DePaul University
Team Red Bull Final Project Report
CSC 468 - Programming Interactive Data Visualization for the Web
11-19-2023

Contributors

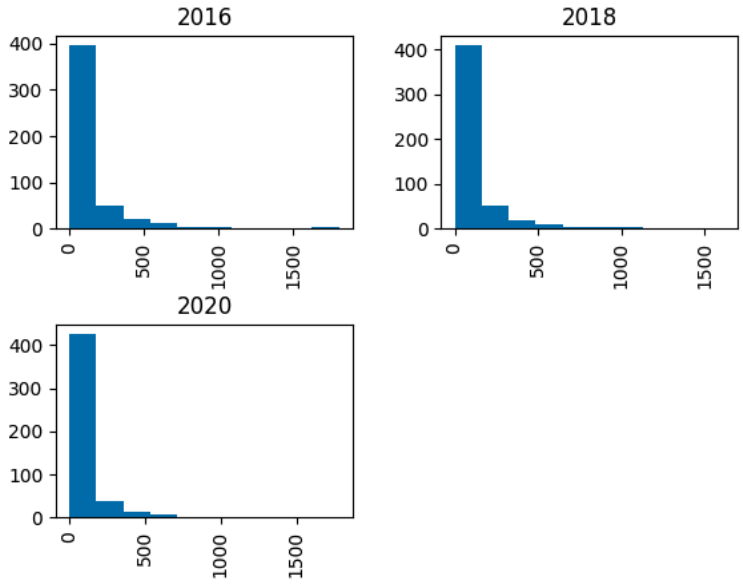
1. Jason Lehmann
2. Tim Anderson
3. Daniel Dut

A. Introduction

Our project ended up being about exploring the results of 3 Chicago Local School Council (LSC) elections, primarily focusing on each school's name, ID, boundaries, election year, candidate name, and vote count for candidates with candidate types of **Parent** and **Community**. In the initial pitch, we also considered different types (Advocate, Community, Education Expert, Non-teaching Staff, Principal, Student, and Teacher). However, we had to limit the scope in order to prioritize core functionality.

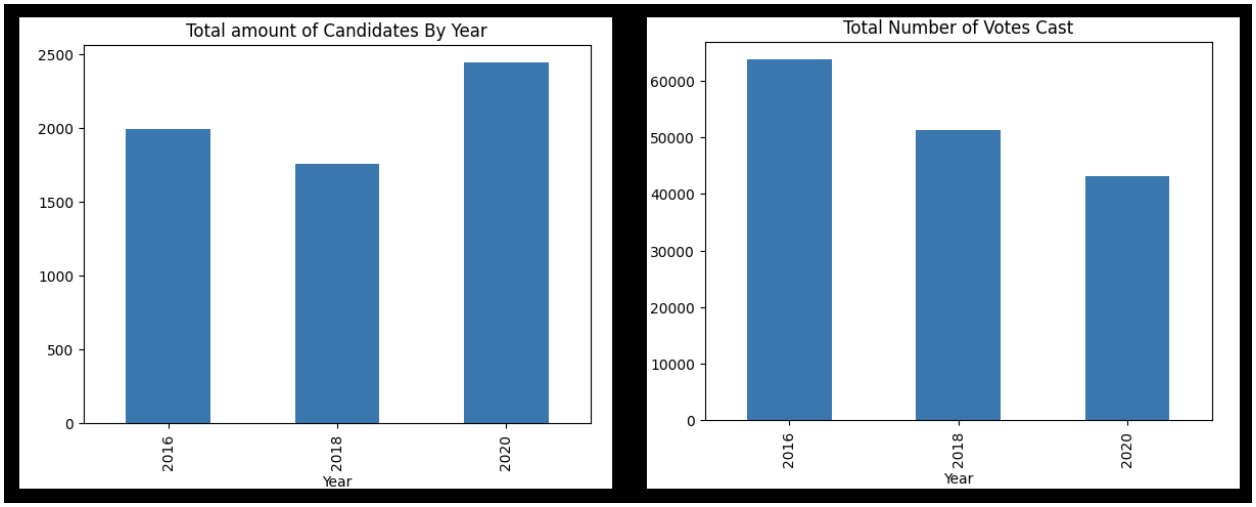
B. Exploratory Analysis

Our exploratory analysis was primarily done by breaking down the very large election dataset into a more manageable chunk of data. By examining Tim's data dictionary, we pruned it into looking at just the school board elections that had the maximum participation. We ended up using Parent elections as the category to focus on and also included hardcoded ways to look into other election categories.



