

Protein-Protein Docking

Bold text means that these files and/or this information is provided.

Italicized text means that this material will NOT be conducted during the workshop

fixed with text means you should type the command into your terminal

If you want to try making files that already exist (e.g., input files), write them to a different directory! (`mkdir my_dir`)

Tutorial

This tutorial presents a cross-docking benchmark experiment. Antibody CR6261 binds to multiple sub- types of influenza antigen hemagglutinin (HA). It has been crystallized with H1 and H5 HA sub-types. Antibody from one crystal structure will be docked to the antigen from the other crystal structure. This type of experiment is useful for protocol optimization and development.

1. Prepare the input template for docking

1. Download the PDB files. The 3gbn.pdb and 3gbm.pdb files are provided in the input_files directory.

1. Download the starting structure from the Protein Data Bank.

1. Go to rcsb.org and type '3gbn' in the search bar.
 2. Create a directory in the protein-protein_docking/soluble/ directory called my_files and switch to that directory. We will work from this directory for the rest of the tutorial.

```
cd tutorials/protein-protein_docking/soluble/  
mkdir my_files  
cd my_files
```

3. Click on 'Download Files' on the right side of the page, then 'PDB File (Text)'.
4. Save the PDB file in the my_files directory.
5. Repeat for '3gbm'

2. *Clean the PDBs.* Cleaned PDBs are provided in the input_files directory as **3gbn_Ab.pdb** and **3gbm_HA.pdb**.

```
rosetta-3.5/rosetta_tools/protein_tools/scripts/pdb_renumber.py \  
--norestart 3gbn_clean.pdb 3gbn_Ab.pdb
```

```
rosetta-3.5/rosetta_tools/protein_tools/scripts/pdb_renumber.py \  
--norestart 3gbm_clean.pdb 3gbm_HA.pdb
```

3. Close the chain break between Ser-127 and Val-128 in chain L of the antibody.

Chain breaks can cause unexpected behavior during docking. Because this chain break is small and away from the expected interface, it can be quickly and easily fixed. The goal is simply to close the chain break within the secondary structure element and not to rigorously build this loop. Build ten models (a minimal computational effort) and pick one with a good score and a good representative structure.

1. Open 3gbn_Ab.pdb with PyMol to identify the chain break between residues 127 and 128 of chain L.

```
pymol 3gbn_Ab.pdb  
type as cartoon or as ribbon  
type show lines, resi 125-130
```

2. Prepare a loops file for closing the chain break. Use a text editor such as gedit. Several amino acids must be mobile for the loop to close successfully. Select several residues on each side of the chain break.

```
gedit chainbreak_fix.loops
```

1. Type L00P 125 130 0 0 1, save the file, and close gedit.

2. Copy the prepared options file from the input_files directory


```
cp ../input_files/chainbreak_fix.options .
```
 3. Run the Rosetta loopmodel application to close the loop.


```
rosetta-3.5/rosetta_source/bin/loopmodel.default.linuxgccrelease @chainbreak_fix.options \
-database rosetta-3.5/rosetta_database/ -nstruct 10 >& chainbreak_fix.log &
```

```
pymol 3gbn_Ab*pdb &
sort -nk 2 chainbreak_fix.fasc
```
 4. Copy the best scoring model (3gbn_Ab_0010.pdb in the example output_files directory) to 3gbn_Ab_fixed.pdb. cp 3gbn_Ab_0010.pdb 3gbn_Ab_fixed.pdb
4. Repack or relax the template structures.

Repacking is often necessary to remove small clashes identified by the score function as present in the crystal structure. Certain amino acids within the HA interface are strictly conserved and their conformation has been shown to be critical for success in docking. RosettaScripts allows for fine control of these details using TaskOperations.

