

Assignment 1 – Protein Structure and Visualization

BIOE/BIOMEDIN/BIOPHYS/CME/CS 279

Due: October 15, 2019 at 3:00 PM

The goal of this assignment is to familiarize yourself with the visualization software PyMOL and the basics of protein structure.

Acknowledgements: Portions of this assignment are based off of the PyRosetta Tutorials.

1 Preliminaries

- Download the assignment zip file from the course website. (<http://web.stanford.edu/class/cs279/index.html#hw>)
- Make sure you are enrolled in CS279 on canvas (<https://canvas.stanford.edu/>) and gradescope (<https://www.gradescope.com/>). You should have received an invitation via email to join both websites. Please contact the CAs before the due date if you do not have access.
- If you wish to work on a VPTL computer, please be sure to use a non-multimedia Mac machine. If you do this, you may skip the instructions for installing PyMol and installing and using Anaconda. Just use the version of python already installed on the machines.
- Download and install PyMOL (<https://www.pymol.org/>) on your computer. If you already have PyMOL installed, check to make sure you have the most recent version – PyMOL 2.3. If you cannot get PyMOL to run on your computer, please contact the CAs.
- Download and install Anaconda (<https://www.anaconda.com/distribution>). Anaconda is a platform for python and R that simplifies managing environment and installing packages. Select the correct installer for your operating system, and install the python 3.7 version (although we will set the environment to use python 2.7). If you already have python 2.7 and matplotlib installed, you can skip this step, but Anaconda will be required for assignment 3.

2 Visualizing Proteins

To start, we'll use the program PyMOL to visualize the molecular structure of some peptides and folded proteins. PyMOL allows you to execute instructions through their GUI or through their built-in command line.

First, unzip the contents of the assignment, then open PyMOL. In the command line, change to the assignment directory.

```
PyMOL> cd <path to assignment directory>/pdbs
PyMOL> load hras.pdb
```

When you load a structure, the default visualization is an all-atom depiction. While there are times when this is necessary/helpful, frequently, we want a coarser view of the protein. In particular, the **cartoon** view allows us to quickly learn about the secondary structures that are present in the folded 3D conformation.

To access this mode, we will hide the current depiction and then show the cartoon depiction. These commands are available through the GUI buttons in the right-most panel. In the row of the molecule we wish to manipulate (in this case **hras**), select the following commands.

H → everything S → cartoon

If you prefer, you can also control the depiction from the command line.

```
PyMOL> hide everything, hras
PyMOL> show cartoon, hras
```

In this visual mode, alpha helices are depicted by twisting coils and beta strands are depicted as fat arrows.

Question 1:

- (a) *How many sections of hras are folded as an alpha helix?*
- (b) *How many strands of hras are folded as a beta strand?*

Note: You may find it useful to color by secondary structure.

C → by ss → Helix Sheet Loop

Alternatively, you can use the command:

```
color red, ss h; color yellow, ss s
```

2.1 Backbone Geometry

Next, we will take a closer look at the typical secondary structures, the alpha helix and beta sheets. Start by reinitializing PyMOL and then running the following commands.

```
PyMOL> reinit
PyMOL> load helix.pdb
PyMOL> hide everything
PyMOL> show cartoon
```

Question 2: *Looking down the helix and moving away from the viewer, which direction is the helix coiled (clockwise or counterclockwise)?*

The geometry of an alpha helix (and a beta strand) refers to a specific orientation of the atoms in the polypeptide backbone. In general, the orientation of the backbone atoms tells us a lot about the overall structure of the protein. To select just the backbone atoms, execute the following commands.

```
PyMOL> hide everything
PyMOL> select bb, name c+o+n+ca
PyMOL> show lines, bb
```

Note: “bb” is not a keyword in this command. The PyMol selection syntax is

```
select <name to be give selection>, selection
```

You should see the backbone atoms represented as connected lines, with carbons in green, oxygens in red, and nitrogens in blue.

Here we are selecting all atoms named ‘n’, ‘o’, ‘c’, or ‘ca’.

