nguyenet_Assignment1

June 2, 2020

1 Assignment #1

2 A. Turn in the code for the three Python functions.

2.1 Prompt 1

1) Write a Python function that takes as input a sequence string and returns a list with 4 entries that are the number of A, C, G, and T in the sequence.

Make test code.

[1]:
```
jiggy = "ATGTTTTATCCAGGTGATTAA"
```

Function for prompt 1.

Function requires one input: a sequence string

Creates a variable for a each base and assigns all to a value of zero. Steps through the sequence, one base at a time, and increases the count for each base as the loop progresses through the sequence. Creates a list of list of 4 entries with each base and its associated count for said sequence. Return the list of list of base counts.

[2]:
```
def baseNum(seq):
    '''counts how many times each base appears in sequence'''
    Abase = 0
    Cbase = 0
    Gbase = 0
    Tbase = 0
    for base in seq:
        if base == "A":
            Abase += 1
        elif base == "C":
            Cbase += 1
        elif base == "G":
            Gbase += 1
        elif base == "T":
            Tbase += 1
    baseAll = ["A: " + str(Abase),
               "C: " + str(Cbase),
               "G: " + str(Gbase),
```

```
                "T: " + str(Tbase)
                ]
    return baseAll
```

Test function using previously defined test code.

[3]:
```
print(baseNum(jiggy))
```

```
['A: 6', 'C: 2', 'G: 4', 'T: 9']
```

2.2 Prompt 2

2) Write another Python function that takes two inputs: a sequence string and a string of two letters (e.g., "CG" or "CT"). This function returns the number of times the two letters occur consecutively in the sequence.

Function for prompt 2.

Function requires 2 in puts: 1) a sequence string 2) a string of 2 bases, creating the 'basePair' of interest

Creates a variable called 'paircount' for the count of the number of 'basePair' encountered. Steps through the sequence, and looks at two bases at a time, but incrementing through the sequence one base at a time. Assigns the two base string to the variable 'temp'. Compares the 'temp' variable to the 'basePair' input. If they match, increase the 'paircount' by 1. Increment i by 1, then continue through the while loop. Return 'basePair' with its frequency in the sequence, which was tracked in 'paircount'.

[4]:
```
def countPair(seq, basePair):
    '''counts how many times a pair of bases appears in sequence'''
    paircount = 0
    i = 0
    while i < len(seq):
        temp = str(seq[i:(i+2)])
        if temp == basePair:
            paircount += 1
        i += 1
    return basePair + ": " + str(paircount)
```

Test function using previously defined test code.

[5]:
```
print(jiggy)
print(countPair(jiggy, "AT"))
print(countPair(jiggy, "TT"))
```

```
ATGTTTTATCCAGGTGATTAA
AT: 3
TT: 4
```

2.3 Prompt 3

3) Write another Python function that takes as input a sequence string and returns a list with 16 entries that are the outputs of function #2 for all 16 possible two letter strings.

Function for prompt 3.

Function requires one input: a sequence string

Creates an empty list called 'allCount'. For said sequence, runs the function 'countPair' for all 16 combination of 2 base pairs. Append the result for each 'countPair' run to the 'allCount' list. Returns 'allCount' list with count of each combination of 2 base pairs.

[6]:
```
def allPairCount(seq):
    '''counts how many times each pair of bases appear in sequence'''
    allCount = []
    allCount.append(countPair(seq, "AA"))
    allCount.append(countPair(seq, "AT"))
    allCount.append(countPair(seq, "AC"))
    allCount.append(countPair(seq, "AG"))
    allCount.append(countPair(seq, "TA"))
    allCount.append(countPair(seq, "TT"))
    allCount.append(countPair(seq, "TC"))
    allCount.append(countPair(seq, "TG"))
    allCount.append(countPair(seq, "CA"))
    allCount.append(countPair(seq, "CT"))
    allCount.append(countPair(seq, "CC"))
    allCount.append(countPair(seq, "CG"))
    allCount.append(countPair(seq, "GA"))
    allCount.append(countPair(seq, "GT"))
    allCount.append(countPair(seq, "GC"))
    allCount.append(countPair(seq, "GG"))
    return allCount
```

Test function using previously defined test code.

[7]:
```
print(jiggy)
allPairCount(jiggy)
```

```
ATGTTTTATCCAGGTGATTAA
```

[7]:
```
['AA: 1',
 'AT: 3',
 'AC: 0',
 'AG: 1',
 'TA: 2',
 'TT: 4',
 'TC: 1',
 'TG: 2',
 'CA: 1',
```

```
 'CT: 0',
 'CC: 1',
 'CG: 0',
 'GA: 1',
 'GT: 2',
 'GC: 0',
 'GG: 1']
```

3 B. For each of the two FASTA files, turn in the output of functions #1 and #3.

Import python script with function for reading in FASTA files

[8]:
```
import lecture2functions as f2
```

Load FASTA files for the human gene PTPN11 and it's Drosophila orthologue csw. Look at beginning of each sequence, as well as the length of each sequence.

[9]:
```
PTPN11 = f2.loadFASTA("PTPN11.fasta")
print(PTPN11[0:20])
print(len(PTPN11))

csw = f2.loadFASTA("csw.fasta")
print(csw[0:20])
print(len(csw))
```

```
AGTCTCCGGGATCCCCAGGC
6073
ATTCATTCATACCCCAGCGC
7664
```

3.1 PTPN11

1) Count how many times each base appears in the sequence (function #1).

[10]:
```
baseNum(PTPN11)
```

[10]:
```
['A: 1773', 'C: 1139', 'G: 1410', 'T: 1751']
```

2) Count how many times each pair of bases appear in the sequence (fucntion #3).

[11]:
```
allPairCount(PTPN11)
```

[11]:
```
['AA: 595',
 'AT: 438',
 'AC: 280',
 'AG: 459',
```

```
 'TA: 340',
 'TT: 606',
 'TC: 308',
 'TG: 497',
 'CA: 394',
 'CT: 371',
 'CC: 275',
 'CG: 99',
 'GA: 443',
 'GT: 336',
 'GC: 276',
 'GG: 355']
```

3.2 csw

1) Count how many times each base appears in the sequence (function #1).

[12]:
```
baseNum(csw)
```

[12]:
```
['A: 2395', 'C: 1876', 'G: 1675', 'T: 1718']
```

2) Count how many times each pair of bases appear in the sequence (fucntion #3).

[13]:
```
allPairCount(csw)
```

[13]:
```
['AA: 870',
 'AT: 523',
 'AC: 521',
 'AG: 481',
 'TA: 404',
 'TT: 495',
 'TC: 402',
 'TG: 416',
 'CA: 621',
 'CT: 348',
 'CC: 452',
 'CG: 455',
 'GA: 499',
 'GT: 352',
 'GC: 501',
 'GG: 323']
```

4 C. Since humans have much higher rates of methylation than Drosophila, we would expect to see far fewer CpGs in humans. Is this what we see?

"The CpG sites or CG sites are regions of DNA where a cytosine nucleotide is followed by a guanine nucleotide in the linear sequence of bases along its 5' → 3' direction. CpG sites occur with high frequency in genomic regions called CpG islands" (Wikipedia, accessed May 27, 2020).

Determine number of CpGs in the human gene PTPN11.

[14]:
```
countPair(PTPN11, "CG")
```

[14]:
```
'CG: 99'
```

Determine number of CpGs in the Drosophila orthologue csw.

[15]:
```
countPair(csw, "CG")
```

[15]:
```
'CG: 455'
```

In support of our hypothesis, we see far fewer CpGs in the human PTPN11 compared to the Drosophila csw gene.