Ligand 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)

In addition to following this sample docking problem, the user is encouraged to review the Rosetta user guide including the section on ligand-centric movers for use with RosettaScripts.

https://www.rosettacommons.org/docs/latest/

Overview

This small-molecule docking tutorial will go over how to prep and run small-molecule docking in Rosetta. For the remainder of this tutorial, the term "ligand" refers to small-molecules: in other words, this is not protein-protein docking.

1. Standard ligand docking: `1_vanilla_docking/`

   Determining the binding conformation of a small-molecule ligand within a pre-defined pocket.

2. Ensemble docking: `2_Ensemble_docking/`

   Rosetta Ligand Ensemble (RLE) was developed based on the assumption that similar ligands bind in a similar manner. The set of ligands is superposed and subsequently docked. This is meant to be run in conjunction with experimental SAR data commonly acquired in medicinal chemistry campaigns.

Due to time, I would highly suggest to go through the standard ligand docking procedure first as an introduction to how Rosetta handles small molecules then moving to Ensemble docking depending on your specific interests.

Ligand Docking with a G-Protein Coupled Receptor

The experimental data for this tutorial is derived from: **Chien, E. Y. T. et al. Structure of the human dopamine D3 receptor in complex with a D2/D3 selective antagonist. Science 330, 1091-5 (2010).**

This particular D3/eticlopride protein-ligand complex was used as a target in the GPCR Dock 2010 assessment, the results of which are discussed here: **Kufareva, I. et al. Status of GPCR modeling and docking as reflected by community-wide GPCR Dock 2010 assessment. Structure 19, 1108-1126 (2011).**

If you are interested in more information on the performance of Rosetta in modeling and docking D3/GPCRs in general, please consult **Nguyen, E. D. et al. Assessment and challenges of ligand docking into comparative models of g-protein coupled receptors. PLoS One 8, (2013).**

Dopamine is an essential neurotransmitter that exhibits its effects through five subtypes of dopamine receptors, important members of class A G-protein coupled receptors (GPCRs). Both subtype two (D2R) and subtype three (D3R) function via inhibition of adenylyl cyclase, and modulation of these two receptors has clinical applications in treating schizophrenia. However, the high degree of binding site conservation between D2R and D3R makes it difficult to generate pharmacological compounds that selectively bind to one but not the other. Today, we will examine how eticlopride, a D2R/D3R antagonist, binds to human D3R.

For the purposes of this exercise we will model a ligand / protein complex with a published structure, eticlopride bound to D3R (PDB: 3PBL), allowing us to compare our modeled poses with the native structure. For this tutorial we will use the crystal structure of DR3. Although, in reality it is more likely you will not have a published structure, and will have to create a comparative model for the protein (see the RosettaCM tutorial), but the steps in this tutorial will apply to both.

For this exercise, we will be preparing our input files in the **protein_prep/** and **ligand_prep/** folders. The modeling will be done in the **docking/** folder. The **scripts/** folder contains helpful ligand docking scripts that we will be using during this tutorial (you should never be copying files to or from this folder). All necessary files are also prepared in the **answers/** directory in case you get stuck.

1. Navigate to the ligand docking directory where you will find the **ligand_prep/**, **protein_prep/**, **docking/**, and **answers/** folders

   ```
   cd ~/rosetta_workshop/tutorials/ligand_docking/1_vanilla_docking
   ```

2. Prepare a human dopamine 3 receptor structure. We will do this by obtaining the crystal structure (3PBL) and removing the excess information.

   1. Change into the **protein_prep/** directory with the cd command

      ```
      cd protein_prep
      ```

   2. The **clean_pdb.py** script will allow you to automatically download a PDB file and clean it of information other than the desired protein coordinates. The 'A' option tells the script to obtain chain A only. The full crystal structure consists of two monomers.

      ```
      python2.7 ~/rosetta_workshop/rosetta/main/tools/protein_tools/scripts/clean_pdb.py 3PBL A
      ```

   3. There are two output files generated by clean_pdb.py: 3PBL_A.pdb contains the single chain A of the protein structure and 3PBL_A.fasta contains the corresponding sequence. 3PBL_A.pdb is the receptor structure we will be using for docking, copy this file into the docking directory.

