Digital Methods: Learning Journal

Mette Emily Kendon

Autumn 2020

1 16/11/2019

1.1 Thoughts/intentions:

Before the hands-on-session I had made sure to download the GitBash properly and to read the Shell tutorial, which showed out to be a good idea because how to use shell and Git did not come naturally. At the hands-on-session I had some difficulties understanding how to use shell and Git, and therefore I did not think I would find this assignment easy. If I had to explain why I find shell and Git challenging it is presumably because of the format. When you are used to using file explorer you are also used to working with a more visual format compared to working with the shell where you type in commands. Because of this, my intention with the assignment was to fully understand the logic behind shell and Git and how you, by using these, can move around with ease and interact directly with the computer without using file explorer.

1.2 Exercises:

- The first thing I did was to find the folder which we had to use for this assignment by going to Sciencedata.dk.
- Here, I typed in the password: 2020CDS in order to access and download the folder, which is a folder that contains 189 images.
- The prime goal for this assignment was to help with a few diagnostics by using Git and shell.

Exercise no. 1: Identify the names and format of the 3 biggest files. Can you come up with a command to generate a numerically ordered list of 3 biggest files? (hint: consider using wc to gauge image size)

- First step was to open the Terminal window on my computer. On Mac it is called Terminal.
- In the window I typed: cd desktop/HW to go to the downloaded folder which was named HW. cd means change directory and is the command you use to change location

followed by a directory name to change the working directory. Desktop tells the computer that I want to go to the computers desktop where I want to find the folder named HW.

- Then I typed: ls which is short for list in order to get a list of everything the folder contains. This command showed that the folder contains a lot of files.
- For inspiration to how I could solve this exercise I went to the Unix Shell tutorial under the heading Pipes and Filters. Here I found a lot of different commands which I tried to combine with each other. I played around with different combinations for a while until I came up with a combination of 3 commands, which I think solves this exercise.
- I typed: wc -l * | sort -nr | head -n 4
    - In order to identify the names and format of the 3 biggest files in the folder I typed one long command, which combines 3 smaller commands separated by the vertical bar | called pipe. The pipe tells the shell that I want to use the output of the command on the left as the input to the command on the right.
    - wc is short for word count and counts the number of lines, words and characters in the files. -l makes sure that the output only shows the number of lines per file. * matches zero or more characters and makes the shell turn the content into a list of all the files in the current directory.
    - Sort indicates that I want to sort the list whilst -nr indicates that the list should be sorted numerically from the highest to lowest numbers.
    - Head -n 4 tells the computer that I want to sort the files numeric with the 4 biggest files first. Even though the exercise only asked for 3 files I had to type 4 because the first line showed the total of all the kb-sizes in the file.
- The names of the 3 biggest files and their format RW2:
    - 87391 9247_Overview_SW.RW2
    - 86819 9243_Detail_Trench.RW2
    - 78622 9236_Overview_W.RW2.

Some of the image files are empty, a sign of error in the data processing or corruption. Can you find the empty photo files (0 kb size), count them, and generate a list of their filenames to make their later replacement easier?

- First step was to go to ‘The Unix Shell’ -tutorial for inspiration. At the hands-on-session Adela had mentioned that it would be a good idea to look under the heading ‘Finding things’, and I therefor chose this as my starting point.
- Here I learned that Unix programmers operate with a word named ‘grep’ when they want to find something, because grep finds and prints lines in files that match a certain pattern.
- To use grep properly, I had to create a text-file, which I could run my search in.
- To do this I typed: cd desktop/HW in order to change the directory to the HW folder.
- Then I typed: wc -l *RW2 *JPG > lengths.txt in order to create a new file containing all of the names of the different files in the HW folder. As mentioned earlier wc is short for word count and counts the number of lines, words and characters in the files. -l makes sure that the output only shows the number of lines per file. RW2 and JPG are the two different file formats of the images in the folder. The greater than symbol >, tells the shell to redirect the command’s output to a file instead of printing it to the screen.
- Now the shell had created a text-file named lengths.txt, and to confirm that it existed I typed: ls lengths.txt.
- To send the content of the new file onto the screen I typed: cat lengths.txt.
- Then I typed: sort -n lengths.txt > sorted-lengths.txt in order to move the content from the file named lengths.txt into a new file named sorted-lengths.txt, where the image-names would be sorted numerically. Sort -n tells the shell that the sort should be made numerically from the lowest to highest numbers, whilst the greater than symbol > worked as mentioned above.
- To confirm that the new file existed I typed: ls sorted-lengths.txt
- To send the content of the new file onto the screen I typed: cat sorted-lengths.txt.
- I had now created a text-file named sorted-lengths.txt, which contained a sorted list of all of the filenames in the HW folder. I was now ready to use grep.
- I ran my grep search for all of the empty photo-files (0 kb-size): grep -w 0 sorted-lengths.txt > zero-lengths.txt
- Here ‘0’ is the pattern that I was searching for via the grep command. The grep command searched through the file sorted-lengths.txt, looking for matches to the pattern

specified. I used -w in order to restrict my matches to lines only containing ‘0’ on its own. The greater than >, worked as mentioned above.

- **I had now found all of the empty photo-files (0 kb-size), and created a new file named zero-lengths.txt containing a list of all the empty photo-files (0 kb-size).**
- To confirm that the new file existed I typed: ls zero-lengths.txt
- To send the content of the new file onto the screen I typed: cat zero-lengths.txt.
- When searching for specific information in your files you can also use a command called ‘find’. The find command finds files themselves, and I wanted to use it in order to count the number of empty photo-files (0 kb-size).
- I typed: wc -l $(find . -name “zero-lengths.txt”)
- When the shell executes this command it starts off by running whatever is inside $( ). It then replaces the $( ) expression with that command’s output. wc -l worked as mentioned above. I made sure to run the command in the zero-lengths.txt, which contained all the empty photo-files (0 kb-size).
- **This command told me that there was 73 lines in the text-file zero-lengths.txt. From this I could conclude that there was 72 empty photo-files (0 kb size) in total, because one of the lines in the file zero-lengths.txt showed the total of all the kb-sizes in the file.**

1.3 Final Thoughts

After completing the assignment, I felt relieved that I had solved the two exercises. At first glance I was a bit skeptical, but it turned out to be okay, and I think that I in the process of answering the two questions got a much better understanding of how the shell and Git can be used. I did not get time to look at the last exercise because I it took me a lot of time to figure out how to solve the second exercise. Despite of this, I am very proud of the fact that I solved the two first exercises. Overall, I see how using the shell is a lot more efficient compared to using file explorer, because it is faster and allows the workflow to be documented and specified in order for others to replicate. When you have understood how to use the different commands the shell allows you to do almost everything (including deleting everything on the computer)!