Assignment 4

VIKRAMADITYA REDDY VARKALA

Z1973679

Data Visualization(CSCI 627)

2023-11-10

1. Illinois Map (25 pts)

```
mapData=d3.json("https://gist.githubusercontent.com/dakoop/d06705a420fb348e7e03c7437bbfe4cb/raw/172303390752b7a224d876582043240ee9e9bd9b/il-counties.geojson")
```

```
cropData=d3.json("https://gist.githubusercontent.com/dakoop/d06705a420fb348e7e03c7437bbfe4cb/raw/172303390752b7a224d876582043240ee9e9bd9b/il-crops.json")
```

```
height = 600
```

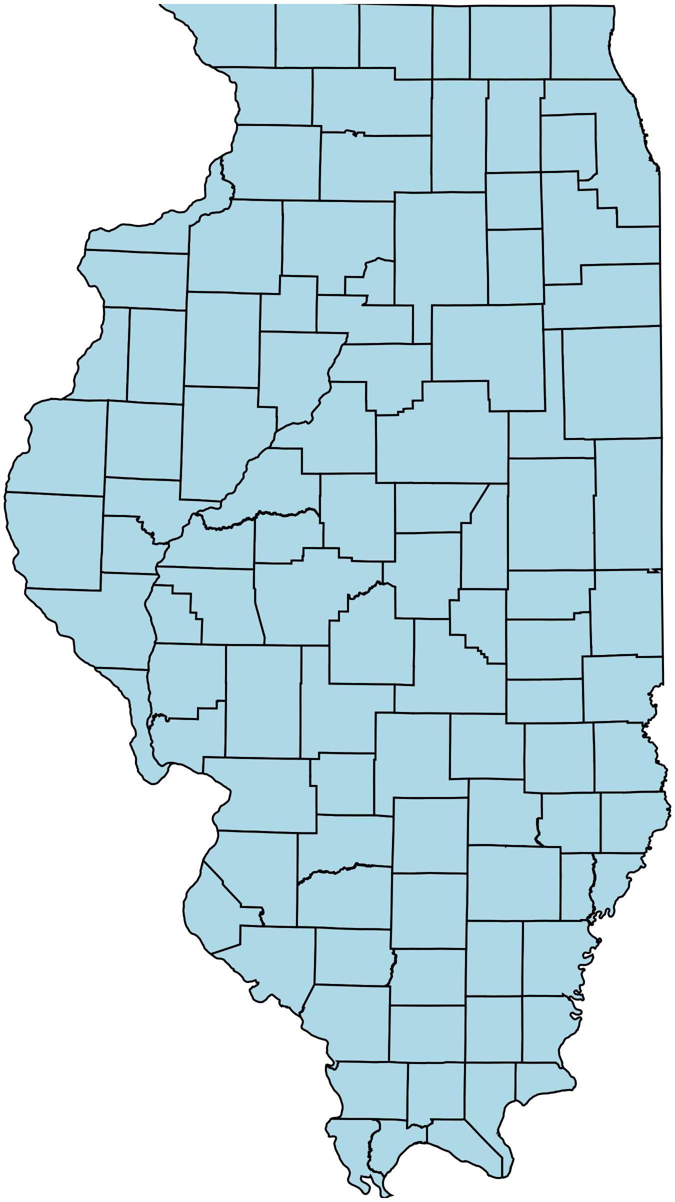
```
width = 900
```

```
colorScale = d3.scaleOrdinal(d3.schemeCategory10);
```

a. Base Map (15 pts)

```
mapProjection = d3.geoTransverseMercator()
  .rotate([88 + 20 / 60, -36 - 40 / 60])
  .fitSize([width, height], mapData)
```

```
path = d3.geoPath().projection(mapProjection)
```

