Programming for Bioinformatics | BIOL 7200

Exercise 9

Instructions for submission

- Download the data file, exercise9_data.tar.gz, and the python package ispcr.tar.gz into your working directory
- Name your package tarball: "gtusername.tar.gz"
- Upload your submission file on canvas.

Grading Rubric

This assignment will be graded out of 100.

Submission specifications

1. Your module ***does*** need to be generalizable to any FNA input files. We're going to be using this code to analyze different files moving forward. Don't hard code anything to the provided example files
2. You may only use Python standard library modules again this week. e.g., No Numpy or Pandas
3. Your submission should consist of a tar.gz file containing your package and the scripts addressing each question (described below).

Assignment description

This week's assignment is a similar format to last week. I have provided you with some data files and some scripts which will call a package that you write. This week we are going to extend the isPCR package that you wrote for the last exercise and add additional functionality that will start to take the shape of your magnum opus for this course.

More details of your assignment are provided below. Briefly, you will perform the following:

1. Slightly refactor your isPCR package so that it runs from start to finish in a single call
2. Write a Python module that includes an implementation of the Needleman-Wunsch algorithm described in class
3. Set up your isPCR and NW modules so that they can be run in series to go from assemblies to alignment using a single script.

The submission format this week is similar to last week. However, you will be combining both the isPCR and Needleman-Wunsch functionality into a single package named "magnumopus". You will again submit a tarball of your package, data directory, and scripts. The structure of that directory should be something like the following:

```
(biol7200) ~/biol7200/class11/ex11$ tree
.
└── gtusername
    ├── data
    │   ├── Pseudomonas_aeruginosa_PAO1.fna
```

```
│   ├── Pseudomonas_protegens_CHA0.fna
│   └── rpoD.fna
├── magnumopus
│   ├── __init__.py
│   ├── ispcr.py
│   ├── nw.py
│   └── more_module_files_if_you_like.py
├── q1.py
├── q2.py
└── amplicon_align.py
```

1. Unifying the steps of your isPCR package (20 points)

Instructions

Last week's assignment required that you maintained the different steps of isPCR as separate functions. However, it would be strange to want to use an isPCR program in a stepwise manner like that. A more sensible usage would be to provide the software with primers and an assembly and get back predicted amplicons in a single step.

For this question, your task is to unify the steps in your isPCR module into a single function named "ispcr". The function should take as input the path to a primer file, the path to an assembly file, and a maximum amplicon size. i.e., it should have the following definition line:

```
def ispcr(primer_file: str, assembly_file: str, max_amplicon_size: int) -> str:
```

Your package will be tested by running the included "q1.py" script to import your "magnumopus" package and call your function.

Expected output

The expected output from the provided "q1.py" script is as follows:

```
(biol7200) ~/biol7200/class11/ex11$ ./q1.py
>Pseudomonas_aeruginosa_PAO1_NC_002516.2:635141-635853 Pseudomonas aeruginosa PAO1,
complete genome
TCTCGGCGACGGTCAGCTCGACCTCGCTTTCCAGGGCCGCCAGCTTCTGCTGGTTGCGCAGGATGTCGTCGCGCAGGCGCTCG
ATGGCCTCGGCGTACTTCGGCTTGCTCTTCAGGACGCTGTCGACCCACTTCTCGTCGGTCTCGTGGTTCGGGAACAGGCGCAG
GAAGTCGGCACGCGGCATGCGCGCGTCACGCACGCAGAGCTGCATGATGGCGCGTTCCTGGGCGCGCACGCCTTCCAGGGCGG
AGCGCACGCGGGCGACCAGGGCGTCGAACTGCTTGGGCACCAGCTTGATCGGCATGAACAGCTCGGCCAGGCCGGTGAGTTCG
GCGGTGGCCTGCTTGCTGCCGCGACCGTGCTTCTTCAGGGCCTTCTTGGCCTTGTCGAGCTGCTCGGAGACCGCGGTGAAACG
CAGGCGGGCTTCTTCCGGATCCGGACCGCCGTCGCCTTCGTCGTCGCTGTCGCTGCTGTCGTCGCTTTCTTCTTCCTCGTCGT
CCTTCTCTTTCGAGTCGGCGGAATCGTCCTTCAGGTTGACCGGCTCCACCTCTTCGGCGGGCAGGCTGCCGTCATCGGGATCG
ATATAGCCGCTGAGGACGTCGGAGAGGCGACCGCCTTCGGCGACGATGCGATTGTAGTCGGCCAGGATGCTGTCCACCGTGCC
CGGGAACTGGGCGATGGCGCTCATCACTTCGCGGATGCCTTCCTCGATG
```

I have provided you with an example solution to last week's assignment. If you would prefer, you can use that as the foundation for this week's assignment instead of the code you wrote yourself last week.