      ```
      cp 3PBL_A.pdb ../docking
      ```

      Note: This structure has a T4-lysozyme domain instead of the third cytoplasmic loop. The T4-lysozyme is a stabilizing feature to aid in crystallography. Normally, we would truncate this lysozyme segment and perform loop modeling (as discussed in the RosettaCM tutorial) to regenerate the intracellular loop. However in the interest of time, we will keep the lysozyme containing structure because the eticlopride binding site is far from the intracellular domain.

3. Next, we will prepare the ligand files. Most of these files are already prepared for you in the interest of time, but the steps are explained.

   1. cd into the directory named **ligand_prep/**

      ```
      cd ../ligand_prep
      ```

   2. In the directory, you will find a pair of already prepared files: eticlopride.sdf and eticlopride_conformers.sdf

      1. eticlopride.sdf: This contains the eticlopride structure found in the 3PBL protein complex.
         Note: You can also find the ligand file from this particular PDB structure by going to the 3PBL page and scrolling down to the "Small Molecules" section. From there, you can click "Download SDF File" under the ETQ identifier.
      2. eticlopride_conformers.sdf: This is a set of conformations for eticlopride generated outside of Rosetta. The downloaded ligand eticlopride.sdf file contains only the single conformation found in the PDB so we must expand the library to properly sample the conformational space. We also need to add hydrogen atoms to the model since they are not resolved in the crystal structure. Feel free to open the file in Pymol and use the arrow keys in the bottom right of the window to scroll through the different conformations:

```
pymol eticlopride_conformers.sdf
```

This particular conformational library was generated using the Meiler lab BioChemicalLibrary (BCL). The BCL is a suite of tools for protein modeling, small molecule calculations, and machine learning. If you are interested in licensing the BCL, please visit http://www.meilerlab.org/bclcommons or ask one of the instructors. The BCL conformer generator tool is described in “BCL::Conf: small molecule conformational sampling using a knowledge based rotamer library” (Kothiwale, Mendenhall, Meiler 2015) Other methods of ligand conformer generation include OpenEye, MOE and web-servers such as Frog 2.1 or DG-AMMOS. The generated libraries will differ depending on the chosen method.

   3. Generate a params file and associated PDB files for eticlopride. The params file contains important information about eticlopride in order to properly build the molecule in Rosetta, including the partial charges of the atoms, the connectivity of the molecule, and internal coordinates. The parameters file is necessary for ligand docking because Rosetta does not have internal records for custom small molecules in its database.

      1. Type

         ```
         ~/rosetta_workshop/rosetta/main/source/scripts/python/public/molfile_to_params.py -h
         ```

         to learn more about the script for generating the params file.

      2. Type the following (‘\’ simply means this is all typed on a single line)

         ```
         ~/rosetta_workshop/rosetta/main/source/scripts/python/public/molfile_to_params.py \
           -n ETQ -p ETQ --conformers-in-one-file eticlopride_conformers.sdf
         ```

         Note: You may encounter a warning about the number of atoms in the residue. This is okay as Rosetta is merely telling you that the ligand has more atoms than an amino acid.
         Three total files will be generated: ETQ.params contains the necessary information for Rosetta to process the ligand, ETQ.pdb contains the first conformation, and ETQ_conformers.pdb contains the rest of the conformational library. I would highly suggest walking through one of these params files while looking at the corresponding PDB structure in a graphics program (Pymol, Chimera, MOE, your favorite).

   4. If you use the tail command on ETQ.params, you will notice the PDB_ROTAMERS property line that tells Rosetta where to find the conformational library. Make sure this line has ETQ_conformers.pdb as the property.

