Name	Rucha Kulkarni
Class	BE Computer Engineering (Batch F)
UID	2021300067
Exp No.	8

Aim: To design interactive dashboards and create visual storytelling using D3.js on a dataset related to Environment/Forest cover, covering basic and advanced charts

Objectives:

- To understand how to use D3.js for data visualization.
- To implement basic charts like Bar chart, Pie chart, Histogram, Timeline chart, Scatter plot, and Bubble plot.
- To implement advanced charts like Word chart, Box and whisker plot, Violin plot, Regression plot (linear and nonlinear), 3D chart, and Jitter.
- To draw observations and insights from each chart.
- To create an interactive storytelling dashboard using the above visualizations.

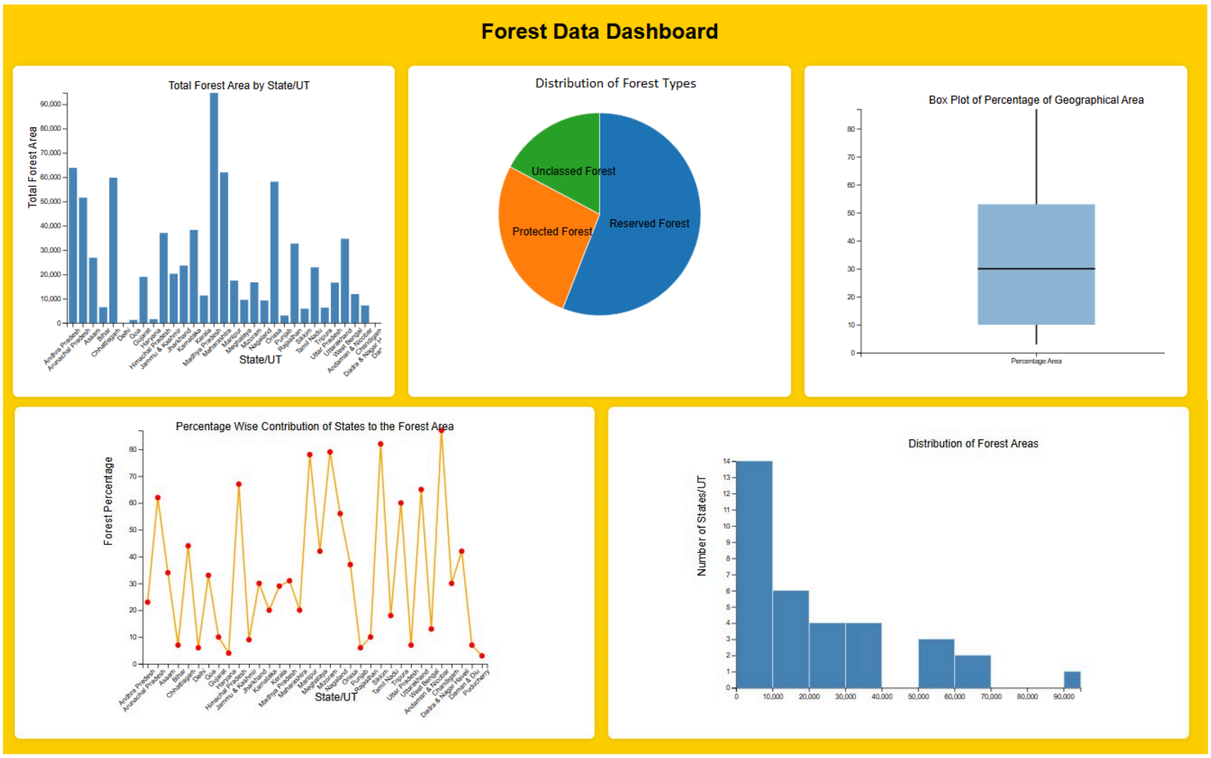
Description:

Dataset used is Forest Cover in India Dataset available at

https://www.kaggle.com/datasets/arjunprasadsarkhel/forest-cover-in-india

Data about Forest Cover in States/UTs in India in 2019, includes state-wise data which contains the geographical area(area in sq. km), various types of forest, percentage of geographical area.

