CSC110 Project: A 50 Year Observation of CO2 Emissions by Each Country

Sophia Abolore, Michelle Chernyi, Emily Chang

Monday, December 14, 2020

Problem Description and Research Question

Climate change, the change in global temperatures, has become a greater and greater issue in the world. It is known to cause floods, heatwaves, sea-level rises, and other environmental disasters. (Ritchie et al.). One of the main contributors to climate change is the emission of carbon dioxide from human activities, specifically, the burning of fossil fuels (Ritchie et al.). Fossil fuels are decomposed carbon-based organisms; they are used as a source of energy, supplying about 80% of the world's energy ("Fossil Fuels and Climate Change"). In 2018, 89% of carbon dioxide emissions in the world were caused by the combustion of fossil fuels and industrial activities ("Fossil Fuels and Climate Change"). In Canada, most of the carbon dioxide released into the atmosphere is caused by the burning of fossil fuels, as well ("Greenhouse Gas Sources and Sinks"). Over the past 50 years, carbon dioxide emissions have been increasing exponentially (Ritchie et al.). In order to study, understand, and hopefully improve the issue of climate change, we need to understand the magnitude of which each country contributes to the global CO2 emissions, that way, attempts to reduce those emissions can be tailored to each country. To achieve this we have decided to create a project that answers the question: **how have the co2 emissions from fossil fuels in the world changed over the span of 50 years and how do they compare between each country?** We will examine 50 years (1964-2013) of CO2 emissions for each country and interactively display these findings by year on a map that shows each country as a function of their CO2 emissions. Countries with higher levels of CO2 emissions will be displayed in a darker shade than those with lower CO2 emissions such that the darker the shade of the countries the more emission. In order to provide numerical context for these emissions we will also include a button on the map that once clicked will generate a bar graph of the total CO2 emissions of each country. We hope having visual representations of this data will help contextualize the disparities of emissions between all the countries on earth.

Dataset Description

The first dataset we are using is a shapefile which is provided by geopandas, one of the libraries we are using. It is titled: 'naturalearth_lowres' and is used specifically for creating maps. It provides a population, continent, country, geometry, and a few other columns, which are used to create maps of countries with boundaries.The program uses the dataset as a whole, merging it with our second dataset to adjust the map. The second data set we are using is a csv file from the Carbon Dioxide Information Analysis Centre. The data set has the carbon dioxide emissions from combustion of fossil fuels and cement production. The dataset shows the above data by country and by year from 1751 to 2014. Our project focus is only on the years 1964 to 2013. The data set was changed manually to only include those years. From the remaining data, the program only uses the columns: "Year", "Country", and "Total" (total represents the total emissions, which are measured in million metric tons). The program uses both this version of the data set for graphing (emissions-edited-graph.csv) as well as a manually edited version of the data set for creating a map (emissions-map.csv).The edited version accounts for country name changes throughout the years that will be discussed in the discussion section.

Computational Plan

The function uses libraries discussed in CSC110 such as pygame and plotly, as well as three new libraries: matplotlib, pandas, geopandas, and Pillow. The uses of the new libraries will be discussed throughout this section. Pandas is a library used for analyzing and using data (Pandas). Geopandas is an extension of pandas used to operate on the data in a spatial way ("GeoPandas 0.8.0.") . Pillow is a library that has many functions used for manipulating

images ("Image Processing in Python with Pillow."). Lastly, matplotlib is used for creating visuals ("Visualization with Python.").

Generating the map

The separate_by_year_map function is used to filter the dataset emissions-map.csv by year and create a new dataset that only has the data for that specific year. The function is later used in create_map. The function separate_by_year_map reads the emissions-map.csv and finds the index of the first time that the year shows up in the emissions-map.csv. After this there is a while loop that appends all of the data that is connected for that year into a list of lists. The function uses a pandas function, pd.DataFrame to filter the list for the specific columns needed for the map. Then this list of lists is tuned into its own file called emissions-by-year-map.csv. This file will be rewritten every time the year changes.
The create_map function mainly uses functions from the new libraries. It is responsible for merging the two datasets, and creating a map that shows the carbon dioxide emission from fossil fuels per country for a given year. First, the function reads 2 datasets, emissions-by-year-map.csv and the one provided by geopandas and stores them under the names: emission_file and world. The library pandas provides a function, pd.read_csv,to read the data sets. The function then merges the two data sets using pd.merge, another pandas function. Using a matplotlib function, plt.subplots, the axes for the map are created. Finally, the map and legend (using argument legend) are plotted using the map_data.plot function from matplotlib and a title is placed using ax.set_title from matplotlib.
In order to generate the map based off the position of the slider, we divided the size of the screen by the 50 (the number of years we're creating graphs for) which resulted in 28 and then used that in our function generate_year_from_pos() which takes in the x axis position of the mouse and divides it by 28 and adds 1964 (The first year we're mapping). To load a map based off the year, we created the function load_picture() which calls the create_map function, saves the map as "world.jpg" (overriding the previous map saved under that name), resizes and resaves the image and returns the name of the image (which is always "world.jpg"). It opens, saves and resizes the image using Image.open(), image.save() and image.resize(), which is part of the Pillow library.

