R Programming Week 1 - Homework

Lab 1: Getting Started with R and Data Loading

Objective: In this lab, you will be introduced to R, RStudio, and data loading methods. By the end of this lab, you should be able to install and set up R and RStudio, load data from a CSV file, and perform a basic exploration of the dataset.

Activity 1: Installation and Setup (5 minutes)

Task: Follow the steps below to install R and RStudio, and configure your environment.

1. **Step 1: Install R**
   - Visit the CRAN R Project website, and download the correct version of R for your operating system (Windows, Mac, or Linux).
   - Complete the installation by following the prompts on your screen.
2. **Step 2: Install RStudio**
   - Go to RStudio's download page and download the free version of RStudio Desktop.
   - After downloading, install RStudio by following the on-screen instructions.
3. **Step 3: Configure RStudio**
   - Once installed, open RStudio. To configure your settings, go to **Tools > Global Options**.
   - Adjust paths, appearance (you can change themes), and other preferences to suit your needs.
   - Now you're ready to begin working in RStudio.

Activity 2: Exploring the RStudio Interface (10 minutes)

Task: In this activity, you'll explore the different components of the RStudio interface and familiarize yourself with how to run commands in R.

1. **Step 1: Console and Source Panel**

   - The **Console** is where you'll enter R commands and see the results immediately. Try typing `2 + 2` and pressing **Enter**. You should see the answer `4` displayed below.

   - The **Source Panel** is where you'll write and save R scripts for later use. Open a new script by clicking **File > New File > R Script**. Here, you can write longer pieces of code.

2. **Step 2: Environment, History, and Files Tabs**

   - The **Environment** tab will show you all the variables and data loaded into your workspace.

   - The **History** tab keeps track of all commands you've run during your session.

   - The **Files** tab displays files in your working directory, where you can access and open data files.

3. **Step 3: Running Basic Commands**

   - In the **Console**, type `print("Hello, World!")` and press **Enter** to see the output.

   - In the **Source Panel**, type the same command, but this time click **Run** or press **Ctrl + Enter** to execute the script. This is how you'll run scripts throughout the course.

Activity 3: Loading Data (15 minutes)

Task: Next, you will load data from a CSV file into R using two different methods: `read.csv()` and `read_csv()` from the `readr` package.

1. **Step 1: Load Data Using** `read.csv()`

   - Download the provided dataset: `Drinking_Water_Quality_Distribution_Monitoring_Data_<date downloaded>.csv`.

Access data set at:

https://data.cityofnewyork.us/Environment/Drinking-Water-Quality-Distribution-Monitoring-Dat/bkwf-xfky/about_data

-Click the “Export” button in the upper right hand corner of the page.

-In the “Export dataset” dialog box that pops up, make sure the “Download file” box is blue.

-In the Export format drop down, select CSV and click the “Download” button in the lower right.

- Save the file in your working directory (you can check your working directory by running getwd() in the Console).
- Load the data by running:

```
dataset <-
read.csv("Drinking_Water_Quality_Distribution_Monitoring_Data_<date
downloaded>.csv")
```

- Confirm the data has loaded by running head(dataset) to display the first few rows of the dataset.

2. **Step 2: Load Data Using** read_csv()

- First, install the readr package if you haven’t done so already:

```
install.packages("readr")
```

- Now, use the read_csv() function to load the data:

```
library(readr)
dataset <-
read_csv("Drinking_Water_Quality_Distribution_Monitoring_Data_<date
downloaded>.csv")
```

- Again, use head(dataset) to check that the data loaded correctly.

Activity 4: Exploring the Dataset (15 minutes)

Task: In this activity, you'll explore the structure of the dataset and learn how to identify missing values.

1. **Step 1: Examine the Structure**

   - To better understand the dataset, use the str() function to check the structure of the dataset:

```
str(dataset)
```

This command will show you the different data types in each column (e.g., numbers, text).

2. **Step 2: View a Summary of the Data**

   - Use the summary() function to get summary statistics for the numeric columns in the dataset:

```
summary(dataset)
```

   - The summary will provide information like the minimum, median, and maximum values for columns like pH, chlorine, and turbidity levels.