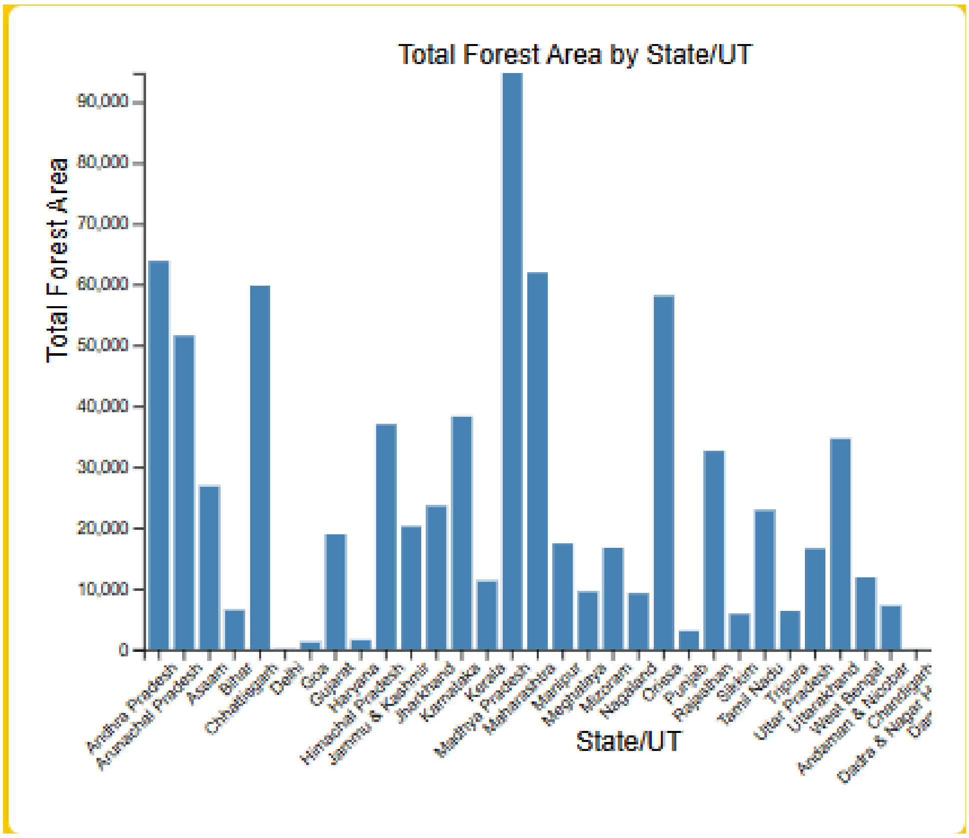
Dashboard:



Story: This dashboard presents a comprehensive overview of forest data across India, focusing on the total forest area by state, percentage-wise contribution of states to the overall forest area, distribution of forest types, and the distribution of forest areas.The data shows that Madhya Pradesh has the highest total forest area, followed by Arunachal Pradesh and Chhattisgarh. The data also shows that the majority of the forest area is reserved forest, followed by protected forest and unclassed forest. The data also shows that the percentage of geographical area covered by forests is highest in the Andaman & Nicobar Islands, followed by Mizoram and Arunachal Pradesh. The data also shows that the distribution of forest areas is skewed towards the lower end, with a large number of states having less than 20,000 sq km of forest area.

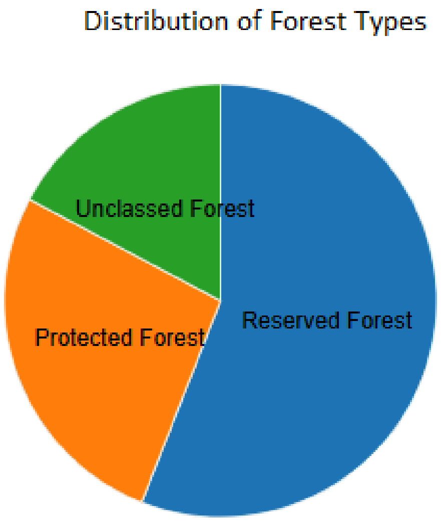
Graphs and Observations:

Bar chart:



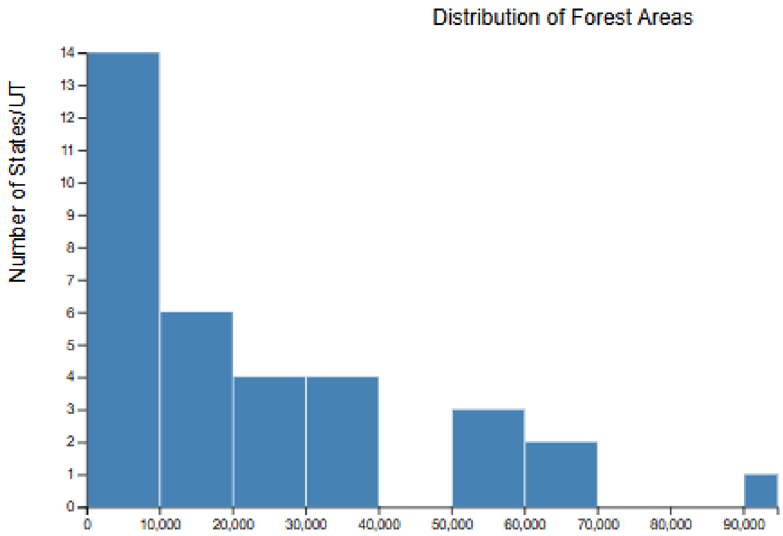
Observation: The bar graph shows the distribution of total forest area by State/ Union Territory.Madhya Pradesh has the largest forest area amongst the states of India. Andhra Pradesh, Chattisgarh, Maharashtra and Orissa also have a good amount of forest area.

Pie Chart:



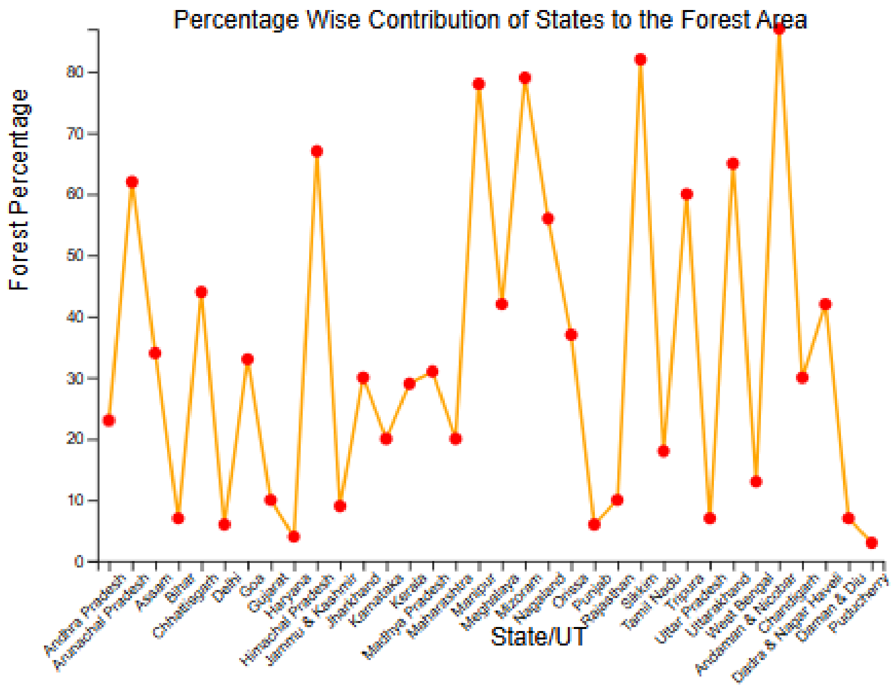
Observation: The pie chart shows the distribution of Forest Types. It indicates that the area for Reserved Forest is significantly larger than Protected Forest and Unclassed Forest.

Histogram:



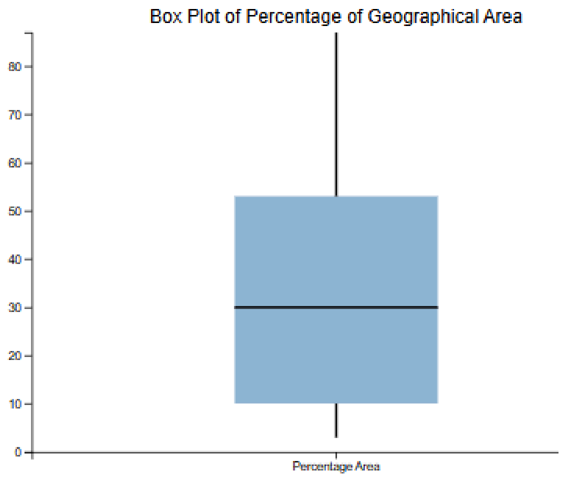
Observation: The histogram of Forest Areas indicates that the highest concentration of forest area is between 0 to 10000 square kilometers and most of the Indian states have their forest area in this group.

Line Chat:



Observation: The line plot indicates how each state contributes to the total forest area in percentages. We see that Andaman and Nicobar contribute a large portion of its geographical area to the forest.

