UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

MULTIVARIATE STATISTICAL ANALYSIS

Report Practice 02
Matplotlib

Lecturers Lý Quốc Ngọc
Nguyễn Mạnh Hùng
Phạm Thanh Tùng

Student Võ Nguyễn Hoàng Kim
21127090

CONTENT

A. SELF-ASSESSMENT FORM **3**
B. IMPLEMENTATION **3**
I. Dataset **3**
II. Visualization data **4**
1. Line plot 4
2. Bar plot 5
3. Pie plot 6
C. BONUS **8**
I. DATASET **8**
II. VISUALIZATION AND ANALYSIS **8**
1. Sample 1 8
2. Sample 2 8
III. OTHER VISUALIZATION LIBRARIES **9**
D. REFFERENCES **9**

A. SELF-ASSESSMENT FORM

Features	Note	Level of completion
Read data	From the given Covid-19 cases CSV file	100%
Visualization data	With 3 different graph using Matplotlip (line, bar and pie)	100%
Comments, analytic	Do on corresponding graphs	100%
Bonus	**Note**	**Level of completion**
Read other data CSV files	From the given Covid-19 deaths CSV file	100%
Visualization data Comments, analytic	With 2 different graph using Matplotlip (line, bar and pie) Do on corresponding graphs	100%
Comments, analytic	With 2 different libraries using Pandas plot and Seaborn	100%

B. IMPLEMENTATION

Note: In this program, we use 2 main libraries:

- **Matplotlib.pyplot** (call as **plt**) for visualizing data in graphs [1].
- **Pandas** (call as **pd**) for data processing and refinement [2].

I. Dataset

1. Read data

- Dataset: The given Covid-19 cases CSV file.
- Read data: To read data from the file, we use the **pd.read_csv()** function with the parameter being the path to the data file. Store the data that has just been read into a variable (call as *df*) in the program (this variable will be used for processing, editing, etc., the data for visualization later).
- The value of *df* after we reading the data file as below (The following figure depicts just a portion of the data table stored in the variable *df*):

	Unnamed: 0	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	...	7/5/21	7/6/21	7/7/21
0	0	NaN	Afghanistan	33.939110	67.709953	0	0	0	0	0	...	125937	127464	129021
1	1	NaN	Albania	41.153300	20.168300	0	0	0	0	0	...	132537	132544	132557
2	2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0	...	141966	142447	143032
3	3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0	...	13918	13991	14021
4	4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0	...	39300	39375	39491
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
274	274	NaN	Vietnam	14.058324	108.277199	0	2	2	2	2	...	21312	22341	23385
275	275	NaN	West Bank and Gaza	31.952200	35.233200	0	0	0	0	0	...	314569	314780	314869
276	276	NaN	Yemen	15.552727	48.516388	0	0	0	0	0	...	6929	6931	6934
277	277	NaN	Zambia	-13.133897	27.849332	0	0	0	0	0	...	165513	167132	169003
278	278	NaN	Zimbabwe	-19.015438	29.154857	0	0	0	0	0	...	56014	57963	60227

279 rows × 545 columns

2. Data group

- To save time with implementing some visualizations below, we first group the data by country/region. This is done using the pd.groupby() function: the data in original dataframe *df* is grouped by the countries in the "Country/Region" column.
- Then, it sum over different states/province of each country. The final result is stored in *df_countries* variable (a part of its value as below):

Country/Region	Unnamed: 0	Province/State	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	7/5/21	7/6/21	7/
Afghanistan	0	0	33.939110	67.709953	0	0	0	0	0	0	...	125937	127464	12
Albania	1	0	41.153300	20.168300	0	0	0	0	0	0	...	132537	132544	13
Algeria	2	0	28.033900	1.659600	0	0	0	0	0	0	...	141966	142447	14
Andorra	3	0	42.506300	1.521800	0	0	0	0	0	0	...	13918	13991	1
Angola	4	0	-11.202700	17.873900	0	0	0	0	0	0	...	39300	39375	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Vietnam	274	0	14.058324	108.277199	0	2	2	2	2	2	...	21312	22341	2
West Bank and Gaza	275	0	31.952200	35.233200	0	0	0	0	0	0	...	314569	314780	31
Yemen	276	0	15.552727	48.516388	0	0	0	0	0	0	...	6929	6931	
Zambia	277	0	-13.133897	27.849332	0	0	0	0	0	0	...	165513	167132	16
Zimbabwe	278	0	-19.015438	29.154857	0	0	0	0	0	0	...	56014	57963	6

195 rows × 544 columns

II. Visualization data

Note: In this program, we use several helper functions from the Matplotlib.pyplot library to set parameters and characteristics for the graph, such as:

- **plt.xlabel()**: Set the label for the X-axis.
- **plt.ylabel()**: Set the label for the Y-axis.
- **plt.title()**: Set the title of the chart.
- **plt.show()**: Display the graph.

