Tutorial 8: Rapid generation of non-canonical amino acid (NCAA) params for Rosetta with BCL

Author: Oanh Vu (oanh.t.vu.2@vanderbilt.edu)
Author: Benjamin P. Brown (benjamin.p.brown17@gmail.com)
Date: 01-2022

Background:

Calpains are intracellular, cytosolic, Ca2+-dependent cysteine proteases that are ubiquitously distributed, along with their endogenous inhibitor, calpastatin. The calpains consist of an 80 kDa catalytic subunit (the large subunit), containing four domains, I–IV (DI through DIV), and a 28 kDa regulatory subunit (the small subunit), containing two domains, V and VI (DV and DVI). Calpastatin is a 120 kDa protein that contains four repetitive inhibitory domains, each having three conserved regions termed A, B and C, and an N-terminal domain. Each of the conserved regions has been shown to bind preferentially to separate domains of calpain. A and C bind to the calmodulin-like DIV and DVI, respectively, while B binds to the protease DII.

Calpain's proteolytic activity is activated by an increase in the intracellular concentration of Ca2+ and then kept under control by the inhibitory effects of calpastatin[1].

Since calpains act as regulatory enzymes, their cleavage modifying the function of their substrates. As their substrates have important roles in many physiological processes, excessive activation of calpain could play a role in the pathology of a variety of disorders, including cerebral ischemia, ischemic myocardial infarction, spinal cord injury, muscular dystrophy, and cataract. Therefore, calpain-calpastatin interactions are considered targets for pharmacological intervention [2].

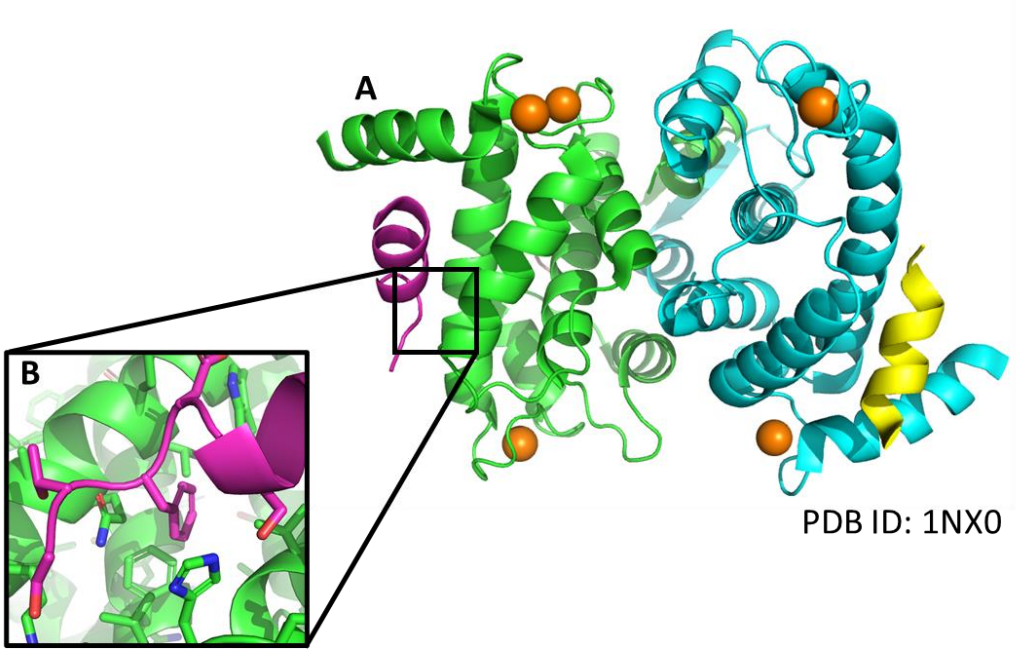

Figure 1. Calpain/Calpastatin designable interface. (A) Zoom-out high-level view of protein-peptide interface. (B) F183 (F610) (magenta).

As you can see in the figure (or in PyMOL), At position 610 (or 183 in the PDB model), the wildtype phenylalanine is buried in a large hydrophobic pocket. Hence, we can potentially add substitutions on the benzine ring to better fill the binding pocket. In other words, we will mutate the PHE into some Non-

natural amino acid (NCAAs), which are PHE derivatives. Noncanonical amino acids are, simply, amino acids that are not among the twenty canonical amino acids nor their 19 D- counterparts. Many of them are still found in proteins with moderate frequency and they can be very effective at improving folding and binding properties of proteins.

