Programming for Bioinformatics | BIOL7200

Week 12 Exercise

November 5, 2019

This week's assignment consists of two main graded assignment.

For this week, assume that the user always specifies the input correctly. Your script will be graded on the output produced and not how all the errors are handled.

The CI for this week will also not be public, i.e. you will not be able to see the autograder. We will run your script on the CI after the submission deadline.

Again, please do not use any modules other than "sys", "re", and "argparse". The CI will not install missing modules. Do not use `input()` for any input either, we do not handle this in the CI. The CI will fail if you do not follow these instructions.

Instructions for submission

- This assignment is due Monday, November 11, 2019 at 11:59pm. Late submissions will receive a 0
- Your code must be available on GitLab at the above time to be graded
- Name the all-to-fasta script as `all2fasta.py` and the self-overlap element counting script as `elementCount.py`
- Your code should run as
    - `./all2fasta.py [-f FOLD] -i <input file name>`
    - `./elementCount.py -i <input file name>`
- `all2fasta.py` should generate an output file while `elementCount.py` should print output to `STDOUT`
- DO NOT HARDCODE any file name!
- Please use the `#!/usr/bin/env python3` as your shebang
- Please also provide an intuitive help text in your `argparse`

All-to-fasta assignment

Max score: 50 points

FASTA is generally acknowledged to be the greatest format ever. Therefore, this week you will be writing a script to convert any file into a FASTA file with the same name, but with an `.fna` or `.faa` extension, depending on whether or not it contains nucleic acid or amino acid sequence. It is up to you and your script to *determine the file format and the sequence type*.

It should accept the following file types:

Format	Example description
EMBL	https://www.genomatix.de/online_help/help/sequence_formats.html#EMBL
FASTQ	https://www.genomatix.de/online_help/help/sequence_formats.html#FASTQ
GenBank	https://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html
MEGA	http://jordan.biology.gatech.edu/biol7200/MegaFormatDescription.docx

Please note that the above descriptions are very minimal, but should help you identify the key items in the format.

Guessing whether input sequence is protein or nucleotide is imperfect but an easy (and correct) assumption will be assuming that the DNA sequence only contains the character set: `[ACGTNacgtn]`.

You script should be invoked like this: `./all2fasta.py -i a_file.gbk`
This would produce a file called `a_file.fna` if `a_file.gbk` has DNA in it or `a_file.faa` if `a_file.gbk` had an amino acid sequence.

Deliverable: Python script (`all2fasta.py`) which takes on file as the argument from the command line.

Syntax:
`./all2fasta.py [-f FOLD] -i <input file name>`

The optional argument that could be provided to `all2fasta.py` is `-f` that specifies the line fold, i.e., after how many bases should a new line be inserted. Default value should be `70`. This option will have no effect on the sequence description line.

Example usage:

```
# this will print 80 bases per line
./all2fasta.py -i gb.in

# this will print 20 bases per line
./all2fasta.py -f 20 -i gb.in
```

Additional instructions:

- Do not print to STDOUT. If the input file has an extension, replace the extension from the input file name with the appropriate extension (`.fna` or `.faa`) to generate the output file name; if the input file does NOT have an extension, append the corresponding extension (`.fna` or `.faa`) to the input file name to generate the output file name.

- Do not make your script work on extension detection – our test files will have no extensions

Self-overlap element counting assignment

Max score: 50 points

The objective of the assignment is simple – given a set of coordinates represented in a BED file, find the number of times each coordinate occurs. As an example, consider the following BED file:

```
chr1    10      15
chr1    12      15
chr1    13      20
```

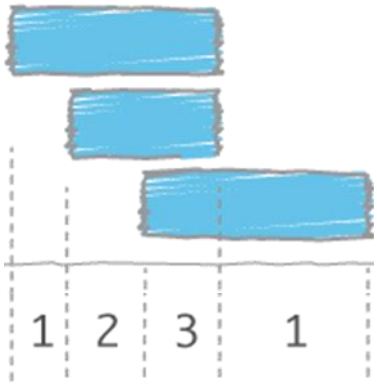
where each line represents a genomic element, the first column is the chromosome, second column is the starting position of the element and the third column is the ending position of the element.

Please note

1. *Write the output to STDOUT*
2. *The third column is exclusive. I.e.,* `chr1 10 12` *represents an element that starts at 10 and ends at 11.*
3. *The BED file may have more than 3 columns, but the first three will always represent chromosome, start and stop.*

Your script should be able to process the BED file and produce the following output:

```
chr1    10      12      1
chr1    12      13      2
chr1    13      15      3
chr1    16      20      1
```



Where the last column is the coverage and represents the coverage of the coordinates for that row/element.

Please ensure that your script produces output coordinates that are non-overlapping.

Deliverable: Python script (`elementCount.py`) which takes on file as the argument from the command line.
Syntax: `./elementCount.py [-h] -i <input file name>`

Example usage:

```
./ elementCount.py -i input.bed
```