1. Line plot

a. Purpose

- Observing the data, we realize that it is aimed at recording the number of Covid-19 infections by each day in each country. The recording period spans from January 22, 2020, to July 14, 2021 (a total of 540 days).
- With the provided data, we can construct a Line graph to observe the trend of the pandemic situation over the statistical period for a particular country.

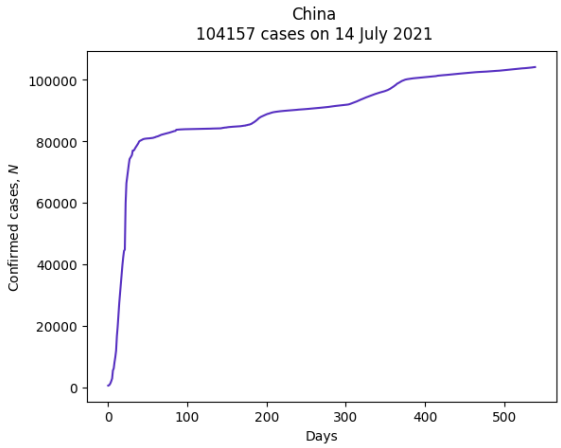
b. Implement

This implementation utilizes the data frame previously processed, which is grouped by country/region.

To make this plot, we initialize a function that is called **visualizeCasesInCountry()** with one parameter is *country* to specify the country (or region) that we want to plot its data.

- Firstly, we extract the data (columns recording the number of infections per day from 01/22/2020 to 07/14/2021) of the corresponding country, resulting in a new data frame named *country_df*.
- By calculating the length of *country_df*, we can determine the number of columns it has, which is equivalent to the number of days during the statistical period (Represent for X axis)
- Additionally, country_df.values provides the corresponding number of infections for each day (representing the Y-axis).
- With the data for the X-axis and Y-axis already available, we use the **plt.plot()** function to create a line graph. Additionally, we can use other functions in the *Note* section to set various characteristics of the graph.

c. Result



d. Analysis

- As we know, China is where the outbreak of Covid-19 was first detected, and the first cases were recorded.
- Based on the chart, this country recorded a very high number of infections and experienced continuous and robust growth during the initial period of the statistical observation, reaching a milestone of 80,000 cases within the first 100 days.
- However, during the remaining period, although the number of infections in China continued to increase, there was a slowdown. Evidence of this is that within the around subsequent 400 days, the number of infections they recorded was only higher by 20,000 compared to the initial period.
- With this chart, we can track the rate of development and spread of the disease daily.

2. Bar plot

a. Purpose

- Observing the data, we realize that it is aimed at recording the number of Covid-19 infections by each day in each country.
- As a result, we can statistic and visualize the top 5 countries with the highest number of infections on a specific day by using a bar graph.

b. Implement

This implementation utilizes the data frame previously processed, which is grouped by country/ region.

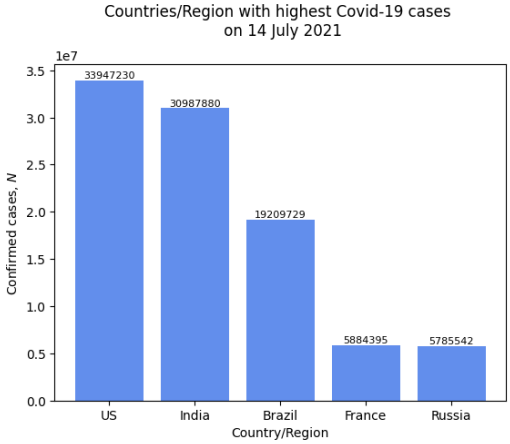
To make this plot, we initialize a function that is called **visualizeTopCountryInDay()** with one parameter is *date* to specify the date that we want to plot its data.

- First, we sort the *df_countries* data frame in descending order based on the values of the *date* column, using the **sort_values()** function.
- From the sorted result, we extract the first 5 countries in the table (corresponding to the top 5 countries with the highest number of Covid infections) using the **head()** function. Record the data of these 5 countries in the variable *top5_df*.
- What we want from the graph is to represent the top countries with the highest number of infections on a specific day. Therefore, the X-axis will represent each country, and the Y-axis will represent the number of cases recorded for that country:
    - For X axis, we use *top5_df.index* to represent countries
    - For Y axis, we use *top5_df[date]* to represent the corresponding number of Covid-19 cases of each country.

