Graded assigment # 3. Anna Kolganova

Part A

To solve this part (steps 1-4), I followed the instructions and used the following commands to initialize my repo :

```
git init
git add README.md
git commit -m "Committing to README for the assignment"
```

Part B

5. Save the script below as scripts/echo.sh. Look at the below three commands to run the script, and describe what you expect to be the output and why. Then, test the commands and if any output was not as expected, reconcile the differences.

I used the following commands:

```
touch scripts/echo.sh
echo '#!/bin/bash
set -euo pipefail

echo "$1"
echo "$2"
echo "$3"
echo "Finished"' > echo.sh
```

For the offered commands, I expect that the first one will print the first two dates (Oct 7 and Oct 8). The second command will print Oct 7-9 dates but will replaced Oct 10 with "finished" in the 4th last line as it is applied by the code. The last command will have Oct 7 and 8 on separate lines and Oct 9 and 10 in 1 separate line because they are in "" (indicated as a string). This is what I see as my output,too.

When I used the commands listed in the assigment, my terminal responded that there's no such file directory. My pwd says /fs/ess/PAS2880/users/kolganovaanna/GA3/scripts and ls -l command confirms the existence of echo.sh. file. I think I probably don't need the scripts part in the code because I'm already in scripts. Without it, the code works and I hope it's not a big deal:

```
bash echo.sh Oct07 Oct08
bash echo.sh Oct07 Oct08 Oct09 Oct10
bash echo.sh Oct07 Oct08 "Oct09 Oct10"
```

The output was:

```
Oct07
Oct08
echo.sh: line 5: $3: unbound variable
```

```
Oct07
Oct08
Oct09
Finished
Oct07
Oct08
Oct09 Oct10
Finished
```

6. Save the script below as scripts/concat.sh. Run it to concatenate the two FASTQ files you copied to data/ earlier. (Input files can remain compressed and therefore, so will the output file be.)

I used the following commands:

```
touch concat.sh

echo '#!/bin/bash
set -euo pipefail

cat "$1" "$2" > "$3"
ls -lh "$1" "$2" "$3"' > concat.sh
```

I then ran the code.

At the end I also committed to the README file again, as it was asked:

```
git add README.md
git commit -m "Committing to README for part B"
```

Part C

7. Write a shell script scripts/printline.sh that accepts two arguments, a file path and a line number, in order to print (not store in a file) the requested line from the specified file. Don't make the script print anything other than the requested line from the file, and make sure to include the discussed best-practice header lines.

In my command, I intend to just use the last line of the file to be printed. For this, I used the following command:

```
#!/bin/bash
tail -n 1 printline.sh | head -1
```

Note: thank you for specifying the commands you want us to use. I want to share this link that has a nice (in my opinion) explanation on how to do the same but using 'sed': https://www.geeksforgeeks.org/linux-unix/write-bash-script-print-particular-line-file/.
I also want to refer to the sources that helped me with the head and line combinations : https://stackoverflow.com/questions/64376028/displaying-selected-

lines-using-head-and-tail-command and https://www.youtube.com/watch?v=4BwcFWhqlA8.

8. Test your script twice by making it print two different lines from garrigos-data/meta/metadata.tsv. In one of these tests, redirect the script's output to an appropriately named file in results.

I had to first move to /fs/ess/PAS2880/users/kolganovaanna using cd to access garrigos-data. I then used the following commands to print 2 lines (the very last in the file and the second line because it's techinally the first line of the dataset). I used +2 instead of just 2 because otherwise it would have printed the 2nd to the last line of the file and that's not what I chose to see:

```
#!/bin/bash

tail -n 1 garrigos-data/meta/metadata.tsv | head -1
tail -n+2 garrigos-data/meta/metadata.tsv | head -1
```

The outputs were the last line of the file and the second line of the file (1st line of the dataset):

```
ERR10802868     24hpi   relictum
ERR10802882     10dpi   cathemerium
```

To redirect the output of the last command (2nd line of the file), I used the followwing commands:

```
tail -n+2 garrigos-data/meta/metadata.tsv | head -1 > GA3/results/secondline.txt
```

9. Run a final test, but now: First, store the file name (just metadata.tsv) in one variable and the line number you chose in another. Then, use these two variables as the arguments that you give to the script. Finally, redirect the script's output to a file in results whose name includes both the file name and the line number. But instead of simply typing the literal output file name, use the two variables you created again, now to "build" the output file name programmatically. The output file name should not include any spaces.

