Programming for Bioinformatics | BIOL 7200

Week 2 Exercise

August 29, 2023

Instructions for submission

- Run a Linux or Mac terminal on your computer
- You may want to create a directory to work in (e.g., "~/biol7200/week2/ex2")
- Download "exercise2_data.tar.gz" from Canvas and extract the contents into your working directory (e.g., "~/biol7200/week2/ex2").
- Prepare a file containing answers to the below questions. Copy the question and either paste the body of a script, write the correct answers, or provide a screenshot of your working below the question as directed in the instructions provided in the question.
- In cases where you need to paste a screenshot of your terminal, you can do that as follows. Screenshot a selected area with Windows: win+shift+S, Mac cmd+shift+4 and then paste in your submission sheet
- Use the `clear` command to tidy up your terminal so you show only relevant commands and outputs
- Name your submission sheet: "gtusername.pdf", or "gtusername.docx"

Grading Rubric

This assignment will be graded out of 100.

Each question includes the points available in parentheses.

1. **Environments** (total 10 points)

   For the following questions, provide a short written answer. No screenshots are required.

   1. What is an environment in a Unix system? (1)
   2. How does the function or behavior of variables and aliases differ? (state 1 difference) (1)
   3. State the command you would use to create a variable called "wizard" which contains the string "Gandalf the gray". (1)
   4. State the command you would use to print the contents of the "wizard" variable. (1)
   5. State the command you would use to make the variable "wizard" available to any subshells of your current shell. (1)
   6. State the command you would use to create an alias, called "view_wizard", for the command you gave for question 1.4 (a command to print the contents of the "wizard" variable). (1)
   7. State the command you would use to view (not run) the alias created in question 1.6. (1)
   8. Consider a situation in which the value stored in "wizard" may change (e.g., "Radagast the Brown"). Describe the difference in the behavior of the alias created in 1.6 if you create the alias using single quotes or double quotes. Explain the difference. (3)

2. **PATH** (total 10 points)

   For the following questions, provide a short written answer. No screenshots are required.

1. What does your `$PATH` variable determine about the behaviour of your shell? (1)
2. Is your `$PATH` available to subshells or do you need to make it available to subshells (as in question 1.5)? (1)
3. Consider a `$PATH` with the following value "/usr/bin:/usr/local/bin:/bin". If I put an executable named `foo` in each of the directories in that `$PATH`, which directory contains the copy of `foo` which will run if I invoke it as follows ($ here denotes a command prompt)? (2)

```
$ foo
```

4. State the command you could use in question 2.3 to determine the location of the copy of `foo` being used.(1)
5. State a command you could use to add the directory "/home/username/bin" to the *beginning* of your `$PATH`. (1)
6. State a command you could use to add the directory "/home/username/bin" to the *end* of your `$PATH`. (1)
7. Which file would you modify to change your `$PATH` for all future shell sessions? (1)
8. Would changing the `$PATH` using the file you named in question 2.7 impact the `$PATH` variable of other users on the same computer? Why or why not? (2)

3. **`mamba` basic usage** (total 10 points)

For the following questions, provide a short written answer. No screenshots are required. All answers which include commands should use `mamba` commands instead of `conda` where relevant.

1. What is the "base" environment in a `conda` or `mamba` installation? (1)
2. When should you install packages into the "base" environment and when should you install packages into another environment? (1)
3. State the command to create a new environment named "week2ex" into which you install python version 3.11, seqtk, and blast.(1)
4. State the command you would use to activate the environment created in question 3.4. (1)
5. State the command you would use to leave the environment created in question 3.4 if you had activated it. (1)
6. State the command you would use to export the environment created in question 3.4 to a file named "week2ex_env.yaml" in a cross-platform compatible way (i.e., so you could give it to someone running OSX or Ubuntu and they could recreate your environment). (1)
7. State the command you could use to create an environment named "new_week2_env" imported from the "week2ex_env.yaml" described in question 3.6. (1)
8. State the command you could use to delete the "new_week2_env" described in question 3.7. (1)
9. State the command you could use to *install* the package "samtools" from the channel "bioconda" into your current environment. (1)
10. State the command you could use to *uninstall* the package "samtools" from your current environment. (1)

4. **Scripting in Bash** (total 70 points)

For the following questions, you will be asked to write a Bash script that performs the described steps or function. Each answer should consist of three components:

- Copy and paste the contents of your script (ensure that any indentation renders correctly before submitting)
- Paste a screenshot your terminal executing the script as described and additional screenshots displaying requested outputs (Note: we won't ask you to show outputs that span more than one terminal window)
- Write any additional explanation requested

1. "dir_script.sh" (10 points)

   Write a script named "dir_script" that performs the following steps. The usage of the script should be `./dir_script.sh <path>`

   - Makes a directory as well as any missing parent directories
   - enters the new directory (i.e., changes its working directory to be the newly created directory)
   - prints its working directory to stdout

   Run the script in your home directory, providing as input the path "~/abcxyz/demo_week2/dirscript/". In the screenshot, show that you do not have the dir "abcxyz" in your home dir before running the script, but that the dirs "~/abcxyz/demo_week2/dirscript/" do exist after you execute the script.

