CSC110 Project Report: The effect of COVID-19 on other major diseases

Savanna Pan, Anna Sophia Lee Pantoja, Tanvi Patel, Vidhi Patel

Tuesday, December 14, 2021

Introduction

There is no doubt that the COVID-19 pandemic has had a significant impact on our daily lives. Most notably, it has drastically affected our healthcare system. In an attempt to manage and control the sudden surge in COVID-19 cases, many resources such as personnel, medical supplies, and hospital spaces were redirected to the treatment of the coronavirus disease (Pelley, 2020).

But although COVID-19 has populated the news with confirmed cases and deaths, it is not the only major cause of death. There are many diseases with high mortality rates, such as malignant neoplasms and diseases of the heart, just to name a few (Statistics Canada, 2021). So, as the severity of the pandemic increases, we must consider how the response to the pandemic may have impacted the healthcare available to people with diseases not associated with COVID-19.

Our interest in this research topic was sparked after coming across an article that reported on a study which observed a relationship between COVID-19 and increased mortality among people with diabetes in the United States (Ran et al., 2021). The article suggested that this could be attributed to the decreased amount of medical attention and resources given to diabetes patients because of the need to address COVID-19 patients. This impact on the healthcare system, combined with the increased chance of complications that a pre-existing medical condition would cause, led us to ponder whether factors such as these, which came as a consequence of the COVID-19 pandemic, would have affected the mortality rate of other diseases, in addition to diabetes.

Our research question is, **"Has the pandemic had an effect on the death counts of other diseases?"** Specifically, we would like to know if we can use data from the start of the pandemic to this year to identify a relationship between the number of hospitalized COVID-19 patients and the number of deaths due to other chronic diseases. We also want to use this data to determine whether changes in the number of COVID-19 hospitalizations affects the mortality rate of other major diseases.

We intend to reach our conclusion by calculating correlation coefficients, which will communicate the strength and direction of the relationship between variables in our dataset, and by performing regression analysis to describe the relationship between two variables as a linear function. This will be done by implementing a simple linear regression algorithm that will help find the equation of the linear function, which we can then plot along with a scatter plot to observe and interpret our data points and discern their relationship.

We will hopefully be able to obtain an answer to our question through the analysis of a dataset and the execution of our computational plan, the results of which will be visualized and communicated.

Dataset Description

The dataset that our program utilizes combines data from two datasets: one containing weekly death counts from several causes and the other, COVID-19 hospitalization data.

The first dataset we used was published by Statistics Canada. This dataset, titled *Provisional weekly death counts, by selected grouped causes of death*, is available for download on the Statistics Canada website in CSV (.csv) format. As the description states, it provides data on the number of deaths that occurred weekly due to various major causes, grouped by province/territory. However, in our final dataset we only used data about deaths from Ontario attributed to three major diseases: diseases of the heart, malignant neoplasms, and chronic lower respiratory diseases. Data is available from January of 2010 to October of 2021, but we only used the data in the date range that was compatible with our second dataset, which corresponds to April 5, 2020 to July 31, 2021 (Government of Canada, 2021).

To verify usage permission: https://www.statcan.gc.ca/en/reference/copyright?MM=as

Reference: Statistics Canada.

https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1310081001

The second dataset is titled *COVID-19 cases in hospital and ICU, by Ontario Health (OH) region*, and is available for download in CSV format on the official website of the Ontario Government's Data Catalogue (citation). Like the title indicates, this dataset provides daily data on the number of COVID-19 cases in hospitals and ICU in Ontario, grouped by region. However, our final dataset only includes data from the following columns of this dataset: date, OH region, and current hospitalizations with COVID-19. In addition, this dataset contains observations from April 2, 2020 to December 10, 2021. But since the frequency of the other dataset is weekly and we would later need to combine the data from both into one, we only included this dataset's observations from April 5, 2020 to July 31, 2021, which represent data for full weeks (Ontario Government, 2021).

Reference: Data Ontario

https://data.ontario.ca/dataset/covid-19-cases-in-hospital-and-icu-by-ontario-health-region

Computational Plan

It should be noted that our final project computational overview differs greatly from the previous computational plan detailed in our project proposal. This is because our project now consists of implementing correlation and regression analysis rather than time series analysis to answer our research question.

