Assignment 5

VIKRAMADITYA REDDY VARKALA

Z1973679

Data Visualization(CSCI 627)

2023-12-06

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

```
ilCrops = ▶ Array(112) [Object, Object, Object, Object, Object, Object, Object, Object, Obje
```

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

```
ilCropsMap = ▶ Map(103) {null => Object, 39 => Object, 107 => Object, 115 => Object, 123 =>
```

```
ilCropsMap = new Map(ilCrops.map(d => [d['County ANSI'],d]))
```

1. Filtering (30 pts)

```
import {interval} from "@mootari/range-slider"
```

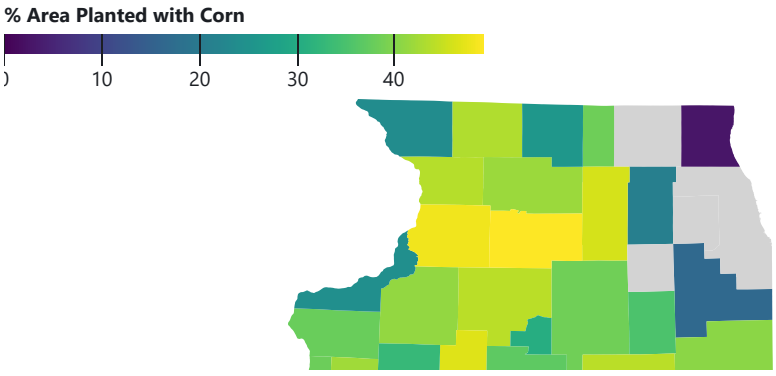
```
soybeanPercentage = ƒ(cdata)
```

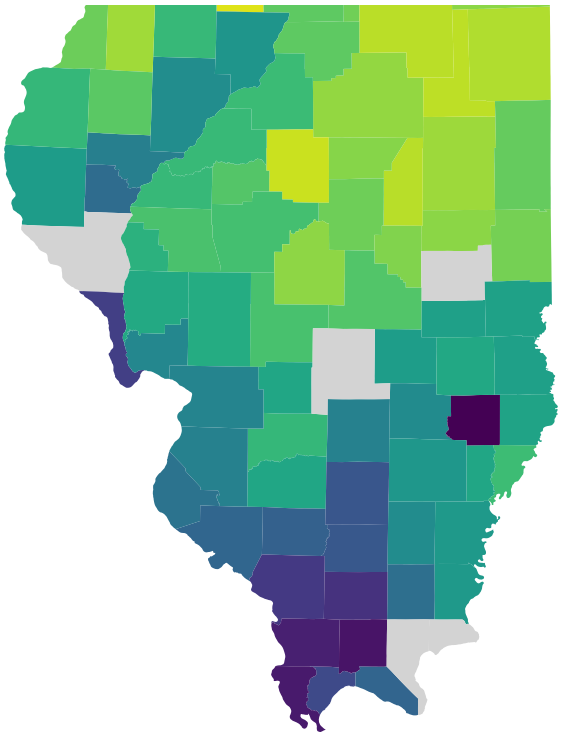
```
cornPercentage = ƒ(cdata)
```

```
maxCornPercentage = 49.145133338506504
```

% Soybean

0 … 100



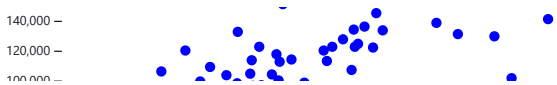


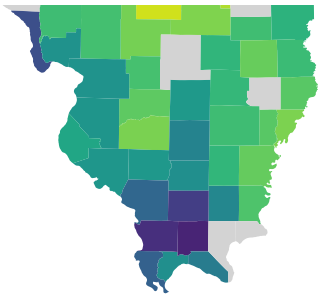
```
Plot.plot({
  projection: {
    type: "transverse-mercator",
    rotate: [88 + 20 / 60, -36 - 40 / 60],
    domain: counties
  },
  width: 500,
  color: {
    label: "% Area Planted with Corn",
    scheme: "Viridis",
    unknown: "lightgray", //for counties outside of range
    legend: true,
    domain: [0, maxCornPercentage]
  },
  marks: [
    Plot.geo(counties.features, {
      fill: d =>
      {
        const cdata = ilCropsMap.get(d.properties.CO_FIPS);
        const soybeanPerc = soybeanPercentage(cdata);
```

