MIC653: Advanced Molecular Genetics Bioinformatics & Computational Genomics

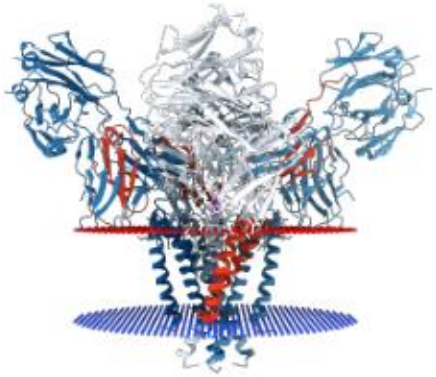
Projects: Computation to Solve Problems

How big is a protein?

(use ViroBIKE)

Rationale

You've probably seen pretty pictures of proteins. For example, the figure to the right shows the structure of the potassium-selective channel from the bacterium *Streptomyces lividens*. Most of the time these pictures give the protein in austere isolation. This one provides something to compare the protein against – the two sides of the cell membrane in which the protein sits (the red and blue circles). In general, it's difficult to get a sense of size of proteins shown in such structures.



Potassium channel from structure 1K4C.

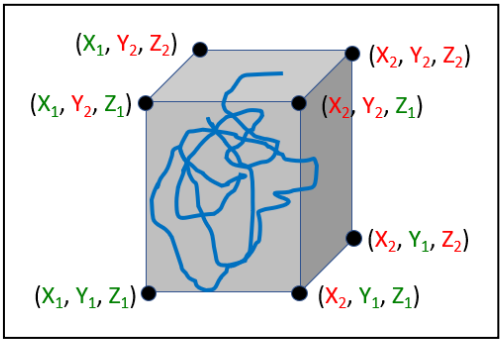
But these pictures are based derived from files containing atomic coordinates, in Angstroms. It should be straightforward to look at such a file and figure out the size of a protein.

Of course it isn't so simple. You can inspect the file for the potassium channel within ViroBIKE by mousing over the black **File** button and clicking **Shared files**. Then find the file called "k-channel.pdb" and click its name. You are now looking at a pdb file (pdb=protein data base), the most common format for protein structures. Scroll down to the section that has atomic coordinates (marked by ATOM as the first word in the line). You'll find that each atom is described by its atomic name (e.g. O for oxygen or CA for alpha carbon), by the amino acid in which it lives (e.g. GLY), by the protein subunit (e.g. A), and most importantly by three numbers – the x, y, and z coordinates. To figure out the size of the protein, it is these coordinates you want.

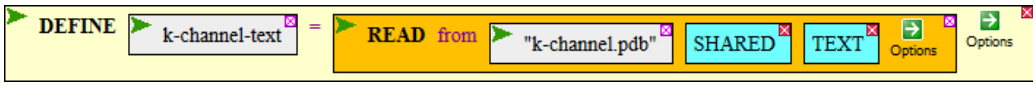
Your goal is to find the coordinates of a box that contains the protein. Then it should be a simple matter to calculate the volume of the box

A. Search a Genbank file containing the annotation of *E. coli* for a specific text pattern

From the diagram to the right, you can see that you need only six coordinates: the minimum and maximum values for the x, y, and z coordinates. From these, you can calculate the sides of the box and from that the volume. You could scan the PDB file for these values, but that would be both demeaning and probably inaccurate. This is what computers are for.



1. Read in the PDB file and save the text as a variable, something like this:



You can get the **DEFINE** function from either the DEFINITION menu or from the ALL menu. You can get the **READ** function from the INPUT-OUTPUT menu. Select the SHARED option (because the file is in the shared directory available to everyone) and the TEXT option (because the file is in pure text format). Notice that executing this function gives you a list of strings. The coordinates you want are embedded within the strings that start with "`ATOM`".

2. Extract the x, y, and z coordinates using **MATCHES-OF-PATTERN**, available from the STRINGS-SEQUENCES menu, SEARCH/COMPARE submenu. The target will be the variable you just defined containing the text of the PDB file. The pattern is more challenging. It should be in double quotes. It should begin with ATOM, followed by any number of characters until you reach the three coordinates, which can be recognized because they have three digits to the right of a decimal point. There might be one, two, or three digits to the left of the decimal point, and the number might be preceded by a negative sign. Each of the three numbers should be captured. Define the output of **MATCHES-OF-PATTERN** in another variable, something like this:

After you fill in the *pattern* and *target* holes, execute the function.

3. You should get a long list of x, y, z coordinates, something like this: `(("124.067" "122.740" "-21.723") ("124.721" "121.508" "-21.203")…)`
Two problems here. First, you want to get the minimum and maximum of each of the three coordinates, but to do this, you need to collect all the x coordinates in one place and the same with the y and z coordinates. The second problem is that **MATCHES-OF-PATTERN** gave you strings not numbers. You'll need to convert the strings to numbers before you look for the minimum and maximum values.

4. To solve the first problem, transpose the list. If you consider the list as a table

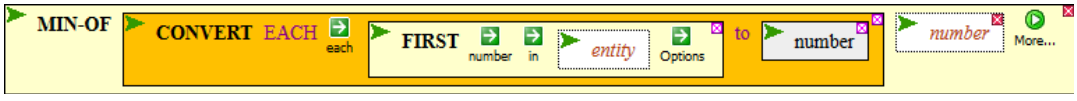
x_1 y_1 z_1
x_2 y_2 z_2
x_3 y_3 z_3
. . .

then transposing this table makes the rows the columns and vice versa:

x_1 x_2 x_3 . . .
y_1 y_2 y_3 . . .
z_1 z_2 z_3 . . .

Do this using the **TRANSPOSE-LIST** function (LIST-TABLE menu, LIST-PRODUCTION sub-menu). **DEFINE** a new variable as the output of the **TRANSPOSE-LIST** function.

5. Now you're ready to use the **MIN-OF** and **MAX-OF** functions on each row of the transposed list, something like this:



The **FIRST** function extracts the first row of the transposed list. The **CONVERT** function converts each string in the row into a number (if you choose the EACH option).

The **MIN-OF** function gives you the minimum value of all the numbers. If you do this for the six conditions ({**MIN-OF**, **MAX-OF**} x {**FIRST**, **SECOND**, **THIRD**} you'll get the six coordinates you want. There are more elegant ways than this, but executing the six functions is probably conceptually the simplest.

B. Calculate the size of the box

From the six coordinates, you should be able to calculate the lengths of the sides of the box, in Angstroms. How does the size compare with the width of a biological membrane?