final-3

May 6, 2020

You will need to read the sequences from the file and concatenate multiple lines into a single sequence for each identifier. For the first part of this problem, you will write code to compute codon usage statistics for the "surface glycoprotein" (spike protein) of the viral coding sequences in covid19codingClean.fasta. The data file is a text file containing sequences in fasta format: a header line starting with the >" character that contains a description of the forthcoming sequence, and then the sequence itself, spread out over many lines.

```
[137]: from Bio import SeqIO
       s_protein_seq = []
       seq_name = []

       for seq_record in SeqIO.parse('covid19codingClean.fasta', "fasta"):
           genename = seq_record.description.split('|')[1].split('[')[0]
           if genename == "surface glycoprotein ":
               seq_name.append(seq_record.name)
               s_protein_seq.append(str(seq_record.seq))
```

```
[138]: seq_name[0]
```

```
[138]: 'NC_045512.2:21563..25384'
```

0.0.1 Your code should compute the number of occurrences of each possible codon in each surface glycoprotein sequence.

This includes stop codons, although you shouldn't see more than one of these in each sequence!

Your code should produce an output file containing data for a table where lines correspond to sequences, columns to codons, and the (integer) values are the number of occurrences of the particular codon in that particular spike protein sequence.

If you are looking at the file with coding sequences (covid19codingClean.fasta), then you have just coding sequences. The ***whole* sequence is coding.** So you **should *not* be finding stop codons in the middle of your sequences.** If you are, that's a sign there is something wrong with your code.

You may discard any codon that contains any letter other than A, C, G, or T.

```
[139]: codon_table = {"TTT" : "F", "CTT" : "L", "ATT" : "I", "GTT" : "V",
                      "TTC" : "F", "CTC" : "L", "ATC" : "I", "GTC" : "V",
                      "TTA" : "L", "CTA" : "L", "ATA" : "I", "GTA" : "V",
                      "TTG" : "L", "CTG" : "L", "ATG" : "Start", "GTG" : "V",
                      "TCT" : "S", "CCT" : "P", "ACT" : "T", "GCT" : "A",
                      "TCC" : "S", "CCC" : "P", "ACC" : "T", "GCC" : "A",
                      "TCA" : "S", "CCA" : "P", "ACA" : "T", "GCA" : "A",
                      "TCG" : "S", "CCG" : "P", "ACG" : "T", "GCG" : "A",
                      "TAT" : "Y", "CAT" : "H", "AAT" : "N", "GAT" : "D",
                      "TAC" : "Y", "CAC" : "H", "AAC" : "N", "GAC" : "D",
                      "TAA" : "STOP", "CAA" : "Q", "AAA" : "K", "GAA" : "E",
                      "TAG" : "STOP", "CAG" : "Q", "AAG" : "K", "GAG" : "E",
                      "TGT" : "C", "CGT" : "R", "AGT" : "S", "GGT" : "G",
                      "TGC" : "C", "CGC" : "R", "AGC" : "S", "GGC" : "G",
                      "TGA" : "STOP", "CGA" : "R", "AGA" : "R", "GGA" : "G",
                      "TGG" : "W", "CGG" : "R", "AGG" : "R", "GGG" : "G" }

          CodonsDict = {  "TTT": 0, "TTC": 0, "TTA": 0, "TTG": 0, "CTT": 0,
                  "CTC": 0, "CTA": 0, "CTG": 0, "ATT": 0, "ATC": 0,
                  "ATA": 0, "ATG": 0, "GTT": 0, "GTC": 0, "GTA": 0,
                 "GTG": 0, "TAT": 0, "TAC": 0, "TAA": 0, "TAG": 0,
                 "CAT": 0, "CAC": 0, "CAA": 0, "CAG": 0, "AAT": 0,
                 "AAC": 0, "AAA": 0, "AAG": 0, "GAT": 0, "GAC": 0,
                 "GAA": 0, "GAG": 0, "TCT": 0, "TCC": 0, "TCA": 0,
                 "TCG": 0, "CCT": 0, "CCC": 0, "CCA": 0, "CCG": 0,
                 "ACT": 0, "ACC": 0, "ACA": 0, "ACG": 0, "GCT": 0,
                 "GCC": 0, "GCA": 0, "GCG": 0, "TGT": 0, "TGC": 0,
                 "TGA": 0, "TGG": 0, "CGT": 0, "CGC": 0, "CGA": 0,
                 "CGG": 0, "AGT": 0, "AGC": 0, "AGA": 0, "AGG": 0,
                 "GGT": 0, "GGC": 0, "GGA": 0, "GGG": 0}
```

