COVID-19 Visualization and Analysis

PSDS Capstone, 19 January 2021

Kathleen Smith

Project Overview

For my capstone project, I created an R shiny app to display a 3d plot of live coronavirus data, updated daily from Our World in Data [1]. I supplemented the data with a handful of extra features gleaned from the CIA World Factbook [2]. After completing this visualization, I explored some relationships within the supplementary data, and started analysis of Coronavirus Government Response Data [4] and the impact of government responses on the spread of coronavirus.

The project and data are all unclassified/open source. All code is available on github at https://github.com/kes256/PSDS_Capstone, and the shiny app is available at https://kes256.shinyapps.io/capstone_basic. Update on 19 Jan 2021: shinyapps.io no longer able to install python dependencies, and app does not run. This is likely because python 3.5 (version in the shinyapps.io environment) is no longer supported. Shiny app code on github has been updated to run locally.

Data Carpentry and Exploration

All scripts used for data carpentry are in the github repository in the carpentry/ folder, and data is in the same repository in the data/ folder.

CIA World Factbook Data

The CIA World Factbook [2] provides a wealth of information on countries and territories across the world. Fortunately, a JSON api [3] is available (though not officially from the CIA). I downloaded the JSON file to process and use locally, without plans to refresh the data. The JSON data was fairly deeply nested, so my first task was to figure out exactly what data it contained. I wrote a recursive function in python to walk through all of the data, and record the full paths to each datum. Next, I compared which data were available across countries, resulting in a file with the header for each datum, and the proportion of countries (out of 210 countries tracked in the Factbook) for which the datum was present, sorted by proportion in descending order. The script to do this is dev.py, and creates keys.txt as its output. Once I had the list of fields and how well populated they were, I manually went through the output file and saved a list of only the fields that were well-populated (populated for at least 75% of countries, with a few exceptions), and seemed pertinent to the coronavirus pandemic. I saved this list in filtered_keys.txt.

At this point I switched over to R, and loaded the JSON file using R's jsonlite and tidyverse libraries. In particular, the tidyr library's pivot_wider function helped reshaped the data into a dataframe with one line for each country. After inspecting the data in R, a few data types were still problematic, so I went back to python, and wrote another script (key_rewrite.py) to read the JSON file, reformat the data causing issues, and rewrite the JSON file (as clean_factbook.json). After that final step, the R code (factbook.R) could read in the JSON file, tidy up the dataframe, add an ISO3c country code (using R's

countrycode library), and write the resulting dataframe as factbook.csv, ready to be used without further processing.

GeoJSON Map Data

I downloaded a geoJSON map file [7] to generate global country boundaries, based on public domain Natural Earth data. After loading the file into an R dataframe, and tidying with the tidyverse broom library, I used the countrycode library to append an ISO3c country code to each region, to facilitate joining the map with other data to plot. A few regions did not get automatically matched with their ISO code, but countrycode allows matches to be added manually. After prepping, I exported as map.csv to reuse without any other processing.

Coronavirus Case Data

The dataset available from Our World in Data [1] (I chose to ingest the csv format out of several options) did not need much processing. It included ISO3c country codes, so I could join it with the other data sources immediately. Some of the columns had sparse data (ICU beds per capita was only populated for a few countries, for example), but I did not do any carpentry to modify the dataset.

Data Visualization

My first objective was create an interactive visualization of the coronavirus data, with the goal of creating something accessible to non-technical people. Working in R Studio, I made a shiny app template, made an account at shinyapps.io, and deployed the template app, creating a live website. The final results of the shiny app are in the github repository in the capstone/ folder.

ggplot2

Having established proof-of-concept for deploying a live web app, I started to experiment with different visualizations in R. First, I used ggplot2 to plot a global choropleth. Using an input slider to control the date in the shiny app, I made a choropleth of supplemental data, with coronavirus case data over time shown as a bubble plot (sample of this plot available as `fig` after running analysis/map.R from the github repository).