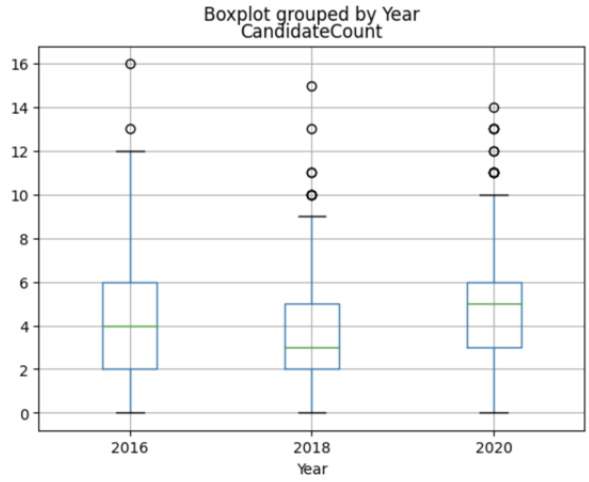
For displaying the election results, the total sum of voters that participated in each school board's parent election was consistently skewed to the right, prompting the need for the legend

of the choropleth, our first graph, to be included as a feature when the graph adapts to each year.



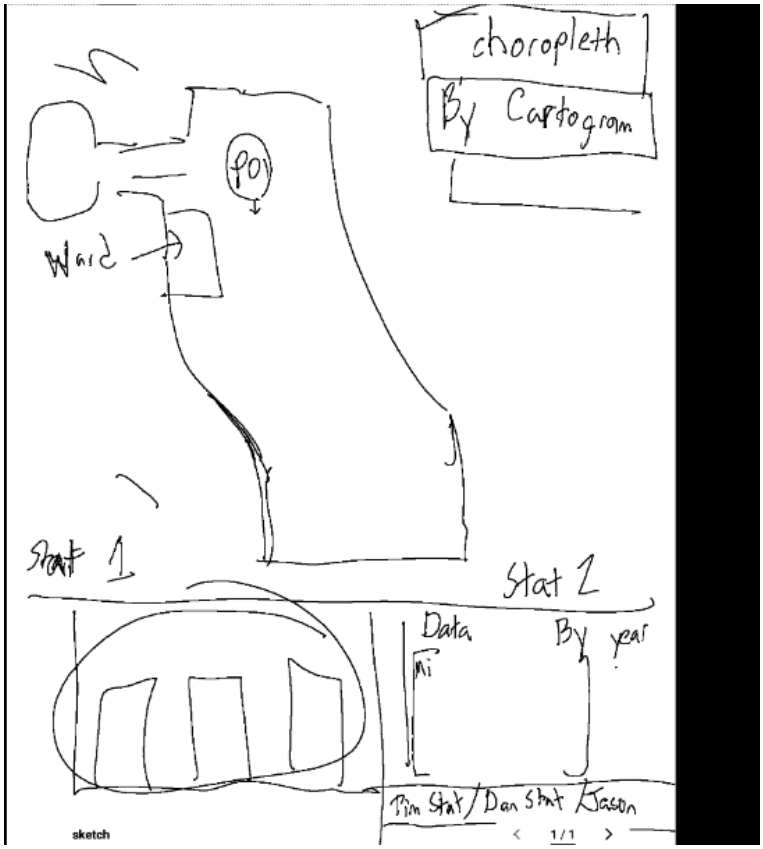
Within specific elections, the number of candidates ranged from 1-16 and so we wanted to represent election results categorically for our second graph.

Based on some of the trends found in some of the commonly selected districts during development, we discovered that repeat candidates were common, and so we decided to look at their success across elections in our third graph.



In terms of repeating candidates, we found in the parental category 116 people who have participated in at least two of the 3 accounted elections and 14 candidates who participated in all 3. We also were able to confirm that

