CSE 60838 Data Visualization Final Project Report
Zeyuan Li

A Visualization of US COVID-19 Statistics

1. What is your project about?

This project's purpose is to display COVID-19 statistics in the US and trends of confirmed cases at a state level in an interactive visualization program. On various sites, there have been some decent hot-spot maps indicating number of cumulative cases with a legend. But only few of them have an animated feature to indicate the trend by coloring the state to represent the number of cumulative confirmed cases on a daily basis, which would be quite intuitive if people want to check trends of reported cases in a certain period of time. Thus, I wanted to retrieve the pandemic data and create an animated geographic visualization along with other graphs.

The end product I created was a visualization program that has two main parts. Firstly, it has a geographic visualization that includes an animated map of the US that allows the user to view a snapshot of the US confirmed cases of COVID-19 at a state level. The second part allows the user to view trends and facts created by other commonly-used visualization techniques. It consists of three charts: a 3D bar chart that has the top 20 states by confirmed cases, a curve graph that shows trends in a single week, and a Nightingale's Rose Chart.

The data I used for this visualization are in two categories: 1) the COVID-19 US state-level statistics (confirmed cases, deaths, date) from 1/21/2020 to 12/6/2022; 2) a file containing US state-level map data in TopoJSON format. Initially I wanted to try retrieving the pandemic data from web API, but I ended up not finding suitable API and turned to another approach; I retrieved the U.S. State-Level Data in csv format (us-states.csv) from a COVID-19 U.S. data repository on Github (https://github.com/nytimes/covid-19-data). This repository and its content are licensed by The New York Times and it's available for broad, noncommercial public use. For this project, I only retrieved the historical U.S. state-level data in csv format.

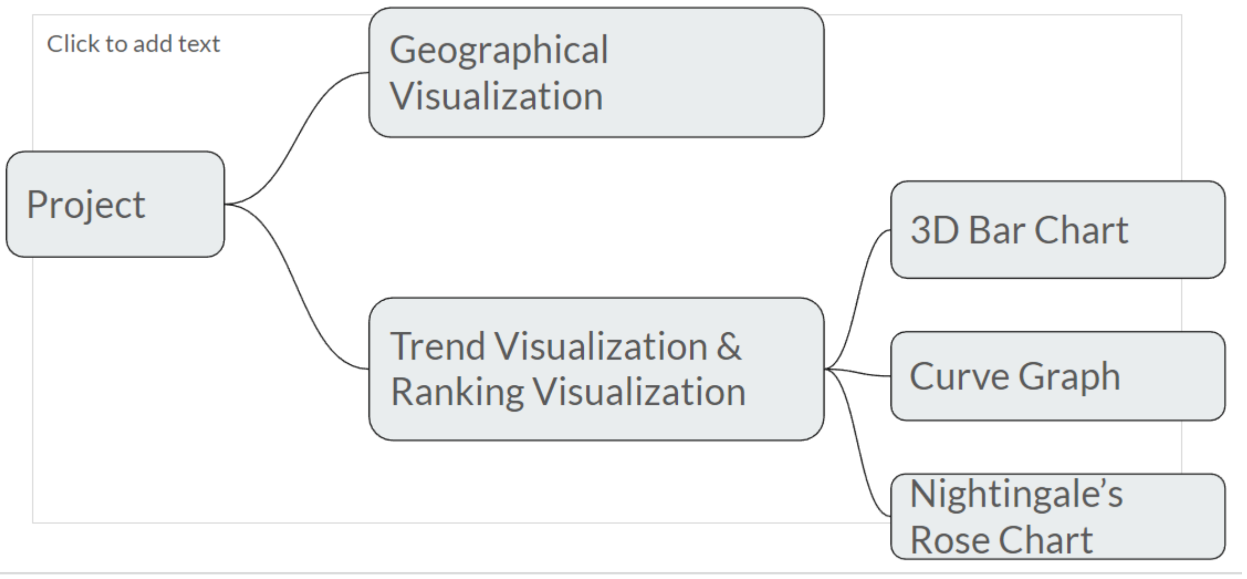
After that, I got the map data from a TopoJSON file from a public source on the web. A TopoJSON file format is a geoJSON format extension that encodes geospatial topology. Here's a screenshot of the TopoJSON file. This file is in 5m resolution and it features all US States. The author of it converted shapefiles from the Unites States Census Cartographic Boundary Files to GeoJSON and KML formats using a vector converter.

