Personal Reflection Report: Accident-Analyzer

Anudeep Reddy Dasari
adasari4@asu.edu

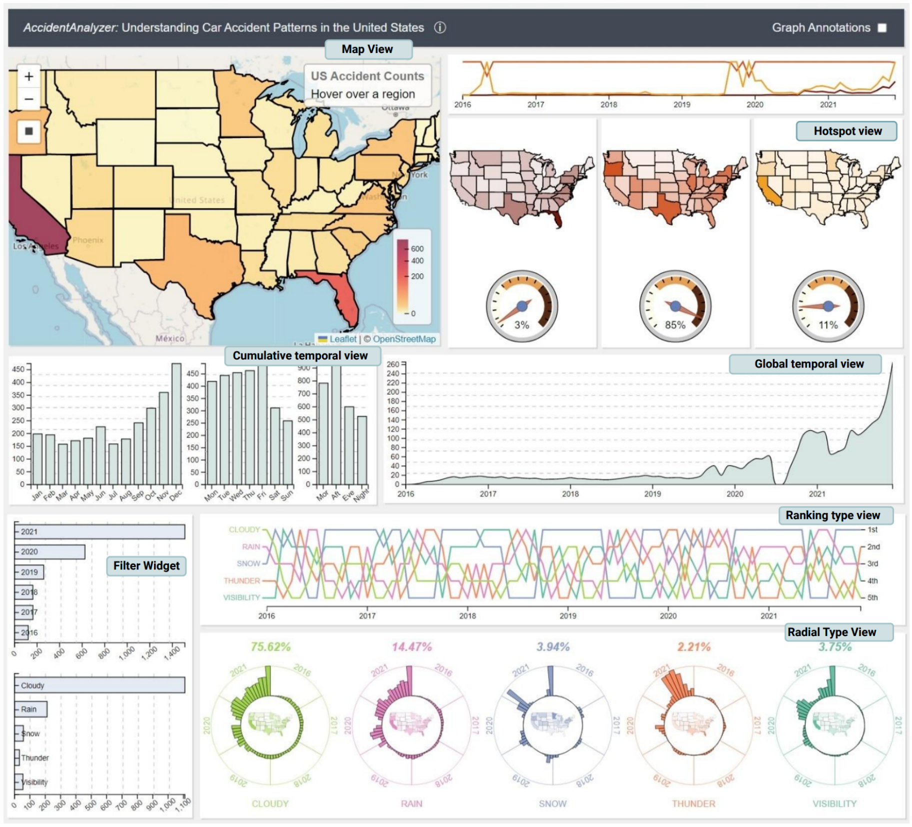


Figure 1. An overview of *Accident-Analyzer*

1 Introduction

In this project, my team and I implemented Accident-analyzer, a modified version of CrimAnalyzer[1] (a visualization that analyzes the crimes in Sao Paulo city) using the accident data from the United States instead of crime data like CrimAnalyzer. Accident-Analyzer can be used to study accidents in the United States using various filters like year, month, time of day, and weather conditions. It can also be used to find hotspots i.e the regions with a higher density of accidents. The visualization was made mostly with the JavaScript tools D3.js (v7) and Leaflet.js, the dataset preprocessing was done with Python, and the data is stored/accessed via a MySQL database.

This system can be used by anyone who wishes to study and analyze accident patterns in the United States. It can be used by the general public to avoid places with a high density of accidents. I can be used by the government to better understand how different laws and regulations in different places affect accident patterns.

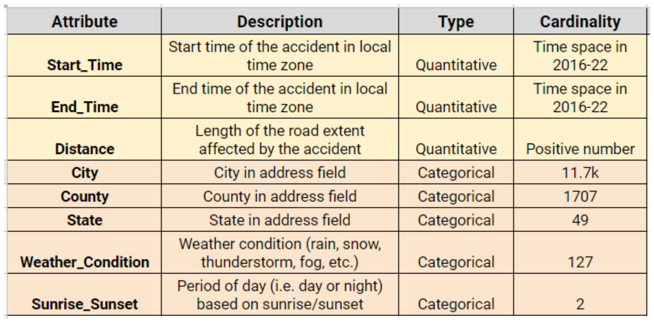
2 Explanation of the solution

2.1 Dataset description

For the purpose of this project me and my teammates used US accident data [2][3] found on Kaggle. This dataset consists of over 2.8 million rows with 47 attributes that provide a detailed overview of the accident. Although all the attributes provide some important information in their own right, for the sake of simplicity and limitations in computational power we narrowed down attributes to the ones which we feel would provide a greater value compared to others. The attributes that we filtered down are Start_time, End_time, Distance, City, County, State, Weather_condition, and Sunrise_sunset. A brief description of the data can be found in the table in Fig 2.

2.2 Functionalities

This section will detail the activities I anticipate users will be able to do by seeing and/or engaging with the traffic accident data in Accident-Analyzer.



