Programming for Bioinformatics | BIOL7200

Week 11 Exercise

November 8, 2022

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

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.

Again, please do not use any modules other than "sys", "re", and "argparse". Do not use `input()` for any input either.

Instructions for submission

- This assignment is due Monday, November 14, 2022 at 11:59pm. Late submissions will receive a 0
- 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 the 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 sequences. 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 Or http://scikit-bio.org/docs/0.5.2/generated/skbio.io.format.embl.html
FASTQ	https://www.genomatix.de/online_help/help/sequence_formats.html#FASTQ Or https://en.wikipedia.org/wiki/FASTQ_format
GenBank	https://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html
MEGA	http://jordan.biology.gatech.edu/biol7200/MegaFormatDescription.docx
SAM	Described below
VCF	Described below

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

Guessing whether input sequence is protein or nucleotide is imperfect, but an easy (and correct) assumption is to assume that a DNA sequence only contains the character set: **[ACGTNacgtn]**.

You script should be invoked like this: **./all2fasta.py -i a_file.gbk**
This will 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 can 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 70 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 file extensions

Format Description – SAM file:
Full format description is here: https://samtools.github.io/hts-specs/SAMv1.pdf

Format has this basic structure:

Header	@SQ SN:sequence_name LN:987				
	@PG ID:bwa PN:bwa VN:0.7.17-r1198-dirty CL:bwa mem -t 4 …				
Reads	BIOL7200:113:000000000-GTECH:1:1101:15700:1335 77	*	0	…	
	BIOL7200:113:000000000-GTECH:1:1101:15700:1335 141	*	0	…	

The actual data fields (SN, LN, ID, VN, CL) can change from file to file but, for the purpose of this assignment, assume that all SAM files will:

1. Have a header present (lines starting with @)
2. The header will always come before the reads data
3. The headers will have at minimum @SQ and @PG. It can have more fields (e.g., @HD) and @SQ and @PG may not be the first two lines. I.e., this is also a valid structure:

   @HD VN:1.6 SO:coordinate
   @SQ SN:ref LN:45
   @PN bwa
   @PG ID:bwa …

The reads will have at minimum 11 columns. Columns 12 and onwards are optional and program specific. Each column will be tab-separated.

How do you make a FASTA file out of this?
Taking the first two lines from the sample.sam file:

1	2	3	4	5	6	7	8	9	10	11	12	13
BIOL7200:	77	*	0	0	*	*	0	0	ATTTTGTT	GGHHHHH	AS:i:0	XS:i:0
BIOL7200:	141	*	0	0	*	*	0	0	TGTTGCTC	GGGHHHH	AS:i:0	XS:i:0

The FASTA format will be:
>line1_column1
line1_column_10
>line2_column1
line2_column_10

So, for this example, the FASTA file will be (assuming fold value of 50 bases):

>BIOL7200:113:000000000-GTECH:1:1101:15700:1335
ATTTTGTTTCTATTGCTTCAGTCACACACGAGAATACTCTGCGCTATGTA
AATCAATTCCGCTTTGACAACCAGCATAATTTAAAACTGGCTCGTCAGAT
AAGTGAAGTAAAGTATGTCAATAATTCGTGGATAGCTTATGATGTGAAAC
AAACTGAATTTGCAGAGGATCATACCAAAGCCTATCATTTCGATACCTTG
CCGTGGGATGTGCCAGTGAAACCGGAAATTC
>BIOL7200:113:000000000-GTECH:1:1101:15700:1335
TGTTGCTCACGTAAATAGCGGTTCAATTCATGTAAAGTCATTTCATCCGG
ATTGGTGCCCGTAATTTTAAGAATTTCCGGTTTCACTGGCACATCCCACG
GCAAGGTATCGAAATGATAGGCTTTGGTATGATCCTCTGCAAATTCAGTT
TGTTTCACATCATAAGCTATCCACGAATTATTGACATACTTTACTTCACT
TATCTGACGAGCCAGTTTTAAATTATGCTGG

Format Description – VCF file:
Full format description is here: https://samtools.github.io/hts-specs/VCFv4.2.pdf

Here is a simplified overview:

```
##fileformat=VCFv4.2
## ...
#CHROM  POS  ID  REF  ALT  QUAL  FILTER  INFO  FORMAT  sample1  sample2  sample3  ...
NC_01234 32  .  C  A  0  .  AB=0;…  GT:DP:…  1:310:…  0:310:…  1:193:…  ...
NC_01234 42  .  G  T,A  0  .  AB=0;…  GT:DP:…  1:266:…  2:288:…  0:237:…  ...
NC_01234 69  .  GGG  GCC  918.101  .  AB=0;…  GT:DP:…  1:280:…  1:290:…  1:280:…  ...
```

The key points to remember here are:

1. A VCF file will start with header lines
2. The header will have multiple sections
3. The first line in the header will be ##fileformat=VCFv…
4. The last line will be #CHROM…

5. A VCF file will have at least 10 columns: 9 columns describing the variant, and columns 10 and onwards describing the presence of the variant in the different samples
6. Columns 10 and onwards are the samples
7. Each column is separated by tabs
8. Columns 10 and onwards contain the structure described in column 9 (FORMAT). We only care about the GT field for this assignment. The GT field will be an integer: 0, 1, 2, 3, etc.
9. Columns 4 (REF) and 5 (ALT) describe the base(s) that is present in the samples
10. REF base is denoted by 0, ALT bases are denoted as 1, 2, 3, etc. The first ALT base is 1, the second ALT base is 2, and so on.

How do you make a FASTA file out of this?
Going with this small example:

```
##fileformat=VCFv4.2
## ...
#CHROM  POS  ID  REF  ALT  QUAL  FILTER  INFO  FORMAT  sample1  sample2  sample3  ...
NC_01234  32  .  C  A  0  .  AB=0;...  GT:DP:...  1:310:...  0:310:...  1:193:...  ...
NC_01234  42  .  G  T,A  0  .  AB=0;...  GT:DP:...  1:266:...  2:288:...  0:237:...  ...
NC_01234  69  .  GGG  GCC  918.101  .  AB=0;...  GT:DP:...  1:280:...  1:290:...  0:280:...  ...
```

The FASTA file will be:
>NC_01234
CGGGG
>sample1
ATGCC
>sample2
CAGCC
>sample3
AGGGG

Note that we are expecting you to also add the reference sequence (NC_01234).

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:

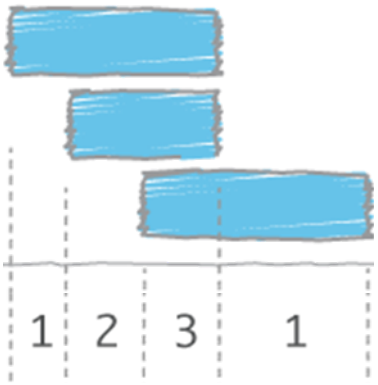
```
chr1  10  15
chr1  12  15
chr1  13  20
```

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    15    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