What we are doing in this tutorial is inspired from a precedent design study [3]. In this study, the author reported that mutating Phe610 to 4-methyl-phenylalanine (4MF) improve the binding affinity from 5.8 mM to 2.6 mM. In this tutorial, we will run design on PHE610 using a set of 26 different PHE derivatives and see if there could be potential more promising mutants for this residue.

Part 1.Generate params files for the set of the NCAAs
Enter the directory 2_generate_NCAA_params_files

```
cd 2_generate_NCAA_params_files
```

The list of PHE derivative compounds is stored in phe_der.sdf in the folder input_files.

Create new directory to store different types of output files

```
mkdir dipeptide_sdf conformers_sdf
```

First, we need to generate the dipeptide form of each of the NCAAs, and an instruction on the output SDF footer to instruct Rosetta the indices of the backbone atoms, and connection points (where the residue will form peptide bonds with the next residues). To do this, we will use the `molecule:GenerateRosettaNCAAInstructions` application in BCL.

To see all the options of the application, run

```
bcl.exe molecule:GenerateRosettaNCAAInstructions -help
```

To generate the dipeptide form with the instruction, run this command

```
bcl.exe molecule:GenerateRosettaNCAAInstructions \
-input_filenames ../input_files/phe_der.sdf \
-output_prefix dipeptide_sdf/phe_der_out \
-generate_3D \
-logger File generate_dipeptides.log \
-extra_properties AROMATIC \
-generate_partial_charge_file \
-chirality auto \
-explicit_aromaticity
```

If you want to know the details of how the app runs, check out the log file generate_dipeptides.log. You should get 1 SDF file of the dipeptide form of the NCAAs and its partial charge file, which contain partial charges for each atom (pi + sigma charges). Look at the dipeptide SDF files on PyMOL to make sure that the CA chirality is preserved compared to that of the original ones.

If you look at the footer of the SDF file, you will find the instruction for Rosetta to generate params files for the NCAAs later under the **RosettaParamsInstructions** property entry. For example, for the phe_der_out_0.sdf file (4-Borono-D-phenylalanine)

```
> <RosettaParamsInstructions>
M  ROOT 3
M  POLY_N_BB 3
M  POLY_CA_BB 4
M  POLY_C_BB 2
M  POLY_O_BB 5
M  POLY_IGNORE 9 10 11 12 13 14 15 16 17 18
M  POLY_UPPER 6
M  POLY_LOWER 8
M  POLY_CHG 0
M  POLY_PROPERTIES ALPHA_AA D_AA PROTEIN AROMATIC POLYMER
M  END
```

this instruction specifies the backbone atoms, connection points, formal charge, and properties of the NCAAs for the molfile_to_params_polymer_main.py script (see next step).

This guide will help you understand each line of the instruction

```
M  ROOT \[the number of the N-terminal atom\]
M  POLY_N_BB \[ the number of the N-terminal atom \]
M  POLY_CA_BB \[the CA atom number\]
M  POLY_C_BB \[the C atom number\]
M  POLY_O_BB \[the O atom number\]
M  POLY_IGNORE \[all the atoms of the capping groups except UPPER and
LOWER\]
M  POLY_UPPER \[the nitrogen atom number of the C-terminal methyl
amide\]
M  POLY_LOWER \[the carbonyl atom number of the N-terminal acetyl\]
M  POLY_CHG \[the charge\]
M  POLY_PROPERTIES \[any properties, like PROTEIN and CHARGED, etc.\]
M  END
```

Question: Check the indices of the backbone atoms in several output dipeptides, what do you notice?
Next, we will generate the rotamer library for each of 26 NCAAs using the BCL `molecule:ConformerGenerator` applications. This command will generate 200 conformations for each NCAA, which will be stored in a SD file inside the output folder conformers_sdf. Since PHE derivatives are fairly rigid, we should be fine with small number of conformations inside the PDB rotamer library. However, more flexible NCAAs will need higher number of conformations.

```
for id in `seq 0 26`;
do
bcl.exe molecule:ConformerGenerator \
-conformation_comparer SymmetryRMSD 0.25 \
-max_iterations 2000 -top_models 100 -cluster \
-ensemble_filenames dipeptide_sdf/phe_der_out_${id}.sdf \
-conformers_single_file \
conformers_sdf/phe_der_out_${id}_Rotamer.sdf \
-explicit_aromaticity \
-logger File phe_der_out_${id}_Rotamer.log;
done
```

By now, for each NCAA, you should have a sdf file of 200 conformers with instruction of how to create Rosetta params file and a file contain computed atomics partial charges. We can now generate the params files for out 26 NCAAs. In Rosetta, params files store a variety of chemical and geometric information used to define the shape and chemical connectivity of an amino acid building block or other small molecule. For more information of params file, read more here (https://www.rosettacommons.org/docs/latest/rosetta_basics/file_types/Residue-Params-file).

