CS5566 Fall 2016 Assignment II. 100 pts. NAME(S):

OBJECTIVES: k-mer and motif analysis; Learning to program in a scripting language
RESOURCES: Textbook, Class website, Internet

Electronic submission on Blackboard is due by 11 pm on Friday, Oct 12th, 2016. Submissions received after the deadline will be graded only for effort and receive a maximum of only 70% of the grade (Refer to class syllabus for detailed grading policy). If you are submitting this jointly as a 2-member team, include all names on a single upload and remember that undergrads and grad students cannot partner with each other. Questions marked as G are optional for undergraduate students. All answers should be in your own words with all sources you referred to cited at the appropriate places (Refer to class syllabus for detailed policy on plagiarism). It is highly recommended to use Python for this assignment.

For programming assignments, the submission should include this page as the first page of your assignment, pseudo-code reflecting your understanding of the solution, and examples of actual input and output to validate your implementation. This report should be a standalone summary of your work. A plain text file version of your code should be submitted (include your name as comments at the beginning of the code file), together with test input file(s), and a README file that includes instructions on how to run the program. Combine all files into a single ZIP file before uploading them to Blackboard. State any assumptions you make and show work for maximum credit. In addition to completeness and accuracy, grading will also take into account the modularity and clarity of your implementation. For example, it is necessary to read and write input and output using filenames as arguments rather than hardcoding filenames. It should be possible to run programs from a command prompt or shell by typing "python programName InputFileName OutputFileName." This will help you reuse your code for subsequent assignments. It is necessary to write this from scratch rather than by using pre-existing code. **Make sure your program prints intermediate output to a file so that the underlying steps and reasoning are self-evident.**

1. (50UG/40G) Download the set of partial sequences for any motif from the JASPAR database (http://jaspar.genereg.net/; 'Browse' and click on a sequence logo to obtain a set of sequences for a motif). Write a program that can recover the motif from the sequences using one of the following approaches - brute force, hash function, or CONSENSUS. Discuss to what extent you were able to recover the motif.
2. 50UG/40G) Design and implement a program that computes the similarity between two FASTA format sequences based on their 2-mer composition. Evaluate the accuracy of your program by comparing it with the output of an existing implementation of global pairwise sequence similarity. Compute a correlation coefficient between the scores generated by the two programs.

3G. (20G)

   a. A multiple sequence alignment is necessary to determine the degree of similarity between the sequences. However, one needs to know the degree of similarity to be able to choose the appropriate amino-acid substitution matrix for a multiple-sequence alignment. How can this paradox be resolved?
   b. DNA motifs can sometimes be variable in length. Outline an approach to detect motifs that can vary in length.

Pseudo Code, Snapshots, and Assumptions for Hw #2

Bridget Mohn and Eric Wilson

Problem 1:

Pseudocode:

Figure out the motif sequence length from user input at the start

- For each number of iterations desired
    - Grab two random indexes
    - For each possible kmer of the first random sequence
        - Set first kmer
        - For each possible kmer of the second random sequence
            - Set its first kmer
            - Score the similarity of the two current kmers by checking if each character at the same index in each sequence match
            - If(that score is greater than current seed score)
                - Set the new seed score
                - Set the new two best kmers
    - Find the median string of the two most similar kmers by finding the in common characters and saving those
    - Set the seed string to the median string found
    - For each remaining sequence in the array
        - For each possible 8 mer of the current sequence
            - Score the similarity between it and the median string using same method used above
            - If(that score is greater than the strings current max score)
                - Set the strings current max score to that score
        - Add current strings max score to the current motifs max score
    - If(current motifs max score is greater than the best motifs score)
        - Set the best motif score to the current motifs score
        - Set the new best motif to the one that matches the current best motif score
- Output the motif

Assumptions:

- The user has seen the file with the set of sequences in it so they know the length motif sequence.

