Mandatory Assignment 6

Web Programming and Data Analysis (30 + 16 points/40 +16 points)

University of Oslo - IN3110/IN4110

Fall 2021

Your solution to this mandatory assignment needs to be placed in the directory `assignment6` in your GitHub repository. The directory must contain a `README.md` file with information on how to run your scripts, required dependencies and packages (versions the code ran with and operating system) as well as how to install them. Documentation on how to run your examples is required. Furthermore, your code needs to be well commented and documented. **All** functions need to have docstrings explaining what the function does, how it should be used, an explanation of the parameters and return value(s) (including types). We **highly** recommend you use a well-established docstring style such as the Google style docstrings[1]. However, you are free to choose your own docstring style. We expect your code to be well-formatted and readable. Black is useful tool for automatically formatting Python code [2]. Coding style and documentation will be part of the point evaluation for **all tasks** in this assignment. This assignment will be followed by a peer-review. Delivery of the assignment is required for participation in the peer review.

A Note on Your Solutions There is no such thing as the "one right solution" to the assignments. As a response to the survey, we wanted to mention here that the assignment text is kept relatively open as we want to give you enough freedom to implement the solution in a way that you feel works best. In real software development, you will not have a step-by-step exact guide on how to implement a certain feature. You will likely know, what the product is supposed to look like in the end and will have to figure out a way, how to best create a product with certain features. With these assignments, we want to give you a chance on learning how to tackle a problem, how to structure a project and realizing that there is not necessarily one perfect solution.

Files to Deliver for this Task

All `.py`-files will be delivered in the same `assignment6` folder.

You will be able to reuse your code from previous tasks in this assignment avoiding code duplication.

[1] `https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html`

[2] `https://black.readthedocs.io`

Files required for all tasks

- `README.md` including dependencies and their installation and commands you used for running your examples and creating your output files

6.0: Background (0 points)

In this assignment, you will be building a web-based visualization of the corona dataset made publicly available by the John Hopkins University (JHU). The data can be found on the following website `https://ourworldindata.org/covid-cases`

We will be working with the dataset "Explore the global data on confirmed COVID-19 cases". Our plot will be based on the **daily** reports. You can download the data as `.csv` for example by clicking the download button and download the **coronavirus-data-explorer.csv**. We have attached a screenshot.

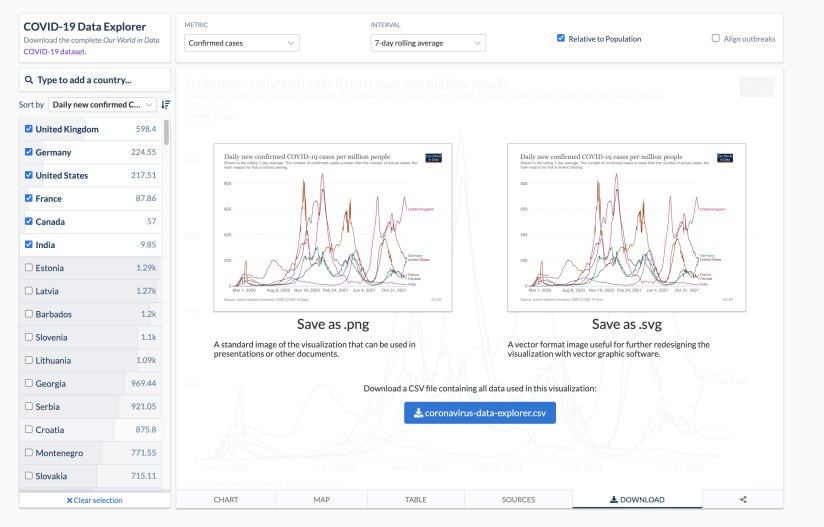

Figure 1: Reported Cases Figure with "Download Button" highlighted.

6.1: Cases Over Time Plotter (10 points)

Build a Python script which reads data from the `.csv` file from 6.0 and generates a labeled,[3] nice plot of "Daily new confirmed COVID-19 cases per million people" by date.

By default (if no countries are specified), plot the 6 countries with the highest occurrence of daily new confirmed cases per one million people based on the end

[3]Imagine you want to show the plotted graph to a friend, that has not worked with that data set. They will need a title for the graph, labels or titles for each axis and the corresponding "units" to understand what is going on in your plotted graph.

date given. If no end date is given, you can base your assumptions on the latest date available.

In the `.csv` file you will use the columns `new_cases_smoothed_per_million` or `new_cases_per_million` for creating the plot, you are free to choose.

You should implement the function `plot_reported_cases_per_million()`, to create a line plot of the cases per million people, by date.

