Micah Busboom

CSCE-155N-210

Final Project Report

5/3/2021

**Project approved by Ms. Quinn Lanik.**

For almost all of my life, I have been involved in athletics or fitness to some degree. I played baseball up until my sophomore year of college. When I stopped playing baseball, I always tried finding something active and competitive to do with my free time to at least some degree. That is to say that fitness has always been a hobby of mine. Until recently, say within the last 8 or so years, maintaining a lean body composition was easy. In recent years it has grown more difficult. I think most everyone can relate to that on some level. I needed to find a way to lose weight. I dug into all the science behind fitness and body re-composition and ended somewhat knee deep in bodybuilding. No, I have never competed or really look like I could, but I learned a lot of valuable lessons and came away with some great tools to become a little bit more athletic. I also have somewhat of a passion for math and engineering, so I wanted to integrate the two approaches of body re-composition and math. I built a spreadsheet to track my daily weigh ins, estimated calorie expenditures, and estimated calorie intake on a daily basis using an excel spreadsheet and came away with a relatively powerful way to visualize my weight loss data and calculate just how many calories my body needed on average to attain a certain level of weight loss. The power behind knowing those figures is that being able to estimate how many calories your body burns on a daily basis is very individual and, frankly, quite difficult. Being able to visualize the data made the process so much more motivating and consistent. However, the spreadsheet was messy and I wanted to use the opportunity of my final project in MATLAB to clean it up a little bit.

The process of cleaning up the data was not very easy. I would estimate that, in total, I spent about six hours researching code techniques and writing out the code. Some of those struggles presented themselves from different domains. The first being the overall planning of the layout of the GUI given how many moving parts there are. In total, there were 18 GUI elements and a plot graph that I needed to combine in a logical and clean way. Drawing it out with pen and paper helped to design the general layout, but converting that into the 18 lines of code was time consuming not only in the sheer volume of characters but in the test and retest precision necessary to line everything up. I had a relatively low working knowledge of how to build GUI elements, so that alone took a large chunk of time. Where I really ran into issues was in setting up the callback functions.

Building the actual functions themselves was not too difficult. Because I had already done a lot of the work in a preliminary excel spreadsheet, I already knew the logic behind bringing all the pieces together. The logic was not the problem. Spelling things correctly and using correct capitalization was the problem. I spent thirty minutes alone testing and retesting changes on a callback that was not working. I only stumbled upon the solution when I re-watched a lecture of Ms. Lanik presenting her first GUI lecture. I was not aware that I needed to capitalize “String” when referencing the string cell of the elements structure that I needed for the callback. Misspellings were not the only problem.

I also had to spend a considerable amount of time debugging my code and doing a little quality control testing to make sure I covered necessary redundancies such as when a user might omit an entry in an edit box that would cause havoc with different array lengths ruining any of the callback functions. Writing out all the if loops to display error strings to the user and direct them to the correct input types took awhile. Not only writing all the if loop processes, but the if loop conditions as well. I had a difficult time covering the situation where a user might input a letter. In order to pull a number out of a string, we usually use the str2double function which will turn any string into class double. That was a problem because the only thing I could think of as a conditional block was to code in a check to see if what was entered was a number. I tried class(‘entry’)==’char’ but that did not work. The str2double functions turns everything into a double. It took a little MATLAB resources research and tinkering to figure out that str2num(‘a’) would return an empty array and str2num(‘any number’) would return the number. I then coded in an isempty(str2num(‘input’))==1 to stop the function call and return an error message. That one took awhile.

In all, I ran into quite a number of problems with writing and debugging the code. Most of them had to do with a lack of familiarity with the code resulting in mistakes spelling, capitalizing, referencing, and calling functions. Another very large road block was in getting all the functions to work together. I did not know how to write to a file and call from a file, so I did not know how to get my GUI function to retain individual inputs. I remembered global variables and ended up making quite a few of the variables global so that the function would retain the information for the duration that it is running and continue concatenating data into an array. Being able to write the arrays into a file and call on the file would make the GUI much more user friendly and powerful in the way that it can process data and save time for the user instead of having to enter everything in manually each time the program is run.

Other than being able to write to and call from files in the future, GUI’s could be very valuable in setting up a clickable machine with a lot of preset functions if working with data where repetitive calculations need to be made. If I need to manipulate a large amount of individual pieces of data using the same function, I could just build a GUI where I can type in the string I need to work with and click a button to manipulate the data instead of having to type in all the functions calls and code necessary to extract the information I need. GUI’s could also be extremely powerful if it is necessary to visualize data in different ways or manipulate the data in different ways such as testing with a weight c1 and retesting with a weight c2 and c3 to see which weight is more applicable to the data. These are just a few of the ways I can see how powerful a GUI can be in working with STEM data. I had an awesome time building a way to visualize and interpret body composition and nutrition data in order to estimate a more consistent and reliable outcome.