Below are the descriptions for each computational phase of our program. For the duration of this section, we will refer to the dataset of weekly death counts as "dataset1" and the COVID-19 hospitalization dataset as "dataset2".

Data Cleaning and Transformation

The load_data module contains two functions to read and perform computations on our datasets, one function to combine the select data from both datasets into one, and various helper functions.

To read dataset1, we use the function read_death_count_data, which takes no arguments when called and returns a list containing a list of dates as datetime.date objects, a list of the combined death counts for three major chronic diseases, and three separate lists with the death counts of each disease. These lists are to be later used as input to create the final dataset.

This function first initializes empty lists to collect the data from the CSV file for dataset1, which is first opened then read by the csv reader. Once the function appends all the elements of the rows in the csv file to a list containing all the rows, the function calls the helper function create_datetime and passes all the collected rows as the parameter. The function create_datetime collects only the rows in the collect_all_rows list and appends it to a list that only contains the rows with the dates, which are all type string objects. The helper function then splits the string into year, month and day before finally removing the punctuation (in this case, commas) and using a for-loop to transform the strings into datetime.date objects. The helper function returns the transformed dates in a list.

Then, the read_death_count_data function goes through a filtering process to extract the rows containing the death counts for the three chronic diseases that our project will consider, storing the death counts for each as lists. The first two elements of these lists are removed because they are not death counts, rather just the name of the diseases and an empty string leftover from the previous rows of data. There is then a for-loop that extracts only death counts from Ontario for the three chronic diseases, and also adds to a list that will contain the combined death counts for the three diseases.

The helper function remove_extra_dates is then called, taking the dt_lst (datetime list) as the parameter. This helper function works by filtering the dates in dt_lst that are not within the date range that our program will analyze. It first collects all the dates in 2020 that are after the month of April, then collects the dates in 2021. This function's return is assigned to the variable return_dates in the read_death_count_data function. From there, we calculate the number of extra dates that need to be removed and we remove those date's death counts from each list by calling the helper function remove_extra_nums. Once we have obtained the lists with the correct individual and combined number of death counts for the dates in dt_lst, the function returns them all.

The function read_hospitalization_data is responsible for reading and processing dataset2. To clean and transform the data for our purposes, we used the pandas library, which also allows us to store this dataset as a DataFrame, a data type that supports many useful operations to manipulate data. We first chose to simplify the data by storing only the variables relevant to our project's goal in the DataFrame, which include the date, number of hospitalizations, and region of Ontario. We then changed the data type of the date column into datetime64, a pandas data type, using the to_datetime method from the pandas library in order to make it easier to later correctly set the frequency of the data. The time component in this column is also normalized to avoid encountering duplicate dates as the result of differing time components, which we would not be taking into account.

We then pivoted the DataFrame using the unstack pandas method, setting dates as the index, the regions as the columns, and the number of hospitalizations per day as the values in the DataFrame. We also re-indexed our DataFrame after identifying a missing date in the data, setting the new index as the correct complete range of dates, and we filled in the missing value with 0 using the pandas method, fillna. This then set the frequency of our DataFrame as daily. The data from dataset2 contains daily observations, but the missing date caused our DataFrame to have a frequency set as "None". By filling in the missing day's data, the frequency became 'D', which would then allow us to safely aggregate the data as needed.

We grouped the number of hospitalizations by region by calculating the sum of the values in the five region columns, putting them into a new DataFrame with a single column representing the number of hospitalizations in all of Ontario. We then wished to upsample the data from daily to weekly to match the frequency of dataset1. We first restricted the DataFrame to include observations from April 5, 2020, to July 31, 2021, which are the days that we could account for that were compatible with dataset1. We then aggregated the data by week by using the resample pandas function and entering "W-SAT" as a parameter, which changes the DataFrame frequency to weekly, where one week starts on Sunday and ends on Saturday, like dataset1. When called, read_hospitalization_data returns our final processed data as a DataFrame.

The function group_datasets is responsible for producing the final dataset that other modules in the program will use. The function first calls the helper function, get_dataset_values, to obtain the data contained in each column of both processed datasets. A dictionary is created, called grouped_datasets, which maps the label of each column from the datasets to their corresponding values. We then used this to input the data into a DataFrame, along with the datetime index that both datasets share, and this final DataFrame is returned by the function when called.