We will create the params files using a python script inside the scripts/molfile_to_params_polymer folder, **molfile_to_params_polymer_main.py.** To understand different options for this script, type the following:

```
python \
../scripts/molfile_to_params_polymer/molfile_to_params_polymer_main.py
-help
```

The output from the script displays the help options:

```
usage: molfile_to_params_polymer_main.py [-h] -i INPUT [-n NM] [-p FILE]
                                         [-c X,Y,Z [X,Y,Z ...]] [-m MAX]
                                         [-k FILE] [--clobber] [--no-param]
                                         [--no-pdb] [--all-in-one-pdb]
                                         [--polymer] [--peptoid]
                                         [--partial_charges FILE]

This script creates Rosetta params files from the molecule mol/sdf file

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        name of the intput mol/sdf file
  -n NM, --name NM      name ligand residues NM1,NM2,... instead of
                        LG1,LG2,...
  -p FILE, --pdb FILE   prefix for PDB file names
  -c X,Y,Z [X,Y,Z ...], --centroid X,Y,Z [X,Y,Z ...]
                        translate output PDB coords to have given heavy-atom
                        centroid coods
  -m MAX, --max-confs MAX
                        dont expand proton chis if above this many total
                        confs
  -k FILE, --kinemage FILE
                        write ligand topology to FILE
  --clobber             overwrite existing files
  --no-param            skip writing .params files (for debugging)
  --no-pdb              skip writing .pdb files (for debugging)
  --all-in-one-pdb      writing all pdb files into 1 file (for debugging)
  --polymer             write a polymer style param file instead of a ligand
                        param file
  --peptoid             modifier for the polymer flag, adjusts PDB style
                        naming to be correct for peptoids
  --partial_charges FILE
                        file that contains the partial charges of each atom in
                        the input file
```

Let's run the script for our NCAAs. Since the code name of the residues should have 3 characters. We will add N in front of the NCAA ids

```
for id in `seq 0 26`;
do
name=`printf "N%.2d" $id` ;
python
../scripts/molfile_to_params_polymer/molfile_to_params_polymer_main.py
\
--clobber --all-in-one-pdb --name $name \
-i conformers_sdf/phe_der_out_${id}_Rotamer.sdf \
--partial_charges dipeptide_sdf/phe_der_out_${id}.PartialCharges.txt;
done
```

Put the path to the rotamer libraries to the params file

```
for i in `seq -w 1 29`; do \
echo PDB_ROTAMERS N${i}_rotamer.pdb >> N${i}.params;
done
```

You can also output the residue from in Rosetta to test if Rosetta can read in the params correctly. Let's try on residue N09.

```
${ROSETTA}/source/bin/restype_converter.linuxgccrelease \
-extra_res_fa N09.params -out:prefix N09_rosetta_convert -sdf_out
```

Look legit, let design the residue with the fancy NCAA params file that we have just created!

Part 2. Run peptide design with NCAAs in Rosetta

We will move the directory of the next step

```
cd ../3_peptide_design_with_NCAAs/
```

Clean the input pdb file and look at the PHE residue

```
# We are downloading the pdb file and clean it and extract chain A and
C from the complex
wget http://www.rcsb.org/pdb/files/1nx0.pdb
../scripts/clean_pdb.py 1nx0.pdb AC
```

Question: You can now look at the complex, which is a structure of Calpain Domain VI in Complex with Calpastatin DIC, on PyMOL. You can try to locate the F183 residue of Calpastatin DIC (chain C) and examine the binding pocket around this residue.

