BIOS 6660, Spring 2019
Homework 4: Data management, code organization, code quality, bash scripting
Due: Tuesday, February 19th at 10:30am

In this assignment, you will complete a simple data analysis project, applying concepts from the lectures on data management, code organization, and code quality. You will also edit and run a Bash script.

Download the **`Homework_4.zip`** file from Canvas and unzip it inside your the copy of your **`BIOS6660`** repository on your laptop.

Instructions for turning in assignment: As always, submit a GitHub URL through Canvas for the repository version containing your final submission. We will look for modifications to the four source files, as well as the new files in appropriate locations: **`README.txt`**, **`.gitignore`**, **`src.Rproj`**, **`workflow.html`**, **`avg_wkday_traffic.png`**.

Problem 1: Create a README file for the dataset

Inside the **`data`** directory, there are two files downloaded from the Denver Open Data Catalog: a CSV file containing traffic count data, and a spreadsheet explaining the fields in the CSV file. Create a simple text file inside the **`data`** directory called **`README.txt`**, which explains how and when the dataset was obtained and describes each field in the CSV file. For how the dataset was obtained, just pretend you downloaded the data from the aforementioned link on today's date.

Problem 2: Using .gitignore

Create a **`.gitignore`** file inside your **`BIOS6660`** directory, directing git to ignore the contents of the **`Homework_4/data/`** directory. (If you already had a **`.gitignore`**, then just add that directory to it.) Normally, **`.gitignore`** files should also be excluded from version control because they clutter the repository and you may not want others to see them (so normally, you would add ".gitignore" to the **`.gitignore`** file itself). However, for this assignment, you will commit **`.gitignore`** to GitHub so we can check it. We will also make sure the dataset is not in the master copy of your repository on GitHub, i.e., that the ignore worked correctly.

Note: you can have multiple **`.gitignore`** files, up to one per subdirectory in your Git repo. Git will find and process all of them. For this problem, put the file at the top level of **`BIOS6660`**.

Problem 3: Using RStudio Projects

Create an RStudio Project for the **`Homework_4/src`** directory. A file should appear inside **`src`** called **`src.Rproj`**. Now, if you work on something else in RStudio and want to come back to the assignment, you can just open that Project to recover your working directory and open files for the assignment. Similarly to **`.gitignore`** files, you would not typically place **`src.Rproj`** under version control or push it to GitHub. However, again, you will add it to GitHub for this assignment.

Problem 4: Testing and debugging

In **`functions.R`**, we have provided an implementation of a function called **`standardize_street`** that is supposed to standardize street names in our data cleaning step. The function uses a tool called regular expressions to identify and replace various abbreviated versions of identifiers like “street”, “avenue”, etc. Regular expressions are a powerful text processing system, but you do not need to understand them to complete this problem.

If you run the tests in **`tests.R`**, you will see that the test of expected cases fails because there are a couple of bugs in **`standardize_street`**. One line of code contains a single extra character that you will need to correct. Another test case reveals a missing case that you can add to **`standardize_street`** by simply copying the other cases. After you fix the implementation, **`tests.R`** should run through without errors.

Finally, add another edge case to the edge case test in **`tests.R`**: specify a weird but legal input to **`standardize_street`** and its expected output. (See the “TODO” in the edge case test.)

Problem 5: Adding a helper function

In the file **`functions.R`**, follow the instructions inside the body of **`plot_avg_wkday_traffic`** to complete the implementation of that function.

Problem 6: Running the full workflow and knitting the report

Read through the workflow in **`workflow.Rmd`**, and complete the simple “TODO”s at the top of the file. Then run the workflow as a script. It should run through with no errors and should save an image file under the **`output`** directory. When the workflow is running properly, knit the HTML report for this document.

Problem 7: Bash script

The file **`src/check_files.sh`** is a Bash script that checks if you have created all the required files for this homework. First, make the script executable with the command **“`chmod +x check_files.sh`”**, and then run it with **“`./check_files.sh`”**. If you have created all the files in the correct directories, the script should terminate with the message “All done!”. Otherwise, you will see a message indicating which files are missing, and you should go back and make sure to generate the correct files.

Next, add code to **`check_files.sh`** as described in the “TODO” comment in the script. There are a few hints there, and you can also look back in the script and read the explanatory comments to find bits of code you can reuse. We encourage you to use nano or another shell-based text editor to edit the script.

We will download your repo and run your copy of **`check_files.sh`** to make sure it works correctly.