Here, I will just use line 2. I used the following commands:

```
#!/bin/bash
file=metadata.tsv
line=2

For check:
echo "$file"
echo "$line"

tail -n+"$line" "garrigos-data/meta/$file" | head -1

tail -n+"$line" "garrigos-data/meta/$file" | head -1 > "GA3/results/${file%.tsv}line${line}
```

The outputs were:

```
metadata.tsv
2

ERR10802882     10dpi   cathemerium

And the file metadataline2.txt was created under results
```

Note: I struggled a bit with the right syntax for the last command. I didn't see us using "%" in class (maybe missed it), so here I would like to note that I used it so the file name is not metadata.tsv but just metadata. Actually, when I ran it without "%", the terminal said "bad substitution". I want to share the geeks webpage that talks about using "%" (which I think is sometimes called a suffix as well as an extension like in this webpage): https://www.geeksforgeeks.org/linux-unix/bash-scripting-file-extension/

At the end I also committed to the README file again, as it was asked:

```
git add README.md
git commit -m "Committing to README for part C"
```

Part D

10. Go to https://seqera.io/containers and find a container image for the program. Back in VS Code, test-run the container with the command trim_galore -v.

I used the following commads:

```
apptainer exec oras://community.wave.seqera.io/library/trim-galore:0.6.10--bc38c9238980c80e
  trim_galore -v
```

The output was:

```
INFO:    Using cached SIF image
INFO:    gocryptfs not found, will not be able to use gocryptfs

Quality-/Adapter-/RRBS-/Speciality-Trimming
[powered by Cutadapt]
version 0.6.10

Last update: 02 02 2023
```

11. It turns out that your collaborator has been using an older version of Trim-Galore to trim other FASTQ files in the same project, so you decide that it will be best if you use that same version as well. Find a container for Trim-Galore version 0.5.0 and test-run it with the same command as above. Are the versions printed in both cases as expected?

I used the following commands:

```
apptainer exec oras://community.wave.seqera.io/library/trim-galore:0.5.0--16bd677ee493f6cd \
  trim_galore -v
```

The output was:

```
INFO:    Downloading oras image
409.7MiB / 409.7MiB  100 % 60.6 MiB/s 0s
INFO:    gocryptfs not found, will not be able to use gocryptfs

Quality-/Adapter-/RRBS-/Hard-Trimming
(powered by Cutadapt)
version 0.5.0

Last update: 28 06 2018
```

The outputs differ due to the different version but it's expected.

At the end I also committed to the README file again, like it was asked:

```
git add README.md
git commit -m "Committing to README for part D"
```

Part E

12. See if Pandoc is available at OSC prior to loading anything, and if so, which version, by running pandoc -v. Then, search the internet to check if that Pandoc version is the most recent one.

I used the following command:

```
pandoc -v
```

The output was:

```
pandoc 2.14.0.3
Compiled with pandoc-types 1.22.1, texmath 0.12.3.3, skylighting 0.10.5.2,
citeproc 0.4.0.1, ipynb 0.1.0.1
User data directory: /users/PAS2880/kolganovaanna/.local/share/pandoc
Copyright (C) 2006-2021 John MacFarlane. Web:  https://pandoc.org
This is free software; see the source for copying conditions. There is no
warranty, not even for merchantability or fitness for a particular purpose.
```

We can see that this is version 2.14.0.3. This is not the latest version according to this Pandoc website: https://pandoc.org/releases.html. The latest version is 3.8.1 and it came out just a couple days ago, apparently.

13. Check what other versions of Pandoc are available in OSC Lmod modules, and load the module with the most recent available Pandoc version.

I used the following commands:

```
module avail pandoc
module load pandoc/3.6.4

For check:
module list
```

The outputs were:

```
pandoc/2.19.2    pandoc/3.6.4

Currently Loaded Modules:
  1) intel-oneapi-mkl/2023.2.0   3) mvapich/3.0      5) project/ondemand        7) pandoc/3.
  2) intel/2021.10.0             4) modules/sp2025   6) app_code_server/4.8.3
```

14. Use the below Pandoc command to create a PDF of your README.md, and check whether the PDF file is there.

I used the following command:

```
pandoc -o README.pdf README.md
```

The output was quite large.

The pdf file was created and I can see it.

15. To view PDF files in VS Code, you'll first need to install a VS Code extension. Install the extension "Papyrus PDF Preview", and take a look at your PDF. Then, download the PDF file to your computer and also take a look at it there.

I didn't have any issues with viewing or downloading the file. I think it's a very cool tool and now it seems like we know where our graded feedback in a PDF file comes from :)

16. Does everything in the rendered PDF look OK? Or did you make formatting errors, or did anything else not render as expected? If so, update your Markdown file, rerender the PDF, and check again.

I would say it's mostly fine but there are places where formatting doesn't look like I want it to, so I basically went through the same steps to render the file again after I adjusted my Markdown file.

At the end I also committed to the README file again, as it was asked:

```
git add README.md
git commit -m "Committing to README for part E"
```

Part F

17. Create a repository on GitHub, connect it to your local repo, and push your local repo to GitHub.

I used the following commands:

```
git add README.md
git commit -m "commit"
git branch -M main
git push -u origin main
```

18. Create a new issue and tag GitHub user menukabh, asking Menuka to take a look at your assignment.

Done in GitHub

At the end I also committed to the README file again, as it was asked:

```
git add README.md
git commit -m "Committing to README for part F"
```