Rosetta can use noncanonical amino acids in packing, minimization, and design. For more documentation regarding protein design and protein design with NCAAs in Rosetta, please also read the relevant RosettaCommons pages:

PackerPalettes
Working with noncanonical amino acids in Rosetta

Here is the Rosetta XML file of peptide protocol (../scripts/design_with_NCAA.xml)

```
<ROSETTASCRIPTS>
    <PACKER_PALETTES>
       <CustomBaseTypePackerPalette name="base_ncaa"
additional_residue_types="%%res_type%%" />
    </PACKER_PALETTES>
    <RESIDUE_SELECTORS>
       <Chain name="prot" chains="A"/>
       <Chain name="pep" chains="C"/>
       <Neighborhood name="interface" selector="pep" distance="8"/>
       <Not name="not_interface" selector="interface"/>
    </RESIDUE_SELECTORS>

    <TASKOPERATIONS>
       Include rotamer options from the command line
       <InitializeFromCommandline name="ifcl" />
       Design and repack residues based on resfile
       <ReadResfile name="rrf" filename="%%resf_file%%"/>
       <RestrictToRepacking name="repack_only"/>
       <OperateOnResidueSubset name="fix_notinterface" selector="not_interface">
           <PreventRepackingRLT/>
       </OperateOnResidueSubset>

    </TASKOPERATIONS>
    <MOVERS>
       FavorNativeResidue name="favor_native" bonus="0.75"/>
       Design the antibody interface
       <PackRotamersMover name="design" scorefxn="REF2015" task_operations="ifcl,rrf"
packer_palette="base_ncaa"/>
       Analyze the resulting interface
       <InterfaceAnalyzerMover name="analyze" scorefxn="REF2015" packstat="0"
pack_input="0" pack_separated="1" fixedchains="A" />
       Optimization
       Backrub name="backrub" pivot_residues="1B-12B" pivot_atoms="CA"/>
       <FastRelax name="relax" scorefxn="REF2015" repeats="1"
task_operations="fix_notinterface">
           <MoveMap bb="false" chi="false">
              <Jump number="0" setting="false"/>
              <ResidueSelector selector="interface" bb="true" chi="true"/>
              <ResidueSelector selector="not_interface" bb="false" chi="true" />
           </MoveMap>
       </FastRelax>
    </MOVERS>
    <FILTERS>
       Ddg name="ddg_f" scorefxn="REF2015" threshold="0" jump="1" repack="false"
repeats="1" />
       ShapeComplementarity name="sc_f" min_sc="0.5" jump="1" />
    </FILTERS>
    <PROTOCOLS>
       Run the design protocol
       Add mover="favor_native" />
       <Add mover="design" />
       Add mover="backrub" />
       <Add mover_name="relax"/>
           Calculate interface metrics for the final sequence
       <Add mover="analyze" />
       Add filter="ddg_f" />
       Add filter="sc_f" />
    </PROTOCOLS>
    <OUTPUT scorefxn="REF2015" />
</ROSETTASCRIPTS>
```

The residue file to instruct Rosetta to perform design (../scripts/design.resf)

```
NATAA # default is repacking only
start

#* A NOTAA ACDEFGHIKLMNPQRSTVWY # disallow canonical aa in the design
of the peptide (type A)
182 C PIKAA
WX[N00]X[N01]X[N02]X[N03]X[N04]X[N05]X[N06]X[N07]X[N08]X[N09]X[N10]X[N
11]X[N12]X[N13]X[N14]X[N15]X[N16]X[N17]X[N18]X[N19]X[N20]X[N21]X[N22]X
[N23]X[N24]X[N25]X[N26]
```

Copy all the params files and pdb_rotamers files that we have generated from the previous step over to the current directory

```
cp ../2_generate_NCAA_params_files/N*.params
../2_generate_NCAA_params_files/N*rotamer.pdb .
```

Command line to run design and generate 10 design models

```
bash ../scripts/design_with_NCAA.sh \
    design \
    1nx0_AC.pdb \
    ../scripts/design_with_NCAA.xml \
    ../scripts/design.resf \
N00,N01,N02,N03,N04,N05,N06,N07,N08,N09,N10,N11,N12,N13,N14,N15,N16,N1
7,N18,N19,N20,N21,N22,N23,N24,N25,N26 \
    "N00.params N01.params N02.params N03.params N04.params N05.params
N06.params N07.params N08.params N09.params N10.params N11.params
N12.params N13.params N14.params N15.params N16.params N17.params
N18.params N19.params N20.params N21.params N22.params N23.params
N24.params N25.params N26.params" \
    10 \
    .
```