Computations

The correlations module is responsible for calculating the correlation coefficient, also known as the Pearson coefficient by implementing the mathematical formula as a Python function which can be called by other modules.

The formula to calculate this coefficient, known as Pearson's r, is:

$$r = \frac{n \cdot \Sigma xy - \Sigma x \cdot \Sigma y}{\sqrt{(n\Sigma x^2 - (\Sigma x)^2)(n \cdot \Sigma y^2 - (\Sigma y)^2)}}$$

Where r is the correlation coefficient, x represents the x-values, y represents the y-values, and n is the number of x (or y) values we have.

Our implementation of this formula involves looping through all the x-values and y-values, calculating the sums in the formula using an accumulator. The function then performs the rest of the operations in the formula after obtaining the sums to calculate the coefficient, which the function will return as a float.

The purpose of this coefficient is to quantify the strength and direction of the relationship between two variables. In our program, this function is used to calculate the correlation between the number of COVID-19 hospitalizations and the death counts in our data to later visualize and interpret to answer our research question. The coefficient can take on values between 1.0 and -1.0, where values approaching 1.0 indicate a strong positive relationship, values approaching -1.0 indicate a strong negative relationship, and a value of 0 indicates no relationship.

The linear_reg module is responsible for implementing a simple linear regression algorithm to calculate the regression line that attempts to describe the relationship between the number of COVID-19 hospitalizations, the x-value, and combined chronic disease death counts, the y-values, as a linear function.

The module contains a function called generate_best_fit_line that returns the slope and y-intercept of the regression line for our data. It first retrieves the values of the two columns from our dataset that we are interested in by calling the helper function retrieve_data, which returns these values as a tuple of numpy arrays, and assigns each element of the tuple to a variable.

The slope of the line is calculated with the formula:

$$m = \frac{\Sigma(x - \bar{x}) \cdot (y - \bar{y})}{\Sigma(x - \bar{x})^2}$$

Where m is the slope, x is the x-values, $\bar{x}$ is the mean of all x-values, y is the y-values, and $\bar{y}$ is the mean of all y-values.

The generate_best_fit_line implements this formula by first calculating the mean of the x-values and y-values, which are then used to calculate the sums in the formula using a for-loop. This function then performs the remaining division to obtain the value of the slope. This function also calculates the y-intercept of the regression line, which is calculated with the formula:

$$b = \bar{y} - (m \cdot \bar{x})$$

Where b is the y-intercept, $\bar{y}$ is the mean of the y-values, m is the slope of the line, and $\bar{x}$ is the mean of the x-values. This function returns the slope of the regression line and the y-intercept of the regression line as a tuple.

Visualization

The visualization module is responsible for displaying the two plots that our program can create. It imports the linear_reg and correlation modules in order to call their functions and uses the matplotlib library to create the graphs our program will produce.

The first graph is a scatter plot with the number of COVID-19 hospitalizations on the x-axis and the combined chronic disease death counts on the y-axis. The scatter plot plots the data points, along with the linear regression line. This visualization is handled by the plot_regression_line function, which calls retrieve_data to obtain the x-values and y-values and calls generate_best_fit_line to obtain the slope and y-intercept of the regression line, the values of which this function uses to create the line that will be plotted. We used the scatter method from the matplotlib library to create a scatter plot with our x-values and y-values and with specific parameters to set the colour, size, and shape of the markers. We then use the values of the slope and y-intercept of the regression line to predict the y-value (death count) corresponding to each x-value and use those points to graph the regression line using the plot method from the matplotlib library and specific parameters to set the colour and width of the plotted line. Calling plot_regression_line displays the scatter plot.

The second graph is a heatmap that is used to display a table of correlations between each pair of variables in our dataset. This visualization is handled by the plot_correlations function. To obtain each coefficient correlation we called the helper function get_correlations, which returns a numpy array with a list of lists, where each inner list contains each row of correlation coefficients. This heatmap is created the imshow method of the matplotlib library, which will display each correlation coefficient in a box and we set the colour scheme of the boxes as shades of green. The matplotlib methods xticks and yticks were used to assign horizontal and vertical label names, which are the names of the columns in our dataset. The rotation of the labels on the y-axis was set as vertical to make them legible to the user. This heatmap is also plotted along with a color bar, created using the colorbar matplotlib method, to communicate to the user how the intensity of the color expresses the strength of the relationship between each pair of variables. This function also uses a for-loop nested in another for-loop to cycle through each of the boxes in the heatmap and add text with the value of the corresponding correlation coefficient, rounded to 3 decimal places. Calling the plot_correlations function displays this heatmap.