C. Scope and Design

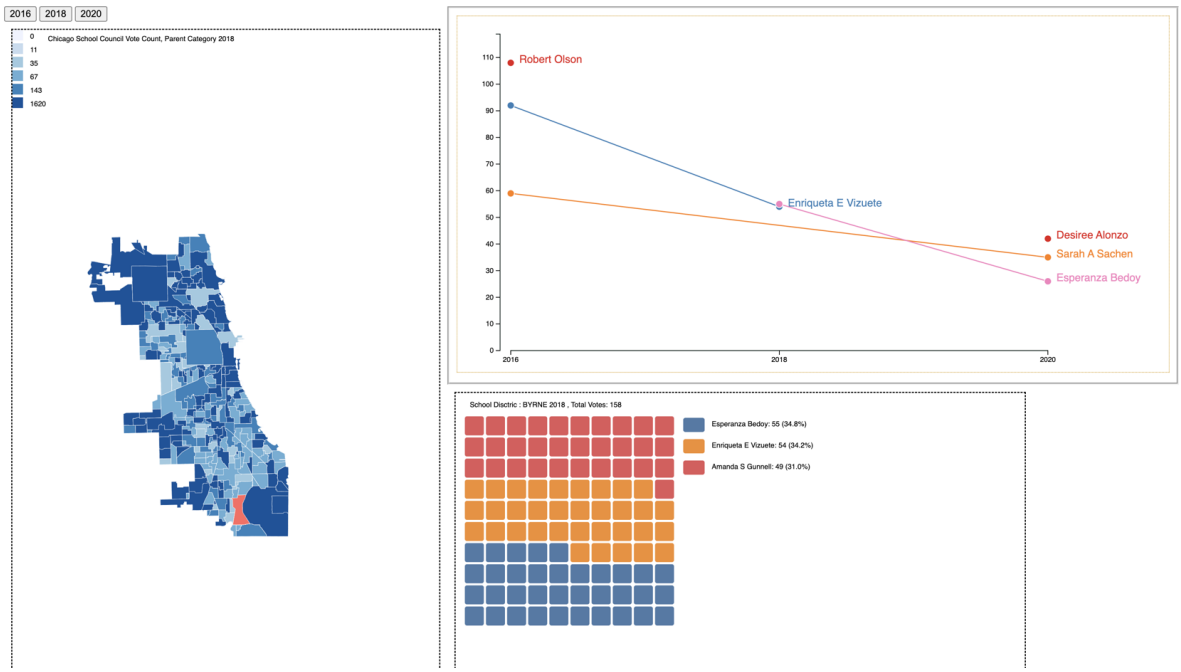


For the scope, Jason wanted to make a choropleth and a way to categorically view any subsection of the election results.

The focus of our application is to use the choropleth selection to initiate an API call and then dispatch that information to the subgraphs. As we developed the application, a level of state management became necessary in order to render additional election types, but those ultimately ended up being cut from the scope.

D. Interactive Visual Analysis Tool

Using the combined geojson taken from https://api.cps.edu/maps/help and Tim's data pool, we were able to generate a working choropleth that visualizes school boundaries, color-coded by the total number of votes, and with buttons to switch between election years. Selecting a school generates a visualization of the detailed election results by the candidate in a waffle chart. With the Line chart we were able to see any trends of a consistent candidate compared in addition to identifying the winner of each election having the highest vote total in each column.