This is the web URL for the website where I retrieved open-source US map data as well as the URL for the JSON file itself:

https://eric.clst.org/tech/usgeojson/

https://eric.clst.org/assets/wiki/uploads/Stuff/gz_2010_us_040_00_5m.json

2. What are the main functions you implemented?

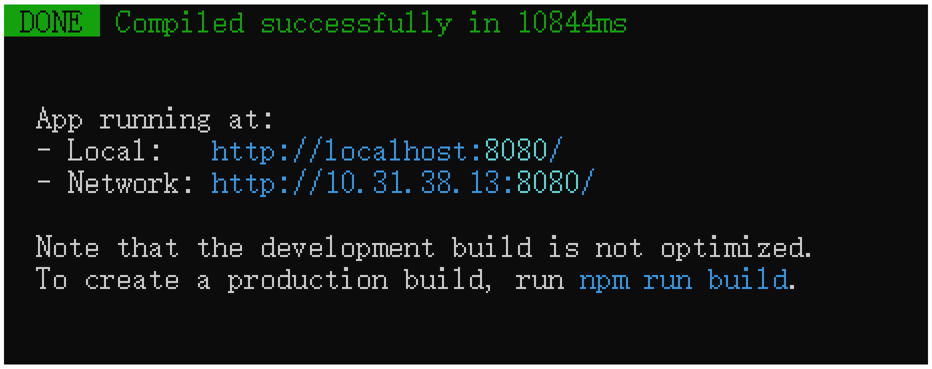


- Animated geographic visualization of COVID-19 statistics a US map
  - An animation triggered when opening up the program
  - Tooltips for each and every state
  - A legend with colors ranged by the confirmed cases
- 3D bar chart
  - Top 20 states by confirmed cases
  - Animated 3D bars
  - Tooltips for each bar
- Curve graph
  - Trends of confirmed cases
  - A dropdown menu for trends of cases in another state
- Nightingale's Rose Chart
  - A typical Rose Chart, with the most recent data of COVID-19 cases in the US

3. How to run your program?

The program was developed using Node.JS v12.12.0.

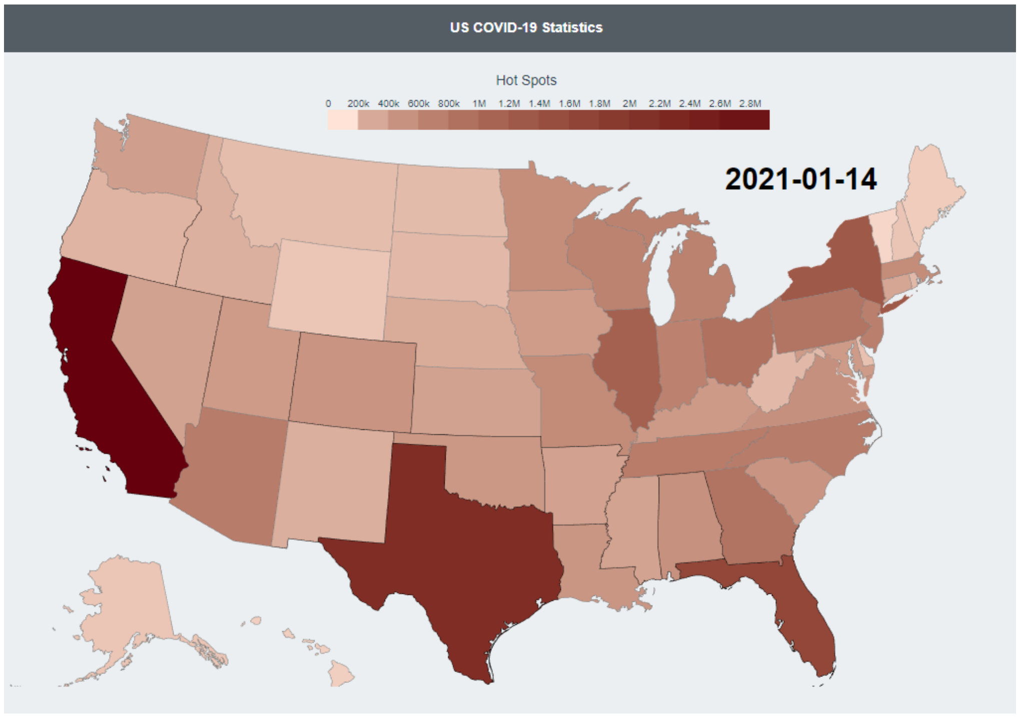
- To run the program on Windows, the user needs to navigate to the program folder (e.g., C:\User\xx\d3-covid-2019-2022)
- Use the following commands:
  - npm install
  - npm run serve
- In a web browser such as Chrome, go to the localhost address as shown in the following screenshot:



