Programming for Bioinformatics | BIOL 7200

Exercise 5

Instructions for submission

- Name your script file(s): "gtusername_questionnumber.py" or as instructed

NOTE starting this week, unless explicitly instructed to write in Bash, only Python solutions will be accepted. No Bash scripts!

Grading Rubric

This assignment will be graded out of 100.

You only need to write one script for this assignment. **No Bash code allowed this week!!!** Your solution must be all Python code in a script.

Beginning this week, incorrectly named scripts will carry a 10 point penalty per script. Proper file naming is important for the grading process. If you resubmit your script and Canvas automatically adds a number to it, that is fine, as long as you had otherwise followed the required naming format.

Question 1 (100 points)

In many cases, the formats used to store biological data are not easy for humans to read and extract information from. Therefore, it is valuable to be able to wrote code to transform commonly used formats into human-readable representations.

An example of such data is sequence alignments. Sequence alignments are often represented in FASTA format. Alignments look much like the FASTA files you have been working with so far in this course, but they can include gap characters to indicate positions that are absent in one or more sequence.

For a human to look at a FASTA formatted alignment and identify the shared and distinct positions is quite difficult. Have a look at the following alignment, for example.

```
>seq1
ATGCAAGTCGAGCGGATGAAGGGAGCTTGCTCCTGGATTCAGCGGCGGAC
>seq2
ATGCAAGTCGAGCGGCAGCACAGAGGAACCTTGGGTGGCGAGCGGCGGAC
```

Identifying bases that are the same requires you to scan through and compare every position in the two sequences manually. This becomes even more intolerable if you recall that FASTA format allows sequences that can be millions of bases long and split over multiple lines. Manual comparison of FASTA format sequences in those cases is impossible.

Luckily, we can convert FASTA formatted sequences to more comfortably viewed forms. We can line up the sequences for easy viewing and can adda additional information to assist with identifying which bases are the

same and which are different. For example, with a fairly simple transformation, we can render the above alignment like this:

```
ATGCAAGTCGAGCGGATGAAGGGAGCTTGCTCCTGGATTCAGCGGCGGAC
|||||||||||||||  | |  |||    ||   |     ||||||||||
ATGCAAGTCGAGCGGCAGCACAGAGGAACCTTGGGTGGCGAGCGGCGGAC
```

It is much easier to view that alignment and spot the positions that differ.

Your task this week is to write a script to perform the above transformation. Your script should do the following:

1. Read a file whose path is provided as a command line argument. You should assume the file contains aligned DNA sequences in fasta format (i.e., the sequences are the same length).
2. print the sequences to the terminal without headers
3. As seen in the above example, add a pipe symbol between bases that are identical and a space between bases that differ

Your script must take command-line input. Do not hard-code the path to the sequence file. The usage of your script should be

`<script name>.py <FASTA file>`

In order for your script to have the described usage, you need a shebang. Unlike Bash, you can't assume that Python is in the user's /usr/bin/ directory. You must use a portable shebang (see the lecture slides for an example). **From now on, incorrect shebangs carry penalties!** This is true for every executable script you submit.

You do not need to worry about representing long sequences. Specifically, you don't need to account for how lines wrap in a terminal window. You can assume that each sequence will fit on a single line in the terminal.

Extra credit (20 points)

If you submit a solution for this question. Name your script "gtusername_EC.py"

It is an effective strategy when learning a second programming language to write scripts in both languages to see the similarities and differences between the languages. For this question, your task is to rewrite the final Bash assignment in Python, but with a few differences. The differences are:

1. You should run BLAST outside of the Python script using the following command:

   ```
   tblastn -query HK_domain.faa -subject Vibrio_cholerae_N16961.fna -outfmt '6
   std qlen' > Vc_blastout.txt
   ```

   Note the absence of `-task tblastn-fast`. You may have noticed last week that using that option you get fewer homolog predictions.

2. You should **NOT** use `awk` to process the BLAST output. Instead, your script should read in the unprocessed BLAST output and the "Vibrio_cholerae_N16961.bed" file and all processing should be

performed within a Python script using only Python code.

To remind you, BLAST hits should be processed (using Python) to only keep hits with greater than 30% identity and >= 90% length.

Your script should write the unique list of identified homolog genes to an output file (specified in the commandline) and should print the number of homologs identified.

The usage of your script should be

```
<script name>.py <blast output> <BED file> <output file>
```

Using the specified BLAST command, you should get 34 homologs for *Vibrio cholerae*. You may use the example Bash scripts included in the exercise files as the basis for this script if the script you wrote did not get the correct number of hits.

You already have the input data from the previous assignment.