Tutorial 6: Simulation of bump-and-hole chemical genetics design

Author: Benjamin P. Brown (benjamin.p.brown17@gmail.com)

Date: 01-2022

Background

Rosetta and its affiliated script repositories collectively compose the most comprehensive toolbox for *in silico* macromolecular design. The greatest utility of Rosetta is arguably not that it can generate random foldable sequences, but that it provides a highly customizable user platform for design and modeling in a huge variety of contexts. One of the primary motivations for integrating the BCL into Rosetta was that we wanted to be able to do small molecule design with all the infrastructure that has been built for *in silico* modeling in Rosetta. In Tutorial 5, you experienced how protein modeling tools in Rosetta could be used in conjunction with small molecule modeling tools in the BCL to perform nuanced structure-based design of Abl kinase inhibitors.

In this tutorial, we will simultaneously perform sequence design on a receptor protein and small molecule design on an inhibitor scaffold. As of this writing, this is a unique feature of the BCL-Rosetta integration that is unavailable in any other software package.

Here, we will perform chemical genetics, or chemogenetics, *in silico* design experiments. Chemical genetics is the process of redesigning proteins in tandem with a small molecule modulator to create a high affinity pair. It can be challenging to selectively inhibitor protein isoforms without also inhibiting their homologs. Chemical genetics provides an important means to determine the functions of specific proteins in the absence of confounds that occur by inhibiting homologous proteins. Moreover, this approach has been utilized extensively in neuroscience to create Designer Receptors Activated Exclusively by Designer Drugs (DREADDs), which can be expressed and activated with drug *in vitro* or *in vivo* to control the activity of neurons.

Our example system will be the Bromodomain and Extra Terminal (BET) family protein BRD2. BRD2 plays an important role in transcription regulation. We will mimic the experimental bump-and-hole experiment that Runcie et al. 2018[1] performed by enumerating each of their designs *in silico*. Subsequently, we will experiment with more aggressive redesign of both the protein and scaffold than was performed in Runcie et al. 2018[1].

Part 1: Preparing a small molecule scaffold for design

Our input scaffold can be prepared just as in Tutorial 5; however, instead use files “Tutorial_6/ligand_prep/XYZ.pdb”. We will not walk through this step explicitly in this tutorial; please feel free to look back at Tutorial 5 Part 1 or use the prepared files in Tutorial_6/ligand_prep/.

Recall that the Rosetta atom indices that we will reference in our XML scripts will differ from the atom indices in “Tutorial_6/ligand_prep/XYZ.pdb”. The “bcl_rosetta.pdb” file was generated from `restype_converter` using “XYZ.params” as input and will correspond to the atom indices referenced in this tutorial.

Congratulations! Move on to Part 2.

Part 2: Ranking antagonists for BRD2 L383V against wild-type BRD2

There are times when you do not want to stochastically sample ligand modifications but instead want to scan a scaffold systematically with a few perturbation types. For example, it is often useful to scan individual ring with a few individual or paired halogen or methoxy substitutions. In these cases, the number of modifications that need to be made often number only in the dozens to hundreds and are quite computationally tractable. The co-space of the two BRD2 variants and inhibitory scaffold modifications qualifies as such a scenario.

While it is on the to-do list, there is currently not a simple Rosetta command-line application with which to perform these types of systematic studies (I swear I am working on it). The closest thing we have is the `molecule:Mutate` application in the BCL; however, if you use that then you need to create params files for every small molecule and then still pass it into Rosetta or PyRosetta to do the scoring anyway. Thus, independent of whether you use RosettaScripts or PyRosetta, there is a bit of scripting that needs to be done.

But not to worry, we can do it.

