## Project 1

**Problem Summary**

Project 1 asked for consideration of any existing correlation between sampled nitrate concentrations and cancer incidence rates broken out by geographic region across the state of Wisconsin. To analyze and evaluate the data, we were asked to prepare a GIS application capable of executing interpolation of the nitrate concentration point data and then conduct regression analysis to determine the correlation between cancer rates and nitrate concentrations. The application needed to be executed outside of commercial off the shelf GIS Software, such as ArcGIS or QGIS. The program needed to take the data provided and output a map readable by the end user to conduct their own analysis.

**Implementation Plan**

I decided to implement my solution using web framework. Specifically, I used turf.js, an open source javascript library for conducting spatial analysis, leaflet.js, an open source map visualization library, jQuery to conduct asynchronous actions and read files, and SimpleStatistics, an open source advanced math library. To render the page, I combined HTML, CSS, and Bootstrap. To render the data readable by turf.js, I used QGIS to convert the shapefiles to the appropriate GeoJSON format. After deciding which libraries I needed, I drafted the execution plan.

**Deliver Original Data**

The original data consisted of point data for the nitrate concentration and polygon data for the census tracts. I first converted the data from shapefiles to GeoJSON using QGIS. After I had the appropriate data, I used jQuery and AJAX event handlers to read the data including the properties and geometry information. I passed the data into an array for javascript to read. After storing the data in an array, I established breaks using the ckmeans method of classification. The ckmeans method of classification is a clustering algorithm based off of the natural jenks breaks method. The algorithm uses an established number of clusters and sorts the data to fit most like data in the same cluster. I iterated through the data using javascript methods built into leaflet.js to display the data on the map. While iterating through the data, I also colored the data based on the attribute values. The user could choose to display one or both of the original layers, and each layer had it’s associated legend. The legends appear and disappear as the layers are turned on or off on the map.

**Interpolation Setup**

To set up the interpolation, I used turf.js built in interpolate method. The interpolate method functions based off of the equation for inverse distance weighting (IDW). IDW operates on the principle that points nearest a sample are more similar to points farther away. In order to successfully run linear regression later, I would have to have the same number of dependent and independent variables, so I had to interpolate data for both the nitrate concentrations and cancer rates. The nitrate concentration interpolation is straightforward. Since the data was already point data, I used the interpolate method and customized the grid options to allow the user to change the k-value for IDW and the area of the hexbins. I chose square miles for the hexbins because of the American understanding of the imperial system. I left 7 as the default value for the users for interpolation because the average size of a census tract in Wisconsin was 42.23 square miles, so a hexagon at 7 square miles was closes to that measurement. I chose hexbins based on the advice of the instructor and previous student examples. Interpolating the data for cancer rates was more difficult. To interpolate that data, I had to first convert the polygons to centroid points, then I had could interpolate those points. In the first run, I had converted the census tracts to a MultiPolygon GeoJSON and not a Polygon GeoJSON, which wouldn’t allow that to happen. After some trial and error, I ended up fixing the problem. I then interpolated the centroids using the same measurements as the nitrate concentration points.

**Spatial Join**

In order to prep for regression analysis, I had to conduct a spatial join. Since we interpolated the data at the same grid options, the grids for the data were the same. Turf.js’ collect method performed most similarly to spatial join, and allowed me to add the attributes for cancer rates and nitrate concentrations into one large array for conducting linear regression analysis.

**Ordinary Least Squares Regression and Analysis Overlays**

I used SimpleStatistics Ordinary Least Squares (OLS) regression to conduct the linear regression analysis for the project. OLS regression was best suited for the project because it tries to provide a line of best fit based off of the formula *y = mx + b.*  SimpleStatistics took the nitrate concentration as the independent variable and then predicted the cancer rate as the dependent variable. Once the calculation was made through the entire array, I added that layer onto the map and colored the classification breaks based on distance from the standard deviation. In addition to the regression overlay, I added visibility for the Nitrate Concentrations IDW layer and the Cancer Rates IDW layer. The latter two layers were colored like their original counterparts, and the classification was using ckmeans clustering.

**User Interface**

I gave the users two options to manipulate data. The user could manipulate the k-value by entering in a text box a decimal value between 1 and 100 and a hexbin size in square miles between 4 and 60. The user could submit their values for analysis or reset the map to the original configuration. I restricted zoom and pan to give the user access only to the state of Wisconsin in order to prevent a user from getting lost on the map, or accidentally disorienting themselves in an errant click. I decided to choose a dark theme for the website. As a migraine sufferer, I understand how bright screens can irritate eyes and aggravate headaches.

Beyond the two input boxes, I offered the user the option to either submit their values to run analysis or reset the values to the original values and display the original data. The javascript has an event listener to listen for the submit button to be pressed. When the submit button is pressed, the values are grabbed, parsed to float values, and then sent to the gridOptions for the interpolate method to function properly. Additionally, the Regression equation and *r2* values are displayed. When the user clicks the Reset button, all layers on the map are cleared and the first two layers are displayed again, with their legends. The equation and *r2* values are hidden.

**Analysis**

After running through k-values between 1.5 and 2.5, there are few important results to note. First, the *r2* values are all below 0.12 at every k-value between 1.5 and 2.5. This low *r2* value would typically indicate a bad fitting line. However, when the residuals are mapped, most of the hexbins fall within one standard deviation of the mean. Each residual hexbin layer between 1.5 and 2.5 has between 73% and 77% of the hexbins falling within one standard deviation of the mean. There are geographic areas, like the Minneapolis metropolitan region and a swath of central Wisconsin that are consistently over and under predicted, respectively. Overall, the analysis predicts fairly accurate cancer rates in each hexbin. This does not mean that higher nitrate concentrations will automatically lead to higher cancer incidences. A problem inherent in this design is that we don’t have any other variables to explore against the cancer rates. A second problem is that this analysis does not do well in a couple of key areas of Wisconsin. For those reasons, I would argue that despite the relatively high value of accuracy in predicting cancer rates, that we can’t use this to correlate nitrate concentrations and cancer rates. The variance in certain areas is too great to ignore. The areas where variance is highest are statistically significant places, as they include a population center of the state.

Resources Used in Application

1. Bootstrap https://getbootstrap.com/

2. jQuery https://jquery.org/

3. leaflet.js https://leafletjs.com/

4. SimpleStatistics https://simplestatistics.org/

5. turf.js https://turfjs.org/