lucasrocmoreira

In [19]:

**import** **random**

**def** randomseq(N):

*"""*

*returns a random sequence of bases of length N.*

*"""*

bases = ['C','G','A','T']

seq\_list = random.choices(bases,k=N)

seq = "".join(seq\_list)

**return** seq

randomseq(10)

Out[19]:

'CTTGCCGTGC'

***COMMENT: Nice job with this code. Quick and efficient. Looks good!***

B. Write a function to calculate and return the frequency of As, Cs, Ts and Gs in a sequence.

In [26]:

**def** basefreq(seq):

*"""*

*calculates and return the frequency of As, Cs, Ts and Gs in a sequence.*

*"""*

sequence = list(seq)

length = len(seq)

print("Frequency of C: ", sequence.count('C')/length)

print("Frequency of G: ", sequence.count('G')/length)

print("Frequency of A: ", sequence.count('A')/length)

print("Frequency of T: ", sequence.count('T')/length)

seq\_example = randomseq(30)

basefreq(seq\_example)

Frequency of C: 0.3

Frequency of G: 0.2

Frequency of A: 0.2

Frequency of T: 0.3

**COMMENT: In this code, the questions asks to return the frequency rather than the fraction of bases in the sequence, so if you take out the “/length” you got it.**

C. Write a function to concatenate (join end-to-end) two sequences and return it

In [27]:

**def** joinseq(seq1,seq2):

*"""*

*concatenates (joins end-to-end) two sequences and return it.*

*"""*

**return** seq1+seq2

seq\_A = randomseq(30)

seq\_B = randomseq(10)

joinseq(seq\_A,seq\_B)

Out[27]:

'CGGTCGGCCTTGCATTAGAAAGTTTGGCTGTTTCCGGTAC'

D. Write a function to take two sequences of different lengths and return both trimmed down to be the same length.

In [31]:

**def** trimseq(seq1,seq2):

*"""*

*concatenates (joins end-to-end) two sequences and return it.*

*"""*

slen = min([len(i) **for** i **in** (seq1, seq2)])

**return**(seq1[:slen],seq2[:slen])

trimseq(seq\_A,seq\_B)

Out[31]:

('CGGTCGGCCT', 'TTTCCGGTAC')

**Comment: Nice job, looks good. Efficient code!**

E. Write a function to return the proportion of bases across the shared length between two sequences that are the same. In this function, use the function that you created in D above to convert the sequences to be the same length (even if this is not necessarily the most efficient way to complete this task).

In [32]:

**def** basefreqseqs(seq1,seq2):

*"""*

*returns the proportion of bases across the shared length between two sequences that are the same.*

*"""*

same\_len\_seq = trimseq(seq1,seq2)

print("Sequence 1")

basefreq(same\_len\_seq[0])

print("Sequence 2")

basefreq(same\_len\_seq[1])

basefreqseqs(seq\_A,seq\_B)

Sequence 1

Frequency of C: 0.4

Frequency of G: 0.4

Frequency of A: 0.0

Frequency of T: 0.2

Sequence 2

Frequency of C: 0.3

Frequency of G: 0.2

Frequency of A: 0.1

Frequency of T: 0.4

**Comment: The wording of the question here was confusing; you answered a different question. What he is asking is for you to compare the two sequences and see what proportion of Cs, for example, are in the same place in both sequences, and do that for all the bases.**

**Finished**

Save this notebook and close it. Push a copy of the notebook to the assignment/ directory with your name in the filename like ./assignment/<myname>-3.4.ipynb.