There were a few problems with this bubble plot. The range of sizes displayed was too small to see much difference around the world and the scale for the sizes was recalculated on each date, so watching the animation didn't display the change in cases over time very well. I considered using R's cartogram library (https://www.r-graph-gallery.com/331-basic-cartogram.html) to create a more compelling visualization, but would have been dealing with the same issue of setting the scale for each date. I also explored the rayshader library (https://www.rayshader.com/), which creates 3d plots from ggplot2 objects, among other things. I experimented with using rayshader to display a single variable in 3d, but didn't find a way to incorporate the resulting plot into the shiny app. It was also very slow to render on my work laptop, so creating an interactive plot seemed unlikely to work smoothly.

plotly (R library)

Having come up with few options for creating a 3d plot in ggplot, I switched over to plotting with the plotly library in R. plotly includes options for plotting 3d data, and allows for built-in reactive elements (such as a slider to select the date). I refactored the shiny app to produce a plotly plot with the input

slider as part of the plot rather than a shiny element. Unfortunately, when creating a map in plotly, any input to a z-variable is interpreted as the fill for a choropleth, which makes producing a choropleth easy, but prevents any 3d plotting on top of a map. To make a 3d plot, I resorted to plotting the world map at height 0 on the x-y plane, then added the data I wanted on top of that. I used the coordinates of each country's capital city to plot case data in 3d over the map. I used the z error bar option to add a line dropping down from the data points to the location of each capital city on the map.

At this point, another issue cropped up. Bar graphs are unsupported in plotly animations in R, and as best as I can tell, the error bars fall under that graph type; they did not show up once I set the plot to animate over time (with input slider for the date). However, given that plotly was originally developed in python, more plotly features are supported in python.

plotly (python package)

To create the final version of the R shiny app, I switched again, using the plotly python package to create a 3d plot. Adding the map in the x-y plane took a little more tweaking in python, and I tested out adding a fill to each country in python, but the fill options didn't stay within the non-concave country boundaries. Once I had python code to create a 3d plot over a global map, I used the reticulate library in R to run the python code within R. At this point, I added drop-down menus to allow users to select from several measures in the Our World in Data set to control the height of each point, and from several measures from both Our World in Data and the World Factbook to control the color of each point. Finally, I deployed the app to shinyapps.io. The last hurdle was getting reticulate to work properly in the shinyapps.io environment using .Rprofile, since both python 2 and python 3 were installed. While this worked during the capstone presentation, the app no longer runs in the shinyapps.io environment. The code on github can be run locally, and screenshots of the app are included in the Appendix; see Figure 1, Figure 2, and Figure 3.

shiny app

My shiny app displays a variety of coronavirus data as well as background data on a global map, animated over time. On loading the app, the most recent version of https://covid.ourworldindata.org/data/owid-covid-data.csv is downloaded and filtered down to a weekly basis. Because I filtered down to weekly data without doing any of my own smoothing, I used the smoothed case/death data for new instances. For both cases and deaths, either total numbers or new numbers can be chosen, as well as total counts vs. counts per million population. The UI for selecting the data is a simple dropdown; a more sophisticated version of the app could use different toggles to make the data selection more intuitive. The plotly 3d plot also makes poor use of the space available, leaving a lot of white space around it, while cutting of some of the axis labels. A new site using Dash (plotly's web design software) would potentially be a lot more polished.

Analysis of CIA World Factbook and Human Development Index

Code for this analysis is in the github repository in analysis/CIA_Factbook_Factor_analysis.py.

On a slight tangent from the coronavirus data, I dug into the CIA World Factbook Data to see what I could do with it. First, I calculated the correlations between each feature saved in factbook.csv and the total cases per million for each country (most recent/maximum number). None of the correlations were particularly strong, so I moved on.

Next, I looked into using the Factbook data to predict the Human Development Index (HDI), which is included from Our World in Data. The HDI is calculated as the geometric mean of three components: health (expected life, scaled to range from 0 to 1), wealth (log of per capita income), and wisdom (expected school life, scaled to range from 0 to 1). I started looking at linear correlations for simplicity, even though the HDI is not linearly dependent on any of its inputs. Expected life and expected school life from the Factbook were unsurprisingly well correlated with the HDI, as were literacy and median age. The most interesting find was that the percent of internet users in each country was very strongly linearly correlated with HDI. It isn't terribly surprising that internet use is correlated with the HDI, but I was very surprised by how strong the correlation was.

