PDB scoring and preparation with Rosetta

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!

Scoring structures with Rosetta

One of the simplest tasks in Rosetta is scoring a given protein structure. The prefered method of scoring a protein structure is with the "score_jd2" application. This is version of the scoring application which uses the unified input/output options (the "JD2" system). There is also an older version of the scoring application called "score", which uses a slightly different set of options.

1. Prepare your working directory. You will work in this directory for the rest of the tutorial.
    1. Create a directory in the protein_folding directory called my_files and switch to that directory.

    ```
    mkdir my_files
    cd my_files
    ```

2. Scoring PDBs.
    1. From the ~/rosetta_workshop/tutorials/scoring_and_prep/inputs/ directory, copy over the PDB files **1QYS.pdb**, **3R2X.pdb**, and **3TDM.pdb**.
    2. Score the PDBs using the score_jd2 application. "-in:file:s" (or just "-s" for short) is used to specify PDB structure input.

    ```
    ~/rosetta_workshop/rosetta/main/source/bin/score_jd2.linuxgccrelease \
    -s 1QYS.pdb 3R2X.pdb 3TDM.pdb
    ```

    3. You should find that you get a number of errors. This is because these are files directly from the PDB, which contain residues which Rosetta does not read by default. (Sugars, phosphates, water, etc.) You can tell Rosetta to ignore these by passing the option "-ignore_unrecognized_res".

    ```
    ~/rosetta_workshop/rosetta/main/source/bin/score_jd2.linuxgccrelease \
    -s 1QYS.pdb 3R2X.pdb 3TDM.pdb -ignore_unrecognized_res
    ```

    4. Examine the tracer output. (The messages which get printed to the console.) This contains a number of informational and diagnostic messages about the run. (The amount of tracer output can be controlled by the "-mute", "-unmute" and "-out:levels" options.) In this output the important things to notice is that Rosetta is able to reinterpret the selenomethionine (MSE) residues in 1QYS.pdb as regular methionine (MET), that it recognizes when atoms are missing and adds them back in, that it will do conformational sampling (packing) for residues which are missing sidechains, and that it is able to autodetect disulfide bonds.
    5. In the directory, you should have a new file named "score.sc". This contains the tabular output of the scoring. For each structure scored, it not only lists the total score, but also has columns for individual components of the scorefunction. (For example, fa_atr is the atractive component of the van der Waals interaction energy, and fa_elec is the Coulombic energy. See https://www.rosettacommons.org/docs/latest/rosetta_basics/scoring/score-types for more details.)

6. score_jd2 can also output the actual edited structures it does the scoring on. Use the "-out:pdb" option to get output of the structures in PDB format.

```
~/rosetta_workshop/rosetta/main/source/bin/score_jd2.linuxgccrelease \
-s 1QYS.pdb 3R2X.pdb 3TDM.pdb -ignore_unrecognized_res -out:pdb
```

7. In the directory, there should be three new PDBs, 1QYS_0001.pdb, 3R2X_0001.pdb, and 3TDM_0001.pdb. These correspond to the names of the input structures, with "_0001" added. Using PyMol or Chimera, compare these structures to the starting structures. Can you see where missing atoms were added, and where the unrecognized residues were removed?
8. Open 1QYS.pdb and 1QYS_0001.pdb in a text editor and compare them side-by-side. Notice that the PDB header lines are gone. (Rosetta for the most part ignores these lines.) At the end of the file is now a table, listing a residue-by-residue breakdown of the scoring terms.
    1. Which residue has the highest total score?
    2. Notice that the residue numbering differs in the score table versus in the ATOM lines and in PyMol. (For example, the first valine is residue 6 by PyMol, but residue 4 in the score table.) The numbering scheme in the score table is called "pose numbering" in Rosetta. It starts at 1 for the first residue in the file, and increments by one for each residue, regardless of gaps, missing density, or different chains. In contrast, the values reported by PyMol and Chimera are refered to as "PDB numbering" in Rosetta. These can start at any number, skip numbers, or even jump around in numbering. In addition, there is a chain letter associated with the PDB numbering. In this way you can have a residue 4 on chain A and another residue 4 on chain B. In pose numbering there is only ever a single residue 4. – The presense or absence of the chain letter designation is a good indication as to which numbering scheme is being used. Different contexts in Rosetta may use one or the other to specify residues. If a chain letter is present, it's likely PDB numbering. If it's absent, it's likely pose numbering.
9. Often it's helpful to renumber your PDBs so that PDB numbering and pose numbering are the same - some protocols in Rosetta actually require this. There is a Python script called clean_pdb.py which will both renumber your PDB, as well as remove any unrecognized (non-protein) residues. To use, give the PDB filename (which must be in all uppercase) without the ".pdb" extension, as well as the chain letters for the chains you want to extract (or ignorechain to get all chains).

```
~/rosetta_workshop/rosetta/tools/protein_tools/scripts/clean_pdb.py 1QYS A
~/rosetta_workshop/rosetta/tools/protein_tools/scripts/clean_pdb.py 3TDM AB
~/rosetta_workshop/rosetta/tools/protein_tools/scripts/clean_pdb.py 3R2X ignorechain
```