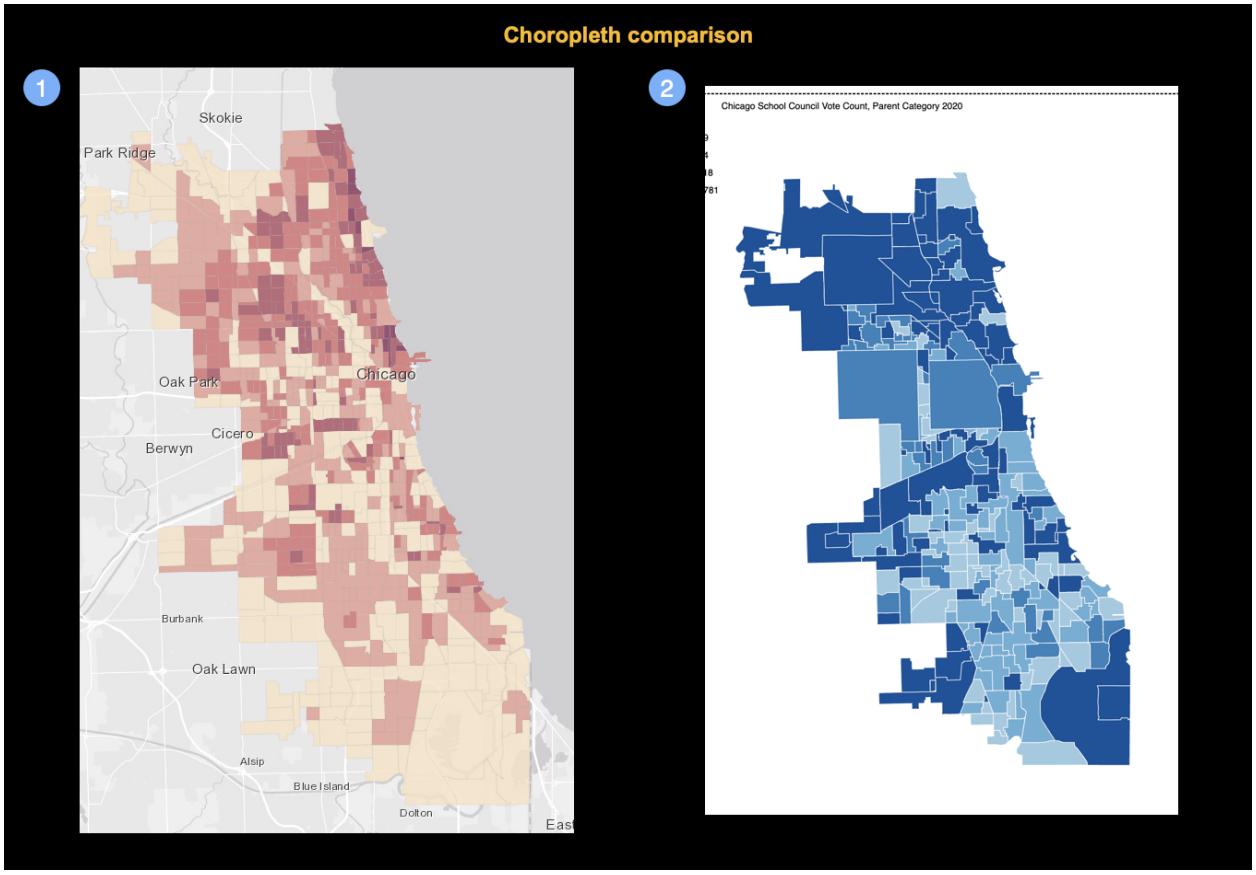
E. Analysis and Discussion

With our smaller sample size it is difficult to determine any conclusive trends, but our visualizations enable us to start to see the history of each school's election results over time.

Future development could provide, for example, tooltips showing how many votes each part of the waffle chart represents. And the choropleth could show the difference between counts between years.

Compared with a Choropleth marking the 2010 population of Chicago, our app lines up with districts containing a higher population marker being reflective of the visual data being displayed in our election count graph.

But with the data, it will be interesting to see if a 2022 sample of the data reflects the greater candidate participation compared to the



F. Code

[Github link to the code]

[Temporary server link]

Appendix:

Jason's Contribution.

For this project, I wanted to explore a dataset involving Chicago that was not a part of a crime statistic. After midterms, we Invited Tim's and started pruning his dataset into something more manageable.

During our first official meeting, I sketched the initial idea for the application. I then created the basic backend server using the approach introduced in Lab 5, got a working Choropleth and then further expanded the backend to have a working API to pull specific sections of the election data for our purposes.
I also implemented the interactive features and waffle chart and was able to modularize most of the figures to be able to accept different sets grouped by year.

Based on Dan's comment I converted my initial project from D3 v4 to v6 and confronted an issue that involved our map's rendering backwards. I also established a pseudo-state management system in order to allow different candidate types to be used and made the initial calls on the basic appearances and color choices.

After Presenting, I worked with Tim to further refine our project, adapting the colors and making the legend dynamically generate its quantiles. I felt like I had enough code clarity that Tim was able to run with both the front and back-end code to contribute at this state.

Any support for data analysis was done using a Jupyter notebook set up by Tim. With this, I started breaking down data candidate stats over the 3 time periods explored. I also refined the line graph to include points for each election winner and to have a label for each repeat candidate.

I also did much of the delegating when other team members asked for tasks to complete. And most if not all of the writeup above was done by me.

Tim's Contribution

I met with the group after the midterm. One topic of discussion was the dataset. Jason and Daniel were considering civic data and we agreed to use a dataset I had previously compiled with demographic and election data divided by a variety of municipal boundaries. Another topic of discussion was an interactive visualization that I had built using D3 for my undergraduate capstone project.

Shortly afterward I uploaded and shared all of the data from a Google Drive folder, adapted an instructions document for working with the data, and created a basic data dictionary. The data included smaller files aggregated by municipal boundaries, a variety of shapefiles for those boundaries, and a very large file divided across boundaries at the most granular level. To make the latter more manageable, I also provided a Tableau file I had previously created which connected to the file hosted on AWS S3.

Data, Data Dictionary, and Data Guide

By our next meeting, we realized that we needed to narrow our focus on the data, and decided to use the local school council (LSC) election results aggregated by LSC boundaries. I had only aggregated by the most recent year's boundaries in the data I provided, so I agreed to reprocess the data to produce aggregations for the prior two years as well, which I did shortly afterward. I also went ahead and got the aforementioned capstone project working again, and hosted it on a local Tomcat server to share with the team.

By the following meeting, it became clear that some more data preparation was required to collaborate on the data, so I agreed to combine the three aggregated years into one file and load them into a Jupyter notebook joining together the LSC election results, other election results, and demographic data, which I did shortly afterward.