[144]:
```
def count_codon(seq):
    dna =  seq
    self_dict = CodonsDict.copy()
    for i in range(0, len(dna)-(3+len(dna)%3), 3):
        codon = dna[i:i + 3]
        if codon in codon_table:
            self_dict[codon] +=1
            if codon_table[codon] == 'STOP':
                break

    return self_dict
```

[123]:
```
import collections
col = ['accnum']
codon_dict = collections.OrderedDict(sorted(count_codon(s_protein_seq[0]).
 ↪items()))
for k,v in codon_dict.items():
    col.append(k)
print(col)
```

```
['accnum', 'AAA', 'AAC', 'AAG', 'AAT', 'ACA', 'ACC', 'ACG', 'ACT', 'AGA', 'AGC',
'AGG', 'AGT', 'ATA', 'ATC', 'ATG', 'ATT', 'CAA', 'CAC', 'CAG', 'CAT', 'CCA',
'CCC', 'CCG', 'CCT', 'CGA', 'CGC', 'CGG', 'CGT', 'CTA', 'CTC', 'CTG', 'CTT',
'GAA', 'GAC', 'GAG', 'GAT', 'GCA', 'GCC', 'GCG', 'GCT', 'GGA', 'GGC', 'GGG',
'GGT', 'GTA', 'GTC', 'GTG', 'GTT', 'TAA', 'TAC', 'TAG', 'TAT', 'TCA', 'TCC',
'TCG', 'TCT', 'TGA', 'TGC', 'TGG', 'TGT', 'TTA', 'TTC', 'TTG', 'TTT']
```

```
[12]: output = open('FPp3a-output','a')
      tmp = ','.join(col)
      output.write(tmp+'\n')
      output.close()
```

```
[13]: output = open('FPp3a-output','a')
      for i in range(0, len(s_protein_seq)):
          res=[]
          res.append(seq_name[i])
          codon_dict = collections.OrderedDict(sorted(count_codon(s_protein_seq[i]).
       ↪items()))
          for k,v in codon_dict.items():
              res.append(v)
          output.write(str(res).split('[')[1].split(']')[0]+'\n')
      output.close()
```

The nucleotide frequencies across the whole genome for that particular sequence are

0.0.2 A: 0.290, C: 0.189, G: 0.180, T: 0.330.

```
[40]: with open('FPp3a-output','r') as f:
          for num, line in enumerate(f):
              if num == 1:
                  first_save = line.strip('\n')
                  break
      observe = first_save.split(',')[1:]
      observe = [int(x) for x in observe]
      print(observe)
```

```
[38, 34, 23, 54, 40, 10, 3, 44, 20, 5, 10, 17, 18, 14, 14, 44, 46, 4, 16, 13,
25, 4, 0, 29, 0, 1, 2, 9, 9, 12, 3, 36, 34, 19, 14, 43, 27, 8, 2, 42, 17, 15, 3,
47, 15, 21, 13, 48, 0, 14, 0, 40, 26, 12, 2, 37, 0, 12, 12, 28, 28, 18, 20, 59]
```

```
[41]: print(col[1:])
```

```
['AAA', 'AAC', 'AAG', 'AAT', 'ACA', 'ACC', 'ACG', 'ACT', 'AGA', 'AGC', 'AGG',
'AGT', 'ATA', 'ATC', 'ATG', 'ATT', 'CAA', 'CAC', 'CAG', 'CAT', 'CCA', 'CCC',
'CCG', 'CCT', 'CGA', 'CGC', 'CGG', 'CGT', 'CTA', 'CTC', 'CTG', 'CTT', 'GAA',
'GAC', 'GAG', 'GAT', 'GCA', 'GCC', 'GCG', 'GCT', 'GGA', 'GGC', 'GGG', 'GGT',
'GTA', 'GTC', 'GTG', 'GTT', 'TAA', 'TAC', 'TAG', 'TAT', 'TCA', 'TCC', 'TCG',
'TCT', 'TGA', 'TGC', 'TGG', 'TGT', 'TTA', 'TTC', 'TTG', 'TTT']
```

