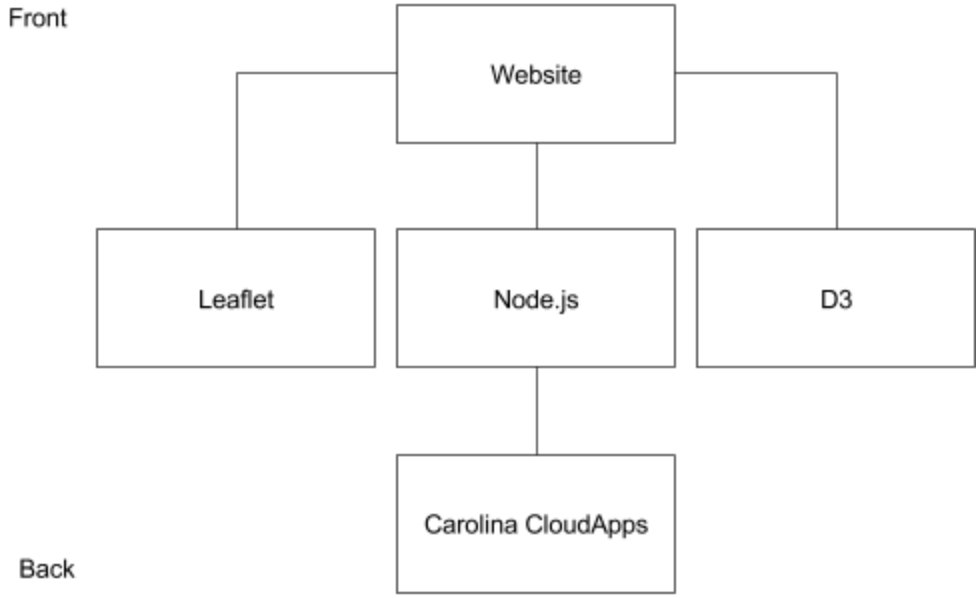
Gerrymander Project

Architecture



Decomposition

Modules

We've decided to use Carolina Cloudapps to host our website, with a Node.js layer to provide communication between the browser (client) and the server. The Node.js layer is responsible for sending the relevant page for the user to see, so that the user can access the website using the URL hosted by Carolina CloudApps (http://gerrymandering-dept-gerrymandering.cloudapps.unc.edu).

We then have two external libraries which are used to better visualise the data related to the districts

- Leaflet - Choropleth maps for each metric
- C3 - Charts such as bar charts for each district and metric

Although not included in this diagram, R was used in order to process shapefiles for different North Carolina district maps and translate them into GeoJSON format, so that the data can be interpreted into a more human-friendly format, like choropleth maps, tables, or charts.

GeoJSON files used for this app contain different types of data depending on what was being visualised:

- Boundary co-ordinates
- Metric data

Each district has a set of co-ordinate pairs, originally stored by the state's shapefile map, which define the boundaries of the district on a world map. These co-ordinates are then used to draw out each district in an

embedded Google Map on the website. **As we are using Leaflet with Mapbox to display each district, as opposed to Google Maps, each generated co-ordinate pair MUST be represented in a format such that the longitude (West/East) is the first element in the pair and the latitude (North/South) is the second element.** We also use decimals numbers rather than minutes and seconds to further refine the positions that make up each boundary. For example, the Sitterson building has a latitude of 35.909 and a longitude of -79.053. If we were to give that information to Leaflet, we would have to pass it with the longitude first as (-79.053, 35.909).

The second set of data the GeoJSON file should have about each district is compactness metrics related to gerrymandering. We decided to use the following:

- Reock
- Polsby Popper
- Schwartzberg
- Convex Hull
- Length Width
- X Symmetry

Each district in a GeoJSON file should have a value for each metric that will be used in the application, our case being the six already listed above. The data is then used to colour the choropleth maps, as well as to generate tables and graphs based on them

Design Decisions

We decided to use Node.js because of it being lightweight and scalable. We've decided to use R to analyze the shapefile maps because R provides convenient access to the Geospatial Data Abstraction Library (gdal) for translating a number of common raster and vector geospatial data formats used in redistricting applications. Because we want dynamic data visualizations, we're using C3.js, and JavaScript for the front-end to populate the website.