Box Plot:

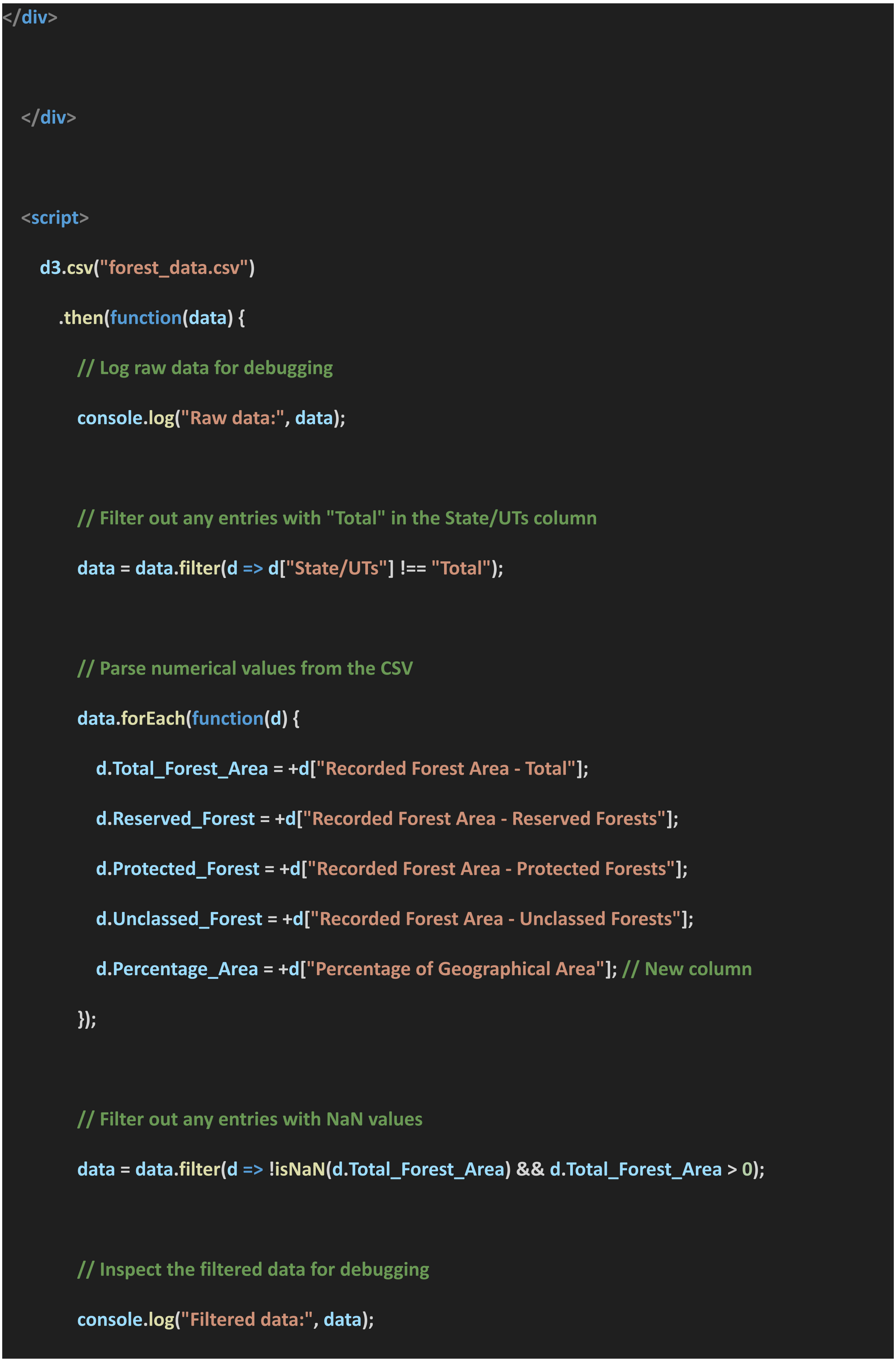


Observation: The box plot of percentage of Geographical area allotted to forest shows that the median forest area amongst the states of India is around 30% of the area of that state.

