Mandatory Assignment 6

Web Programming and Data Analysis (30 + 10 points/40 +10 points)

University of Oslo - IN3110/IN4110

Fall 2020

Your solution to this mandatory assignment needs to be placed in the directory `assignment6` in your Github repository. The repository **must** contain a `README.md` file with information on how to run your scripts, required dependencies and packages (versions the code ran with) as well as how to install them. Documentation on how to run your examples[1] 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[2]. However, you are free to choose your own docstring style. We expect your code to be well formatted and readable. Coding style and documentation will be part of the point evaluation for **all tasks** in this assignment.

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.

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 Folkehelseinstituttet (FHI). The data can be found on the following website `https://www.`

[1] Not only important to get all of the credits, but it is a really important habit we need to establish and should never underestimate :)

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

`fhi.no/sv/smittsomme-sykdommer/corona/dags--og-ukerapporter/dags--og-ukerapporter-om-koronavirus/.`

We will be working with the dataset "Reported cases" ("Antall meldte"). Our plot will be based on the **daily** reports. You can download the data as `.csv` for example by clicking on the three bars and **Download CSV**. We attached a screenshot. You will need to do this for all 11 counties separately and "All counties".

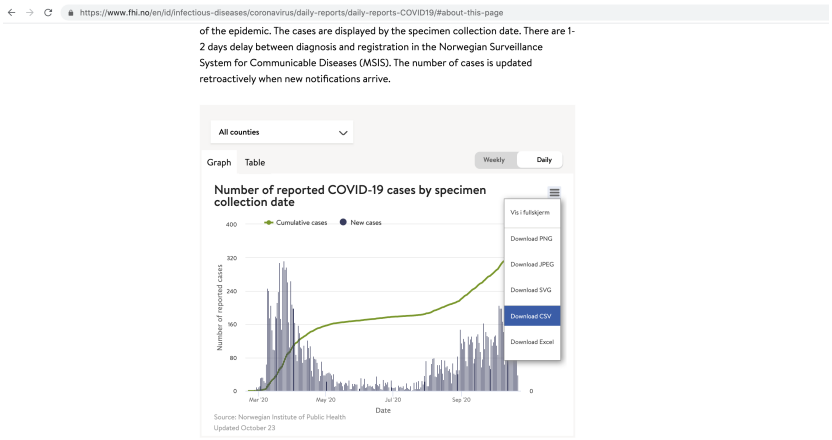

Figure 1: Reported Cases Figure with "Download CSV" highlighted.

6.1: Cases Over Time Plotter (10 points)

Build a Python script which reads data from a `.csv` file and generates a labeled,[3] nice plot of date vs. "number of reported cases" or date vs. "cumulative number of cases".

It should implement the methods `plot_reported_cases()` and `plot_cumulative_cases()`. `plot_reported_cases()` should plot a bar plot while `plot_cumulative_cases()` should plot a line plot.

The user of those methods should be able to control at least the following:

- Which county, if not "all counties" to plot the data from - *optional argument*
- the time range to be plotted e.g. Start = "29/03/2020" End = "04/08/2020" - *optional argument* [4]

[3] Imagine you want to show the plotted graph to a friend, that has not worked with that data set. He/she 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.

[4] You are free to choose your own style of the "Start" and "End" Date, as long as you mention it in your README.md.

Note: You should use Pandas in the task as much as you can, for handling the data and plotting.

Make sure your methods take appropriate arguments. The default county should be "all counties". The time range could be given either in string format or as a datetime object - you can decide. The default time-range can be chosen by you. Make sure to mention what your default arguments are set to.

Hint: We encourage you to use Altair for plotting, but you are free to choose your favorite tool yourself.

Files Required in this Subtask

- `web_visualization.py`

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

Continue working on `web_visualization.py`. Create a function `plot_both` displaying `Number of reported cases` and `Cumulative number of cases` in one plot. This means one y-axis (right-hand side) will refer to the `Number of reported cases` and the other y-axis (left-hand side) will refer to the `Cumulative number of cases`. Build a Flask app which uses your adapted script from 6.1 to generate a plot of number of date vs. "number of reported cases" and "cumulative number of cases" (using `plot_both`) as well as the single plots created by `plot_reported_cases()` and `plot_cumulative_cases()` and display them on the web page.

Hint: We encourage you to use Altair for plotting, but you are free to choose your favorite tool yourself.

