CSC 314, Exam II Review

Exam II Outline

1. Bioinformatics Databases and Tools
    a. Searching Entrez
    b. GenBank
    c. UCSC Genome Browser and UCSC Table Browser

2. Python
    a. Writing functions
    b. Reading in files

3. Biopython
    a. DNA/RNA sequences using Bio.Seq objects
        i. Finding complements
        ii. Transcribing sequences
        iii. Translating sequences
        iv. Using standard string methods and actions
    b. Reading and parsing sequences in
        i. GenBank Flatfile format
        ii. FASTA format

4. Sequence alignments
    a. Finding the alignment score using BLOSUM matrices and gap penalties
    b. Finding the optimal *global* alignment using the Needleman-Wunsch dyamic programming method
    c. Finding the optimal *local* alignment using the Smith-Watterman dynamic programming method

5. Using BLAST to identity similar sequences and conserved protein domains

Exam II Practice Questions

Note: This review is not comprehensive, but contains several practice problems to help prepare for Exam II. In addition to these practice problems, you should make sure to understand all labs assigned since the first exam.

1. **Sequence Database questions**

   Look at the GenBank entry with accession number NM_001185098 to answer the questions below:

   a. What is the length of this sequence?
   b. When was the sequence last modified?
   c. How many exons does this gene have?
   d. What position marks the beginning and end of the poly-adenylation signal sequence, and what is this sequence (biological note: this sequence is recognized in the process of poly-adenylation, which is the addition of a poly-A tail to the end of a mRNA.
   e. What are the first nine nucleotides in the coding sequence (CDS), and the corresponding amino acids (you can give the 1-letter amino acid codes)?

2. UCSC Genome Browser

   a. Use the UCSC Genome Brower to display only the NCBI Genes track (under Genes and Gene Predictions) and Spike Mutations (under Variability and Repeats) for SARS-Cov2, using Pack View. What is the first mutation listed that is in red (red means bad), and why is this bad. Note that Regeneron can be used to treat COVID-19. You can view the SARS-Cov2 genome map here: https://genome.ucsc.edu/cgi-bin/hgTracks?db=wuhCor1

   b. Use the UCSC Table Browser to display the NCBI gene names and protein products of SARS-Cov2. How many genes are present?

3. **Sequence Alignments**

   For (a) and (b), use a linear gap penalty of 4 points, a match score of +5 points, and a mismatch score of -1 point.

   a. Find the optimal global alignment and optimal global alignment score for the words *handy* and *say*. You must show your dynamic programming matrix to receive credit.

b. Find the optimal local alignment and optimal local alignment score between the words *stars* and *that*. You must show your dynamic programming matrix to receive credit.

c. Using the BLOSUM-62 matrix, a gap opening penalty of 5, and a gap extension penalty of 1, find the score of the *semiglobal* alignment given below (Recall that semiglobal alignments do not penalize gaps at the beginning or end of the alignment).

```
F R I D A - - Y
- - P - A R T Y
```

4. ***BLAST***. The protein sequence for the C. elegans (worm) gene F55F8.2 can be found at accession NP_491652. BLAST this protein to find similar proteins in humans (use the default databse, but limit your analysis to *Homo sapiens*). Note: the protein DDX24 is a known human ortholog of this worm gene, based on the fact that there is a high similarity.

   a. What is the % identity and % similarity of the alignment between F55F8.2 and DDX24 (this is labeled “DDX24 protein [Homo sapiens]”)

   b. What is the first pfam domain present in this protein? Based on these domains, what do you infer about the function of F55F8.2? Do you think it may be involved in *translation*? Why or why not?

5. ***Coding questions*** **(also see *biopython_practice.ipynb* in the code folder)**

   a. The file ‘contacts.txt’ contains a tab-separated list of names and e-mail addresses for several individuals.

      i. Write Python code that reads in ‘contacts.txt’ and creates a dictionary for looking up a person’s email address by their last name.

      ii. Modify (i) to create a function that does this and returns the dictionary

      iii. Prompt the user to enter a last name, and then output the e-mail address, or “e-mail not found”.