Homoework #7
MSBA 201

Write the appropriate code in R to solve the following problems. Note that while you can use other IDE systems or plain Notepad, R Studio is strongly recommended for most students.

PROBLEM 1

This exercise starts with a single data set located in a database. We will extract this data, and then save it and load it as a JSON data set and a CSV file. Each time we load it we will print out the dataset to make certain the contents are the same.

Libraries required: jsonlite, RSQLite

1. You have been given a data set stored in an SQLite database, named PlanetData.DB.
    a. Connect to the database using dbConnect().
    b. List the tables in the database, using dbListTables(), and print out the result. You should see only one table.
    c. Load that table using dbReadTable(). The name should be "PlanetMass". This should return a dataframe. Assign it to a variable named planetSQL.
    d. Disconnect from the database using dbDisconnect().
    e. Print the contents of planetSQL.
2. Now we want to convert the dataframe to a JSON object.
    a. Use the toJSON() function convert planetSQL into JSON code. Name the variable containing the code outJSON.
    b. Write out outJSON using the writeLines() function, with a filename of "planet.json".
    c. Use the readLines() function to read the file "planet.json". Name the variable containing the code incomingJSON.
    d. Use the fromJSON() function to parse the JSON code in the incomingJSON variable into a dataframe. Name the dataframe planetJSON.
    e. Print out the contents of planetJSON.
3. We want to convert the dataframe to a CSV file at this point.
    a. Use the write.table() function to write out the planetJSON dataframe to a file named "planet.csv".
    b. Use the read_delim() function to read in a datatable (data.table) from the file named "planet.csv". Store it in a variable named planetDataTable.
    c. Print out planetDataTable, as well as the class of the variable. Note that it is **not** a dataframe, but is instead a related class.
    d. Use the read.table() function to read "planet.csv" a second time. Store it in a variable named planetCSV.
    e. Print out planetCSV. Note that this one is a dataframe.

PROBLEM 2

You have been given an Excel file, BeaverData.xlsx. This contains a number of observations of beaver body temperature versus time of day, as well as other information. The following libraries are required: readxl and ggplot2
Libraries required: ggplot2, readxl

1. Load this file into a dataframe using the read_excel() function. Assign it to a variable named BeaverData.
2. Print the contents of BeaverData.
3. Using ggplot2, create a plot that makes a line of best fit (regression line), as shown in class. The x should be time, and the y should be temp (short for temperature). Coloration is not required. You should include the data points (either the jitter points, or just regular points). The method should be the linear model ("lm" method).
4. Using ggplot2, create a line plot that makes a (very jagged) line connecting observations. As with the prior problem, the x should be time and the y should be temp.

PROBLEM 3:

Write a function named getMddle(). It takes in one parameter, a vector. The function should take the vector and remove the first and last elements, and then return the middle elements in a new vector. Error checking is not required (so if you get a vector with one element, do not worry if the function malfunctions or there is a crash). For instance, if I write the following code:

```
testVector <- c(1, 2, 3, 4, 5)
getMiddle(testVector)
```

The result of getMiddle() should be a vector with the values 2, 3 and 4. It is recommended that you include this test code at the bottom of your own code for problem 3 to demonstrate that your function works. It should also adapt to the size of the vector, so you should determine the size of the vector before extracting the elements into a new vector. The length() function may be of use here. Do not use conventional loops – only use the programming capabilities discussed up to chapter 8. NOTE: This should be doable in a few lines of code.

IMPORTANT NOTES:

- ALL of these problems have code examples in the PowerPoint slides, particularly problems 1 and 2. Often you can copy them and insert the new values, and much of the code will be correct. BEFORE you ask me for assistance or give up on the problem, be sure you looked at the PowerPoint slides for the lectures on handling external data and graphics.
- In case it is helpful, the SQLite database we are using was constructed using data sourced from NASA. The specific data was retrieved from this website: https://nssdc.gsfc.nasa.gov/planetary/factsheet/

- In case it is helpful, the beaver data was extracted from data available on the web. You are free to look at the information, but to get full credit you need to load the data from an Excel file.
    - The basic site is https://vincentarelbundock.github.io/Rdatasets/datasets.html listed as the 'beaver' data set.
    - The original CSV file is available at https://vincentarelbundock.github.io/Rdatasets/csv/boot/beaver.csv
    - The documentation on the data set is available at https://vincentarelbundock.github.io/Rdatasets/doc/boot/beaver.html