[101]:
```
n = sum(observe)
ref = {"A": 0.290, "C": 0.189, "G": 0.180, "T": 0.330}
exp = []
for i in range(1,len(col)):
    prob = 1.0
    tmp = col[i]
    for j in range(0,3):
        if tmp[j] in ref:
            prob *= ref.get(tmp[j])
    exp.append(prob*n)
```

[102]:
```
import numpy as np
obs = np.array(observe)
print("exp ", np.array(exp))
print("obs ", obs)
```

```
exp  [31.047197   20.2342077  19.270674   35.329569   20.2342077  13.18712157
 12.5591634  23.0251329  19.270674   12.5591634  11.961108   21.928698
 35.329569   23.0251329  21.928698   40.202613   20.2342077  13.18712157
 12.5591634  23.0251329  13.18712157  8.59436544  8.18510994 15.00603489
 12.5591634   8.18510994  7.7953428  14.2914618  23.0251329  15.00603489
 14.2914618  26.2010133  19.270674   12.5591634  11.961108   21.928698
 12.5591634   8.18510994  7.7953428  14.2914618  11.961108    7.7953428
  7.424136   13.610916   21.928698   14.2914618  13.610916   24.953346
 35.329569   23.0251329  21.928698   40.202613   23.0251329  15.00603489
 14.2914618  26.2010133  21.928698   14.2914618  13.610916   24.953346
 40.202613   26.2010133  24.953346   45.747801  ]
obs  [38 34 23 54 40 10  3 44 20  5 10 17 18 14 14 44 46  4 16 13 25  4  0 29
  0  1  2  9  9 12  3 36 34 19 14 43 27  8  2 42 17 15  3 47 15 21 13 48
  0 14  0 40 26 12  2 37  0 12 12 28 28 18 20 59]
```

[103]:
```
chi_suqare = ['NC_045512.2']
for i in range(0,len(exp)):
    X = (np.square(obs[i] - exp[i]))/exp[i]
    chi_suqare.append(X)
```

[104]:
```
col[0] = 'chi_suqare'
with open('FPp3b','a') as f:
    tmp = ','.join(col)
    f.write(tmp+'\n')
    f.write(str(chi_suqare).split('[')[1].split(']')[0]+'\n')
```

[112]:
```
chi_sort = np.argsort(chi_suqare[1:])[::-1]
print(chi_sort)
last_codon = []
last_idx = []
for i in range(0,len(chi_sort)):
    if obs[chi_sort[i]]<=exp[chi_sort[i]]:
        last_codon.append(col[chi_sort[i]+1])
        last_idx.append(chi_sort[i])

print(last_codon)
print(last_idx)
```

```
[43 39 48 16 56 50 47 35  4  7 36 23 24 32 20 54  3  1 30 28 12 22  6 41
 17 25  9 55 19 38 26 63 60 31 49 13 33 45 14 42 61 21 44 40 27  0 11 62
 18  5  2 29 53 52 59 57 15 34 10 58  8 46 37 51]
['TAA', 'TGA', 'TAG', 'CGA', 'TCG', 'CTG', 'CTA', 'ATA', 'CCG', 'ACG', 'CAC',
'CGC', 'AGC', 'CAT', 'GCG', 'CGG', 'TTA', 'TAC', 'ATC', 'ATG', 'GGG', 'TTC',
'CCC', 'GTA', 'CGT', 'AGT', 'TTG', 'ACC', 'CTC', 'TCC', 'TGC', 'AGG', 'TGG',
'GTG', 'GCC', 'TAT']
[48, 56, 50, 24, 54, 30, 28, 12, 22, 6, 17, 25, 9, 19, 38, 26, 60, 49, 13, 14,
42, 61, 21, 44, 27, 11, 62, 5, 29, 53, 57, 10, 58, 46, 37, 51]
```

[117]:
```
with open('FPp3b-rare','a') as f:
    for i in range(0,5):
        f.write(last_codon[i])
        f.write('\t'+str(round(chi_suqare[last_idx[i]]))+'\n')
```