Data Visualization Concepts

BINF4234

Prof. Dr. Renato Pajarola

Exercise and Homework Completion Requirements

1. Exercises and reading assignments are **mandatory** and they must be completed successfully to finish the class and get a sufficient passing final grade.
2. Exercises are graded coarsely into categories **pass** or **fail***.*
   - *A **fail** is given to failed submissions and incomplete solutions, and no points are awarded.*
   - *A **pass** indicates that the exercise is sufficiently good to receive the corresponding points.*
   - *Late submissions (up to one day) will result in "-1" point.*
3. The five exercises give rise to the following point distribution: 2 – 3 – 5 – 5.
   - *A **minimum of 7 points** from all four exercises must be achieved to pass the module. Failure to achieve this minimum will result in a failing grade for the entire module.*
   - *Thus at least two exercises have to be correctly solved, and one has to be from the more advanced ones.*
4. We give **bonus points** for students who have completed more than 8 points from all the exercises.
   - *Thus **7 points** from the exercises is required, **8 points** is still normal passing, and **9 and above** would give 1 or more extra points.*
   - *Only the bonus points can and will be added directly to the final grade.*
5. Do not copy assignments, tools to detect copying and plagiarism will be used.
   - *The exercise results are an integral part of the final course grade and therefore the handed in attempts and solutions to the exercises **must be your personal work**.*

Submission Rules

- Please hand in your solutions in a .zip archive which contains: **dvc_ex4.py**, **dvc_ex4.html** (screenshot is acceptable), and readme.txt if needed.

  Name the zip folder as *dvc_ex4_MATRIKELNUMBER.zip* (e.g. dvc_ex4_01234567.zip)
- Submitted code must compile and run without errors using the indicated Python environment, using the included libraries, packages and frameworks. If additional libraries/packages are needed, please specify in your 'readme.txt' file.
- The whole project source code must be submitted before the given deadline.

- **Deadline is Thursday, 17 December 2020 at 23:59h**

Exercise 4

For this assignment, we will explore how to create an interactive map visualization based on statistics data of Switzerland. The visualization mainly consists of two layers, a map as the base layer, and circles encoding daily new cases per capita as the top layer. There are four data sources involved to provide different information, three of them should be accessible online, and the geographical shape file is provided as “data.zip” file. Please do **NOT** submit the “data.zip” file or data folder with your solution, you can assume that it is at the same directory as your code.

Task1: Data Preprocessing.

T1.1: Read and filter data. In order to read geometry data, you need first install package “geopandas” into your environment. Filter and extract relevant data as described in the code skeleton.

T1.2: Merge data and build a GeoJSONDataSource. With all necessary data prepared, now you can merge them into one dataframe, in which each row contains information about a canton, and create a GeoJSONDataSource from the merged dataframe.

Task2: Data Visualization.

T2.1: Form two linear color mappers from attributes “Density” and “BedsPerCapita” respectively.

T2.2: Draw a Switzerland map using patches, and add a color bar to the plot. The color of each canton encodes the “Density” by default. When hovering over one canton, it should display canton, density, beds per capita, and daily new cases per capita in the hovertool.

T2.3: Add circles at the locations of capital cities for each canton, and the sizes are proportional to daily new cases per capita.

T2.4: Create a radio button group with labels ‘Density’ and ‘BedsPerCapita’. Clicking on one button will trigger to update map displaying corresponding information, and the color bar will be changed as well. Note that the circles will remain the same for both Density map and BedsPerCapita map.

T2.5: Add a date slider to control which per capita daily new cases information to show and complete the callback function.

T2.6: Add a play button to change slider value and update the map plot dynamically. It is recommended to play the plot backwards.

Remarks:

- In general, the code skeleton is well structured and divided into groups based on the tasks. However, you may want to change the structure of the skeleton for readability reasons of your own code.
- We recommend to use Jupyter Notebook for your implementation as it can visualize the intermediate output which helps for debugging. However, the final delivery of your code should be .py file rather than .ipynb.
- Try to make good use of the hints and references provided in the skeleton code. (**very important**)
- Try to google first for any Python related issues/bugs.
- Due to the special situation, we don’t arrange in person meeting in this semester. Please contact the TA **Fan Feng (fan.feng@uzh.ch)** for technical questions regarding the exercise only if needed.
- More than one day late submission will not be accepted and graded.

- The deliverables of this exercise will be a clean version of your code with proper comments, a "readme.txt" file for your comments or remarks (if necessary), and an export of the final output result in .html or .jpg/.png format. The absence of any required deliverable files will automatically lead to a **FAIL**.