As was done in Runcie et al. 2018[1], we will modify three distinct regions of our starting scaffold:

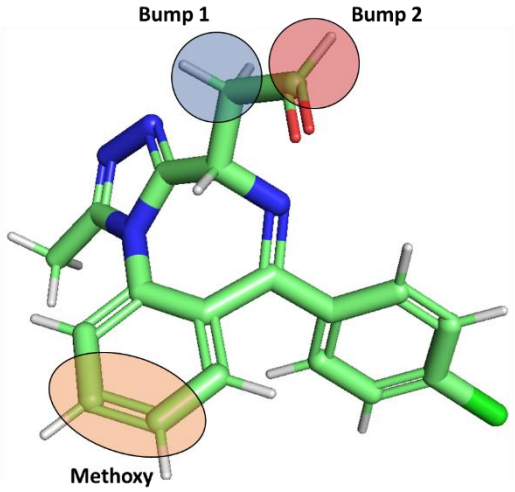

Figure 1. BRD2 chemogenetics design scaffold from Runcie et al. 2018. The region highlighted in orange will be derivatized with methoxy groups. The regions highlighted in blue and red will be the "bump" regions in our bump-and-hole simulation.

Because each of these modifications are effectively just substitutions, we can easily prepare three `AddMedChem` mutates in our RosettaScripts protocol. We will make one mutate for each of the "bump" regions and one mutate controlling whether the methoxy is added to position 8 or 9 on the scaffold (as represented in Runcie et al., 2018,[1] not based on our atom indexing).

```
<BCLFragmentMutateMover name="add_medchem_bump_r1"
ligand_chain="X"
object_data_label="AddMedChem(
ov_shuffle_h=false,ov_reverse=false,
medchem_library=%%bump_r1%%,
druglikeness_type=None,
mutable_atoms=0)"
/>
```

```
<BCLFragmentMutateMover name="add_medchem_bump_r2"
ligand_chain="X"
object_data_label="AddMedChem(
ov_shuffle_h=false,ov_reverse=false,
medchem_library=%%bump_r2%%,
druglikeness_type=None,
mutable_atoms=0)"
/>
```

```
<BCLFragmentMutateMover name="add_medchem_methoxy_scan"
ligand_chain="X"
object_data_label="AddMedChem(
ov_shuffle_h=false,ov_reverse=false,
medchem_library=%%methoxy_scan_frag%%,
druglikeness_type=None,
mutable_atoms=%%methoxy_res%%)"
/>
```

Notice that the first two mutates, `add_medchem_bump_r1` and `add_medchem_bump_r2`, each receive a variable for a different medchem fragment. Similarly, the `add_mechem_methoxy_scan` mutate has a variable to specify the atom index to which the methoxy will be added. All we need to do is loop over the combinations in our wrapper shell script.

But before we get to that, there is another important detail. Let's look at our scaffold.

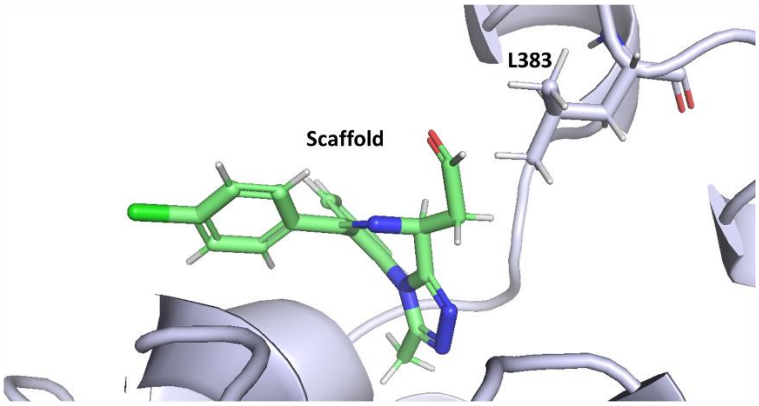

Figure 2. Bound pose of our example scaffold in the BRD2 binding pocket. The residue that was mutated in Runcie et al. 2018 is L383.

The first bump site will be grown from a sp3 carbon atom. Adding two heavy atom substituents to this carbon atom will create a stereocenter. We need to be careful to create the correct stereoisomers of our molecules. You will notice in the above mutate definitions that `ov_shuffle_h` and `ov_reverse` are set to false in all cases. This means that each and every time we run this script and add a fragment to this carbon atom, we will remove the lowest index hydrogen atom to make the addition. It will not be random. It will not start with the highest index hydrogen atom.

