Quick-start guide to protein-ligand docking using Rosetta 3.1

This is one of a series of tutorials designed to get you started with protein/ligand docking using Rosetta 3.1. It was produced to accompany Kaufmann et. al. (2010) Biochemistry, and the latest version can be found at http://meilerlab.org/. The purpose of ligand docking is to predict the structure of a protein/ligand complex, given the structures of the protein and the ligand separately. The RosettaLigand docking process randomly selects an initial position for the ligand that doesn't overlap the protein. Next the structure is refined through small movements of the ligand and sampling of side chain rotamers within the binding pocket. Small adjustments in the ligand torsion angles and backbone φ/ψ angles near the ligand further refine the docked model. The algorithm is described in Davis and Baker (2009) J Mol Biol 385, 381-92.

Steps to execute the docking run

1. The Rosetta suite is available for download from http://www.rosettacommons.org/. You will want to become familiar with the documentation that can be found in the /manual/ subdirectory, as well as the online Rosetta 3 User Manual: http://www.rosettacommons.org/manuals/rosetta3_user_guide/app_ligand_docking.html
2. Prepare the small molecule for docking. Generate a .mol, .sdf, or .mol2 file describing the 3D geometry of the small molecule. Use /src/python/apps/public/molfile_2_params.py to create a pdb file and a ".params" file describing the chemistry of the small molecule. Type molfile_2_params.py --help for more information. Append the small molecule PDB to the end of the protein PDB. At this point you can either place the small molecule into the putative binding pocket manually, using Pymol, or specify the X,Y,Z starting coordinates for the small molecule in a flags file.
3. Prepare the protein receptor for docking. Only sidechains near the initial ligand position are repacked during docking, to save time. This means *all* sidechains should be repacked before docking, so that any pre-existing clashes (according to Rosetta's energy function) can be resolved. Otherwise, a ligand placed near the clashing residues will allow them to repack and thus gain a large energy bonus that does not accurately reflect its binding affinity in that position. A program `ligand_rpkmin` is provided for this purpose; one should use the same -ex# flags as will be used during docking:

```
~/mini/bin/ligand_rpkmin.linuxgccrelease -database ~/rosetta3_database/ -ex1 -ex2
-ex1aro -extrachi_cutoff 1 -no_optH false -flip_HNQ -docking:ligand:old_estat
-docking:ligand:soft_rep -nstruct 10 -s 7cpa.pdb
```

The prepared structure labeled 7cpa_7cpa_input.pdb is found in the /ligand_docking/ directory. The small molecule has been placed in the putative binding site for this receptor.

4. If the native structure of your protein is already known, you can optionally provide it for benchmarking purposes. If provided, an RMSD between the native and predicted structures will be calculated. The /ligand_docking directory contains 7cpa_7cpa_native.pdb. This file has no header and contains no solvent and only heteroatoms of the docked ligand.
5. Create a flags file to specify the parameters Rosetta will use during docking, including the location and format of input and output files, and options that specify details of the algorithm. There is a sample flags file in the /ligand_docking/ directory included with this tutorial. It is worth reading the file in a text editor

becuase it is well-commented and contains pointers to more documentation. The parameters of the run can be changed by editing this file.

6. Conduct the docking run:

```
ligand_dock.linuxgccrelease @flags > ligand_dock.log
```

Analysis

1. Examine the output files generated during this docking run. This includes the log file and the silent.out file. The silent file is in a format unique to Rosetta, which minimizes file size by only recording differences between decoys and a reference structure.
2. Use the extract_atomtree_diffs program to create PDBs from the silent file:

```
./rosetta_source/bin/extract_atomtree_diffs.linuxgccrelease -database
~/rosetta_database -extra_res_fa input/1t3r.params -s silent.out
```

3. Load some of the decoys into your favorite molecular visualization program (Chimera, PyMOL, etc) and compare them with the native carboxypeptidase complexe (7cpa_7cpa_native.pdb).
4. The output structures can be ranked by the Rosetta energy score, where the lowest energy indicates the best structure. The Rosetta scores can be found in both the silent.out file, and at the end of the output PDB files. A common ranking approach is to take the top 5% of decoys by total energy, and then ranking the remaining decoys by interaction energy ("interface_delta"). A script called best_iface.py has been written to do just this. Use the help option for more information:

```
./rosetta_source/src/apps/public/ligand_docking/best_iface.py --help
```

5. Another method for analyzing the typically thousands of models produced by a Rosetta run is to cluster them by structural similarity, with the idea that the deepest (native) energy well is frequently the widest. Therefore the largest cluster should correlate with the most native-like structure. Rosetta can perform clustering analysis (see http://www.rosettacommons.org/manuals/rosetta3_user_guide/app_cluster.html) and the final selection of the best structure frequently involves a combination of clustering and energy score ranking.