* In addition to following this sample docking problem, the user is encouraged to review the following sections of the Rosetta 3.2 user guide

- RosettaScripts
- Loop Building Tutorial
- Ligand Docking
- Extract Atomtree Diffs

Ligand Docking Example Problem

HIV-1 protease inhibitors often lose their efficacy as HIV-1 protease acquires drug-resistance mutations. We are interested in how these resistance mutations will affect binding of various protease inhibitors.

1) Decide which protein ligand interaction you seek to model, and find a structure of a similar complex to use as a template.
    a. *Navigate to the directory tutorials/ligand_docking/1-file_prep*
    b. We have downloaded 2 PDBs: 2Q5K and 3EL4. 2Q5K contains wild-type HIV-1 protease bound to protease inhibitor Lopinavir. 3EL4 contains the protease mutant V82T/I84V bound to protease inhibitor Saquinavir. For the purposes of our docking study we will pretend that 3EL4 is not available to us. We want to know how well Saquinavir will bind to a V82T/I84V mutant. We will use 2Q5K as a template to Rosetta and attempt to predict the structure of the mutant complex. We will compare our predictions to the “correct answer”, 3EL4. Steps related to preparing the “correct answer” 3EL4 for comparison are colored red.
2) Generate a Saquinavir .sdf file with conformational sampling
    a. *Navigate to the directory tutorials/ligand_docking/2-Saquinavir/1-from_ideal*
    b. Find or create a 3-D structure for Saquinavir. We downloaded “ROC-ideal.sdf” from http://www.pdb.org/pdb/ligand/ligandsummary.do?hetId=ROC.
    c. Create a .mol file from the .sdf file by opening the file in PYMOL and saving it as a .mol file (“ROC_ideal.mol”)
    d. Generate conformations for the .mol file using OpenEye OMEGA or MOE. Pregenerated conformations are saved in “1-SDF_conformations/ROC_confs.sdf”.
        i. Here I used MOE to generate 200 conformations.
3) Generate a .params file and associated PDB conformations with Rosetta atom types for Saquinavir.
    a. Type “src/python/apps/public/molfile_to_params.py --help”.
    b. Type “molfile_to_params.py -n ROC -p ROC ROC_with_aromatic.sdf”
    c. Now ROC.params and a series of ROC PDBs have been created (these are found in the directory *2-ROC_pdb_confs*).
    d. Concatenate the PDBs into a single PDB. “cat ROC_000*.pdb > ROC_confs.pdb”
    e. Add the line “PDB_ROTAMERS ROC_confs.pdb” to the end of your ‘ROC.params’ file
4) Generate a .params file and Rosetta atom types for the Saquinavir conformation within 3EL4
    a. *Navigate to the directory tutorials/ligand_docking/2-Saquinavir/2-from_3EL4*
    b. Copy 3EL4 Saquinavir atoms to a new file: “grep ROC 3EL4.pdb > 3EL4_ROC.pdb”
    c. Open this file using MOE and save it as a .mol file. “3EL4_ROC.mol”.
    d. Add hydrogens using pymol “h_add”, and visually confirm that they were added correctly.
    e. Save the new .mol file as “3EL4_ROC_with_H.mol”
    f. Type “molfile_to_params.py -n ROC -p ROC 3EL4_ROC_with_H.mol”

