GeoVis: an interactive map to visualize common climate indicators

MUHAMMAD ALAFIFI[1,2] AND JIAWEI WU[1,3]

[1] Computer Science Department, Rensselaer Polytechnic Institute, 1999 Burdett Ave, Troy NY, 12180
[2] email: alafim@rpi.edu
[3] email: wuj22@rpi.edu

Compiled April 21, 2024

Geovis is a map rendering of the United States that aims to provide climate enthusiasts or curious individuals with an interactive webpage to visualize common climate indicators such as wind speed, precipitation, and temperature. With the increase in greenhouse gas emissions around the world, it is crucial to visualize such trends that leverages ones understand in climate change. To compile this project, we utilized datasets from Kaggle. Our visualization garnered much important feedback, with many feedback being appreciative of our minimalistic and intuitive design. We found that many testers were curious at our web application because they were found to be clicking on multiple things for the sake of visualizing the data.

1. INTRODUCTION AND BACKGROUND

We are both climate enthusiasts, which is why we decided to pursue a project that was closely related to visualizing climate. Our intention of creating this project was to provide a platform for individuals to visualize common climate indicators such as wind speed, precipitation, and temperature. We believe that with the increase in greenhouse gas emissions around the world, it is crucial to visualize such trends that leverages ones understand in climate change. Creating a map visualization was what we believed to be the most intuitive way to visualize data, we were inspired by wind fm and earth nullschool. These sites provide the user with great flexibility in manipulating and showing specific climate indicators, something that we took inspiration in creating our own.

2. DATA COLLECTION

To compile this project, we primarily utilized datasets of type csv. There are two types of data that we had to process, one is for every airport, and another for every county.

A. Airport Data

Utilizing airports was the most intuitive way to gather weather data. Initially we had planned to use large global datasets, such as NOAA ICOADS, but we found it expensive, given the size of the dataset being 31 million rows and 75 columns. Not only that, but the points that were collected over the United States were so abundant, that it was difficult to visualize and also hindered our efforts to make it interactive. Instead, we pivoted over to a dataset that was more manageable, which was the weather data for 265 airports in the United States. This dataset was much more manageable, with 265 airports. Airports resembled a cleaner visualization as well, as some states have at most 3 major airports, making it simple to visualize the difference.

Processing the data was by no means simple. There wasn't a dataset that was "perfect" for our project, so we had to look for multiple datasets. The datasets we found was always missing some key features. Our airport weather data, while it was useful in providing us with the weather data, it was missing the latitude and longitude of the airports. We had to find another dataset that provided us with the latitude and longitude of the airports. We had to merge these two datasets together to create a dataset that had both the weather data and the latitude and longitude of the airports. However, it would be simple if we could just merge the two datasets together to create one, but we ran into one problem. The airport dataset providing us with climate information contained only 265 airports, while our other dataset contained over 3375 datapoints, which are all the airports in the United States.

Our final dataset came out to contain the following columns: **Month, Day, Temperature (F), Dew Point (F), Humidity (%), Wind Speed (mph), Pressure (in), Precipitation (in), IATA code, Name of airport, Longitude, Latitude**. With all this data, we were able to begin creating our intended visualization.

B. County Data

On top of point by point data for major cities and regions, we desired to have a filled or colored in map of the weather data over time. The goal was to have a climate statistics map as a user could easily recognize from weather channels. Initially, we were also looking at the NOAA's NOMADS data set, which contained point data as accurate as a quarter of a degree of longitude and latitude, but several issues arose. Given the data was broken down by every 6 hours and by different levels of altitude, it gave way to issues regarding compiling time and location values. However, more presciently, we ran into the difficulty of formatting the data from the raw GRIB2 (Gridded Binary) format, which is used by the national departments to compactly plot weather data, to a usable CSV or JSON file format,