2. Implementing the Needleman-Wunsch global sequence alignment algorithm (50 points)

Instructions

The Needleman-Wunsch algorithm was described in class. For this question you must write a Python implementation of that algorithm and add it to your "magnumopus" module. Your Needleman-Wunsch implementation should be called using a function named needleman_wunsch() with the following definition line:

```
def needleman_wunsch(seq_a: str, seq_b: str, match: int, mismatch: int, gap: int) -
> tuple[tuple[str, str], int]:
```

Where the return value is a tuple containing a tuple of your aligned sequences, and the score of the alignment. i.e., your return line will look something like

```
return (aligned_1, aligned_2), score
```

Your package will be tested by running the included "q2.py" script to import your "magnumopus" package and call your function. Note that you do not need to write a function that identifies all optimal alignments. You need only write a function that finds one of the optimal alignments.

If you would like to, you can write a function that finds all optimal alignments, but for the purposes of this exercise it should only return one of them. We will discuss a function that finds all alignments when we cover recursion.

Expected output

The expected output from the provided "q2.py" script is something like as follows:

```
(biol7200) ~/biol7200/class11/ex11$ ./q2.py
CTTCTCGT-CGGTCTCGTGGTTCGGGAAC
CTT-TCATCCACT-TCGTTGCCCGGGAAC

True
11
```

If you look at the code of the q2.py script, you will see a range of possible alignments. The alignment produced by your function will be printed and, beneath that alignment, a boolean will be printed indicating whether your alignment is one of the possible alignments I expected to see. If you do not get the exact alignment shown above, but see the word "True" printed, then your output is correct. If you see False, that means your alignment is not one of the expected alignments. Any of the expected alignments will get the points for this question.

3. Putting it together (30 points)

For this question, you must unite the two components of your package (isPCR and Needleman-Wunsch) into a single script, named amplicon_align.py. This script should have the following characteristics:

1. It should import and use your "magnumopus" package to perform isPCR and Needleman-Wunsch alignment. You may also define functions within the script
2. It should accept as command line input two assembly files, a primer file, a maximum amplicon size, and a match, mismatch, and gap score
3. It should have a command line interface that includes named arguments, not simply positional arguments. See the example help message below.
4. It should perform isPCR on both provided assembly files with the provided primer file
5. The amplicons from each assembly file should be aligned. There should only be one amplicon from each file so you don't have to worry about dealing with multiple
6. It should check which orientation the two sequences align best in and only return the best alignment (i.e., reverse complement one sequence to check)
7. It should print the alignment followed by the alignment score to the terminal

An example of the help message printed by the example script I have written is shown below. You should use the same command line options to make it feasible for the TAs to test your code. i.e., the usage of your script must match that in the expected output example below. However, you are free to add long options and better help messages if you wish. They just won't be used or considered when grading your assignment. They'll just be extra flair.

Note that when providing negative numbers as command line inputs, argparse is prone to interpreting them as options. You can get around that in two ways: first, quote the number with a leading space (e.g., `--gap ' -1'`). Second, you can associate an option and its value using and "=" symbol (e.g., `--gap=-1`).

```
(biol7200) ~/biol7200/class11/ex11$ ./amplicon_align.py -h
usage: amplicon_align.py [-h] -1 ASSEMBLY1 -2 ASSEMBLY2 -p PRIMERS -m
MAX_AMPLICON_SIZE --match MATCH --mismatch MISMATCH --gap GAP

Perform in-silico PCR on two assemblies and align the amplicons

options:
  -h, --help            show this help message and exit
  -1 ASSEMBLY1          Path to the first assembly file
  -2 ASSEMBLY2          Path to the second assembly file
  -p PRIMERS            Path to the primer file
  -m MAX_AMPLICON_SIZE  maximum amplicon size for isPCR
  --match MATCH         match score to use in alignment
  --mismatch MISMATCH   mismatch penalty to use in alignment
  --gap GAP             gap penalty to use in alignment
```