      ```
      tail ETQ.params
      ```

   5. Now that we have the necessary files for ligand docking, we can copy them over to the docking directory.

      ```
      cp ETQ* ../docking
      ```

4. Now we want to make our final preparations in the docking directory.

   1. Change to the **docking/** directory

      ```
      cd ../docking
      ```

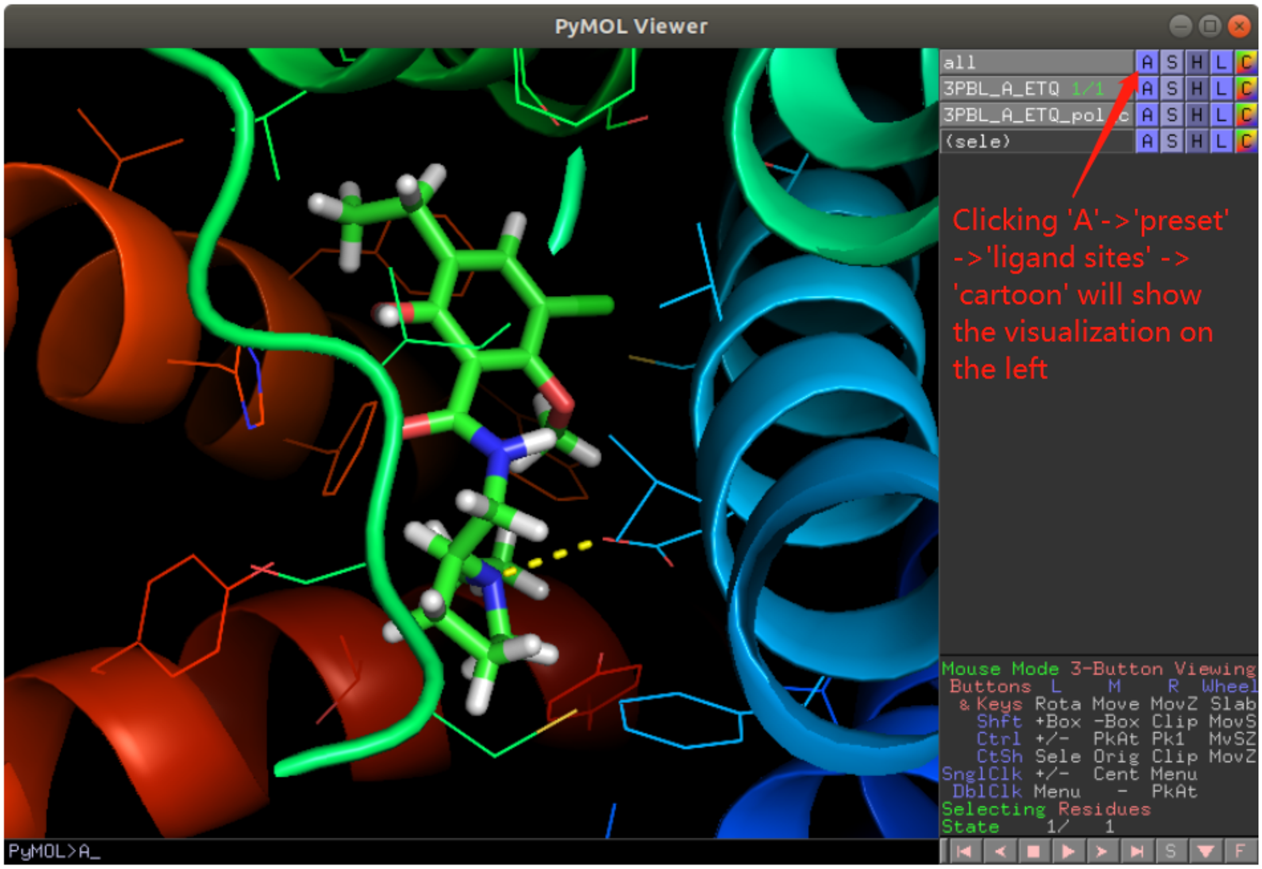
   2. Concatenate the ligand and protein pdb files together into one pdb file

      ```
      cat 3PBL_A.pdb ETQ.pdb > 3PBL_A_ETQ.pdb
      ```

   3. Open up our prepared pdb file to examine the receptor / ligand complex

      ```
      pymol 3PBL_A_ETQ.pdb
      ```

   4. Tip: ‘all->A->preset->ligand sites->cartoon’ will help you visualize the protein/ligand interface. The “Action” button is denoted by a single letter “A” in Pymol.



