region of the protein, delete this. Additionally, change the line 8 633 to 7 633, which tells rosetta that there are 7 transmembrane helices.

Save the output as model_r_a_aligned.spanfile

2. Perform Docking Utilizing the RosettaScripts Application

1. Prepare a RosettaScripts XML file for docking.

This file outlines a protocol that performs docking. It then further minimizes the interface. Familiarize yourself with the contents of this script.

1. Copy **docking_full.xml** from the **docking_files** directory. Go through the .xml script to understand the protocol.

```
cp ../docking_files/xml_files/docking_full.xml .
cat docking_full.xml
```

2. Prepare an options file for docking.

Copy the options file (**docking.options**) from the **docking_files** directory. Familiarize yourself with the options in the file.

```
cp ../docking_files/options_files/docking.options .
cat docking.options
```

3. Generate one (1) model using the full docking algorithm.

This will take approximately 15 minutes to complete.

```
~/rosetta_workshop/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
@dockering.options -database ~/rosetta_workshop/Rosetta/main/database/ \
-parser:protocol docking_full.xml -out:suffix _full -nstruct 1 \
-restore_pre_talaris_2013_behavior >& docking_full.log &
```

4. Copy the docking_minimize.xml from the docking_files directory.

Go through the .xml script to understand the protocol. The docking_minimize.xml file differs from the docking_full.xml only in the PROTOCOL section. The movers dock_low, srsc, and dock_high have been “turned off” by deleting the angle brackets at the beginning of these lines.

```
cp ../docking_files/xml_files/docking_minimize.xml .
cat docking_minimize.xml
```

5. Generate ten (10) models the minimization stage of the XML protocol.

This will take about 5 minutes.

```
~/rosetta_workshop/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
@dockering.options -database ~/rosetta_workshop/Rosetta/main/database/ \
-parser:protocol docking_minimize.xml -out:suffix _minimize -nstruct 10 \
-restore_pre_talaris_2013_behavior >& docking_minimize.log &
```

3. Characterize the models and analyze the data for docking funnels.

There are many movers and filters available in RosettaScripts for characterization of models. The InterfaceAnalyzerMover combines many of these movers and filters into a single mover. The RMSD filter is useful for benchmarking studies.

The native structure used in this step (3SN6_A_R_cleaned.pdb) has been cleaned as seen above. A complete structure is necessary for comparison to models. Missing density has been repaired through loop modeling or grafting of segments from 3SN6.pdb.

1. Characterize your models using the InterfaceAnalyzer mover in RosettaScripts and calculate the RMSD to the native crystal structure with the RMSD filter. (This takes about 5 minutes.)

```
cp ../docking_files/xml_files/docking_analysis.xml .
cp ../docking_files/options_files/docking_analysis.options .
cp ../input_files/3SN6_A_R_cleaned.pdb .
```

```
~/rosetta_workshop/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
@dockering_analysis.options -database ~/rosetta_workshop/Rosetta/main/database/ \
-restore_pre_talaris_2013_behavior
>& docking_analysis.log &
```

2. Sort the scores by binding energy (dG_seperated, the seventh column in the score file). Open the top scoring structures in pymol. Look for unsatisfied hydrogen bonds, regions of hydrophobic packing, voids in the interface, etc.

```
sort -nk 7 docking_analysis.csv
```

```
pymol 3SN6_A_R_cleaned.pdb model_r_a_renumbered_minimized_0001.pdb
```

3. Plot various scores against RMSD for total_score, dG_seperated, etc. to identify a binding funnel.

1. Open docking_analysis.csv as a spreadsheet and create a scatter plot.

```
ooffice2 docking_analysis.csv &
```

2. Or use the provided R script to make score vs. RMSD plots

```
cp ../input_files/sc_vs_rmsd.R .
```

```
./sc_vs_rmsd.R docking_analysis.csv total_score
```

```
./sc_vs_rmsd.R docking_analysis.csv dG_seperated
```

```
./sc_vs_rmsd.R docking_analysis.csv dG_seperated.dSASAx100
```

```
Gthumb *.png &
```

Appendix:

1. Cleaning the PDB:

1. Remove extra information from PDB file:

1. Remove chain B, G, N and the residues that correspond to the nanobody on the extracellular region of the receptor (keep the A chain and R chain - minus the extra nanobody residues 1-159)

2. Remove HETATOMs, REMARKs, SEQUEs, and all non-ATOM lines. Final file: **input_files/3SN6_R_cleaned.pdb** and **3SN6_A_cleaned.pdb**

```
gedit 3SN6_R.pdb 3SN6_A.pdb
```

