Bioinformatics Lab Assignment 3: Translation and Classes in Python

By Noah Segal-Gould

Background: For biologists who are interested in nucleotide data provided through organizations like the National Center for Biotechnology Information, it is useful to programmatically translate that nucleotide data into amino acid data. Using an object-oriented programming language like Python, these individuals are able to create classes which can then be used to analyze the potential reading frames present in amino acid sequences.

Methods: I created a `Sequence` class that has a constructor which accepts a single filename string as input and three methods which take no input: `translateDNA` shows potential reading frames by user selection, `predictRF` returns all potential matches between start and stop codons in all potential reading frames of the translated sequence in the form of a tuple of tuples, and `longestRF` returns a tuple for which the first value is the number of the reading frame with the largest start and stop codon match and the second value is that amino acid sequence match. I instantiate this class in a `main` function which includes several tests of the methods contained within the class. I decided to have my class' constructor do the majority of the work in my program. All translation, separations into possible reading frames, and searching through all of those reading frames for start and stop codons occurs in the constructor, and the other methods mostly just make small changes to the class attributes to return useful information. I decided to structure my program in this way because in practice, if the input file either does not open or contains invalid nucleotide characters, then the instantiation of the class into an object will fail immediately inside its constructor. I use regular expressions to find potential sequences with start and stop codons, as well as split apart the nucleotide sequences in the input files into groups of three nucleotides to make it easier to find amino applicable amino acids. I've also made a class attribute called `nucleotide_totals` that is a dictionary containing total counts of nucleotides as well as non-nucleotides. In this dictionary, the "`N`" key should always have a value of `0`.

Results: When the program is run, the `main` function is called and the tests are shown. I chose to make the `predictRF` method of the `Sequence` class return the entire tuple of 6 reading frames, each of which contains a tuple of numerous potential sequences with start and stop

codons, so the output overall is quite long. I also have the user inputting a whole number between 0 and 6 for my test of the `translateDNA` method which is similarly long depending on the length of the input file. Because the output is so long, below is a screenshot instead of plain output text:

Program Output:

```
Statistics for data/RFdata.txt:
{'A': 15, 'T': 18, 'C': 13, 'G': 23, 'N': 0}
Example use of longestRF method on data/RFdata.txt:
(2, 'MVHLTPEEKSAVTALWGKV_')
Statistics for data/RFdata2.txt:
{'A': 8213, 'T': 7789, 'C': 3445, 'G': 4575, 'N': 0}
Example use of longestRF method on data/RFdata2.txt:
(2, 'MLKKQFGNFGEKSRKVRVKMRKSGKHWVKSVMTQIGYVILSRFSGKEKSSKVQTTSEDLSRTKTSASILTAVAALGAVVGGTTDTTSVSAEETPTATELTGNEKTLATAETV
Statistics for data/RFdata3.txt:
{'A': 8213, 'T': 7789, 'C': 3445, 'G': 4575, 'N': 0}
Example use of longestRF method on data/RFdata3.txt:
(2, 'MLKKQFGNFGEKSRKVRVKMRKSGKHWVKSVMTQIGYVILSRFSGKEKSSKVQTTSEDLSRTKTSASILTAVAALGAVVGGTTDTTSVSAEETPTATELTGNEKTLATAETV
Example use of translateDNA method on data/RFdata3.txt:
You will be prompted to enter a specific reading frame.
Valid frames are 1, 2, 3, 4, 5, and 6.
If you would like to see all reading frames, you may enter 0.
==================== WARNING: ALL OTHER INPUTS WILL CAUSE THE PROGRAM TO HALT ====================
Enter a specific reading frame: 1
QKQLIGIF_T_IKISMVSPNCL_PIYNY_Q_NKAESCIDRTASKIFNFFHSRNFL_FFS_R_NLRLNYNIKFCYIR_RKVEFSMKKDKNLL_YCL_INLK_ILSLKEKSNQVLSYWK
Example use of predictRF method on data/RFdata3.txt:
(('MVSPNCL_', 'MKKDKNLL_', 'MITLYRYSKNLERSVKNLKNLAKQQPTACLIRMEITLVA_', 'MLFFRGTMNLFILIDK_', 'MKKFVSKV_', 'MMG_', 'MMS
```

Conclusion: I completed tasks 1 through 4 as assigned in the Lab Report 3 Assignment file. I may have also effectively completed the first challenge due to the tests I included within my program but I'm not completely sure. Overall, my outputs including the reading frames seem correct to me based on examples we did in class.

Acknowledgements: I worked alone on this assignment and made use of code shown in class as well as code that was provided on the course's Moodle 2.