Ty Laughlin

Dr. Hansen

BIO-247

2 October 2022

Part 1:

For the first part of the lab, I partnered up with Emily to work on using the UNIX command line shell. I have a PC and Emily had a MAC. Because Emily has a MAC, she already has a UNIX command line shell built into her laptop. For this reason, she did not need to download one like a did, but she did help me with downloading mine.

To download UNIX, I first clicked on the following link: [The Unix Command Line on Your Own Computer | CS Codex (iwu.edu)](https://sun.iwu.edu/~mliffito/cs_codex/posts/unix-on-your-own-computer/). I scrolled down to the instructions to download a UNIX command line shell on Windows. Within the instructions there was another link to Cygwin. After clicking that link, it allowed me to go on the Cygwin website. On the website, there was a place to click to download Cygwin DLL 3.3.6. This took some time to download, but once it did, I right clicked on my Windows start button and then clicked on the Admin Windows terminal. Following the instructions on Cygwin, I typed in “wsl --install”. This began downloading the actual UNIX shell onto my laptop. This again took some time.

Once the installation of my UNIX shell was complete, my partner Emily and I began to test that our commands did the same things between both of our computers. We did two different commands. We started with “ls” and for both of us, it listed the broad folders on our computers. We then decided to try and change our working directories to our BIO247 folders. To do this I had to type “cd Desktop”, and then I had to type “cd BIO247”. We both typed “cd”, and this showed us that we both successfully navigated to our BIO247 folders.

After checking in with Dr. Hansen, we found out that I was not actually using the UNIX command line shell, despite the instructions on Cygwin leading me to believe that. I only downloaded UNIX, I was not actually using it. To fix this problem, I had to download another terminal to actually use UNIX. I had to download an Ubuntu terminal, then restart my laptop. After it restarted, I opened the Ubuntu terminal and had to create a UNIX username and password. Then I had to type out a string of cd commands to get back to my BIO247 folder. These commands were “cd ../..”, “cd mnt”, “ls”, “cd c”, “ls -lrt”, “cd Users”, “cd TyCam”, “cd Desktop”, then “cd BIO247”. Emily was able to navigate to her BIO247 folder as well, so we were good to go.

Loop: For each row

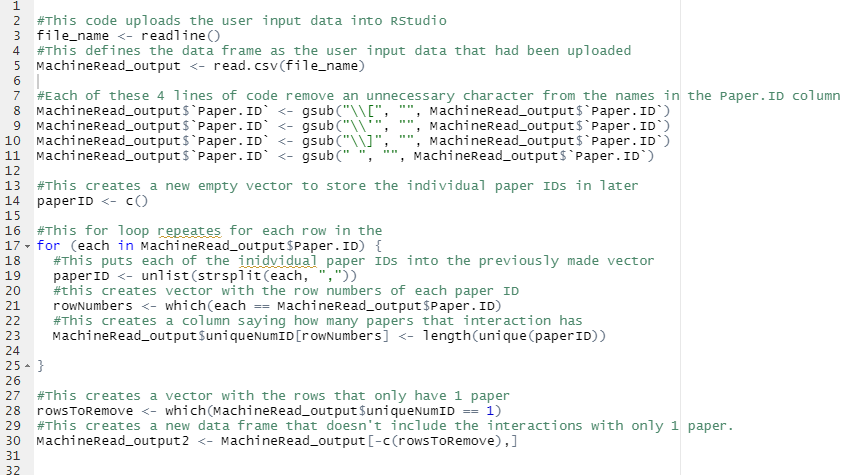
Part 2:

Part 3:

Graphical user interface, text, application

Description automatically generated

Opening the spreadsheet and removing the unnecessary characters was very easy, as I have used these commands before, all I needed to do was change the names of the file and the dataframe. At first, I struggled with the for loop, but once I reviewed the syntax in my notes, I was able to figure it out. I also struggled with the which() function. I typed it into the help section of RStudio and still couldn’t figure out exactly how to use the arguments, so I talked to Rowan about it. Rowan was able to understand it and explained to me how to use it. I struggled with writing the code for the last two parts of my coding diagram. Again, I was struggling with how to properly use the which() function in order to create the vector of which rows only had 1 paper. I again talked to Rowan, and he was able to help me use the right argument. After I had all of my code down, I tried to run it. I ran into some errors because I did not use the right names for all of the arguments. One example of this was in my line of code saying, “MachineRead\_output$uniqueNumID <- length(unique(paperID))”. After some time trouble shooting with my lab group, we figured out that we needed to add “[rowNumbers]” after “MachineRead\_output$uniqueNumID”. This solved the problem, and the code ran smoothly. We ended up being left with 83 papers.

 Part 4:

Creating the code for this was largely very easy because I was able to copy and paste the majority of the code from part 3. The one part that was difficult was using user input in place of reading a spreadsheet off of my computer. In order to do this, we looked up how to use user input data in R. We found that we need to use the function, readline(). I typed in this function and placed its output into a vector so that the data could be used later. Then I created a new data frame by uploading the data from the readline() function into RStudio. As I said, this part was very easy as most of the code was able to be reused from part 3. The only new part was easily solved by using stack overflow to find the solution to the problem of using user input in R.