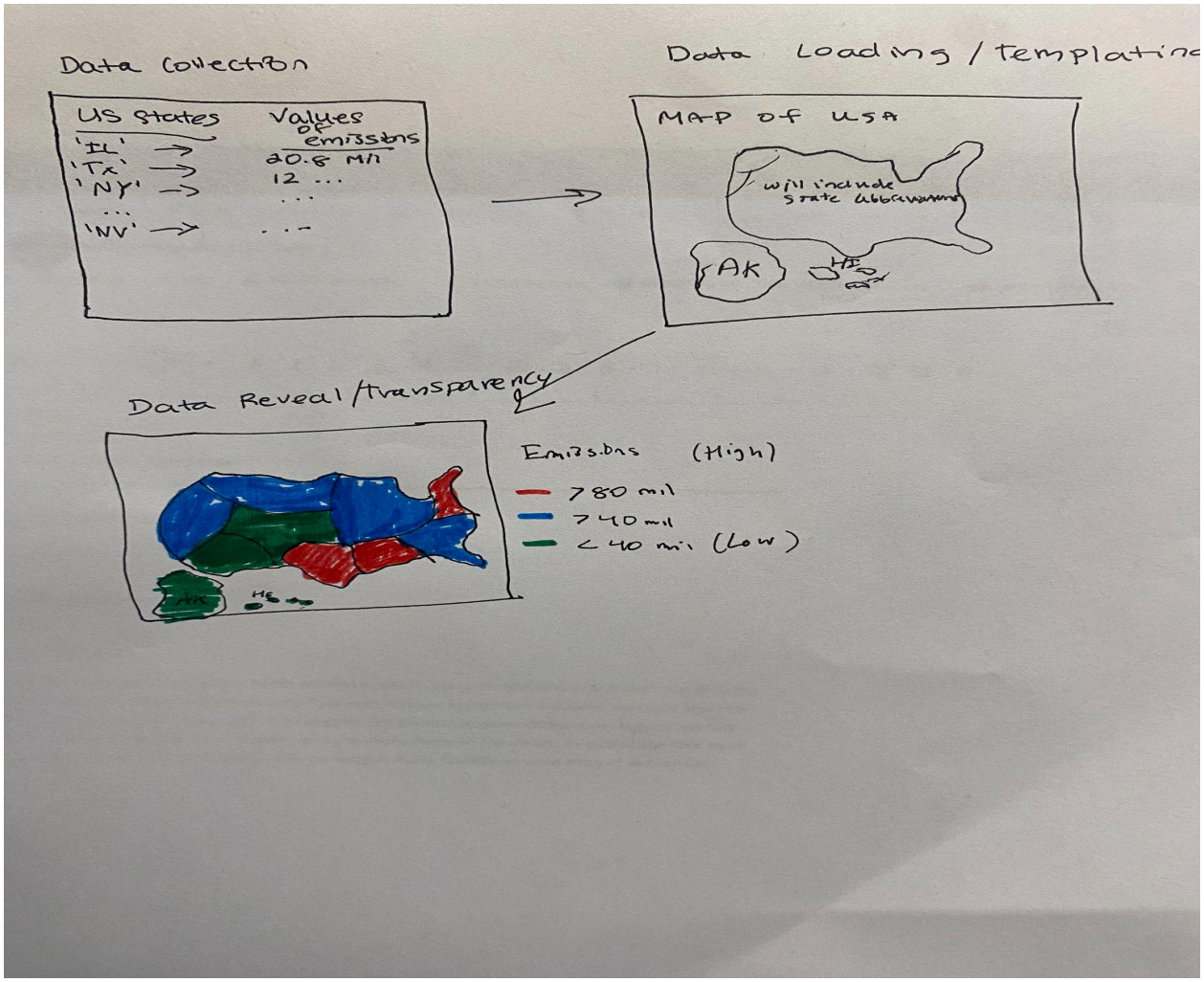
Interactive Visualization Homework 4

D3 Project

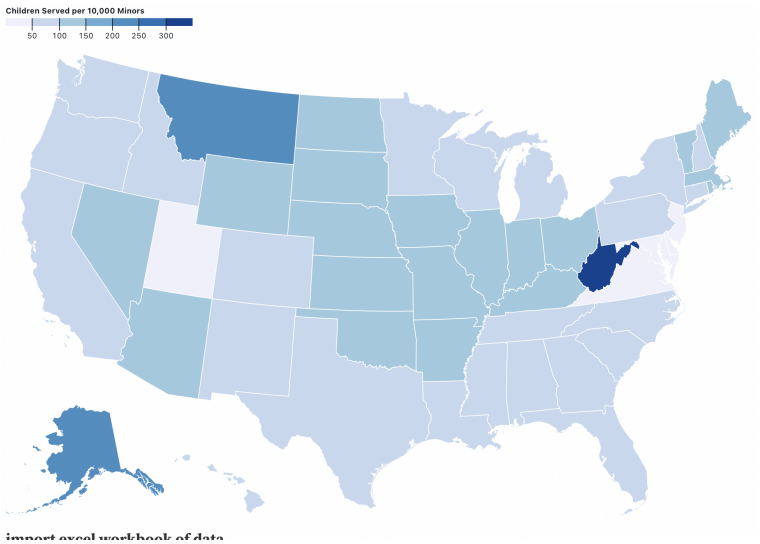
Devanshu Haldar

Story Board Initial Sketch



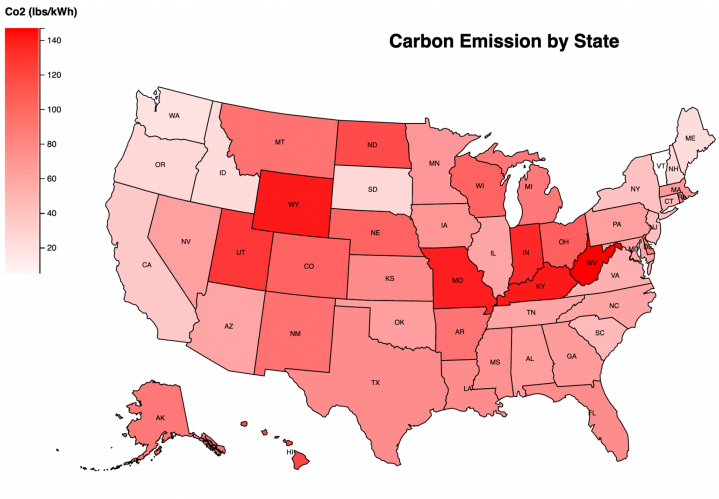
The goal above was to find data online (at the time I was not sure of what) to populate in the US map. Also, I didn't like the idea of fully different colors to show regions. I ended up going by state populus and used transparency of the color red to highlight which states have a higher value of emissions.

Results



Rox. (2023, December 16). Adapt U.S. state chloropleth - Foster Data. Observable. https://observablehq.com/d/cb8f8d90f423217e

By exploring the D3 library, I found the chloropleth by Rox to be very visually appealing. I wanted to create something very similar to this in delivering different data. I was able to add state names and abbreviations. Furthermore, I added my own scale to provide insight on color and transparency. Here is my result below:



I explored data on carbon emissions due to the use of proof of work blockchain technologies such as Bitcoin. I wanted to find what states in the US have the highest CO2

emissions due to blockchain technologies and create a simple graphic for users to see and compare regions across the United States. Evidently, the darker the color red the higher CO2 emissions. Also note that this data was collected in a previous project I worked on several semesters ago. I was not sure, initially, what data I wanted to use and incorporate into this type of visualizations, but I remembered I had this data stored within my drive.

Review of D3

D3 is an extremely powerful tool in representing your data. I found, especially in comparison to GraphViz, that the dynamic visualizations are significantly more appealing and interesting than static ones. However, D3 is quite difficult to pick up in my opinion. It requires an understanding of HTML, Javascript and CSS which is not beginner friendly. I was able to find a lot of material online in support of questions that arose over