2. Renumber chain R starting from residue 1:

```
python2.7 ~/rosetta_workshop/tutorial/protein-protein_docking/membrane/scripts/pdb_renumber.py \
3SN6_R_cleaned.pdb 3SN6_R_renumbered.pdb
```

Final file: **input_files/3SN6_R_renumbered.pdb**

2. Fixing Chain Breaks: Loop Modeling:

1. Identify chain breaks within the structures:

1. Open 3SN6_A_cleaned.pdb and 3SN6_R_renumbered.pdb in PyMol to identify chain breaks within the two structures:

```
pymol 3SN6_A_cleaned.pdb 3SN6_R_renumbered.pdb
```

Type the following command into PyMol to visualize secondary structural elements:

```
as cartoon
```

2. Create .loops files for both the alpha and receptor proteins based on the missing loop regions

1. Create .loops file for the receptor to fix break between residues 146-147, and 207-208. Final file: **input_files/loop_model/chainbreak_3sn6_fix.loops**

```
LOOP 144 149 0 0 1
```

```
LOOP 205 210 0 0 1
```

2. Create .loops file for the alpha subunit to fix breaks between residues 50-51, 166-167, and 217-218. Final file: **input_files/loop_model/chainbreak_alpha_fix.loops**

```
LOOP 49 53 0 0 1
```

```
LOOP 164 171 0 0 1
```

```
LOOP 213 220 0 0 1
```

3. Create .options for each protein to rebuild the loops

1. Create .options file for the receptor

```
gedit chainbreak_3sn6_fix.options
```

Copy the following text into the blank file you just created:

```
-s 3sn6_R_renumbered.pdb
-loops:loop_file chainbreak_3sn6_fix.loops
-loops:fast
-loops:max_kic_build_attempts 200
-in:file:fullatom
-out:file:fullatom
-ex1
-ex2
-use_input_sc
-nstruct 1
-loops:remodel perturb_kic
-loops:refine refine_kic
-out:file:scorefile chainbreak_fix.fasc
-corrections:score:use_bicubic_interpolation true
```
2. Create .options file for the alpha subunit

```
gedit chainbreak_alpha_fix.options
```

Copy the following text into the blank file you just created:

```
-s 3sn6_A_cleaned.pdb
-loops:loop_file chainbreak_alpha_fix.loops
-loops:fast
-loops:max_kic_build_attempts 200
-in:file:fullatom
-out:file:fullatom
-ex1
-ex2
-use_input_sc
-nstruct 1
-loops:remodel perturb_kic
-loops:refine refine_kic
-out:file:scorefile chainbreak_fix.fasc
-corrections:score:use_bicubic_interpolation true
```
4. Run the loopmodel application for both template structures:
(For teaching purposes, we are only creating 1 model. 500 output models are provided for later analysis.)

```
~/rosetta_workshop/rosetta_source/bin/loopmodel.default.linuxgccrelease \
@chainbreak_3sn6_fix.options -database ~/rosetta_workshop/rosetta_database/ -nstruct 1

~/rosetta_workshop/rosetta_source/bin/loopmodel.default.linuxgccrelease \
@chainbreak_alpha_fix.options -database ~/rosetta_workshop/rosetta_database/ -nstruct 1
```
5. Sort the models for the best scoring (lowest energy) structures:

```
grep total_energy 3SN6_A_cleaned_0* | sort -nk 2 | head > chainbreak_alpha_sort.txt

grep total_energy 3SN6_R_renumbered_0* | sort -nk 2 | head > chainbreak_r_sort.txt
```
6. Visually inspect the top ten scoring models in pymol

```
pymol 3SN6_R_renumbered_occupancy_cor_0111.pdb \
3SN6_R_renumbered_occupancy_cor_0065.pdb \
3SN6_R_renumbered_occupancy_cor_0276.pdb \
3SN6_R_renumbered_occupancy_cor_0500.pdb \
3SN6_R_renumbered_occupancy_cor_0494.pdb \
3SN6_R_renumbered_occupancy_cor_0387.pdb \
```

```
3SN6_R_renumbered_occupancy_cor_0386.pdb \  
3SN6_R_renumbered_occupancy_cor_0282.pdb \  
3SN6_R_renumbered_occupancy_cor_0229.pdb \  
3SN6_R_renumbered_occupancy_cor_0028.pdb
```

7. Rename the top models for each template. (These are the models which were used for the main part of the tutorial.)

```
mv 3SN6_R_renumbered_occupancy_cor_0111.pdb model_r_1.pdb  
mv 3SN6_A_cleaned_occupancy_cor_0057.pdb model_a_1.pdb
```