Question 3: *Of these four choices:*

- (a) *Which atom(s) is/are bonded to the side chain?*
- (b) *Which atom(s) is/are bonded to a hydrogen?*
- (c) *Which atom(s) is/are the hydrogen bond acceptor(s)?*

The backbone geometry can be succinctly described by the *dihedral angles* (also known as torsional angles). In general, a dihedral angle is an angle between two planes. In the context of backbone molecular geometry, we are concerned with the angle between atoms while “looking down” a bond.

Question 4: We will be looking at residue 159. Which amino acid does this correspond to?

(a) Compute the dihedral angle ϕ_{159} (between $C_{i-1} - N_i - C_i^\alpha - C_i$).

(b) Compute the dihedral angle ψ_{159} (between $N_i - C_i^\alpha - C_i - N_{i+1}$).

(c) Compute the dihedral angle ω_{159} (between $C_i^\alpha - C_i - N_{i+1} - C_{i+1}^\alpha$).

Select the atoms in the correct order, and then use the following command to compute the appropriate dihedral angles.

```
PyMOL> get_dihedral (pk1), (pk2), (pk3), (pk4)
```

Note: Use the `pkAt` mouse command (by default double-clicking with the right click).

Alternatively, PyMol has a powerful atom selection language. You can select by residue number using

```
PyMOL> select resid 159
```

and by atom name using

```
PyMOL> select name ca
```

Combining these into one command selects the C^α in residue 159 using

```
PyMOL> select resid 159 and name ca
```

or the equivalent short form

```
PyMOL> select 159/ca
```

You can then combine these selections to compute a dihedral

```
PyMOL> get_dihedral resid1/name1, resid2/name2, resid3/name3,  
resid4/name4
```

Next, perform the same computations for the beta sheet segment (contained in `pdbs/bstrand.pdb`).

Question 5: *Are the beta-strands running parallel or antiparallel to one another?*

Question 6:

(a) Compute ϕ_{41} .

(b) Compute ψ_{41} .

(c) Compute ω_{41} .

Now that we have computed the dihedral angles for two amino acid residues, it is natural to ask whether these measurements are typical. In fact, one way of characterizing proteins' secondary structures is by their ϕ and ψ angles.

In particular, one way to represent the distribution of backbone dihedral angles is a Ramachandran plot. Ramachandran plots are two-dimensional heat maps, which represent the number of residues present for a given ϕ and ψ .

Run the following commands to generate a Ramachandran plot for the alpha helices, the beta strands, and full hras protein.

```
$> python Ramachandran_Plot.py pdbs/helices.pdb
$> python Ramachandran_Plot.py pdbs/bstrands.pdb
$> python Ramachandran_Plot.py pdbs/hras.pdb
```

Question 7: *Include your plots in your submission, and identify the regions in the plot for hras that corresponds to alpha helices and beta strands.*

Question 8: *Why are ω angles not used in Ramachandran plots?*

Hint: How do the typical values for ω angles differ from the typical values for ϕ and ψ angles? If you are unsure, choose a couple residues from `helices.pdb` and calculate the angles!

2.2 Hydrogen Bonds

A *hydrogen bond* is an attractive force between molecular dipoles involving hydrogen. While the hydrogen atoms are not drawn in the backbone (because they were not resolved in the experimental structure), there is a partially positive hydrogen bonded to (almost) every nitrogen involved in the peptide backbone, which tends to be attracted to the partially negative oxygens in the backbone. To visualize these “bonds”, we can measure the lengths between groups of atoms, specifically where polar contacts occur. First, we will add hydrogens into the structure in PyMOL and then show the backbone H-bond distances. Reinitialize PyMOL, and load `helix.pdb` from the `pdbs` subdirectory in the `asn1` folder, then enter the following commands.

```
PyMOL> hide everything
PyMOL> select bb, name c+o+n+ca
PyMOL> show lines, bb
PyMOL> h_add
PyMOL> distance hbonds, name o, name n, mode=2
```

Note: “hbonds” is not a keyword in this command, i.e. you could replace it with “foo” and see the same results. The specifier `mode=2` selects polar contact distances.

