Soham Govande

Freshman, Class of 2026

CS 279, Professor Ron Dror

Fall 2022

**Assignment 1**

**Question 1**

**(a) How many separate alpha helices does HRas have?**

HRas has 5 separate alpha helices.

**(b) How many separate beta strands does HRas have?**

HRas has 6 separate beta strands.

**Question 2**

**Orient the structure so that you are looking through the helix (imagine looking through a telescope). Which direction (clockwise or counterclockwise) is the helix coiled as it extends away from you?**

As the helix extends away from me into the computer screen, it is coiled clockwise.

**Question 3**

A) Residue #159 corresponds to leucine.

B) -55.071 degrees

C) -43.672 degrees

D) 179.630 degrees

**Question 4**

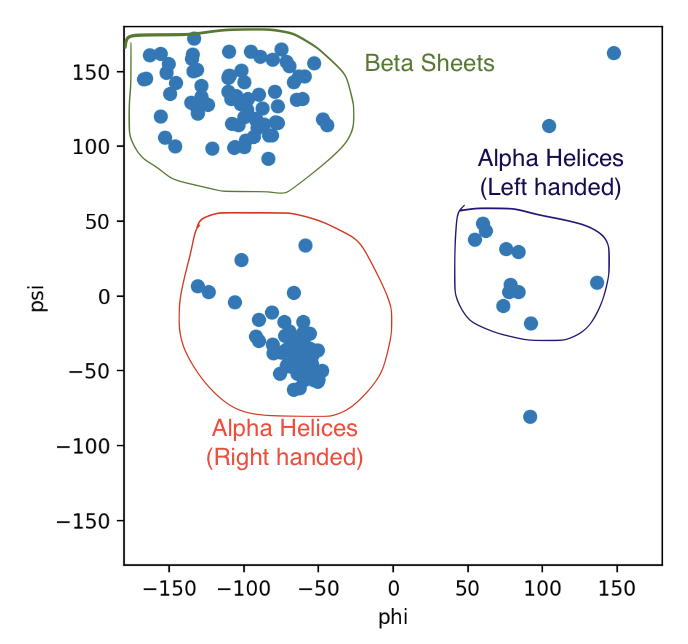
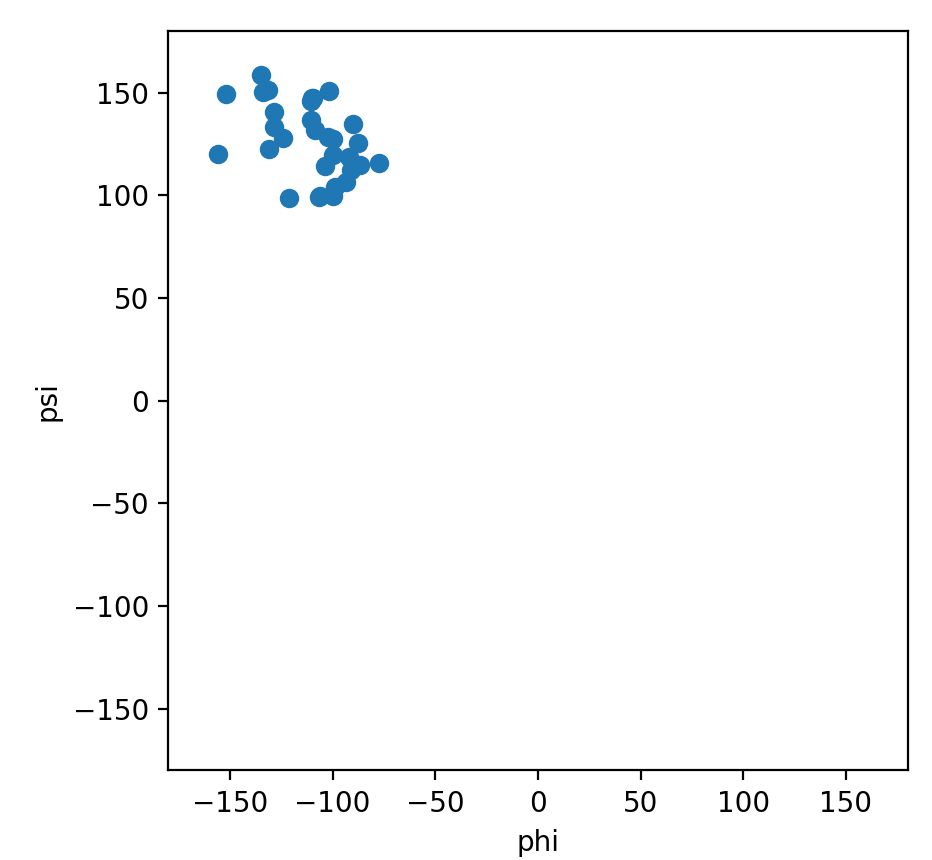
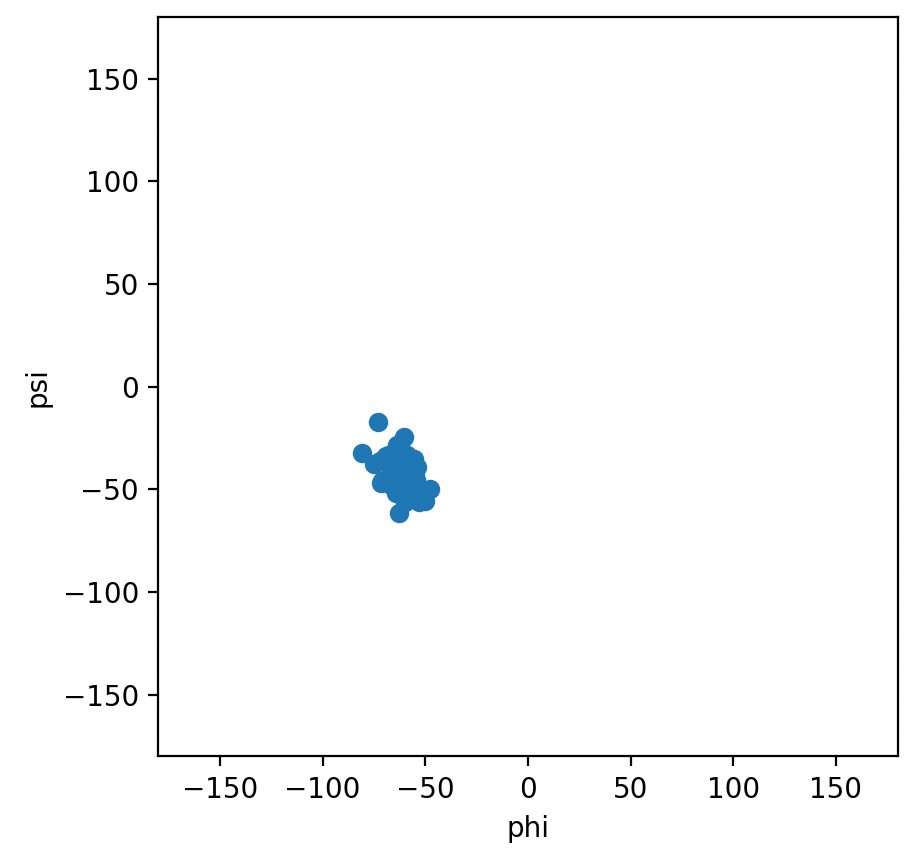
A) -110.272 degrees