Clicking 'A'->'preset' ->'ligand sites' -> 'cartoon' will show the visualization on the left

Since this is a rudimentary exercise, we will start with the ligand in the known protein binding site. In practical application, it is unlikely that we will know the exact location of the binding site. Therefore we may need to try multiple starting locations, defining a starting point using the StartFrom mover or manually place the ligand into an approximate region using Pymol.

5. Once you close Pymol, make sure Rosetta has these three necessary input structure/parameter files in the docking directory. If you are missing any of these, copy them from ../answers/docking/
    1. 3PBL_A_ETQ.pdb: a single chain of the protein receptor structure with a default starting conformation for eticlopride
    2. ETQ_conformers.pdb: A pdb file containing all conformers generated from the eticlopride library
    3. ETQ.params: a Rosetta parameter file that provides the necessary properties for Rosetta to treat eticlopride
6. Next we need to make sure we have the proper RosettaScripts XML file, input options file, and "crystal complex" (the correct answer for comparison) in our directory. These files are provided to you as dock.xml, options.txt, and crystal_complex.pdb
    1. dock.xml - This is the RosettaScripts XML file that tells Rosetta the type of sampling and scoring to do. It defines the scoring function and provides parameters for both low-resolution coarse-grain sampling and high-resolution Monte Carlo sampling.
    2. options.txt - This is the options file that tells Rosetta where to locate our input PDB structures and ligand parameters. It also directs Rosetta to the proper XML file.
    3. crystal_complex.pdb - This is the D3-eticlopride complex from the PDB. It will serve as the correct answer in our case allowing us to make comparisons between our models and actual structures.

5. Run the docking study:

```
~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.linuxgccrelease \
  @options.txt -nstruct 5
```

This should take a few minutes at most, as we are using a reduced number of output structures. Feel free to generate more if you want, but we do provide an example of 500 models in the `answers/docking/out/`.

6. The Rosetta models are saved with the prefix 3PBL_A_ETQ_ followed by a four digit identifier. Each model PDB contains the coordinates and Rosetta score corresponding to that model. In addition, the model scores are summarized in table format in the score.sc file. The two main scoring terms to consider are:

    1. total_score: the total score is reflective of the entire protein-ligand complex and is good as an overall model assessment.
    2. interface_delta_X: the interface score is the difference between the bound protein-ligand complex and the unbound protein-ligand. The interface score is useful for analyzing ligand effects and for comparing different complexes.

7. One other metric to keep an eye on is the Transform_accept_ratio. This is the fraction of Monte Carlo moves that were accepted during the low resolution Transform grid search. If this number is zero or very low, the search space may be too restrictive to allow for proper sampling AKA the binding pocket is too small for the ligand to fit into.

8. In benchmarking examples when we have a correct crystal structure, ligand_rms_no_super_X will give us the RMSD difference between our model ligand and the crystal structure ligand given in crystal_complex.pdb. This is an important metric when benchmarking how well your models correlate to reality. When the crystal structure is unknown, we can also calculate model RMSDs using the best scoring structure as the "true answer".

9. Use pymol to visually compare your best-scoring model and worse-scoring model with the crystal structure provided in crystal_complex.pdb. Best scoring models will have negative scores. The "all->A->preset->ligand sites->cartoon" setting in Pymol is ideal for visualizing interfaces. What interactions were successfully predicted by Rosetta?

10. The visualize_ligand.py script in the scripts directory is a useful shortcut for doing quick visualizations of protein-ligand interfaces. It takes in a PDB and generates a .pse Pymol session by applying common visualization settings. The example below shows the command lines for using this script on the 0001 model but you are free to try it on any one (or more!) of your models. Note that for this step, you will need to activate the `python2.7` environment, known as `py27`. To do this, activate with the command `conda activate py27`.

```
python ~/rosetta_workshop/tutorials/ligand_docking/1_vanilla_docking/scripts/visualize_ligand.py \
    3PBL_A_ETQ_0001.pdb

pymol 3PBL_A_ETQ_0001.pse
```

Analysis

Since we generated such a small number of structures, it is unlikely to capture all the possible binding modes that you would expect to encounter in an actual docking run. In the **1_vanilla_docking/answers/docking/out/** directory, there are 500 models pre-generated using the exact same protocol. We will look at an example of how we can analyze this dataset.