Data Preparation Jupyter Notebook

By the next time we regrouped, we determined that even more data preparation was required to incorporate the data into the prototype that Jason had begun creating, which I agreed to do in the Jupyter notebook. I narrowed down the LSC election result columns to those belonging to a given district–removing columns for overlapping and non-overlapping results–sorted those columns by number of votes, documented the candidate types based on the rules on the Chicago Public Schools website, and narrowed the scope to the top two most important candidate types. I left out demographic and other election columns for simplicity until we could determine whether we wanted them for the project.

We next regrouped after Jason's presentation, by which point he had incorporated the data into the visualizations he built, pulled the shapefiles for one year from an API separately from the rest of the data, and created a backend partly based on the Jupyter notebook. I agreed to combine the three years of shapefiles I had, convert them to the GeoJSON format, and join them to the main data file, which I went ahead and did using the Jupyter notebook.

Finally, I set up my environment to collaborate on the visualizations and backend. I familiarized myself with the code; implemented my combined dataset and switching between years for the shapefiles; made some tweaks to the choropleth style and browser compatibility; filtered out schools without any relevant votes from the choropleth; and narrowed the scope to parent candidates for the simplicity of further development.

Daniel's Contribution

In the initial sketching of the ideas to visualize, I got together with Jason Lehmann to look into what we could get onto a map of some sort. We started with a pretty small dataset of the City of Chicago Employees but that was later deemed not large enough to make good sense on a Choropleth. When Tim got on board, he suggested a bigger dataset that he worked with prior. We took that on and requested him to clean and trim down the data to fit our scope.

During our first meeting as a team of 3, we divided up tasks to get a working MVP out. I took up the tasks I got assigned. I wrote up the milestone 3 document and submitted it on D2L. While preparing the milestone document, I made the "Action Items" tab in the Google sheet we were using into some easy-to-use "feature/bug tracking" document.

On the code, I wrote the following (features and/or enhancements):

1. Default route for the app's home page to 'index.html'.
2. Refactored the code to semantic HTML, CSS, Python and to conform to Flask's code organization standards.
3. Added the Line Chart (on selection) feature to the Choropleth:

a. Wrote data transformation functions on the backend (Python) for easy ingestion by D3. i.e. functions for reconstructing JSON to a more palatable format and converting from JSON to CSV (in **server.py**).

From this (single year):

```
[
 {
   ...
   "Year": 2020,
   "Community Candidate 1 Name": "Peter J Chico",
   "Community Candidate 1 Votes": 32.0,
   "Community Candidate 2 Name": "Lucia Gonzalez",
   "Community Candidate 2 Votes": 20.0,
   ...
 }
]
```

To this (multiple/all years):

```
{
 "2016": "[{\"ID\":609739,\"Name\":\"WASHINGTON
HS\",...]",
 "2018": "[{\"ID\":609739,\"Name\":\"WASHINGTON
HS\",...]",
 "2020": "[{\"ID\":609739,\"Name\":\"WASHINGTON HS\",...]"
}
```

To this (all years (better JSON)):

```
[
 {
   "boundary_type": "Attendance Area",
   "candidates": [
     {
       "name": "Veronica Arias",
       "type": "Community",
       "votes": 172
     },
     {
       "name": "Patricia M Montijo",
       "type": "Parent",
       "votes": 42
     },
     // more data
   ],
   "community_total": 633,
```

```
  "id": 609739,
  "name": "WASHINGTON HS",
  "parent_total": 799,
  "year": 2016
},
{
  "boundary_type": "Attendance Area",
  "candidates": [ // more data ],,
  // more data
  "year": 2018
},
{
  "boundary_type": "Attendance Area",
  "candidates": [ // more data ],
  // more data
  "year": 2020
}
]
```

And then finally to this (CSV for the line chart):

```
year,name,votes
2016,Veronica Arias,172
2016,Yolanda Deanda,139
...
2018,Veronica Arias,69
2018,Yolanda Deanda,52
...
2020,Glenda Fuentes,16
2020,Toni L Gonzalez,14
```

b. Wrote D3 portion (as a module) for line chart to use the functions above in the choropleth.
*Excerpt from **line-chart.js** file*:

```
const LineChart = () => {

   return {

       draw: selectedDistrict => {

           // D3's drawing code here

       }

   };

}
```

4. Added significant info to the main README file for the project (markdown formatting, adding links and steps for running the app).
5. Wrote a bash script for running the app (run.sh, instructions for using it in the README).

6. Tidied up the CSS where there were clashes that made graphics not show as expected (leveraging CSS specificity in most cases).