Make sure that you notice Rosetta read in the rotamer library of NCAAs correctly by looking at the output log on the screen

```
protocols.rosetta_scripts.ParsedProtocol: ======================BEGIN MOVER
PackRotamersMover - design======================
core.pack.task: Packer task: initialize from command line()
core.pack.rotamer_set.RotamerSet_: Using simple Rotamer generation logic for
N00
core.pack.rotamers.SingleLigandRotamerLibrary: Added 1800 rotamers for N01
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N02
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N03
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N04
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N05
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N06
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N07
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N08
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N09
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N10
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N11
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N12
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N13
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N14
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N15
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N16
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N17
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N18
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N19
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N20
core.pack.rotamers.SingleLigandRotamerLibrary: Added 800 rotamers for N21
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N22
core.pack.rotamers.SingleLigandRotamerLibrary: Added 800 rotamers for N23
core.pack.rotamers.SingleLigandRotamerLibrary: Added 600 rotamers for N24
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N25
core.pack.rotamers.SingleLigandRotamerLibrary: Added 200 rotamers for N26
```

Questions: What are the PHE derivatives that are selected in design? Did you expect those selected residues? Feel free to generate more than 10 models to see more diverse set of mutants.

Part 2 Extension (Optional): Alternative approach for params file storage/loading

It can be undesirable to have to pass NCAA params from the `extra_res_fa` to use them. An alternative approach is to store your newly generated params files in a Rosetta database. Let's try that here.

Using one (or all) of your newly generated parameter files, perform the following two steps:

(1) Copy the params files to ${ROSETTA}/main/database/chemical/residue_type_sets/fa_standard/residue_types/l-ncaa/
(2) Add the lines "residue_types/l-ncaa/XXX.params" and "residue_types/l-ncaa/XXX.params" to ${ROSETTA}/main/database/chemical/residue_type_sets/fa_standard/residue_types.txt. Try to place your new NCAAs in an intuitive section of that file (i.e., near other NCAAs) and provide a comment.

Next, there are two additional important considerations.

(1) There is not a formal location in the Rosetta database for storage of PDB rotamers. Therefore, find a convenient place to store them either in a separate directory or in a local branch of the database.
(2) Make sure that your newly-added params file contain an absolute path to the PDB rotamers in the PDB_ROTAMERS section of the params file.

With this approach, you may also use the `MutateResidue` mover to create your NCAA without specifying a special packer palette. Here is an example using `MutateResidue` to mutate the Phe in position 183 (PDB numbering of original complex) into a new N09 residue.

```
<MutateResidue name="mutate" target="183" new_res="N09"
preserve_atom_coords="true" mutate_self="false" />
```

Try it out!

Part 3. Designing a target-specific amino acid library with BCL-Rosetta

When we start venturing into the world of NCAAs, what we are arguably doing is adding more "small molecule" or "medicinal chemistry" character to our protein. I say that because many NCAAs contain very typical small molecule chemistry functional groups – halogens, sulfonyls, heterocycles, etc. – that are not found in canonical amino acids (CAAs). This means that the chemical space available for exploration in NCAA design is vastly larger than design with CAAs. If you have a pre-defined set of NCAAs available through a vendor (e.g., Sigma-Aldrich), then it may be better to simply pre-design NCAA libraries as described in Parts 1 and 2. If you have not settled on a single vendor, need to explore structure-activity relationships, and/or have the ability to custom-make your own NCAAs, then it may be beneficial to perform design on NCAAs the same way that we do with small molecules.

An ongoing development project in Rosetta is to add the ability to perform these types of NCAA design simulations on-the-fly. Despite this still being in-progress, you can achieve a similar effect without too much trouble if you are willing do a small amount of pre-processing. As an example, let's re-design the same PHE residue, but this time to do as a BCL-Rosetta integration drug design project. We will then take our favorite design and generate NCAA params for it so that it can be used in subsequent simulations.

We have already extracted the residue for you. We will start by generating params for it as if it were a small molecule.