D3 and the 3 programming languages stated above. A strong suit of D3 is its variety in visualizaiton templates that it provides. From networks and country bubble charts to simple bar charts, D3 has many tools available for representing your data. Practically speaking however, I believe D3 could be more user/friendly. The code intensitivity provides many options and possibilities, but can be dawning on new users and difficult to execute your desired visualization. Lastly, especially with a lot of software now adays becoming code-free, D3 should adapt to have a little bit of both.

Note:

I have provided the code below as well as submitted it along with this document. The code requires the 2 data files (the json and csv) to be in the same folder when running the html.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<script src="https://d3js.org/d3.v4.min.js"></script>
<style type="text/css">
 /* Legend Font Style */
 body {
   font: 11px sans-serif;
   background-color: #ffffff;
 }

 /* Legend Position Style */
 .legend {
   position:absolute;
   left:20px;
   top:30px;
```

```
 }


 .axis text {
   font: 10px sans-serif;
 }


 .axis line, .axis path {
   fill: none;
   stroke: #000;
   shape-rendering: crispEdges;
 }


 text {
   background-color: #000;
   color: white;
 }
</style>
</head>
<body>
 <header><h3>Co2 (lbs/kWh) </h3></header>
 <header><h1><center>Carbon Emission by State</center></h1></header>
 <script type="text/javascript">
   //Width and height of map
   var width = 960;
```

```
var height = 500;



var lowColor = '#ffffff'

var highColor = '#ff0000'



// D3 Projection

var projection = d3.geoAlbersUsa()

  .translate([width / 2, height / 2]) // translate to the center of the screen

  .scale([1000]); // scale things down to see the entire US



// Define path generator

var path = d3.geoPath() // path generator that will convert GeoJSON to SVG paths

  .projection(projection); // tell the path generator to use albersUsa projection



// Create SVG element and append map to the SVG

var svg = d3.select("body")