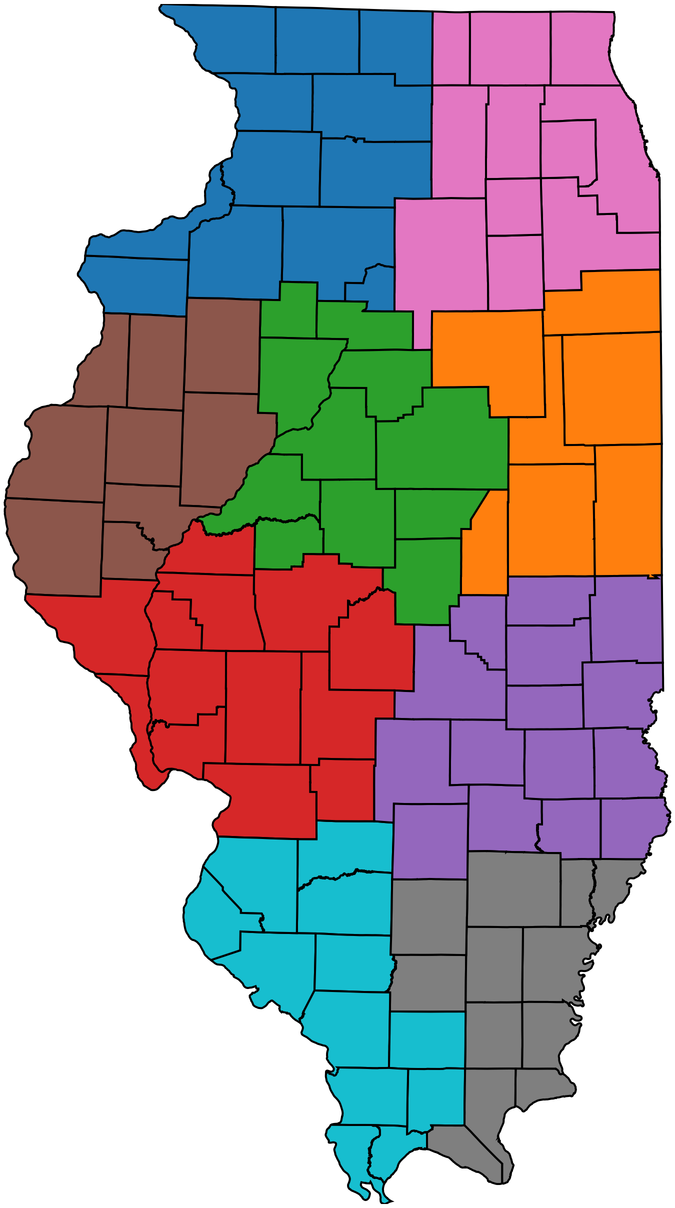


```
Basemap = {
  const svg = d3.create("svg")
    .attr("width", width)
    .attr("height", height);

  const path = d3.geoPath().projection(mapProjection);

  const counties = svg.selectAll(".county")
    .data(mapData.features)
    .join("path")
      .attr("class", "county")
      .attr("d", path)
      .attr("fill", "lightblue")
      .attr("stroke", "black");

  //tooptip
  counties.append("title")
    .text(d => d.properties.COUNTY_NAM);
  return svg.node();
}
```

b. Agricultural Regions (10 pts)

```
district = ▸ Object {1: "WEST", 3: "SOUTHWEST", 5: "WEST SOUTHWEST", 7: "NORTHEAST", 9: "WES
```

```
district = cropData.reduce((lookup, d) => {
  lookup[d["County ANSI"]] = d["Ag District"];
  return lookup;
}, {});
```



```
AgriRegionMap = {
  const svg = d3.create("svg")
```

```
      .attr("width", width)
      .attr("height", height);


  const path = d3.geoPath().projection(mapProjection);

  const counties = svg.selectAll(".county")
    .data(mapData.features)
    .join("path")
      .attr("class", "county")
      .attr("d", path)
      .attr("fill", d => {
        const agDistrict = district[d.properties.CO_FIPS];
        return agDistrict ? colorScale(agDistrict) : 'grey';// in case of  missing data
      })
      .attr("stroke", "black");

  // tool tip to sjow county name and ag district name
  counties.append("title")
    .text(d => `County: ${d.properties.COUNTY_NAM} \n Ag
District:${district[d.properties.CO_FIPS]}`);

  return svg.node();
}
```