1. Copy the XML scripts and options file for repacking from the input_files directory.


```
cp ../input_files/repack.xml .
cp ../input_files/repack_HA.xml .
cp ../input_files/repack.options .
```
2. Familiarize yourself with repack.xml and repack_HA.xml. Notice that repack_HA.xml is a modified version of repack.xml, representative of the versatility of RosettaScripts.
3. Run the XML script with the rosetta_scripts application.


```
~/rosetta_workshop/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease\
@repack.options -s 3gbm_HA.pdb -parser:protocol repack_HA.xml\
-database ~/rosetta_workshop/Rosetta/main/database/ >& repack_HA.log &
```

```
~/rosetta_workshop/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease\
@repack.options -s 3gbn_Ab_fixed.pdb -parser:protocol repack.xml\
-database ~/rosetta_workshop/Rosetta/main/database/ >& relax_Ab.log &
```
4. When the repacking runs are done (in about 3-4 minutes), copy the best scoring HA model to 3gbm_HA_repack.pdb and the best scoring antibody model to 3gbn_Ab_repack.pdb. (For brevity, we only generated a single structure. For actual production runs, we recommend generating a number of output structures, by adding something like “-nstruct 25” to the commandline. – For the example outputs in the output_files/ directory, the lowest energy structures are 3gbm_HA_0011.pdb and 3gbn_Ab_fixed_0005.pdb.)


```
cp 3gbm_HA_0011.pdb 3gbm_HA_repacked.pdb
cp 3gbn_Ab_fixed_0011.pdb 3gbn_Ab_repacked.pdb
```

It can also be useful to pre-generate backbone conformational diversity prior to docking particularly when the partners are crystallized separately. The Rosetta FastRelax algorithm can be accessed through RosettaScripts. XML scripts, input files, options files and a command are available in the input_files directory. Backbone conformational diversity will not be explored in this tutorial due to time constraints.

5. Orient the antibody in a proper starting conformation.

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. In this benchmark case we will use the ideal starting conformation.

 1. Align the structures with pymol.


```
cp ../input_files/3gbm_native.pdb .
pymol 3gbm_native.pdb 3gbm_HA_repacked.pdb 3gbn_Ab_repacked.pdb
```

 1. Type ‘align 3gbn_Ab_repacked, 3gbm_native’
 2. Type ‘save 3gbm_HA_3gbn_Ab.pdb, 3gbm_HA_repacked + 3gbn_Ab_repacked’
 2. Renumber the pdb from 1 to the end without restarting.

```
python2.7 ~/rosetta_workshop/Rosetta/tools/protein_tools/scripts/pdb_renumber.py \
--norestart 3gbm_HA_3gbn_Ab.pdb 3gbm_HA_3gbn_Ab.pdb
```

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 the script.

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

```
cp ../input_files/docking_full.xml .
cat docking_full.xml
```

2. Prepare an options file for docking.

1. Copy the options file (docking.options) from the input_files directory. Familiarize yourself with the options in the file.

```
cp ../input_files/docking.options .
cat docking.options
```

3. Generate fifty models using the full docking algorithm.

```
~/rosetta_workshop/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
@docking.options -database ~/rosetta_workshop/Rosetta/main/database/ \
-parser:protocol docking_full.xml -out:suffix _full -nstruct 50 >& docking_full.log &
```

4. Generate twenty-five models with only the minimization stage of the xml protocol.

1. Copy docking_minimize.xml from the input_files directory.

The docking_minimize.xml file differs from docking_full.xml only in the PROTOCOL section. The movers dock_low, srsc, and dock_high have been turned off by deleting the angle bracket at the beginning of these lines.

```
cp ../input_files/docking_minimize.xml . cat docking_minimize.xml
```

2. Generate ten models using only the refinement stage of docking.

```
~/rosetta_workshop/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
@docking.options -database ~/rosetta_workshop/Rosetta/main/database/ \
-parser:protocol docking_minimize.xml -out:suffix _minimize -nstruct 10 >& \
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 (3gbm_native.pdb) has been cleaned as above. A complete structure is necessary for comparison to models. Missing density has been repaired through loop modeling or grafting of segments from 3gbn.pdb.

1. Characterize your models using the InterfaceAnalyzer mover in RosettaScripts and calculate the RMSD to the native crystal structure with the RMSD filter.