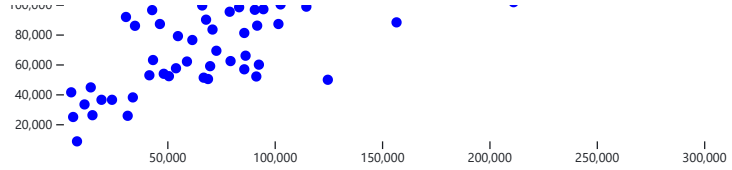
```
        return soybeanPerc >= soybeanRange[0]&&soybeanPerc <= soybeanRange[1] ?
cornPercentage(cdata) :"lightgray";
      }
    }),
  ],
});
```

```
farmedMap = Plot.plot({
  className: "farmedMap",
  projection: {
    type: "transverse-mercator",
    rotate: [88 + 20 / 60, -36 - 40 / 60],
    domain: counties
  },
  width: 500,
  color: {
    label: "% Area Planted with Corn or Soybeans",
    scheme: "viridis",
    unknown: "lightgray",
    legend: true
  },
  marks: [
    Plot.geo(counties.features, { fill: d =>
((ilCropsMap.get(d.properties.CO_FIPS).cornPlanted['2022'] ?? NaN) +
(ilCropsMap.get(d.properties.CO_FIPS).soybeansPlanted['2022'] ?? NaN)) /
ilCropsMap.get(d.properties.CO_FIPS)['LAND AREA'] }),
  ],
})
```

```
cropScatter = Plot.plot({
  className: "cropScatter",
  marginLeft: 60,
  width: 600,
  marks: [
    Plot.dot(counties.features, { x: d =>
(ilCropsMap.get(d.properties.CO_FIPS).cornPlanted['2022'] ?? NaN), y: d =>
(ilCropsMap.get(d.properties.CO_FIPS).soybeansPlanted['2022'] ?? NaN), fill: "blue", r: 4}),
  ],
})
```







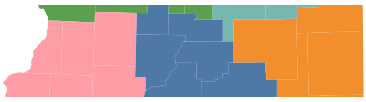
undefined

```
{

  const counties = d3.select(farmedMap).selectAll("path");
  const points = d3.select(cropScatter).selectAll("circle");

  // Map highlihting
  counties.on("pointerover", function(event, d) {
    counties.classed("highlight", false);
    points.classed("highlight", false);

    d3.select(event.currentTarget).classed("highlight", true).raise();
    points.filter(dd => dd === d).classed("highlight", true).raise();})
    .on("pointerout", function()
        {
          counties.classed("highlight", false);
          points.classed("highlight", false);
        });
```

```
  // Scatter points highlighting
  points.on("pointerover", function(event, d) {
    counties.classed("highlight", false);
    points.classed("highlight", false);

    d3.select(event.currentTarget).classed("highlight", true).raise();
    counties.filter(dd => dd === d).classed("highlight", true);})
    .on("pointerout", function()
        {
          counties.classed("highlight", false);
          points.classed("highlight", false);
        });
}
```