    g. Two files should have been created: ROC.params and ROC_0001.pdb

5) Clean up the PDBs from step 1.
    a. *Navigate to the directory tutorials/ligand_docking/3-clean_pdbs*
    b. Use the clean PDB script to keep only 'ATOM' lines and single side chain conformations: "clean_pdb.py 2QK5.pdb > 2QK5_clean.pdb"; " clean_pdb.py 3EL4.pdb > 3EL4_clean.pdb"
    c. Append Lopinavir back onto the 2Q5K_clean.pdb.
        i. "grep AB1 2Q5K.pdb >> 2Q5K_clean.pdb"
        ii. We keep Lopinavir around so that we know where to place Saquinavir
    d. Append the contents of ROC_0001.pdb from step 5g onto the end of 3EL4_clean.pdb
        i. "cat ROC_0001.pdb >> 3EL4_clean.pdb"
6) Create the protease mutant template based on the wild-type 2Q5K_clean.pdb.
    a. Copy 2Q5K_clean.pdb to 2Q5K_mutant.pdb
    b. Open 2Q5K_mutant.pdb with a text editor. Remove the side chain atoms from residues 82 and 84 in chains A and B. Change the 3-letter residue codes to produce the mutant sequence V82T/I84V (VAL -> THR, ILE -> VAL).
    c. Note that if there were insertions or deletions between our homology model and the structure we are modeling, we would employ the loop building application to model such changes.
7) Next we want to place Saquinavir in the binding pocket of 2Q5K.
    a. Open a Saquinavir conformation from step 4c (ROC_0001.pdb), and 2Q5K_mutant.pdb (which contains Lopinavir) using pymol. "pymol 2Q5K_mutant.pdb ROC_0001.pdb"
    b. Superimpose Saquinavir on Lopinavir using the align command
        i. "align ROC_0001, 2Q5K_mutant and resn AB1"
    c. Delete the atoms associated with Lopinavir
    d. Save the 2Q5K/Saquinavir structure: "save 2Q5K_mutant_with_ROC.pdb, 2Q5K_mutant or ROC_0001"
    e. This structure now has Saquinavir with Rosetta atom types in the binding pocket.
8) Now we prepare 3EL4 so that we can compare our docking results with the "correct" answer.
    a. Open 3EL4_clean.pdb and 2Q5K_clean_with_ROC.pdb in pymol.
    b. Align 3EL4_clean.pdb with 2Q5K_ROC.pdb.
        i. "align 3EL4_clean.pdb, 2Q5K_ROC"
    c. Save the aligned structure: "save 3EL4_aligned.pdb, 3EL4_clean"
9) Next we prepare our docking options file
    a. *Navigate to the directory tutorials/ligand_docking/4-docking*
    b. An example options file is provided. Note that in addition to the input PDB we must provide a .params files for our ligand. We use several packing options that allow more fine-grained rotamer sampling.
    c. The "atom_tree_diff" file format is a condensed format for ligand docking. Usually only small changes within the interface occur during ligand docking. Atom_tree_diffs take advantage of this by storing a full PDB coordinate set only once – followed by a series of changesets. Omit this option for direct PDB output. Atom tree diff does not work properly with MPI

   d. You may want to try "–nstructs 10" and look at 10 models before running a large docking study.
   e. You may want to add "–mute all" to minimize output.

10) Now we prepare an XML-style Rosetta scripts file for docking.
   a. Copy the standard integration test docking script to your working directory:
      i. "cp test/integration/tests/ligand_dock_script/ligand_dock.xml ."
   b. To better understand the elements within this script, review the RosettaScripts documentation on ligand-centric movers: http://www.rosettacommons.org/manuals/archive/rosetta3.2_user_guide/RosettaScripts.html#Ligand-centric_Movers
   c. Note that during docking a soft repulsive score allows small clashes to exist, but that a standard repulsive score is used during final minimization
   d. The StartFrom element is unnecessary since we used pymol to position our ligand
   e. We'll reduce the size of our allowed translation to 2.5 Å, since we are confident about the location of our binding pocket.
   f. We'll allow full 360 degree rotation of the ligand and allow up to 1000 attempts at finding rotations that lead to acceptable attractive and repulsive scores.
   g. We'll allow the ligand torsion angles to be minimized during docking.
   h. During a final minimization we'll allow the backbone to minimize.
   i. The InterfaceScoreCalculator is provided with our prepared "correct_answer", 3EL4_aligned.pdb

11) Run the docking study: "$ROSETTA_BIN/rosetta_scripts.linuxgccrelease @options.txt –database $ROSETTA_DATABASE". Example output for 100 models ("-nstruct 100") is found in "atom_tree_diff.out". For a production study many 1000s of models would be produced.

12) Analyze your results
   a. *Navigate to the directory tutorials/ligand_docking/5-analysis*
   b. Use the get_scores.py script to produce a table of scores from the atom_tree_diff.out file.
   c. "src/apps/public/ligand_docking/get_scores.py > score_table.txt"
   d. Open score_table.txt in Excel and plot total energy vs. RMSD ("ligand_rms_no_super_x") and interface energy ("interface_delta") vs. RMSD. This RMSD value is calculated by comparing our predicted model with our "correct answer", 3EL4.
   e. Filter the score lines by total energy and then by interface energy. A standard practice is to take the top 5% by total energy and the top 10 by interface energy. Since we only have 100 models I selected the top 20% by total energy and the top 5 of these by interface energy. These models were # 19, 97, 15, 32, and 47.
   f. Extract these PDBs: "$ROSETTA_BIN/extract_atomtree_diffs.linuxgccrelease @extract_options.txt –database $ROSETTA_DATABASE <TAGS>"
   g. Inspect extracted models using pymol.

Congratulations, you have performed RosettaLigand docking study!

Now use your docked models to generate hypotheses and test them in the wet lab!