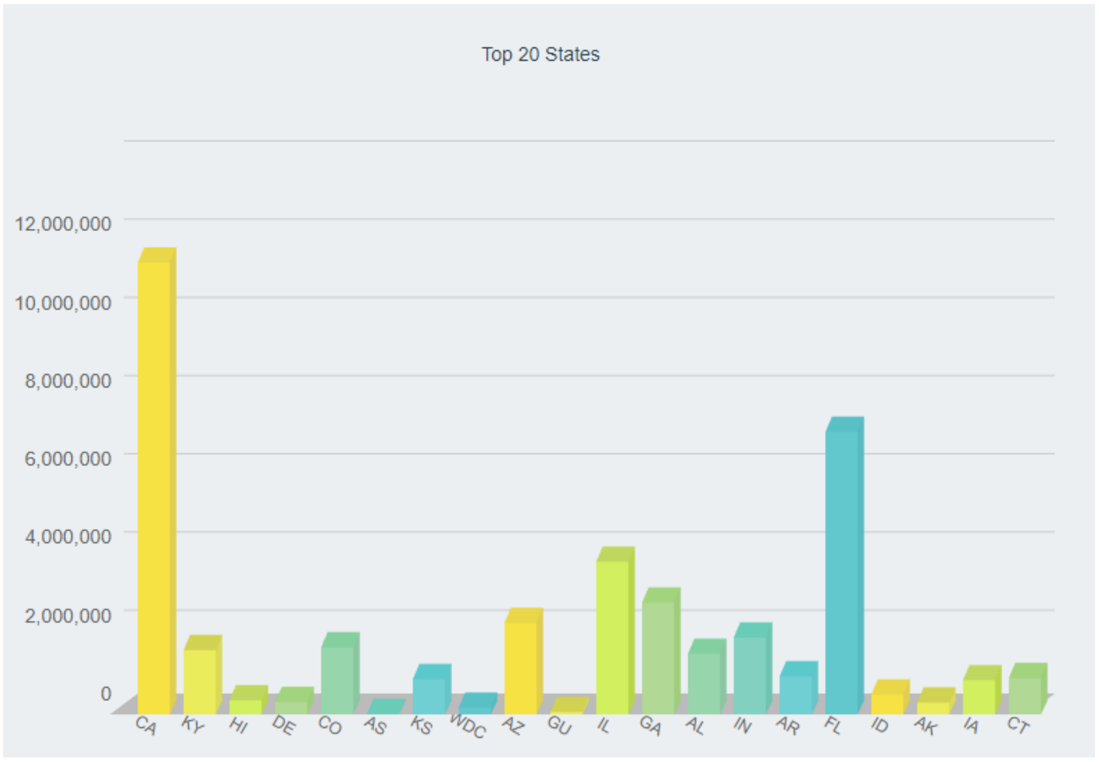
```
 DONE  Compiled successfully in 10844ms

  App running at:
  - Local:   http://localhost:8080/
  - Network: http://10.31.38.13:8080/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

4. **Screenshots of your program results with an explanation.**

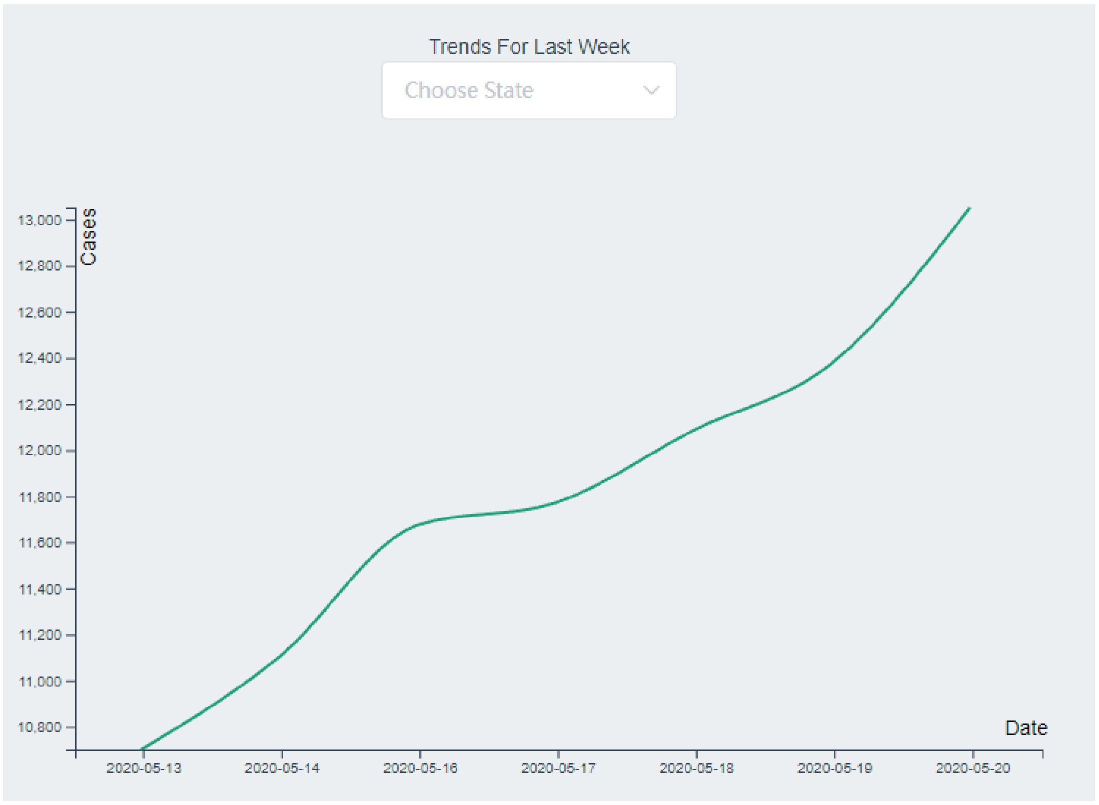
Firstly, the geographic visualization displays an animated geographic visualization of US with a legend. The map is created from the data of the first year of COVID-19. It is colored by the confirmed cases of each state and the color changes on a daily basis. A tooltip shows up when the user hover on a state, indicating the name of the state, confirmed cases, and deaths. There were two reasons for me to only use the data from 1/21/2020 to 1/19/2021. One thing is that I could create a good animation effect out of it; another thing is that I also wanted to visualize the trend of the first year after the pandemic hit.



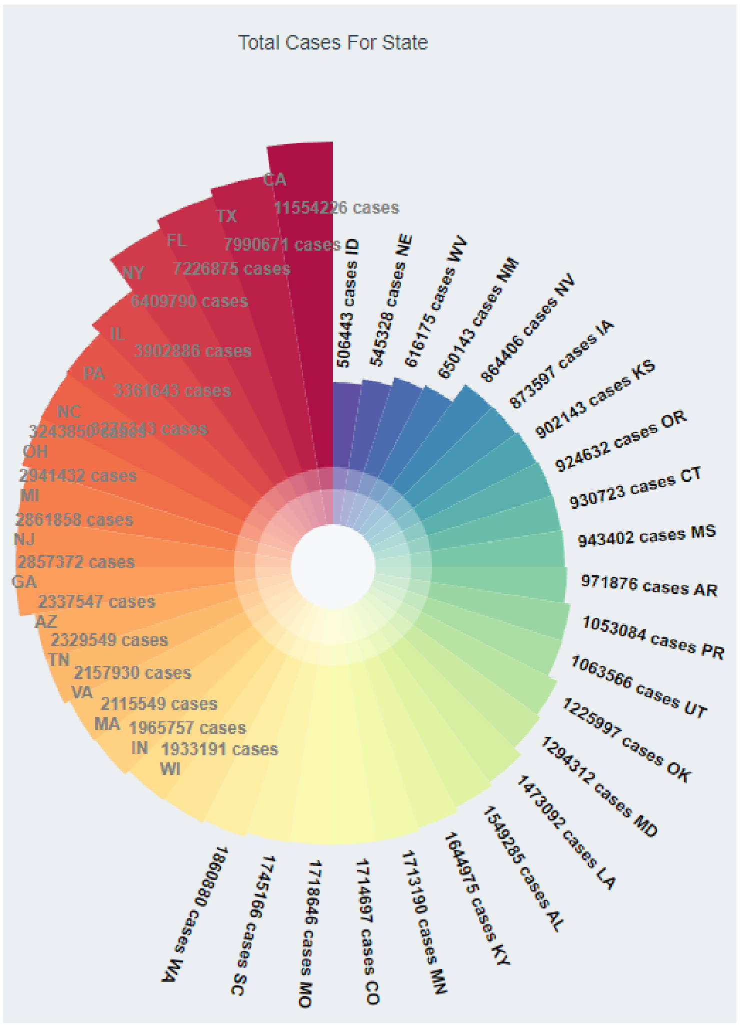
For the next section, I have a 3D bar chart for the top 20 states by number of confirmed cases. A tooltip is also added for this chart that shows the user the full name of the state being represented and the exact number of confirmed cases. The data used for this chart is from 12/06/2022.



