Relationship Between Earthquake Magnitude and Geographical Location

Team 18
Team Members: Chloe Lai, Kexin Zhou, Lomani Mullikin
GitHub: https://github.com/anncy0413/COP3530-Final-Project.git
Video Demo: https://www.youtube.com/watch?v=kIQmCJfAzFc

Final Project Proposal

The problem we are trying to solve is to determine the trends of the relationship between the earthquake magnitude and geographical location to help determine trends of earthquakes and location. Researchers rely on probabilities based on historical earthquake data to help determine common geographical locations that experience earthquakes and the frequencies of magnitudes in that area. We can then better prepare city infrastructure and resources based on probabilities of higher magnitude earthquakes. By being able to look at trends over time on the magnitudes of earthquakes, we can determine if a certain location is of higher risk of being hit with a higher magnitude than others.

The public data set we will be using is the CORGIS Datasets Project that consists of our earthquake data with information on the magnitude, location (latitude, longitude, country), and even the time. Tools and libraries that we've used include PANDAS, Matplotlib, and React. We will be using both Python and Javascript to help with our analysis and website implementation. The main features we will be implementing is filtering the earthquake data which takes user input and implements a map visualization of the magnitude of the earthquake and location. It will also have another tab where you can look at a time series chart that shows the date and magnitude of an earthquake.

The main algorithms we are comparing are merge sort and quicksort. These sorting algorithms will be used when filtering through our entire dataset based on what the user selects on what range of magnitude or locations. We are also using a tuple as a data structure that holds

our information that our filtering algorithm has. To follow this, a filtering algorithm is utilized to organize data by location, picking out only the data with the specified location from the row. Aggregation is then used for counting the frequency of earthquakes (which is grouped by month and day) at the given location once the table is sorted into a smaller one. The two algorithms aforementioned (filtering and aggregation) are what works to return the data to plot the for the line chart visuals.

Chloe will be in charge of data handling/filtrations, where she will take in the earthquake data and be able to filter through a given input. Kexin will be in charge of visualization - this includes the map interface and the selection process for the user. Lastly, Lomani is in charge of data analysis by creating charts on earthquake data. However, all parts of the project will still be worked and discussed together.

Analysis

Our initial idea was to represent and predict earthquake magnitude based on longitude and latitude. However, we then decided instead to base it on geographical location to observe frequency of the magnitudes at a location over time. Another change that we made was including Javascript in our code. Originally, we had planned on only using Python, but since one of our members knew how to use Javascript and create the website visual, we were able to incorporate that into our project.

The function for my merge sort consists of two functions: merge and alphabeticalSort. The overall time complexity would be O(n*log(n)), where n is the number of elements in the given list of tuples that consists of our earthquake data. We are splitting our function into half before merging it to a full sorted list, with each of our levels requiring a time complexity of O(n) and the number of levels being only O(log(n)).

The other sort we are comparing it with is quick sort, which consists of the two functions partition and magnitudeSort. The average time complexity of using this quicksort in our algorithm would be O(n*log(n)). N is the number of elements in the given list of tuples that consists of our earthquake data that we have filtered. The time complexity of this algorithm depends on if the pivot we choose: if the pivot we chose consistently is able to split the list/array, then we would get an average complexity of O(n*log(n)). In the code we wrote, the pivot is chosen based on the first element at random, and so the time complexity of O(n*log(n)) applies. As a result, between the two sorts, the time complexity is the same in how we utilize our data.

Lastly, the analyzeFrequency function. This function is heavily dependent on the CSV file that is being sorted through, the read time is a time complexity of O(n) (the same as filtering), affected directly by the number of rows and columns. The groupby() operation in the function is a logarithmic factor to the overall time complexity, this is based on the number of unique groups that must be made (dates in terms of month and day). If the number of unique dates is very close to the number of rows, the worst case time complexity will be O(n*log(n)). Both the string and plotting operations are linear, in direct congruence with the data provided, the amount of rows for the string, and in the case of plotting, the number of data points to plot. So, in turn, the worst case time complexity as well as the average will happen to be around O(n*log(n)).

Reflection

Through this project, we were able to learn how to use GitHub in a more collaborative environment, such as branching, pushing, pulling, and committing. While we had a lot of struggles in finding the right project that we would all be willing to work on and knowing what to analyze, we were able to come to a compromise at the end. As a group, we've learned a lot

about sorting algorithms, front-end visualizations, and understanding how to use different data analytical libraries in Python that we had never really used prior.

One of the major challenges we had as a group was deciding how to split the work up. Each of us had very different backgrounds when it comes to programming and what our strengths were. There were many different moving parts in the project, so deciding how to split the tasks was difficult. There were times where one group member seemed to be doing more work because they had more experience with a certain software or code, but we all tried to put our best foot forward and help each other as much as possible.

If we were to start again as a group, one change we would make would be to start off with a more specific strategy on what our goal of the project was. Our group didn't really decide how we wanted to go about the project until we were actually implementing it, and that was when we were able to firmly decide what we wanted to accomplish. Additionally, we would try to streamline the workflow more efficiently by laying out what needs to be done more specifically. In the current project, we only listed larger tasks to accomplish, which made it difficult for us to understand what we needed to do.

Chloe: Through this project, I learned more about the sorting algorithms that we implemented. Additionally, I also learned how to delegate tasks among the members through understanding what each of our strengths were and what we were most interested in. However, one thing I would like to change would most likely to foster more open communication in terms of strategy on how to approach our project since it was vague and made it harder to work.

Lomani: For me, this project was a great way to get a lot more experience with sorting algorithms and how we would actually use them in a real project as it was not something I could think to immediately implement. Working with everyone, especially in the space of Github like

this was very unique, and I would say personally, helped me greatly when struggling with the scope of the project. Similar to what Chloe stated, understanding just what we are trying to accomplish was difficult at the start, and the use of Github made that easier, as there were clear tasks to complete. Coming out of this project I learned a lot and am more confident in my ability to use algorithms and work in a team and really hope to improve these skills in the future.

Kexin: Working on the visualization aspect of this project helped me better understand how to bridge the gap between raw data and users. I was responsible for implementing the interactive map interface using React, allowing users to filter and explore earthquake data by location and magnitude. Through this, I learned how to handle frontend data rendering based on user input and integrate visualization tools with Python backend logic. One major challenge was ensuring the responsiveness and clarity of the visual components, especially when dealing with different ranges of magnitudes and varying data densities. In the future, I'd like to improve the user experience further by adding tooltips and animations to make the data exploration more intuitive. Overall, this project strengthened my skills in frontend development, user interface design, and collaborative coding in a team environment.

References

CORGIS Datasets Project

Matplotlib — Visualization with Python

Pyplot tutorial — Matplotlib 3.10.1 documentation

Clustering in Machine Learning - GeeksforGeeks

Can you predict earthquakes? | U.S. Geological Survey

pandas - Python Data Analysis Library

React