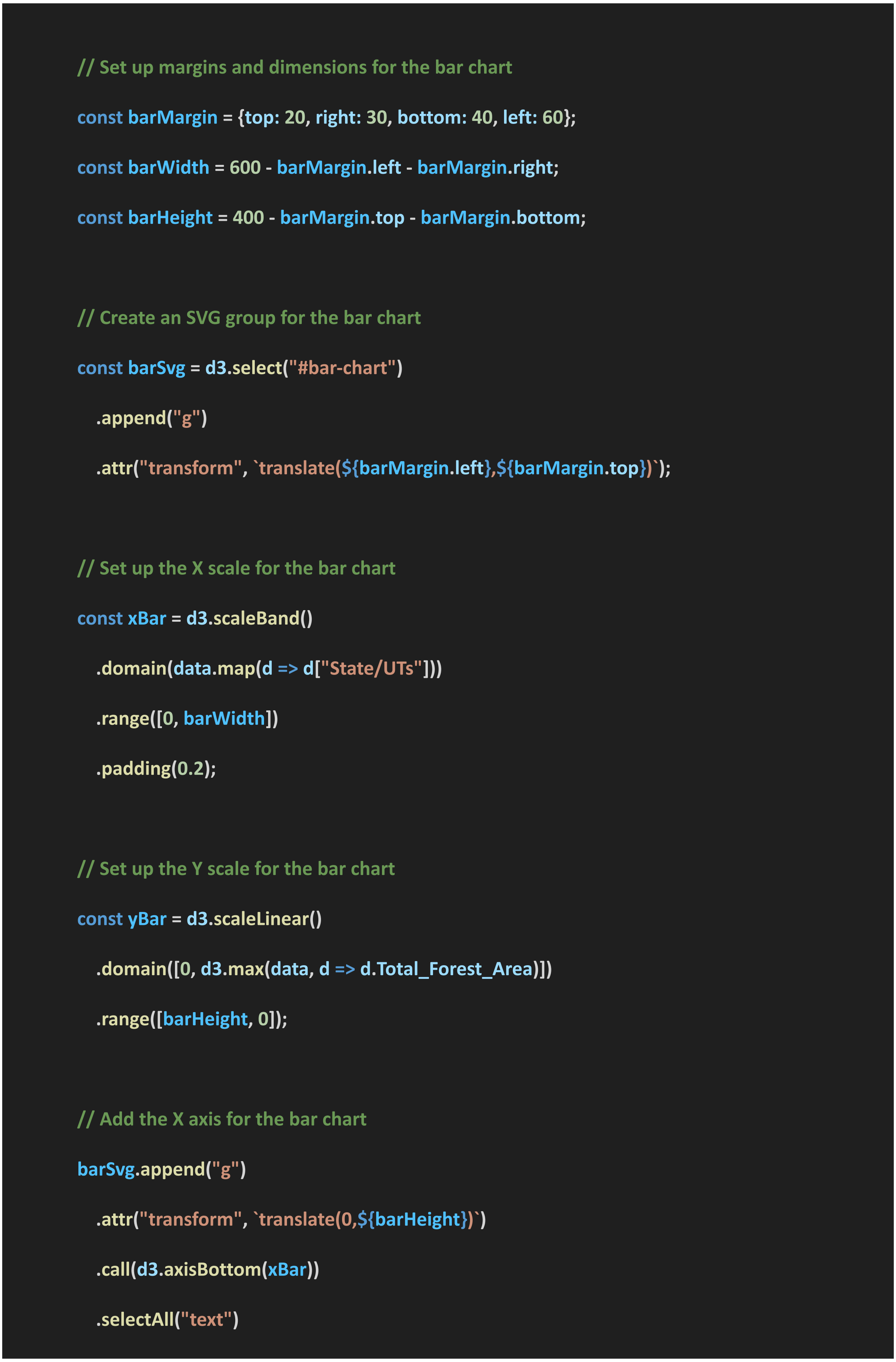
Code in D3.js:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Forest Data Dashboard</title>
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <style>
    body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 20px;
  background-color: rgb(255, 208, 0);
}