Generating the Graph

To plot the graph we use the library, plotly.
To start off, we call the separate_by_year_graph function everytime the map is loaded. This ensures that the year in the emissions-edited-graph.csv file will always match the year in the emissions-by-year-map.csv file. This is so that the graph will plot the data for the same year that is currently being displayed on the map. This function works the same way as the separate_by_year_map function except it reads the dataset 'emissions-edited-graph.csv' and produces 'emissions-by-year-graph.csv'.
When plot_graph() is called, it reads the emissions-by-year-graph.csv file and saves the information using the read_csv_data() function. This function transforms all of the relevant information into a tuple that it then returns. The plot_graph() function then creates a bar graph based on the information in the tuple.

Generating the Totals

When totals() is called, the emissions-by-year-graph.csv file is read using the pandas library. Then we use the sum() function to get the sum of all the totals in the file. We then return a string with the totals and the text needed to display the totals.

Graph Button

The button is a Button object that has its own dataclass in a separate file. In the Button dataclass, there are 6 instance attributes. These are colour, x, y, width, height and text. colour is a tuple containing the rgb colour code that the user wants the button to be. x and y are the coordinates of where the button will show up on the surface. width and height are the dimensions of the button and text is the text that should appear on the button.

Displaying everything

To create our surface, slider, and button, we use the refresh_ all function to draw and display the surface, slider, button and map.
For our surface we created a surface object of 1400 by 1400 dimensions using pygame. We called display_surface.fill(white) at the beginning of the function, through the refresh_all function so when the function is called the background is

initialized as white, and we also called display_surface.fill(white) (also through the refresh_all function) in the while loop so the background remains white while the function is being run. For our slider we also used pygame to draw two rectangles: a small one to act as the slider itself, and another longer one as the "holder" for the slider to guide users on how far to scroll. We created and displayed these rectangles at the start of the function when we call the refresh_all function. Like the surface we also redrew and redisplayed the rectangles in the while loop the second time we called the refresh_all function. The rectangle representing the slider is redrawn to take the x-axis of the mouse, simulating movement, whereas the holder is just redrawn in place. To display our map we followed the same process as we did in displaying the surface and slider, we loaded the image based on the mouse's position which we did by calling generate_year_from_pos() inputting the mouse's x-axis and then inputting the year generated into our function load_picture() to generate an image and then display that image. We did this both outside the while loop at the beginning of the function when we called refresh_all (So the map of the first year, 1964, is displayed when the function is first executed) and then again in the while loop (So the map is repeatedly redisplayed based off the year indicated by the slider when the function is running). To display the button, a Button object is first created with all the necessary information about what the button will look like and its location on the screen. It is then drawn on the screen in the refresh_all() function. The way the button works is that it checks to see when the user's mouse finishes clicking on it. At that point it gets the position of the mouse, if the position is over the button, then it will plot the graph.