<detached>

```
regionMap = Plot.plot({
  className: "regionMap",
  projection: {
    type: "transverse-mercator",
    rotate: [88 + 20 / 60, -36 - 40 / 60],
    domain: counties
  },
  width: 500,
  color: {
    label: "Agricultural District",
    scheme: "Tableau10",
    legend: true
  },
  marks: [
    Plot.geo(counties.features, { fill: d => ilCropsMap.get(d.properties.CO_FIPS)['Ag
District'] }),
  ],
});
```




```
cropScatter2 = Plot.plot({
  className: "cropScatter2",
  marginLeft: 60,
  height:600,
  width: 800,
  marks: [
```

```
    Plot.dot(counties.features, { x: d =>
  (ilCropsMap.get(d.properties.CO_FIPS).cornPlanted['2022'] ?? NaN), y: d =>
  (ilCropsMap.get(d.properties.CO_FIPS).soybeansPlanted['2022'] ?? NaN), fill: "blue", r: 4}),
    ],
  })
```

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

```
undefined
```

```
AgDistMap = Object.fromEntries(counties.features.map((county) =>
[county.properties.CO_FIPS,ilCropsMap.get(county.properties.CO_FIPS)["Ag District"]]));
```

```
{
const AgRegions = d3.select(regionMap).selectAll("path");
const ScatterPoints = d3.select(cropScatter2).selectAll("circle");

// fucntion for district highlight
function regionHighlight(agDistrict) {
```

```
  AgRegions.classed("highlight", (d) => agDistrict ===
AgDistMap[counties.features[d].properties.CO_FIPS]);
  ScatterPoints .classed("highlight", (d) => agDistrict ===
AgDistMap[counties.features[d].properties.CO_FIPS]);
}

//pointer on
function mouseOn(event, d) {
  const id = counties.features[d].properties.CO_FIPS;
  const agDistrict = AgDistMap[id];
  regionHighlight(agDistrict);
}

//pointer out
function mouseOut() {
  AgRegions.classed("highlight", false);
 ScatterPoints.classed("highlight", false);
}
AgRegions.on("pointerover", mouseOn);
AgRegions.on("pointerout", mouseOut);
}

<style>
.highlight{
  stroke: red;
  stroke-width: 2px;
/*   fill:red;
  fill-opacity:0.4 */
}
</style>

<!-- i have used same styling for both part 2a and 2b and part 3 -->




farmedMap2 = Plot.plot({
  className: "farmedMap",
  projection: {
    type: "transverse-mercator",
    rotate: [88 + 20 / 60, -36 - 40 / 60],
    domain: counties
  },
  width: 500,
  color: {
    label: "% Area Planted with Corn or Soybeans",
    scheme: "viridis",
```

```
      unknown: "lightgray",
      legend: true
    },
    marks: [
      Plot.geo(counties.features, { fill: d =>
((ilCropsMap.get(d.properties.CO_FIPS).cornPlanted['2022'] ?? NaN) +
(ilCropsMap.get(d.properties.CO_FIPS).soybeansPlanted['2022'] ?? NaN)) /
ilCropsMap.get(d.properties.CO_FIPS)['LAND AREA'] }),
    ],
})
```


```
CropBinned = Plot.plot({
className: "cropBinned", marginLeft: 90, width: 700,
color: {
scheme: "Inferno",
legend:true,
},
marks: [
Plot.rect(counties. features,
          Plot.bin({fill: "count"},
        {
          x: d => (ilCropsMap.get(d.properties.CO_FIPS) .cornPlanted ['2022'] ?? NaN),
          y: d => (ilCropsMap.get (d.properties.CO_FIPS) .soybeansPlanted['2022'] ?? NaN),
fill: "", r: 4})),
  ],
})
```

Reference: https://talk.observablehq.com/t/how-to-get-the-bin-values-from-a-plot/8090/6

```
undefined
```

