Programming project 2

Submission: ipython notebook with your code and result displayed.

Part 1

A permutation p of n symbols is an ordered list of n symbols (no repetitions). We call n the length of the permutation.

In the following, we use the symbols $a, b, c,$ ..... For example, $p = cefdab$ is a permutation of length 6 on the symbols a, b, c, d, e, f.

We denote by $p[i]$ the symbol of p at position i, $i = 1, \cdots, n$, and by $pos_p[x]$ the position of symbol x in p. If $p = cdab$ is a permutation on 4 symbols we have $p[3] = a$ and $pos_p[a] = 3$.

Given two permutations p and q of the same length n we want to compute their distance. Many definitions of distance between two permutations exist in the literature. In this project, we consider the following common definitions: **Hamming distance**, **Kendall's distance, Spearman distance.**

You have to write a Python program that takes two permutations from the standard input (use function input(), example code from example1.ipynb). The input permutations have length less than 26 and are on symbols of the English alphabet. It then:

1. checks that each input permutation is valid, i.e. there are no repeated symbols. If not the program stops.
2. checks that the permutations are on the same set of symbols. If not the program stops.
3. Computes and prints the above 3 distances between the input permutations: Hamming distance, Kendall's distance, Spearman distance.

Below are the definitions of the distances:

Hamming distance: Given two permutations p and q of length n, it counts the number of positions at which p and q have a mismatch.

Ex. $p = cefdab$ and $q = cfebad$

The Hamming distance of p and q is 4 since the two permutations differ in 4 positions; i.e. 2,3,4,5.

What is the maximum value of the hamming distance between any two permutations as a function of n? No written answer is necessary, just think about it.

Kendall's distance: The Kendall's tau distance KD counts the number of inversions occurring between two permutations, i.e. the number of pairs of symbols that appear in opposite order. Formally,

$KD = the\ number\ of\ pairs\ of\ symbols\ (x, y)$such that (($pos_p[x] < pos_p[y]\ and\ pos_q[x] > pos_q[y]$) or ($pos_p[x] > pos_p[y]\ and\ pos_q[x] < pos_q[y]$)).

Ex. $p = cefdab$ and $q = cfebad$

The distance is 4 since there are 4 inversions, precisely (e,f), (a,b), (b,d) (a,d)

What is the maximum value of the Kendall distance between any two permutations as a function of n? No written answer is necessary, just think about it.

Implementation hint: (However, you can use any other way except, of course, functions in statistical packages)

Represent permutation p by a *precedence matrix P*, that is a $n \times n$ matrix defined as follows:

$P[x, y] = 1$ if symbol x precedes y, and $P[x, y] = 0$ otherwise.

For example, the matrix P associated to $p = cefdab$ is

	a	b	c	d	e	f
a	0	1	0	0	0	0
b	0	0	0	0	0	0
c	1	1	0	1	1	1
d	1	1	0	0	0	0
e	1	1	0	1	0	1
f	1	1	0	1	0	0

You can compute the Kendall distance between p and q as follows. Construct the two matrices P and Q as above and …. (you continue).

Spearman distance: The Spearman distance or position-based distance is the sum of the absolute differences of the positions of the symbols, i.e

$$SF = \sum_x | pos_p[x] - pos_q[x] |.$$

Ex. $p = cefdab$ and $q = cfebad$

The symbol *a* is at the same position 5 in both permutations and so is *c*. Thus, they contribute 0 to the above sum. The symbol b is at position 6 in *p* and 4 in *q* thus the absolute value of the difference in position is 2, and so on. In total, the Spearman distance of p and q is 6.

What is the maximum value of the Spearman distance between any two permutations as a function of n? No written answer is necessary, just think about it.

Problem 2.

Given a set of *t* DNA strings of the same length *n*. The frequency of a base at position *j* is the number of times the base appears in position *j* in the set of DNA strings divided by the number t of DNA strings.

Ex.

AAAGGTTTAA

ATTTTCCCCC

GGGGCCAAG

ATTTTCAAAT

ATTAACTCCT

Here t=5 and n=10. The frequency of A at positions 0 to n-1= 9 are 4/5, 1/5, 1/5, 1/5, 1/5, 0, 1/5, 2/5, 3/5,1/5, respectively.

The frequency of T at position 1 is 0, and so on.

Let base A correspond to the integer 0, T to 1, G to 2, and C to 3. The frequency matrix is a $4 \times n$ matrix M such that $M[i, j]$ (i=0,⋯,3, j=0,⋯, n -1) is the frequency of base i in position j in the set of DNA strings. Note that the frequency is between 0 and 1.

You have to write a program that takes as input the set of strings from the file "DNA-sequences.txt", computes and print the frequency matrix M. The program generates the following output (keep the fraction):

Frequencies:

A: 4/5, 1/5, 0, 1/5, 1/5, 0, 1/5, 2/5, 3/5,1/5

T: 0, …

G: 1/5, …

C: 0, …