    - The "-out:file:renumber_pdb" option of JD2 can be used to get score_jd2 to renumber the output PDBs as well, though this will keep all the chains.
10. Score the output structures (1QYS_A.pdb 3TDM_AB.pdb and 3R2X_ignorechain.pdb)
    1. This time we use a text file which lists the filenames of the PDBs. The format is one PDB per line.

        ```
        cp ~/rosetta_workshop/tutorials/scoring_and_prep/inputs/cleaned_structures.txt .
        ```

    2. Score the files using the "-in:file:l" ("-l"; PDB filename list file) to specify the list file. To separate out the scores, we specify the output scorefile name with "-out:file:scorefile"

        ```
        ~/rosetta_workshop/rosetta/main/source/bin/score_jd2.linuxgccrelease \
        -l cleaned_structures.txt -out:pdb -out:file:scorefile cleaned.sc
        ```

    3. Examine the ouput files (1QYS_A_0001.pdb 3TDM_AB_0001.pdb and 3R2X_ignorechain_0001.pdb) in PyMol and a text editor. Note the renumbering.
11. Sometimes it's helpful to get a more detailed analysis of the scoring of structures. There are two utilities in Rosetta for detailed breakdowns of scoring information
    1. per_residue_energies gives the same information as in the table at the end of a PDB, but in a tabular format without the atom lines, which can be easier to deal with.

```
~/rosetta_workshop/rosetta/main/source/bin/per_residue_energies.linuxgccrelease \
-s 1QYS_A_0001.pdb -out:file:silent per_res.sc
```

2. residue_energy_breakdown decomposes the energy contributions further, into internal residue energies (onebody) as well as residue interaction energies.

```
~/rosetta_workshop/rosetta/main/source/bin/residue_energy_breakdown.linuxgccrelease \
-s 1QYS_A_0001.pdb -out:file:silent energy_breakdown.sc
```

3. The files are in whitespace separated tabular format, and can be read by many data processing programs, including spreadsheets.
   1. Open the files in the Open Office spreadsheet program. It's easiest to launch it from the commandline:

      ```
      ooffice -calc per_res.sc energy_breakdown.sc
      ```

   2. In the import/formatting option window, make sure that you separate by space, and that you are merging delimiters.
   3. You can sort the table by using the sorting quick buttons, or by going to the data menu and selecting the sort entry.
   4. All energy values are negative better, so higher numbers are worse scores.
   5. From the per_res.sc file, which residue in 1QYS has the highest (worse) total energy (score)? Which has the worst steric clash score (fa_rep - the Lennard-Jones repulsive contribution). Which two residues have the worst internal conformation energy (fa_dun - derived from the rotamer probability)?
   6. From the energy_breakdown.sc file, which residue pair has the highest score in 1QYS? Which three pairs have the worst fa_rep score?

3. Working with silent files

   Silent files are a Rosetta-specific file format. The benefit of this format is that they take up less space on disk than PDBs do, and can store a large number of structures in a single file. (Many filesystems have difficulties working with directories that contain thousands of files. Silent files allow you to simultaneously work with thousands of structures without encountering filesystem issues.)

