

# 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/>

## 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 most 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
2. Prepare a human dopamine 3 receptor structure.
  1. Change into the **protein\_prep/** directory with the `cd` command

```
cd protein_prep
```
  2. Copy the selected result structures from homology modeling to this directory.

1. No further preparation of the structure is needed. Copy that to the docking directory  
`cp D3R_1.pdb ../docking/`
3. If you already have crystal structure of your receptor, you can use that instead.
  1. 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.  
`~/Rosetta/tools/protein_tools/scripts/clean_pdb.py 3PBL A`
  2. There are two output files generated by `clean_pdb.py`: `3PBL_A.pdb` contains a single chain 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.

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 use the lysozyme containing structure as 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's BioChemicalLibrary (BCL). The BCL is a suite of tools for protein modeling, small molecule calculations, and machine learning. If you're interested in licensing the BCL, please visit <http://www.meilerlab.org/bclcommons> or ask one of the instructors. Other methods of ligand conformer generation include OpenEye's MOE software 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/main/source/scripts/python/public/molfile_to_params.py -h
```

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

2. Type

```
~/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.

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, let's 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. Make a combined receptor/ligand complex.

- You can manually place the ligand into the appropriate starting location
  1. Combine the receptor and the ligand by opening both in the same pymol window.

```
pymol D3R_1.pdb ETQ.pdb
```
  2. Change the Mouse settings to 3-Button Editing mode, and then use shift-left-click and shift-middle-click to move the ligand into the appropriate location
  3. Go to File->Save Molecule, then ctrl-click to select both molecules. Click OK, and save as D3R\_1\_ETQ.pdb

- Alternatively, you can combine the two arbitrarily, and rely on a starting positions file

1. Combine the receptor and the ligand

```
cat D3R_1.pdb ETQ.pdb > D3R_1_ETQ.pdb
```
2. Create a new file (D3R\_binding\_sites.pdb) which has waters placed in each of the appropriate starting locations

```
cp ../protein_prep/D3R_binding_sites.pdb starting_points.pdb
```

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

```
pymol D3R_1_ETQ.pdb D3R_binding_sites.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

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 four 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
4. starting\_points.pdb: A file specifying the starting sites for docking.

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 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 (This should take a few minutes at most, as we're using a reduced number of output structures):

```
~/Rosetta/main/source/bin/rosetta_scripts.linuxgccrelease @options.txt -nstruct 5 -s D3R_1_ETQ.pdb
```

6. The Rosetta models are saved with the prefix D3R\_1\_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.
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. 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:

```
../scripts/visualize_ligand.py D3R_1_ETQ_0001.pdb
```

```
pymol D3R_1_ETQ_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 **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, and a rmsds\_to\_best\_model.data file.

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.

This file was tabulated using the `extract_scores.bash` script and the `score.sc` file as input. 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:

```
../../scripts/extract_scores.bash score.sc > score_vs_rmsd.csv
```

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 ../
```

```
../../scripts/calculate_ligand_rmsd.py \  
-n D3R_1_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
```

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 the `score_vs_rmsd.sc` file with a spreadsheet program.

```
loffice --calc score_vs_rmsd.sc
```

You can then do an X-Y scatter plot of the various columns against each other. Common plots are rmsd on the X-axis and total score or interface score on the Y-axis. You can also examine total score on the X-axis and interface score on the Y-axis, which is a good plot in cases where you don't have a native structure. (You're looking for models which have a good interface score while simultaneously having a decent total score.

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. For this example, we compare the structures with the best model by interface score, `D3R_1_ETQ_0442.pdb`.

```
loffice -calc rmsds_to_best_model.data
```

Combine with the score columns from `score_vs_rmsd.sc`, and then plot `X_rms` (the rmsd of the ligand chain) against the interface score.

5. Finally let's 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
```

To compare a certain structure to the native in Pymol, use:

```
pymol D3R_1_ETQ_0442.pdb D3R_1_ETQ_0089.pdb ../crystal_complex.pdb
```

I used D3R\_1\_ETQ\_0442.pdb as the sample structure because it is one of the best scoring models, but feel free to examine any model you like. Don't 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
```

D3R\_1\_ETQ\_0405.pdb should come up as a poor scoring, high RMSD structure. When we open it up in Pymol, we can see that it's sitting lower in the binding pocket, at the second site. pymol D3R\_1\_ETQ\_0405.pdb ../crystal\_complex.pdb