```
cp ../input_files/docking_analysis.xml .
cp ../input_files/docking_analysis.options .
cp ../input_files/3gbm_native.pdb .
~/rosetta_workshop/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
@docking_analysis.options -database ~/rosetta_workshop/Rosetta/main/database/ >& \
docking_analysis.log &
```

```
sort -nk 7 docking_analysis.csv
pymol 3gbm_native.pdb 3gbm_HA_3gbn_Ab_full_0001.pdb
```

2. Plot various scores against rmsd for total_score, dG_separated, 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_separated
./sc_vs_rmsd.R docking_analysis.csv dG_separated.dSASAx100
gthumb *png &
```

## Appendix

### 1. *Cleaning the PDBs.*

1. Remove extraneous information and atoms from the PDB file.
  1. Open the PDB in a molecular graphics viewer.

```
pymol 3gbn.pdb 3gbm.pdb &
```
  2. Open the file for text editing in a text editor.

```
gedit 3gbn.pdb 3gbm.pdb &
```
  3. Delete header lines from the pdb. These are lines such as HEADER, REMARK, and other non-ATOM lines.
  4. Delete HA chains (A, B) from 3gbn.pdb. Delete antibody chains and chains of the asymmetric complex from 3gbm.pdb (C, D, H, L, I, M).
  5. Delete waters, sugars, and other non-amino acid (HETATM) residues in the text file.

Inclusion of HETATMs in docking can be useful if they are at the interface. However, they require special care, such as generation of a params file or non-canonical amino acid, and will not be modeled in this tutorial.

### 2. Delete unresolved regions of the antibody.

Domains which do not impact the structure at the interface are often removed to improve the speed of model generation. For this reason, and because the antibody constant region is poorly resolved, we will model only the variable domain of the antibody.

1. Click the 'S' in the lower right corner of PyMol to bring up the sequence for 3gbn.pdb.
  2. Type 'as cartoon' to show the structure as a cartoon.
  3. Look for chain breaks in the structure or regions that are clearly incomplete (i.e., incomplete domains).
  4. Click on residues within unresolved regions and find the highlighted region in the sequence at the top of the screen.
  5. Delete residues 113-188 of the H chain in the text file.
  6. Delete residues 106-195 of the L chain in the text file.
  7. Save the text files as 3gbn\_clean.pdb and 3gbm\_clean.pdb. Close PyMol and gedit.
3. Renumber each pdb from 1 to the end, without restarting at the beginning of each chain.

1. Write the output to 3gbn\_Ab.pdb and 3gbm\_HA.pdb as appropriate.

```
python2.7 ~/rosetta_workshop/Rosetta/tools/protein_tools/scripts/pdb_renumber.py\
--norestart 3gbn_clean.pdb 3gbn_Ab.pdb

python2.7 ~/rosetta_workshop/Rosetta/tools/protein_tools/scripts/pdb_renumber.py\
--norestart 3gbm_clean.pdb 3gbm_HA.pdb
```

### 2. *Closing the chain break between Ser-127 and Val-128 in chain L of the antibody.*

Chain breaks can cause unexpected behavior during docking. Because this chain break is small and away from the expected interface, it can be quickly and easily fixed. The goal is simply to close the chain break within the secondary structure element and not to rigorously build this loop. Build ten models (a minimal computational effort) and pick one with a good score and a good representative structure.

1. Open 3gbn\_Ab.pdb with PyMol to identify the chain break between residues 127 and 128 of chain L.

```
pymol 3gbn_Ab.pdb
```

```
type as cartoon or as ribbon
```

```
type show lines, resi 125-130
```

2. Prepare a loops file for closing the chain break. Use a text editor such as gedit. Several amino acids must be mobile for the loop to close successfully. Select several residues on each side of the chain break.

```
gedit chainbreak_fix.loops
```

1. Type LOOP 125 130 0 0 1, save the file, and close gedit.

2. Copy the prepared options file from the input\_files directory

```
cp ../input_files/chainbreak_fix.options .
```

3. Run the Rosetta loopmodel application to close the loop.

```
~/rosetta_workshop/Rosetta/main/source/bin/loopmodel.default.linuxgccrelease\
@chainbreak_fix.options -database ~/rosetta_workshop/Rosetta/main/database/
-nstruct 10 >& chainbreak_fix.log
```

4. When the application is done running (it will take about 15 minutes), examine the output models.

```
pymol 3gbn_Ab*.pdb &
```

```
sort -nk 2 chainbreak_fix.fasc
```