Therefore, we can define the sequence in which we perform our mutates to achieve the correct stereo-chemistry.

```
<ParsedProtocol name="run_a" mode="sequence">
<Add mover_name="add_medchem_methoxy_scan"/>
<Add mover_name="add_medchem_bump_r2"/>
<Add mover_name="add_medchem_bump_r1"/>
</ParsedProtocol>
```

Let's add a few more options to the run_a protocol so that we can apply the correct residue to position 383, minimize the scaffold, and relax after adding the bumps.

```
<ParsedProtocol name="run_a" mode="sequence">
<Add mover_name="mutate"/>
<Add mover_name="min_cycle_soft"/>
<Add mover_name="min_cycle_hard"/>
<Add mover_name="add_medchem_methoxy_scan"/>
<Add mover_name="add_medchem_bump_r2"/>
<Add mover_name="add_medchem_bump_r1"/>
<Add mover_name="min_cycle_soft"/>
<Add mover_name="min_cycle_hard"/>
<Add mover_name="relax_cycle"/>
<Add filter_name="score_filter"/>
<Add mover_name="add_scores"/>
</ParsedProtocol>
```

Run the following command-line to create just one sample structure using this protocol:

```
bash scripts/submit.local_sample.sh ./
```

Compare that structure to the crystallographic structure from Runcie et al. 2018. Does the stereochemistry match? How about the binding mode? Run that command again, but first set `ov_reverse`=true for one or more of the `AddMedChem` mutates. Did it change the resulting structure the way that you anticipated?

Cool beans! Let's keep going. We'll make a second protocol where we skip performing `add_medchem_bump_r1`, which will mimic designs that have a hydrogen atom instead at that position. Once this is completed, we're ready to loop over all our fragments.

```
<ParsedProtocol name="run_b" mode="sequence">
<Add mover_name="mutate"/>
<Add mover_name="min_cycle_soft"/>
<Add mover_name="min_cycle_hard"/>
<Add mover_name="add_medchem_methoxy_scan"/>
<Add mover_name="add_medchem_bump_r2"/>
<Add mover_name="min_cycle_soft"/>
<Add mover_name="min_cycle_hard"/>
<Add mover_name="relax_cycle"/>
<Add filter_name="score_filter"/>
<Add mover_name="add_scores"/>
</ParsedProtocol>
```

```
bash scripts/submit.local_serial.sh ./
```

Then we can analyze our results:

```
bash scripts/get_best_ave.sh 1
```

These may not be terribly impressive. Instead of making one model of each, let's make 10 and take the average to potentially reduce some of the noise in the ranking. Change the nstruct to 10 in the Chemogenetics.sh script and re-run.

While you wait for these to run, you can skip this block of questions highlighted in light gray. Just come back and fill it out when all of your samples are done.

Let's analyze the results using the **best** scoring protein-ligand interface from each of the 10 runs:

```
bash scripts/get_best_scores.sh
```

Hmm… And one more time by taking the final result as the **average** across all 10 structures.

```
bash scripts/get_best_ave.sh 10
```

Averaging across multiple structures seems to improve the quality of the binding affinity estimates (or at least, it did when I gave it a whirl). You do not have to do it, but previously I benchmarked our ranking ability on this system by averaging across 50 models of each system. The best round is shown plotted below:

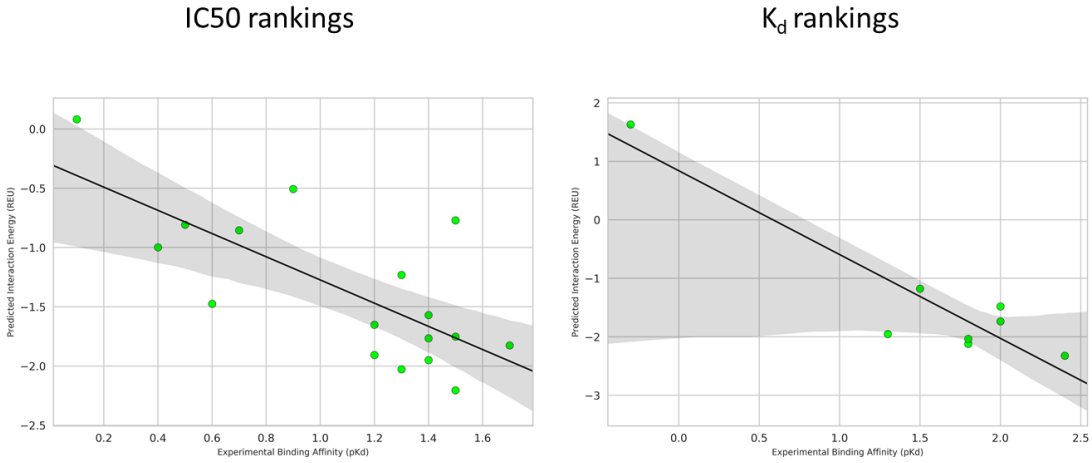

Figure 3. *In silico* replication of Runcie et al. L383 vs. V383 bump-and-hole design benchmark. Experimental binding affinity differences between L383 and V383 on the x-axis[1]; predicted differences on y-axis.

However, I did these 5 independent times and found that generally our Pearson correlations ranged from 0.65 to 0.80 for the IC50 set and 0.70 to 0.90 for the Kd set, which is consistent with the accuracy we would expect based on Shannon's RosettaLigand benchmarks[2]. As the Rosetta score function for protein-ligand interactions is improved, our ability to perform chemogenetics design with these types of protocols will naturally improve as well.

Repeat the enumeration but change the script to only run on the L383A variant. This way, we can compare results of L383A against wild-type and L383V. Once you have completed that, see if you can answer the questions below.

Which BRD2 variant, wild-type, L383V, or L383A, is most frequently found amongst the optimized mutant-ligand pairs? In other words, which variant has better RosettaLigand interaction energies with more ligand designs?

Which scaffold modifications are most frequently found in the optimized mutant-ligand pairs? Are your findings consistent with the bump-and-hole approach? How do they compare to the results in the manuscript[1,3]?

I found BRD2 L383A to be the most frequent mutant in the high affinity pairs. I also found the hole introduced by L383A was frequently filled with either an ethyl or propyl sidechain. These results are consistent with the findings of both manuscripts. This is good news – it suggests that we can successfully capture the primary effects of the bump-and-hole approach for this system.

Awesome, you finished Part 2. Let's continue with some more extensive design in Part 3.

Part 3: Let's just redesign the binding pocket simultaneously with the scaffold

In this section, we will redesign the protein binding interface while simultaneously designing a small molecule. For sake of the tutorial, we will continue to use the fragments employed in Runcie et al., 2018.

First, we will make a new resfile to define the allowed residue identities at each position in the pocket. This has already been done for you and should look like this:

```
## Sample BRD2 BD2 conserved L383 (41 in Rosetta numbering); the first
residue in the PIKAA list corresponds to native
NATAA
USE_INPUT_SC
start
28 A PIKAA WFYLV
30 A PIKAA FYWLV
34 A PIKAA VLA
39 A PIKAA LVA
41 A PIKAA LVA
44 A PIKAA YFWLV
86 A PIKAA YFWLV
87 A PIKAA NQDES
83 A PIKAA CMS
91 A PIKAA HNQ
93 A PIKAA VLA
96 A PIKAA MSC
```

Visualize these residues in PyMOL. Note that they are interface residues with our ligand scaffold.

