CSE 527, Fall 2017
Problem Set #4

(Due Dec 11th 11:59pm)

---

1. **[40 points] Hidden Markov Model (HMM)**

   Write a program that implements a 2-state HMM for detecting G+C rich regions in the Dictyoglomus thermophilum H-6-12 sequence . Conceptually, state 1 will correspond to the more frequent 'A+T rich' state, whereas state 2 will correspond to the less frequent G+C-rich state. Specifically: The starting parameter values (taken from Klein et al (2002), PNAS 99: 7542-47) should be as follows:

   Transition probabilities a_{ij} are $a_{11} = .999, a_{12} = .001, a_{21} = .01, a_{22} = .99$.

   Initiation probabilities for each state (i.e. the transition probabilities from the 'begin' state into state 1 or 2) should be .996 for state 1, and .004 for state 2; these should be held fixed throughout the Viterbi training

   Emission probabilities (which should also be held fixed) are $e_A = e_T = .291, e_G = e_C = .209$ for state 1; $e_A = e_T = .169, e_G = e_C = .331$ for state 2.

   Use Viterbi training as described in class to find improved parameter estimates for the transition probabilities, holding the emission and initiation probabilities fixed at the above values. Run the training for 10 iterations, where for each iteration you:

   - Use dynamic programming to find the highest probability underlying state sequence.
   - Using this state sequence, compute
     - The number of states of each of the two types (1 and 2), and the number of segments of each type (where a segment consists of a contiguous series of states of the same type, that is preceded and followed by states of the opposite type or the beginning or end of the sequence).
     - New transition probabilities to be used in the next iteration.

   Your output should provide:

   - the name and first line of the .fna file
   - the information described above (in 2. – i.e. numbers of states and segments, and new probability values), for each of the 10 iterations. Give probabilities to 4 decimal places only.
   - the list of G+C-rich segments (corresponding to the segments having state 2 as the underlying state) after the final (10th) round of Viterbi training, sorted by genomic position.
   - your description of the first 5 of the segments found above, as found by looking up the Genbank annotations.

   Sequence: `ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old_refseq/Bacteria/Dictyoglomus_thermophilum_H_6_12_uid59439/NC_011297.fna`

   Genbank file: `ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old_refseq/Bacteria/Dictyoglomus_thermophilum_H_6_12_uid59439/NC_011297.gbk`

   (Acknowledgement: This problem was generated by Benjamin Vernot for Prof. Phil Green's GS540.)

2. **[60 points] Dynamic Programming**

Implement the Needleman-Wunsch global sequence alignment algorithm described in lecture. For programming language, you may use any of C, C++, C#, Java, Perl, Python, Ruby, R; ask if you prefer something else.

Use the simple linear gap cost of -4 (an arbitrary value, not a recommended default) together with the BLOSUM62 scoring matrix from the online supplement to:
* SR Eddy, "Where did the BLOSUM62 alignment score matrix come from?" Nat. Biotech., 22, #8 (2004) 1035-6.

[URL] `http://www.ncbi.nlm.nih.gov/pubmed/15286655?dopt=Abstract`
[Offcampus] `http://offcampus.lib.washington.edu/login?url=http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=pubmed&dopt=Abstract&list_uids=15286655`

You may ignore the B, Z, X and * rows/columns; they are various wild-cards. Also ignore any sequence characters other than the codes for the 20 standard amino acids. Besides the lecture notes 17-18, the following paper is highly recommended.
* SR Eddy, "What is dynamic programming?" Nat. Biotech., 22, #7 (2004) 909-10.

[URL] `http://www.ncbi.nlm.nih.gov/pubmed/15229554?dopt=Abstract`
[Offcampus] `http://offcampus.lib.washington.edu/login?url=http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=pubmed&dopt=Abstract&list_uids=15229554`

Please do not find code on the web to do this. Please avoid proprietary or nonstandard frameworks, libraries or packages. Libraries like BioPerl are not needed, but allowed. Obviously, avoid their alignment code, but if you want to use them, e.g. to download the relevant sequence files, that's fine.

(a) **[15 points]** Implement the part of computing the values F_{ij} of optimal alignments of between all substrings $X_1, ..., X_i$ and $Y_1, ..., Y_j$, i.e., filling in the "F-matrix". Provide your (commented and reproducible) source code.

(b) **[15 points]** Implement the "trace-back" procedure to display (one of) the optimal alignment(s). Provide your source code.

(c) **[5 points]** Run your algorithm on the pair of strings "deadly" and "ddgearlyk", print the full alignment matrix, optimal score, the alignment produced by the trace-back, and p-value resulting from permuting "ddgearlyk". (Don't print the matrix for the larger examples below.)

(d) **[10 points]** Using your implementation, align and score protein sequence 1 below against each of the other six.

	Species	Protein	Name	Accession
1	Homo sapiens (Human)	Hemoglobin subunit beta	HBB_HUMAN	P68871
2	Pan troglodytes (Chimpanzee)	Hemoglobin subunit beta	HBB_PANTR	P68873
3	Mus musculus (Mouse)	Hemoglobin subunit beta	HBB1_MOUSE	P02088
4	Gallus gallus (Chicken)	Hemoglobin subunit beta	HBB_CHICK	P02112
5	Fugu rubripes (Japanese pufferfish)	Hemoglobin beta subunit	Q802A3_FUGRU	Q802A3
6	Vigna unguiculata (Cowpea)	Leghaemoglobin	Q540F0_VIGUN	Q540F0
7	Homo sapiens (Human)	Insulin-like 3 precursor	INSL3_HUMAN	P51460

You can download the amino acid sequence for each from SwissProt (`http://www.expasy.org/`). Select "UniProtKB" in the search pulldown and paste in the accession number. Select the accession number under Entry. Look for the "FASTA" link under "Sequences" near the middle of the page. But do stop to browse the primary page a bit before you click through. What does this protein do? Do you expect it to be similar to HBB_HUMAN? Note: you should put answers to those questions into your writeup.

(e) **[5 points]** Calculate and report p-values for the significance of each of these alignments using the permutation method sketched in lecture. Do at least 1000 random permutations for each, preferably 10,000 or more.

(f) **[10 points]** Modify your program to implement Smith-Waterman algorithm as described in the lecture. You can find an additional reference here: `http://goo.gl/K6zukh`. Run it on the following pair of amino acid sequences: "SRGMIEVGNQWT" and "RGMVVGRW". Use BLOSUM62 matrix and constant gap penalty of 8. Provide the best alignment and it's score. While you outputting your alignment, think about the fact, that it is a local alignment, not a global one. In practice, one of the strings can be millions of symbols in length, so be careful with your output.