Files Required in this Subtask

- `web_visualization.py`

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 county chosen using a drop down menu. The drop down menu should allow the user to switch between all 13 options (12 counties and "all Counties"). Furthermore, allow for the user to "hover" over the data displaying a "pop-up" box containing:

- `date`
- you implement and display the boxes with one of the following options:

  - **(A)** make a pop-up showing the date picked and both `New Cases: Number` and `Cumulative Cases: Number`

- **(B)** make a pop-up showing the `New Cases: Number`, if hovering over the bar plot area representing the "Number of reported cases" or the `Cumulative Cases: Number`, if hovering over the line plot area representing the "Cumulative number of cases". **Disclaimer:** It is a bit trickier to get the pop-up to work independently and exactly if choosing to different pop-ups, one for the line plot and one for the bar-plot. But if you like the challenge - Go for it.

The size, color and font of the box and its content is up to your judgement of best readability and appearance.

Files Required in this Subtask

- `web_visualization.py`

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" or "dropdown"-menu between the "weekly" and "daily" illustration of reported cases as on the FHI website.

Therefore, you will need to download the additional data from the same FHI website `https://www.fhi.no/sv/smittsomme-sykdommer/corona/dags--og-ukerapporter/dags--og-ukerapporter-om-koronavirus/`.

Now you will download the **weekly** reports. You can download the data as `.csv` for example by clicking on the three bars and **Download CSV**. Generate the same type of layout and interactive visualization as done in 6.4. This also includes that the interval aka time range to be plotted should also be able to be specified for the weekly presentation.

Files Required in this Subtask

- `web_visualization.py`

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. The help page should display help for your methods 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[5].

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

6.6: Did Someone Say Bonus Points?! - Creating an Interactive Map (10 bonus points)

In this task you will make an interactive Visualization using Pandas and Flask to visualize the reported COVID-19 rate per 100000 inhabitants by county in Norway.

The data can be found on the following website `https://www.fhi.no/sv/smittsomme-sykdommer/corona/dags--og-ukerapporter/dags--og-ukerapporter-om-koronavirus/`.

We will be working with the dataset "By County" ("Fylkesoversikt"). Our visualization will be based on the plot on the right-hand side referring to **reported COVID-19 rate per 100000 inhabitants by county**. You can download the data as `.csv` for example by clicking on the three bars and **Download CSV**. We attached a screenshot.

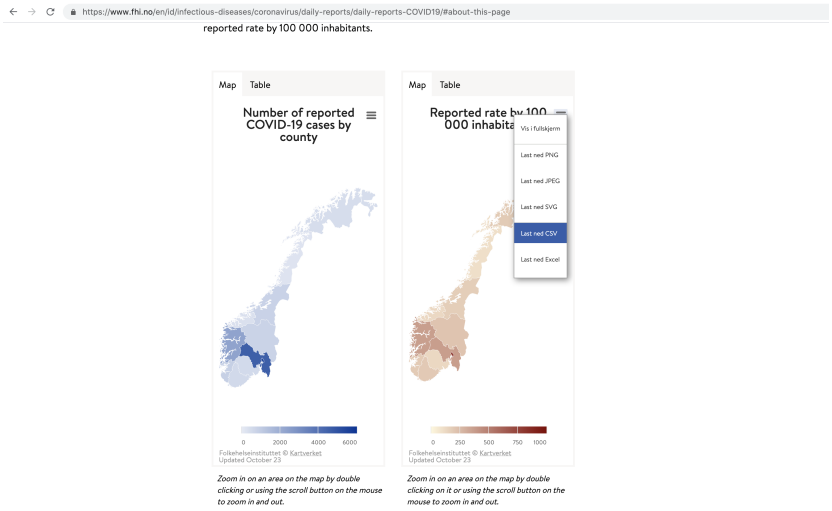

Figure 2: COVID-19 rate per 100000 inhabitants by county with "Download CSV" highlighted.

Feel free to re-use or adapt functions created in earlier tasks. The end-product should be an interactive visualization using pandas and flask, which creates a plot in form of the counties in Norway as seen on the FHI website. The plot should be created by a function called `plot_norway()`.

Your plot should have a legend responding to the color-coding of the reported rates in the different counties. Allow for the user to "hover" over the map displaying a "pop-up" box containing the `name of the county` they are hovering over and the `incidence rate in the county`.

You can choose how you plot the data over the interactive map. We recommend using `topoJSON`, which is a file format encoding topology. A `topoJSON` file

of Norway and its counties can be found here: `https://raw.githubusercontent.com/deldersveld/topojson/master/countries/norway/norway-new-counties.json`. It can be extracted using the altair function `topo_feature()`[6].

Files Required in this Subtask

- `web_visualization.py`

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.

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

[6] `https://altair-viz.github.io/user_guide/generated/api/altair.topo_feature.html`