BHARATIYA VIDYA BHAVAN'S

SARDARPATEL INSTITUTE OF TECHNOLOGY

(Empowered Autonomous Institute Affiliated to University of Mumbai)

[Knowledge is Nectar]

Department of Computer Engineering

Advance Data Visualization

Name & UID	Bhagya Bijlaney (2021700010)
Class & batch	BE CSE DS (Batch L)
Subject	Advanced Data Visualization
Experiment No.	8
Title	Experiment to design interactive dashboards and create visual storytelling using D3.js on a dataset related to Environment/Forest cover, covering basic and advanced charts

Dataset Link :- https://www.kaggle.com/datasets/karnikakapoor/global-forest-data-2001-2022

About the Dataset

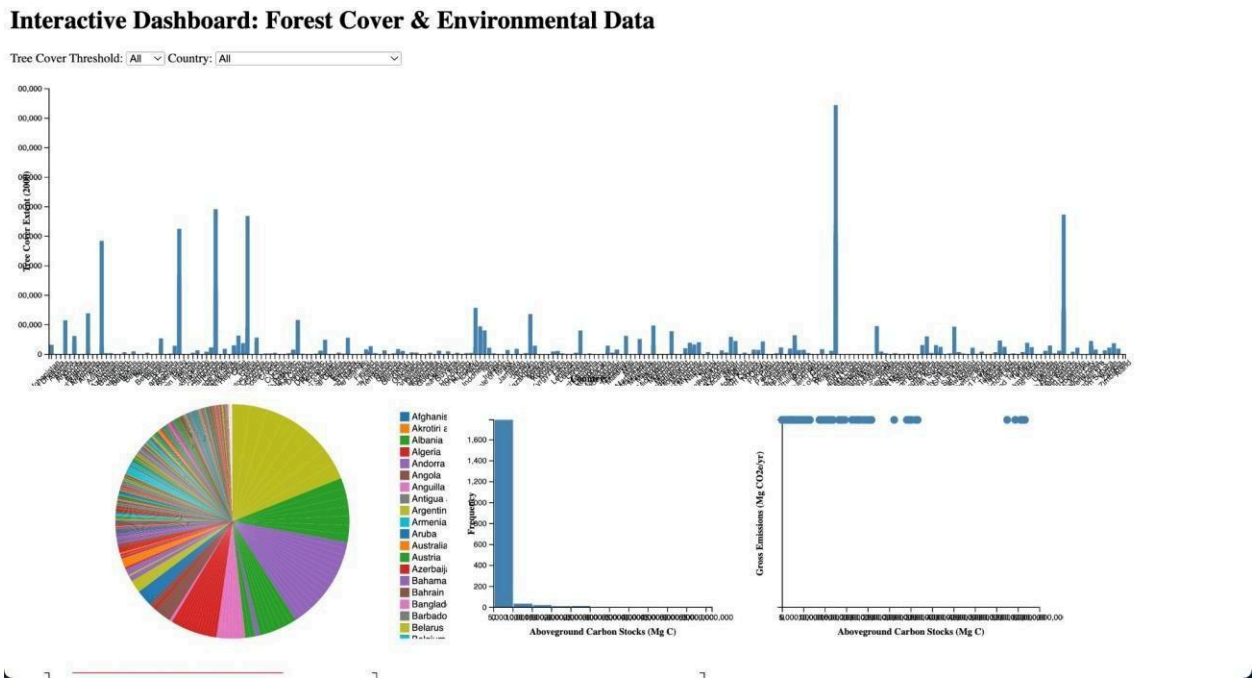
1. **Tree Cover Data (2001-2022)**:
   - Provides annual data on tree cover loss and associated emissions from 2001 to 2022.
   - Includes tree cover statistics for the years 2000 and 2010 for historical context.
2. **Biomass Stocks and Densities**:
   - Presents data on aboveground biomass stocks and densities for the year 2000.
3. **Carbon Metrics**:
   - Includes detailed carbon data: carbon densities, emissions, removals, and net fluxes.
   - Data is given in **megagrams of CO2-equivalent per year (Mg CO2e/yr)**.
   - Focuses on emissions from tree cover loss and forest disturbances, with data on CO2, methane (CH4), and nitrous oxide (N2O).
4. **Canopy Cover Thresholds**:

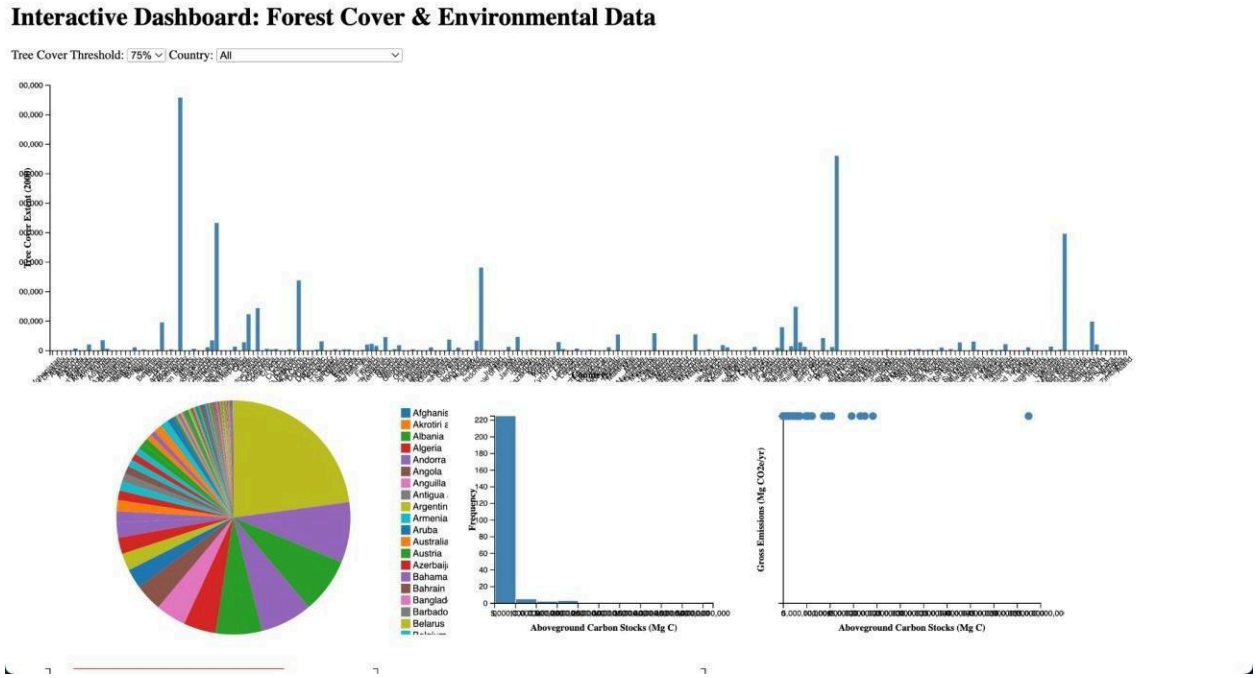
- Categorizes data based on various canopy cover thresholds (e.g., >10%, 15%, 20%, 25%, 30%, 50%, 75%).
- Emissions, removals, and net flux data are specifically provided for canopy covers greater than 30%, 50%, and 75% in the year 2000.

5. **Research and Collaboration**:
   - Developed through collaborations with the **University of Maryland's GLAD Laboratory**, **Google**, and other partners.
   - Based on the foundational research of **Hansen et al. (2013)** and **Harris et al. (2021)**.
6. **Use Cases**:
   - Essential for analyzing tree cover loss, forest carbon dynamics, and the climate impact of forest disturbances.
   - Valuable for environmental researchers, policymakers, and educators focusing on **forest conservation**, **carbon management**, and **climate change mitigation strategies**.

This dataset is a comprehensive resource for understanding the dynamics of forest ecosystems, carbon emissions, and the broader implications for climate change.

