CSC110 Project Report
Educational Crisis - A Closer Examination on the Correlations Between Covid-19 and School Closures Around the Globe

Shouyi (Ray) Hung, Yuhan (Charlotte) Chen, Mengyuan (Alyssa) Li, and Fuyang (Scott) Cui

December 8, 2021

Contents

1 Introduction

COVID-19 profoundly impacted students' learning environment and strategies. **Therefore, we are curious about how this global pandemic correlates with school closures all around the world as time passes, which is one of the main influencing factors that entirely changed our way of learning and living.** We will compare the levels of School Closure with the severity of COVID-19 of different countries in a specified time frame.

As a group of students, COVID-19 changed our way of learning from face-to-face to online for quite a long time. In 2021, results from the National Survey of Public Education's Response to COVID-19 had shown that more than 46%[1] of the students in the US across all grades are studying remotely. However, after COVID-19 eased a little bit, some of our schools reverted to the traditional in-person learning classes. A very prevalent issue amongst students is frequently switching between online and in-person learning. Under such circumstances, learning became increasingly hard as time passed.

Besides, as international students, we were energetic and excited about future university life. However, everything became harsh and unpredictable after the emergence of COVID-19. We are now bothered by expensive flight tickets, personal safety issues, and potential school closures as a result of the pandemic. For example, two of our group members are currently living in China because of COVID-19 and are troubled by the inconveniences of timezone differences.

Therefore, we aim to discover a general relationship between COVID-19 and school closures. With the help of the observed correlation, we could be more prepared in countering the impacts of COVID-19 as individuals. For example, we could reasonably predict the next virus outbreak based on our project and switch to online classes beforehand.

Additionally, from a broader scope, our project could provide intuitions to educational institutions about the trend of school closures and COVID-19 cases. Therefore, they could identify whether they made a correct decision of closing/opening schools during the pandemic, and draft plans to minimize the impacts in the future. In other words, we could utilize our project as a guide to help prevent future impacts that could rain onto the educational sectors that suffered during the current pandemic.

[1] National Survey of Public Education's Response to COVID-19 Infographic, https://www.air.org/sites/default/files/2021-07/infographic-results-national-covid-survey-june-2021_1.pdf

2 Dataset Description

We have identified two main datasets that will be relevant for our project's implementation.

These are:

1. Global School Closures for COVID-19[2] – Obtained from Kaggle, compiled by Saleh Ahmed Rony, sourced from UNESCO
2. COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University[3] – Obtained from GitHub, compiled by JHU, sourced from WHO, ECDC, DXY, US CDC, etc.

Both datasets will be stored in a Comma Separated Value file, which will allow us to read from them easily through Python's csv library.

Furthermore, both datasets are very credible as they are sourced from multiple sites, including but not limited to WHO, ECDC, and US CDC. Furthermore, these datasets are also licensed under the Creative Commons Attribution 4.0 International (CC BY 4.0), which allows us to utilize these data for our own needs.

The datasets that we have downloaded and utilized in this project were chosen because they were compiled in a way that allows easy access and manipulation. By using datasets that are already organized could improve the efficiency and robustness of our program.

The Global COVID-19 Dataset (Time series) has the following structure:

Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	...
	Afghanistan	33.93911	67.70995	0	0	0	...
	Albania	41.1533	20.1683	0	0	0	...
...	...	...	...	...	...	...	...

The headers extend up until November 1, 2021 for the dataset that we will be using in our application.

Starting from column 5 and onwards (for the Global COVID-19 Dataset) contains the amount of cumulative cases for the specified country in the date shown in the header row.

The School Closure Dataset has the following structure:

Date	ISO	Country	Status	Note
17/02/2020	CHN	China	Partially open	
17/02/2020	MNG	Mongolia	Closed due to COVID-19	
...	...	...	...	...

The data is organized by entries of different country each day.

Some of the country names in our datasets contain characters that are not part of the standard ASCII table. An example would be "Curaçao", where "ç" is a French (Latin script) letter.

Since these letters are not in the standard ASCII Table, displaying and encoding issues may arise. So for convenience, we will ignore them.

For the purpose of this project, we will simplify the standard ASCII Table to only contain characters from "a-z", "A-Z", "0-9", "!-~".

These letters also correspond to "string.ascii_lowercase", "string.ascii_uppercase", "string.digits", and "string.punctutation" respectively, from the Python string library.

[2] School Closure survey report, www.kaggle.com/salehahmedrony/global-school-closures-covid19

[3] One of the most used COVID-19 Cases Database, github.com/CSSEGISandData/COVID-19

3 Computational Overview

Our project will be separated into 3 main parts of implementations: Data Processing, Graphical User Interface (GUI) implementation, and Misc.

3.1 Data Processing

3.1.1 Downloading Data

In this part of the implementation, we have utilized the libaries requests, os, and hashlib to download and verify files downloaded respectively.

The requests library allows the application to reach out to a certain URL and get the contents that are contained in that URL. We will use this library in combination with os to download and save the file in the user's working directory, so our application can run with the specified data.