2. Crop Production by County (45 pts)

a. 2022 Corn Harvested (15 pts)

```
// cornHarvested = new Map(cropData.map(d => [d['County ANSI'], d.cornHarvested['2022']]));
// only for corn
```

```
harvestedData = ▸ Map(103) {null => 9000, 39 => 98000, 107 => 180100, 115 => 147200, 123 =>
```

```
harvestedData = {
  const map = new Map();

  cropData.forEach(d => {
    const cropValue = d[selectedCrop] && d[selectedCrop][selectedYear];

    if (cropValue)
    {
      map.set(d["County ANSI"], cropValue);
    } else
    {
      map.set(d["County ANSI"], null);
    }
  });
  return map;
}
```

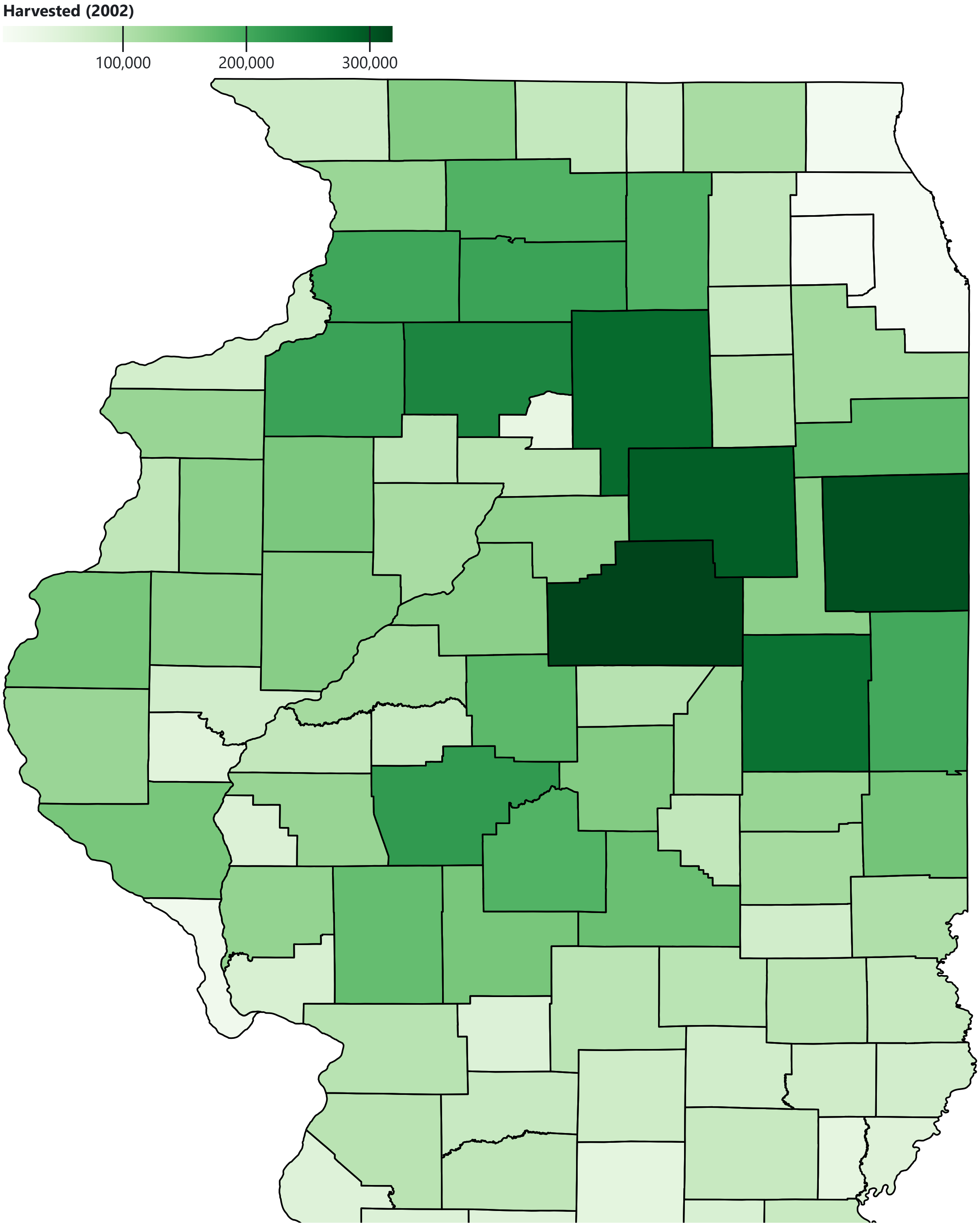
Extra credit

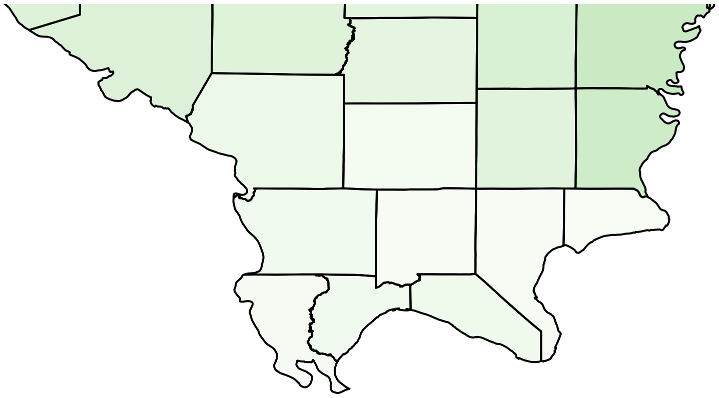
Select Year

2,002

Select Crop

cornHarvested

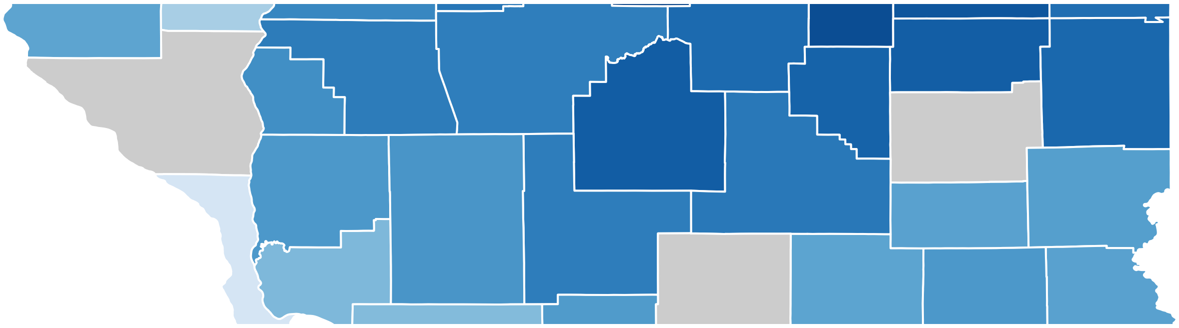




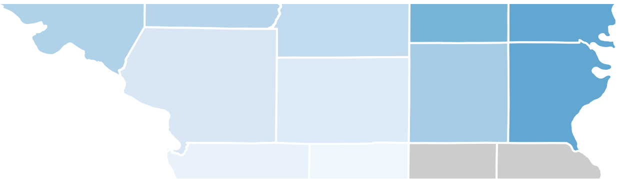
Reference : https://observablehq.com/@observablehq/build-your-first-choropleth-map-with-observable-plot

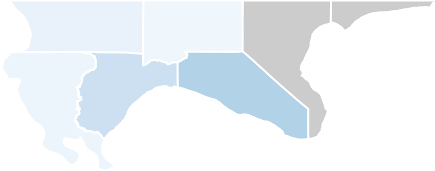
b. 2022 % Corn Acres Harvested (15 pts)

```
harvestedPercentMap = Map(24) {39 => 39.54363624356843, 107 => 44.41848841742763, 115
Plot.plot({
  width: 600,
  height: 900,
  color: {
    scheme: "Greens",
    type: "linear",
    legend: true,
    label: `Harvested (${selectedYear})`,
    unknown: "#ccc"
  },

  marks: [
    Plot.geo(mapData, {
      fill: d => {
        const value = harvestedData.get(d.properties.CO_FIPS);
        return value
      },
      title: d => {
        const value = harvestedData.get(d.properties.CO_FIPS);
        return `County-${d.properties.COUNTY_NAM}: ${value != null ? value : "Data not
available"}`;
      }
    }),
    Plot.geo(mapData, { stroke: "black", fill: "none" })
  ],

  x: { axis: null },
  y: { axis: null }
})
```



```
cornHarvestedPercentMap= new Map(
  cropData.filter(d => d['County ANSI'] !== null && d.cornHarvested['2022'] !== null &&
d['LAND AREA'] !== null)
    .map(d => [d['County ANSI'],(d.cornHarvested['2022'] / d['LAND AREA']) * 100 ])//
Calculate the percentage
);
```