   1. Copy **4TQ5_silent.out** from the inputs directory to your working directory
   2. Open 4TQ5_silent.out in a text editor and take a brief look - this file is in the "binary" silent file format. (You don't need to understand ther details of the format.)
   3. Most applications (including the set of "JD2" applications) can read in silent files with -in:file:silent

      ```
      ~/rosetta_workshop/rosetta/main/source/bin/score_jd2.linuxgccrelease \
      -in:file:silent 4TQ5_silent.out -out:file:scorefile 4TQ5_silent.sc
      ```

   4. Silent files contain scorelines from the application that produced them. You can extract these

      ```
      grep SCORE 4TQ5_silent.out > 4TQ5_silent.scorelines
      ```

      The numbers in 4TQ5_silent.sc should be the same as for 4TQ5_silent.scorelines, as when the silent file was generated it was with the default talaris2013 scorefunction.

   5. There are different scorefunctions in Rosetta depending on the application and protocol. The talaris2013 scorefunction is the current default and works well for most cases, but you may need to specify a different scorefunction for different applications. For example, 4TQ5 is a membrane protein, so would be better scored using a membrane scorefunction. Here we use the "membrane_highres_Menv_smooth" scorefunction. This scorefunction requires additional information - a "spanfile" which specifies where the membrane is. The **4TQ5.span** file is provided for you in the inputs directory.

      ```
      ~/rosetta_workshop/rosetta/main/source/bin/score_jd2.linuxgccrelease \
      -in:file:silent 4TQ5_silent.out -score:weights membrane_highres_Menv_smooth \
      -in:file:spanfile 4TQ5.span -out:file:scorefile 4TQ5_membrane.sc
      ```

- Note that the scores are different for the membrane scoring versus the default talaris2013 scoring.
- Most protocols can take alternative scorefunctions, although some may not have been adequately tested for use with membrane proteins, symmetric proteins, etc.

6. The score_jd2 application can be used to convert silent files to PDBs. We'll specifically be extracting the four best scoring (lowest total_score) structures. We can do so with the -in:file:tags, which takes the names (tags) of the structures we want to extract. Keep in mind that rescoring added a "_0001" to the structure name, so the input tag should be without it

```
~/rosetta_workshop/rosetta/main/source/bin/score_jd2.linuxgccrelease \
-in:file:silent 4TQ5_silent.out -score:weights membrane_highres_Menv_smooth \
-in:file:spanfile 4TQ5.span -in:file:tags 4TQ5_0002 4TQ5_0007 4TQ5_0009 4TQ5_0019 \
-out:pdb
```

- You should get PDB structures 4TQ5_0002_0001.pdb 4TQ5_0007_0001.pdb 4TQ5_0009_0001.pdb and 4TQ5_0019_0001.pdb as output, which can be opened in a regular structure viewer.
- The option "-no_nstruct_label" can be used to keep Rosetta from adding "_0001" to the structure names.
- This conversion is general - you can convert multiple PDBs to silent files with "-in:file:s" and "-out:file:silent", or combine multiple silent files with "-in:file:silent" and "-out:file:silent"

4. Preparing structures with relax

What you saw with the 1QYS residue scores - that starting structures have residues with very poor scores - is common with experimental structures, or structures prepared with other computational modeling software. This is because the Rosetta scorefunction can be sensitive to small changes in atom location. Small, sub-angstrom changes which do not substantially change the fit to experimental structural determination data can have big effects on Rosetta scoring. As these poor scoring regions can affect protocols, it is often useful to relax a structure into the Rosetta energy function.

1. Use the relax application to create a number of models relaxed into the Rosetta energy function:

```
~/rosetta_workshop/rosetta/main/source/bin/relax.linuxgccrelease \
-s 1QYS_A_0001.pdb -nstruct 3
```

- "-nstruct" specifies the number of structures to output.