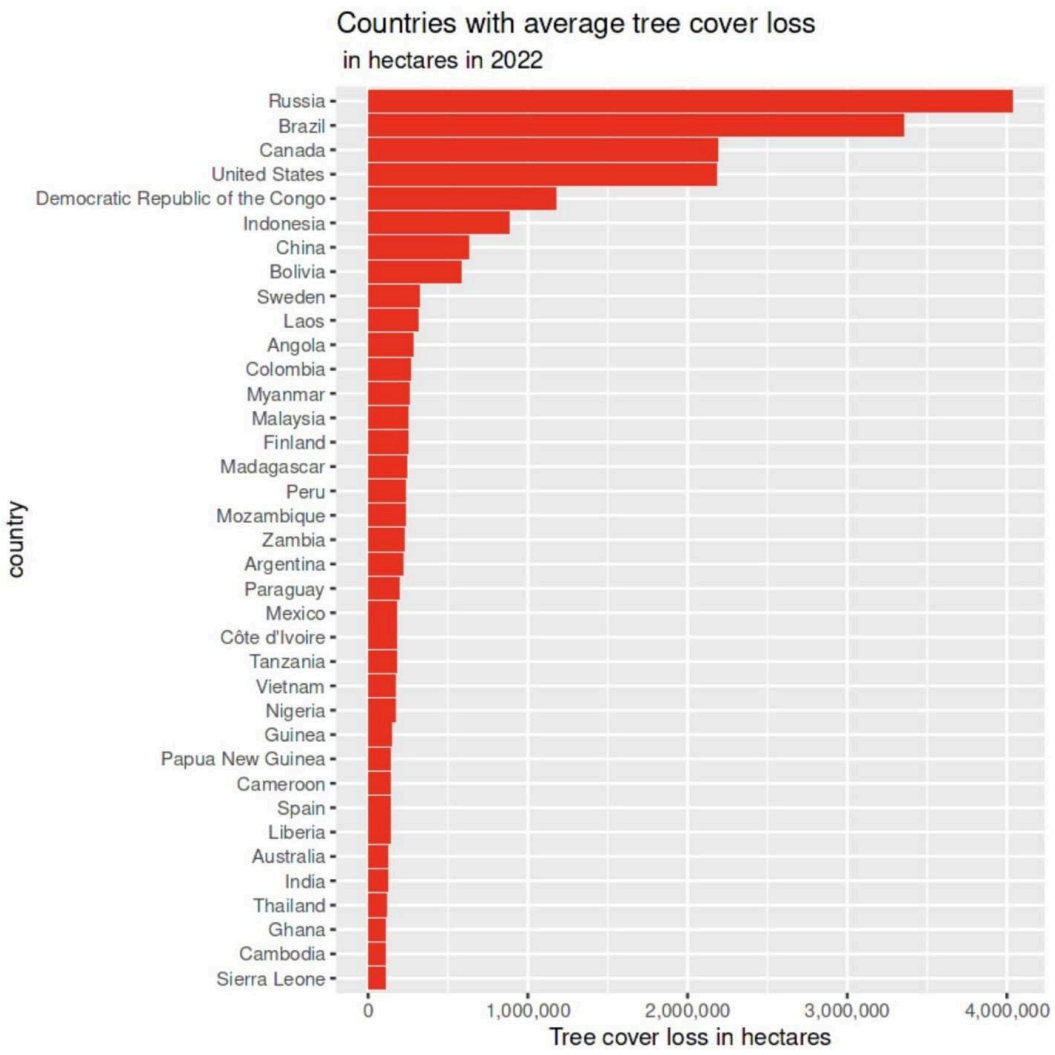
Charts





Observation -

- **The Aboveground carbon stocks are highest in argentina followed by Albania and Andorra**
- **For >=75% we have most sparse tree cover extent distribution**



Observation -

- **Russia has highest average tree cover loss followed by brazil and canada**
- **Distribution is almost equal for the lower quartile (countries like India, Ghana Sierra)**

Hypothesis testing

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Forest Cover Visualization</title>
```