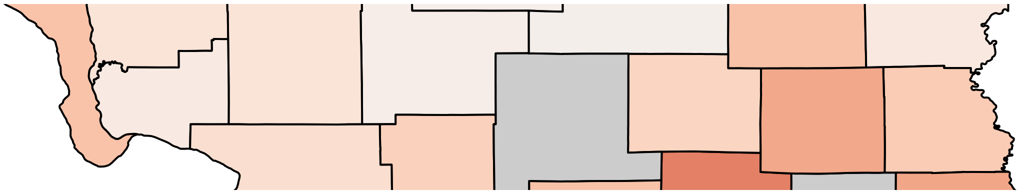
c. 2022 Corn-Soybean Difference (15 pts)

```
cornsoyDiffData = ▸Map(92) {39 => 0.19653392765193056, 107 => 5.865161949256237, 115 => 1.0
```

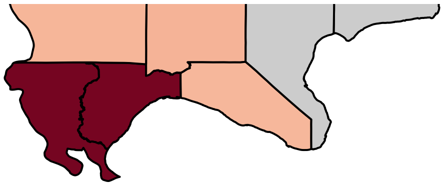
Corn-Soybean Difference (%)

```
  Plot.plot({
```

```
  width: 600,
  height: 900,
  color: {
    scheme: "Blues",
    type: "linear",
    label: "% Corn Acres Harvested",
    legend: true,
    unknown: "#ccc"
  },
  marks: [
    Plot.geo(mapData, {
      fill: d => cornHarvestedPercentMap.get(d.properties.CO_FIPS),
      title: d => {
        const value = cornHarvestedPercentMap.get(d.properties.CO_FIPS);
        return `${d.properties.COUNTY_NAM}\n% Corn Harvested: ${value != null ? value :
"Data not available"}%`;
      },
      stroke: "white"
    })
  ],

  //remove axes
  x: { axis: null },
  y: { axis: null },
})
```



```
cornsoyDiffData = new Map(
  cropData.filter(d => d['County ANSI'] !== null && d.cornHarvested['2022'] !== null &&
d.soybeansHarvested['2022'] !== null && d['LAND AREA'] !== null)
    .map(d => {
      const difference = ((d.cornHarvested['2022'] - d.soybeansHarvested['2022']) / d['LAND
AREA']) * 100;
      return [d['County ANSI'], difference];
    })
);
```