To allow more sequence diversity during design, we will combine sampling backbone and sidechain degrees of freedom. We will optimize the sequence in a short 100 iteration Monte Carlo – Metropolis mover.

```
<Backrub name="backrub"/>
<PackRotamersMover name="mutate" scorefxn="hard_rep" nloop="5"
task_operations="ifcl,rrf,fix_notinterface"/>

<ParsedProtocol name="pseudo_coupled_moves">
      <Add mover_name="backrub"/>
      <Add mover_name="mutate"/>
</ParsedProtocol>

<GenericMonteCarlo name="run_pcm" mover_name="pseudo_coupled_moves"
trials="100" sample_type="low" filter_name="ifscore"
progress_file="%%progress_file%%.pseudo_coupled_moves.log"
temperature="0.593" drift="1" recover_low="1" reset_baselines="0"
adaptive_movers="0" preapply="0" />
```

The combination of backrub and rotamer sampling is inspired by work in the Kortemme lab[4]. They found that evaluating backbone and sidechain perturbations together as a single MCM improves redesign at

protein-ligand interfaces. I named our combined mover "`pseudo_coupled_moves`" to reflect that (and the "pseudo" because the CoupledMoves mover and application are a bit more involved than that).

Recall from the BCL-Rosetta integration talks that when the BCL passes a chemically perturbed small molecule back to the Rosetta pose, it saves conformers of the new molecule as rotamers. Specifically, it restricts conformer sampling to only the chemically perturbed atom dihedrals. Thus, only the "new" parts of the molecule are sampled. Movement in the rotation/translation of the ligand with respect to the molecule occurs in subsequent minimization and/or relaxing of the complex.

To prove to yourself that the MCM sampling will not rotate or translate the ligand, you can modify the downstream protocol to end after the MCM step so that you can visualize the output in PyMOL (not shown here).

Now that we have our new resfile and our sequence redesign movers, we need to make our ligand design movers. We will gently modify the movers we used in Part 2.

```
<BCLFragmentMutateMover name="add_medchem_bump_r1"
ligand_chain="X"
object_data_label="AddMedChem(
ov_shuffle_h=false,ov_reverse=false,
medchem_library=%%bump_r1%%,
druglikeness_type=None,
mutable_atoms=0)"
/>
```

```
<BCLFragmentMutateMover name="add_medchem_bump_r2"
ligand_chain="X"
object_data_label="AddMedChem(
ov_shuffle_h=false,ov_reverse=false,
medchem_library=%%bump_r2%%,
druglikeness_type=None,
mutable_atoms=0)"
/>
```

```
<BCLFragmentMutateMover name="add_medchem_methoxy_scan"
ligand_chain="X"
object_data_label="AddMedChem(
ov_shuffle_h=false,ov_reverse=false,
medchem_library=%%methoxy_scan_frag%%,
druglikeness_type=None,
mutable_atoms=16 17)"
/>
```

In addition, we will change our input shell script so that we randomly choose fragments from our input SD files instead of using SDFs of only one fragment each.

```
$ROSETTA \
-parser:protocol $XML \
-in:file:s "$PROTEIN $LIGAND" \
-parser:script_vars prefix="${PREFIX}" \
-parser:script_vars resfile=${RESFILE} \
-parser:script_vars methoxy_res=${METHOXY_RES} \
-parser:script_vars bump_r1=${BUMP_R1} \
-parser:script_vars bump_r2=${BUMP_R2} \
-parser:script_vars \
methoxy_scan_frag=Tutorial_6/methoxy_scan_fragment.sdf \
-parser:script_vars progress_file="${PREFIX}.gmc.log" \
-extra_res_fa ${PARAMS} \
-out:prefix $PREFIX \
-out:pdb_gz true \
-nstruct 10 \
-in:file:fullatom \
-restore_pre_talaris_2013_behavior \
-score:weights ligand \
-ignore_zero_occupancy false \
-linmem_ig 10 #> ${PREFIX}.log
```

Finally, we need to update our run protocols to reflect the changes in our sequence design strategy.

```
<ParsedProtocol name="run_a" mode="sequence">
<Add mover_name="min_cycle_soft"/>
<Add mover_name="min_cycle_hard"/>
<Add mover_name="add_medchem_methoxy_scan"/>
<Add mover_name="add_medchem_bump_r2"/>
<Add mover_name="add_medchem_bump_r1"/>
<Add mover_name="min_cycle_soft"/>
<Add mover_name="min_cycle_hard"/>
<Add mover_name="run_pcm"/>
<Add mover_name="relax_cycle"/>
</ParsedProtocol>
```

Notice that now we perform an initial minimization of our scaffold, we run design on the ligand, minimize again to remove obvious penalties, and then perform MCM sequence optimization before ending with a relax. Similarly, we need to modify `run_b`. I also added a `run_c`, which is effectively a control sequence design on the scaffold.