```
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <style>
    .chart { display: inline-
      block;
      margin: 10px;
    }
    .axis-label { font-
      size: 12px;
      font-weight: bold;
    }
    .legend { font-
      size: 12px;
      font-family: sans-serif;
    }
  </style>
</head>
<body>
  <h1>Interactive Dashboard: Forest Cover & Environmental Data</h1>
  <div>
    <label for="thresholdFilter">Tree Cover Threshold: </label>
    <select id="thresholdFilter">
      <option value="all">All</option>
      <option value="0">0%</option>
      <option value="10">10%</option>
      <option value="20">20%</option>
      <option value="30">30%</option>
      <option value="50">50%</option>
      <option value="75">75%</option>
    </select>

    <label for="countryFilter">Country: </label>
    <select id="countryFilter">
      <option value="all">All</option>
    </select>
```

```
</div>
```

```
<div class="charts">
```

```
    </div>


  <script>
    d3.csv("forest.csv").then(function(data) {

        data.forEach(d => {
            d.umd_tree_cover_density_2000_threshold = +d.umd_tree_cover_density_2000_threshold;
            d.umd_tree_cover_extent_2000_ha = +d.umd_tree_cover_extent_2000_ha;
            d.gfw_aboveground_carbon_stocks_2000_Mg_C = +d.gfw_aboveground_carbon_stocks_2000_Mg_C;
            d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr =
+d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1;
        });
        const uniqueCountries = [...new Set(data.map(d => d.country))];
        const countryFilter = d3.select("#countryFilter");
        uniqueCountries.forEach(country => {
            countryFilter.append("option").text(country).attr("value", country);
        });


        function filterData() {
            let threshold = d3.select("#thresholdFilter").property("value");
            let country = d3.select("#countryFilter").property("value");


            let filteredData = data;
            if (threshold !== "all") {
                threshold = +threshold;

                filteredData = filteredData.filter(d => d.umd_tree_cover_density_2000_threshold === threshold);
            }
            if (country !== "all") { filteredData = filteredData.filter(d =>
                d.country === country);

            }
            return filteredData;
        }


        // Function to update charts when filters change
```

```
function updateCharts() {
    const filteredData = filterData();

    // Clear existing charts
    d3.select(".charts").html("");

    // Add basic charts
    createBarChart(filteredData);

    createPieChart(filteredData);

    createHistogram(filteredData);

    createScatterPlot(filteredData);

    // Add advanced charts

    createBoxPlot(filteredData);

    createViolinPlot(filteredData);
    createRegressionPlot(filteredData);

}

d3.select("#thresholdFilter").on("change", updateCharts);
d3.select("#countryFilter").on("change", updateCharts);

updateCharts();

function createBarChart(data) { const margin = {top: 20,
    right: 30, bottom: 40, left: 40};

    const width = 1450 - margin.left - margin.right;

    const height = 400 - margin.top - margin.bottom;
```

```
const svg = d3.select(".charts").append("svg")
    .attr("class", "chart")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", `translate(${margin.left},${margin.top})`);

const x = d3.scaleBand()
    .domain(data.map(d => d.country))
    .range([0, width])
    .padding(0.1);

const y = d3.scaleLinear()
    .domain([0, d3.max(data, d => d.umd_tree_cover_extent_2000_ha)])
    .nice()
    .range([height, 0]);

svg.append("g")
    .selectAll("rect")
    .data(data)
    .enter().append("rect")
    .attr("x", d => x(d.country))
    .attr("y", d => y(d.umd_tree_cover_extent_2000_ha))
    .attr("width", x.bandwidth())
    .attr("height", d => height - y(d.umd_tree_cover_extent_2000_ha))
    .attr("fill", "steelblue");
```

```
    svg.append("g")
        .attr("class", "axis")
        .attr("transform", `translate(0,${height})`)
        .call(d3.axisBottom(x))
        .selectAll("text")
        .attr("transform", "rotate(-45)")
        .style("text-anchor", "end");
    svg.append("g")
        .attr("class", "axis")
        .call(d3.axisLeft(y));

    svg.append("text")
        .attr("x", width / 2)
        .attr("y", height + margin.bottom - 5)
        .attr("text-anchor", "middle")
        .attr("class", "axis-label")
        .text("Country");

    svg.append("text")
        .attr("x", -height / 2)
        .attr("y", -margin.left + 15)
        .attr("text-anchor", "middle")
        .attr("transform", "rotate(-90)")
        .attr("class", "axis-label")
        .text("Tree Cover Extent (2000)");
}
```