  .append("svg")

  .attr("width", width)

  .attr("height", height);



// Load in my states data

d3.csv("state_co2.csv", function(data) {

  var dataArray = [];

  for (var d = 0; d < data.length; d++) {
```

```
      dataArray.push(parseFloat(data[d].value))
    }
    var minVal = d3.min(dataArray)
    var maxVal = d3.max(dataArray)
    var ramp = d3.scaleLinear().domain([minVal, maxVal]).range([lowColor, highColor])

    // Add state abbreviations


// Create a mapping between state names and abbreviations
 var stateAbbreviationMap = {
 "Alabama": "AL",
 "Alaska": "AK",
 "Arizona": "AZ",
 "Arkansas": "AR",
 "California": "CA",
 "Colorado": "CO",
 "Connecticut": "CT",
 "Delaware": "DE",
 "Florida": "FL",
 "Georgia": "GA",
 "Hawaii": "HI",
 "Idaho": "ID",
 "Illinois": "IL",
```

```
"Indiana": "IN",

"Iowa": "IA",

"Kansas": "KS",

"Kentucky": "KY",

"Louisiana": "LA",

"Maine": "ME",

"Maryland": "MD",

"Massachusetts": "MA",

"Michigan": "MI",

"Minnesota": "MN",

"Mississippi": "MS",

"Missouri": "MO",

"Montana": "MT",

"Nebraska": "NE",

"Nevada": "NV",

"New Hampshire": "NH",

"New Jersey": "NJ",

"New Mexico": "NM",

"New York": "NY",

"North Carolina": "NC",

"North Dakota": "ND",

"Ohio": "OH",

"Oklahoma": "OK",

"Oregon": "OR",
```

```
"Pennsylvania": "PA",
"Rhode Island": "RI",
"South Carolina": "SC",
"South Dakota": "SD",
"Tennessee": "TN",
"Texas": "TX",
"Utah": "UT",
"Vermont": "VT",
"Virginia": "VA",
"Washington": "WA",
"West Virginia": "WV",
"Wisconsin": "WI",
"Wyoming": "WY"
};



    // Load GeoJSON data and merge it with states data
    d3.json("us-states.json", function(json) {

      // Loop through each state data value in the .csv file
      for (var i = 0; i < data.length; i++) {

        // Grab State Name and Abbreviation
        var dataState = data[i].state;
```

```
      var dataAbbreviation = stateAbbreviationMap[dataState];


      // Grab data value
      var dataValue = data[i].value;


      // Find the corresponding state inside the GeoJSON
      for (var j = 0; j < json.features.length; j++) {
        var jsonState = json.features[j].properties.name;


        if (dataState == jsonState) {


          // Copy the data value into the JSON
          json.features[j].properties.value = dataValue;
          json.features[j].properties.abbreviation = dataAbbreviation;


          // Stop looking through the JSON
          break;
        }
      }
    }


    // Bind the data to the SVG and create one path per GeoJSON feature
    svg.selectAll("path")
      .data(json.features)
```

```
        .enter()
        .append("path")
        .attr("d", path)
        .style("stroke", "#000")
        .style("stroke-width", "1")
        .style("fill", function(d) { return ramp(d.properties.value) });


      // Add state abbreviations as text
      svg.selectAll("text")
        .data(json.features)
        .enter()
        .append("text")
        .attr("transform", function(d) { return "translate(" + path.centroid(d) +
")"; })
        .attr("dy", ".35em")
        .style("text-anchor", "middle")
        .style("font-size", "8px")
        .style("fill", "black")
        .text(function(d) { return d.properties.abbreviation; });


      // add a legend
      var w = 140, h = 300;


      var key = d3.select("body")
        .append("svg")
```

```
        .attr("width", w)

        .attr("height", h)

        .attr("class", "legend");


      var legend = key.append("defs")

        .append("svg:linearGradient")

        .attr("id", "gradient")

        .attr("x1", "100%")

        .attr("y1", "0%")

        .attr("x2", "100%")

        .attr("y2", "100%")

        .attr("spreadMethod", "pad");


      legend.append("stop")

        .attr("offset", "0%")

        .attr("stop-color", highColor)

        .attr("stop-opacity", 1);


      legend.append("stop")

        .attr("offset", "100%")

        .attr("stop-color", lowColor)

        .attr("stop-opacity", 1);


      key.append("rect")
```

```
        .attr("width", w - 100)
        .attr("height", h)
        .style("fill", "url(#gradient)")
        .attr("transform", "translate(0,10)");

      var y = d3.scaleLinear()
        .range([h, 0])
        .domain([minVal, maxVal]);

      var yAxis = d3.axisRight(y);

      key.append("g")
        .attr("class", "y axis")
        .attr("transform", "translate(41,10)")
        .call(yAxis);
    });
  });
 </script>
</body>
</html>
```