Protein-Protein Docking Tutorial

This tutorial presents a cross-docking benchmark experiment. The antibody CR6261 binds to multiple sub-types of influenza antigen hemagglutinin (HA). It has been crystallized with both H1 and H5 HA sub-types. In this tutorial the antibody from one crystal structure (3GBN) will be docked to the antigen from the other crystal structure (3GBM). This type of experiment is useful for protocol optimization and development.

Initial set-up

1. Create a directory in the protein-protein_docking directory called my_files and switch to that directory. We will work from this directory for the rest of the tutorial.

   ```
   mkdir my_files
   cd my_files
   ```

2. Download the PDB files. The 3GBN.pdb and 3GBM.pdb files are also provided in the input_files directory.

   1. Download 3GBN from the Protein Data Bank.
      1. Go to rcsb.org and type '3gbn' in the search bar.
      2. Click on 'Download Files' on the right side of the page, then 'PDB Format'.
      3. Save the PDB file in the my_files directory as `3GBN.pdb`.
   2. Repeat for '3GBM'.
   3. Open the two structures in pymol, and check the number of chains for each structures. For structure 3GBM we have two complexes in the same crystal unit: ABHL and CDIM. The chains AB/CD represents the head domain and the stalk domain of the hemagglutinin protein, and HL/IM the CR6261 heavy and light chains. For structure 3GBN we have the single complex ABHL. In the next step we will extract the chain of interest for this cross-docking experiment: chains AB for 3GBM and chains HL for 3GBN.

      ```
      pymol 3GBM.pdb 3GBN.pdb
      ```

      1. Type 'set seq_view, 1'.
      2. Type 'align 3GBM, 3GBN'.

Structures pre-processing

To extract the chains of interest (AB for 3GBM, HL for 3GBN), remove all hetero-atoms including water, and renumber the residues from 1 to the end of the complex we will use the command clean_pdb.py. The pdb_renumber.py script is used in this protocol only to rename the pdb file, but it also can be used to renumber different chains (see next steps).

1. We want the hemagglutinin (chains A and B) from 3GBM

   ```
   python2.7 /PATH/TO/ROSETTA/tools/protein_tools/scripts/clean_pdb.py 3GBM AB

   python2.7 /PATH/TO/ROSETTA/tools/protein_tools/scripts/pdb_renumber.py \
     --norestart 3GBM_AB.pdb 3gbm_HA.pdb
   ```

2. We want the antibody (chains H and L) from 3GBN. (Note that chains A and B are not the antibody "Ab").

   ```
   python2.7 /PATH/TO/ROSETTA/tools/protein_tools/scripts/clean_pdb.py 3GBN HL
   ```

3. We will only need the variable domain which is actually involved with binding HA. The crystal structure also contains a partially resolved portion of the constant domain. You should manually edit the PDB file with a text editor to remove the unnecessary portions.

1. Use PyMol to determine which residues you need to delete: it should be residues 121-160 of the heavy chain (chain H) and residues 268-311 of the light chain (chain L) in the cleaned structure (3GNB_HL.pdb).

   ```
   pymol 3GBN_HL.pdb
   ```

   1. Type 'sele to_delete, resi 121-160+268-311'
   2. Type 'remove to_delete'
   3. Type 'save 3GBN_trim.pdb, 3GBN_HL' and close the pymol session.
2. Rename the file.

   ```
   python2.7 /PATH/TO/ROSETTA/tools/protein_tools/scripts/pdb_renumber.py \
     --norestart 3GBN_trim.pdb 3gbn_Ab.pdb
   ```

Close chain break through loop modeling

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. Identify the chain break

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

      ```
      pymol 3gbn_Ab.pdb
      ```

      1. Type '`as cartoon`' or '`as ribbon`'
      2. Type '`color red, 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. The loop file begins with the word "LOOP", and the second and third columns identify the start and end residues for loop modeling. The fourth column allows the user to define a cut point residue, but we will set it default = 0 to allow the code to choose the cut point. The fifth column represent the skip rate and we will set it to 0 to never skip the modeling of the loop. The last column indicates if the loop should be re-built from the native loop conformation (set as 0), or from scratch (set as 1). In our case we will re-build the loop from scratch without using information from the native conformation.

      ```
      gedit chainbreak_fix.loops
      ```

      1. Type '`LOOP 125 130 0 0 1`'.
      2. Save the file, and close gedit
2. Close the chain break

   1. Familiarize yourself with the options file

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

   2. Run the Rosetta loopmodel application to close the loop (~10 minutes).

      ```
      /PATH/TO/ROSETTA/main/source/bin/loopmodel.default.linuxgccrelease \
        @chainbreak_fix.options -nstruct 10 >& chainbreak_fix.log
      ```
3. Analyze the outputs and select the best scoring model

1. Inspect the generated models and identify the lowest scoring models by comparing the total energy values (column 2 in the chainbreak_fix.fasc scorefile). The lowest score model among the 10 pre-generated structures available in the input_files folder is 3gbn_Ab_0002, however your lowest score model might be a different one.

