Visualizing COVID-19 Data Beautifully in Python

Making Matplotlib a Little Less Painful!

Nik Piepenbreier (Apr 5)

Matplotlib may be the de facto data visualization library for Python, but it's not always the prettiest. In this post, we'll explore how to turn a drab, default Matplotlib graph into a beautiful data visualization. We'll explore COVID-19 data to see how the virus has spread throughout different countries.

Let's Load in Our Data

We'll be using data from this wonderful Github repository that auto-updates the data daily. We'll load our data into a Pandas' dataframe based on the URL so that it'll update automatically for us every day.

MOSTRAR CÓDIGO

	Confirmed	Recovered	Deaths	Cases	Mortality
count	664.000000	664.000000	664.000000	664.000000	549.000000
mean	30296.310241	8240.198795	1630.530120	40167.039157	2.208281
std	67603.850987	19291.278607	3892.664256	82592.981719	2.702718
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	3.000000	0.000000	0.000000	3.000000	0.000000
50%	287.500000	12.000000	3.000000	301.500000	1.580742
75%	35988.250000	1315.250000	1059.000000	40430.250000	2.769535
max	580619.000000	78039.000000	23529.000000	647630.000000	11.210124

- In **Section 1** of the Gist above, we're loading our libraries. We'll be making use of Pandas and Matplotlib for this tutorial.
- In **Section 2**, we read in the data into a dataframe df, and then select only the countries in our list countries. Selecting the data makes the resulting visualization a little more readable. https://github.com/datasets/covid-19/tree/master/data
- In **Section 3**, we create a summary column that aggregates the total number of cases across our confirmed cases, recovered cases, and any individuals who have died as a result of COVID-19.

Preparing our Dataframes for Data Visualization

Now that we have our data stored within a dataframe, let's prepare two further dataframes that will hold our data in crosstabs, which will allow us to more easily visualize the data.

MOSTRAR CÓDIGO

	Argentina	Chile	China	Germany	Italy	Spain	US	United Kingdom
Date								
2020-04-09	3.225806	0.780501	2.037205	1.505240	9.601576	6.994847	3.273829	10.768754
2020-04-10	3.371711	0.798820	2.035692	1.547098	9.573803	6.991070	3.417103	10.662136
2020-04-11	3.322658	0.823556	2.035510	1.478567	9.530383	6.955626	3.539521	10.943931
2020-04-12	3.333333	0.855432	2.033047	1.580742	9.454419	6.983293	3.607937	11.018961
2020-04-13	3.439716	0.822138	2.032236	1.616675	9.500223	7.029796	3.633093	11.210124

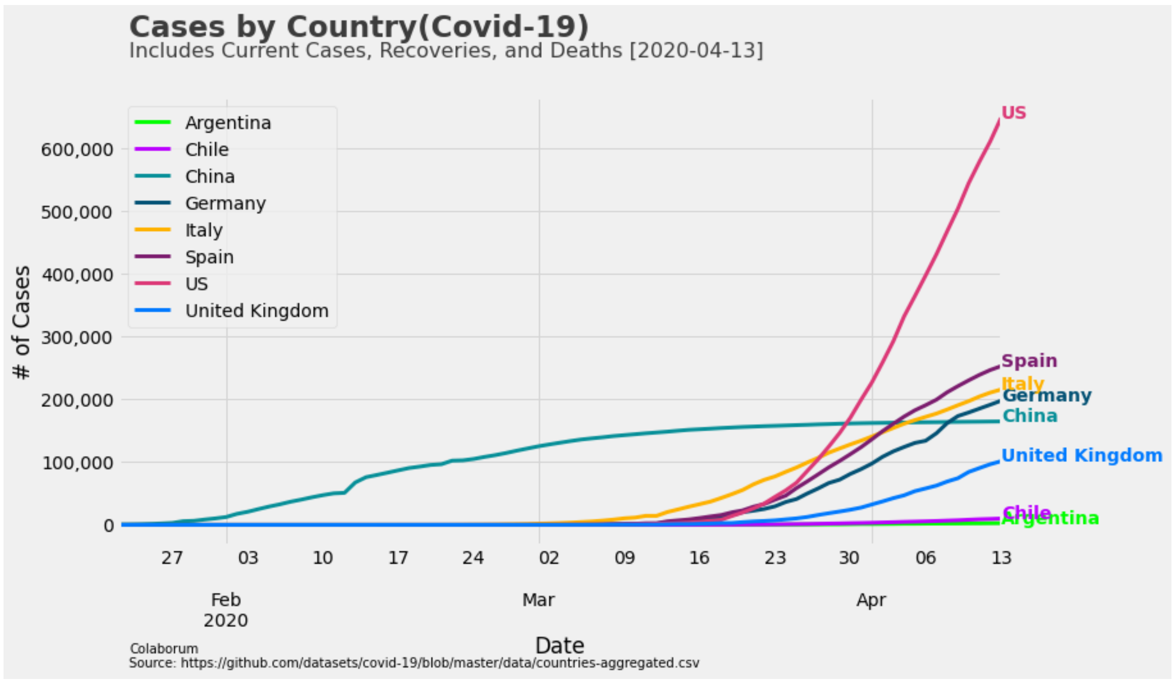
Let's explore what we did here in a bit of detail:

- In **Section 4**, we pivot our dataframe df, creating columns out of countries, with the number of cases as the data fields. This new dataframe is called covid. We then set the index of the dataframe to be the date and assign the country names to column headers.
- In **Section 5**, we copy our dataframe covid and call it percapita. We use a dictionary that is storing all our countries' populations and divide each value by the population and multiply it by 100,000 to generate a number of cases per 100,000 people. https://www.worldometers.info/world-population/population-by-country/

Creating our First Visualization — Cases over Time

Let's begin by creating our first visualization that will demonstrate the number of total cases over time in various countries:

MOSTRAR CÓDIGO



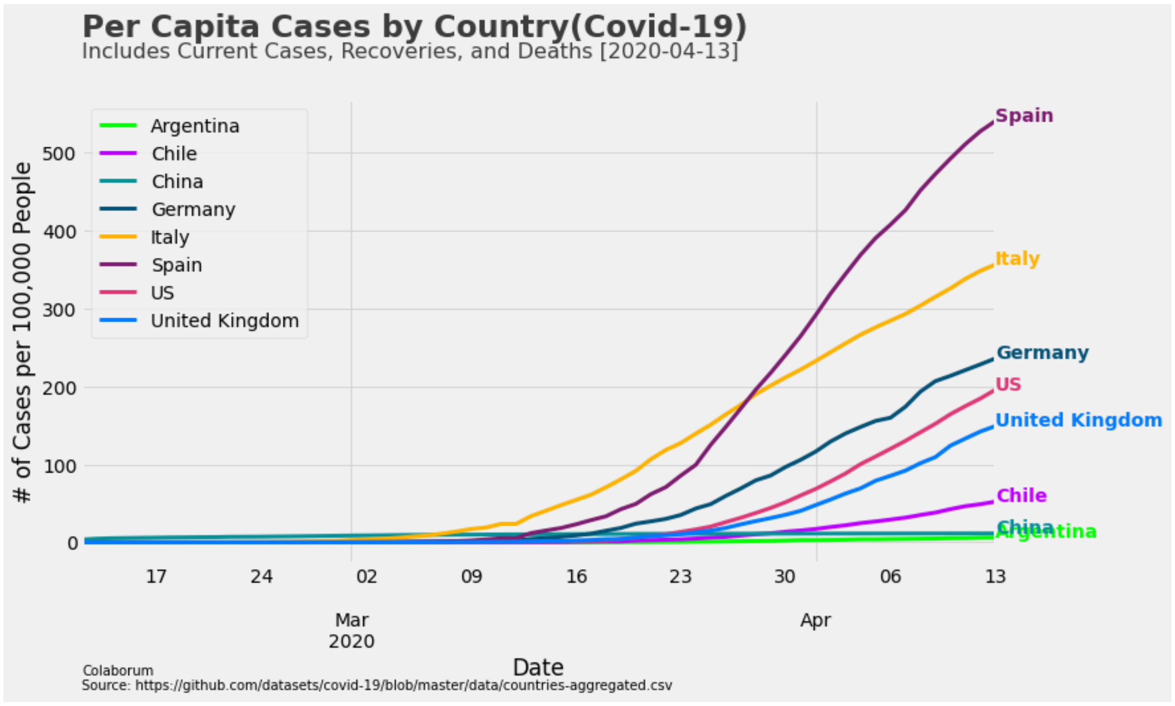
Let's explore what we did her in a bit more detail:

- In **Section 6**, we created a dictionary that contains hex values for different countries. Storing this in a dictionary will allow us to easily call it later in a for-loop. We also assign the FiveThirtyEight style to add some general formatting, which we'll heavily build upon.
- In **Section 7**, we create our first visualization using Pandas' plot function. We use the colors parameter to assign the colors to different columns. We also use the set_major_formatter method to format values with separators for thousands.
- Then, in **Section 8**, we create a for-loop that generates label text for the various countries. This for-loop gets each country's name from the keys in the dictionary in the form of a list and iterates over this list. It places text containing the country's name to the right of the last x-value (covid.index[-1] → the last date in the dataframe), at the current day's y-value (which will always be equal to the max value of that column).
- Finally, in **Section 9**, we add a title, subtitle, and source information about the chart. We use variables again to position the data so as the graph updates these positions are updated dynamically!

Creating our Second Visualization — Cases per 100,000 People

To create our second visualization, we'll make use of the code below:

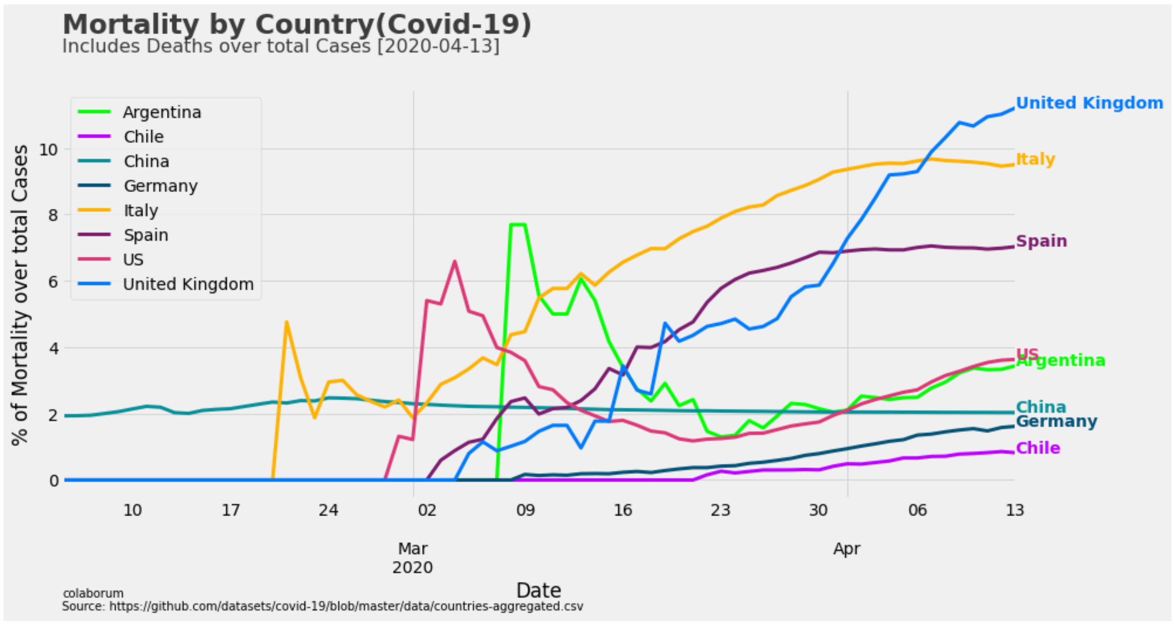
MOSTRAR CÓDIGO



Creating our Tirth Visualization — Mortality

To create our Tirth visualization, we'll make use of the code below:

MOSTRAR CÓDIGO



Conclusion: Beautiful COVID Visualizations with Matplotlib