Programming assignment 1: Visualizing COVID-19 data

Process Data Analytics (CHE1148H)-N.Anesiadis

General statements (apply to all assignments)

- The assignments are long and simulate the daily work that you will perform in a project. Start immediately upon posting of the assignment.
- Time management is a very important skill. Now you may have other midterms/assignments etc; in a work environment you will be working on multiple projects.
- The instructions are long and tell you what you are supposed to do step-by-step and in detail.
- There are multiple ways to solve any problem. This applies to coding in Python.
- My code is not necessarily performed in an optimal way. It is developed for educational purposes, to explain a complex transformation.

Specific statements (apply only to this assignment)

- This is not an epidemiology study. It is simply a visualization exercise. Some epidemiology terms may not be correct. Related to this tweet.
- ”Not enough testing was done in my country” is a true statement for every country/province/county/city. This is the data we have to work and infer some understanding of what's happening. Data is **always** incomplete and inaccurate.
- Please use only the raw data I share. This makes marking consistent for the TAs. You can use more recent data to further customize your own dashboard **outside** this assignment.

Geographic distribution of worldwide COVID-19 data

Your team wants to develop a dashboard to present daily visualizations of COVID-19 data collected from the European Centre for Disease Prevention and Control. Every day your

team downloads the .xlsx file from the website, reads it, perform some transformations and produces several visualizations that are presented in the dashboard.
Each row of the table contains the number of new cases and deaths reported per day and country. Every section below describes a specific task or visualization to perform.

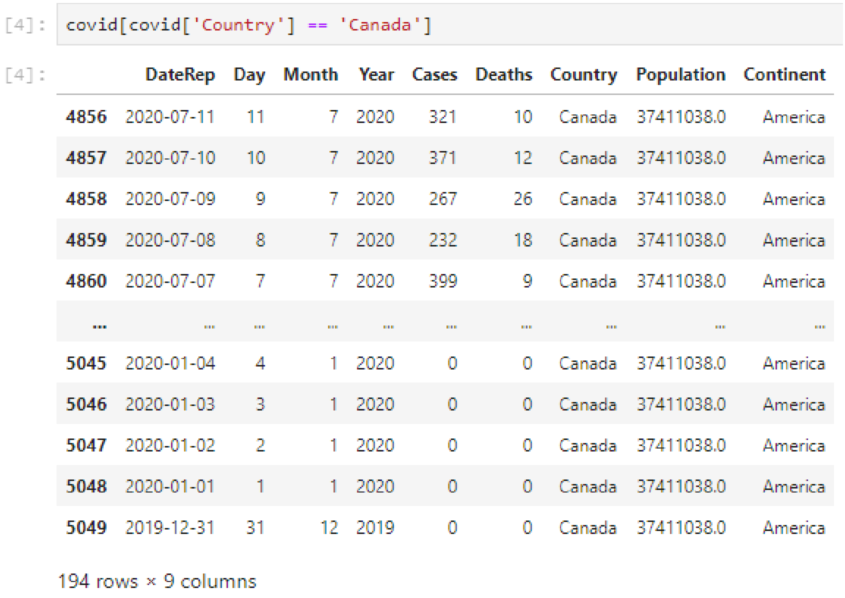
1 Generate the file: covid_a_master_cumulative_table

The data table contains the daily counts. Here, we want to create the table that contains the daily cumulative counts for every country. This will be the main table we will use to generate the visualizations in the next steps. The steps in the first .ipynb file are as following:

1. **Import** packages Pandas, glob and os.

2. **Read** the Excel file using Pandas' pd.read_excel. Packages glob and os may help you automate the reading of the latest .xlsx file that you download in the same folder.

3. **Rename** some columns and **replace** some country names with shorter versions. **Drop** unnecessary columns.
   The snapshot below shows the **pandas dataframe** filtered (or sliced) by 'Country' == 'Canada'. We select all rows that have the value 'Canada' in column 'Country'. Note the dimensions of the slice at the bottom and the indexes in bold font as the left column.
   As we mentioned earlier, the daily counts of the cases and deaths are not what we want. We want to generate and plot the cumulative counts for every country.



