CS 483

Assignment 4: Protein Overlap and Surface Studies

Due dates: Thurs. Mar. 19. (Tues. Mar. 17 to get 5 bonus marks).

Read the questions carefully and **please start early!**

Exercise 1:

Motivation

In a paper by Shen, Davis, and Sali[1] a highly simplified sphere-packing model is used to suggest that the optimal size of a globular domain ranges from 117 to 213 residues with an average size of 165 residues. For humans the median length of a protein is 375 residues[2] (longer than 213 because of the formation of multi-domain configurations).

It has been observed that most domains have a residue count that is usually more than 100 if the hydrophobic residues are to be properly buried in a hydrophobic core. An interesting measure to assess burial effectiveness was devised by Spassov, Karshikoff, and Ladenstein[3] in a 1995 paper. The formula:

$$\xi^n = \frac{SA_f^n}{SA_u^n} = \frac{SA_f^n}{\sum_{i=1}^{n} sa_u^i}$$

is used to compare the solvent accessible area of the protein in the folded and unfolded states. The value of SA_f^n represents the solvent accessible area of the folded protein with all residues removed except for the first *n* residues. The value of sa_u^i represents the solvent accessible area of the *i*th residue when it is in an unfolded peptide. Values for sa_u^i can be found in the Spassov et al. paper and have been set up in the following dictionary for your convenience:

```
areaASAaxa_D = {"ALA": 113.0, "ARG": 241.0, "ASN": 158.0, "ASP": 151.0,
                "CYS": 140.0, "GLN": 189.0, "GLU": 183.0, "GLY":  85.0,
                "HIS": 194.0, "ILE": 182.0, "LEU": 180.0, "LYS": 211.0,
                "MET": 204.0, "PHE": 218.0, "PRO": 143.0, "SER": 122.0,
                "THR": 146.0, "TRP": 259.0, "TYR": 229.0, "VAL": 160.0}
```

The goal of this exercise is to plot the values of ξ^n versus *n* to get a curve similar to that seen on page 1522 of the paper by Spassov et al. My version can be seen in Figure 1 on the next page.

[1] M. Shen, F.P. Davis, and A. Sali. The optimal size of a globular protein domain: A simple sphere-packing model. *Chemical Physics Letters*, **405** (2005) 224-228.
[2] L. Brocchieri and S. Karlin. Protein length in eukaryotic and prokaryotic proteomes. *Nucleic Acids Research*, **33** (2005) 3390-3400.
[3] V. Z. Spassov, A.D. Karshikoff, and R. Ladenstein. The optimization of protein-solvent interactions: Thermostability and the role of hydrophobic and electrostatic interactions. *Protein Science*, **4** (1995) 1516-1527.

Since it will not be possible to steadily "build" the protein to get ξ^n for increasing values of *n*, we will evaluate ξ^n for the full protein and then calculate ξ^n for decreasing values of *n* within a loop that deletes the final residue of the chain for each pass through the loop. You can use append to collect the ξ^n values in a list that is reversed prior to the plotting activity.
Your script should go through the following steps:

- Fetch the protein and delete all atoms except for those in the chain that is to be processed.
- Use the `areaASAaxa_D` dictionary to compute SA_u^n for the entire protein.
- Use `runCommand("surface")` to put a surface around the entire protein. The model for the surface can be referenced as: `chimera.openModels.list()[1]` and the SA_f^n value can be derived as the `areaSAS` attribute of that model.
- In each pass through a loop you should delete the final residue and recalculate the new ξ^n value. Note: In Chimera, a protein object has a method called `deleteResidue`. The new value for SA_f^n can be derived by invoking the `update_surface()` method of the surface model. Do **not** execute `runCommand("surface")` again (this is very time consuming and unnecessary). The new value for the denominator can be derived by subtracting, from the current value of the denominator, the value of sa_u^i that corresponds to the residue that was just deleted.
- When the loop is finished, reverse the list used to collect the ratios and work with a plot routine to generate the plot. You can do the plot by modifying the script on page 372 of the text. Be sure to use `aspect = 'auto'` to avoid getting a plot that has almost no height for the y axis.