B) 147.630 degrees

C) 179.745 degrees

**Question 5**

Plots (in order) are alpha helices, beta sheets, and HRAS.



**Question 6**

We don’t plot ω angles in Ramachandran because almost always, ω is equal to 180 degrees. In certain rare cases of amino acids, though, ω can equal 0, but this behavior is sporadic.

**Question 7**

RNA: Red; DNA: Green

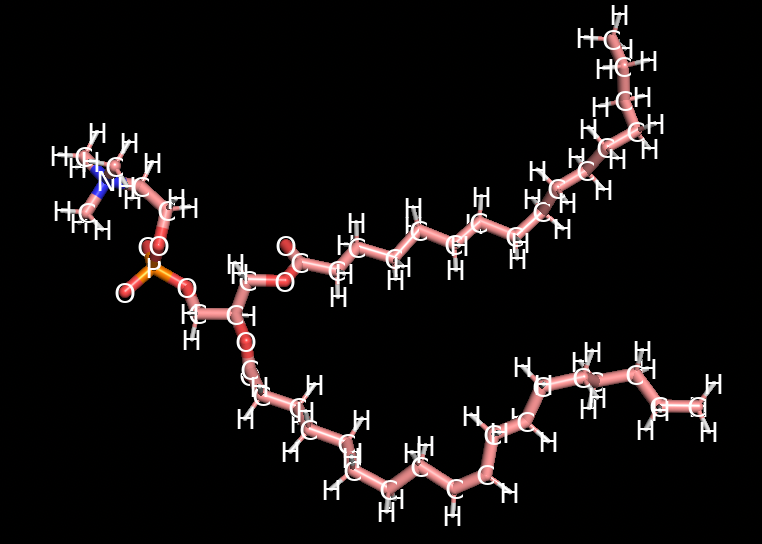


**Question 8**

A

**Question 9**

B2AR, a nanobody, BI-16710, Na+, Cl-, H2O, Phospholipid (C42H82O8NP - screenshot below)



**Question 10**

Broadly speaking, the positively-charged residues are spatially distributed on both sides of chain L (the phospholipid bilayer), since the water surrounding the bilayer is highly polar; the positively-charged residues are attracted to the polar water molecules. The nonpolar residues are largely concentrated on the inner part of the bilayer (the nonpolar lipids). This helps to stabilize B2AR in the bilayer because polar molecules are attracted to other polar molecules, and nonpolar molecules are attracted to other nonpolar molecules, which holds the B2AR molecule in place.

If the B2AR molecule were turned on its side, the entire molecule would be inside of the membrane. Here, the hydrophobic regions of B2AR would interact with the hydrophobic interior of the membrane, but the hydrophilic regions of B2AR would be unstable because of the hydrophobic surroundings. Similarly, if the B2AR molecule were removed from the membrane, the molecule would lie in a hydrophilic sea of H2O molecules. The hydrophilic regions of B2AR would be stable, but the hydrophobic regions of B2AR would be unstable. Both of these scenarios would result in the B2AR protein molecule denaturing and deforming, which highlights the role of the membrane-protein interactions in stabilizing the molecule.

