BAN405 mandatory assignment #2

Deadline: October 28 at 16:00

Wrangle and Visualize Global COVID-19 Deaths

In this assignment, you'll work with real-world, messy data that must be wrangled into a format suitable for analysis.

The Center for Systems Science and Engineering at Johns Hopkins University has a GitHub repository where they publish data related to the spread of COVID-19. The file "time_series_covid19_deaths_global.csv" contains time series on covid deaths. This data set was updated daily during the pandemic (updates to the repository ended on March 10th, 2023).

Each row in the file represents the cumulative number of COVID-19 deaths reported for a given Province/State (if applicable) and Country/Region on each date during the pandemic:

- The deaths are *cumulative* and therefore increasing over time
- Some countries (e.g., Canada) report deaths at a subnational level so these countries have multiple rows (one for each province or state)
- The file covers all countries in the world, but it also includes a few non-country entities, such as "Diamond Princess" or "Summer Olympics 2020", which were special cases during the pandemic.
- Date columns follow the format m/d/yy (e.g., "1/22/20") starting from January 22, 2020 and until March 9, 2023.
- The file also contains two columns with the longitude and latitude of each location. You can ignore these columns as they will not be used.

The goal in this assignment is to wrangle the data into a format the is suitable for visualizing COVID-19 deaths over time in different countries and regions. Note the following:

- All data files can be found in the "data" subfolder
- Examples of the required graphs can be found in the "plots" subfolder.
- A few tasks are marked as "Optional", and you are not required to complete these to receive a pass on this assignment.

Task 1: Data wrangling

1. Load the COVID data set and explore the data. Make sure that you understand the structure and content of the data set, such as the data types, missing values, and unique countries observed etc.
   Note: You can also import the file directly from GitHub using this link.

2. Reshape the data from wide to long so that dates are in a single column (i.e., tidy format). This will make it easier to visualize the time series later.
   Hint: Use the pandas method `melt()`.

3. Convert dates to timestamps with the correct date format.
   Hint: Dates in the file are in a non-standard format (e.g., "1/22/20"), so use `format='%m/%d/%y'` in `pd.to_datetime()`.

4. Aggregate the data to the country-level. As some countries have multiple provinces or states, you must sum up the deaths across province/state in each country on each day.

5. The data contains the cumulative sum of deaths over time. Create a new column that contains the daily number of *new* deaths in each country on each day.
   Note: Daily new deaths are simply the difference between the total number of deaths between two dates.
   Hint: Use the pandas method `diff()` to subtract the value between two adjacent rows for each country.

Task 2: Data visualization

1. Using the final data from the previous task with total and new deaths, produce the following two visualizations:

   a) A *single* graph that contains line plots of total deaths over time for the three countries with the highest total number of COVID-19 deaths in the data. Save the graph as "total_deaths.png".
   Note: You should choose the three countries based on the total deaths as of the *last* date in the data set.

   b) A figure with three *subplots* that show the daily number of new deaths for Norway, Denmark and Sweden. Save the graph as "new_deaths.png".
   Note: You should plot each country in its own subplot (either 1x3 or 3x1).

2. **OPTIONAL**: Create a reusable plotting function called `plot_total_deaths(countries, data)` that displays a single line graph that contains the total number of deaths over time for one or more countries. The function must:
    - Accept a DataFrame (`data`) with the covid deaths and a list of countries (`countries`) to include in the plot.
    - Plot each country as a separate line on the same axes.
    - Plot the total number of deaths for the selected countries.

    You should ensure that the function validates user inputs to avoid potential errors. Test your function with the following list of countries:

    ```
    ['Norway', 'Denmark', 'Sweden']
    ['Norway', 'Denmark', 'Atlantis']
    ['The moon', 'Mars', 'Atlantis']
    []
    ```

Task 3: Data merging

In this final part, you'll expand your COVID-19 data set with continent information to explore how the pandemic impacted different parts of the world. The file "Countries Continents.csv" is downloaded from the OWID GitHub repo and it contains an overview of countries and their corresponding continent in 2015.

In this task, you'll need to merge your COVID-19 data with this new data set.

1. Load and explore the new data set.
   Note: There are slight name variations across the two data sets. For example, "US" vs "United States" and "Korea, South" vs "South Korea". To improve the merge quality, you can update these country names in one of the data sets so that they match.

2. Perform a *left join* to add the continent information to your data on covid deaths. After merging:
    - Verify that most countries now have a valid continent value.
    - Count how many countries are missing continent data.

3. Using the merged data, calculate the total number of deaths per continent. Which continent had the highest number of COVID-19 deaths in total?

4. Create a bar plot that shows the total number of COVID-19 deaths per continent. Save the graph as "continent_bar_plot.png".

5. **OPTIONAL**: Create also a *stacked* bar chart showing the total number of deaths per *year* for each continent (see "continent_stacked_plot.png" for an example). **Hint**: It is easier to create stacked bar charts using the plotting methods from pandas instead of matplotlib.

Submission instructions

1. Your submission should be a **zipped** folder that contains:
    a. One Jupyter notebook (.ipynb). Your notebook should run without errors from top to bottom (check with: `Restart` and `Run all`).
    b. Any additional files (e.g., data files, subfolders) required to run the notebook.
    c. If something cannot run, leave the code as-is and add a short markdown note explaining what fails and why.
2. Your notebook should be well-structured and documented with markdown cells as headers and to explain the steps in your analysis. Add also in-line comments in the code where appropriate.
3. Add a statement on your use of generative AI in a markdown cell at the end of the notebook. In case you have not used generative AI for assistance, this should also be stated.
4. To receive a pass, you must demonstrate an honest attempt to complete all tasks in the assignment (except the ones marked as "optional").
    a. In case you are unable to complete a task, you should explain this in a markdown cell.