3. **Step 3: Identify Missing Values**

   - To check if there are any missing values in the dataset, run:

```
sum(is.na(dataset))
```

This will count the total number of missing values in the dataset.

- To view the rows with missing values, use:

```
dataset[!complete.cases(dataset), ]
```

Lab 2: Data Manipulation Techniques

Objective: This lab will teach you how to manipulate data in R using packages like dplyr and tidyr. You'll learn how to clean, filter, and transform your data.

Activity 1: Basic Data Structures (10 minutes)

Task: In this activity, you'll practice working with basic R data structures like vectors and data frames.

1. **Step 1: Access Columns as Vectors**

   - In a dataset, each column is a vector. To access the `**Residual Free Chlorine (mg/L)**` column from the dataset, run:

```
dataset$`Residual Free Chlorine (mg/L)`
```

2. **Step 2: Access Specific Rows**

   - You can access specific rows and columns of a data frame by specifying the row and column indices. For example, to get the first 10 rows of the dataset:

```
dataset[1:10, ]
```

3. **Step 3: Create a List**

•Lists can contain different data types. Create a list with `Residual Free Chlorine (mg/L)` and `E.coli(Quanti-Tray) (MPN/100mL)` values:

```
my_list <- list(RFC = dataset$`Residual Free Chlorine (mg/L)`, EColi = dataset$`E.coli(Quanti-Tray) (MPN/100mL)`)
```

Activity 2: Basic Data Cleaning (15 minutes)

Task: In this activity, you'll learn how to handle missing data and clean the dataset.

1. **Step 1: Remove Rows with Missing Data**

•To remove rows that contain missing values, use the na.omit() function:

```
cleaned_data <- na.omit(dataset)
```

Check to see if clean (It should return 0):

```
sum(is.na(cleaned_data))
```

2. **Step 2: Fill Missing Values in a Specific Column**

•Sometimes, rather than removing missing data, you'll want to fill it in. For example, fill in missing chlorine values with the mean chlorine value:

How many missing values are there:

```
length(dataset$`Residual Free Chlorine (mg/L)`
    [is.na(dataset$`Residual Free Chlorine (mg/L)`)])
```

Replace missing values:

```
dataset$`Residual Free Chlorine (mg/L)`[is.na(dataset$`Residual Free Chlorine (mg/L)`)] <- mean(dataset$`Residual Free Chlorine (mg/L)`, na.rm = TRUE)
```

How many missing values are there after the repace:

```
length(dataset$`Residual Free Chlorine (mg/L)`[is.na(dataset$`Residual Free Chlorine (mg/L)`)])
```

Activity 3: Introduction to Piping (10 minutes)

Task: This activity will teach you how to use the pipe operator (%>%) to make your code easier to read and write.

1. **Step 1: Install and Load** dplyr

   - If you haven't installed dplyr, do so now:

```
install.packages("dplyr")
```

   - Load the package with:

```
library(dplyr)
```

2. **Step 2: Use Pipes for Cleaner Code**

   - Pipes allow you to chain together multiple operations. Here's how to select two columns (pH and chlorine) and filter for rows where pH is greater than 7:

```
dataset %>%
    select(`Residual Free Chlorine (mg/L)`,`Fluoride (mg/L)`) %>%
    filter(`Residual Free Chlorine (mg/L)` > 1)
```

Activity 4: Advanced Data Manipulation (10 minutes)

Task: This activity will introduce more advanced data manipulation techniques, including summarizing and grouping data.

1. **Step 1: Select and Filter Data**

- Select specific columns from the dataset:

```
selected_data <- dataset %>% select(`Residual Free Chlorine (mg/L)`,`Fluoride (mg/L)`,`Turbidity (NTU)`)

selected_data %>% print(n=200)
```

- Filter the dataset for rows where chlorine levels are above 0.5:

```
filtered_data <- dataset %>% filter(`Residual Free Chlorine (mg/L)` > 0.5)

filtered_data
```

2. **Step 2: Group and Summarize Data**

- Group the data by monitoring site and calculate the mean pH for each group:

```
dataset %>% group_by(`Sample Site`) %>% summarize(mean_chlorine = mean(`Residual Free Chlorine (mg/L)`, na.rm = TRUE))
```