Timeseries analysis with Coronavirus Government Response Tracker

I started two in-depth analyses of data from the Coronavirus Government Response Tracker [4]. Code for these analyses is in the github repository in analysis/Timeseries Analysis.py and analysis/policy.py.

Our World in Data Timeseries Analysis

In the first analysis, I attempted to develop a method of measuring qualities of a government's coronavirus response. I used the stringency index, produced by the Coronavirus Government Response Tracker, and reported in Our World in Data. Before beginning the analysis, I filtered out countries with population under 2.5 million, removing about one quarter of the countries in the dataset. I did this because I planned to use death and case rates per million population, and didn't want to end up working with numbers skewed by low population. For example, 7 new coronavirus cases in Vatican City on Oct. 15 translated to a case rate 40 times as high as that in the United States, which had over 64,000 cases on the same day.

In order to measure the effectiveness and responsiveness of a government's response, I calculated lagged correlations of the stringency compared to both the rate of new cases per million population, and the reproduction rate, both from Our World in Data. To come up with a measure of responsiveness, I looked at the correlation between stringency and past case measures, recording the highest correlation over lags up to 14 days. To come up with a measure of effectiveness, I looked at the correlation between stringency and future case measures, recording the lowest (negative) correlation over lags up to 14 days.

While these measures are interesting, they still require further refinement. For example, I used the entire history of each country's response to calculate each version of responsiveness and effectiveness, but I think it would make more sense to calculate these measures for smaller time spans, since a government's strategy is likely to change over time; a single measurement wouldn't necessarily represent the government's response as a whole.

However, proceeding with these preliminary measures, I compared each government's responsiveness and effectiveness to its worst death rate over the course of the pandemic. This measure of the impact of the pandemic certainly has value, but also discounts any fluctuation in government policies, their effects, and changing best practices discovered over the course of the pandemic. Given the roughness of the measures I used, it's unsurprising that no compelling patterns emerged at this point in my analysis.

Government Response Tracker policy analysis

In order to refine my analysis of government responses, I downloaded the raw data available from the Coronavirus Government Response Tracker. This new data was more detailed in two respects. First, it included state-level data for the US, Canada, and Brazil, allowing for comparisons between these subregions. At least in the US, each state was responsible for developing its own strategy, so this level of detail makes a lot of sense to me. Second, the data included the individual measurements used to calculate the stringency index, such as ratings on how widely/strictly facial coverings are required, whether large gatherings are restricted, and how much money has been invested in vaccine research.

An additional dataset [5] also contains a risk of openness index, based on case numbers from Our World in Data, several of the stringency measures, and mobility data from Apple and Google used as measures of the WHO's recommendations for re-opening. Finally, a drawback of these data is that they only include cumulative case and death counts, so more work is required to measure coronavirus impact in each region.

I calculated the number of new cases and deaths each day from the cumulative counts, and ran some quick correlations against a few of the new measures, but did not find anything notable. I did not get any further in data exploration due to time.

Challenges and Next Steps

Visualizations

I would like to refine the web app by porting it from a shiny app to something using Dash, which I hope would allow for more control over how the plot is displayed. It would also simplify the app to use a single language, rather than having part of the app in R, and part in python. I'd like to make better use of the space and eliminate a lot of the white space surrounding the plot. Improving the user inputs from the simple drop-downs to would also improve the look of the web app. After polishing the display, more types of plots could be added, to allow users to explore more aspects of the coronavirus pandemic.

Analysis

The analysis of stringency data I did was very brief, with a lot of room for improvement. One step would be to supplement the stringency data with better case data. Rich country level data could be joined from Our World in Data, but state level case data would need to be found from other sources. Finding appropriate sources, normalizing data from each source, and identifying or creating state level keys to join the data together is a non-trivial task.

Once a dataset with satisfactory case data as well as stringency and risk of openness data is put together, I'd be very interested to calculate the responsiveness and effectiveness of government policies over 60 day time periods, looking for patterns of particular types of government policies that result in higher effectiveness, or identifying regions with consistently high effectiveness and/or responsiveness, or looking for more correlations between responsiveness or effectiveness and death rates, to answer the question “Does government response to new cases surging in an area help reduce deaths?”, for example. Depending on how the results at this point, almost any data analysis technique could be on the table.