Instructions (Obtaining Datasets and Running the Program)

Please note that all Python script files, datasets, and the requirements.txt file must be in the same folder.

Install required Python libraries

In PyCharm, to install all Python libraries needed to run this program, double-click the file from the Project pane. A message should appear at the top of the opened file informing you of unsatisfied package requirements. Click on "Install requirements" and wait for the installation to complete. Alternatively, the individual libraries may be installed from the PyCharm "Python Packages" window, where one can search the name of each individual library and select the option to install.

Download datasets

To obtain the two datasets that this program will utilize, please go to UTSend (which you can access using this link: https://send.utoronto.ca/) to access and download the processed versions of these datasets. Select the option to "Pick-up" a file and enter the following information:

Claim ID: WHPSZbnS8r6g2tmv
Claim Passcode: ZMgyYkD6x344FgdK

The datasets will be in CSV (.csv) format, and the files should be named "dataset1.csv" and "dataset2.csv". Please keep these two files in the same folder as the Python script files and requirements.txt.

Alternatively, these datasets may also be downloaded directly from MarkUs.

Run main.py

Please download the following PNG (.png) files from MarkUs to see screenshots of what you should expect to see when you run the file: "screenshot1.png", "screenshot2.png", "screenshot3.png".

To run the program, run the main.py file. We recommend that you run this file by right-clicking the PyCharm window and selecting "Run File in Python Console". What you should expect to see first is a prompt in the Python Console asking you to input a number to select which graph to display. After making a selection, to reselect your choice of number you must rerun the main.py file and enter the input again. Please refer to "screenshot1.png" for an example of what you may see.

To view the correlation table, enter "1" into the Python Console. A new window should appear that displays the heatmap that visually represents the correlation between variables. Refer to "screenshot2.png" for an example of what you may see. If the table is not completely visible, click on the "configure subplot" button at the bottom left of the window, which should be the second last button, and adjust the slider labelled, "bottom".

To view the scatter plot with the regression line, enter "2" into the Python Console. A new window should appear that displays the scatter plot which plots the data points with the line that attempts to describe a linear relationship between the variables on the x-axis and y-axis. Refer to "screenshot3.png" for an example of what you may see.

In the case of both graphs, you may interact with the plot through the buttons at the bottom, which allow you to pan around the plot, zoom-in on sections of the plot, adjust subplot settings, and save the figure. In addition, you can hover over data points to see the corresponding x-value and y-value at the bottom of the window. For instance, hovering over each square in the scatter plot will show you the exact number of COVID-19 hospitalizations (x-value) and combined chronic disease death count (y-value) represented by that data point at the bottom of the window.

Changes in Project Plan

The primary change made to our project was the method we intended to use to answer our research question. We had originally planned to use an ARIMA model to forecast normal mortality rates for major chronic diseases since the start of the pandemic, and compare it with the real mortality rates recorded during the same period. This would allow us to determine whether there has been a significant and unusual increase (or decrease) in these rates since the start of the pandemic, which we would then attribute to the response to the pandemic.

But after reflecting on TA feedback and discussing these suggestions as a group, we decided to shift our project from time series analysis to correlation and regression analysis to answer the same research question. We began following a Coursera Time Series Analysis course offered by IBM to learn about how to use the model to fulfill our purpose, and we encountered quite a few obstacles (Grover & Maldonado). For instance, our data would need to be made stationary in order to be used by the model, and this would require us to perform multiple additional transformations such as using the differencing method and taking the logarithm of the data. And since we wanted to avoid relying heavily on libraries such as pandas to transform our data, we were concerned about whether we would be able to learn how to achieve this using only Python. And after looking into the logistics of implementing the ARIMA model in Python and producing a forecast, we came to the conclusion that it required a larger amount of background knowledge in statistics to utilize properly, which we lacked and would likely not be able to effectively develop in a reasonable amount of time. So, we decided to use correlation and regression analysis to answer our research question instead, which would allow us to implement a wider range of functions and calculations ourselves in Python.