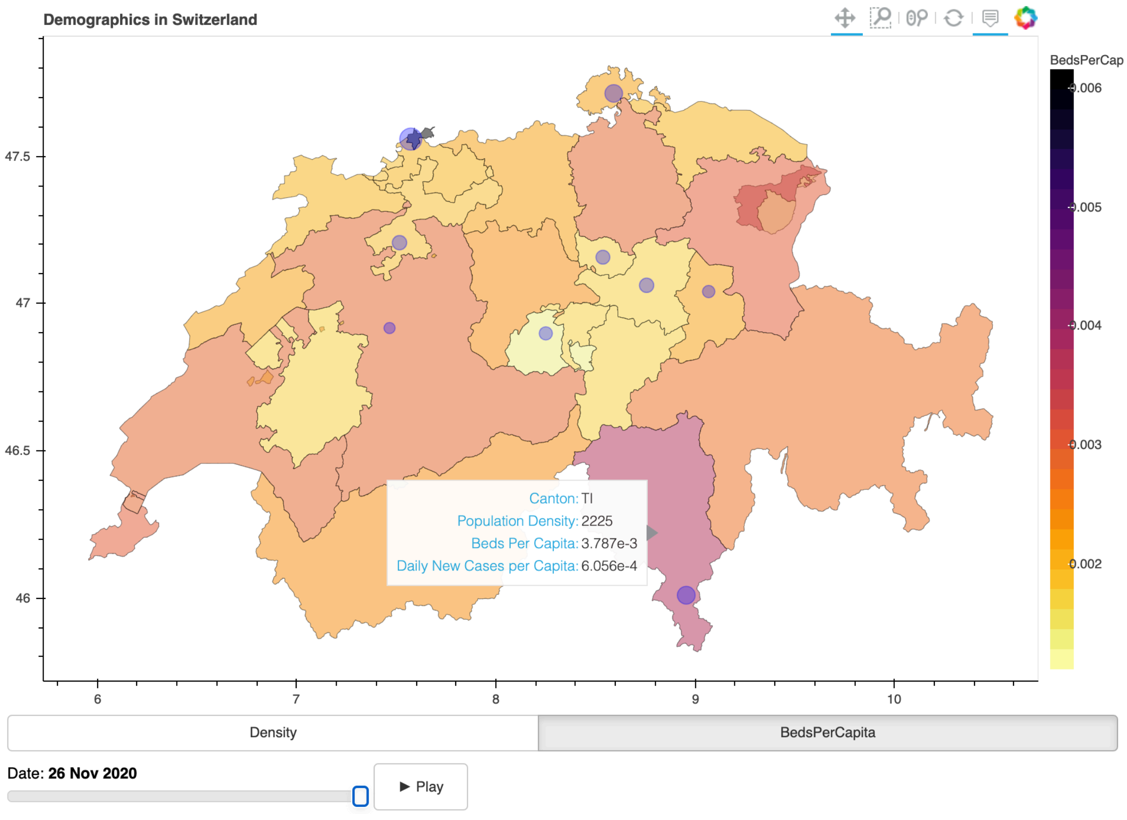

Figure 1: Density map

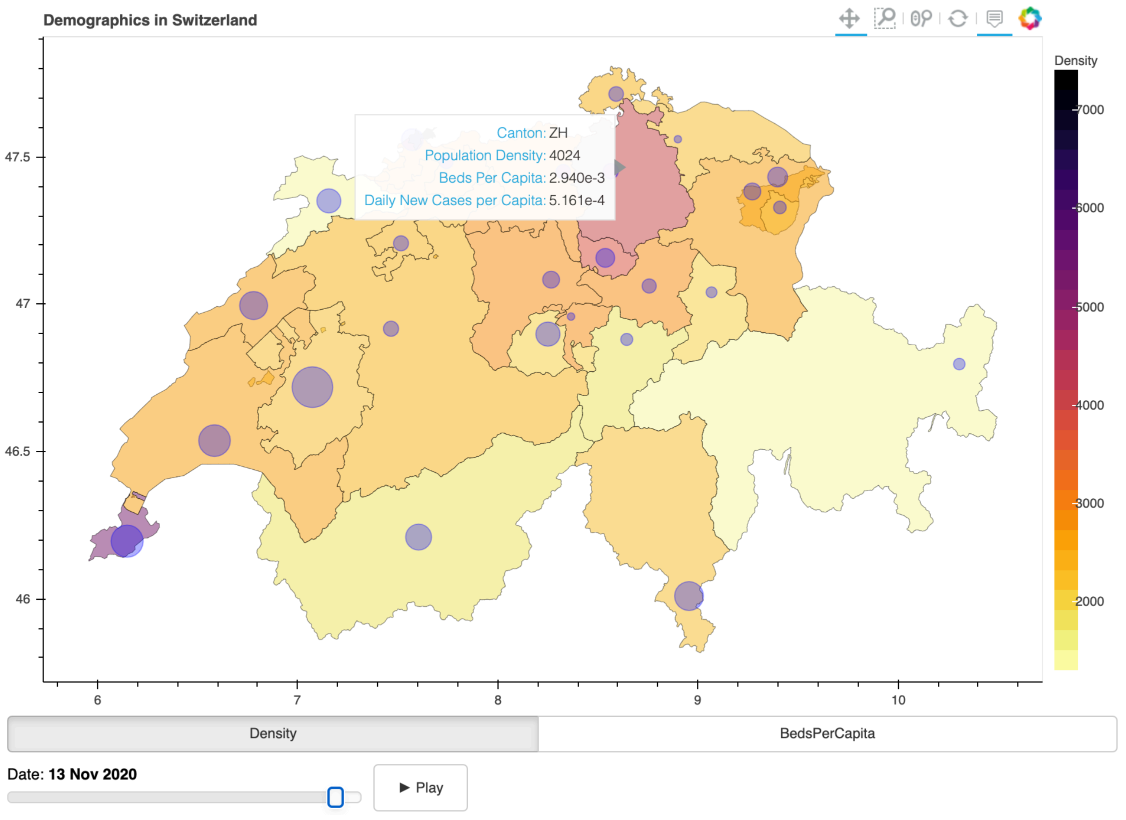

Figure 2: Beds per Capita