Methods in Biophysical Chemistry – CH 8613 01
Programming Assignment

Due Monday, November 28

Introduction and Goals

It is virtually impossible to be a successful scientist today without some understanding of computer programming. Modern instrumentation relies on computation to process and store large datasets, and frequently it is necessary to use programming as a means of examining the data in novel ways. This is especially true in the field of protein science because of how protein coordinates are stored. Complex molecular structures require proportionately complex methods for data analysis, and even those who aren't involved in molecular simulation must have some practical understanding of how to manipulate structural data. In this assignment, you will be asked to solve a simple problem in protein structure analysis by writing a computer program in any language you choose.

If you already know a programming language, you should strongly consider using that language to complete the assignment. Your instructor can assist you in any number of programming languages, including C, C++, FORTRAN, Python, and Perl. If you don't already know a programming language, this assignment will force you to learn one. It does not matter which programming language you use, but Python is strongly recommended if you have not programmed before. It is straightforward and easy to use while being very powerful. It is also popular and looks good on your CV.

Another goal of this assignment is to force you to understand how structural data is stored in the PDB, as well as how to access that data. PDB files contain all the information needed to reconstruct the structure of a protein or nucleic acid: there are atom names, residue names, residue numbers, and the X, Y, Z positions for each atom. With the exception of some "header" information, PDB files are simply a list of this information, which each line corresponding to a separate atom. Your program will have to make sense of this file and perform some useful computation on the coordinates.

Although not explicitly required, a final goal of this assignment is familiarize you with the UNIX operating system environment. The Python interpreter is available as a download for PCs, and all Macs already have Python installed in their UNIX back-end, but as a student of this class, you will be given access to my Linux server so you can write your program there if desired. If you would like an account, please email me and I will give you the details in class. You will not be required to use my Linux server, but I guarantee that knowing UNIX/Linux will benefit you in the future. Employers like to see UNIX experience on CVs, but it also frequently comes up in scientific environments. A good UNIX tutorial can be found at http://www.ee.surrey.ac.uk/Teaching/Unix/.

There is a lot to learn from this assignment, and very little of this material will be taught in class. You will be expected to take several weeks and *teach yourself* the skills needed to succeed. The does not mean you cannot ask for help, but it does mean you will need to pace yourself in

your learning. Therefore, **if you do not have programming background or experience with UNIX, I recommend you start this assignment *right now***. Even though you have more several months to complete this assignment, that time will pass quickly. Your instructor is happy to offer help and advice, but waiting until Thanksgiving will probably be too late to learn everything needed for this assignment.

The Project

The project is (conceptually) simple: You are to write a program that computes a contact plot for a given PDB file. These plots are discussed in section 1.5 of your textbook (van Holde). The protein you will use is Staphylococcal nuclease (Snase), and you will examine two PDB files for this protein: PDB ID 2SNS and 1SNC.

The input of your program will be the PDB file itself. At the command line, you will specify the name of the PDB file that you wish to run. For example, if my program is called `contact.py` and I wanted to run my program on the `1SNC.pdb` file, I would run:

```
% python contact.py 1SNC.pdb
```

Your program must take as input the text file that you download from the PDB (http://rcsb.org/). There is a lot of extra stuff in this file; your program must filter it out. Submissions that require modified or "pre-processed" inputs will not receive full credit.

The output of your program will be a series of coordinates, x and y, stored in a file called "`contact.txt`" In this output, x and y correspond to residue numbers. If x and y appear in the list of output, it means that the C_α atoms are less than 6.0 Å apart. Consider the following example output:

```
1     1
1     2
1     3
2     1
2     2
3     1
3     3
```

The interpretation of this output is that residue 1 C_α is within 6.0 Å from residue 1 C_α (this is always the case, because a residue is always close to itself). Additionally, residue 1 is close to residue 2 and 3. Note that residue 2 is *not* close to residue 3. There is no "`2  3`" nor "`3  2`" pair. Thus, what's left out of your output is just as important as what is included.

The astute reader will note that these coordinates can be used to create a 2D graph. As long as the points are not connected by lines, and as long as the symbols are big enough, you can plot the results of `output.txt` in a software package like Excel, and the result will be a true contact plot. To submit your assignment, you will include both your code and a printout of your contact plots (clearly and neatly labeled) for 2SNS and 1SNC.

Students wishing to submit their plots in Grace AGR format will receive an additional 10% bonus. This accomplishment demonstrates mastery of UNIX as well as programming. To run Grace at the UNIX prompt, type `xmgrace`.

As you think about your code, you will need to consider a structured approach to solving the problem. The following points may be helpful:

1. An approach to solving the problem would be to scan through the PDB file and collect the coordinates of the necessary C_α atoms as you go along. C_α atoms are named "CA" in the PDB file itself; since you are only interested in C_α-C_α distances, you can ignore all other atoms.

2. If you have two atoms, each with x, y, and z coordinates in space, then the distance between those atoms is given by:
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

3. You will need to keep some memory of which residues have which coordinates. In many programming languages, this is done by creating an array. Creating an array of x, y, and z coordinates, where the index is the residue number.

4. Once you have an array of coordinates, you can create a double loop that compares distances for every combination of residues *i* and *j*. Then, print out those combinations where the distance between *i* and *j* is less than 6 Å.

5. PDB files have a standard text format, the definition for which can be found at http://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html. You are most interested in the ATOM records, and you can safely ignore the other lines for this project.

Programming Language

You are welcome to use **any programming language you like.** As mentioned above, however, I would recommend Python if you haven't programmed before. The following websites offer some resources for learning this extremely powerful language.