We also decided to load additional data from another dataset, since our new method of answering the research question would not allow us to relate the death count data with time. And then we chose to focus on analyzing the data for a specific location, Ontario, and for a smaller number of diseases, which we also grouped into one category, as suggested by the TA feedback. We also read and processed the data without using solely pandas; we processed one dataset using Python and Python modules, while the other dataset was processed using the pandas library, which allowed us to explore these two different ways of cleaning and transforming data.

Discussion

The results of our computational exploration do not match the results we had anticipated. We had expected that greater hospitalizations would coincide with greater mortality in other chronic conditions, since a greater influx of COVID-19 patients would strain the resources of a hospital, leaving many people facing longer wait-lists for medical procedures, less access to medical attention, and more hesitant to seek treatment to avoid infection. In fact, a report from the Deloitte/Canadian Medical Association confirmed our initial thoughts, which found that between August and December 2020 alone, there were over 4000 excess deaths due to delayed medical care unrelated to COVID-19 (Global National, 2021).

Using our data to generate the correlation matrix (heatmap) and the scatter plot with the regression line, we were not able to observe a strong relationship between the number of COVID-19 hospitalizations and the death counts for any of the three diseases included in our final dataset. So, based on our results alone, the answer to our research question would be no; the pandemic has not affected the death counts of other diseases.

Our correlation table displays the correlation coefficient for each pair of variables in the dataset, which can take on values between 1.0 and -1.0, where values approaching 1.0 indicate a strong positive relationship, values approaching -1.0 indicate a strong negative relationship, and values approaching 0 indicate a very weak relationship. For the purposes of this project, we are only interested in the correlation between hospitalizations with each of the death counts (individual diseases and combined). The heatmap generated by our program shows the intensity of the relationship as white, corresponding to values between 0 and 0.2, which suggest that there is little to no relationship between these pairs of variables. In other words, these results suggest that greater COVID-19 hospitalizations do not cause greater deaths in other major diseases and vice versa.

This conclusion is reinforced by our regression analysis. In this plot, the variables we considered are the weekly number of COVID-19 hospitalizations as the independent variable and the combined weekly death counts for the three diseases in our dataset as the dependent variable. Regression is meant to examine how the independent variable causes the dependent variable to change. We employed simple linear regression to attempt to describe the relationship between the two variables as a linear function which best fits the data, whose equation we calculated and plotted over a scatter plot. Viewing the scatter plot, the overall shape of the data does not resemble a line. And by graphing the regression line, we can see that the regression line fits very little of the data points from our dataset, suggesting that there is no linear relationship between the two variables.

The disparity between our expected and actual results may be due to the quality of our data. In particular, the data for weekly death counts for various causes of death is provisional, which may not reflect the real counts, as this data may be revised. The metadata for this dataset even specifies that "data for the reference years 2020 and 2021 are provisional due to the shortened duration of data collection", which exactly includes the date range we considered (Government of Canada, 2021). Also, the Statistics Canada website containing this dataset was not functioning while we finalized our project, so we did not have access to the latest revised death counts and the date range we could consider was limited.

Other factors to consider related to this dataset and mentioned in the metadata are that certain deaths may take longer to investigate and be report to Statistics Canada and death counts currently categorized as "Information Unavailable" may later move to other categories. Also, COVID-19 is considered a cause of death, so this category may include counts of people who experienced a worse response due to a health condition which may not have otherwise resulted in death.

To continue exploring our research question, we would utilize the latest versions of our datasets, as any revisions made to the data would be more accurate. Or we may consider including datasets that include a wider range of chronic diseases, allowing us to better investigate our question since the pandemic is not only affecting the small subset of diseases that we considered in our analysis. Using datasets with complete data for more areas of North America would also likely help us reach more comprehensive results. Achieving this would require us to familiarize ourselves more with libraries for data manipulation/visualization and we would likely need to modify our code for loading data. This would allow us to both further develop our knowledge and skills and more efficiently process a larger amount of data, which may help us reach results that more accurately reflect the relationship between COVID-19 and the mortality of other diseases.

References

Bewick, V., Cheek, L., & Ball, J. (2003, November 5). Statistics Review 7: Correlation and regression. Critical care (London, England). Retrieved December 13, 2021, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC374386/.