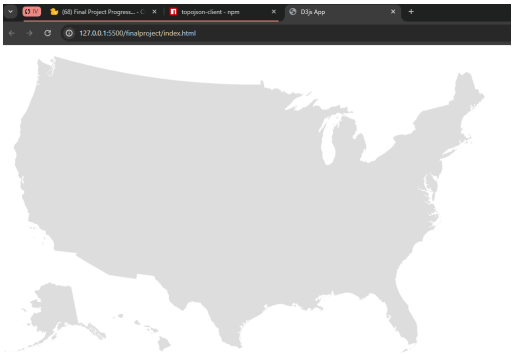

Fig. 1. Our first map rendering of the United States

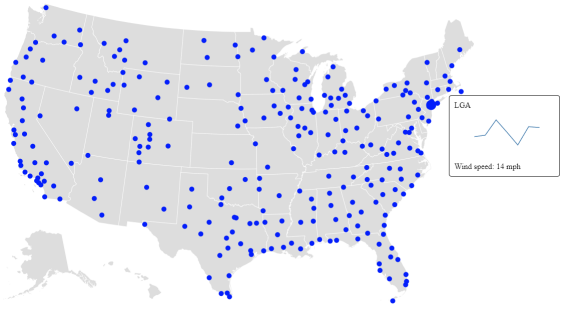

Fig. 2. Plotting the airports on the US map

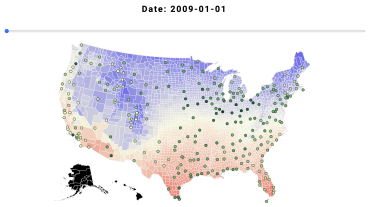

Fig. 3. Implementing the time slider

which would have required a lot more time building a constant data translator and using more unused languages/softwares for us such as Perl. Instead, we chose to use county data, as city data won't be as complete, to represent the map data.

Working with county data would be much more manageable given the limited amount of data, given there are 3,143 counties in the United States. The NCEI contained appropriate CSV county data sets broken up by month and state, which contained different types of weather statistics from average temperature to precipitation. We chose to use those as the two major map data sets given that they could can be contrasted well on each other and along with the airports' weather parameters. The first measure of business was having to manually download the CSV data for each state. When looking at our map data set and temperature data set, we realized that the map data set used the FIPS county code for its set, so - using python - we compiled the states all together in one file and another python script to convert the counties into a usable FIPS code. Finally, the last issue to deal with that was breaking the data set was county names that contained commas or illegal characters for the csv, this meant manually running through every file and editing them out.

3. VISUALIZATION AND PROGRESS

For our visualization, our intention was to create a map of the United States that allows individuals to be able to access climate data in our chosen airports.

A. Progress Report 1: Map Rendering

Map rendering is a key feature of our project. Rendering a map of the United States allowed us to begin learning the D3 stack and also envision the data that we will need to plot the points onto the map. We use topojson and follow this D3 guide to create and render our intended US map.

While topojson is our choice, it is not the only way to render a map. We also considered using geojson, but we found that topojson was more efficient in terms of file size, that is to say, they claim that their file sizes are over 80% smaller than geojson files, as claimed on their Github project description.

B. Progress Report 2: Plotting the Airports

Fig 1 provides us with a starting point to plot the airports. For our dataset, we have information like the latitude and longitude of the airports, which is crucial for plotting them in topojson. Fig 2 shows the 265 airports all plotted out onto our map of the United States. An additional feature that we included was a slight hover effect. Hovering over each airport would enlarge the airport point, this decision was done so that it would be easier for users to see which airport they are hovering over. At this point, if you decide to select on an airport by clicking on it, it would display information about that airport. More specifically, the airport name, the wind speed for that airport, and also a line chart to show the trend of the wind speed over the last 7 days.

While this provides a starting point to what we want to do, there are still many things missing. For example, the line chart that we have included for each point's tooltips was unlabeled, it had no axis, making it difficult for the user to intuitively understand what chart is displaying. Not only that, but the points are all colored blue, which is not very intuitive. We need to color the points based on the climate indicator that we want to display.

C. Progress Report 3: Implementing the Time Slider

Time slider implementation was a crucial feature that we wanted to implement. We wanted to allow users to be able to select a specific date and see the climate data for that specific date. Fig 3 shows the implementation of the time slider. The time slider allows users to select a specific date, and the points on the map would update to show the climate data for that specific date. This was done on client-side rendering with an event listener.

