How to Create a Protein PDBQT File for Docking

Dr. S. Eagon
California Polytechnic State University

Before we begin, you'll need the following software:

- ChimeraX
- OpenBabel
- Notepad++ (or other text editor)

These programs are freely available for most platforms.

Getting Started:

First, you'll need to have a PDB file of your protein with a known inhibitor, most commonly from the protein data bank. You can use a protein bound to the native ligand if you can't find one with a bound inhibitor. If no crystal structure exists at all, you can build one using a homology modeling program. The SWISS-MODEL website is an excellent free resource for generating homology models given a known genetic sequence.

Trimming Down the PDB File:

Once you've got the PDB file, open it up in ChimeraX. You'll first need to determine the center of the inhibitor/ligand in the crystal with the following command:

```
$ measure center ligand
```

This will output the coordinates in the x,y,z dimensions. Write this down, as you'll need it for the docking runs.

Next we'll need to remove as much of the protein and all the non-interacting small molecules/atoms. This generally means removing any cofactors, salts, additives, solvent molecules (water), and the ligand itself. By default, we remove all water molecules <u>unless</u> we know a particular molecule is critical to binding, in which case we leave it in. We do leave in co-factors in they reside close to the inhibitor binding area (note the discussion about metal atoms in the "Known Issues" section below).

In addition to removing non-interacting small molecules/atoms, we also try to trim down the protein as much as possible. For some docking programs, a smaller PDBQT will make the program faster. More importantly, proteins can be too large to convert, so we want to minimize the size to avoid this problem.

For example, take the following protein (inhibitor is colored green):

Delete all of this
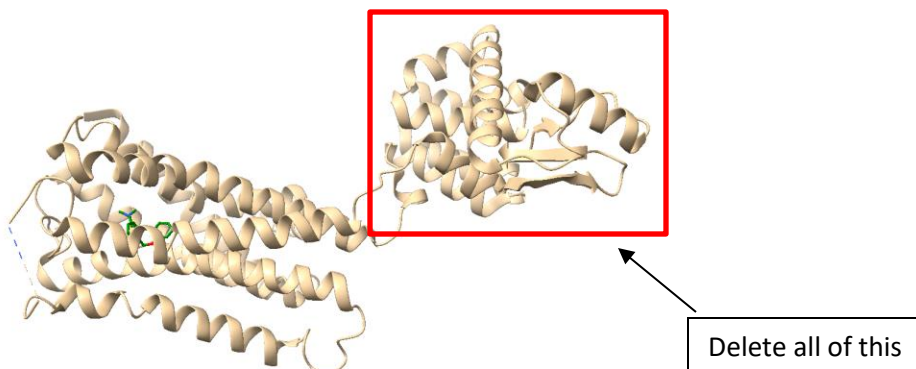
You can see that there is a part of this protein that's no where near the ligand, so we'll remove all of this.

Once you've trimmed down as much as you can, you should be left with only part of the protein, and maybe a nearby cofactor. Save this file as a PDB file. We usually add the name DOCK after our protein ID to identify them, like 3k60_DOCK.pdb to indicate the trimmed version of protein 3k60.

Converting the Trimmed PDB to a PDBQT File:

In our experience, Babel is best used by line command. Use the following command to convert your PDB file (use foo.pdb as an example):

```
$ babel foo.pdb foo.pdbqt -p 7.4 -xr
```

The -p flag indicates proper addition of H's at 7.4 (or whatever number you put in). The -xr flag is to force it into a rigid molecule (i.e. minimize branches to help keep the file small). There are version of OpenBabel that have a GUI, and it can do the same thing—just be sure to indicate the same flags as above.

Once your PDBQT has been output, open it with a text editor and visually inspect it. **Note that if there are too many residues, the charges don't get added or the conversion fails**. **It's best to cut down to the smallest possible size, and a single chain if possible.** In our experience, the size limit is around 7,000 atoms total. Remember, you only really need the protein pocket space around the inhibitor.

Known Issues:

1. Non-Standard Atom Types

This happened to use when we were targeting a crystal which had some Se atoms. Babel couldn't handle this, and output 0.000 for all charges. So, went into the PDB and replaced all instances of Se with S, then it worked just fine. Our standard protocol is now to change non-standard atom types to their standard supported types.

2. ANISOU Lines in the PDB File

If you have lines with "ANISOU" these are anisotropic temperature factors. You can either use find/replace to change all instances of "ANISOU" with "REMARK" to ignore them, or you can remove them with:

```
$ sed '/^ANISOU/ d' input.pdb > output.pdb
```

3. Metals in the PDB File

If you have a metal in your file when you try to convert it, this often will give you a bad output file where there are no charges (all Q lines are 0.000) or it fails to convert after the line containing the metal.

To fix this, we go into the trimmed DOCK.pdb file with a text editor, and find the metal atom lines. Remove these lines from the PDB file, but keep the information (we're going to paste it back in later). We save this as a different file, usually something like foo_DOCK_nometal.pdb

Next, convert your PDB file without the metal with your normal command:

```
$ babel foo_DOCK_nometal.pdb foo_DOCK_nometal.pdbqt -p 7.4 -xr
```

Once you have the output file, open the PDBQT file, then manually add the metal atom back in using the coordinates from the original PDB file and adding the appropriate charge. Most metals are in the formal +2 state, and literature suggests setting a charge of +0.95 (see _J. Mol. Graph. Model._, **2003**, _22_, 115). Here's an example from a previous structure, where the line in blue was manually added in a text editor.

```
ATOM   2572 O  ALA A  1    52.804 40.724 28.923 0.00 0.00   -0.270 OA
ATOM   2573 CB ALA A  1    50.882 42.830 29.197 0.00 0.00   +0.074 C
HETATM 2574 ZN  ZN A  1    43.055 33.160  4.739 0.00 0.00   +0.950 Zn
TER
```

Important note -> When writing in a line into the file in a text editor like the above line with Zinc, NEVER use the TAB button! Always use the space bar to add spaces and align as best as possible. If you use the TAB button, the program will interpret that as a break and place your atom in the wrong place entirely!

We once had an unusual problem protein with a heme in it. We had to strip the Fe from the input file, removed all of the "CONNECT" lines, then use the regular

```
$ babel foo.pdb foo.pdbqt -p 7.4 -xr
```

Then we put back in the iron the same way that I did with the Zn in the previous example.

4. Residues with Multiple Conformations

Some PDB files will have multiple instances of some sidechains in the same crystals. You'll need to delete one of these prior to converting to the PDBQT. If one of the residue conformations is near the binding pocket, we choose the closest one by default.