You should get a plot like this:

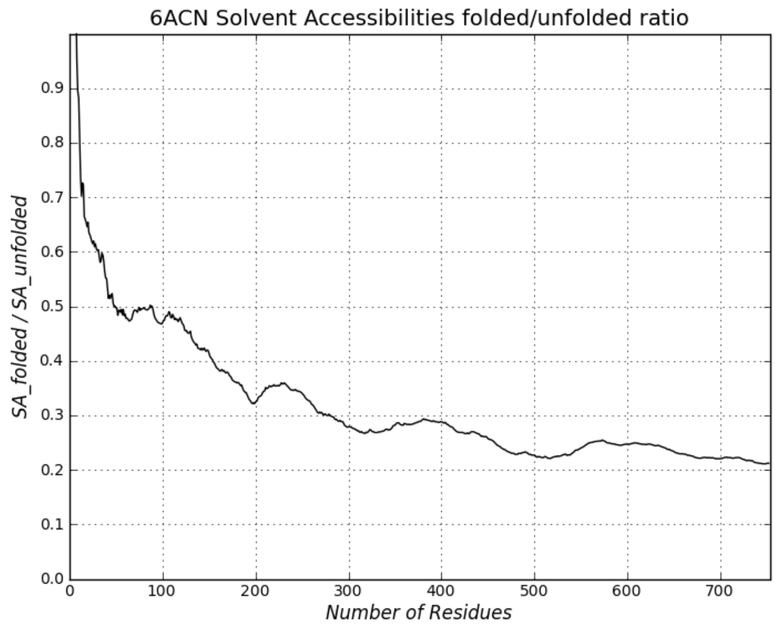


Figure 1

The significance of the plot
For very low values of *n*, the ratio is over 0.5 indicating a low level of hydrophobic residue burial as indicated by an overall solvent accessibility that is similar to that seen in the unfolded state. As the number of residues increases, there will be enough residues to form a hydrophobic core and the ratio drops but eventually approaches an asymptotic value that is approximately 0.2. It is worthwhile to point out that the aconitase protein (PDB ID = 6ACN) has four domains: D1 (residues 1–201), D2 (202 – 319), D3 (320 – 512), and D4 (537 – 754). The residues between positions 512 and 537 form a linker going from D3 to D4 and it is not considered to be part of a domain. By inspecting the graph you can see that for values of *n* just beyond the start of a domain we get a rise in the ξ^n ratio corresponding to a stretch of residues that are not long enough to develop a new hydrophobic core. As *n* gets larger, a new hydrophobic core develops and the curve moves down to values that are closer to 0.2. The aconitase protein was chosen for this study because each of the domains is comprised of a single stretch of residues (the peptide chain does not go into another domain and then return to the earlier domain).

Run your script on both 6ACN and 2B3Y. The latter protein is also an aconitase with domains D1 (residues 2–240), D2 (241 – 368), D3 (369 – 592), linker (593 – 654), and D4 (655 – 889).

Exercise 2:

Motivation

The fusion protein Abelson tyrosine kinase (c-Abl) has been implicated in the development of chronic myelogenous leukemia. Management of the cancer can usually be done by a drug therapy that involves inhibition of the c-Abl protein. The PDB contains several co-crystallizations of c-Abl with various drugs including imatinib sold under the trade name Gleevec. A very common procedure in drug design research involves the 3D alignment of two such protein structures so that we can do a comparison of the inhibitor conformations within the binding site. Because the overlap involves the same protein (only the ligand differs), the selection of alpha carbon pairs is relatively straightforward. Part (a) starts with the easiest case involving residues that have consistent position numbers. In Part (b) we require a sequence alignment to get the required alpha carbon pairs.

(a) In this exercise you will be dealing with four PDB files:

2HYY	(ligand: STI)	Use chain A
3CS9	(ligand: NIL)	Use chain A
2GQG	(ligand: 1N1)	Use chain A
1FPU	(ligand: PRC)	Use chain A