The next graph is a curve graph indicating the trend of cumulative cases of states on the state level for a single week. The date is on x-axis and the number of cases is on y-axis. The user can check the curve of another state by selecting from the dropdown menu.



Moving on to the next one, The Nightingale's Rose Chart. This chart is known to be used by Florence Nightingale to communicate the avoidable deaths of soldiers during the Crimean war. It's basically a circular histogram, with multiple axes radiating out of the center. Bars are drawn for each axis to indicate the quantity, which in our case, is the confirmed cases of each state. The Rose chart could be viewed as a combination of the Radar Chart and Stacked Column Chart. This particular Rose chart was created from COVID data retrieved recently (12/06/2022). I have the bars colored from cold colors to warm colors to represent the ranking of confirmed cases in a total of 40 states. The exact number of cases and state abbreviations are also included in the graph. Looking at this Rose chart, the user may find strong comparison between the data of some states.



5. **Any technical challenges worth mentioning and how you solved them?**
I faced some challenges creating various visualizations. I was creating duplicated graphs and it took me a while to figure out that I just needed to remove the previously created one before creating a new one. This was a problem caused by the usage of the command npm run serve when debugging; every time I made a change to the code and used that command, the code was automatically compiled and the whole page refreshed with the

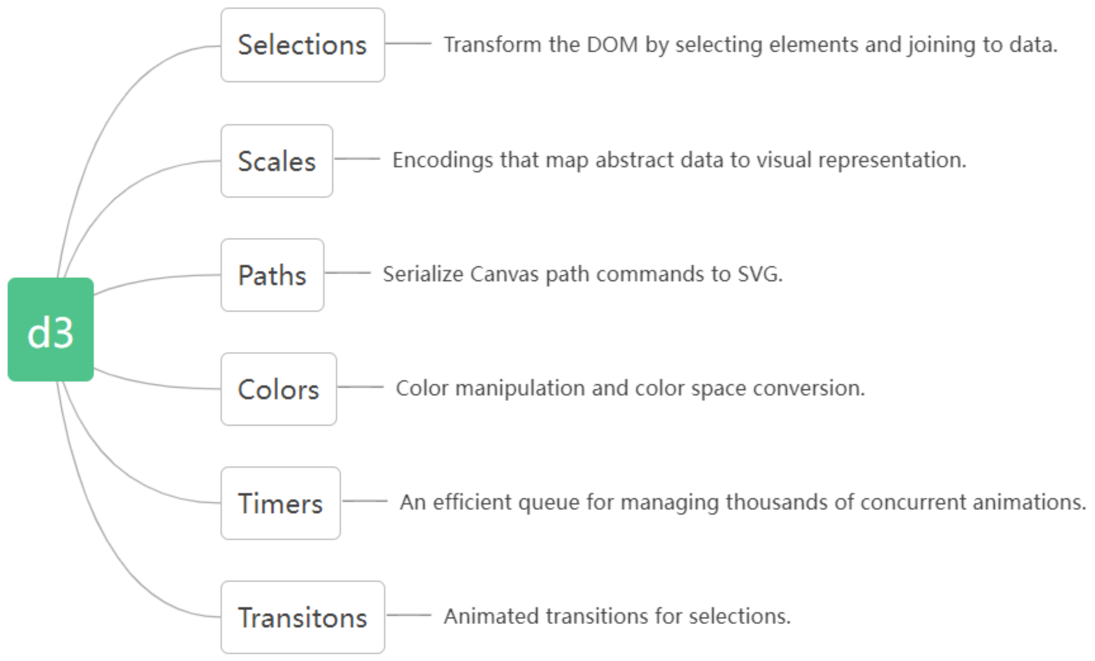
previously created graph still presenting itself. Also, there was a dropdown menu for the curve graph that is used for choosing another state; I added a click event but it was not working at all. I figured it out that I only have to change click to click.native because it simply has to be done when binding events to vue modules. Another headache was that I was creating blank graphs. I had to employ callback functions to make sure graphs are starting to be created after d3.csv and d3.json had finished loading data files.

6. Assessment of your effort in this project.

For my self-assessment of the efforts made in this project, I would say they were quite productive. I was able to build all the functionalities I wanted for the project. There is still work to be done. For example, the geographical map could use a play button that changes to a pause button after the animation starts. Some parts of the code for tooltips (tipTimerCfg, createMyTipTableData, clumnMyMouseout) can be refactored. Overall, I'm happy with the end product of the efforts I put in.

7. Anything you learned from this project?

All in all, I learned quite a lot from this project. I learned how to use many D3.js API functions and the following image includes the mostly used types:


d3
Selections
Transform the DOM by selecting elements and joining to data.
Scales
Encodings that map abstract data to visual representation.
Paths
Serialize Canvas path commands to SVG.
Colors
Color manipulation and color space conversion.
Timers
An efficient queue for managing thousands of concurrent animations.
Transitons
Animated transitions for selections.

When displaying multiple visualizations, I learned the importance of using callback functions as parameters in the function that loads the data and calling the callback function when the data is completely loaded, otherwise the graph created would be a blank one since the data was not fully loaded. I also learned the importance of removing previously created graphs before creating a new one since the page would compile the code and refresh automatically each time I used the command npm run serve when debugging. This prevents creating overlapped graphs that leads to confusing visualizations. Finally, I was glad that I was able to find the COVID-19 statistics in a New York Times licensed Github repository and to retrieve the map data from a public source. Accessing them and checking out data they had other than what I actually used were quite interesting.