Appendix

Code repository:

https://github.com/kes256/PSDS_Capstone

Shiny app:

https://kes256.shinyapps.io/capstone_basic

Data sources:

https://github.com/iancoleman/cia_world_factbook_api

https://covid.ourworldindata.org/data/owid-covid-data.csv

https://www.bsg.ox.ac.uk/research/research-projects/coronavirus-government-response-tracker

https://geojson-maps.ash.ms/

List of libraries used (both in development and final code):

Python packages	R libraries
json pandas plotly/plotly_express requests seaborn statsmodels	countrycode geojsonio ggplot2 jsonlite mapproj rayshader reticulate shiny tidyverse – broom, dplyr, lubridate viridis

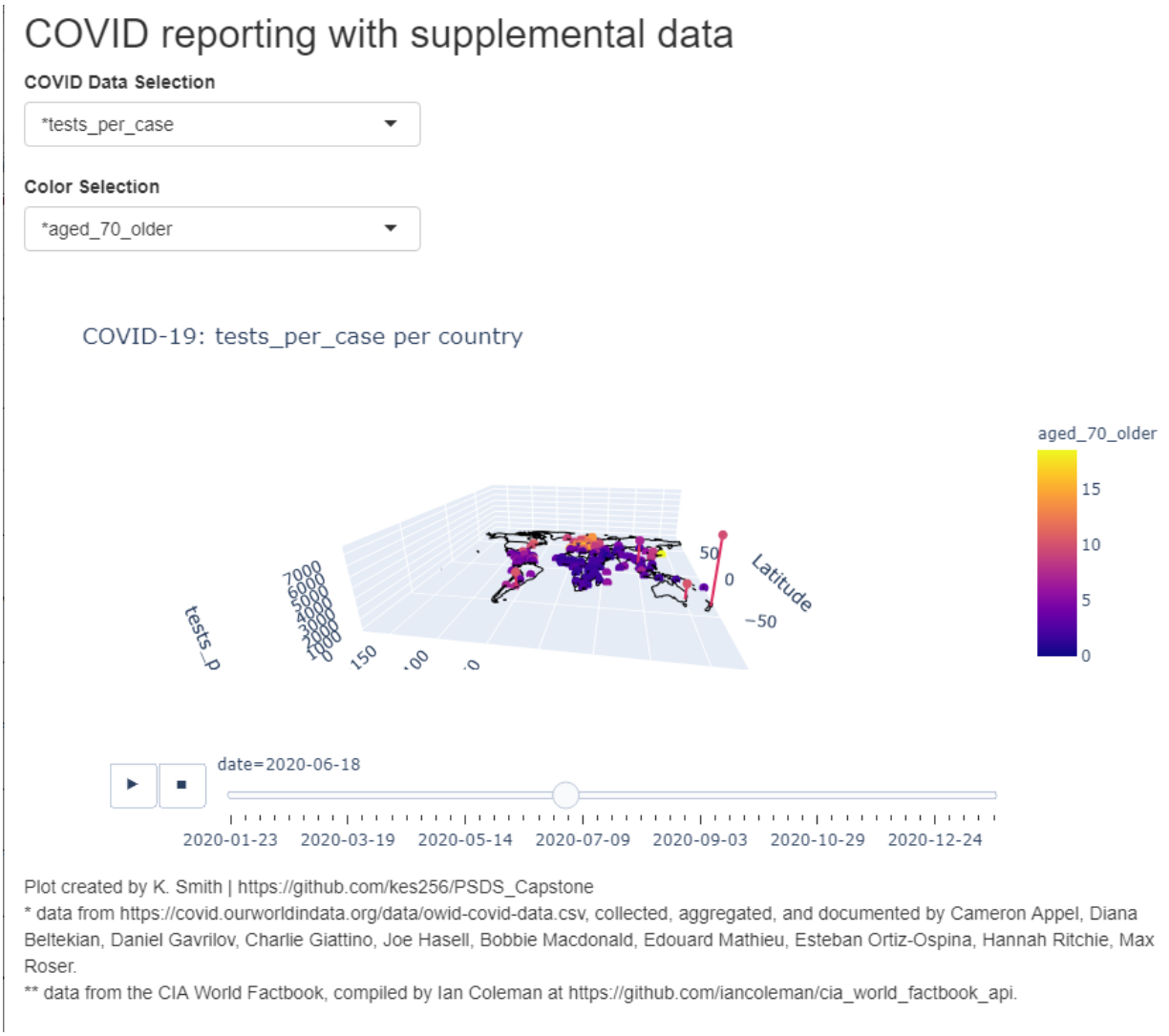

Figure 1: Screenshot of shiny app showing testing rates and percent population over 70