Specifically, we used requests.get, which allows us to retrieve the information contained in the specified URL, and os.makedirs, which allows us to create directories to store our files in.

We utilized a Public GitHub repository to upload all assets, so the application can visit the specified URL and obtain all the assets needed without needing any Verification Token.

We also implemented a MD5 Hash function[4] that uses the hashlib library.

Specifically, we used hashlib.md5 to create the MD5 value of a certain file, and hashlib.md5.hexdigest to return the result in a string.

For the files that are being downloaded, the application will compare it with the specified MD5 Checksum value we ran on our machines to ensure that the files downloaded are the same as the ones we are working with.

In our resource_manager.py, we have created a class to store all of our different files. These resource objects will contain a remote path, the MD5 identifier, and the local path so we can download and verify the files.

The purpose of this sub-sub-section is to ensure that our application will run on any machine with our predetermined datasets.

3.1.2 Data Manipulation Algorithms

We have written an algorithms.py file to contain most of the sorting algorithms that we know. This includes but not limited to: Merge Sort, Bubble Sort, Selection Sort, Insertion Sort, Linear Search, Binary Search.

The purpose of this part of the implementation is to demonstrate the impact on running time if we utilized another sorting algorithm instead of the original one.

Upon initialization of the application, the User will be prevented to do any operations on the application until the users selects a sorting algorithm and initialize the data with it.

The data can be initialized multiple times for the user to notice a difference in different sorting algorithms.

The different searching algorithms will be utilized with our GUI application to display the covid case results that are linked to the country that the user selected.

3.1.3 Data Initialization

We read the data into Python with help of the Python *csv* library.

For our COVID-19 Dataset, we only used the columns with the headers "Country" and "Dates", as they provide information for us to plot in the application.

For our School Closure Dataset, we used all of the columns aside from the "Note" column, as it provides information in a way that is hard to process.

Then, by using the data we have loaded into Python, we performed multiple filtration to the loaded data so that we can obtain GLOBAL CONSTANTS that will be used by the other Python files.

[4] Implementation referenced from StackOverFlow, https://stackoverflow.com/questions/3431825/generating-an-md5-checksum-of-a-file

Our program created the following data classes:

- ClosureStatus - An Enum class that maps the different status of school closure to an Enum value
- Location - A class that represents a location
- Country - A class that represents a country, based on its location
- BaseData - A basic class
- TimeBasedData - A data class that is based on time
- CovidCaseData - A class that represents an entry in Covid data based on a Time and Country
- SchoolClosureData - A class that represents an entry in Closure Data based on a Time and Country

Our program will generate the following GLOBAL CONSTANTS:

- ALL_COVID_CASES - A list of all CovidCaseData objects read from the csv file, including Provinces and Cities
- COUNTRIES_TO_COVID_CASES - A dictionary mapping from a country name to all of its respective CovidCaseData
- GLOBAL_COVID_CASES - A list of all CovidCaseData that contains cases summed across all countries on a certain date
- ALL_SCHOOL_CLOSURES - A list of all SchoolClosureData that is read from the csv file
- COUNTRIES_TO_SCHOOL_CLOSURES - A dictionary mapping from a country name to its respective SchoolClosureData
- GLOBAL_SCHOOL_CLOSURES - A list of SchoolClosureData with only dates and the status of the majority of the schools in that date
- COUNTRIES - A list of Country objects that are read from the CSV Files
- SORTED_COUNTRIES - A list of sorted Country objects from COUNTRIES

Then, we converted the raw data read from the csv files into the data classes specified above, so we could easily utilize those data in practice.

We stored all of our data in data.py as GLOBAL CONSTANTS, which will allow the other Python files to import and utilize them directly. We will refer to data.py in our gui.py file to display graphs.

This brings us to the next part of our project, which is GUI Implementation.

3.2 GUI Implementation

We mainly utilized *PyQt5* to generate an interactive user interface and *Matplotlib* to plot graphs.

We have implemented the following functionalities:

- A fully functional Graphical User Interface with menu bar, multiple buttons, input date edits, and sliders
- A loading progress bar in the main UI Window when the user clicked "Initialize"
- Panning and Zooming functionalities on the graph by left click & drag and scrolling
- Navigation bar that allows the user to return the plot to the original state
- Scatter plot COVID-19 cases and school closures visualizations supporting filtering by country names
- Customizable color and marker types for plots
- An application with icon

3.2.1 Main Window

We utilized PyQt5.QtCore, PyQt5.QtGui, and PyQt5.QtWidgets to create the main window.

This window consists of the following classes:

- Standard Label - The class we use to add a text label
- Standard Push Button - The class we use to add a confirm button
- Standard Combo Box - The class we use to choose which country's data to display
- Standard Date Edit - The class we use to select the start date and end date
- Standard Checkbox - The class we use to select options
- Standard Progress Bar - The class used to display the progress
- Standard Menu Bar - The class used to setup the menu bar on the top left corner of the application
- Plot Canvas - The class that is used to plot figures on the Main Window

The PyQt Main Window that we displayed will use a combination of the classes above to output the desired window.