Algorithm 1. Time Slider Implementation

```
1: procedure TIMESLIDER
2:     selectedDate ← sliderValueToDate(this.value)
3:     updateDisplayedDate(selectedDate)
4:     updateVisualization(selectedDate)
```

This algorithm is called with a 'onchange' event listener, which is triggered whenever the user changes the value of this slider. To remove excessive rendering, we implemented the slider in a way that it only re-renders when you release the

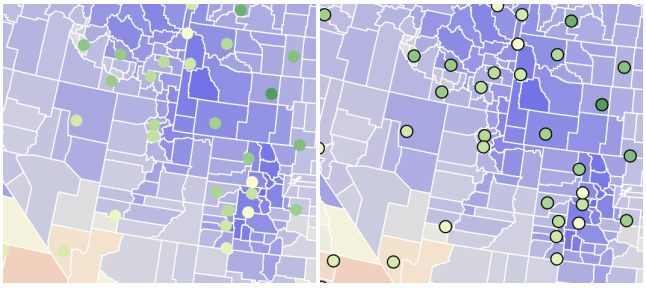

Fig. 4. Color choice on the airports based on wind speed

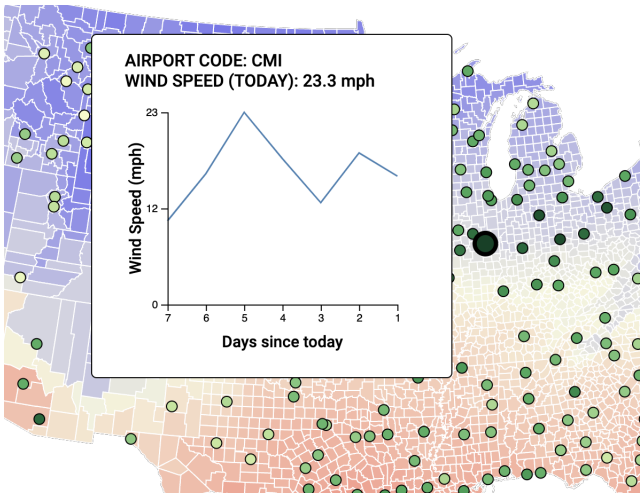

Fig. 5. Tooltip improvements

point, not when you move it along the slider. We made this decision because our dataset is about 70mb in total, and re-rendering the data when you slide along 365 days caused some delays and lags in rendering. So rendering it only when you release the point was a good compromise.

Fig 4 shows the choice made in coloring each airport. We decided to select a sequential green hue coloring for our color representation. On the left side, we have filled the circle with the color respective to their wind speed, however, for the lighter colors such as on the south west side, it became more difficult to distinguish between the point and the background. The choice that we decided to make to fix this issue was by implementing a border. The effects of the border is shown in the right image, which makes it easier to distinguish between the points and the background of the map.

Fig 5 shows the improved tooltip. It shows you the airport IATA code, the wind speed for the day, and also the improvement line chart from progress update 2. Since it was a line chart, we made the decision to flip the x-axis so it would easier to visualize the trend, with the right side being the nearest day, and the left side being the further day.

D. Final Visualization

For our final visualization, we included not only a legend for the points, but also a toggle for users to switch between the climate indicators such as temperature, dew point, and wind speed. Fig 6 shows the final implementation. We have the time slider on the top, the legend for the airport on the bottom right, and a stats box that on the bottom left, indicating the highest selected climate indicator for that day, along with the location in which it occurred as shown in Fig 7.

In addition, one of the crucial improvements that we wanted to implement was the tooltip for each of the airports. Our final tooltip (Fig 8) shows the full name of the airport that you selected. This felt more intutiive in the audience knowing which airport it is, given that airports with an unfamiliar airport code were unknown to users. In addition, since we had data on all the climate indicators, we thought it would also be the best to include statistics on all the weather indicators, and the line chart should only show the climate indicator selected. In Fig 8, temperature was selected as the toggle for Point Information, and the line chart reflects by graphing the temperature data for the past 7 days.

4. FEEDBACK

A. Jiawei's Feedback

I was fortunate enough to receive a lot of useful feedback. My question during the feedback process was to ask about what should be placed in the blank spaces in the webpage, as the left and right sides were mostly empty and we could implement some sort of left and right rail to fill up to gaps. Our most common feedback was to implement a stats box that allows the user to view some of the highest climate indicators for that day. I thought this was a great idea, which is why we implemented this into our application for the final version.

A legend for the wind speed was also important, many users were confused about what the colors of the points represented, some of them believed the colors are indications of how "large" an airport is, with darker colors meaning a larger airport, and lighter colors meaning a smaller airport. This was a great suggestion, and we implemented this by adding a legend for our points.

Another suggestion was to add a toggle for the climate indicators. Some people may not only be interested in wind speed, but also other types of climate indicators. This was a great suggestion, mostly because we already had the data for other climate indicators, so it would be simple to implement it onto our application. We went with a toggle version to select different climate indicators. To do so, we needed to improve our python script to include the other climate indicators and create a new final dataset that includes all the climate indicators along with their respective values.