**Question 11**

Moving outwards creates space for the nanobody to bind because it’s a small molecule that fits in the crevice created.

**Question 12**

The active-state agonist has a ligand that is more outwardly projected due to an extra hydrogen hexagonal ring, compared to the ligand of the inactive-state antagonist. This means that the position of residue 207 is able to penetrate deeper in the active-state molecule compared to the inactive molecule. Thus, the agonist is able to form hydrogen bonds with the ligand receptor, whereas the antagonist is unable to form these bonds.

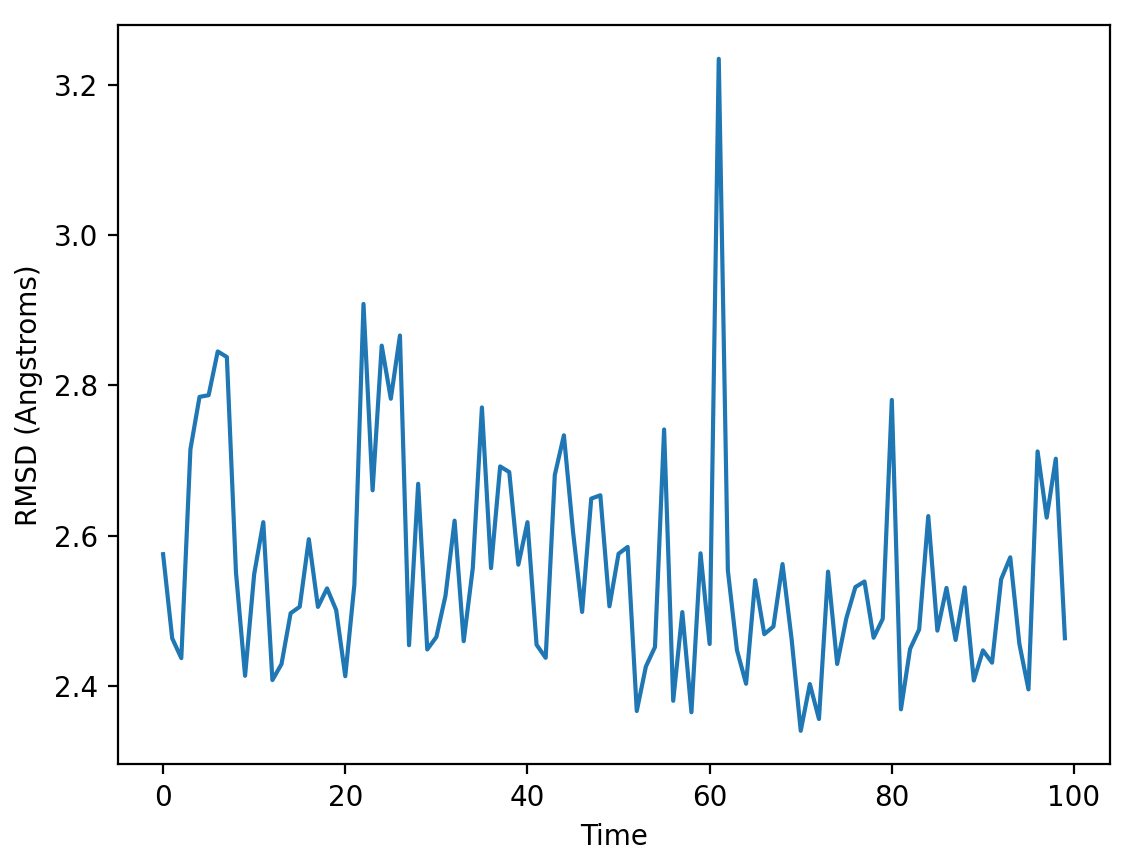
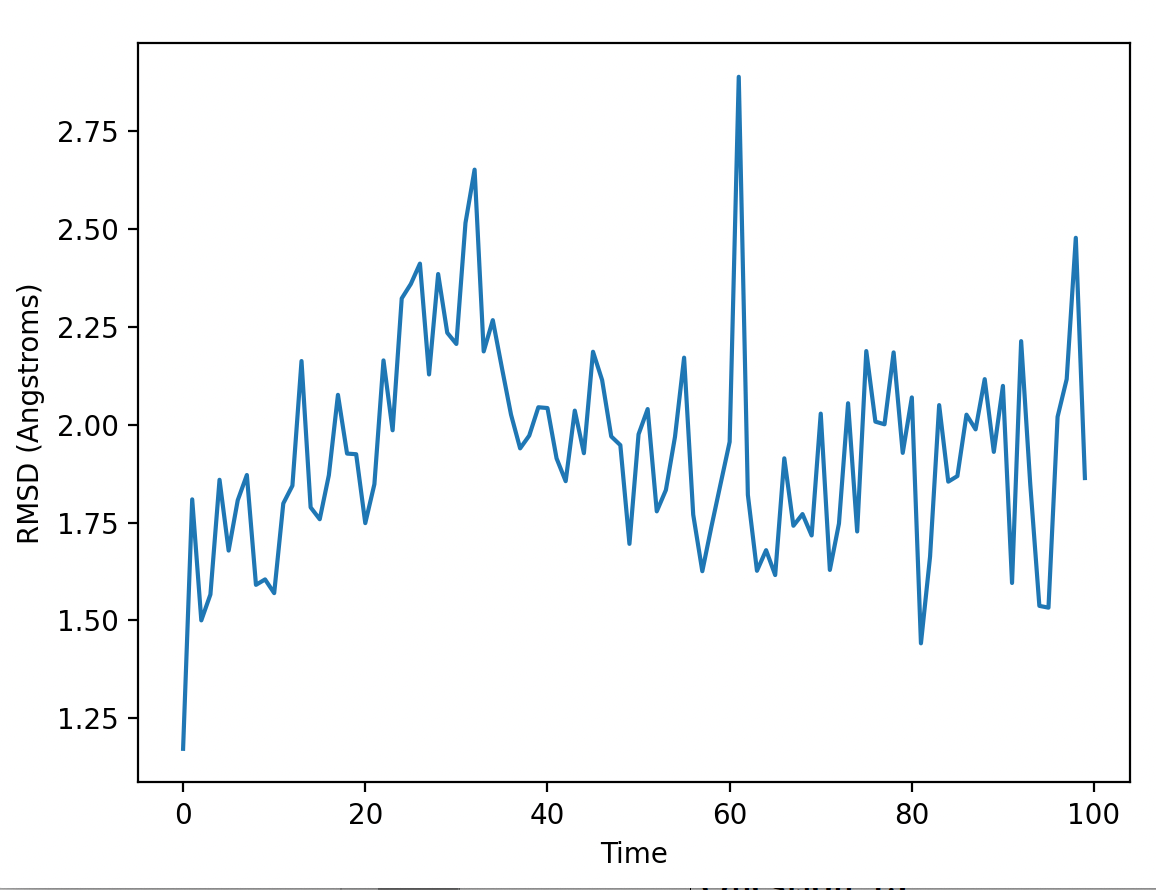
**Question 13**