Navigate to the output directory.

```
cd Tutorial_8/1_NCAA_design
```

Generate local conformers of the residue.

```
bcl.exe molecule:ConformerGenerator \
-ensemble_filenames ../input_files/XXF.extracted.sdf \
-conformers_single_file XXF.confs.sdf \
-max_iterations 1000 -top_models 100 \
-skip_rotamer_dihedral_sampling
```

Create params:

```
Rosetta/main/source/scripts/python/public/molfile_to_params.py \
-n XXF -p XXF \
--root_atom=1 --extra_torsion_output --clobber \
--conformers-in-one-file XXF.confs.sdf
```

We have also prepared an alchemical design XML script for you. If you have completed Tutorials 5 – 7, you should feel comfortable with the contents of the XML script. Make sure to read it and verify that you understand the mutate options. Consult Tutorials 5 – 7 for assistance.

```
bash ../scripts/AffinityDesignD4.sh \
../scripts/AffinityDesignD4.xml \
../input_files/1nx0_A.pdb XXF.pdb XXF.params
```

Visualize the output in PyMOL. Choose a favorite. My favorite is phenylpropyl derivative (not based on score, but just because I think it is a cool use of the `ExtendWithLinker` internal extension). I will create polymer-style params for it following the protocol we discussed in Parts 1 and 2.

Extract the design from the output PDB file and convert it to SDF.

```
egrep -w "ATOM|HETATM" 1nx0_A_XXF_0005.pdb | grep LIG > XPF.pdb
```

```
obabel -i pdb XPF.pdb -o sdf -O XPF.sdf
```

If you do not have OpenBabel installed, use the provided "XPF.sdf" file and install OpenBabel later. Create the Rosetta instructions file with BCL.

```
bcl.exe molecule:GenerateRosettaNCAAInstructions \
-input_filenames XPF.sdf \
-output_prefix NCAA_XPF \
-generate_3D \
-logger File NCAA_XPF.log \
-extra_properties AROMATIC \
-generate_partial_charge_file \
-chirality auto \
-explicit_aromaticity
```

```
bcl.exe molecule:ConformerGenerator \
-conformation_comparer SymmetryRMSD 0.25 \
-max_iterations 2000 \
-top_models 100 \
-cluster \
-ensemble_filenames NCAA_XPF_0.sdf \
-explicit_aromaticity \
-conformers_single_file NCAA_XPF_0_Rotamer.sdf
```

```
python \
../scripts/molfile_to_params_polymer/molfile_to_params_polymer_main.py
\
--clobber \
--all-in-one-pdb \
--name XPF \
-i NCAA_XPF_0_Rotamer.sdf \
--partial_charges NCAA_XPF_0.PartialCharges.txt
```

And then you're ready to perform a Rosetta simulation! As an example, we can go back to the complex of our peptide and target protein. We will mutate residue 183 from PHE into our new residue, XPF, relax the interface, and perform a restrained Metropolis-Hastings simulation of our complex:

```
bash ../scripts/MutateIntoNCAA.sh \
../scripts/MutateIntoNCAA.xml \
1nx0_AC.pdb \
XPF.params \
183 \
XPF \
MUTATE_TEST_
```

Does the new NCAA at position 183 make good contacts at the interface? Compare the rotamer distribution of the complex to the rotamer distribution of the peptide alone:

```
bash ../scripts/MutateIntoNCAA.sh \
../scripts/MutateIntoNCAA.xml \
1nx0_C.pdb \
XPF.params \
183C \
XPF \
MUTATE_TEST_PEPTIDE_
```

Do you think this is a good residue to use? Which contribution to binding free energy may make this residue a poor choice? It's not required for the Tutorial, but for fun one estimate the enthalpic contributions to binding by decomposing the intramolecular and intermolecular interaction energies of the complex, peptide, and receptor. Then, one could estimate the entropic contributions to binding using the quasi-harmonic approximation.

Congratulations! You have finished Tutorial 8!

References

1. Todd, B.; Moore, D.; Deivanayagam, C. C. S.; Lin, G.-d.; Chattopadhyay, D.; Maki, M.; Wang, K. K. W.; Narayana, S. V. L., A Structural Model for the Inhibition of Calpain by Calpastatin: Crystal Structures of the Native Domain VI of Calpain and its Complexes with Calpastatin Peptide and a Small Molecule Inhibitor. *J. Mol. Biol.* **2003,** 328, (1), 131-146.
2. Dókus, L. E.; Yousef, M.; Bánóczi, Z., Modulators of calpain activity: inhibitors and activators as potential drugs. *Expert Opin Drug Discov* **2020,** 15, (4), 471-486.
3. Renfrew, P. D.; Choi, E. J.; Bonneau, R.; Kuhlman, B., Incorporation of noncanonical amino acids into Rosetta and use in computational protein-peptide interface design. *PloS one* **2012,** 7, (3), e32637-e32637.