EXTRA CREDIT

select multiple counties in Part 2a

<detached>

```
binnedData = Plot.rect(counties.features,Plot.bin({fill: "count"}, {
  x: d => (ilCropsMap.get(d.properties.CO_FIPS).cornPlanted['2022'] ?? NaN),
  y: d => (ilCropsMap.get(d.properties.CO_FIPS).soybeansPlanted['2022'] ?? NaN),
  fill: "blue", r: 4
})).initialize()["data"];
```

```
{
  const countyMap = d3.select(farmedMap2).selectAll("path");
  const scatterplotBins = d3.select(CropBinned).selectAll("rect");

  scatterplotBins.on("pointerover", (event, id) => {
```

```
      d3.select(event.currentTarget).classed("highlight", true);

      const binHighlight = binnedData[id];
      const countyHighlight = binHighlight.map((d) => d.properties.CO_FIPS);

      countyMap
        .filter((dd) => {return
  countyHighlight.includes(counties.features[dd].properties["CO_FIPS"]);})
        .classed("highlight", true);
    });

    scatterplotBins.on("pointerout", (event, d) => {
      scatterplotBins.classed("highlight", false);
      countyMap.classed("highlight", false);
    });
  }
```

Click on counties for multiple selections. click again to deselect

```
undefined
```

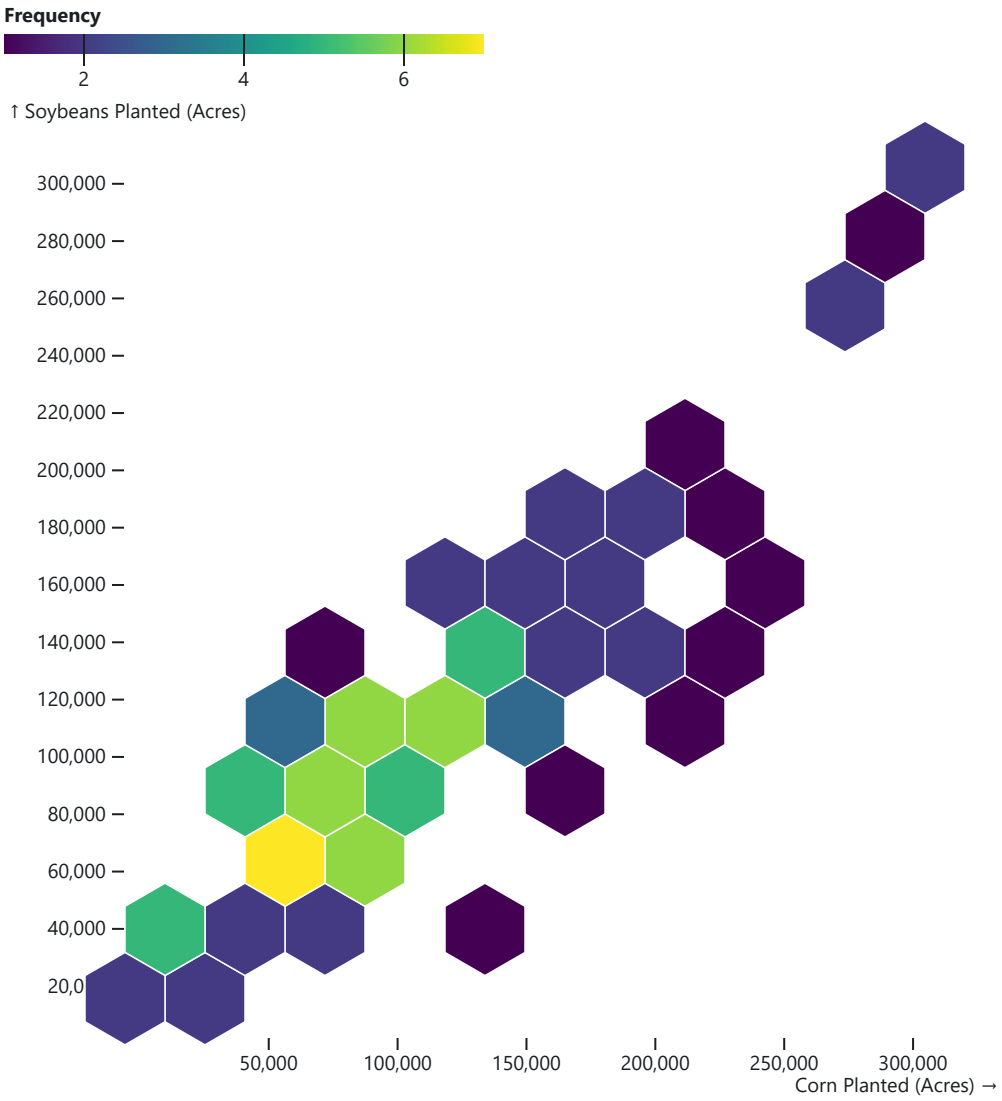
```
  farmedMapextra = Plot.plot({
    className: "farmedMap",
    projection: {
      type: "transverse-mercator",
      rotate: [88 + 20 / 60, -36 - 40 / 60],
      domain: counties
    },
    width: 500,
    color: {
      label: "% Area Planted with Corn or Soybeans",
      scheme: "viridis",
      unknown: "lightgray",
      legend: true
    },
    marks: [
      Plot.geo(counties.features, { fill: d =>
  ((ilCropsMap.get(d.properties.CO_FIPS).cornPlanted['2022'] ?? NaN) +
  (ilCropsMap.get(d.properties.CO_FIPS).soybeansPlanted['2022'] ?? NaN)) /
  ilCropsMap.get(d.properties.CO_FIPS)['LAND AREA'] }),
    ],
  })
```

```
  cropScatterextra = Plot.plot({
```

```
    className: "cropScatter",
    marginLeft: 60,
    width: 600,
    marks: [
      Plot.dot(counties.features, { x: d =>
(ilCropsMap.get(d.properties.CO_FIPS).cornPlanted['2022'] ?? NaN), y: d =>
(ilCropsMap.get(d.properties.CO_FIPS).soybeansPlanted['2022'] ?? NaN), fill: "blue", r: 4}),
    ],
})
```

