BIO 792 Data Science II

Homework 1

Part 1

**Part I: Questions**

1. What is a Graphical User Interface (GUI) and how does that differ from the Command Line Interface (CLI)?  
    A Graphical User Interface (GUI) is a way a human use to interact with a personal computer by clicking a mouse and using a menu-driven interaction. It differs from the Command Line Interface (CLI) in a way that CLI works by writing command lines to let the computer do the task, allowing repetitive tasks to be done easily and fast (relative to GUI).
2. What does the shell do?  
   The shell is a program where users can write command lines that run other programs. Complicated tasks can be done on the shell.
3. What is the command used for listing things in a directory?  
   ls
4. What is the command used for changing directories?  
   cd
5. What command would you use to get your current working directory?   
   pwd
6. How do you get the manual for these commands?  
   man
7. What does the shell prompt look like?  
   your current directory ~%
8. How would you list things in a directory in chronological order?  
   ls -t
9. Name two ways to get to the ‘home’ or ‘root’ directory.  
   cd  
   cd ~
10. What is the difference between an absolute path and a relative path?  
    An absolute path is a complete directory that always contain a root element to a location of a file while a relative path is a path you take from your current directory to your new directory relative to where you are. For example, you can us .. to move up a directory without typing the complete directory.

#relative paths are relative to where you are . means my current directory

1. What are the two relative path directories we talked about and what do they mean?  
   A dot (.) after cd takes you to a current directory while double dots (..) take you to up a directory (a directory before the current directory).
2. Why shouldn’t you put spaces in filenames?  
   Unix doesn’t like spaces in filenames, and you shouldn’t put spaces in file name because the software (unix, linux, terminal) can incorrectly interpret the filenames with spaces. The white spaces are commonly used to separate commands.
3. Name a way to have multiple readable words in a filename without spaces?  
   You can add a dash (-), an underscore ( \_ ) or a dot (.) to separate words in a filename. You can us a capital letter when you start a word in the filename.
4. What is the program nano, what does it do? Do you use nano? If not what do you use?  
   Nano is a text editor where you can write a script for things you want to run in your terminal (including a bash fiile). I don’t use nano. Instead, I use BBEdit when writing a script.
5. \* is a form of a wildcard. What does it mean?  
   It means a \* in a wildcard matches any beginning or ending with things that are included with the \*. For example, you use “\*.txt” to match and or select all text files that end with “.txt” You use “Homework\*) to match or select files that start with a word “Homework.”
6. The following code would match what? ls \*.txt  
   This code would match text files showing a list of text files that end with .txt.
7. Why do we need to be careful with the mv command?  
   The mv command is used to move a file or folder. It can be used to copy or replace the file. We need to be careful with the mv command because the original file or folder will be affected. We don’t want to accidentally move the file to a wrong place or rename it to something else.
8. Why do we need to be careful with the rm command?  
   The rm commands is used to remove files. We need to be very careful with this command because we cannot recover the files after they are removed.
9. What is the difference between > and >>?  
   Both > and >> tell the shell to redirect the command’s output to a file instead of printing it to the screen. However, they work slightly different. The output file gets overwritten each time when we run the command and use the >operator. The >> operator also writes the output to a file but appends the string to the existing file for the second time the command is run.

#excellent!

1. What does head do?  
   The head command prints lines from the start of a file.
2. What is the purpose of | (pipe)?  
   The purpose of the | (pipe) is to tell the shell that we want to use the output of the command on the left as the input to the command on the right.
3. Interpret the following command:

cat huge\_file.fasta | uniq | head -n 5 >proteins.fasta  
We first show and print contents of “huge\_file.fasta” file. We then filter out replicated lines of the output from the cat command. Next, we us the head command to get the first 5 lines of the unique lines from the previous command (uniq). Finally, the final output (the first 5 lines) is stored in an output file called “proteins.fasta.”

1. What is a loop and when would you use it?  
   A loop is a programming construct that allow us to repeat a command or set of commands for each item in a list. I will use the loop command when I want to repeatedly run the same commands to multiple items. It helps reduce the amount of typing required and typing mistakes
2. In a directory with the following files. What would the following loop do? apple.txt, banana.txt, orange.txt $ for filename in \*.txt; do > cat ${filename} >>fruits.txt > done

#?

1. What is a shell script?  
   It is a small program where you can type and save your code. The shell script (that has the code) allows us to run programs.
2. What are the benefits of writing and running a script over typing the code in?  
   You can keep the script for future uses in case you need to rerun the code written on a script again. You can also run the entire script using a single command on shell. Lastly, it is easy to track errors on a script compared with just typing the code in the shell.
3. When looking at a script what does # mean? Why would you use one?  
   The # on a script indicates a comment. Anything written after # (that is on the same line) is not included in a code you write. You use the # (comment) to document your code. Documenting the code is important because it helps you understand a meaning of each code and command when you look at your code a month later.
4. What does grep stand for and what does it do?  
   The grep command stands for “global regular expression print.” It is used to search texts or any given input files.
5. What does find do?  
   The find command is used to find files and directories and perform subsequent operations on the searched files or directories.

## Part II - Writing Code

As part of this exercise please open the shell and practice moving around into different files. Best practice is to try a little bit every day. Try to challenge yourself by not using the mouse. For each of the questions below type your code and the result from the shell prompt.

