JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Individual Programming Assignment 1

Goal

Practice working with the **shiny**, **ggplot2**, **readr**, and **tidyr** packages. The expected result is published on the following website: Programming Assignment 1.

Note: You do not have to publish your app on the web.

Description

In this programming assignment, you will build a web app that visualizes the **data_PA1.csv** dataset. **data_PA1.csv** is a smaller version of the original JHU Center for Systems Science and Engineering (CSSE) WHO COVID-19 dataset.

It is recommended that you consult the materials provided in Module 1 and Module 2 to familiarize yourself with some examples of the functions mentioned in this assignment.

In this practice assignment, you will complete **app.R**, which is provided to you to create a basic Shiny application that shows the cumulative number of COVID-19 deaths and confirmed cases in each region from March 17, 2020, to June 19, 2020. Please see the guidelines for the detailed instructions.

Supplemental Materials **data_PA1.csv** and **app.R**

Submission: Your completed **app.R**

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Guidelines

STEP 1:

Open a new R script and complete the following steps:

a) Required libraries

Load the following packages for this assignment:

- **tidyverse** for data manipulation and visualization
- **shiny** for building the web application

b) Reading data

- Use the **read_csv** function to read **data_PA1.csv** and save it on a variable named **rawdata**.
- Below is a short description of the dataset's columns:
    - **Type**: It is either **Confirmed** or **Deaths**. It shows whether a specific row includes the information regarding the number of confirmed cases or deaths.
    - **Region**: It shows the geographical regions. It can have one of the following values:
        - Western Pacific Region
        - European Region
        - South-East Asia Region
        - Eastern Mediterranean Region
        - Region of the Americas
        - African Region
    - **Columns 3–97**: Each column represents data for a specific date.

c) Transform data

- As you can see, date information is spread across many columns. To tidy the **rawdata**, use **pivot_longer()** to convert columns 3–97 to only two columns: **Date** and **Count**. Save the converted table on a new variable named **data**.
- Note: You can use **3:97** or **-c(1, 2)** for the **cols** argument of the **pivot_longer()**; **-c(1, 2)** means all the columns except for the first and second columns.

- The new **Date** column is in the **character** format. Use the **as.Date()** function to convert it to date format. You need to tell R what format your date is. In this case, the format is **format = "%m/%d/%y"**. See more examples here. Note: Do not forget to save the converted **Date** column back on itself.

d) Filter data

- Use one of the following two lines of code to filter data only for the number of deaths:

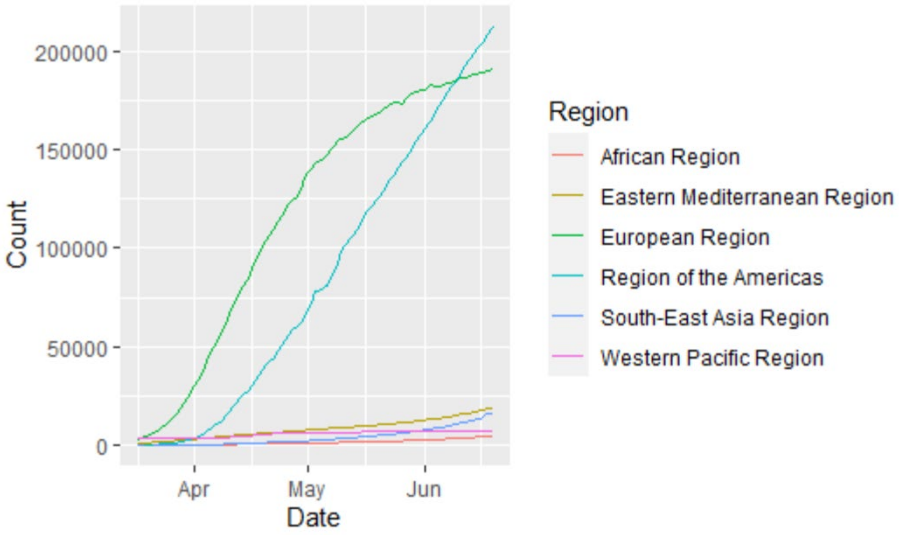
  data1 = data[data$Type=="Deaths",]

  or

  data1 = data %>% filter(Type=="Deaths")

e) Visualize data

- Use **ggplot** and **geom_line** functions to visualize the **data1** dataset.
- The expected result should be a figure similar to the following figure:



STEP 2:

f) Complete the shiny application

- Now it is time to move the code that you wrote into the **app.R** file, but before that, run the **app.R** file to see how this application looks. The user interface (**ui**) part of the code includes a header, a select input, and an invisible place for a plot.

- In the **ui**, change **yourname** to your full name in the code.
- Copy and paste parts b–e in the **server** part of the app. Read the comments in **app.R** for the exact locations.
- Run **app.R** one more time. You will see the plot shown in part e in the application. But if you change the select input choice from **Deaths** to **Confirmed**, the plot does not change accordingly.
- Go back to **app.R** and change part d from
  data1 = data[data$Type=="Deaths",] to **data1 = data[data$Type==input$DeathConf,]**
  or change it from
  data1 = data %>% filter(Type=="Deaths") to **data1 = data %>% filter(Type== input$DeathConf)**
  Now your application should work properly. Basically, you are telling your application to filter data based on the value of **selectinput**, which the user picked (**input$DeathConf**). If this value is **Deaths**, the app will show the plot for the number of deaths; if it is **Confirmed**, it shows the plot for the confirmed cases.

Note: Please only submit the completed **app.R** and do **not** submit your script for parts a–e.