3.Corn Harvest Treemap (40 points)

```
Plot.plot({
  width: 600,
  height: 900,
  color:
  {
    type: "diverging",
    scheme: "RdBu",
    label: "Corn-Soybean Difference (%)",
    legend: true,
    unknown: "#ccc" // Color for counties without data
  },

  marks: [
    Plot.geo(mapData, {

      fill: d => {
        const value = cornsoyDiffData.get(d.properties.CO_FIPS);
        return value
      },
      title: d => {
        const value = cornsoyDiffData.get(d.properties.CO_FIPS);
        return `County: ${d.properties.COUNTY_NAM}\nCorn-Soybean % Difference: ${typeof
value === 'number' ? value : "Data not available"}`;
      },
      stroke: "black"
    })
  ],
  x: { axis: null },
  y: { axis: null },
})
```

```
// treemapData = {
//   const root = {
//     name: "root",
//     children: []
//   };


//   const dataByDistrict = d3.group(cropData, d => d['Ag District']);


//   for (const [district, entries] of dataByDistrict) {
//     const districtNode = {
//       name: district,
//       children: entries.map(entry => ({
//         name: entry.County,
//         value: entry.cornHarvested["2022"]
//       }))
//     };
//     root.children.push(districtNode);
//   }

//   return root;
// }


// leaves = {

//   const hierarchyRoot = d3.hierarchy(treemapData)
//     .sum(d => d.value)
//     .sort((a, b) => b.value - a.value);


//   const treemapLayout = d3.treemap()
//     .size([width, height])
//     .padding(1);


//   treemapLayout(hierarchyRoot);
//   return hierarchyRoot.leaves();
// }


// Plot.plot({
//   marks: [
//     Plot.rectY(leaves, {
//       x1: d => d.x0,
//       x2: d => d.x1,
//       y1: d => d.y0,
//       y2: d => d.y1,
//       fill: d => colorScale(d.parent.data.name),
//       title: d => `${d.parent.data.name}: ${d.data.name} (${d.value})`
//     }),
//     Plot.text(leaves, {
//       x: d => (d.x0 + d.x1) / 2,
//       y: d => (d.y0 + d.y1) / 2,
//       text: d => {
//         const v = d.value.toFixed(1);
//         const width = (v.length + 1) * 10;
//         return (d.x1 - d.x0) > width ? d.data.name : "";
//       },
//       fill: "white",
//       textAnchor: "middle",
//       baseline: "middle"
//     })
//   ],
//   color: {
//     scale: colorScale,
//     legend: true
//   },
//   x: { axis: null },
//   y: { axis: null },
//   width,
//   height,
//   marginTop: 20
// })
//Reference: https://observablehq.com/@ee2dev/making-a-treemap-and-sankey-diagram-with-observable-plot
```

```
treemapData = {
  const root = {
    name: "root",
    children: []
  };

  const dist = d3.group(cropData, d => d['Ag District']);

  for (const [district, entries] of dist) {
    const districtNode = {
      name: district,
      children: entries.map(entry => ({
        name: entry.County,
        value: entry.cornHarvested["2022"]
      }))
    };
    root.children.push(districtNode);
  }

  return root;
}
```

```
hierarchyRoot = d3.hierarchy(treemapData).sum(d => d.value).sort((a, b) => b.value - a.value);
```

```
treemap = {
  const layout = d3.treemap()
    .size([width, height])
    .padding(1);

  layout(hierarchyRoot);
  return hierarchyRoot;
}
```

Reference:https://observablehq.com/@dakoop/treemap

EXTRA CREDIT (upto 20 points)

all students may implement a way for users to interactively update which year (or crop) is shown for the visualizations shown in **Part 2 or Part 3** (up to 20 points).

=>I have implemented dropdown boxes that allows users to interactively update both year and croptype in **part 2** of the assignment for extra credit

```
// Cell 5: Drawing the SVG with Tooltip
TreeMap = {
  const svg = d3.create("svg")
    .attr("viewBox", [0, 0, width, height])
    .style("font", "10px sans-serif");

  const leaf = svg.selectAll("g")
    .data(hierarchyRoot.leaves())
    .join("g")
      .attr("transform", d => `translate(${d.x0},${d.y0})`);
  const lrect = leaf.append("rect")
    .attr("fill", d => colorScale(d.parent.data.name))
    .attr("width", d => d.x1 - d.x0)
    .attr("height", d => d.y1 - d.y0);

  // Tooltip to show Ag Distict Coutnty and Value
  lrect.append("title")
    .text(d => `Ag District: ${d.parent.data.name}\nCounty: ${d.data.name}\nValue: 
${d.data.value}`);
  //country label
  leaf.append("text")
    .attr("x", 3)
    .attr("y", 10)
    .text(d => d.data.name)
    .attr("font-size", "0.8em")
    .attr("fill", "white");

  const regionLabels = svg.selectAll("g.region")
    .data(hierarchyRoot.descendants().filter(d => d.depth === 1))
    .join("g")
      .attr("transform", d => `translate(${(d.x0 + d.x1) / 2},${(d.y0 + d.y1) / 2})`);

  //ag district label
  regionLabels.append("text")
    .attr("text-anchor", "middle")
    .text(d => d.data.name)
    .attr("fill", "white")
    .attr("font-size", "1.5em")
    .attr("font-weight", "bold");

  return svg.node();
}
```