The user of those functions should be able to control the following with input arguments:

- Which countries to plot the data from - *optional argument*
- the time range to be plotted e.g. Start = "2020-10-02" End = "2021-10-02" - *optional argument*
- You are free to choose your own default of the "Start" and "End" Date
- The time range could be given either in string format or as a datetime object - you can decide.

Make sure to mention what your default arguments are in your `README.md`

You should use Pandas in this task for handling the data. We encourage you to use Altair for plotting, but you are free to choose your favorite tool yourself and will not lose points, using your preferred tool.

An unfinished outline for your script `webvisualization_plots.py` can be found in the file `webvisualization_plots_outline.py`. Feel free to use it or not.

The script `webvisualization_plots.py` should be run as followed:

```
python webvisualization_plots.py
```

The output produced by the script should be a line plot of the cases per million people created by your function `plot_reported_cases_per_million()`. An example plot is shown below.

The plot created can be displayed to the user or saved to a file. In the outline provided, we are triggering the chart display using the built-in `chart.show()` method.

A demo using altair and FastAPI on a comparable issue can be found here `https://github.com/UiO-IN3110/UiO-IN3110.github.io/tree/main/lectures/12-visualisation/altair_web_demo`

Note that, viewing the chart created by altair outside jupyter notebook or vs code for example, might require installing the altair viewer. The altair viewer can be installed via:

```
pip install altair_viewer
```

Files Required in this Subtask

- `webvisualization_plots.py`

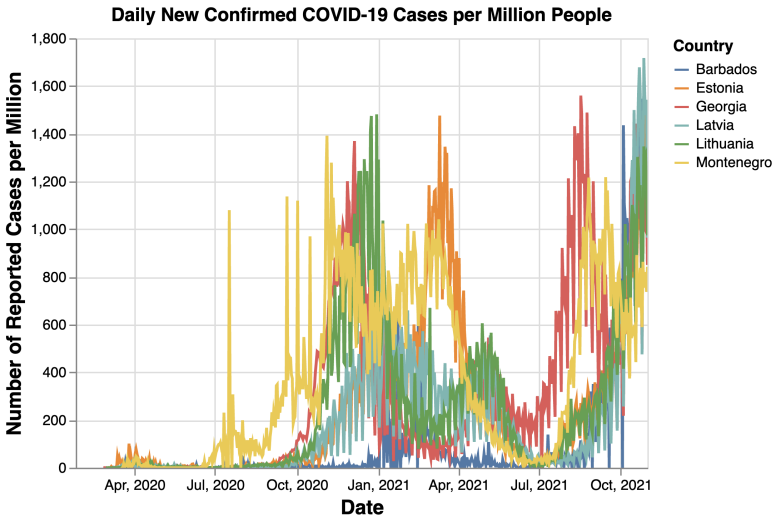

Figure 2: Example line plot of the cases per million people by date.

6.2: Becoming a Web App Developer Using FastAPI (5 points)

Build a FastAPI app which uses your script `webvisualization_plots.py` from 6.1 to generate a plot of "Daily new confirmed COVID-19 cases per million people" by date and display it on the web page.

For the displayed plot, you are free to choose the default time frame.

An outline for your webapp can be found in `webapp_outline.py`. We also added an html template you can use in the directory `templates` named `plot_reported_cases_per_million.html`.

Files Required in this Subtask

- `webvisualization.py`
- `webvisualization_plots.py`
- `plot_reported_cases_per_million.html`

6.3: Interactive Visualization: Upgrading to Pro-Level (10 points)

Modify your solution in 6.2 so that the visitor of the web page can change the countries shown in the plot using a drop down menu or a checkbox menu. The menu should allow the user to add the different countries to the interactive plot.

The user should furthermore be able to interactively change the time range, e.g. start and end, of the interactive plot.

To add the interactive checkbox menu, you can add the countries as query parameters, e.g. in form of a comma-separated list. The query is the set of key-value pairs that go after the `?` in a URL, separated by & characters.

For the countries, start and end parameter this could look like this:

```
@app.get("/plot_reported_cases_per_million.json")
def plot_reported_cases_per_million_json(countries: Optional[str] =
    None, start: Optional[str] = None, end: Optional[str] = None):
  # YOUR CODE
  fig = plot_reported_cases_per_million(countries, start, end)
  return fig.to_dict()
```

We encourage you to have another look at the `altair_web_demo` example in the course repo, which shows how to add checkboxes.