Take a moment to make sure you understand how the contents of the submission script and the XML script match together. If you need help understanding anything, please let one of us help.

Once you are confident in the functionality of your scrip, run the pocket design to create 10 quick designs:

```
bash scripts/ChemoGeneticsDesign.gmc.sh \
scripts/ChemoGeneticsDesign.gmc.xml BRD2.pdb XYZ.pdb XYZ.params \
resfiles/pocket.resfile output/GMC_POCKET_ > output/GMC_POCKET_.log &
```

Look at a few of your best scoring models. Which protein-ligand interface residues are mutated? Did residue L383 mutate? If so, did co-mutation with other residues change the preference of L383X for a particular bump modification on the ligand? Are there any residues that did not impart a beneficial change in ligand binding energy? If you are unsure, feel free to create more designs.

Here are a couple of my designs:

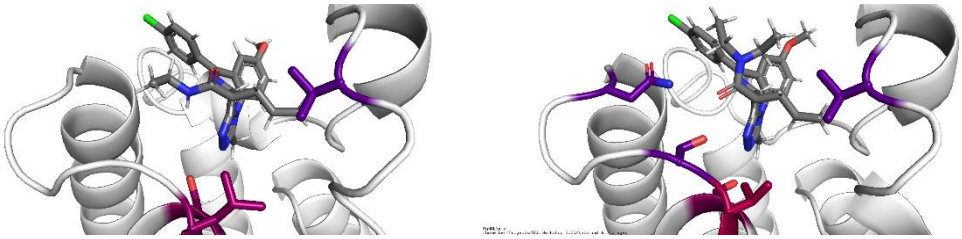

Figure 4. Illustration of a couple example *in silico* chemogenetics designs.

Try running the design again, but this time set up your simulation to sample different stereoisomers. **Hint: Look at Tutorial 5 Part 3.**

I encourage you to try new designs – expand the residue types available at each mutable residue position, use different `BCLFragmentMutateInterface` ligand mutates, etc. Good luck and have fun!

Congratulations! You have completed Tutorial 6.

References

(1) Runcie, A. C.; Zengerle, M.; Chan, K.-H.; Testa, A.; van Beurden, L.; Baud, M. G. J.; Epemolu, O.; Ellis, L. C. J.; Read, K. D.; Coulthard, V.; Brien, A.; Ciulli, A. Optimization of a “Bump-and-Hole” Approach to Allele-Selective BET Bromodomain Inhibition. *Chem Sci* **2018**, *9* (9), 2452–2468. https://doi.org/10.1039/c7sc02536j.

(2) Smith, S. T.; Meiler, J. Assessing Multiple Score Functions in Rosetta for Drug Discovery. *PLoS One* **2020**, *15* (10), e0240450. https://doi.org/10.1371/journal.pone.0240450.

(3) Baud, M. G. J.; Lin-Shiao, E.; Cardote, T.; Tallant, C.; Pschibul, A.; Chan, K.-H.; Zengerle, M.; Garcia, J. R.; Kwan, T. T.-L.; Ferguson, F. M.; Ciulli, A. Chemical Biology. A Bump-and-Hole Approach to Engineer Controlled Selectivity of BET Bromodomain Chemical Probes. *Science* **2014**, *346* (6209), 638–641. https://doi.org/10.1126/science.1249830.

(4) Coupling Protein Side-Chain and Backbone Flexibility Improves the Re-design of Protein-Ligand Specificity https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004335 (accessed 2022 -01 -13).