We were not able to implement all the feedback that we received. One of the important feedbacks that we got was to find a way to color the points to represent which airport is classified as a "large airport" and "medium airport". Since we are using colors of the points to represent the values for the chosen climate indicator, it was difficult to find a common ground on trying to represent the size of the airport. Some of the thoughts that we considered was making it a different shape, perhaps large airports would be a triangle instead of a circle. But this was never added to our final visualization.

Another feedback that we attempted to implement was a comparison between dates. Users could select a specific date, and then select another date, and the points would update to show the difference between the two dates. This was a great suggestion, but we were not able to implement this feature due to time constraints and also difficult bugs that we struggled to fix, so it never made it to our final visualization.

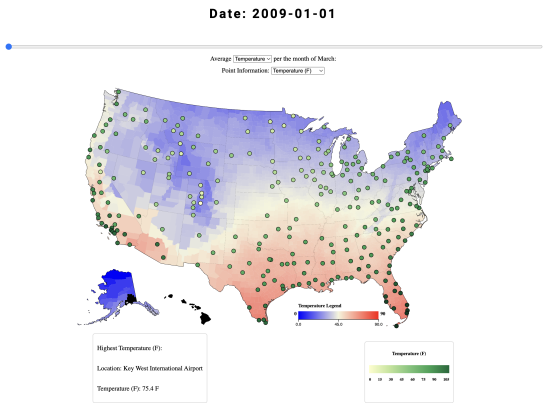

Fig. 6. Final Visualization

Highest Pressure (in):

Location: Nantucket Memorial Airport

Location: Logan International Airport

Location: Ronald Reagan Washington National Airport

Location: Coastal Carolina Regional Airport

Location: Newark Liberty International Airport

Location: John F Kennedy International Airport

Location: La Guardia Airport

Location: Norfolk International Airport

Location: Southwest Oregon Regional Airport

Location: Philadelphia International Airport

Pressure (in): 30.2 in

Fig. 7. Multiple Airports have the Highest Pressure for the selected date

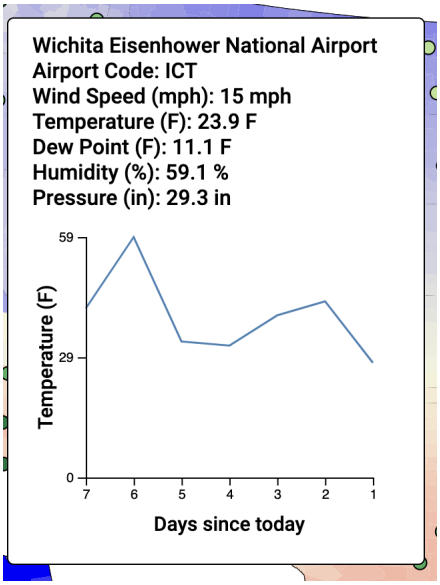

Fig. 8. Final tooltip design

5. CHALLENGES

Data collection was definitely one of the more significant barriers to our success. We struggled with finding a dataset with all the information that we needed, so we ended up having to combine different datasets for the airport data.

Stack also posed a challenge for us. We were not familiar with the D3 stack, and we had to rely heavily on trying to map the examples given on the D3 gallery to our own project.

Time was also a significant challenge. Our initial proposal was a similar idea, but we wanted to apply it globally. We would first render a map of the entire globe with scrolling and zooming in features, and then plot the points as seen in the United States map. However, we quickly realized that the data collection was going to be exponentially more difficult and time-consuming. We struggled to find the data that we needed. We later realized after a week that this was not going to be feasible given the time that we had, so we pivoted to a more manageable dataset and focused instead on the United States.

6. WORK DISTRIBUTION

A. Muhammad Alafifi

B. Jiawei Wu

For our visualization, I was responsible for gathering and processing all the datasets relating to the airports as well as the climate data on each airport. I was responsible for implementing the time slider and its functionalities, creating the rendering of the US map, plotting the airports in their respective location, selecting colors for their respective climate indicators, and creating the pop-up window that shows the climate data of each airport. I was also responsible for creating the legend that shows the range of values for each climate indicator. In addition, I also implemented a selection for the climate indicators for the airport points for users to switch between temperature, dew point, wind speed, humidity, and air pressure.

For our project write up, I was responsible for writing the abstract, [1] introduction and background, [2a] data collection for the airport data. I was responsible for writing the progress

report for anything relating to aiport data and visualization with the points.

7. CITATIONS