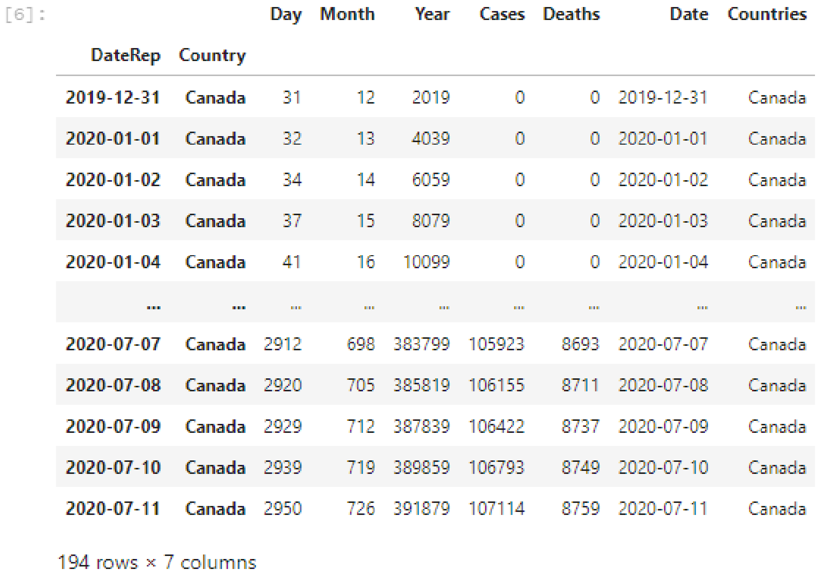
```
[4]: covid[covid['Country'] == 'Canada']
```

[4]:

	DateRep	Day	Month	Year	Cases	Deaths	Country	Population	Continent
4856	2020-07-11	11	7	2020	321	10	Canada	37411038.0	America
4857	2020-07-10	10	7	2020	371	12	Canada	37411038.0	America
4858	2020-07-09	9	7	2020	267	26	Canada	37411038.0	America
4859	2020-07-08	8	7	2020	232	18	Canada	37411038.0	America
4860	2020-07-07	7	7	2020	399	9	Canada	37411038.0	America
...	...	...	...	...	...	...	...	...	...
5045	2020-01-04	4	1	2020	0	0	Canada	37411038.0	America
5046	2020-01-03	3	1	2020	0	0	Canada	37411038.0	America
5047	2020-01-02	2	1	2020	0	0	Canada	37411038.0	America
5048	2020-01-01	1	1	2020	0	0	Canada	37411038.0	America
5049	2019-12-31	31	12	2019	0	0	Canada	37411038.0	America

194 rows × 9 columns

4. Generate the cumulative counts per day and country using **groupby()**, **sum()** and **cumsum()**. **Re-assign the indexes** 'DateRep' and 'Country' as columns. Also, drop the cumulative sums of: 'Population', 'Day', 'Month', 'Year' as they are meaningless.



[6]:

		Day	Month	Year	Cases	Deaths	Date	Countries
DateRep	**Country**							
2019-12-31	**Canada**	31	12	2019	0	0	2019-12-31	Canada
2020-01-01	**Canada**	32	13	4039	0	0	2020-01-01	Canada
2020-01-02	**Canada**	34	14	6059	0	0	2020-01-02	Canada
2020-01-03	**Canada**	37	15	8079	0	0	2020-01-03	Canada
2020-01-04	**Canada**	41	16	10099	0	0	2020-01-04	Canada
...	**...**	...	...	...	...	...	...	...
2020-07-07	**Canada**	2912	698	383799	105923	8693	2020-07-07	Canada
2020-07-08	**Canada**	2920	705	385819	106155	8711	2020-07-08	Canada
2020-07-09	**Canada**	2929	712	387839	106422	8737	2020-07-09	Canada
2020-07-10	**Canada**	2939	719	389859	106793	8749	2020-07-10	Canada
2020-07-11	**Canada**	2950	726	391879	107114	8759	2020-07-11	Canada