3.3 Misc

Aside from the main window and data manipulation libraries, we also included the libraries sys, logging, time, typing, and ctypes.

The libraries sys and logging are used to generate a log in the console to inform the users of the different actions that our application is doing.

The time library is used to evaluate the time that it takes for us to initialize the data and to output the time in the console log.

We used typing to specify the types of inputs that our functions will be taking, in algorithms.py, data.py, and gui.py.

Lastly, we used ctypes to specifically set an identifier for our application, so the icon will be displayed normally when the application window is active.

4 Instructions

1 Download the zipped file we provided on MarkUs. The zipped file should contain ”algorithms.py, data.py, gui_main.py, gui_utils.py, main.py, resource_manager.py, settings.py, requirements.txt”

2 Unzip the file in a directory with *ONLY* English characters (i.e. Do not place it in a directory that contains non-English characters)

3 Boot up PyCharm and open ”requirements.txt” in the directory you just unzipped the files in. Click ”Install requirements” to automatically download all the libraries needed

4 Either use PyCharm or a terminal to run ”main.py” and you should be able to see a window popup

5 Select a ”Sorting algorithm” and click initialize. After a short amount of time, the application should be ready

6 You should be able to set start and end dates for the plot, pan and zoom on the plot, select countries and plot the result again, and utilize the menu bar for more customizable configurations.

7 Feel free to explore the application from this point onwards!

Side Notes:

- The application we've created is only fully compatible with the Windows operating system.
- It can run on a M1 Mac with very extensive setup, but it will be too difficult to set it up. (You need to run Terminal on Rosetta and install the x86_64 version of all libraries used through pip)
- We are unsure of the performance of our application in an Intel Mac

5 Changes from the Proposal

- Added MLA Style Footnote in multiple occasions
- Added more details in Computational Overview
- Added features to download dataset and write in working directories
- Added features to hash a file in MD5 to ensure its completeness
- Removed some unused references and added new ones
- Removed the US Covid Dataset as it is unnecessary for our analysis purposes

6 Discussion

6.1 Results and Conclusion

Our application can be utilized to visually determine if there are any correlations between COVID-19 Cases of a certain country and School Closure status of the certain country. This would be enough for us to answer the research question, because we can clearly identify a trend between COVID-19 Cases and School Closure statuses.

Furthermore, we can also utilize this application to generate plots about data around the world that can be shared to raise awareness, which could allow educational sectors to be more prepared for future events like the COVID-19 Pandemic.

We can also identify some of the ”Outliers” from the general trend for some countries. These may exist because their country officials decided for schools to continue operating amidst the severe conditions posed.

6.2 Limitations

6.2.1 Overall Limitations

A major limitation of our application is the fact that our data is not a 100% accurate data from all around the globe. School Closure Data being used is uploaded by an individual on Kaggle. The COVID-19 Data can also contain inaccurate information.

Furthermore, we recklessly removed some of the country names in our datasets because they contain non-ASCII characters, which limits the amount of data.

We also removed some country names because they were not present in both datasets, making the actual data presented in our application even more limited.

6.2.2 Obstacles

Our application crashes all day, all night, 24 * 7, non-stop. Making us want to drop out of CSC110. But nevertheless, we still overcame the difficulties and produced a ”Functioning” application.

6.3 Next steps

We will be carrying the skills we developed in this project into other projects in the future. We have now acquired the ability to download, process, and hash data. These abilities would be necessary to create any automated application in the future.

We also acquired one of the most important skills in the Computer Science Industry, which is to design a Graphical User Interface. This is the ultimate version of ”Designer to User” contract, which provides a very ”Easy to use” interface so that users without any knowledge in Computer Science can also use this application to their advantage.

This project will be held public on GitHub repository for anyone to fork and create an updated version of the application. We have included docstrings and comments everywhere around our code for anyone that wishes to fork our Repository.

We will also claim copyright, specifically MIT License 2.0 in the GitHub repository, so no harm will be done to the University of Toronto’s CSC110Y1 Fall’s instructing team, specifically Professor Mario Badr, Professor Thomas Fairgrieve, and Professor David Liu.

7 References

"Coronavirus." World Health Organization, WHO, 10 Jan. 2020, www.who.int/health-topics/coronavirus#tab=tab_1.

"CSV File Reading and Writing — Python 3.10.0 Documentation." Python Documentation, docs.python.org/3/library/csv.html. Accessed 30 Oct. 2021.

Johns Hopkins University. "GitHub - CSSEGISandData/COVID-19: Novel Coronavirus (COVID-19) Cases, Provided by JHU CSSE." GitHub, CSSEGISandData, github.com/CSSEGISandData/COVID-19. Accessed 30 Oct. 2021.

Kaggle — Global School Closures for COVID-19. Saleh Ahmed Rony, www.kaggle.com/salehahmedrony/global-school-closures-covid19. Accessed 30 Oct. 2021.

https://stackoverflow.com/questions/3431825/generating-an-md5-checksum-of-a-file