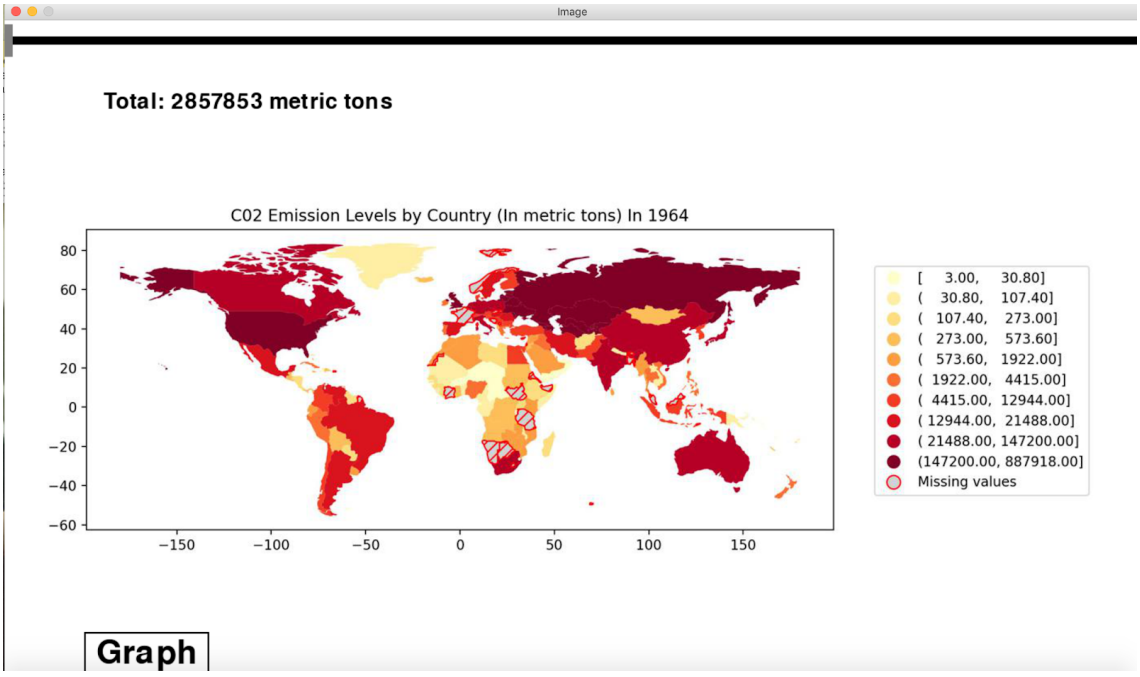
Overall

Overall, the program uses the map, graph, slider, button, and totals function created from computations to produce an interactive and visual display of how the map, graph, and total emissions change throughout the years as a user uses a slider to slide through the years and the graph button to open the graph on different years.

Instructions

The datasets should be saved in the same folder as the Python files. The datasets should not be in their own separate file in the folder with the Python files. They should be on the same level as the Python files.All datasets should be downloaded from the send.utoronto link. The shape file that we use to create the map is built into the geopandas library. This is the claim ID and passcode to download our modified datasets: Claim ID: HiEMAvrFj4SR5Yv6 Claim Passcode: ZC2CiXAVRuiiwRRu This is the link to the original dataset. https://datahub.io/core/co2-fossil-by-nation

When the main.py file is run, the TA will need to call display(). After that the pygame window will appear. This window will start off white as it is loading, however, after it loads, the TA should see a slider at the top, a map and a button that says 'Graph' on it. It should look like this after it loads:



To interact with it, the TA can click on different points on the slider to change the map. After a small loading

period, the map will change as the years change depending on what part of the slider is clicked. The TA can also click the 'Graph' button, this will generate a graph which corresponds to the year that is currently being displayed on the map. As the map changes, the total emissions will change with it. Totals is the total emissions for the whole world for that year.

Changes From Proposal

In our proposal we wanted to answer the question "how did carbon dioxide emission patterns from fossil fuels combustion, a leading cause of climate change in Canada, change from 50 years prior?" by creating a machine learning model that took 50 years of available CO2 emissions data from all the countries and made predictions of future emissions based off that data. We have changed the objective and execution of our project since receiving feedback from our TA about our initial objective being too broad and possibly too simple.We also realized making predictions might be difficult considering our dataset doesn't contain data for all countries during all 50 years and therefore some predictions would be less accurate than others. Also, technology in the past was not as accurate as it is today, so making predictions about data nowadays based on old data, might not be very accurate. We took the TA's advice on adding specificity to scope to our project which is why we have instead focused on visually representing the data to reveal disparities between all the countries in the world and between the years in addition to creating an accompanying bar graph which addresses the issue of complexity and scope. In order to address the issue of missing data we have coloured the countries that lack data gray with red slashes through it on the map, and simply setting it's bar graph value as zero. To make our project more complex we added a graph for every year. This graph plots the emissions for each country for that year in a bar graph. We also added a totals label on the map which displays the total CO2 emissions for the world for that year. Overall, while our program is still on carbon dioxide emissions, our focus changed to creating an interactive experience, which offers both map and graph display options, rather than a mathematical predictions approach. As well, the project was changed to include information for all countries, rather than only Canada.

Discussion

The results of our computational exploration visually display the differences of CO2 emissions for the countries all over the world, through both the map and the graph. Using this visual display, users can compare CO2 emission from different countries compared to other countries or compared to other years. This answers our question concerning the changes in co2 emissions from around the world over the span of 50 years and the differences between each country. One limitation we encountered was combining the map, slider and button and adding them all onto a pygame display. We were able to overcome this by extensively researching the pygame documentation and a few rounds of trial and error before realizing we would have to draw the objects and then continually display it while our function was running. In addition to this we were initially lost on how to generate a new map for each year and how to change the map that's being displayed based off the position of the slider on it's holder. We combatted this by saving the map as a jpg image under the name "world.jpg" and then regenerating a new image when the slider indicated another year, saving that image under the same name and displaying that image instead. In regards to our slider, we divided the length of our pygame surface by the total number of years and created a function (generate_year_from_pos()) which used integer division and addition to translate the mouse's position (and by extension the position of our slider) into a year.