The b2ar\_nanobody\_free simulation goes towards the inactive state.

**Question 14**

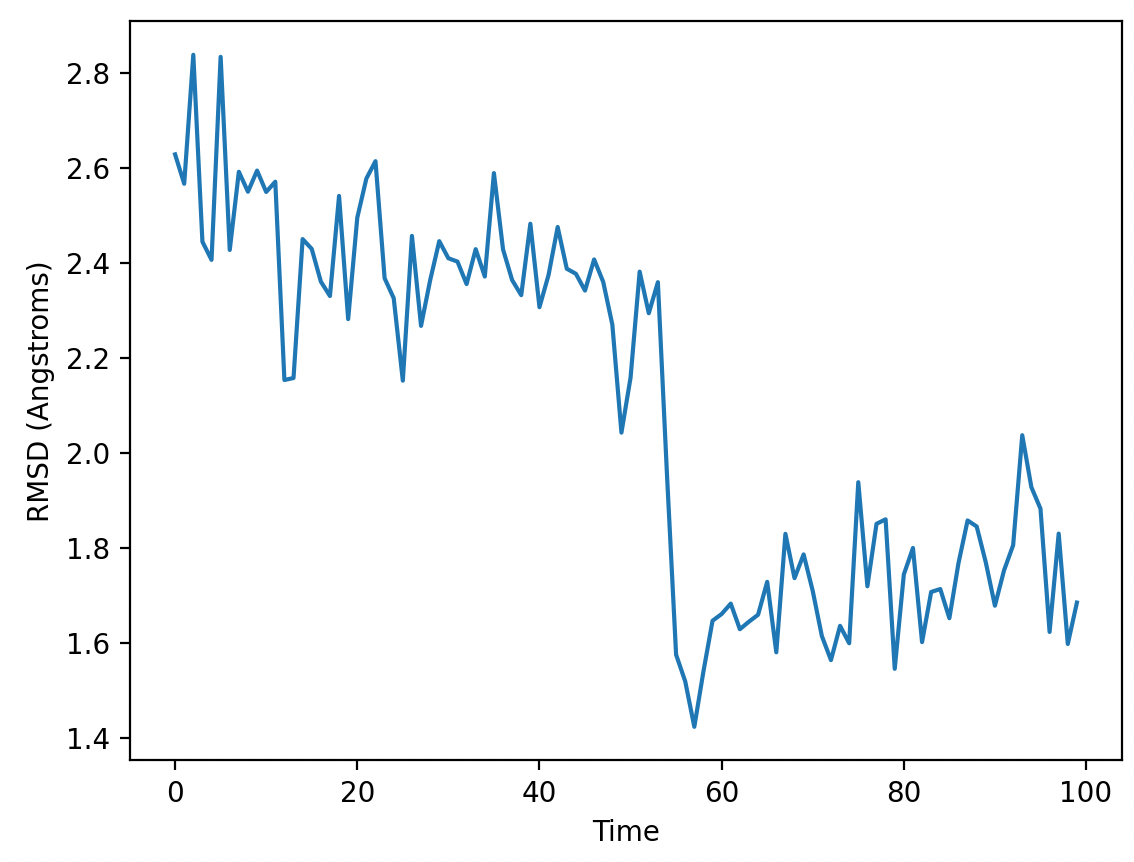
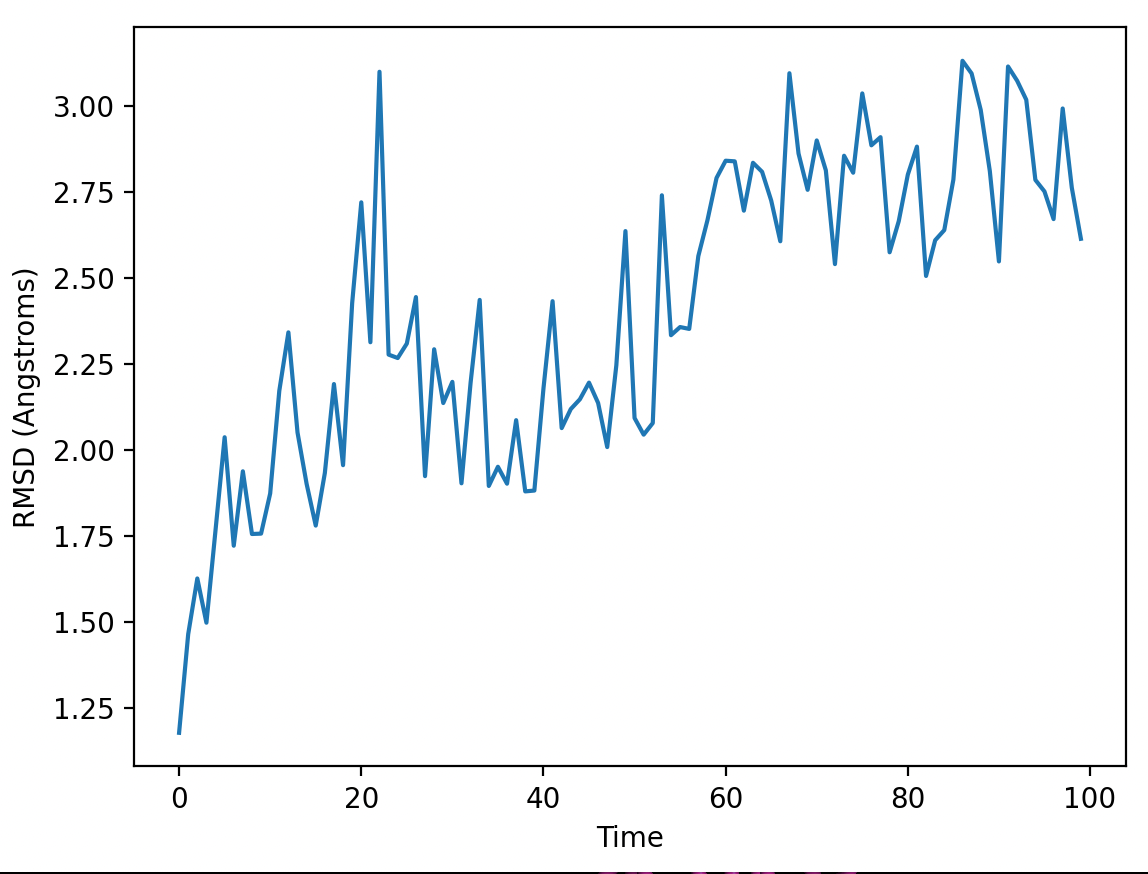
A) Nanobody-bound