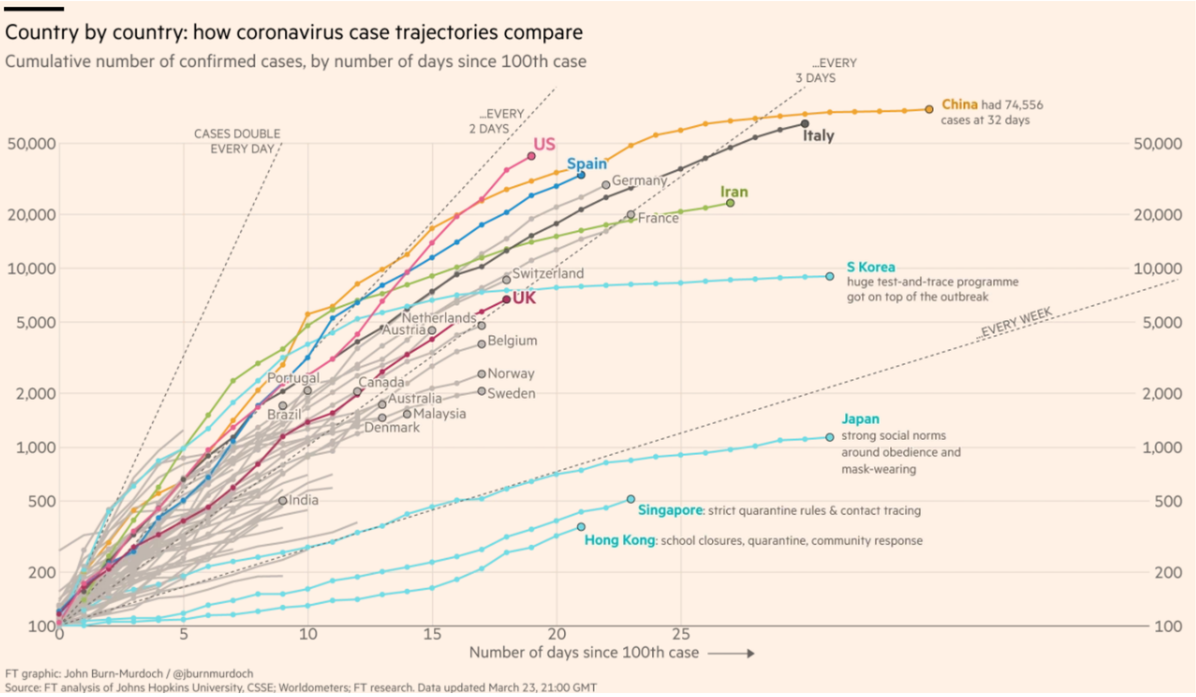
194 rows × 7 columns

5. **Join** (or **merge**) with the original table to obtain the population data for every row.

6. **Reset the index**, drop irrelevant columns and **save** the final table with the name: 'covid_a_master_cumulative_table.xlsx'.

2 Plot the cases: covid_b_cases_cumulative_analysis

Here, we start by replicating a very popular graph that appeared in early March in the Financial Times. We will pick the N = 25 countries with the highest cumulative number of confirmed cases and plot versus the number of days since the K = 100th case. Also, plot the dashed lines that show the doubling of cases every 1, 2, 3 and 7 days.


Country by country: how coronavirus case trajectories compare

Cumulative number of confirmed cases, by number of days since 100th case

FT graphic: John Burn-Murdoch / @jburnmurdoch
Source: FT analysis of Johns Hopkins University, CSSE; Worldometers; FT research. Data updated March 23, 21:00 GMT

1. **Import** all the packages needed and **load** the file you created in the 1st part: (covid_a_master_cumulative_table).

2. **Filter** the last day for every country and create the **pandas.DataFrame** 'last_day_per_country'.

3. Define the number of top countries you wish to plot (N = 25). **Join** with the dataframe 'covid_cumulative' with 'last_day_per_country' and **sort** the values of 'Cases' in descending order. **Slice** the top N countries and convert the country names to a **list**.