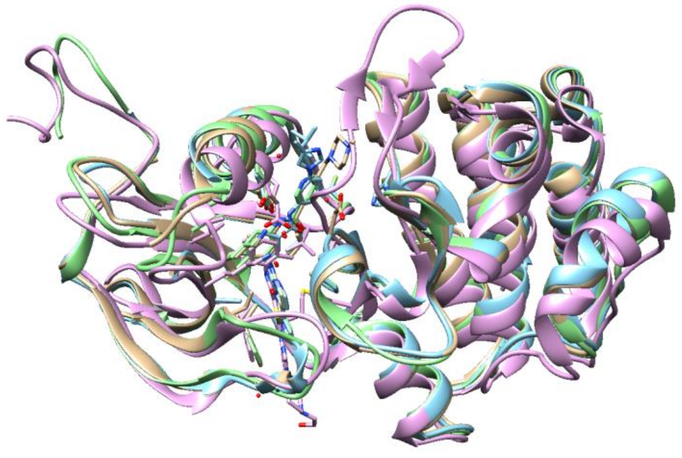
During development of the drug, the STI ligand was originally labeled STI-571 and after successful drug trials it was named imatinib. The NIL ligand refers to the drug nilotinib and the 1N1 ligand refers to the drug dasatinib. PRC is a modified version of STI.

Your script should go through the following steps:

1. Import the `Overlapper` class:
   `from StructBio.StructureOverlap.overlapper import Overlapper`
2. Fetch the four PDB files (2HYY, 3CS9, 2GQG, 1FPU).
3. Delete all chains that are not chain A.
4. Instantiate an `Overlapper` object for each of the three proteins: 3CS9, 2GQG, 1FPU.
5. Use the `moveToOverlap` method to bring each of these protein structures into a structural alignment with 2HYY. The alpha carbon atom pairs that determine the overlap will be in the residues with positions in the range [279, 498]. To construct a list of pairs for this method, consider using Python's `zip` function.
6. Ask the user for permission to proceed and, when the user responds with the Enter key, use the `undoMove` method to restore the three structures to their original positions.
7. Then repeat Step 5 with the alpha carbon pairs taken from residues in the binding site. For your convenience, here is a set containing the residue positions:
   `Set([248, 286, 290, 299, 313, 315, 317, 318, 370, 381, 382])`

Notes:

- The c-Abl protein presents difficulties for the crystallographer and consequently not all of the structure can be accurately determined. The rather high position numbers in Step 5 designate those residues for which the structure has been analyzed across all four co-crystallizations.
- You should notice that the completion of Step 5 does not necessarily give an accurate overlap for the binding site (especially for 2GQG which has a stretch of residues with a large backbone deviation that can be seen in the overlap with 2HYY). After Step 7 you should see the ligands in overlapped conformations that are more related to the structure of the binding site.
- After completion of Step 7 you should be able to work with Chimera's Model Panel to selectively show any subset of the overlapped proteins.
- After the overlap operations you can get the scene to be centered in the display by executing the statement:
  `chimera.viewer.viewAll()`
- Please reinstall the `StructBio` package from the StructuralBioinformatics website. The latest version corrects a program bug in the `Overlapper` class.



(b) In the previous Exercise, the overlap specification required some user knowledge, namely the residue range for the overlap and the positions of the residues in the binding site. In this exercise your script will derive this information by means of a sequence alignment. You will be dealing with five PDB files:

2HYY	(ligand: STI)	Use chain A
3CS9	(ligand: NIL)	Use chain A
2GQG	(ligand: 1N1)	Use chain A
1FPU	(ligand: PRC)	Use chain A
1OPJ	(ligand: STI)	Use chain A

The last file (1OPJ) uses the same ligand as 2HYY but it also has the MYR ligand which is important in c-Abl research. We did not use 1OPJ in Exercise 1(a) because the position numbers of the residues are not the same as those used in the other four files (they are shifted by 19 positions).

Your script should do the following:

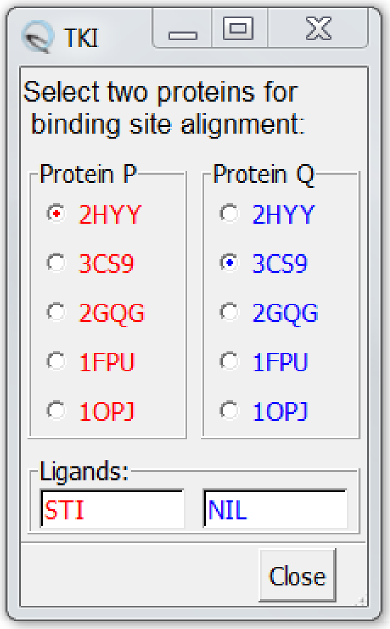
1. Import the following class definitions:
```
from StructBio.StructureOverlap.overlapper import Overlapper
from StructBio.StructureOverlap.sequencesLocalAlignment import SequencesLocalAlignment
from StructBio.Utilities.shellNeighbors import Shell
```
2. Fetch the five PDB files (2HYY, 3CS9, 2GQG, 1FPU, 1OPJ).
3. Delete all chains that are not chain A.
4. Instantiate an `Overlapper` object for each of the four proteins: 3CS9, 2GQG, 1FPU, and 1OPJ.
5. Use the `SequencesLocalAlignment` class to get a sequence alignment between the "target' protein 2HYY and each of the other "moving" proteins. Be careful: the instantiation of the `SequencesLocalAlignment` class requires two lists of residues each of which must be a standard residue. The `alignment_L` attribute of the instantiated object will be a list of pairs each pair being two residues in the alignment.
6. Use the `moveToOverlap` method in the `Overlapper` class to bring each of these protein structures into a structural alignment with 2HYY. The alpha carbon atom pairs that determine the overlap should be taken from the list of pairs generated in the previous step.
7. Ask the user for permission to proceed and when the user responds with the Enter key, use the `undoMove` method to restore the four structures to their original positions.
8. Then repeat Step 6 with the alpha carbon pairs taken from residues in the binding site. To get the alpha carbon pairs that determine the overlap, do the following:
    a. Generate a set of residues that are close to the ligand in the target protein.
    b. Generate a set of residues that are close to the ligand in the moving protein.
    c. Filter the alignment pairs created in Step 5 and keep only those pairs that obey the following two constraints: The first entry of the pair must be in the set generated from Step (a) and the second entry of the pair must be in the set generated from Step (b).

Definition of "close to the ligand": We will say that a residue is close to the ligand if any one of its atoms is within 4 Angstroms of some atom in the ligand. That is, the shortest inter atomic distance is less than or equal to 4. Your script should use a function that returns a list of standard residues that are close to a specified ligand. This can be easily and quickly done in O(*n*) time if you use the `Shell` class.

Your final result should be the same as that seen in the previous script for Exercise 2(a) with the addition of protein 1OPJ (as in the previous Figure).

The next two exercises should give you some experience in Graphical User Interface (GUI) development. For these exercises, you are to imagine that you have a job with a fictitious company that we will call TKI Drugs Inc. To secure research funding, your boss is going to give a talk on c-Abl binding sites and she wants to have a Chimera application that can quickly show the overlap of any **two** of the binding sites that we have encountered in Exercise 2(b).

(c) Start by reading pertinent sections in Appendix A of the text: Pages 297-304, Section A.4.3.3, Section A.4.3.8, Section A.4.4 and Section A.5. This should give you enough information to generate a dialog window that has the following appearance:



Protein P: [✓ 2HYY] 3CS9 2GQG 1FPU 1OPJ

Protein Q: 2HYY [✓ 3CS9] 2GQG 1FPU 1OPJ

Ligands: STI NIL

Note that the two columns of radio buttons are in separate parent frames and so the two groups act independently. For example, a radio button activation in the "red column" will affect the currently selected button in that same column but it has no effect on the radio button in the "blue column". To keep our requirements simple, this exercise will not involve any structure overlap (that is the goal of the next exercise). The script should define a class with header:

```
class TKinhibitors(ModelessDialog):
```

and the `fillInUI` method should generate the widgets shown in the previous figure. For now, the callback function can be a simple return statement (or a print statement for debugging purposes).

Before the dialog is instantiated in the mainline, do the following:

Fetch the five proteins in the red column, delete everything but chain A and color both ribbons and atoms "hot pink". The ligands should have a "ball and stick" representation (all other atoms are in "stick" representation). Repeat for the blue column but this time use the colour "cornflower blue".

Note that `runCommand` can be used to colour the proteins and change the representation of atoms. You can hide a protein using `~display` and bring it back with `display`. Needed information can be found on the Chimera website (download the UCSF Chimera Quick Reference Guide pdf). You should also learn about atom specification. For example, if `p` contains the protein object, then you can colour the ribbons cornflower blue by using:

```
runCommand("color cornflower blue,r #" + str(p.id))
```

In this example, `p.id` is used to get the model number of the protein. The `str` function will convert this integer to a character string that immediately follows the # sign.