| Active State | Inactive State |
| --- | --- |

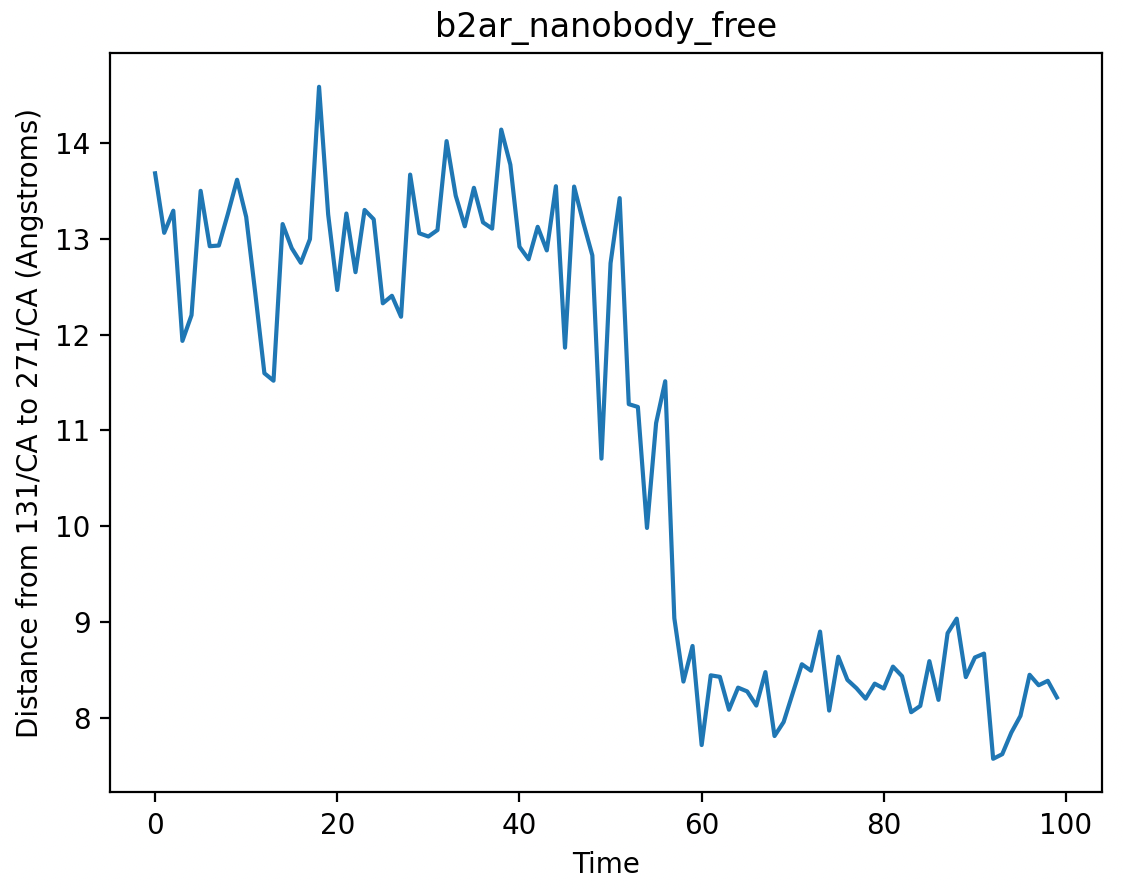
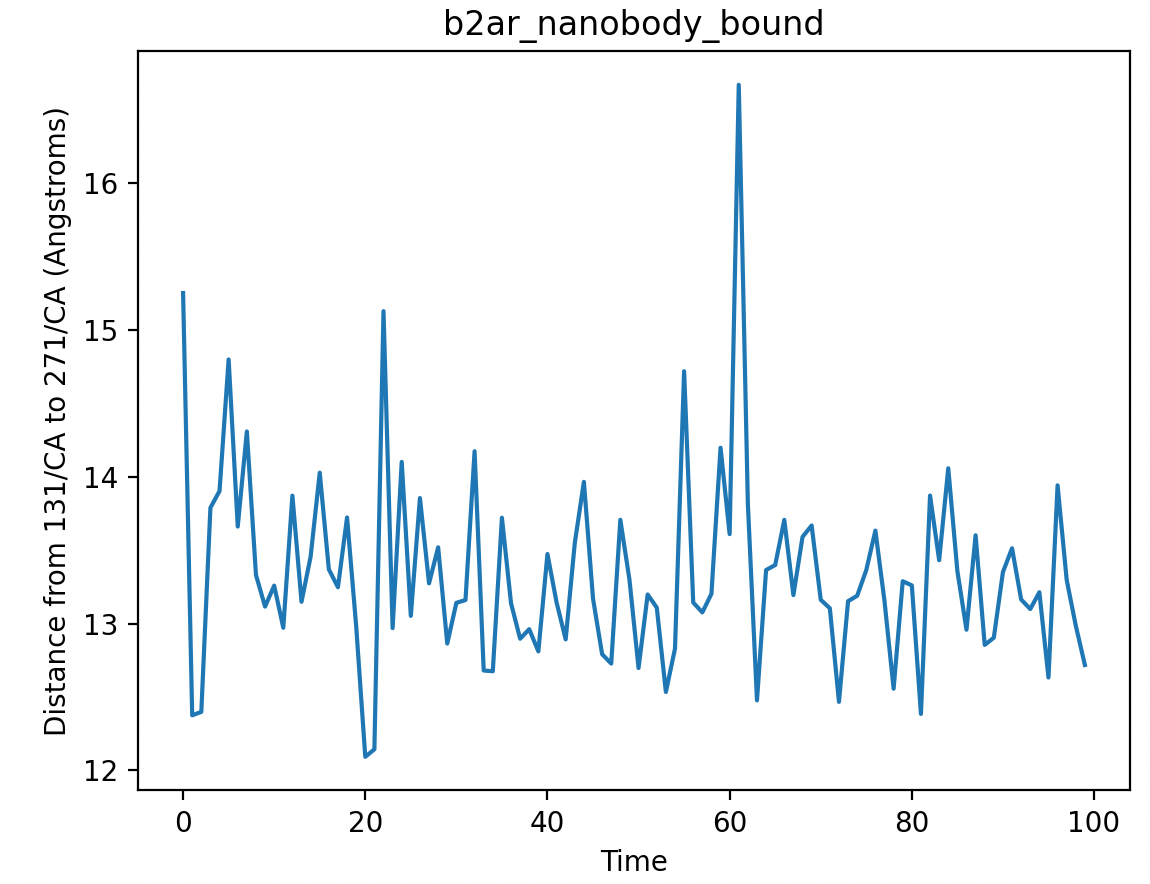


B) Nanobody-free

| Active State | Inactive State |
| --- | --- |



**Question 15**



Generally, these two graphs agree with the graphs created in question 14. However, these distance plots capture the distance difference at a much more specific scale (at the level of our residue of interest), which can be valuable to answer questions like, does this particular residue agree with the general RMSD trend? Are there any unexpected changes in the position of this particular residue?