4. **Filter** rows in the dataframe 'covid_cumulative' that contain values of the top N countries and more than K = 100 confirmed cases.

5. Create dataframe 'day_0_for_every_country' that identifies Day_0 for each of the top N countries by applying **groupby** and **min** in the previously filtered dataframe.

6. **Merge** it back with the original dataframe 'covid_top_countries' and subtract columns 'Date_x' and 'Day_0' to calculate the 'Days_from_0'.

7. Generate the df 'max_cases' and save it as 'covid_b_max_cases.xlsx'. This df is helpful in labeling the graphs and contains the snapshot of the countries with statistics such as: Case fatality rate, Infection rate per 1 million, Mortality rate per 1 million.

8. Create the dataframes: x1,..., x7 and y_1,..., y_7 that capture the doubling of cases every 1, 2, 3 and 7 days. The simple formula is $K \times 2^{(days)}$, where K is the number of cases on the first day, which is 100 for this dataset.

9. Use the **Seaborn** package (abbreviated as sns) to plot the lines of the countries cases and the theoretical lines for the doubling of cases. Some aesthetic considerations:
    - Set the y-scale to logarithmic. The data is highly non-linear.
    - Format the y-axis tickers to have the number format: ,.0f. That is: comma to separate thousands and zero decimal places.
    - Use the package **adjust_text** to optimize the position of the labels. This package tries to minimize the overlap of the text labels whenever possible (results may vary).
    - Label every line and add text to define Day 0.

3 Plot the daily cases moving averages

Plotting the daily cases is straight-forward. In most dashboards, they are plotted as bar graphs. However, it is common with time-series data that fluctuations or noise obscures the long-term trends or cycles. **Moving average** is a commonly used method that smooths out short-term fluctuations and we will use this technique to plot the data in the dataframe 'covid_top_countries' we already built with the cumulative cases.

Since we have the cumulative cases in the last df, we will calculate the moving slope of the last 7 days (back_calc = 7) using the stats.linregress, a statistical function from the **SciPy** package.

10. We use a **for** loop to iterate through every country in the 'top_countries' and calculate the moving slope of the previous 7 days. The steps we follow are:
    - Initialize the back window and the slope_df with the columns we want to capture.
    - Create the high level loop that filters one country at a time (one_country_data).
    - Create the low level loop that iterates from the 7th to the last row of each country's data:
        - creates the 7-day windows x and y
        - calls the stats.linregress to calculate: slope, intercept, R^2, p-value and standard error
        - appends the slope, R^2 and p-value to the 'slope_df' dataframe.

    Notice that when we print the top 15 rows (slope_df.head(15)), the first 6 rows contain **NaN** values. This happens because the first slope is calculated with the first 7 rows and the first result appears in the 7th row.

11. Set 'Countries' as the index and save the file: covid_b_slope_daily.xlsx

12. Create the df **max_slope** with the maximum slope value per country and sort them in descending order. This way, we then filter appropriately to create the 4 dataframes data_1_5, data_6_10, data_11_15 and data_16_20 to plot them in the 4 subplots below.

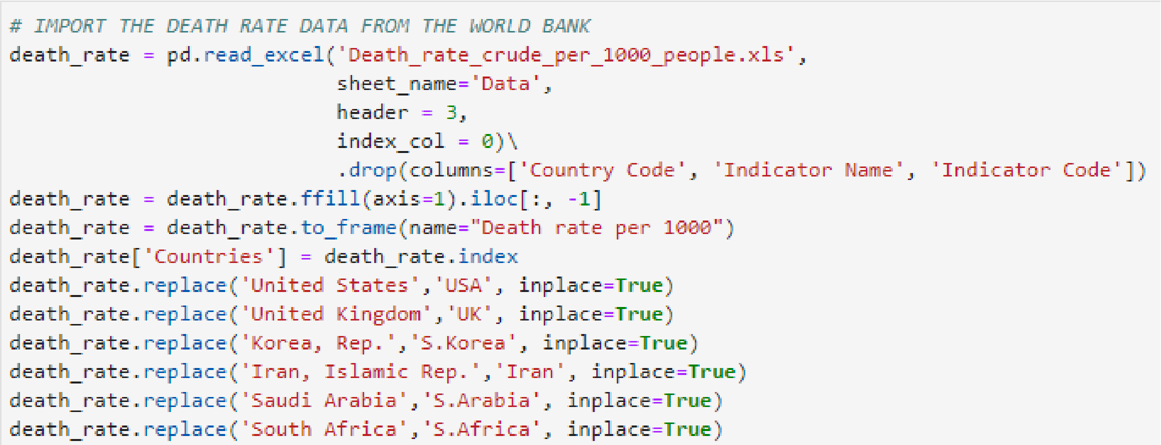
13. Use Seaborn's **lineplot** and plot the 4 dataframes with the top 20 countries in a 2×2 subplots image. Use the package **adjust_texts** to annotate the lines and linear scale in y-axis.