Attribute	Description	Type	Cardinality
Start_Time	Start time of the accident in local time zone	Quantitative	Time space in 2016-22
End_Time	End time of the accident in local time zone	Quantitative	Time space in 2016-22
Distance	Length of the road extent affected by the accident	Quantitative	Positive number
City	City in address field	Categorical	11.7k
County	County in address field	Categorical	1707
State	State in address field	Categorical	49
Weather_Condition	Weather condition (rain, snow, thunderstorm, fog, etc.)	Categorical	127
Sunrise_Sunset	Period of day (i.e. day or night) based on sunrise/sunset	Categorical	2

Figure 2. Description of the Data

- **Interactive selections**- the user can interactively select to filter data, like selecting a specific county or filtering data by time or weather condition
- **Accident patterns over the country** - the user will be able to see how accident densities change from place to place
- **Dynamic of accidents over time** - the user will be able to see how accidents change over time.

2.3 Implementation

The implementation of this project contains two major sections Frontend and backend.

2.3.1 Frontend. This part of the project focuses on how the visualization looks and functions. We used HTML, CSS, and JavaScript to implement the frontend part. Specifically, we use D3 (a JavaScript library for producing dynamic, interactive data visualizations in web browsers) for building all the visualizations. HTML and CSS are used to make the system look pretty while Javascript handles all the functionality.

2.3.2 Backend. This part of the project focuses on preprocessing of the data and hosting the system. For this purpose, we used MySQL and PHP. MySQL(an open-source relational database management system) is used to store, view and filter data and PHP(a server scripting language used for making dynamic and interactive Web pages) acts as a middleman between JavaScript and MySQL fetching data from JavaScript and passing it to MySQL and vice versa.

3 Description of the results

In this section, we discuss how the functionalities described in the above section are implemented. We discuss in detail the different views which we have implemented. There are a total of 7 views as shown in Fig.1. A brief description of all the views is given below.

3.1 Map View

In this view, we implemented a choropleth map(a map that uses differences in shading, coloring, or the placing of symbols within predefined areas to indicate the average values of a property or quantity in those areas) using leaflet.js (an open-source JavaScript library for mobile-friendly interactive maps). This view has the following features and interactions.

- **Region selection** For this interaction user can select a particular state on the map and the region gets highlighted and zooms in to provide a county-level overview, consequently, the data is filtered based on the selection and all the other views get updated to reflect this change.
- **Rectangular selection** This feature is similar to the above feature except instead of selecting a specific state the user can click and draw a rectangle to select multiple states in a single interaction with the system.
- **Tooltip** Furthermore, when the user hovers over any state the tooltip gets updated to reflect the change and display the respective state's name and the accident number.

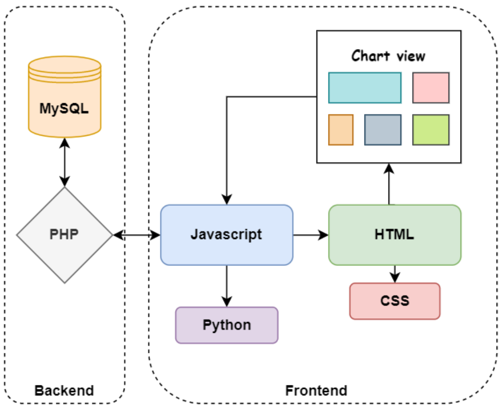


Figure 3. Flowchart of the system

3.2 Hotspot View

This view shows a collection of three hotspots, one hotspot highlights the regions where the accidents occur in high number and frequently, the second one shows the regions where the accidents occur in low numbers but more frequently, the third one shows the regions where accidents occur in high number but less frequently. We calculated these hotspots using the Non-negative matrix factorization(NMF)[4] algorithm in python and used it in Javascript using the Pyodide library. In addition to this map view, we also implemented an extension that shows the temporal distribution of hotspots i.e the distribution of hotspots over time.

3.3 Global Temporal View

This view shows a time-series area chart that shows the number of accidents that occurred over time. The user can also drag and select a period of time to focus on that specific time period and all the other views get updated accordingly.

3.4 Cumulative Temporal View

This view shows the distribution of accidents over different months, days of the week, and times of the day in the form of a bar chart. This view also acts as a filter, the users can select a bar to filter the data by that particular attribute.

3.5 Ranking Type View

This view shows a cluster of poly-lines each representing a corresponding weather condition and the vertical position of the line represents the severity of the accident. This severity is calculated by calculating how long the road was closed. The user can also hover over a line to highlight it.

3.6 Radial Type View

This view shows a group of bar charts using a radial layout. Each chart is organized in such a way that the bars are grouped by years and each chart represents a specific weather condition and the number on the top represents the percentage of accidents caused by that weather condition.