**Question 16**

When the nanobody is not present, the B2AR protein gradually transitions towards the inactive state, since this macrostate has less free energy than the active macrostate. When the nanobody is present, the B2AR protein remains in the active state, which means that the active macrostate has less free energy than the inactive macrostate.

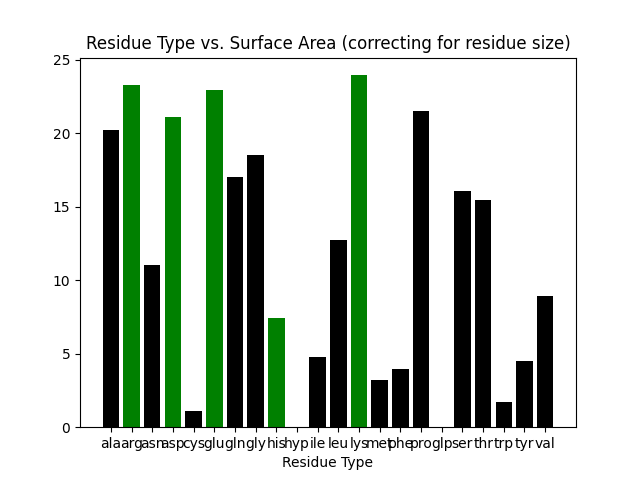
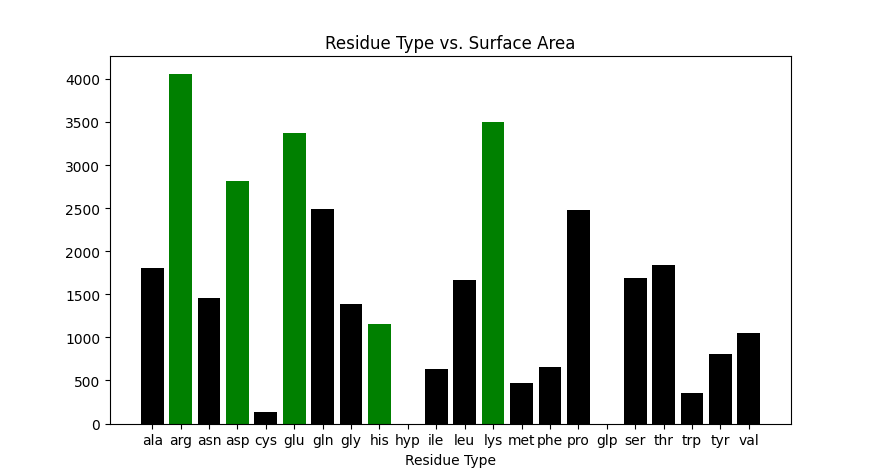
**Question 17**

My favorite part of this assignment was doing the B2AR timestep simulation, since it was really interesting to watch the gradual change of the protein towards the inactive state when there is no nanobody. My least favorite part of this assignment was figuring out how to do the Ramachandran plots, since my Python code was erroring. I had to add a line to ensure that the residue numbers were contained in the array, in order to be able to graph a full Ramachandran plot of all residues. To improve the assignment, I would add more description on how to debug PyMol errors, especially “Error: Selection 1 not found” since I experienced that (as well as a friend in this class).

**Question 18**

Question 5 took the longest time for me because my code for Ramachandran plots was not working as intended—I was getting a PyMol error. After fixing this bug, the rest of the assignment was straightforward.

**Question 19**



Left: Uncorrected for residue size; Right: adjusted for residue size

Legend: **Green** - Charged residue; **Black** - Uncharged residue (charged residues were obtained from [this online list](https://www.imgt.org/IMGTeducation/Aide-memoire/_UK/aminoacids/charge/))

Overall, charged residues have a much larger SASA than non-charged residues; 4 out of the top 5 SASA values belong to charged residues. This makes sense because charged residues experience electrostatic attraction to the polar solvent molecules (water), giving them a greater SASA.

I decided to approximate residue size with molecular weight, obtained from [this table](https://www.sigmaaldrich.com/US/en/technical-documents/technical-article/protein-biology/protein-structural-analysis/amino-acid-reference-chart). Although molecular weight isn’t a precise measurement of size, generally, it can be used to provide a good estimate of which residues contain more molecules than others. After factoring in residue size (dividing the raw SASA measurements by molecular weight), I notice that the trend remains: charged residues have a greater SASA than non-charged residues. However, the range of values has been drastically reduced, and non-charged residues have a greater adjusted surface area.

**Code**

**Code for Ramachandran Plot**

**def** ramachandran(sel):

"""

Produce a Ramachandran plot for residues in the given selection.