Input/Output:

Input	Output
>MA0011 agcaaAACTATTTgat gcgtgTGCTAGTTgtg attgaTCCTAGTTctc tacctTTCTATTCtgt gctttTACTATAAatt ttttcAACCATTTttt tgtgtATCTAATTggc cactgTCCTACACttc gtcttGTCTATGTcta accagCTCTATTGttt ttgttTATTATTTtgt ttgttTATTATTTatt	BEST MOTIF: ttgtttat
>MA0019 GGGTGCAATCGCc AGGTGCAATACCc GAATGCAATCCCc AGGTGCAATAGCc AAATGCAATCGCc ACATGCAATCCCa GGTTGCAATACCt CAGTGCAATACCc CTGTGCAATCCCc GTGTGCAATCCTa GTGTGCAATCCTc CAATGCAATACCc CAATGCAATACCc AGGTGCAATCGTt ACATGCAATCGCc AATTGCAATCCCc CCGTGCAATACCt AGCTGCAATACCc GAGTGCAATCGGt ATGTGCAATCCGc GCGTGCAATCGTg GGATGAAATACCc TAGTGCAATCGTg ACATGAAATACCc GGGTGCAATTATg TTGTGCAACACCc GGCTGCAACCCCc TCATGCAACCCCc AGACGCAATACCc CCATGAAATACCc CGATGAAATCACc TAATGCAATGTGt TAATGCAATGTGt GTATGCAACTACt TGATGTAACACCc ATATGCAGTGGGg GACTGCAACTTCt AGTTGCACTGATc AGATTTCCTGCGt	BEST MOTIF: --atgcaatccc

Observations:

- Recovered motif for some sample input was shifted by one index from original motif. This can be attributed to the first or last column in the original motif having a similar score to the column just outside of the original motif on the opposite side.

Problem 2:

Pseudocode:

- Get input
    - -Expected input is single file with two sequences, one sequence per line
- Create array of two arrays each filled with string representation of all 2mers for respective sequence
- build dictionary with key for each possible combination of 2 amino acid characters
    - populate value for each key with array [0.0, 0.0] (frequency pair for individual 2mer)
- For each array of 2mers
    - for each 2mer
        - increment corresponding value in dictionary for correct index in frequency pair
- sum = 0
- for each k,v pair in dictionary,
    - if v != 0,0
        - normalize frequency pair by dividing value by number of 2mers in 2mer arrays
        - find baby deltas by reassigning v to abs(pair[0] - pair[1])
        - add baby delta to sum
- print score

Assumptions:

- Assumes input file to be a single file with two lines. Each line is a biological sequence.
- Assumes low score indicates high pairwise similarity.

Input/Output:

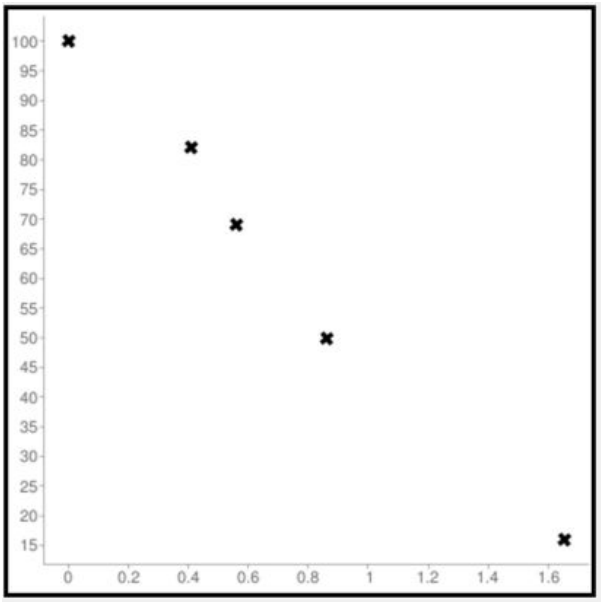
Input	Output	
	HW2-P2	Needle
ATTTGCGCGCATTCGCCGTTCGATCTGCC ATTTGCGCGCATTCGCCGTTCGATCTGCC	Score: 0.000	Similarity: 29/29(100.0%)
ATTTGCGCGCATTCGCCGTTCGATCTGCC ATTTGGCCGTATTCGCCGTTGAATCTGCC	Score: 0.407	Similarity: 24/29 (82.8%)
ATGCGTCATGCATGCATGCACTGTGTGGGT DCACABCDBCABCDBACBDBCABCBDCDAB	Score: 1.655	Similarity: 6/36 (16.7%)
ATTTGCGCGGATTCGCCGTBCGATCTGAA ATTTGGPPGTATTCGCCGTTGAATCTGCC	Score: 0.558	Similarity: 20/29 (69.0%)

ATTTGCGAAATTCAAAACGTBCGATCTGAA ATTTGGPPGTATTCGCCGTAATCTGCC	Score: 0.863	Similarity: 15/30 (50.0%)

Needle output Source: http://www.ebi.ac.uk/Tools/psa/emboss_needle/

Correlation coefficient:

-0.99 (really good!)



Source: alcula.com