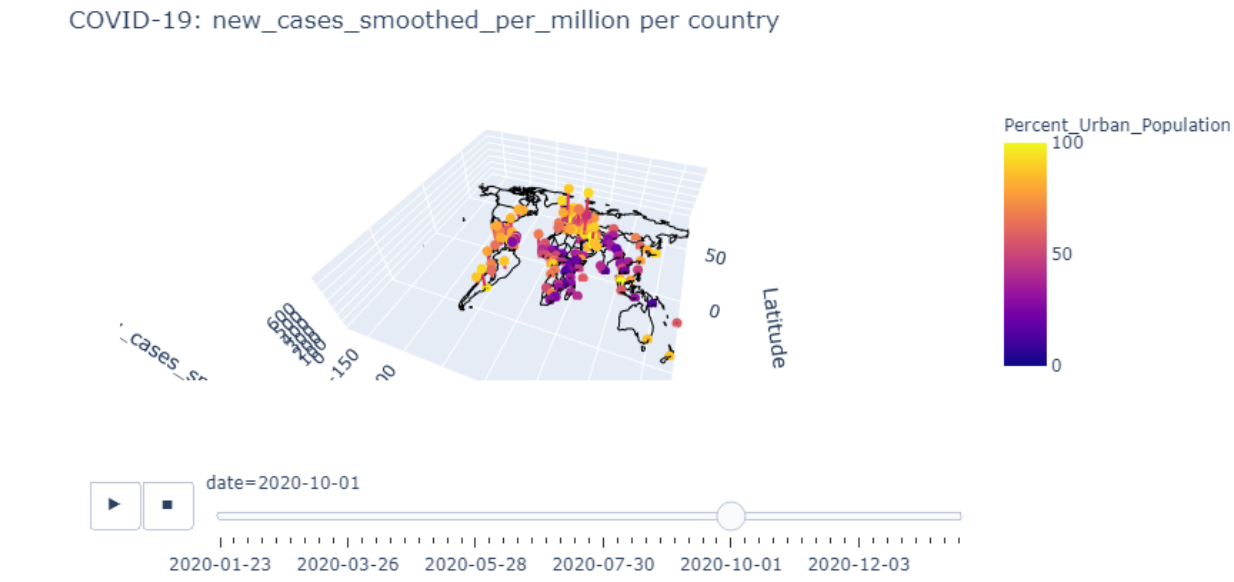

Figure 2: Screenshot of shiny app showing smoothed case rates and percent urban population