1. cd into the **out/** directory

    ```
    cd out/
    ```

2. In addition to the 500 structures here, you will find the score.sc, a score_vs_rmsd.csv file, a rmsds_to_best_model.data, and several .png image files.

    1. score.sc: summary score file for the 500 structures as outputted by Rosetta

   2. score_vs_rmsd.csv: a comma separated file with the filename in the first column, total_score for the complex in the second column, the interface score in the third column, and ligand RMSD to the native structure in the fourth column.

      These values are generated directly from the scorefile using the interface_delta_X score and the ligand_rms_no_super_X columns. This can be done with `awk` commands, but we provided extract_scores.bash script to do this. This is a very specific script made for extracting useful information in ligand docking experiments. However, the script can be easily customized for extracting other information from Rosetta score files. If you have any in-depth questions about how it works or how to modify it, feel free to ask. To see how it in action, run:

      ```
      ~/rosetta_workshop/tutorials/ligand_docking/1_vanilla_docking/scripts/extract_scores.bash \
          score.sc
      ```

   3. rmsds_to_best_model.data: a space separated file containing RMSD comparisons with the best scoring model (not crystal structure!) for all PDB files. A more detailed discussion of this file will come further down in the tutorial. This file has the filename in the first column, an all heavy-atom RMSD in the second column, a ligand only RMSD without superimposition in the third column, a ligand only RMSD with superimposition in the fourth column, and heavy atom RMSDs of side-chains around the ligand in the fifth column.

      This file is generated using the calculate_ligand_rmsd.py script. It uses pymol to compare PDB structures containing the same residues and ligand atoms. It's a quick way of calculating the ligand RMSDs of Rosetta models. To see how this works, let's try it on the five models we generated in the previous steps:

      ```
       cd ../
       ~/rosetta_workshop/tutorials/ligand_docking/1_vanilla_docking/scripts/calculate_ligand_rmsd.py \
           -n 3PBL_A_ETQ_0003.pdb -c X -a 7 -o rmsds_to_best_model.data *_000*.pdb
      ```

      This command compares all five of your models to the one after the -n option. Your best scoring model may not be the one labelled 0003 so feel free to customize that option. The -c tells the script that the ligand is denoted as chain X. The -a tells the script to use 7 angstroms as the cutoff sphere for side-chain RMSDs. The -o option is the output file name. Lastly, we provided a list of PDBs using the wildcard selection.

      The script produces the rmsd_to_best_model.data file that you can open in any text editor. Feel free to ask questions if you would like to discuss more of how to customize this script for your own applications. Now let's go back to the pre-generated model directory:

      ```
       cd out/
      ```

   4. PNG files: plots made from the various data file mentioned above. The Python matplotlib package was used here but you are free to use any plotting software you prefer. There is one plotting script in the script folder. You can use that to plot xy scatter plot for rmsd vs score. To use this,

      ```
      python ../../scripts/score_vs_rmsd.py score_vs_rmsd.csv name_of_your_output_file
      ```

3. In this case, we have the correct answer based on the crystal structure so we can examine a score vs rmsd plot to see if the better scoring models are indeed closer to the native ligand binding mode. Open up the plot with the following command:

   ```
   gthumb score_vs_crystal_rmsd_plot.png
   ```

   If gthumb command is not found on your local workstation, try using the following:

   ```
   open score_vs_crystal_rmsd_plot.png
   eog score_vs_crystal_rmsd_plot.png
   ```

   On the X-axis you will see the ligand RMSD to the ligand in the crystal structure. On the Y-axis you will see the interface delta X score in Rosetta Energy Units. Notice the general correlation between RMSD and Rosetta Score, with a large cluster of highly accurate and low scoring models in the lower left hand corner.