```
function createPieChart(data) { const margin = {top: 20,
  right: 30, bottom: 40, left: 40};

  const width = 550;
```

```
  const height = 300;

  const radius = Math.min(width, height) / 2;


  const svg = d3.select(".charts").append("svg")

    .attr("class", "chart")

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

    .append("g")

    .attr("transform", `translate(${width / 2},${height / 2})`);


  const pie = d3.pie()

    .value(d => d.gfw_aboveground_carbon_stocks_2000_Mg_C);

  const arc = d3.arc()

    .innerRadius(0)
    .outerRadius(radius);


  const color = d3.scaleOrdinal(d3.schemeCategory10);


  const arcs = svg.selectAll("arc")

    .data(pie(data))

    .enter().append("g")

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


  arcs.append("path")
```

```
        .attr("d", arc)
        .attr("fill", d => color(d.data.country));

    svg.append("text")
        .attr("text-anchor", "middle")
        .attr("y", radius + 20)
        .attr("class", "axis-label")
        .text("Aboveground Carbon Stocks (2000)");

    const legend = svg.append("g")
        .attr("class", "legend")
        .attr("transform", `translate(${width / 2 - 60}, ${-height / 2 + 10})`);

    uniqueCountries.forEach((country, i) => {
        legend.append("rect")
            .attr("x", 0)
            .attr("y", i * 15)
            .attr("width", 12)
            .attr("height", 12)
            .attr("fill", color(country));

        legend.append("text")
            .attr("x", 15)
            .attr("y", i * 15 + 10)
            .text(country);
    });
}

function createHistogram(data) {
    const margin = {top: 20, right: 30, bottom: 40, left: 40};
```

```
const width = 350 - margin.left - margin.right;
const height = 300 - margin.top - margin.bottom;



const svg = d3.select(".charts").append("svg")
    .attr("class", "chart")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
```

```
    .attr("transform", `translate(${margin.left},${margin.top})`);


const x = d3.scaleLinear()
    .domain([0, d3.max(data, d => d.gfw_aboveground_carbon_stocks_2000_Mg_C)])
    .range([0, width]);


const histogram = d3.histogram()
```

```
    .value(d => d.gfw_aboveground_carbon_stocks_2000_Mg_C)
    .domain(x.domain())
    .thresholds(x.ticks(10));


const bins = histogram(data);



const y = d3.scaleLinear()
    .domain([0, d3.max(bins, d => d.length)])
    .range([height, 0]);



svg.selectAll("rect")
    .data(bins)
    .enter().append("rect")
    .attr("x", 1)
```

```
    .attr("transform", d => `translate(${x(d.x0)}, ${y(d.length)})`)
    .attr("width", d => x(d.x1) - x(d.x0) - 1)
    .attr("height", d => height - y(d.length))
    .attr("fill", "steelblue");


svg.append("g")
    .attr("class", "axis")
    .attr("transform", `translate(0,${height})`)
```

```
    .call(d3.axisBottom(x));


svg.append("g")
    .attr("class", "axis")
    .call(d3.axisLeft(y));


svg.append("text")
    .attr("x", width / 2)
    .attr("y", height + margin.bottom - 5)
    .attr("text-anchor", "middle")
    .attr("class", "axis-label")
    .text("Aboveground Carbon Stocks (Mg C)");


svg.append("text")
    .attr("x", -height / 2)
```

```
    .attr("y", -margin.left + 15)
    .attr("text-anchor", "middle")
    .attr("transform", "rotate(-90)")
    .attr("class", "axis-label")
```