4 Task #1: covid_c_mortalities_cumulative_analysis

A. Replicate the two graphs we created in Section 2 and 3 for the column 'Deaths'. Pick N = 20 countries and start Day 0 at K = 10 deaths.

B. A term that people started studying after the first few months of the pandemic is that of **excess deaths**. Excess deaths are the additional deaths to the typical death rate that are due to Covid-19. E.g. if the typical death rate for a country is 1000 deaths/day, and the Covid-19 deaths for a day is 500, then the excess death rate is 50%. You should see 1500 deaths/day in that country if the Covid-related deaths are independent of the expected deaths (this is not the case).

To calculate the excess deaths, you need an estimate of the typical death rate per country. Use the table 'Death_rate_crude_per_1000_people.xls' with the code below to import and make some changes in the dataframe (the data set is downloaded from the World Bank Data).



```
# IMPORT THE DEATH RATE DATA FROM THE WORLD BANK
death_rate = pd.read_excel('Death_rate_crude_per_1000_people.xls',
                           sheet_name='Data',
                           header = 3,
                           index_col = 0)\
                           .drop(columns=['Country Code', 'Indicator Name', 'Indicator Code'])
death_rate = death_rate.ffill(axis=1).iloc[:, -1]
death_rate = death_rate.to_frame(name="Death rate per 1000")
death_rate['Countries'] = death_rate.index
death_rate.replace('United States','USA', inplace=True)
death_rate.replace('United Kingdom','UK', inplace=True)
death_rate.replace('Korea, Rep.','S.Korea', inplace=True)
death_rate.replace('Iran, Islamic Rep.','Iran', inplace=True)
death_rate.replace('Saudi Arabia','S.Arabia', inplace=True)
death_rate.replace('South Africa','S.Africa', inplace=True)
```

You have to **left join** the dataframe 'death_rate' with the dataframe 'covid_top_countries' on 'Countries' to make sure that your main table has the information you need.

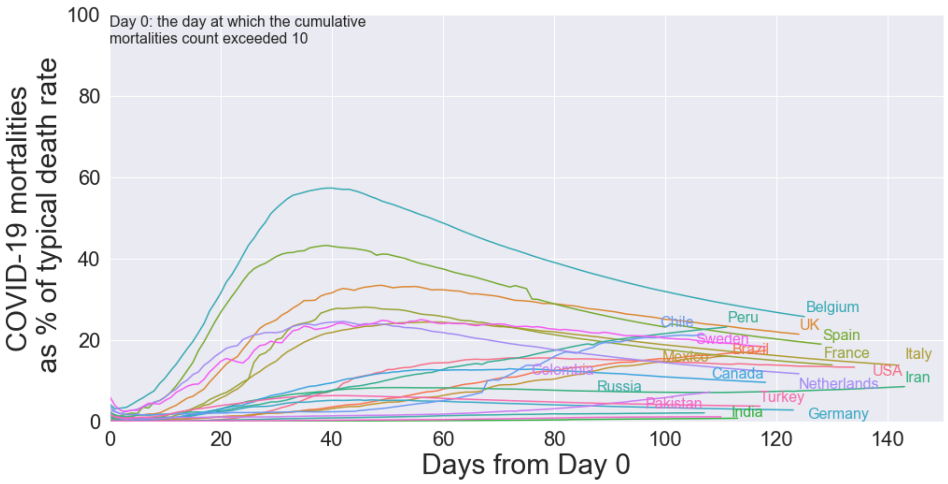
The crude death rate reported in the World Bank Data is an annual average per 1000 people. Therefore, in order to convert it to daily typical deaths you need to do:

```
df['Typical_deaths_per_day'] = df['Death rate per 1000']*
                               df['Population']/1000/365
```