Another limitation we discovered was that our dataset was not completely compatible with our shape file. To merge the two files together, it was dependent on the names of the countries, however, the country names did not entirely match up with each other (ex. China (mainland) instead of just China). To fix this, we had to manually go into our dataset and change the names of the countries. Another limitation with the dataset was that since we are using data from over 50 years ago, some countries no longer have the same name and as such do not match with the country names from 'naturalearth_lowres' dataset. If a country name does not match, the data about emissions is not seen on the map for that country. So, countries that became separate countries later on had to be adjusted. For instance, USSR is now 15 different countries and does not show up on the geopandas dataset as USSR but rather as those 15 different countries. As such, USSR was replaced by the 15 different countries in the co2 dataset, with all 15 countries having the same value in the 'Total' column as USSR initially had. The same was done for Yugoslavia.

A limitation regarding the button was that the user would have to click 3 or 4 times for the program to register it. To fix this, we had to change the code to check for when the click is over rather than when the mouse is being clicked. Some next steps for further exploration could be potentially relating the disparities in CO2 emissions of the countries to the differences in the lifestyle of those who live in those countries. Since the burning of fossil fuels

causes an increase in CO2 emission levels, and since fossil fuels are a source of energy, it would be interesting to note the differences in energy sources (for example, solar, wind, biomass etc.) and their usage (ie. attention to energy conservation or lack thereof) for the countries with lower emissions and potentially emulate them. It would also be worthwhile to examine if there's a correlation (which there likely is) between the lifestyle led by the population of each country and their emissions.

Conclusion

In conclusion, our project aimed to show the disparities of CO2 emissions between countries, in a way that's visually engaging yet informative. While we experienced some setbacks and struggled with some of the concepts and ideas we were trying to work with such as the slider, the map and using pygame and geopandans, referring to the documentation of the libraries we used was extremely helpful. All in all the creation of this project was a great learning experience and introduced us to some of the many ways real data, that can at times be intimidating, is transformed into the graphs and maps we see on a daily basis and the work that goes behind making that happen.

References

Ankthon. "Python: Display Text to PyGame Window." GeeksforGeeks, 2 Dec. 2020, www.geeksforgeeks.org/python-display-text-to-pygame-window/.

"CO2 Emissions from Fossil Fuels since 1751, By Nation." DataHub, Datopian, 24019, datahub.io/core/co2-fossil-by-nation.

Echessa , Joyce. "Image Processing in Python with Pillow." Auth0, 2 Dec. 2020, auth0.com/blog/image-processing-in-python-with-pillow/.

"Fossil Fuels and Climate Change: the Facts." ClientEarth, 19 Dec. 2019, www.clientearth.org/latest/latest-updates/stories/fossil-fuels-and-climate-change-the-facts/.

"GeoPandas 0.8.0." GeoPandas 0.8.0 Documentation, GeoPandas Developers, geopandas.org/.

"Greenhouse Gas Sources and Sinks: Executive Summary 2020." Canada.ca, Government of Canada, 25 Sept. 2020, www.canada.ca/en/environment-climate-change/services/climate-change/greenhouse-gas- emissions/sources-sinks-executive-summary-2020.html.

"Mapping Tools." GeoPandas 0.8.0 Documentation, GeoPandas Developers, geopandas.org/mapping.html.

"Matplotlib.legend." Matplotlib 3.1.0 Documentation, matplotlib.org/3.1.0/api/legend$_a pi.html$.

"Merging Data." GeoPandas 0.8.0 Documentation, GeoPandas Developers, geopandas.org/mergingdata.html.

Pandas, NumFOCUS, pandas.pydata.org/.

Pygame.draw - Pygame v2.0.1.dev1 Documentation, www.pygame.org/docs/ref/draw.html.

Ritchie, Hannah, and Max Roser. "CO and Greenhouse Gas Emissions." Our World in Data, 11 May 2017, ourworldindata.org/co2-and-other-greenhouse-gas-emissions.

SHUBHAMSINGH10. "Matplotlib.axes.Axes.set$_t itle()in Python."Geeks for Geeks, 19Apr.2020, www.geeksforgeeks.org/ma$
$axes - axes - set_t itle - in - python/$.

Tech with Tim. "How to Create a Button in Pygame." YouTube, YouTube, 14 Mar. 2018, www.youtube.com/watch?v=4$_9 twn$

"Visualization with Python." Matplotlib, 12 Nov. 2020, matplotlib.org/.

ZhongTr0n. "Create a Python Map from Categorical Data." Medium, Towards Data Science, 28 May 2019, towardsdatascience.com/create-categorical-choropleth-with-python-122da5ae6764.