4. In practical applications, we would not have the crystal structure for comparison. However, we can treat the best scoring model as the native model and see if we generate a similar funnel. This is one application of how we might use the calculate_ligand_rmsd.py script discussed earlier. Once we identify a desired “best model”, we can run the script to generate the rmsds_to_best_model.data. Some scripting may be required to put the information from multiple files together, depending on which software package you choose to graph with. To identify the best scoring model for this example, I selected the top 200 models based on the best overall score and then identified the best model by interface score. The best model for these plots is 3PBL_A_ETQ_0347.pdb. Open up the first plot with:

   ```
   gthumb score_vs_low_rmsd_plot.png
   ```

   Again, we see a cluster of good scoring models near the best scoring model with a general downward trend further away. We can zoom in on the cluster in the lower left hand corner to get an even better picture.

   ```
   gthumb score_vs_low_rmsd_zoom_plot.png
   ```

   We see the same overall trend in this cluster, suggesting that the top scoring models in this run are likely to be good predictors of the true ligand binding position.

5. Finally, and very importantly, take look at some structures. To sort the CSV file by interface score and take the top twenty, type:

   ```
   sort -t, -nk3 score_vs_rmsd.csv | head -n 20
   ```

   These should all be very low RMSD models. To compare a certain structure to the native in Pymol, use:

   ```
   pymol 3PBL_A_ETQ_0211.pdb ../crystal_complex.pdb
   ```

   I used 3PBL_A_ETQ_0211.pdb as the sample structure because it is one of the best scoring models, but feel free to examine any model you like. Do not forget the ligand site preset mode for visualizing interfaces or use the visualize_ligand.py script to generate pymol sessions. If you like, we can also look at some of the poor scoring models to see exactly what went wrong. To find the top 20 worse models by interface score:

   ```
   sort -t, -nk3 score_vs_rmsd.csv | tail -n 20
   ```

   3PBL_A_ETQ_0424.pdb should come up as a poor scoring, high RMSD structure. When we open it up in Pymol, we can see that the ligand binding direction is flipped 180 degrees compared to the native position. This can happen when there is an extended binding pocket, but in this case, the Rosetta score was able to discern the difference between these models.

   ```
   pymol 3PBL_A_ETQ_0424.pdb ../crystal_complex.pdb
   ```

Congratulations, you have performed RosettaLigand docking study! Now use your docked models to generate hypotheses and test them in the wet lab!

RosettaLigandEnsemble

The following protocol and commentary was directly extacted from Fu D., Meiler J. RosettaLigandEnsemble: A Small-Molecule Ensemble-Driven Docking Approach (2018).

Structure-activity relationships (SARs) refer to differences in binding affinity or biological efficacy following chemical scaffold derivatizations. Medicinal chemistry makes use of such minor modifications to optimize lead compounds for desired affinity and other pharmacological properties. This creates a massive wealth of SAR data on related ligands for a single protein target. The PubChem database alone contains over 200 million measurements of biological activities on approximately 10000 protein targets.(3) BindingDB specifically organizes a portion of its database into collections of congeneric ligands with at least one co-crystallized with the common protein target.(4) It is generally expected that highly similar ligands form similar interactions when binding to the same target.(5) We hypothesize that a docking algorithm that leverages this information can eliminate a portion of false-positive binding poses, i.e., poses that score well, but are incorrect. We have extended RosettaLigand to RosettaLigandEnsemble (RLE), an algorithm that can identify a binding mode favorable to a superimposed ensemble of congeneric ligands. This allows users to simultaneously dock a series of ligands in unison instead of individually as single ligands. We hypothesize that this will increase the efficiency and accuracy of sampling. We illustrate the hypothesized sampling advantage of RLE in Figure 1. Due to the presence of functional groups of varying sizes found within a SAR series, there may be binding modes available to certain molecules, but not others. RLE is capable of eliminating binding orientations not available to the ensemble as a whole. Furthermore, highly similar ligands are expected to bind in a similar fashion with common interactions to the chemical core

The receptor structure used in this example is the p53 core domain bound to a stabilizing small molecule (PDB: 4AGQ, protein.pdb) The ligand series should share a core scaffold by which the ligands can be aligned. This example contains fivecongeneric ligands, but any number between three and eight is a reasonable use case.