Display the pandas DataFrame in Heatmap style. GeeksforGeeks. (2020, August 17). Retrieved December 13, 2021, from https://www.geeksforgeeks.org/display-the-pandas-dataframe-in-heatmap-style/.

Edureka. (2018, June 26). Linear Regression Algorithm — Linear Regression in Python — Machine Learning Algorithm . Youtube. Retrieved December 13, 2021, from https://www.youtube.com/watch?v=E5RjzSK0fvY.

Global National: Nov. 30, 2021 — More travel restrictions as Omicron raises vaccine questions. (2021, December 1). [Video]. YouTube. https://www.youtube.com/watch?v=6njAgUgu4KM

Government of Canada. (2021, October 14). Provisional Weekly Death Counts, by selected grouped causes of death. Statistics Canada. Retrieved November 1, 2021, from www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1310081001.

Grover, M. J., & Maldonado, M. (n.d.). Specialized models: Time Series and survival analysis. Coursera. Retrieved November 1, 2021, from www.coursera.org/learn/time-series-survival-analysis.

How to Plot a DataFrame using Pandas. Data to Fish. (2020, November 25). Retrieved December 13, 2021, from https://datatofish.com/plot-dataframe-pandas/.

The Matplotlib development team. (n.d.). Matplotlib 3.5.1 documentation. Matplotlib. Retrieved December 1, 2021, from https://matplotlib.org/stable/index.html#.

Matplotlib Tutorial. W3Schools. (n.d.). Retrieved December 2, 2021, from https://www.w3schools.com/python/matplotlib_intro.asp.

Noonan, R. (2019, July 4). Python Correlation Heatmaps with Seaborn & Matplotlib. Youtube. Retrieved December 13, 2021, from https://www.youtube.com/watch?v=UgtjatBt3vY.

Ontario Government. (2021, December 13). COVID-19 cases in hospital and ICU, by Ontario Health (OH) region. Ontario Data Catalogue. Retrieved December 5, 2021, from https://data.ontario.ca/dataset/covid-19-cases-in-hospital-and-icu-by-ontario-health-region.

Pandas Tutorial. W3Schools. (n.d.). Retrieved December 2, 2021, from https://www.w3schools.com/python/pandas /default.asp.

Pelley, L. (2020, October 9). Ontario experiencing spike in new COVID-19 ICU admissions not seen since June, data shows — CBC News. CBCnews. Retrieved November 4, 2021, from https://www.cbc.ca/news/canada/toronto/ontario-experiencing-spike-in-new-covid-19-icu-admissions-not-seen-since-june-data-shows-1.5755925.

Ran, J., Zhao, S., Han, L., Ge, Y., Chong, M. K. C., Cao, W., & Sun, S. (2021, July 1). Increase in diabetes mortality associated with covid-19 pandemic in the U.S. Diabetes Care. Retrieved November 1, 2021, from https://care.diabetesjournals.org/content/44/7/e146.

Reshaping and pivot tables. pandas. (n.d.). Retrieved December 13, 2021, from https://pandas.pydata.org/pandas-docs/stable/user_guide/reshaping.html.

Stojiljković, M. (2021, March 19). Linear regression in python. Real Python. Retrieved December 13, 2021, from https://realpython.com/linear-regression-in-python/.

Stojiljković, M. (2021, March 15). NumPy, scipy, and pandas: Correlation with python. Real Python. Retrieved December 13, 2021, from https://realpython.com/numpy-scipy-pandas-correlation-python/.

Swaminathan, S. (n.d.). Data Visualization with Python. Coursera. Retrieved November 1, 2021, from https://www.coursera.org/learn/python-for-data-visualization?specialization=ibm-data-analyst.

The NumPy community. (2021, June 22). NumPy: the absolute basics for beginners. NumPy. Retrieved November 4, 2021, from https://numpy.org/doc/stable/user/absolute_beginners.html.

The pandas development team. (n.d.). DataFrame. pandas 1.3.5 documentation. Retrieved December 3, 2021, from https://pandas.pydata.org/docs/reference/frame.html.

Time Series / date functionality. pandas. (n.d.). Retrieved December 13, 2021, from https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#dateoffset-objects.

What is the difference between correlation and linear regression? GraphPad. (2019, October 3). Retrieved December 13, 2021, from https://www.graphpad.com/support/faq/what-is-the-difference-between-correlation-and-linear-regression/.