- In this plot, we use a **plt.text()** function to make clearly the number of Covid-19 cases of each country.

- Using **plt.bar()** function to plot a bar graph that perform top countries or regions has the highest Covid-19 cases in the *date*.

c. Result



d. Analysis

- At the time of selecting to draw this bar graph, the order of countries by the number of Covid-19 infections is as follows: US, India, Brazil, France, and Russia, ranked in descending order.
- The number of Covid-19 infections recorded in the US is 33,947,230 cases, significantly surpassing the other countries, nearly 5.8 times the number of cases recorded in Russia.
- Thanks to the above graph, we can clearly grasp the alarming situation of the US at that time.

3. Pie plot

a. Purpose

- With the initial data, the statistics include information on the number of infections in the states/provinces of each country.
- Therefore, the pie chart is utilized for the purpose of analyzing and statistic the situation of the pandemic occurring within the states or provinces of a country.

b. Implement

This implementation utilizes the original data frame is *df*.

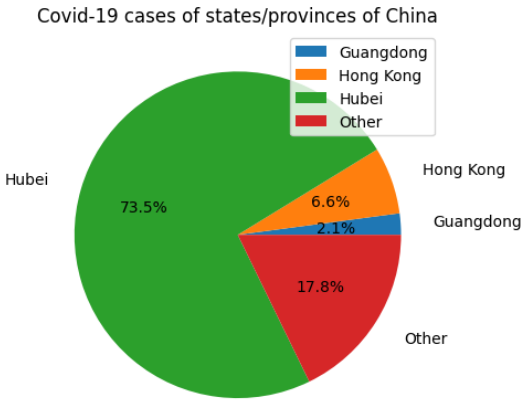
To make this plot, we initialize a function that is called **visualizeCaseInStatesOfCountry()** with two parameters are country and threshold to specify the country that we want to plot its data and a threshold is a filtering parameter for the number of infections used to evaluate the "contribution" of the states/provinces in the graph.

Note: For the purpose stated above, this chart is only drawn when a country has recorded data for its states/provinces in the dataset. Therefore, if an inappropriate country is intentionally passed as an argument, the function will return an "Invalid" message, and no graph will be drawn.

- Firsly, we extract the data of the specified country from the original dataframe *df*, resulting in a new dataframe *df_subset*.

- Check the validity of the function: verify if the input country has data of states/provinces recorded in the dataset (by checking the length of *df_subset*).
- Reset the index for *df_subset* using the **set_index()** function, with the new index value set as "Province/State".
    - Setting the index for *df_subset* to ensure that the names of states/provinces are retained for further steps.
- Calculate the total number of Covid-19 infections for each province/state. Use *sum_rows* variable to store the corresponding results for each province/state.
    - In this step, use the **iloc[]** function to access the necessary positions and calculate the sum using the **sum()** function with *axis = 1* to sum the number of cases during the statistical period (sum by horizontal direction).
- Use a threshold to filter the values:
    - *greater_cases*: cases where the province/state has a number of Covid-19 infections greater than or equal to the *threshold*.
    - *small_cases*: cases where the province/state has a number of Covid-19 infections less than the *threshold*.
    - For elements belonging to *small_cases*, combine their data into a new component named "Other" (the label used for a component in the graph), and store it as a new data frame *sum_small_cases* (using the **pd.Series()** function).
- Combine the components in *greater_cases* and *sum_small_cases* using the **pd.concat()** function to obtain a new dataframe *df_final*.
- With data in *df_final*, use the **plt.pie()** function to represent the percentage of infections in each province/state of a country.
    - In this pie plot, we use the **plt.legend()** function to add a legend about the labels in the graph.

c. Result



d. Analysis

- During the statistical period, in China, the number of Covid-19 infections in Hubei province accounted for approximately ¾ of the total number of infections recorded in China.
- Meanwhile, other provinces, such as Hong Kong and Guangdong, accounted for a very small proportion, less than 10% of the total number of infections in the country.
- The remaining portion comprises other provinces and states, totaling approximately 17%.

- With this chart, a country can be analyzed to understand the complex situation of the pandemic and its most affected provinces/states.
- If more detail is desired, one can simply decrease the *threshold* value.

