Introduction to Bioinformatics – AS 250.265
Assignment 3

Pairwise alignment is one of the fundamental topics in bioinformatics. While it is important to understand how to use the necessary tools, it is equally important to understand the concepts upon which those tools are based. In this assignment, we will investigate the basics of how these matrices are formed.

Included in this assignment are two Python libraries for you to use. They are `matrix.py`, a library of simple matrix-manipulation functions, and `needleman.py`, a file containing substitution matrices and a simple implementation of the Needleman and Wunsch algorithm. For this assignment, is intended that you download both of these files to the same directory and import them into the programs you write (also in the same directory). It is important that you study this code carefully, as you will need to piece it together to perform parts of your assignment.

To illustrate the use of `matrix.py` and `needleman.py`, two additional files have been provided for you. They are `algebra.py` and `align.py`. One of the first things you should do in this assignment is download all four files to your homework three directory and run `algebra.py` and `align.py`. This will not only give you an idea of what these libraries can do, but it will also confirm that you have everything set up correctly.

As you work with this assignment, you will be creating text output, some of which you will be asked to upload to the assignment website. The following useful UNIX trick will make your programming much easier: By appending a greater than sign to the end of your command, you will be able to "redirect" the output of your Python programs to a text file. For example, the following command will dump all the output from the python program `test.py` into the file `test.out`. If necessary, `test.out` will be created for you in the current directory.

```
python test.py > test.out
```

You will not be able to observe the output from `test.py` until you open up `test.out` in xemacs (or use the `less` command), but you will have a hard copy of your output that you can save, edit, or upload to the website.

This assignment is due February 23 at the start of class. Programming portions of the assignment should be aptly named (e.g. `q01a.py`) and submitted through the course web server. The written portions of the assignment may be submitted by hand or uploaded as a Word document.

Question 1: Substitution Matrices (45 points)

The following questions ask for you to manipulate the PAM matrices (stored in `needleman.py`) with the functions that are stored in `matrix.py`. Read the directions carefully—some questions require you to submit both your code and your output.

a. [programming] Using the functions provided for you, write a program that prints the PAM1 mutation matrix, multiplied by 10,000. Each column of the matrix should be four characters wide and there should be no decimal places. Your program output should be correspond to the table in your book on page 55. Submit both the program (q1a.py) and the output (q1a.out) of your program. Do not submit the `matrix.py` or `needleman.py` libraries. (5 points)

b. [written] Using the rule of thumb discussed in class, what is the approximate percentage sequence identity specified by the PAM25 substitution matrix? What is the actual percentage sequence identity specified by this matrix? (Hint: estimate the answer from page 62 of your book.) How do these two values compare? (5 points)

c. [programming] Write a program to compute the PAM25 scoring matrix. Your program should display both the mutation probability matrix as well as the log-odds scoring matrix. Display the matrices in a reasonable format as far as total column widths and decimal places. (20 points)

d. [programming] In the `needleman.py` file, you will notice an empty matrix called `mtx_mymtx`. Using any resources available to you (for example, the chart on page 45 of your book, any biochemistry textbook, etc.), use this template to construct your own similarity matrix. In your matrix, use only whole numbers from negative three to three, where negative three is the least similar and three is the most similar. As long as you have put some thought into your comparisons, you will be given full points—there is no absolutely correct answer here. Submit your entire `needleman.py` file (renamed for this question, e.g. `q01d.py`), with the modified `mtx_mtmtx`, as your solution to this problem. (15 points)

Bonus: The scoring matrix from the class that produces the highest sum of aligned residues plus aligned sequence identity when aligning apolipoprotein D and RPB4 will receive an extra 10 bonus points on this assignment. The second-best matrix will receive 5 bonus points. The file `align.py` has been provided to give you an example of how to perform alignments with the `needleman.py` module, and it includes the protein sequences for both apolipoprotein D and RBP4.

As an example, if you have an aligned sequence identity of 100%, but only align 20 residues, your sum would be 120. This would be worse than another solution that aligned 130 residues and had an aligned sequence identity of 60 percent, for a sum of 180.

Question 2: Sequence Alignments (30 points)

a. [written] Below is a sequence alignment. Calculate the score for this alignment using the PAM250 log-odds matrix (on page 58 of your book). Assume a gap opening penalty of -10 and a gap extension penalty of -2. What is the aligned sequence identity for this alignment? What is the aligned sequence similarity? (10 points)

```
--MQEKV-QTSSVFY---LLLFGAEDDEYF
  ||. | |||.      |||  |..|.
AVMQDVVHQTSTLSSPAGLLL--ADEDD--
```

b. [written] Use the Needleman-Wunsch algorithm by hand to determine the optimal alignment between the following sequences. (15 points)

sequence 1: GGTYHERS
sequence 2: GTYERSE

Use an identity matrix to score residues, and do not score gaps. You should submit a table that displays the scoring matrix as well as the alignment your function produces. The table may be in Microsoft Excel format if this is convenient. On your table, highlight the cells that you used to form your alignment (for an example, see the lecture four slides given on the course website).

c. [written] On the Biology Workbench site, you will notice that one of the protein tools, RANDSEQ, will allow you to uniformly randomize a protein sequence. Perform 10 global alignments, between RBP4 and randomized apolipoprotein D using the PAM250 matrix and the default gap parameters. Calculate the mean and standard deviation of these random alignment scores, and compute a Z value for the alignment between RBP4 and apolipoprotein D. The standard deviation is defined by:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)}$$

where the set has N elements, $\{x_1, x_2, \ldots, x_N\}$, whose mean is μ. Submit your 10 alignment scores, the mean, and the standard deviation along with your Z value. Briefly explain why the Z-value is not as informative as we would like. (5 points)

Question 3: Sequence Degeneracy (25 points)

In this question, you will use your computer skills to solve a practical problem as a warm-up for our discussion of local alignment tools such as BLAST. When performing a database search, it is often the case that you would like to search for nucleotide sequences that could code for a given protein sequence. One naïve way of searching the database would be to attempt to "reverse translate" the protein sequence into all the possible DNA sequences before performing the comparison. Then, once the possible sequences are obtained, one could search for the appropriate nucleotide sequences in the database.

In practice, this is impossible to do, because the number of possible DNA sequences encoding for one protein sequence is huge.

a. [programming] Write a program that will compute the number of RNA sequences that can code for the human histatin 3 protein. Do not actually compute the RNA sequences—just calculate the number of them. To do this, you will need to obtain the protein sequence for histatin 3 first, then you will have to implement a program that calculates the number of RNA sequences. Think about how you might do this given the types of problems we have had before. In planning your solution ask yourself the following questions: Why are there multiple RNA sequences for one protein sequence? What type of programming construct could you use to capture the degeneracy between an amino acid and the number of different ways to translate that amino acid? (20 points)

b. [written] Assuming that each of the RNA sequences above could be compared with an entire nucleotide database in one second, how long (hours, days, months, years) would it take to compare all of the possible RNA sequences for histatin 3 to the entire database? (5 points)