   After running the script did your shell move its working directory to "~/abcxyz/demo_week2/dirscript/"? Why or why not?

2. "change_headers.sh" (30 points)

   The next two questions will be linked as you will be using the same data to answer them both. Each question will begin with a description of the data you will be manipulating and a explanation of why you are performing the manipulation. These scripts are genuinely useful in a bioinformatic analysis and we shall develop them over the coming weeks using the concepts we cover in class.

   The following section is a description of FASTA format data. If you are already familiar with FASTA format sequence data, all the information required to complete this task is given in the "Your task" section below.

   FASTA sequences

   The data we will be working with is in FASTA format. FASTA format is defined by two types of line: header lines which begin with ">" (i.e., the first character in the line **has to be** ">"), and sequence lines which do not contain ">" characters. Each header line is associated with all sequence lines following it until another header line is reached. For example, in the following FASTA file, "header 1" has 3 lines of sequence, while "header 2" has 2 lines of sequence

```
>header 1
ATCGTCGCAAA
ATCGTGTGGTA
CCGTGTTG
>header 2
```

```
GGTGTGTGGTG
AGTAGTAA
```

Sequence lines can be any length, although a common practice is to restrict lines to 60 characters in length. (Consider what that would mean for searching the file using something like `grep`!)

FASTA is used to store DNA, RNA, and protein sequence. There is no metadata lines inside the file which tell you whether the data are DNA, RNA, or protein. However, some organizations have defined specifications to provide more information about the nature of sequences (see for example https://www.ncbi.nlm.nih.gov/genbank/fastaformat/).

Note that DNA and RNA is often double stranded. FASTA representations of DNA and RNA sequences only represent one of the two strands.

Instead of metadata within the file, file extensions are commonly used to identify the nature of sequences contained in a FASTA file. For example, generic file extensions like ".fasta" and ".fa" can be used for any FASTA sequences, but do not indicate the contents. ".fna" and ".faa" specify that the file contains **n**ucleic **a**cids (DNA/RNA) or **a**mino **a**cids (protein), respectively. As with other files, it is good practice to use informative file extensions to help to organize your data.

A common datatype that is stored in FASTA format is genome assemblies. Genome assemblies are hypotheses about the sequence of an organism's genome that was generated using sequencing data (e.g., Illumina sequencing reads). Sequence assembly software often produces genome assemblies with generic sequence headers. For example, Spades (https://github.com/ablab/spades) produces headers that contain length and coverage information, but nothing that links headers from the same assembly (e.g., ">NODE_1_length_1121141_cov_33.426083").

In many cases, bioinformaticians need to analyze large numbers of FASTA files at once. In those cases it is often useful to modify sequence headers to contain additional information, such as which sample they are associated with. Your task for this question is to write a Bash script that does that.

In this weeks assignment folder are two genome assemblies of the pathogenic bacterium *Pseudomonas aeruginosa*. Their filenames contain unique identifying information (their accession numbers in the European Nucleotide Archive).

Your task

Write a script that adds the accession number (i.e., the filename **without the extension**) to the beginning of each header line in the file and outputs the modified FASTA sequence to a new file. The output file must be a valid FASTA format (i.e., follow the format rules described above).

The usage of your script should be `./change_headers.sh <input file> <output file>`

Hints:

- Don't forget that you can break a complex problem down into simpler steps by using a pipeline.

- sed can use environment variables in expressions when the expression is in *double* quotes.

```
bar="goodbye"
echo "hello there" | sed "s/hello/$bar/"
```

In your submission sheet include the following:

- The content of your script
- A screenshot of a terminal in which you show the following using issued commands:
  - The first 3 headers in one of the assembly files
  - The execution of your script on the file you showed headers from
  - The first 3 headers in the output file showing the expected modifications

3. "find_perfect_matches.sh" (30 points)

For this question, you will analyze the outputs of the script you made in question 4.2.

One application in which you may want to analyze many assemblies at once is when searching for the presence of a sequence of interest. You may wish, for example, to determine which samples contain a gene (or allele) of interest.

The following sections provide some background on the BLAST tool and biological question we are using BLAST to address. If you are already familiar with running BLAST in the command line, all the information required to complete this task is given in the "Your task" section below.

