nguyenet_Assignment3

June 9, 2020

1 Assignment #3

Write a function (or functions) to find all ORFs in a sequence string. Since we are only interested in relatively long ORFs, the function should only return those ORFs longer than an input threshold value. Compute the length of the ORFs as the number of codons (counting the start and stop codon). The start codon is "ATG". The stop codons are "TAG", "TAA", and "TGA".

The inputs to the function should be the sequence string and the threshold value. The output should be a list of lists. The length of the list should be the number of ORFs. Each element of the list should be another list with two numbers describing an individual ORF: the nucleotide position of the first position of the start codon and the nucleotide position of the first position of the in-frame stop codon.

For the example: sequence = AATGCCCAAATGCTTTTAAAACCCATGTAGC Your code (with threshold 5) should return [[1,16],[9,27]].

1.1 Write and test function

Make test sequence

[14]:
```
testseq = "AATGCCCAAATGCTTTTAAAACCCATGTAGC"
```

Function

[15]:
```
def ORFall(seq, thres):
    """finds all Open Reading Frames in 'seq' with at least 'thres" number of␣
↪codons"""
    ORFs = []  # keeps track of in frame ORFs, listing positions of beginning␣
↪of start and stop codons
    startcod = "ATG"
    stopcod = ["TAG", "TAA", "TGA"]
    for i in range(len(seq)):
        if seq[i:(i+3)] == startcod:  # if first three bases, beginning with␣
↪'i' is 'startcod'
            codons = 0  # keeps count of total codons in ORF
            stopPos = -1 # placeholder for position of stop codon
            for j in range(i, len(seq), 3):  # walks through sequence, three␣
↪bases (one codon) at a time, starting with 'i'
                codons += 1  # increments 'codons' count by 1
```

```
                if seq[j:(j+3)] in stopcod:  # if codon in 'stopcod'
                    stopPos = j # updates position of stop codon
                    break  # breaks out of 'for loop'
            if stopPos != -1:  # stopPos would only update IF the stopcodon was␣
↪found in the sequence (in frame)
                if codons >= thres:  # assesses if 'codons' is greater than or␣
↪equal to 'thres'
                    ORFs.append([i, j])
    return ORFs
```

Test function.

[16]:
```
testFun = ORFall(testseq, 2)
print(testFun)
```

```
[[1, 16], [9, 27], [24, 27]]
```

1.2 Apply function

1) Use the "loadFASTA" function from "lecture2functions.py" to study the "ACE2.fasta" file from Lecture 2.

Import lecture2functions and read in 'fasta' file.

[17]:
```
import lecture2functions as f2
ACE2 = f2.loadFASTA("ACE2.fasta")
```

2) Use your new ORF-finding function, with threshold 700, to find all ORFs.

[18]:
```
ACE2_ORFS = ORFall(ACE2, 700)
print(ACE2_ORFS)
print(len(ACE2_ORFS))
```

```
[[49, 2464], [232, 2464], [292, 2464]]
3
```

Using a threshold length of 700 codons, the forward strand of ACE2 ('ACE2.fasta' file) has 3 ORFs. Each of the three ORF has a unique start codon. However, all three ORFs have the same stop codon at position 2464. This was possible because "ATG" also codes for the amino acid methionine, in addition to being a start codon in eukaryotes.

3) Use the "revcomp" function also from "lecture2functions.py" to find all ORFs (if there are any) on the opposite strand.

[19]:
```
ACE2rev = f2.revcomp(ACE2)  # f2.revcomp returns the reverse complement of seq
ACE2rev_ORFS = ORFall(ACE2rev, 700)
print(ACE2rev_ORFS)
```

```
[]
```

Using a threshold length of 700 codons, there are no ORFs on the reverse complement strand of ACE2.