```
        .text("Frequency");
}

function createScatterPlot(data) {
    const margin = {top: 20, right: 30, bottom: 40, left: 40};
    const width = 400 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const svg = d3.select(".charts").append("svg")
        .attr("class", "chart")
        .attr("width", width + margin.left + margin.right)
        .attr("height", height + margin.top + margin.bottom)
        .append("g")
        .attr("transform", `translate(${margin.left},${margin.top})`);

    const x = d3.scaleLinear()
        .domain(d3.extent(data, d => d.gfw_aboveground_carbon_stocks_2000_Mg_C))
        .nice()
        .range([0, width]);

    const y = d3.scaleLinear()
        .domain(d3.extent(data, d => d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1))
        .nice()
        .range([height, 0]);

    svg.append("g")
```

```
    .attr("class", "axis")
    .attr("transform", `translate(0,${height})`)
    .call(d3.axisBottom(x));


svg.append("g")
    .attr("class", "axis")
    .call(d3.axisLeft(y));
```

```

svg.append("g")
    .selectAll("circle")
    .data(data)
```

```
            .enter().append("circle")
            .attr("cx", d => x(d.gfw_aboveground_carbon_stocks_2000_Mg_C))
            .attr("cy", d => y(d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1))
            .attr("r", 5)
            .attr("fill", "steelblue");


        svg.append("text")
            .attr("x", width / 2)
            .attr("y", height + margin.bottom - 5)
            .attr("text-anchor", "middle")
            .attr("class", "axis-label")
            .text("Aboveground Carbon Stocks (Mg C)");


        svg.append("text")
            .attr("x", -height / 2)
            .attr("y", -margin.left + 15)
            .attr("text-anchor", "middle")
            .attr("transform", "rotate(-90)")
            .attr("class", "axis-label")
            .text("Gross Emissions (Mg CO2e/yr)");
    }


    function createRegressionPlot(data) { const
margin = {top: 20, right: 30, bottom: 40, left: 40};

const width = 400 - margin.left - margin.right;

const height = 300 - margin.top - margin.bottom;


const svg = d3.select(".charts").append("svg")
```

```
    .attr("class", "chart")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", `translate(${margin.left},${margin.top})`);
```

```
const x = d3.scaleLinear()
    .domain(d3.extent(data, d => d.gfw_aboveground_carbon_stocks_2000_Mg_C))
    .nice()
    .range([0, width]);

const y = d3.scaleLinear()
    .domain(d3.extent(data, d => d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1))
    .nice()
    .range([height, 0]);

svg.append("g")
    .attr("class", "axis")
    .attr("transform", `translate(0,${height})`)
    .call(d3.axisBottom(x));

svg.append("g")
    .attr("class", "axis")
    .call(d3.axisLeft(y));

// Fit a linear regression line
```

```
  const lr = linearRegression(data, d => d.gfw_aboveground_carbon_stocks_2000_Mg_C, d =>
```

```
d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1);


  // Plot regression line

  svg.append("line")

     .attr("x1", x(d3.min(data, d => d.gfw_aboveground_carbon_stocks_2000_Mg_C)))
     .attr("y1", y(lr.intercept + lr.slope * d3.min(data, d => d.gfw_aboveground_carbon_stocks_2000_Mg_C)))
     .attr("x2", x(d3.max(data, d => d.gfw_aboveground_carbon_stocks_2000_Mg_C)))
     .attr("y2", y(lr.intercept + lr.slope * d3.max(data, d => d.gfw_aboveground_carbon_stocks_2000_Mg_C)))
     .attr("stroke", "red")
     .attr("stroke-width", 2);



  svg.append("g")

     .selectAll("circle")
     .data(data)

     .enter().append("circle")
```

```
     .attr("cx", d => x(d.gfw_aboveground_carbon_stocks_2000_Mg_C))
     .attr("cy", d => y(d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1))
     .attr("r", 3)
     .attr("fill", "steelblue");


  svg.append("text")
     .attr("x", width / 2)
     .attr("y", height + margin.bottom - 5)
     .attr("text-anchor", "middle")
     .attr("class", "axis-label")
     .text("Regression Plot of Emissions vs. Carbon Stocks");
}
```