3.7 Filter Widget

This view shows two bar charts. The top one shows the distribution of accidents over the years 2016 to 2021 and the bottom one shows the distribution of accidents with respect to the 5 weather conditions. This view also acts as a filter widget in the sense that users can select multiple years or weather conditions and the data gets filtered based on selection. This action also updates all the other views based on filtered data.

4 Personal Contributions

My main role in this project was to build the Filter Widget, Radial Type view, and the months part of the Cumulative temporal view alongside my teammate Hruthik. Along with this I also played a key role in writing the report and designing the poster.

4.1 Filter Widget

As mentioned in the above sections this view contains two bar charts that act as filters for years and weather conditions. My task was to construct both the horizontal bar charts and add the filter functionality to them. The first part to implement was querying the original data to build the bar charts. I used SQL to query the data and JavaScript's' D3.js library to build the bar chart. Furthermore, I had to add functionality to the chart such that whenever the user selects a particular region in the map view the chart gets updated. For this purpose, I implemented a function that changes the data and redraws the chart every time a selection or filtering is done in other views.
For example, when the user selects California, the data is updated such that it only contains the data related to California and the chart is redrawn to reflect the filtering using the updated data. Similarly, if the user selects a particular month or day of the week the data gets updated and the graph is redrawn to reflect those changes. Furthermore, I implemented the filtering functionality i.e when the user selects multiple years or weather conditions, these selections are stored in a variable and based on these selections the data is updated and all the other views are also updated based on these changes. Additionally, I added a transition functionality so that whenever the view is updated instead of drawing the graph from scratch, the previous graph is transitioned into the new graph.

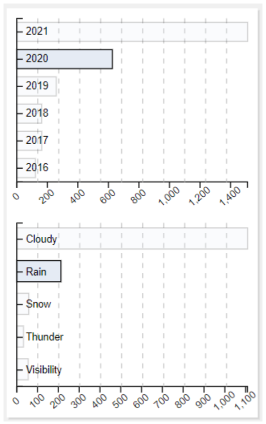


Figure 4. Filter Widget

4.2 Radial Type View

As mentioned above, radial-type view is a collection of map views surrounded by bar charts in a radial fashion. This view has 5 circular bar charts each representing a separate weather condition.
In this view, my task was to implement the circular bar plot including the data filtration and data processing as per the visualization. In order to implement the visualization, I have followed the following approach- Firstly, with the help of SQL queries, I have obtained the data in the grouped format, and the grouping of the data here is done on the basis of weather conditions. Secondly, after the data is filtered, I implemented the choropleth maps for each group using the d3.js functions such as geopath and geoMercator. Finally, I have then filtered the above-filtered data for every year and every month in order to implement the bar charts surrounding the choropleth maps for each weather condition with the help of the d3.js library. I grouped the bar charts using years and gave an appropriate color coding to represent each weather condition such that it matches the color coding in the Ranking type view.

4.3 Cumulative Temporal view

I have also been tasked with implementing the third chart in this view. I implemented a bar chart that shows how accidents are distributed based on the time of the day. For this

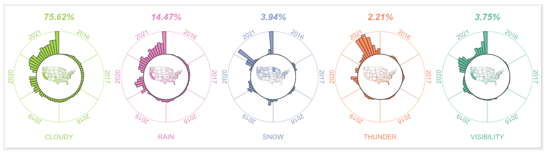


Figure 5. Radial Type View

purpose, I first used SQL to query the data based on the time of the day and implemented the graph using d3.js. In addition to this I have also added a filtering functionality so that whenever the user selects a particular time of the day, all the views get updated. Furthermore, I also added gridlines to the chart to get a better sense of the y- axis values.

5 Lessons learned

I have detailed the lessons I have learned myself and from my teammates during the course of the project here.

- Firstly, I learned how to collaborate with people on version control systems like git and how to modify, commit, and submit pull requests in GitHub
- I also learned how to organize a project and divide the deliverables among the team members to work efficiently and deliver results on time
- I also learned how to set up a system using a database like MySQL and use PHP to query data to and fro from frontend and backend.
- I also learned in depth about many d3 functionalities including how to build choropleth maps and circular bar charts. Furthermore, I learned how python functions can be implemented inside Javascript using Pyodide library

6 Team members

- Kyle Otstot
- Jaswanth Reddy Tokala
- Shreyash Gade
- Hruthik Reddy Sunnapu
- Nitesh Valluru

References

[1] Garcıa, Germain and Silveira, Jaqueline and Poco, Jorge and Paiva, Afonso and Nery, Marcelo Batista and Silva, Claudio T. and Adorno, Sérgio and Nonato, Luis Gustavo, *CrimAnalyzer: Understanding Crime Patterns in São Paulo.* IEEE Transactions on Visualization and Computer Graphics, 2021

[2] Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. *A Countrywide Traffic Accident Dataset.* 2019

[3] Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath, *Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights.* In proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019

[4] H. Kim and H. Park, *Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis.* Bioinformatics, 2007