Hexbin Scatterplot

```
d3 = ▶ Object {format: ƒ(t), formatPrefix: ƒ(t, n), timeFormat: ƒ(t), timeParse: ƒ(t), utcFo
```




```
{
  const counties = d3.select(farmedMapextra).selectAll("path");
  const points = d3.select(cropScatterextra).selectAll("circle");

  // Function to county selection
  function countyselection(element, data) {
```

```
      const click = d3.select(element).classed("selected");
      d3.select(element).classed("selected", !click);


      points.filter(dd => dd === data).classed("selected", !click).raise();
      counties.filter(dd => dd === data).classed("selected", !click);
    }

    // For selecting/deselecting counties
    counties.on("click", function(event, d) {
      countyselection(event.currentTarget, d);
    });

    // For selecting/deselecting scatterplot points
    points.on("click", function(event, d) {
      countyselection(event.currentTarget, d);
    });
  }
```

```
<style>
  .selected {
  stroke: red;
  stroke-width: 2px;
  fill:red;
  fill-opacity:0.4
}
</style>
```

```
d3 = require("d3@7", "d3-hexbin@0.2")
```

```
CropHexBinned = Plot.plot({
  width: 500,
  height: 500,
  marginLeft: 60,
  inset: 10,
  x: { label: "Corn Planted (Acres)", type: "linear" },
  y: { label: "Soybeans Planted (Acres)", type: "linear" },
  color: { scheme: "Viridis", legend: true },

  marks: [
    Plot.hexagon(
      counties.features.map(d => ({
        cornPlanted: ilCropsMap.get(d.properties.CO_FIPS).cornPlanted['2022'] ?? NaN,
        soybeansPlanted: ilCropsMap.get(d.properties.CO_FIPS).soybeansPlanted['2022'] ?? NaN
      })),
      Plot.hexbin(
        { fill: "count" },
        {
          binWidth: 40,
          x: d => d.cornPlanted,
          y: d => d.soybeansPlanted,
```

```
            stroke: "white",
            strokeWidth: 0.75
          }
        )
      )
    ],
  });
```