1. **Open the shell and change directories to the desktop and list the contents, paste your code here.**  
   cd ~/Desktop. #This line changes a directory from my home directory to my desktop  
   ls #This command shows a list of contents on the desktop directory
2. **Draw the file structure from the root directory into one of the folders on the desktop. You can submit a pdf with a drawing. Select three locations and type the commands for changing directories to those locations use a combination of absolute and relative paths.**

Diagram

Description automatically generated

Change from the current directory to location 1:  
 % cd .. (to move up a directory)

Change from the current directory to location 2:  
 % cd (move to home directory)  
 Then  
 % cd ~/Downloads

Change from the current directory to location 3:  
 % cd ..  
 Then  
 % cd Planets

1. **On your computer create a directory on your desktop. Create two files without opening them that have file extensions .txt. Then use wildcards to list the files in that directory. Paste you code here.**  
     
   mkdir homework1 #making a new directory called “homework1” on my desktop  
   cd homework1 #changing a directory from the desktop to “homework1”  
   touch file1.txt #creating a new .txt file in the current “homework1” directory  
   touch file2.txt #creating another .txt file  
   ls \*.txt #showing a list of files in the current directory using a wildcard \*
2. **Download the file Hutia\_DNA.fasta from the Github repo. Make a new directory on the desktop. Move the file into that directory. List the contents of the directory. How many lines are in this file? Fasta files are a type of data file that holds DNA sequences. They are all formatted the same. The first line starts with a > and a name followed by a line break, and then DNA sequence followed by a line break. The next line starts with an > and another name followed by a line break and the DNA sequence. With that in mind, how could you use linux commands to determine how many sequences are in this file? How many sequences are in this file?**

% cd ~/BIOL792\_SP

% cd week1 #This is a directory I previously stored the file Hutia\_DNA.fasta

% ls #Showing a list of files in this current directory

Hutia\_DNA.fasta data-shell data-shell.zip

% mv Hutia\_DNA.fasta ~/Desktop/homework1.1 #Moving the file to a new directory

% cd

% cd ~/Desktop/homework1.1 #Changing the directory to the new directory on Desktop

% ls #Listing the contents of the directory

Hutia\_DNA.fasta  
  
% wc -l Hutia\_DNA.fasta #Counting the number of lines in the Hutai\_DNA.fasta file  
 2382004 Hutia\_DNA.fasta #There are 2382004 lines.  
  
% grep -o '>' Hutia\_DNA.fasta | wc -l   
#Using a grep command to search for lines that start with “>.” The “>” is followed by a #name of the sequence.   
  
1191002 #There are 1191002 sequences in the file.

#excellent!

1. **Build a single line of code that would take the first 100 sequences of this file and put them in a new file called ‘MyOutputFile.fasta’. Use a combination of wc, sort and uniq in a single line of code to tell me how many unique lines are in this file. Paste your commands and the answer here.**

% head -n 200 Hutia\_DNA.fasta > MyOutputFile.fasta   
#Taking the first 100 sequences and storing them in a new file. Each sequence has two lines #(name and the actual sequence), so I take the first 200 lines of the Hutai\_DNA.fasta  
   
% wc -l MyOutputFile.fasta #Checking the output file  
200 MyOutputFile.fasta #The output file has 200 lines  
  
% sort -n MyOutputFile.fasta | head -n 100 | uniq| wc -l   
#Here, I first sort the file MyOutputFile.fasta by numeric, then select the first 100 lines (they #are sequence neames). Once I have list of the sorted lines, I use “uniq” command to keep #the unique lines and use “wc-l” to count the number of unique lines   
100 #There are 100 unique sequence names.

#Alternatively, I can check the unique sequences

% sort -n MyOutputFile.fasta | tail -n 100 | uniq| wc -l.   
#Here, I select the actual sequences (i.e., the last 100 lines in the file), keep the unique lines #and count them.

99 #There are 99 unique sequences in the file.

#well done!!

1. **Write code that would create a loop to copy all the files in one directory ending in .txt to another directory. Paste the code here.**  
     
   $ for filename in \*.txt   
   > do  
   > mv $filename ~/Desktop/homework1.1   
   #This command line moves each “filename” to a new directory called “homework1.1”  
   > done
2. **Write a bash script with in-line documentation (hint #) to show how to run a made up python program (script) called 'find\_taxa.py' on set of files ending in ‘.fasta’. Show in the script how to run the program on each of those files. What is the name of your script? Type the script here.**

#!/usr/bin/sh

#Bash scripts always start with #!/usr/bin/sh

#To check if this bask script works in the shell, I print a sentence to show if it works properly.

echo "Sample bash script for homework 1"

#The following lines demonstrate how to run a made up python script "find\_taxa.py" in shell

#on set of files ending in ".fasta"

#option 1

python3 find\_taxa.py \*.fasta

#This line first opens Python, calls a python script, and tells the program what the input files are.

#This option will run the python script on all of the .fasta files in the current directory

#option 2

python3 find\_taxa.py filename1.fasta filename2.fasta filename3.fasta

#This option will allow a user to specify what input files to be included

#The following for loop is what I think it is how to run the program on the set of files

INfile = \*.fasta #The input can be all .fasta files or a selected set of .fasta filesin the current directory

OUTfile = output #The output can be anything

for filename in INfiles

do

python3 find\_taxa.py $filename #this is where the pyton scrip will work

done

#To open this bash script in terminal, I would type

# "bash script.sh"

The name of my script is “script.sh”

#excellent work!

1. **From the Hutia\_DNA.fasta file tell me how on how many lines do we find the pattern ‘GAGA’. What was the code used to find this?**% grep -n "GAGA" Hutia\_DNA.fasta | wc -l #Here, I use a “grep -n” command to search for a pattern “GAGA” and return the line #numbers of the matches. Using a pipe (|) and a “wc-l” command to count the number of #lines we find the pattern “GAGA.”  
     
   304806 #The “GAGA” pattern is found in 304806 lines.