"""

cmd.delete('phi')

cmd.delete('psi')

resnums = get\_residue\_numbers(sel)

phis, psis = [], []

# For each residue in resnums, add the phi and psi angle to phis and psis, respectively.

# PyMol has two commands related to dihedral angles:

# cmd.dihedral(name, sel1, sel2, sel3, sel4) will plot the dihedral on the

# structure however it will not return the value.

# cmd.get\_dihedral(sel1, sel2, sel3, sel4) will return the dihedral but will

# not show it on the structure.

# Only cmd.get\_dihedral is strictly required in your implementation, but

# we highly recommend that you call both commands with the same selections

# so that you can visually see the angles for debugging purposes.

# Note that cmd.dihedral has an additional ``name'' argument, which you

# should set to "phi" for the phi angles and psi for the "psi" angles.

# Some tips on what various error messages mean:

# "Error: Selection 1: Not found": The first selection matches no atoms.

# "Error: Selection 1: Invalid selection name": The first selection matches multiple atoms.

# Equivalent messages for Selection 2 mean the second selection is invalid, and so on.

############################################################################

# Edit here.

valid\_resnums = get\_residue\_numbers('resid \*')

**for** resnum **in** resnums:

# We need to check to ensure that there exists a residue 1 before and 1 after the current residue

# Without this check, we get "Error: Selection 1: Not found"

**if** resnum-1 **in** valid\_resnums **and** resnum+1 **in** valid\_resnums:

# Calculate phi angle

cmd.select('pk1', str(resnum-1) + '/c')

cmd.select('pk2', str(resnum) + '/n')

cmd.select('pk3', str(resnum) + '/ca')

cmd.select('pk4', str(resnum) + '/c')

cmd.dihedral('phi', 'pk1', 'pk2', 'pk3', 'pk4')

phis.append(cmd.get\_dihedral('pk1', 'pk2', 'pk3', 'pk4'))

# Calculate psi angle

cmd.select('pk1', str(resnum) + '/n')

cmd.select('pk2', str(resnum) + '/ca')

cmd.select('pk3', str(resnum) + '/c')

cmd.select('pk4', str(resnum+1) + '/n')

cmd.dihedral('psi', 'pk1', 'pk2', 'pk3', 'pk4')

psis.append(cmd.get\_dihedral('pk1', 'pk2', 'pk3', 'pk4'))

############################################################################

plt.scatter(phis, psis)

plt.xlabel('phi')

plt.ylabel('psi')

plt.ylim(-180, 180)

plt.xlim(-180, 180)

plt.gca().set\_aspect('equal')

plt.show()

**Code for Assignment 1 Challenge Question**

**from** pymol **import** cmd

**import** matplotlib.pyplot **as** plt

**import** os

**import** numpy **as** np

**def** plot\_sasa():

files = os.listdir('sasa\_pdbs/')

# Amino acid codes used for searching

res\_names = ['ala', 'arg', 'asn', 'asp', 'cys', 'glu', 'gln', 'gly', 'his', 'hyp', 'ile', 'leu', 'lys', 'met', 'phe', 'pro', 'glp', 'ser', 'thr', 'trp', 'tyr', 'val']

# We approximate residue size with molecular weight; although this isn't precise it's a good estimate

res\_sizes = [89.10, 174.20, 132.12, 133.11, 121.16, 147.13, 146.15, 75.07, 155.16, 131.13, 131.18, 131.18, 146.19, 149.21, 165.19, 115.13, 139.11, 105.09, 119.12, 204.23, 181.19, 117.15]

# Calculate SASA instead of molecular surface area

cmd.set('dot\_solvent', 'on')

# The "data" dict stores SASA information on every amino acid

data = {}

**for** file **in** files:

# Temporarily store it as name "object"

cmd.load('sasa\_pdbs/' + file, 'obj')

# Add hydrogens

cmd.h\_add('obj')

# Count the surface area of each residue

size\_i=0

**for** res\_name **in** res\_names:

cmd.select('temp', 'resn ' + res\_name + ' and model obj')

**if** res\_name **not** **in** data.keys():

data[res\_name] = { "count": 0, "total": 0 }

data[res\_name]["count"] += 1

data[res\_name]["total"] += cmd.get\_area('temp')

data[res\_name]["size"] = res\_sizes[size\_i]

size\_i = size\_i + 1

# Free up memory

cmd.delete('obj')

avgs = []

lbls = []

colors = []

**for** res\_name **in** data.keys():

lbls.append(res\_name)

avgs.append(data[res\_name]["total"] / (data[res\_name]["count"] \* data[res\_name]["size"]))

**if** res\_name **in** ['asp', 'glu', 'lys', 'arg', 'his']:

colors.append('green')

**else**:

colors.append('black')

xaxis = np.arange(len(res\_names))

fig, ax = plt.subplots()

hbars = ax.bar(xaxis, avgs, color=colors)

ax.set\_xlabel('Residue Type')

ax.set\_title('Residue Type vs. Surface Area (correcting for residue size)')

ax.set\_xticks(xaxis, labels=lbls)

plt.show()

cmd.extend('plot\_sasa', plot\_sasa)