```
function createViolinPlot(data) {
  const margin = {top: 20, right: 30, bottom: 40, left: 40};
  const width = 400 - margin.left - margin.right;

  const height = 300 - margin.top - margin.bottom;


  const svg = d3.select(".charts").append("svg")
     .attr("class", "chart")
     .attr("width", width + margin.left + margin.right)
     .attr("height", height + margin.top + margin.bottom)
     .append("g")
     .attr("transform", `translate(${margin.left},${margin.top})`);


  const y = d3.scaleLinear()
     .domain([0, d3.max(data, d => d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1)])
     .range([height, 0]);


  const x = d3.scaleBand()
     .domain(["Emissions"])
     .range([0, width])
     .padding(0.2);


  const histogram = d3.histogram()
     .value(d => d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1)
     .domain(y.domain())
     .thresholds(y.ticks(20));


  const bins = histogram(data);


  const density = bins.map(b => {
     const densityValue = b.length / (d3.sum(bins, b => b.length) * (b.x1 - b.x0));
```

```
    return {x: (b.x0 + b.x1) / 2, y: densityValue};
});
```

```
svg.append("path")
    .datum(density)
    .attr("fill", "steelblue")
    .attr("opacity", 0.5)
    .attr("d", d3.area()
        .x(d => x("Emissions") + x.bandwidth() / 2)
        .y0(height)
        .y1(d => y(d.y))
    );

svg.append("g")
    .attr("class", "axis")
    .attr("transform", `translate(0,${height})`)
    .call(d3.axisBottom(x));

svg.append("g")
    .attr("class", "axis")
    .call(d3.axisLeft(y));

svg.append("text")
    .attr("x", width / 2)
    .attr("y", height + margin.bottom - 5)
    .attr("text-anchor", "middle")
    .attr("class", "axis-label")
```

```
    .text("Violin Plot of Forest Emissions");

}
```

```
function createBoxPlot(data) {
  const margin = {top: 20, right: 30, bottom: 40, left: 40};
  const width = 400 - margin.left - margin.right;
  const height = 300 - margin.top - margin.bottom;


  const svg = d3.select(".charts").append("svg")
    .attr("class", "chart")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", `translate(${margin.left},${margin.top})`);

  const y = d3.scaleLinear()
    .domain([0, d3.max(data, d => d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1)])
    .range([height, 0]);

  const x = d3.scaleBand()
    .domain(["Emissions"])
    .range([0, width])
    .padding(0.1);


  const q1 = d3.quantile(data.map(d =>
d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1).sort(d3.ascending), 0.25);
  const median = d3.quantile(data.map(d =>
d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1).sort(d3.ascending), 0.5);
  const q3 = d3.quantile(data.map(d =>
d.gfw_forest_carbon_gross_emissions_Mg_CO2e_yr-1).sort(d3.ascending), 0.75);
  const iqr = q3 - q1;
```

```
svg.append("rect")
    .attr("x", x("Emissions"))
    .attr("y", y(q3))
    .attr("height", y(q1) - y(q3))
    .attr("width", x.bandwidth())
```

```
    .attr("fill", "steelblue");
svg.append("line")
    .attr("x1", x("Emissions"))
    .attr("x2", x("Emissions") + x.bandwidth())
    .attr("y1", y(median))
    .attr("y2", y(median))
    .attr("stroke", "black");
svg.append("line")
    .attr("x1", x("Emissions"))
    .attr("x2", x("Emissions") + x.bandwidth())
    .attr("y1", y(q3 + 1.5 * iqr))
    .attr("y2", y(q3 + 1.5 * iqr))
    .attr("stroke", "red");
svg.append("line")
    .attr("x1", x("Emissions"))
    .attr("x2", x("Emissions") + x.bandwidth())
    .attr("y1", y(q1 - 1.5 * iqr))
    .attr("y2", y(q1 - 1.5 * iqr))
    .attr("stroke", "red");
svg.append("g")
    .attr("class", "axis")
    .attr("transform", `translate(0,${height})`)
    .call(d3.axisBottom(x));
svg.append("g")
    .attr("class", "axis")
    .call(d3.axisLeft(y));
svg.append("text")
    .attr("x", width / 2)
```

```
      .attr("y", height + margin.bottom - 5)
      .attr("text-anchor", "middle")
      .attr("class", "axis-label")
      .text("Box Plot of Forest Emissions");
}
      });
  </script>
</body>
```

```
</html>
```

Conclusion - In this experiment, we visualized the forestation data using d3.js library allowing users to interact with the threshold values with the dashboard.