C. BONUS

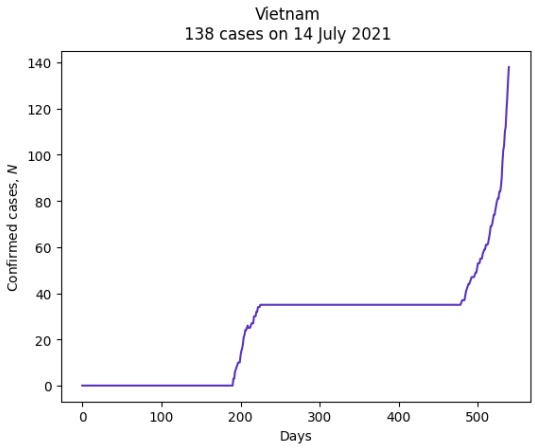
I. DATASET

- In this section, we use data from the Covid-19 deaths CSV file. We use the pd.read_csv() function to read the data from the file.

II. VISUALIZATION AND ANALYSIS

1. Sample 1

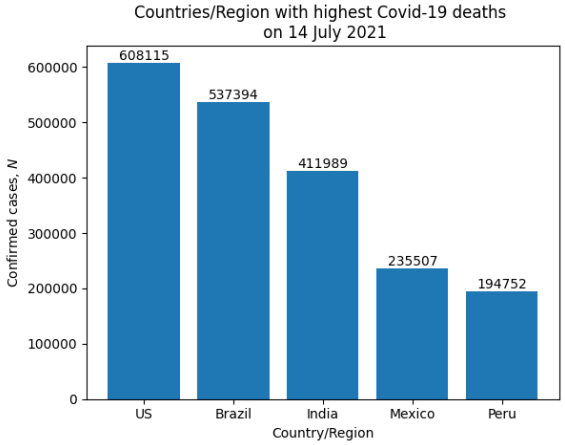
- Purpose
    - Observing the data, we realize that it is aimed at the number of people who died from Covid-19 recorded daily in each country. The recording period spans from January 22, 2020, to July 14, 2021 (a total of 540 days).
    - With the provided data, we can construct a Line graph to observe the trend of the pandemic situation over the statistical period for a particular country.
- Visualization



- Comments, analytic
    - The chart clearly depicts the number of deaths due to Covid-19 in Vietnam during the period from January 22, 2020, to July 14, 2021.
    - In the first 200 days of the statistics, Vietnam did not have any reported deaths.
    - However, towards the end of the statistical period, the number of deaths due to Covid-19 increased rapidly, reaching around 150 cases, with no signs of slowing down.

2. Sample 2

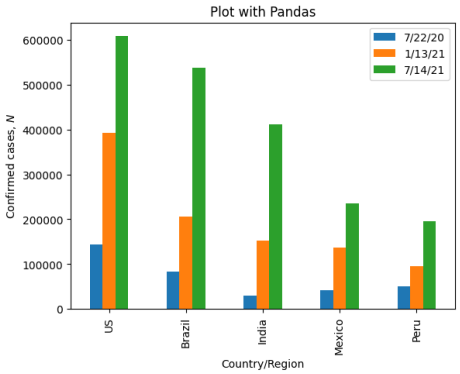
- Purpose
    - Observing the data, we realize that it is aimed at recording the number of people who died from Covid-19 by each day in each country.
    - As a result, we can statistic and visualize the top 5 countries with the highest number of deaths on a specific day by using a bar graph
- Visualization

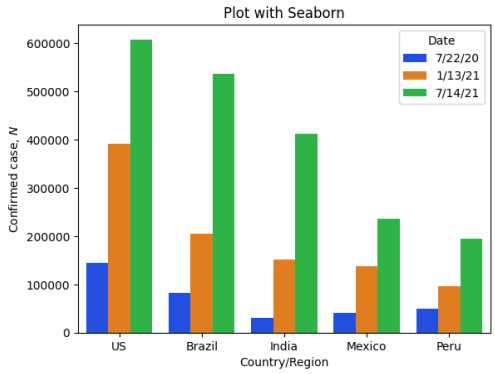


- Comments, analytic
    - At the time of selecting to draw this bar graph, the order of countries by the number of people died from Covid-19 is as follows: US, Brazil, India, Mexico, and Peru, ranked in descending order.
    - The number of deaths by Covid-19 recorded in the US is 608115 cases, significantly surpassing the other countries, nearly 3 times the recorded number of Peru.

III. OTHER VISUALIZATION LIBRARIES

- In addition to Matplotlib, we can use other libraries such as Pandas or Seaborn to visualize data. Below are the graphs created by these libraries using the same dataset and interpretation.





- Even though they yield similar results, the syntax and methods to visualize data on the dataset of each library are different. Therefore, there is a need for being carefull in adjusting the data for visualization.

D. REFFERENCES

[1] The Matplotlib development team., "matplotlib.pyplot," Matplotlib, 2012-2024. [Online]. Available: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html.

[2] pandas, "User Guide," pandas, 2024. [Online]. Available: https://pandas.pydata.org/docs/user_guide/index.html.