Question 9: *Examine the H-bonds that have been added to the alpha helix. By default, the distance command has an extremely liberal hydrogen bond angle cutoff. We generally only consider hydrogen bonds if, among other constraints, the hydrogen bond donor (the atom the hydrogen is covalently bound to), the hydrogen, and the hydrogen bond acceptor are approximately in line. For the alpha helix, the correct hydrogen bonds are the ones parallel to the overall direction of the helix (see image below).*

- (a) *How many H-bonds fit this criterion?*
- (b) *What is the **integer** offset between residues that are interacting (i.e. if two residues share a H-bond in an alpha helix, what are their positions in the protein backbone relative to each other)?*

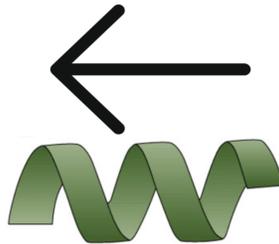


Figure 1: “Good” hydrogen bonds are roughly parallel to the overall direction of the helix.

2.3 Statistics of non-bonded interactions

As you saw (or will see) in lecture, a major class of energy functions, called knowledge-based or statistical potentials, are typically created using patterns observed in solved crystal structures, as opposed to physics-based models. These potentials rely on the simple hypothesis that, on average, atoms in structures will be positioned in energetically favorable conformations. For example, earlier in this problem set, you saw that the ϕ and ψ angles often favor particular values. This information could be used to create an energy function that favored combinations of angles that are observed most often.

In this problem, we will examine statistics of specific non-bonded interactions. In particular, we will generate statistics about the distance between positive and negatively charged residues as compared to positive and neutrally charged residues. We have provided a small set of randomly

selected structures from the PDB to use as training data, as well as code to parse the files. While our analysis will lack the rigor and completeness needed to create the most widely used knowledge-based potentials, we hope that this problem will show you the intuition behind knowledge-based potentials and serve as a gentle introduction to Python programming.

If you are using Anaconda (recommended), run the following commands to import a python environment named ‘cs279’ from the provided `environment.yml` file, and then activate the environment.

```
$ conda env create -f environment.yml
$ conda activate cs279
```

When you’re done working with python on the command line, run:

```
$ conda deactivate
```

Whenever you want to use the environment again, simply run the ‘conda activate’ command. The environment never needs to be created again.

Question 10: *Open the file `nonbonded_distances.py` in your favorite text editor (think `emacs`, `vim`, `sublime text`). Examine the contents of the file, first concentrating on the code and comments about overall workflow at the bottom, then moving on to the functions at the top. The only thing you need to do is fill in the function “`get_minimum_distances`”. When you are ready, you can run the script by opening your terminal, changing to the `asn1` directory and executing*

```
python nonbonded_distances.py nonbonded_pdbs/*
```

The script will create a file “`nonbonded_dists.png`” containing a histogram of the minimum distance from each positively charged nitrogen in arginine or lysine in the set of structures to the closest negatively charged oxygen in aspartate or glutamate, as compared to the neutrally charged terminal carbons in valine and leucine. Include this histogram in your submission.

Do the results agree with your intuition? Please explain.

2.4 Mutations

We will now become familiar with the mutagenesis wizard in PyMOL. In class, we covered how proteins can take on different conformations, in large part by adopting different torsional angles, and how certain conformations are favored because they minimize steric clashes and maximize favorable interactions. Torsional angle rotation can occur along bonds on the protein backbone or bonds of amino acid side chains. The different conformations that a side chain can adopt are called “rotamers.” The mutagenesis wizard allows you to make mutations to protein residues and visualize possible resulting rotamers. It is a useful tool to assist with protein engineering tasks.

We will first use the mutagenesis wizard to visualize how it can be a useful tool to visualize rotamers of existing residues. We will use the alpha helix structure and examine how different rotamers of Leu159 interact with the protein backbone. Load `helix.pdb` and then enter the following commands in PyMOL.

```
PyMol> hide everything
PyMol> select bb, name c+o+n+ca
PyMol> show lines, bb
PyMol> h_add
PyMol> wizard mutagenesis
PyMol> set cartoon_side_chain_helper, on
PyMol> show sticks, resi 159
```

You should now be able to see the hydrogen backbone and Leu159 clearly.