Then, you can calculate the typical cumulative deaths and the **excess deaths rate** as the ratio of the cumulative deaths over the typical cumulative deaths expressed as %, with the help of the following code:

```
df['Typical_cumulative_deaths'] = (df['Days_from_0'] + 1)*
                                    df['Typical_deaths_per_day']
df['Excess_death_rate'] = df['Deaths'] /
                          df['Typical_cumulative_deaths'] * 100
```

The final output and what is asked in this part should look similar to the graph below.



Submit: code and graphs.
Naming convention for .ipynb file:
covid_c_mortalities_cumulative_analysis_LastName_FirstName
Naming convention for .png files:
Fig_1_1_LastName_FirstName, Fig_1_2_LastName_FirstName, etc

5 Task #2: Plot the profile rates per 1M residents

Since the countries populations vary a lot, counting the absolute cases and deaths is not a fair metric. Normalizing the cases and deaths per 1 million residents is a typical procedure to get an unbiased view of the data. To do do, simply use the logic below:

```
df['Cases\_per\_1m'] = df['Cases']/df['Population']*1000000
```

Use the same logic we used in Section #3 (steps 12-13) to plot the 2×2 subplots with:

a) cumulative infection rate per 1M

b) cumulative mortality rate per 1M

Here, notice that the cumulative rate profiles are smooth and you don't need to calculate the moving average. Also, in both cases, make sure that you sort by each rate per 1M so that the top 5 countries appear in the [0, 0] subplot (top left), the 6-10 appear in the [0, 1] subplot (top right), and so on.

Submit: code and graphs.
Naming convention for .ipynb file:
covid_d_infection_rate_profiles_LastName_FirstName
covid_e_mortality_rate_profiles_LastName_FirstName
Naming convention for .png files:
Fig_2_1_LastName_FirstName, Fig_2_2_LastName_FirstName

6 Interactive visualizations: covid_f_bokeh

So far, we have only seen dynamic visualizations; plots with time in x-axis. The challenges are obvious: too many lines, too many colours, difficulty in identifying the countries. In this Section, we will use Bokeh to plot the current rates per 1M (i.e. the last day for every country). This is considered a static plot since it only shows the current state of the country and not the dynamics. Bokeh however can be used to generate the plots described in the previous sections.

Bokeh is a Python package that enables you to make simple, complex and specialized interactive plots. The core data structure in Bokeh is called **ColumnDataSource**. Here, we will convert the pandas dataframe to a ColumnDataSource to be able to build a Bokeh plot. The main building block of Bokeh plots is called **glyphs**. These are the circles, squares and other symbols you use to create plots.

In file covid_f_bokeh.ipynb we follow the steps:

1. **Import** necessary packages and functions. Look in the code below to see where each function is used.

2. **Read** the master table with the cumulative cases.

3. **Filter** the last day of the results and the top N = 50 countries in terms of cases.

4. **Calculate** 3 relevant metrics: Case fatality rate, Infection rate per 1m and Mortality rate per 1m.

5. **Create** a new column to represent the color according to the continent of each country.

6. **Calculate** the average values for the 3 relevant metrics. These will be used as baselines in the graphs we will plot later.

7. **Generate** the **ColumnDataSource** data structure from the pandas dataframe max_cases. This is the data structure needed for further plotting in Bokeh.