Expected output

```
(biol7200) ~/biol7200/class11/ex11$ ./amplicon_align.py -1
data/Pseudomonas_aeruginosa_PAO1.fna -2 data/Pseudomonas_protegens_CHA0.fna -p
data/rpoD.fna -m 2000 --match 1 --mismatch=-1 --gap=-1
TCTCGGCGA-CGGTC-AG-CTCGACCTCG-CTTTCCAGGGCCGCCAGCTTCTGCTGGTTGCGCAGGATGT-CGTCGC-
GCAGGCGCTCGATGGCCTCGGCGTACTT-CGGCTTG-CTCTT---CAGGACGCTGTCGACCCA-CTTCTCGT-
CGGTCTCGTGGTTCGGGAACAGGCGCAGGAAGTCGGCACGCGGCATGCGCGCGTCACGCACGCAGAGCTGCATGATGGCGCGT
TCCTG-
GGCGCGCACGCCTTCCAGGGCGGAGCGCACGCGGGCGACCAGGGCGTCGAACTGCTTGGGCACCAGCTTGATCGGCATGAACA
```

```
GCTCGGCCA-GGCCGGTGAGTTCGGCGGTGGCCTGCTTGCTGC-CGCGACCGTGCTTCTTCAGGGCCTT-CTTG-
GCCTTGTCGAGCTGCTCGGAGACCGCGGTGAAA-CGCAG-GCGG-
GCTTCTTCCGGATCCGGACCGCCGTCGCCTTCGTCGTCG-CTGTCGCT-GCTGTCGTCGC-T-TTCTTCTTC-
CTCGTCGTCCTTCTCTTTCG-AGTCGGCGGAATCGTCCTTCAGGTTGACCGGCTCCACCTCTTCGGCGGGCAGGC----
TGCCGTCAT-
CGGGATCGATATAGCCGCTGAGGACGTCGGAGAGGCGACCGCCTTCGGCGACGATGCGATTGTAGTCGGCCAGGATGCTG-
TCCACCGTGCCCGGGAACTGGGCGATGGCGCTCATCACTTCGCGGATGCCTTCCTCGATG
TCTCGGCGATC-TTCAAGCCT-G-TTTCGAC-TTCAAGAGCGGTCAGCTTCTGCTGGCAACGGATGATGTCCG--
GCTGCAGGCGGGCGATGGCTTCGGCGTACTTGC-TCTTGCCT-TTGGCCAGG--GC-GTCGGTCCAGCTT-TCATCCACT-
TCGTTGCCCGGGAACTGGCGCAGGAAGTCGGCACGTGGCATGCGGGCATCACGCACGCAGAGCTGCATGATGGCGCGTTCCTG
CTG-
GCGCAGGCGATCCAGGGCACTGCGTACACGTTCAACCAGGCCTTCGAATTGCTTCGGAACCAGTTTGATCGGCATGAACAGTT
CAGCCAGGGCCAG-CATTTCGGCGATGGCTTGCTTG-TTCTCGCGGCCGTGTTTCTTCAGCGCCTTGCGGGTGAC-T-
TCCATCTGGTCGGCGACGGC-GCCAAAGCGCTGTGCGGCG-AT-GACCGGATCCGGACCGCTTTCGGCTTCTTCTTCGTCT-
TCGGTCGCT-TC--CGCTTCTTCGTCTTCGCT-GTCGTCATCCGCTTTCGCGGTC-TTGG--T-GT-CGACA-G--G--
CGGCGGCACTTCGGCGGCAGGC-GGCGTAATGCCGTCATCCGGG-
TCGATGTACCCGCTGAGAACGTCGGACAGGCGTCCACCTTCGGTGGTGACGCGAGTGTACTCGGAGAGGATG-
TGATCAACCGTGCCAGGGAAGTGCGCGATTGCGCCCATCACTTCACGGATACCCTCTTCGATA
368
```