Files Required in this Subtask

- `webvisualization.py`
- `webvisualization_plots.py`
- `plot_reported_cases_per_million.html`

6.4: Implement the Weekly Dropdown (10 points IN4110 ONLY)

Extend your script from 6.3 so that the user can choose using a "slider", "dropdown"-menu, or the interactive tool of your choice, between the "daily" illustration of reported cases you already created or the 7-day rolling average as in

`https://ourworldindata.org/covid-cases`

The 7-day rolling average takes the values of the surrounding 7 days (either 3 days prior to 3 days after or 6 days prior), adds them up, and divides them by 7. To calculate the 7-day rolling average from you daily data, you can use for example the pandas function `pandas.DataFrame.rolling()`. The documentation can be found here. `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rolling.html`

In the next step, you can apply for example the `mean()` function on the 7-day rolling windows.

Altair's `transform_window` can also be used for rolling averages.

`https://altair-viz.github.io/user_guide/transform/window.html`

Generate the same type of layout and interactive visualization as done in 6.3. This also includes that the interval (aka time range) to be plotted should also be able to be specified for the 7-day rolling average. Take into consideration that you will not be able to calculate the rolling average for the first 6 days in the datasets and hence might need to exclude them from your plot.

Files Required in this Subtask

- `webvisualization.py`
- `webvisualization_plots.py`
- `plot_reported_cases_per_million.html`

6.5: Documentation and Help Page (5 points)

Extend your web app with a help-page, and add a link to this on your plot page. Furthermore, add a link to the FastAPI docs, that are created automatically. One option to solve this could be by adding a navigation bar like the one shown below.

Figure 3: Example Navigationbar.

In the html file the code snippet for the navigation bar could look like this:

```
<!-- Add Navigation Bar. -->
  <div class="topnav">
    <a class="active" href="/">Home</a>
    <a href="/docs">FastAPI Docs</a>
    <a href="/help">Help</a>
  </div>
```

The help page should display help for your functions implemented to generate the plot shown. The help page can be generated generated automatically from docstrings using your favorite tool for this. Options here include pydoc or Sphinx[4].

We recommend watching this short lecture video, where we walk you through creating the automated documentation. `https://www.youtube.com/watch?v=hh-5orX1Zt4`

Using Sphinx, you can generate what is called a "static website" (make html). FastAPI can then be used to serve the "static files". Together, these can be combined to serve static HTML pages from a directory.

Files Required in this Subtask

- `webvisualization.py`
- `.html` file for your help page. Naming depends on the tool of choice you used for creation.

[4] `https://docs.readthedocs.io/en/stable/intro/getting-started-with-sphinx.html`

6.6: Upgrading your App to the Next Level (6 bonus points for ALL)

Add a cumulative cases option as a dropdown menu to your plot (in addition to the daily (and 7-day rolling average for IN4110) options. Furthermore, allow for the user to "hover" over the data displaying a "tooltip" box containing the date picked and either the `New Cases: Number` or the `Cumulative Cases: Number`, depending on the dropdown option chosen. The size, color and font of the box and its content is up to your judgement of best readability and appearance.

We recommend having a look at the altair example for creating the box.
`https://altair-viz.github.io/gallery/scatter_tooltips.html`

6.7: Did Someone Say More Bonus Points?! - Creating an Interactive Plot of the Climate Status (10 bonus points for ALL)

In this task you will make an interactive Visualization using Pandas and FastAPI to reproduce the interactive graphic about the global average temperature in 2021 as shown here in "Global snittemperatur så langt i 2021".

`https://www.nrk.no/klima/status/`

The data needed to create the plot can be found on the following websites. You will work with the combined mean surface temperature. The first link will provide you with the data for the anomalies recorded, which can be downloaded in form of a `.csv` file. `https://www.ncdc.noaa.gov/cag/global/time-series/globe/land_ocean/all/1/1880-2020`

The second link takes you to a website providing you with the mean monthly temperature for surface, water, and surface and water combined. `https://www.ncdc.noaa.gov/cag/global/time-series/globe/land_ocean/all/1/1880-2020`

By taking the sum of the anomalies and the monthly mean temperature, you can calculate the mean temperature for each month for the different years.

For each year, plot the average temperature you calculated over the months of the year as a line plot. The graphs for different years should be made in a single plot, with this years data highlighted bold. Feel free to re-use or adapt functions created in earlier tasks. The end-product should be an interactive visualization using Pandas and FastAPI.

Allow for the user to "hover" over the line plot displaying a "tooltip" box containing the month they are hovering over, the coldest year (1904) and the corresponding temperature for that month. The "tooltip" should also contain the temperature measured for that month this year, as well as the warmest year (2016) measured.

Files Required in this Subtask

- `climate_status_plots.py`
- `climate_status_app.py`
- `climate_status_plots.html`

Clarifications

- If a later exercise asks you to add functionality to a previous exercise, you do not have to submit both the old version and the updated version - it is fine to just submit the updated version.

Pat yourself on the shoulder - you finished the last big assignment! Gratulerer!