Introduction to Python
BioPython Exercises

Stephanie Spielman
Email: stephanie.spielman@gmail.com

Working Sequence Data in BioPython

1. The first exercise will focus on manipulating sequences in the file "dopamine_sequences.fasta". Perform the following tasks:

   (a) Use the SeqIO module to read and save records from the file "dopamine_sequences.fasta".

   (b) Determine the length for each sequence, and save this information to a dictionary. The keys should be the sequence ID, and the values should be the length.

   (c) Translate each sequence to amino acids, and count the number of tryptophans (coded as "W") per sequence. Again, save this information in a dictionary. (Hint: BioPython has implemented certain useful string methods, like `.count()` for `Seq` object sequences.)

   (d) Re-do parts (b) and (c) by writing *functions* to count the sequence length and determine the number of tryptophans. Each function should take a *single* `SeqRecord` object as the argument.

2. The first exercise will focus on manipulating sequences in the sequence alignment file "dopamine_alignment.fasta", from Spielman et al. [2015]. This file contains the same sequences as does "dopamine_sequences.fasta", except the sequences are aligned and in amino-acid space. Perform the following tasks:

   (a) Use the SeqIO module to read and save records from the file "dopamine_alignment.fasta".

   (b) Translate each records to amino-acids. These sequences are known to have a motif "NPxxY", where the "xx" pair is either "II", "VV", or "VI", towards the end of their sequence. In this alignment, this motif appears at positions 745 - 750 (although remember that Python indexing starts at 0!). Determine how many sequences have each motif, and then print your results to screen (e.g. # sequences have NPIIY, # sequences have NPVIY, and # sequences have NPVVY). Make sure that the total number of motifs you count equals the number of sequences in the file (you should code this, not manually/visually check it).

   (c) Save these sequences to a new file, called "dopamine_alignment.phy" in phylip format. Perform this step twice, once using the `.convert()` method and once using the `.write()` method.

Scripting

1. Write a complete python script which determines the average GC-content across all sequences in a given file. Your script should use the `sys` module (specifically, the `sys.argv` variable) to obtain the file name, as a command line argument. Your script should include two functions: a function which reads in a sequence file, and a function which computes the GC-content for a single sequence.

2. Write a complete python script which determines the pairwise distance between the first two sequences in a provided *sequence alignment* file. Your script should use the `sys` module (specifically, the `sys.argv` variable)

to obtain the file name, as a command line argument. our script should include two functions: a function which reads in a sequence alignment file, and a function which computes pair-wise distance.

References

SJ Spielman, K Kumar, and CO Wilke. Comprehensive, structurally-informed alignment and phylogeny of vertebrate biogenic amine receptors. *PeerJ*, 3:e773, 2015. doi: 10.7717/peerj.773.