2. Open the relaxed structures (1QYS_A_0001_0001.pdb 1QYS_A_0001_0002.pdb and 1QYS_A_0001_0003.pdb) and the starting structure (1QYS_A_0001.pdb) in a structure viewer and compare them. You may want to use the alignment command to overlay the structures
3. The loop region of residues 10-13 moves significantly (2-3 Angstroms). This may be due to a crystal contact that isn't being represented in the monomer form, or it may be an artifact of how the relax application works. If the structure of the backbone from the input structure is important to us, we can have relax restrain ("constrain" in Rosetta terminology) the backbone coordinates to their input values. By default, relax will reduce the influence of the constraints (ramp them down) as the protocol progresses. The option "-relax:ramp_constraints false" will keep the constraints on throughout the protocol.

```
~/rosetta_workshop/rosetta/main/source/bin/relax.linuxgccrelease -s 1QYS_A_0001.pdb \
-relax:constrain_relax_to_start_coords -relax:ramp_constraints false -nstruct 3 \
-out:prefix bb_
```

4. Open the output PDBs (bb_1QYS_A_0001_0001.pdb, bb_1QYS_A_0001_0002.pdb, and bb_1QYS_A_0001_0003.pdb) in a structure viewer and compare them to the starting structure (1QYS_A_0001.pdb). Notice how the backbone atoms stay more-or-less in the same place as the input structure, but how the sidechain atoms are free to move.

5. If you want to preserve the coordinates of the sidechain atoms as well, the "-relax:coord_constrain_sidechains" option, in conjunction with the -relax:constrain_relax_to_start_coords option, will do that by restraining the positions of all the sidechain heavy atoms.

```
~/rosetta_workshop/rosetta/main/source/bin/relax.linuxgccrelease -s 1QYS_A_0001.pdb \
-relax:constrain_relax_to_start_coords -relax:coord_constrain_sidechains \
-relax:ramp_constraints false -nstruct 3 -out:prefix sc_
```

6. Rescore all the relaxed structures into a single scorefile.

```
~/rosetta_workshop/rosetta/main/source/bin/score_jd2.linuxgccrelease \
-s 1QYS_A_0001.pdb 1QYS_A_0001_*.pdb bb_1QYS_A_0001_*.pdb sc_1QYS_A_0001_*.pdb \
-out:file:scorefile relax_rescore.sc
```

7. Sort the structures by total score, and examine which runs gave the lowest energies, and by how much the energies were reduced in comparison to the starting structure.

```
sort -nk2 relax_rescore.sc | awk '{print $2, $NF}'
```

   - This sorts all the lines from the relax_rescore.sc file numerically with the sorting key being the second field, and then prints just the second and last entry on each line.

8. Examine the contributions to the score for the lowest energy backbone restrained structure.

```
~/rosetta_workshop/rosetta/main/source/bin/residue_energy_breakdown.linuxgccrelease \
-s bb_1QYS_A_0001_0001.pdb -out:file:silent relax_breakdown.sc
```

   1. Open the relax_breakdown.sc file with OpenOffice as before, and examine the energies. Which residue pair has the highest fa_rep score? How does that compare numerically to those for the structure before relax? Which residue has the worst fa_dun score? How does that compare numerically to those before relax?

5. Computing other metrics.

   There are a number of other metrics which can be calculated on proteins besides scores. Many protocols will add relevant metrics to the scorefile that is output. Alternatively, you can calculate the metrics afterwards using analysis applications. The following are a few examples.

   1. Evaluating interactions. The InterfaceAnalyzer application https://www.rosettacommons.org/docs/latest/application_documentation/analysis/interface-analyzer allows you to evaluation the interaction across an interface, for example, in protein-protein interactions.

```
~/rosetta_workshop/rosetta/main/source/bin/InterfaceAnalyzer.linuxgccrelease \
-s 3R2X_ignorechain_0001.pdb -interface AB_C -out:file:score_only interface.sc \
-compute_packstat
```

   2. Custom metric calcuation. The RosettaScripts framework can be used as a custom metric calculation tool. Most RosettaScripts filters have an associated metric. (See https://www.rosettacommons.org/docs/latest/scripting_documentation/RosettaScripts/Filters/Filters-RosettaScripts) If the filter is used in the main protocol block, the calculated value for the final output PDB will be placed in the scorefile as the name of the filter. Adding "confidence=0" to the filters prevents them from acting as filters, leaving them with just their calculation function.

      - We'll talk about RosettaScripts more later in the course.
      - **metrics.xml** and **4TQ5.pdb** as well as **4TQ5_silent.out** are provided for you in the inputs directory.

```
~/rosetta_workshop/rosetta/main/source/bin/rosetta_scripts.linuxgccrelease \
-in:file:silent 4TQ5_silent.out -parser:protocol metrics.xml -in:file:native 4TQ5.pdb \
-out:file:score_only metrics.sc
```