Katherine Filpo Lopez
Professor Kerri-Ann Norton
September 30th 2020
Lab Report #3: DNA Translations andReading Frames

Introduction

The purpose of DNA is to code for proteins that carry out many important mechanisms for the cell to continue living efficiently. The focus of this lab was to simulate this. We were building code to model the way that DNA is converted to RNA which is then made into amino acids and proteins, the theory of which is better known as Central Dogma. It is especially important for any organism that is replicating its DNA to do these processes well as another cell cannot be created well if the DNA strand is not replicated correctly. Issues in DNA replication and making this DNA to proteins can be deadly to the cell or the organism that has this cell. Broken cells can be harmless, or they can be as deadly, as a cancer cell is a cell with problems in its DNA or the mechanism of replication.

Modeling DNA translations and the different reading frames of how this happens is also important for understanding other types of cells. Bacterial cells are very interesting and thinking of how they use their reading frames is needed because they replicate so rapidly. This is because bacteria are able to find multiple start sequences for their replication process and use those multiple starting points and DNA enzymes to replicate their whole genome at much faster speeds than eukaryotic cells. Overall, studying replication and translation has many advantages.

Methods

The code for this lab uses a class named Sequence to do most of the work. The Sequence class has all the important methods in it and they are called outside of the class wherever the user decides. The class starts by initializing a sequence and checking that it is a valid DNA sequence. Then the methods that do the required actions are named translateDNA, predictRF, longestRF. The latter two methods take no input and just run on a string that the method is called upon. The method translateDNA takes the input for the reading frame that the user wants. Input 0 returns and prints all six reading frames. Input 1 through 3 returns and prints a sequence that has been

translated from DNA to RNA and transcribed into amino acid codons and then their abbreviations. The difference between all the inputs is that the reading frame has been shifted to the right by one character. The inputs of 4 through 6 do the same as the previous inputs but they use the reverse translation of the sequence that is the input. These inputs also shift the reading frame of the sequence but instead of moving to the right, it moves to the left.

PredictRF calls on translateDNA and returns some frames of the sequences when there is a start codon. LongestFR calls on PredictRF and looks at all the possible frames and returns one frame that is the longest of all the other frames. The dictionaries of the codon keys and amino acids are outside of the Sequence class, simply because there seemed no reasons to put them in the sequence class.

Results

The code works somewhat adequately. The method translateDNA works very well and gives the user the reading frames that they ask for. As aforementioned, it works differently based on what the user decides to pick, but the code does the heavy work in the beginning and calculates all reading frame results and translates those frames to amino acids regardless of whether or not the user asks for those frames. The range that is set for the initialization is 1 to 8 even though we are only looking at 6 reading frames. I think that it is because I have made it so the 1st stored value is what is given when the user types in 0, but I'm not sure. If I make the range go to 7, the code breaks, so I kept it like this. Here is the beginning of the method:

```
1.  def translateDNA(self, rfP):
2.          rfs = []
3.          #starts from 0
4.          for i in range(1,8):
5.              if i>0 and i<4:
6.                  seqRep = self.dna
7.                  seqRep = self.transcribe()
8.                  rfs.append(self.translateRNAtoAA(seqRep[i:]))
9.              else:
10.                 seqRep = self.reverseTrans()
11.                 rfs.append(self.translateRNAtoAA(seqRep[i:]))
```

Here is an example of what prints out when the user inputs 0. This is just the first three results that I included but there are more:

```
Reading Frame 1 ['L', 'V', 'W', 'C', 'I', '_', 'L', 'L', 'R', 'R', 'S', 'L',
'P', 'L', 'L', 'P', 'C', 'G', 'A', 'R', 'C', 'E']
Reading Frame 2 ['W', 'Y', 'G', 'A', 'S', 'D', 'S', '_', 'G', 'E', 'V', 'C',
'R', 'Y', 'C', 'P', 'V', 'G', 'Q', 'G', 'V', 'K']
Reading Frame 3 ['T', 'P', 'C', 'P', 'T', 'G', 'Q', '_', 'R', 'Q', 'T', 'S',
'P', 'Q', 'E', 'S', 'D', 'A', 'P', 'Y', 'Q']
```

As for the predictFrame function, there are some small issues that I can't figure out. I originally had it finding both the starts and stop points of the reading frame being looked at, but I could not get it to work so I made a helper function to help remove some of the code so I could see it better, but unfortunately, I could not get it to work. I get the error that "slice indices must be integers or None or have an __index__ method" which confuses me because I am trying to use the index of the variable: "stop = tracker.index(el)." I don't know how to fix it. Without fixing and having a working predictFrame function, I cannot test longestFrame well, but I am confident that it is either good, or close to it.

Discussion

The code can read in sequences and the functions are alright, but it is overall incomplete.

Credits

I worked on this code with Sam and Olivia during the class session, and later worked on collaborating with Olivia for the translate and predict methods.