8. Here, we will configure all the components needed to generate the Bokeh figure:

   - Define the title and tools we want to include.
   - Create a new Figure for plotting with the arguments defined in the parenthesis.
   - Configure colours, axis labels, fontsize and number format.
   - Call the **glyph** method **circle** to plot the x-y in source_1 data structure. Use 'Continent' as the legend, the respective 'Color' as the circle color and black color as the line of each circle.
   - Configure the information we want to present when **hovering** over the different data points. Define variables and formats.
   - **Annotate with labels:** use the country names text to annotate each circle. Some offset is defined along with other properties of the text such as font size, color and style. The annotation is considered an additional component or layout that is arranged or overlaid along with many other components of the plot. The **add_layout** function adds the newly created component to the existing plot.
   - **Annotate with horizontal and vertical lines** representing the average x and y (hline and vline, respectively). The lines extend to the edge of the plot area.
   - **Add text annotation** to explain what the dashed lines show.
   - **Output the file** as an .html file. This type of file can be used in a website and contains all the information to allow the used to interact with it.
   - Finally, we show the plot generated in a new tab.

7 Task #3: Generate your Bokeh plot

The case fatality rate depends heavily on the testing strategy of each country (i.e. the denominator) and is not considered a robust metric.

Here, we want to plot the Infection rate per 1m (x) versus Fatality rate per 1m (y). **In the same .ipynb file**, add one extra cell of code where you generate the same interactive plot as in the previous section with Fatality rate per 1m as y. Make sure you change the annotations to the appropriate names (e.g. title, y-axis label, hover tips info, hline). Also, both axes need to be logarithmic.

Submit: code (the same as the one given, covid_f_bokeh, with your extra code) and html graph file.
Naming convention for .ipynb file:
covid_f_bokeh_LastName_FirstName

Naming convention for .html files:
Fig_3_LastName_FirstName

8 Plot the global cases and per continent

In this section, we will create time profiles similar to the previous ones but at the continent level. The steps we follow in each cell are:

1. **Import** necessary packages and functions.
2. **Read** the master table with the cumulative cases.
3. Perform the **groupby** aggregation to **sum** the 'Cases' and 'Deaths' over **Date** and **Continent** (notice that here we use the original data, not the cumulative).
4. Generate the **cumulative sum** of Cases and Deaths per Continent.
5. Generate the **global sum** aggregation.
6. Generate the **cumulative sum** at the global level.
7. Define two functions to convert values of millions and thousands to M and k format, respectively. E.g. 1,000,000 will be represented in the graph as 1M and 1,000 as 1k.
8. Configure the x-axis ticks format. Here, we want to show the months instead of Day from 100 cases. The format specified here takes care of that.
9. Create the data frame with the values on the last day. This will be helpful to annotate the graphs. Also, generate the list of the continents.
10. Generate the 2×2 subplots of the daily and cumulative cases for continents and globally.

9 Task #4: Smoothing the noisy data

The graphs clearly show a noisy pattern with significant drops every 7 days for Europe and America. This is probably due to reporting being limited in some countries on certain days.

Here, you are asked to use the 7-day smoothing method we used in section 3 that uses **stats.linregress** to generate the same 2×2 subplots of the daily cases with the smoothed daily cases in the top 2 graphs.

Also, in a separate 2×2 subplots graph show the 7-day smoothed plots of the daily mortalities and the cumulative mortalities.

Submit: code (the same as the one given, covid_h_master_continent_cumulative, with your extra code) and the two .png files.
Naming convention for .ipynb file:
covid_h_master_continent_cumulative_LastName_FirstName
Naming convention for .png files:
Fig_4_1_LastName_FirstName and Fig_4_2_LastName_FirstName

Deliverables

This is an overview of the files you need to submit by the deadline mentioned in the Syllabus:

Task 1:	covid_c_mortalities_cumulative_analysis_LastName_FirstName + 3 png figures
Task 2:	covid_d_infection_rate_profiles_LastName_FirstName
	covid_e_mortality_rate_profiles_LastName_FirstName + 2 png figures
Task 3:	covid_f_bokeh_LastName_FirstName + 1 html figure
Task 4:	covid_h_master_continent_cumulative_LastName_FirstName + 2 png figures

and each task is worth 6.25 points for a total of 25 points for this assignment.