Question 11: *Now, navigate to Wizard → Mutagenesis → Protein. You should see the mutagenesis wizard displayed to the right. Click on Leu159 and investigate the different possible rotamers of Leu159 itself without making any mutations, i.e. leave ‘No Mutation’ selected in the mutation wizard. You should be able to see clashes between residues visualized with red circles.*

- (a) *How many rotamers of Leu159 are there? What are their strains?*
- (b) *What other amino acid residue(s) clash with rotamer 1? Display the residue(s) as sticks and include a screenshot with the rotamer 1 clash. With which atom(s) does the clash occur?*

Note: The rotamer strain refers to the conformational strain introduced by the rotamers clashing with the protein backbone.

We will now more deeply investigate the green fluorescent protein (GFP), a protein that emits green light when stimulated with blue light. GFP is from the Pacific Northwest jellyfish *Aequorea victoria* and is used as a fluorescence marker in experiments, often to tag and visualize different proteins of interest.

Let us examine the structure of GFP. The PDB identifier for GFP is 1EMA; pull GFP into PyMOL by using the fetch command. Enter the following:

```
PyMol> reinit
PyMol> fetch 1EMA
PyMol> remove resn hoh
```

Question 12: *You should now see GFP in PyMOL. Describe the structure in terms of the secondary structures you have learned – be sure to mention the number of each type of secondary structure.*

Run the following commands to get a better view of the inner part of the molecule:

```
PyMol> hide cartoon, resi 1-50
```

The fused rings you see in the center of the GFP structure are referred to as the chromophore; interactions between the chromophore and its local environment contribute to GFP’s fluorescence. The chromophore in GFP is formed by the fusion of the residues Thr65-Tyr66-Gly67. We will now use the mutagenesis wizard to make a mutation that changes the interactions of the chromophore with its local environment. This causes the chromophore fluorescence to become red-shifted, that is, the emitted light has a longer wavelength and becomes more red-colored. To do this, we will mutate Thr203 to Tyr.

Enter the following commands in PyMOL (note that the mutagenesis wizard can also be accessed

from the command line):

```
PyMol> show sticks, resi 203
PyMol> color slate, resi 203
PyMol> wizard mutagenesis
PyMol> set cartoon_side_chain_helper, on
```

Thr203 should now be clearly visible in a different color.

Question 13: *Click on Thr203. Now, mutate it to Tyr using the mutagenesis wizard.*

- (a) *How many rotamers are there? What are the strains for each one?*
- (b) *Structurally, how are the interactions with the chromophore different between the lowest and highest strain rotamers? Click "apply" to incorporate the lowest strain mutation and include a screenshot.*

3 Feedback

You will receive full-credit for this portion for providing any response. We encourage constructive criticism.

Question 14: *What was your favorite aspect of the assignment? What was your least favorite aspect of the assignment? Why? Any suggestions for improvement?*

Question 15: *Approximately how long did this assignment take you? Where did you spend most of this time?*

4 Challenge Question

This question is intentionally more challenging than the rest of the homework. It is not required, and you can receive full credit for the assignment without attempting it. We will never deduct points for a challenge question submission and will award extra credit depending on the quality of your solution.

Question 16: *Write a Python script that computes the average solvent accessible surface area (SASA) for each residue type in a PDB file. Create a bar chart graphing average SASA for each type of amino acid in GFP (PDB code 1EMA). What trends do you notice for charged vs. noncharged residues? Notice that the size of an amino acid will also affect its SASA. Come up with a strategy to account for different residue sizes and plot an adjusted average SASA per residue as before. What differences in SASA do you notice after factoring in a residue's size?*

Look at the PyMol documentation for tools to calculate SASA and for how to turn Python scripts into custom PyMol commands that can be executed from the PyMol command line.

5 Submission Instructions

Please submit this homework on gradescope (<https://www.gradescope.com/>). Type up your answers in the text editor of your choice, making sure to clearly label each question, then convert the entire document to a pdf.

Go to gradescope, navigate to CS279, 'Assignments', then the endpoint for Assignment 1. Choose the option to upload a pdf and **tag each question with the corresponding page**. You do not need to separate each question on its own page.

You do not need to submit your code for the main assignment. Include your results and written explanation of the challenge problem, if you choose to tackle it, in your main pdf. **Please submit your code for the challenge problem on canvas**. Let the course staff know if you have any issues.