   ```
   pymol 3gbn_Ab*pdb

   cat chainbreak_fix.fasc | awk '{print $NF, $2}' | sort -nk 2
   ```

2. Copy the best scoring model (`3gbn_Ab_0002.pdb` in the example output_files directory) to `3gbn_Ab_fixed.pdb`.

   ```
   cp ../output_files/3gbn_Ab_0002.pdb 3gbn_Ab_fixed.pdb
   ```

Refine the template structure

Structure refinement is essential for any post-processing step in Rosetta to remove clashesthat might be present in the crystal structure. Refinement can be performed at multiple levels: repack (side-chains only), minimization (side-chains and backbone) or relax, which involve multiple cycles of repack and minimization. Due to time constraints, in this tutorial we will perform only a side-chain repacking. Certain amino acids within the HA interface are strictly conserved and their conformation has been shown to be critical for success in docking. We will maintain the native conformation of those critical residues and repack everything else. 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 (line 8) is a modified version of repack.xml, representative of the versatility of RosettaScripts.

3. Run the XML script with the rosetta_scripts application (both should take less than 5 min).

   ```
   /PATH/TO/ROSETTA/main/source/bin/rosetta_scripts.default.linuxgccrelease \
     @repack.options -s 3gbm_HA.pdb -parser:protocol repack_HA.xml \
     -out:file:scorefile repack_HA.fasc >& repack_HA.log

   /PATH/TO/ROSETTA/main/source/bin/rosetta_scripts.default.linuxgccrelease \
     @repack.options -s 3gbn_Ab_fixed.pdb -parser:protocol repack.xml \
     -out:file:scorefile repack_Ab.fasc >& repack_Ab.log
   ```

4. When the repacking runs are done, 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_0018.pdb` and `3gbn_Ab_fixed_0022.pdb`).

   ```
   cp ../output_files/3gbm_HA_0018.pdb 3gbm_HA_repacked.pdb
   cp ../output_files/3gbn_Ab_fixed_0022.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. However, backbone conformational diversity will not be explored in this tutorial due to time constraints.

Protein-protein docking

We will 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. Orient the antibody in a proper 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 /PATH/TO/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. Prepare the RosettaScripts XML and the options files 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 .
        gedit docking_full.xml
        ```

    2. Prepare 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 .
        gedit docking.options
        ```

    3. Generate ten models using the full docking algorithm. You might obtain less than 10 output models, don't be worried. This will take ~5 hours - open a new terminal tab and move on to the next steps while this is running.

        ```
        /PATH/TO/ROSETTA/main/source/bin/rosetta_scripts.default.linuxgccrelease \
          @docking.options -parser:protocol docking_full.xml -out:suffix _full \
          -nstruct 10 >& docking_full.log
        ```

    4. To have a good comparison for the native structure, we want to minimize the experimental structure with the Rosetta energy function. To do this, we run a similar protocol, but skipping the coarse and fine resolution search stages, keeping only the minimization stage. This provides us with a like-to-like comparison of the native structure.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 .
        gedit docking_minimize.xml
        ```

    5. Generate two models using only the minimization refinement stage of docking. (This will take ~1 hour - open a new terminal tab and move on to the next steps while this is running.)

        ```
        /PATH/TO/ROSETTA/main/source/bin/rosetta_scripts.default.linuxgccrelease \
          @docking.options -parser:protocol docking_minimize.xml -out:suffix _minimize \
          -nstruct 2 >& docking_minimize.log
        ```

Analysis of docking results

We will 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.

Since your docking run is not finished yet, you can try out this step with pre-generated results.

1. Make a new directory and copy the files `3gbm_HA_3gbn_Ab_full_00\*.pdb` and `3gbm_HA_3gbn_Ab_minimize_00\*.pdb` from the output_files/ directory into this new directory.

   ```
   mkdir Analysis_pregenerated
   cd Analysis_pregenerated
   cp ../../output_files/3gbm_HA_3gbn_Ab_full_*.pdb .
   cp ../../output_files/3gbm_HA_3gbn_Ab_minimize_*.pdb .
   cp ../../input_files/3gbm_native.pdb .
   ```

2. Also copy the xml and option files for the docking analysis and get familiar with them:

   ```
   cp ../../input_files/docking_analysis.xml .
   gedit docking_analysis.xml

   cp ../../input_files/docking_analysis.options .
   gedit docking_analysis.options
   ```

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

   ```
   /PATH/TO/ROSETTA/main/source/bin/rosetta_scripts.default.linuxgccrelease \
     @docking_analysis.options -in:file:s *.pdb >& docking_analysis.log
   ```

4. Analyze the results according to total score, dG_deparated and RMSD (columns 2, 7 and 37, respectively)

   ```
   cat docking_analysis.fasc | awk '{print $NF, $2, $7, $37}' | sort -nk 3
   ```

5. Analyze the best docking model (`3gbm_HA_3gbn_Ab_full_0035.pdb`) and the worst one (`3gbm_HA_3gbn_Ab_full_0047.pdb` in comparison to the native conformation

   ```
   pymol 3gbm_native.pdb 3gbm_HA_3gbn_Ab_full_0035.pdb 3gbm_HA_3gbn_Ab_full_0047.pdb
   ```

   1. Type 'align 3gbm_HA_3gbn_Ab_full_0035, 3gbm_native'
   2. Type 'align 3gbm_HA_3gbn_Ab_full_0047, 3gbm_native'

   (If you don't use the pre-generated results, your low energy file will likely be something other than `3gbm_HA_3gbn_Ab_full_0035.pdb`)

6. Plot various scores against rmsd for total_score, dG_separated, etc., to identify a binding funnel.

   ```
       cp ../../input_files/sc_vs_rmsd.R .
       Rscript ./sc_vs_rmsd.R docking_analysis.fasc total_score
       Rscript ./sc_vs_rmsd.R docking_analysis.fasc dG_separated
       eog *png &
   ```