BLAST

The most widely used program for searching a sequence database for matches to a query sequence is BLAST. We don't have time to go over the BLAST algorithm in detail in this course. However, if you would like to learn more about it, the short videos here offer an excellent description: https://www.youtube.com/watch?v=8A-msg23u0w&list=PLpPXw4zFa0uLMHwSZ7DMeLGjIUgo1IBbn&index=30

BLAST is able to rapidly search for approximate sequence matches between a query sequence and a subject sequence or database. You can search online databases hosted by the NCBI (like those you would search using the web application https://blast.ncbi.nlm.nih.gov/Blast.cgi) or local sequences and databases.

The BLAST command line application allows you to control parameters of the algorithm as well as the output format of the results. For our purposes we are going to stick with the tab-delimited format, `-outfmt 6`. This format has twelve default fields as well as a number of optional fields. You can view all possible fields in the help message (note: BLAST does not follow the short and long option convention described in class. All options are single dash. Help is called with, e.g., `blastn -help`). Output fields can be specified as follows: `-outfmt '6 [std] <field> ...'`, where std can be included if you would like the default twelve fields output. For example, `-outfmt '6 sseq'` would output only one field - the matched sequence in the subject, while `-outfmt '6 std sseq'` would output the default twelve fields and a thriteenth field containing the matched sequence in the subject. You can find a full list of possible fields that can be included in outfmt 6 here

BLAST output when run with `-outfmt 6` is tab-delimited, with one sequence match reported on each line. Note that your BLAST results may wrap to multiple lines depending on your terminal size and text size. However, the representation in your terminal does not impact the organization of the output.

Another option to be aware of for this exercise is `-task <task>`. See the online documentation for a description of the various options https://www.ncbi.nlm.nih.gov/books/NBK569839/. We will be searching for matches for a query sequence of 28 nucleotides in length. For short nucleotide sequences, `-task blastn-short` should be used.

As we are comparing nucleotide sequences here, `blastn` is the program we will use. The basic usage of `blastn` for our purposes is:

```
blastn -query <query fasta file> -subject <subject fasta file> -task <task> -
outfmt <desired format> [-out <outfile>]
```

Some things to note about the above usage:

- If no output file is specified, the BLAST results will be output to stdout.
- Only one query file and one subject file can be provided. If you want to blast multiple queries against multiple subjects, you must combine the corresponding files. All queries and subjects must be valid FASTA format.

CRISPR

To illustrate a case in which BLAST can be used, I have provided you with two assemblies ("ERR430992.fna", "ERR431227.fna") as well as another FASTA file ("CRISPR_1f.fna"). The "CRISPR_1f.fna" file contains a single, short sequence. This sequence is associated with a bacterial immune system called CRISPR.

CRISPR (Clustered Regularly Interspersed Short Palendromic Repeats) systems are present in about 40% of bacterial species. They are defined by loci containing repeat sequences that are evenly spaced, with intervening sequences (called spacers) that are typically homologous to viruses and plasmid sequences. The function of CRISPR systems as an immune system is acheived through sequence-specific targeting of CRISPR-associated (Cas) proteins to destroy invading viruses etc. Cas proteins typically achieve this destruction of invading genetic elements through DNA/RNA cutting activity. Spacers are the immune memory that provides the sequence specific targeting that guides the Cas proteins to destroy only specific non-self sequences.

In recent years, the sequence-specific cutting activity of the Cas proteins associated with CRISPR systems has been developed into genetic tools that allow the genetic editing of almost any organism! You may have already heard of CRISPR systems because of these advances. For the exercise at hand, we are going to focus on the bacterial immunity aspect of CRISPR systems, rather than the gene editing.

The most common type of CRISPR system encoded by *P. aeruginosa* is type I-F. If a strain of *P. aeruginosa* has CRISPR repeats and *cas* genes, then it has a (likely) active CRISPR immune system. The absence of either repeats or *cas* genes indicates no active CRISPR system. Your task here is to perform an analysis to determine which (if any) of the provided assemblies encodes type I-F CRISPR repeats.

Your task

Write a script that runs a BLAST of a query file against a subject file, writes **only perfect matches** to an output file, and prints the number of perfect matches to stdout.

The usage of your script should be `./find_perfect_matches.sh <query file> <subject file> <output file>`

Hint: Think about what information you need to identify a perfect sequence match. In order for two sequences to be considered identical, they must have 100% sequence identity and equal length. Which output fields do you need to add (if any) to the BLAST output to determine which matches meet those criteria?

In your submission sheet include the following:

- The content of your script
- A screenshot of a terminal in which you show the use of your script to determine which (if any) of the assembly files included with this week's assignment ("ERR430992.fna", "ERR431227.fna") contain perfect matches of the type I-F CRISPR repeat contained in the file "CRISPR_1f.fna".