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 width 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. 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. 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 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 File (Text)’.
         3. Save the PDB file in the my_files directory as `3GBN.pdb`.
      2. Repeat for ‘3GBM’
   2. Clean the PDBs.
      1. We want the hemagglutinin (chains A and B) from 3GBM

         ```
         python2.7 ~/rosetta_workshop/rosetta/tools/protein_tools/scripts/clean_pdb.py 3GBM AB

         python2.7 ~/rosetta_workshop/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”). We 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. You should be able to see them in a structure viewer. 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.

         ```
         python2.7 ~/rosetta_workshop/rosetta/tools/protein_tools/scripts/clean_pdb.py 3GBN HL

         cp 3GBN_HL.pdb 3GBN_trim.pdb
         pymol 3GBN_trim.pdb
         gedit 3GBN_trim.pdb

         python2.7 ~/rosetta_workshop/rosetta/tools/protein_tools/scripts/pdb_renumber.py \
           --norestart 3GBN_trim.pdb 3gbn_Ab.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 '`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 -nstruct 10 >& chainbreak_fix.log &

      pymol 3gbn_Ab*pdb &
      sort -nk 2 chainbreak_fix.fasc
      ```

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

      ```
      cp 3gbn_Ab_0002.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 >& repack_HA.log &

      ~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
        @repack.options -s 3gbn_Ab_fixed.pdb -parser:protocol repack.xml >& repack_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_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.

   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
         ```

3. 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 ten models using the full docking algorithm. (This will likely take a while - move on to the next steps while this is running.)

         ```
         ~/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 10 >& docking_full.log &
         ```

      4. To have a good comparison for the native structure, we want to minimize the experimental structure into 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.

         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 two models using only the minimization refinement stage of docking.

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

4. 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.

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

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 -in:file:s *full*pdb *minimize*pdb >& docking_analysis.log &
   ```

   (**If you use the pre-generated results from the output_files/ directory the line should read** `../output_files/*full*pdb ../output_files/*minimize*pdb`)

   ```
   sort -nk 7 docking_analysis.csv
   pymol 3gbm_native.pdb 3gbm_HA_3gbn_Ab_full_0035.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. (Check the boxes for space separation and merging delimiters.)

      ```
      ooffice docking_analysis.csv &
      ```

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

      ```
      cp ../input_files/sc_vs_rmsd.R .
      Rscript ./sc_vs_rmsd.R docking_analysis.csv total_score
      Rscript ./sc_vs_rmsd.R docking_analysis.csv dG_separated
      Rscript ./sc_vs_rmsd.R docking_analysis.csv dG_separated.dSASAx100
      gthumb *png &
      ```