Callback functions for the radio buttons:
When the dialog is instantiated, start with the radio buttons activated as shown in the previous figure (the red 2HYY and the blue 3CS9). Your script should set up the display so that it is consistent with that selection. That is, you should see the red 2HYY and the blue 3CS9 proteins and no other proteins. When a different radio button is activated, the display should change to be consistent with the radio button selection. Your callback function for the radio buttons should make this happen. Note that the proteins may be separate or colliding in the display depending on the positions defined by the PDB files.

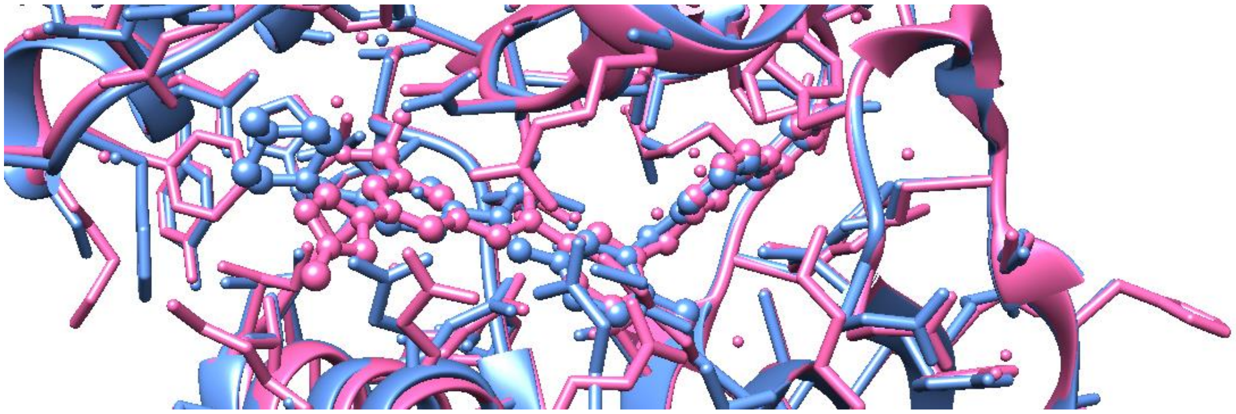
As shown in the previous figure, there are text boxes in the frame at the bottom of the dialog window. The text in these boxes should give the name of the ligand for the selected protein. Your script in the radio button callback function should facilitate this as well.

(d) In this exercise the call back function should show the proteins as done in the previous exercise, but this time with an overlap of the binding site alpha carbon atoms (use a modified version of the code developed for exercise 2(b). The overlap is to be done so as to avoid any repositioning of both molecules. To explain further: Suppose the display is in its initial state with the red 2HYY and the blue 3CS9 in the display. If the user does a click on the blue 1OPJ, the following events should ensue:

- The current blue protein 3CS9 is "undisplayed" using `~display` in a `runCommand`.
- The blue 1OPJ undergoes an `undoMove` operation in preparation for the overlap that is to be done next.

- The blue 1OPJ protein is overlapped with the current red protein (in this case 2HYY). The overlap is done with respect to the binding site alpha carbon atoms, as in Exercise 2(b). Note that 2HYY should not move. Consequently, if the two proteins have been zoomed or rotated in the display to inspect some particular aspect of the binding site, then this positioning in the display will be maintained.
- The blue 1OPJ can then be displayed to show the overlap.

Naturally, if the user does a click in the red column, then a red protein is moved and the blue protein stays where it is. If you want these steps to be done very rapidly, then it would be wise to do as much computation as possible before the dialog is instantiated. For example: set up `Overlapper` objects for each of the ten proteins, get lists of pairs of alpha carbons for all possible overlaps, etc. The next figure shows a close up view of the overlapped binding sites for red 3CS9 and blue 2HYY. Note the overlapping of the red NIL ligand and the blue STI ligand.



Marks:

Exercise 1:	[16] (10 for the calculation + 6 for the plot)
Exercise 2a:	[16]
Exercise 2b:	[12]
Exercise 2c:	[20]
Exercise 2d:	[16]
Total:	**80**

END OF ASSIGNMENT 4

Note: Since everybody selected a project from the "Project Requirements" document circulated earlier, it is not required that you generate a proposal for your project as part of this assignment (as originally intended). However, you should make plans for the project and contact me if there are any design or implementation issues.