Ligand Preparation

For clarity, these scripts will be written for a single ligand, PDBID=4AGL, however in practice, this process must be done for each ligand in the set.

PyMol pair fitting is an easy way to manually align ligands by minimizing distance between core scaffold atoms. Automated ligand alignment tools mayalso be used but generally do not perform as well compared to manual inspection. Examples of aligned ligands can be found in the /prep/aligned_ligands/ directory.

We need to again generate conformers for the ligands. Below is an template BCL script used to generate conformer files, but you can use others if you choose. Examples of generated conformers files are in prep/conformers.

```
cd ~/rosetta_workshop/tutorials/ligand_docking/2_Ensemble_docking
bcl.exe molecule:ConformerGenerator -ensemble_filenames prep/aligned_ligands/4AGL.withH.sdf \
    -conformers_single_file prep/conformers/4AGL.conformers.sdf
```

Params File Preparation

Rosetta requires params file to properly handle small molecule ligands. Prior to this step, concatenate the aligned ligand structure with the corresponding conformers into a single SDF file such that the aligned structure is first in the file. This will insure that the inputs will maintain the core scaffold alignment when generating the conformers.

```
cat prep/aligned_ligands/4AGL.withH.sdf prep/conformers/4AGL.conformers.sdf >> \
    prep/make_params/make_params.4AGL.sdf
```

Since PDB files use three digit residue codes and single digit chain designations, it's helpful to assign a code for each ligand file. The example uses the prep/ligands.list file to label each ligand as residues 00B through 00F and corresponding chains B through F. This file also contains pK values for each ligand binding to the target receptor.

```
~/rosetta_workshop/rosetta/main/source/scripts/python/public/molfile_to_params.py \
    prep/make_params/make_params.4AGL.sdf --chain B -n 00B -p 00B --conformers-in-one-file
```

For each ligand, this command generates a PDB file containing the single aligned ligand structure, a PDB file containing the remaining ligand conformers, and a params file containing connectivity and charge information for the ligand. Examples of these files can be found in the /prep/rosetta_inputs/ directory using the previously discussed letter designations. If you wish to incorporate SAR during docking, then use a text editor to add NUMERIC_PROPERTY AFFINITY #### to the end of the params file, where #### represents an SAR measurement of the user's choice. Rank correlation is used in SAR mode and hence units only need to be self-consistent.

Input File Organization

For RLE runs, it is convenient to prepare a single PDB file containing the aligned ligands but concatenating the individual ligand PDB files. The conformer and params do not need to be joined. This is done as ligands.pdb in the /inputs/ folder. You'll also find the previously prepared protein receptor PDB in the same directory.

The example dock.xml provided uses the settings from the benchmark. Actual application use may require the user to alter these values according to biological context. The defined scoring function is based on the existingRosettaLigand scoring function, but may be substituted in the XML script. The provided options file defines Rosetta input and output directories along with a number of sampling parameters. A full options list is available on the documentation website. Theligand_ensemble option is necessary to use RLE; a weight of 0 can be used to run RLE without taking SAR data into consideration

Each simulation will produce X models, where X is the number of input ligands. These example output models are in the /outputs/ directory along with a score.sc scorefile.

```
~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.linuxgccrelease \
    @inputs/options -docking:ligand:ligand_ensemble 0 -nstruct 1
```

Output and Analysis

Individual protein-ligand predicted structures are labeled by a chain and a number designation, B_1.pdb through F_1.pdb. Structures with the same numeric label are based on the same docking simulation and have a common binding pose. The protein interface contacting each ligand are optimized independently. The score.sc file contains all score terms for each simulation across a single row. Generally, individual ligand interface scores are used to rank models, with a negative score indicating a better model. These ligand interface scores are listed as interface_delta_*, where * is the single letter ligand chain ID. The values are appended at the end of each output PDB, and also in the scorefile for each protein-ligand pair. One suggestion is for the end user to examine the top ten percent of models for each pair.