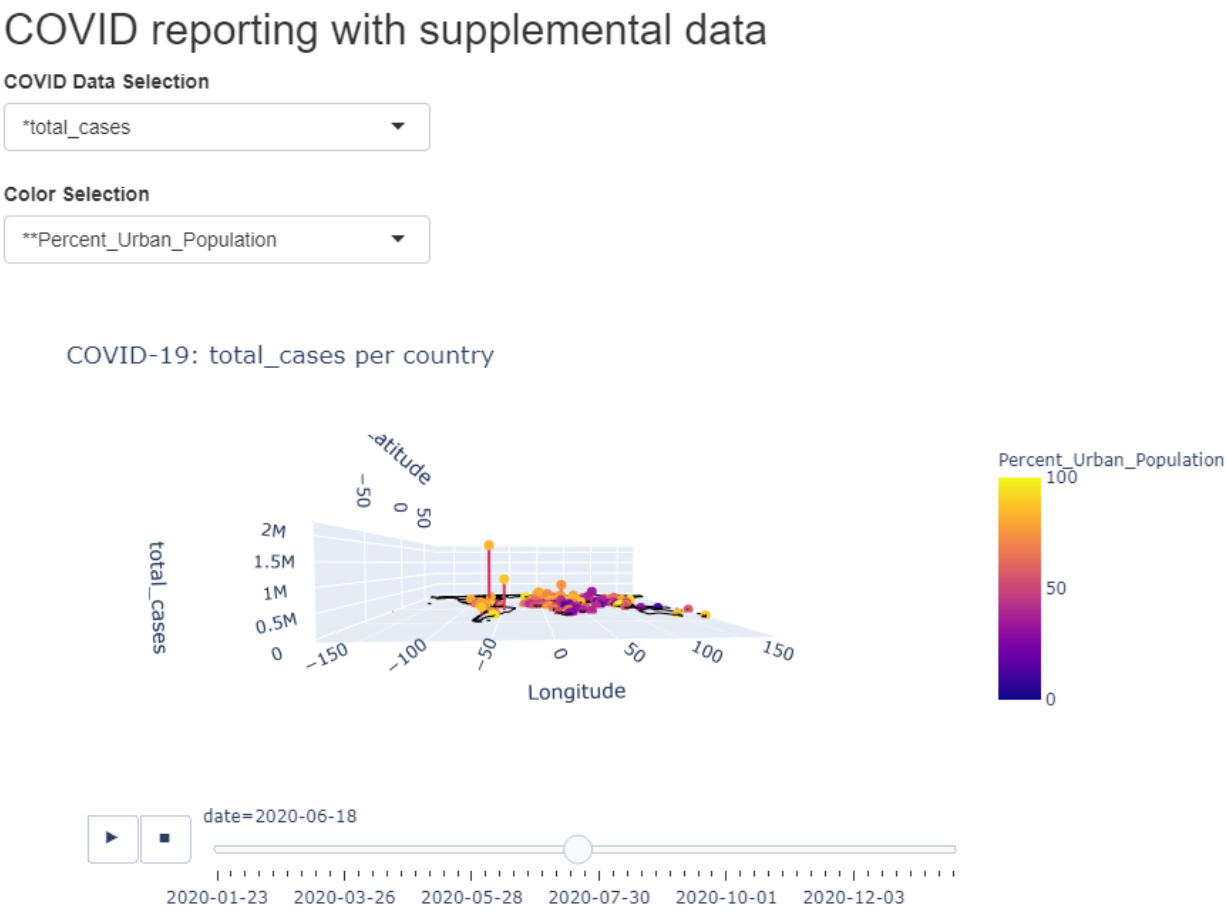

Figure 3: Screenshot of shiny app showing total cases and percent urban population

Bibliography

[1] Cameron Appel, Diana Beltekian, Daniel Gavrilov, Charlie Giattino, Joe Hasell, Bobbie Macdonald, Edouard Mathieu, Esteban Ortiz-Ospina, Hannah Ritchie, Max Roser. Coronavirus data collected, aggregated, and documented at https://ourworldindata.org/coronavirus-data-explorer. Accessed January 2021.

[2] Central Intelligence Agency. (2020). *The World Factbook*. https://www.cia.gov/the-world-factbook/. Accessed November 2020.

[3] Ian Coleman. (2020). CIA World Factbook API GitHub repository. https://github.com/iancoleman/cia_world_factbook_api. Accessed November 2020.

[4] Thomas Hale, Sam Webster, Anna Petherick, Toby Phillips, and Beatriz Kira. (2020). *Oxford COVID-19 Government Response Tracker*. Blavatnik School of Government. Accessed January 2021.

[5] Thomas Hale, Toby Phillips, Anna Petherick, Beatriz Kira, Noam Angrist, Katy Aymar, Sam Webster, Saptarshi Majumdar, Laura Hallas, Helen Tatlow, Emily Cameron-Blake (2020). *Risk of Openness index: When do government responses need to be increased or maintained?* Blavatnik School of Government. Accessed January 2021.

[6] Hasell, J., Mathieu, E., Beltekian, D. et al. *A cross-country database of COVID-19 testing.* Sci Data **7**, 345 (2020). https://doi.org/10.1038/s41597-020-00688-8. Accessed January 2021.

[7] Ash Kyd. (2020). GeoJSON Regions. https://geojson-maps.ash.ms/. Accessed November 2020.