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.

Tutorial Files Organization- The tutorial is organized into four folders that correspond with the four steps to a complete docking experiment. In each folder, you'll find the necessary Rosetta XML and option files provided for you. Each folder also contains an ANSWERS directory that includes an example of all files you'll generate during this tutorial. For time consuming steps, the ANSWERS folder also contains pre-generated models and scorefiles.

1. PDB-file-prep: This is where we will download and prepare the PDB files for Rosetta. We'll also use the Rosetta loopmodel application to close a chain break in the protein.
2. repack: This is where we will energetically minimize the crystal structures and generate some conformational diveristy for docking.
3. docking: This is where we will perform protein-protein docking. We'll be doing an example of full docking as well as high resolution focused docking.
4. analysis: This is where we will analyze the docking output for an energy funnel.

PDB file preparation and chainbreak closure

1. Change into the PDB-file-prep directory in the protein-protein_docking directory. We'll begin preparing PDB files in this folder:

   ```
   cd ~/rosetta_workshop/protein-protein_docking/PDB-file-prep
   ```

2. Prepare the input template for docking
   1. Download the PDB files. **The 3GBN.pdb and 3GBM.pdb** files are also provided in the ANSWERS 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'
   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. The hemagglutinin 3gbm_HA.pdb is now ready for the next step of repacking. Copy it to the repacking folder and then move on to cleaning the antibody.

      ```
      cp 3gbm_HA.pdb ../repack/3gbm_HA.pdb
      ```

   3. 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. The options file is provided for you as **chainbreak_fix.options**

      ```
          gedit chainbreak_fix.loops
      ```

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

   3. Run the Rosetta loopmodel application to close the loop. This step will take approximately 20 minutes. You might want to grab a snack and have a chat. After completion, you'll be able to look at the structures in pymol. The sort command will allow you to easily see what the best scoring model is (you can scroll through the score terms with the left and right arrow keys, press q to exit)

      ```
      ~/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 | less -S
      ```

   4. Copy your best scoring model to the repacking folder for the next step as 3gbn_Ab_fixed.pdb.

      ```
      cp PDB-OF-YOUR-BEST-SCORING-MODEL ../repack/3gbn_Ab_fixed.pdb
      ```

5. USE THIS COMMAND ONLY IF YOU DID NOT MAKE YOUR OWN MODELS: The best scoring model in the ANSWERS directory is **3gbn_Ab_0010.pdb**

   ```
   cp ANSWERS/3gbn_Ab_0010.pdb ../repack/3gbn_Ab_fixed.pdb
   ```

Repacking

Now we move on to repacking (also called relaxing) the crystal 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. The necessary XML scripts and options files are provided for you.

1. Change into the repack directory to begin

   ```
   cd ../repack
   ```

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. This should take about 3-4 minutes.

   ```
   ~/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. For the sake of brevity, we only generated a single structure. For actual production runs, we recommend generating additional output structures by adding "-nstruct 25" to the commandline. We have provided the full complement of models in the ANSWERS directory. We'll use the best scoring pre-generated models of the hemagglutinin (3gbm_HA_0023.pdb) and the antibody (3gbn_Ab_fixed_0005.pdb) for the next steps. Copy these files into your working directory.

   ```
    cp ANSWERS/3gbm_HA_0023.pdb 3gbm_HA_repacked.pdb
    cp ANSWERS/3gbn_Ab_fixed_0021.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 also be accessed through RosettaScripts. 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, which is provided for you as **3gbm_native.pdb.**

   1. Align the structures with pymol.

      ```
      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. This is the file we will need to do docking. Move it over to the docking directory before switching directories

   ```
   cp 3gbm_HA_3gbn_Ab.pdb ../docking/
   cd ../docking
   ```

Docking

Perform docking utilizing the RosettaScripts application.

1. The RosettaScripts XML file and options file for docking are providied for you as **docking_full.xml and docking.options.** The provided protocol performs docking and then minimizes the interface. Familiarize yourself with the contents of the files.

2. Generate a model using the full docking algorithm. This process will take roughly 20 minutes. For the sake of time, we've also pre-generated 50 full docking models for you in the ANSWERS directory. We outputted these in the silent file format by adding the option -out:file:silent docking_full.silent . The corresponding scorefile is **docking_full.sc** and the log file is **docking_full.log**. Feel free to take a look at the files.

   ```
   ~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
     @docking.options -parser:protocol docking_full.xml -out:suffix _full_user \
     -out:file:scorefile docking_full.sc -nstruct 1 >& docking_full.log &
   ```

3. 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. The necessary XML file is provided for you as **docking_minimize.xml**. Note that the options file is the same as the full docking step.

   1. 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. Be sure to understand the differences.

      ```
      cat docking_minimize.xml
      ```

   2. Generate a model using only the minimization refinement stage of docking. This process will take roughly 15 minutes. Again, we've pre-generated 10 minimization only models for you in the ANSWERS directory. We outputted these in the silent file format by adding the option -out:file:silent docking_minimize.silent . The corresponding scorefile is **docking_minimize.sc** and the log file is **docking_minimize.log**. Feel free to take a look at the files.

      ```
      ~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
        @docking.options -parser:protocol docking_minimize.xml -out:suffix _minimize_user \
        -out:file:scorefile docking_minimize.sc -nstruct 1 >& docking_minimize.log &
      ```

4. As you may recall, silent files are Rosetta specific and cannot be opened directly by other software packages (ex: Pymol). In order to look at PDB models contained within a silent file, you'll need to extract them first. This command should extract all 50 full docking PDBs and 10 minimize PDBs into your directory. These PDB files will not have the _user suffix to avoid overwriting the single example model you generated.

```
~/rosetta_workshop/rosetta/main/source/bin/extract_pdbs.default.linuxgccrelease \
  -in:file:silent ANSWERS/*.silent
```

5. Feel free to look at as many or as few of these models as you wish. You can identify the best models with the sort command and then scrolling with the arrow keys. The best full docking model should be 3gbm_HA_3gbn_Ab_full_0002.pdb and the best minimization only model should be 3gbm_HA_3gbn_Ab_minimize_0005.pdb.

   ```
   sort -nk 2 ANSWERS/docking_full.sc | less -S
   sort -nk 2 ANSWERS/docking_minimize.sc | less -S
   ```

Analysis

1. 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. We will be using our pre-generated full docking models to demonstrate the analysis process. Navigate to the analysis working directory. You should find the **docking_fill.silent** file already waiting for you.

      ```
      cd ../analysis
      ```

   2. Characterize your models using the InterfaceAnalyzer mover in RosettaScripts and calculate the RMSD to the native crystal structure with the RMSD filter. We've provided the necessarily XML file, options file, and 3GBM native PDB for you. This process should take ~10 minutes. You'll notice the InterfaceAnalyzer will automatically extract the PDBs from the silent files. In addition to the coordinate information, these PDBs also contain a scoretable at the bottom that breaks down Rosetta energies on a per-residue basis.

      ```
      ~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease \
        @docking_analysis.options -in:file:silent docking_full.silent >& docking_analysis.log &
      ```

   3. You can open up the top scoring and the lowest RMSD full docking model for comparison with the native structure. Note that you may have to use 3gbm_native -> A -> align -> all to this Pymol options to properly align structures for comparison.

      ```
      sort -nk 7 docking_analysis.csv | less -S
      pymol 3gbm_native.pdb 3gbm_HA_3gbn_Ab_full_0002.pdb 3gbm_HA_3gbn_Ab_full_0008.pdb
      ```

   4. 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 **sc_vs_rmsd.R** to make score vs rmsd plots

         ```
         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 &
         ```