- *How to Think Like a Computer Scientist* is a free, on-line book that covers the basics of Python. It covers Python from a programming point of view: rather than starting with syntax, it starts with the "big picture" and shows you how Python applies. It's available at http://openbookproject.net/thinkcs/python/english3e/.
- A very good Python tutorial can be found at https://docs.python.org/3/tutorial/.
- *Learning Python* by Mark Lutz is an extremely good book on the subject. This is the book that I started with (back in 2001), although several editions have been published since then. It's not free, but it's a worthwhile investment.
- The Python module index (https://docs.python.org/3/py-modindex.html) contains a listing of all the modules (libraries) available in the default distribution, including the math library. It's more for advanced users, but it is extremely useful.

What to Submit

When you have finished your program, submit the following, either digitally or in printed form, by the beginning of class on the due date given above:

1. A complete copy of your code (sent digitally).

2. Your contact plots for 2SNS and 1SNC (printed or digital copies are fine). I don't need the Excel files. Make sure your name is on your submission and that you've labeled each plot clearly. An example of how your plot should look is on the next page.

3. Your agr data files (if you used Grace to create your graph). This should be sent digitally.

Requirements

Below is listed a summary of the key requirements for this assignment:

1. Your program must take the PDB file as input. It can either prompt the user for the PDB file, or it can take it as a command line argument (demonstrated above).

2. Your program must write the output to a separate file called "`contact.txt`" and not to the screen. The requirement is to demonstrate that you have mastered file I/O.

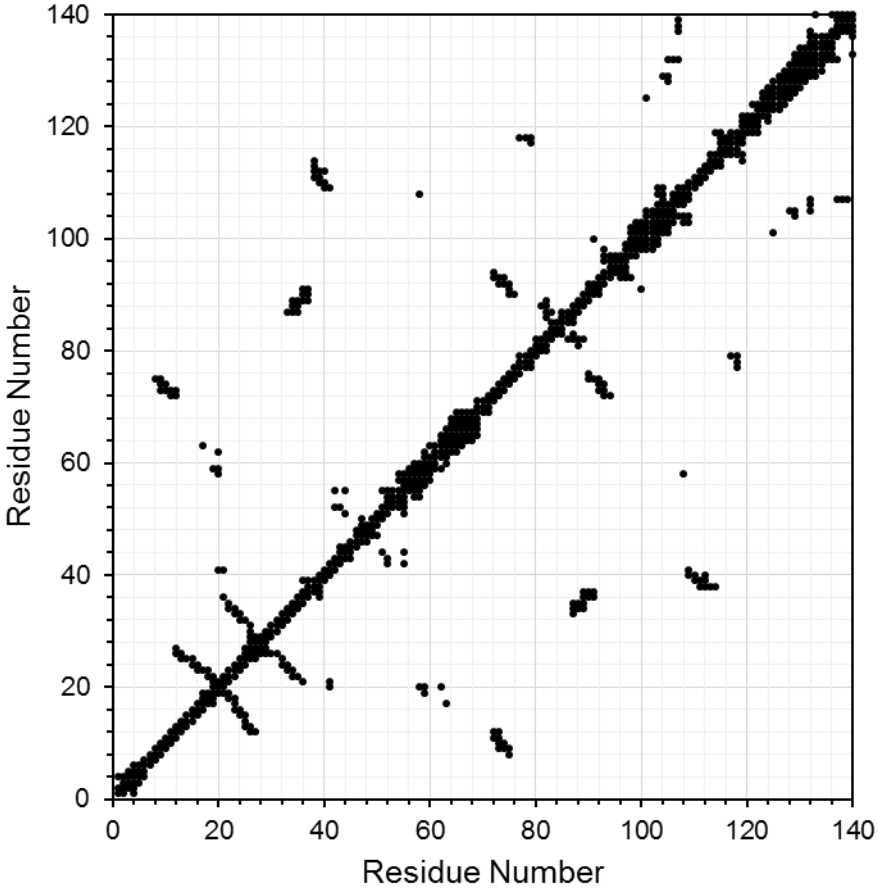
3. Your program must calculate the distances between all CA atoms within the protein; these distances must be calculated from the atoms within the PDB file and not from a lookup table or any other source.

4. You can assume that the PDB file contains no missing atoms, but you cannot assume that the PDB file starts at residue number 1. If you inspect the PDB file for 1SNC, you'll notice that the first residue is number 7. Your code must account for this numbering offset automatically.

5. Your code cannot use any third-party protein/PDB processing libraries. General purpose libraries (such as Python's built-in modules) are fine.

Example Contact Plot

The following is an example of what your output should look like for the PDB file 2SNS. This graph was created in Excel.



Category	Evaluation Criteria	Max. Score	Student Score
Code	**Input: PDB Parsing** • Program can read a PDB file • Unnecessary lines are ignored (REMARK, HETATM, etc.) • Program prompts for name or takes command line input	5	
	Atom Selection • Program properly selects atoms for distance calculation • Side chain atoms are ignored	10	
	Calculation • Formula for distance is correct • Output is x, y pairs of residue i, residue j • Output is written to a separate file (not stdout)	10	
	Style • Comments denote flow of execution • Good use of indentation, spacing, etc. • Functions are used to simplify calculation	3	
Plots	**Plot Correctness** • Both 2SNS and 1SNC match solution	15	
	Plot Format • Axes are labeled properly, etc. • Range of axes from 0-150 (not 200, 300, etc.)	7	
Grace Plots (Extra Credit)	**Plot #1: 2SNS** • Plot can be opened in Grace	3	
	Plot #2: 1SNC • Plot can be opened in Grace	3	
Total Score		**50**	