Data Visualization & More Tools Exercises

June 26, 2020

Before you start:

- create a suitable directory for this exercise
- download the lecture material from `http://www.physik.uzh.ch/~python/python/lecture_visual/` for reference.

Exercise 1: Replicating a plot

Your predeccessor has created the graph shown below for some publication. Since then, some things have changed and you need to reproduce the plot with new data. However, the code that produced it was lost (it only ever existed in an interactive `ipython` session, the computer on which it was run has long been replaced and nobody bothered with using `git`). In addition, the person who created the plot is no longer in academia and cannot be reached.

From looking at the plot you notice multiple features:

- There are two subplots (`plt.subplot` or `plt.subplots`?). The lower one is smaller (`gridspec_kw`?) and the two subplots share an x-axis.
- The top subplot shows both the PDF and a normalized histogram of $N = 1000$ randomly generated values of (presumably) a normal distribution (`scipy.stats.norm`). It has a meaningful title.
- The top plot also contains the corresponding CDF on a separate y-axis (`ax.twinx`). It is a different color than the other plots.
- The second plot contains the residual between the PDF and the histogram, using `plt.step` in order to match the binning of the histogram. Only this second plot has x-axis ticks and label.
- The overall plot style is not the default one. Hopefully a preset style was used, but which one? Use `plt.style.available` to see which are available.
- The top plot contains a grid matching the right y-axis and the bottom plot has a grid matching both the x- and y-axis.
- In addition, the number of events is added to the plot (`plt.text`) as well as a legend in the upper left corner, which even has a title It only contains the label for the histogram and the PDF.
- The axis are all properly labeled. The x-axis even has a fancy LaTeXlabel.

Try to replicate the plot as closely as possible.

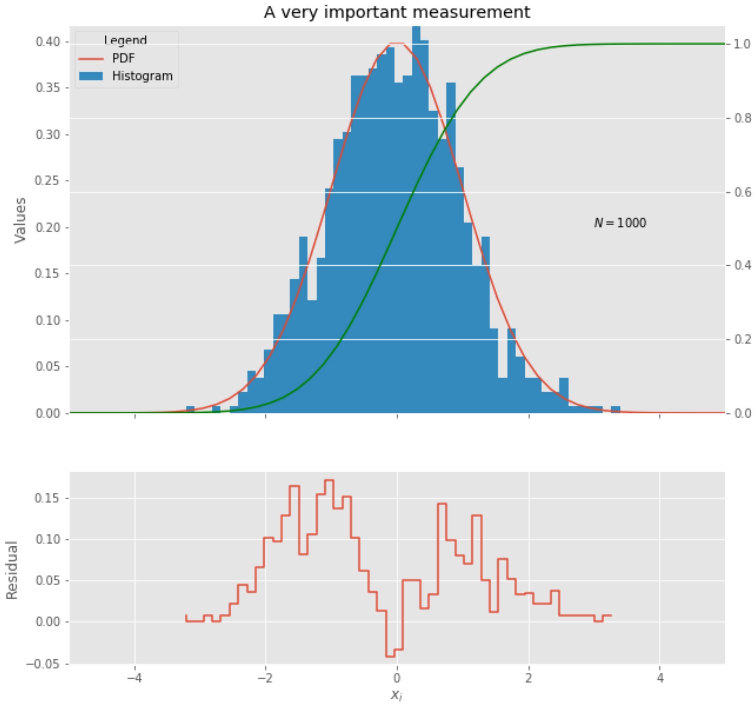

Figure 1: Plot to be replicated in Exercise 1.

Exercise 2: Plotting geospatial data

Use `geopandas` to create an interesting visualization of data on a map.

- The file `london.zip` contains two files obtained from data.london.gov.uk, the `London_Borough_Excluding_MHW.shp` file contains the shapes of the London boroughs and the `london-borough-profiles.csv` files contains a lot of data on each borough.
- Load the two datasets (the second one needs to be opened with `encoding=’iso-8859-1’`) into dataframes. Think about how to best combine the two dataframes.
- Use the `plot` method of the `geopandas.DataFrame` to plot some statistics per borough. An example of what this can look like can be found in Fig. 2.

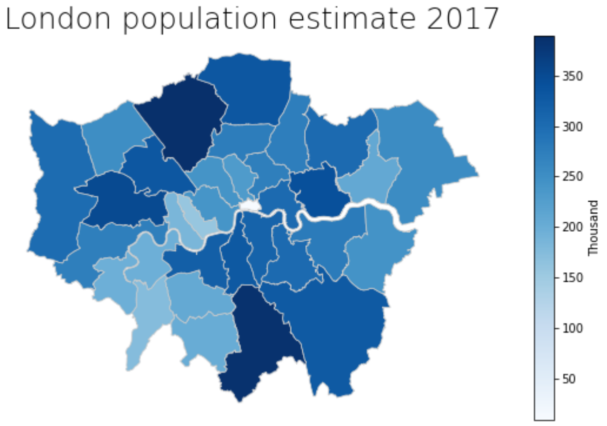

Figure 2: Example plot for Exercise 2.

Exercise 3: Web scraping

Use the `requests` library and `bs4.BeautifulSoup` to parse our homepage for the material of each lecture[1].

- You can use the developer tools of your browser to figure out the names and attributes of elements.
- CSS selectors can be used with `soup.select`, or you can directly operate on the tags with `soup.find/soup.find_all`.
- Write a `download` function that automatically downloads the material of one lecture to a specified directory. For this use `response.content`, instead of `response.text`, and `open(file_name, "wb")` in order to directly write binary content (such as PDF files) to a file. Make sure to create the directory if it does not exist and that you deal with file names which would be illegal (for example file names containing `"/"`).
- Extend this to automatically download the material of all lectures.

[1] The after-school email will contain a link to a zip file containing all material, so don’t worry if you don’t do this exercise