h1 {
  text-align: center;
}

.dashboard {
  display: flex;
  flex-wrap: wrap;
  justify-content: center; /* Center charts horizontally */
  width: 100%;
  margin: 0 auto; /* Center the container itself */
}
```

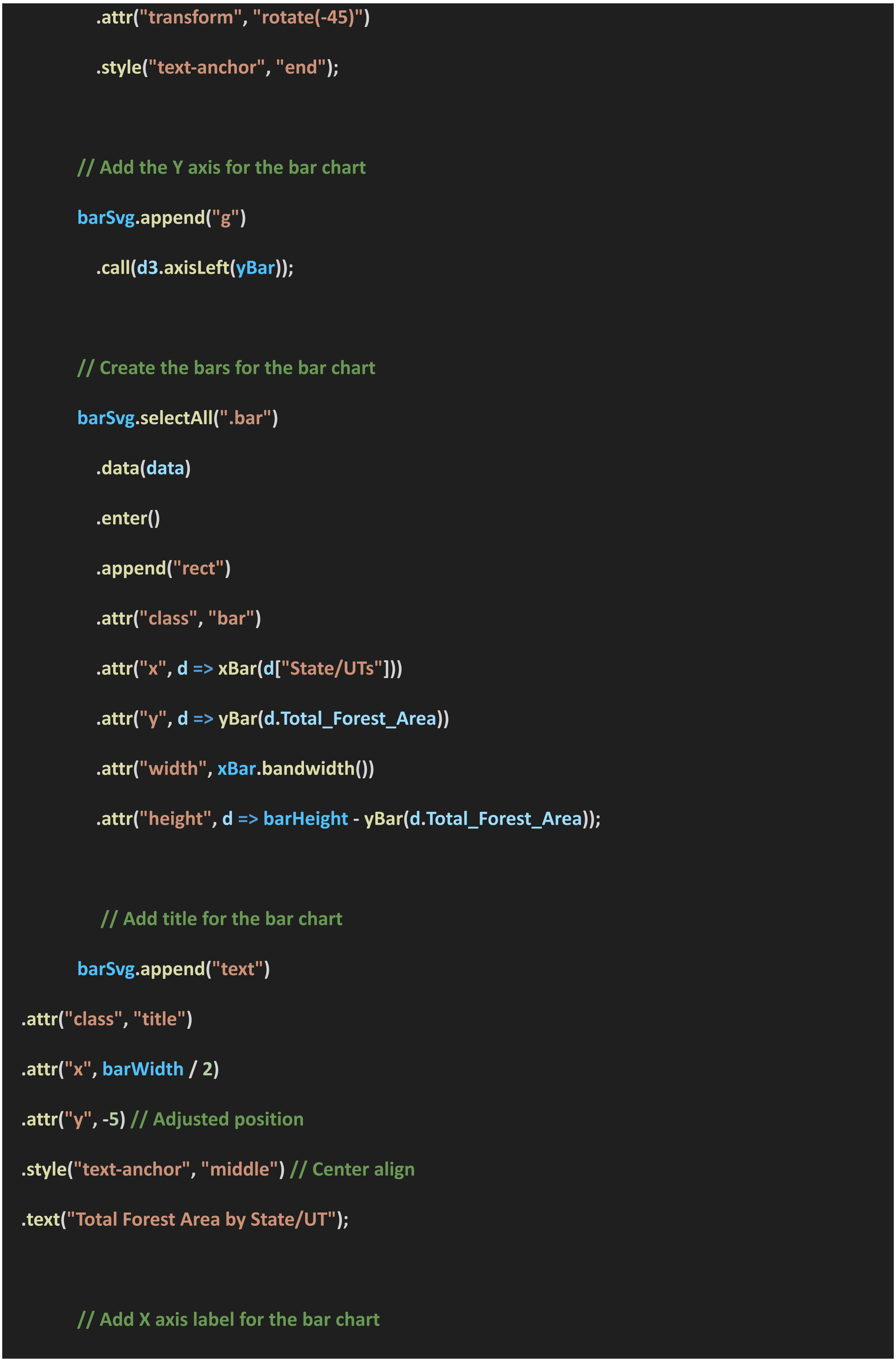
```css
.chart-container {
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    margin: 10px;
    padding: 20px;
    flex: 1;
    min-width: 400px; /* Minimum width for each chart */
    height: auto; /* Adjust height automatically based on content */
    display: flex;
    justify-content: center; /* Center the chart horizontally */
    align-items: center; /* Center the chart vertically */
}

/* Center the SVG within the container */
svg {
    display: block;
    margin: 0 auto; /* This centers the SVG horizontally */
    max-width: 100%; /* Prevents the SVG from overflowing its container */
}

/* Chart hover and styles */
.bar {
    fill: steelblue;
}
.bar:hover {
    fill: orange;
}
.line {
    fill: none;
    stroke: orange;
    stroke-width: 2;
}
.line-point {
    fill: red;
}
.box {
    fill: steelblue;
    opacity: 0.6;
}
.median-line {
    stroke: black;
    stroke-width: 2;
}
.axis line, .axis path {
    fill: none;
    shape-rendering: crispEdges;
}
.whisker {
    stroke: black;
    stroke-width: 2;
}
.outlier {
  fill: red;

}



  </style>

</head>

<body>

  <h1>Forest Data Dashboard</h1>

  <div class="dashboard">

    <div class="chart-container">



  <svg id="bar-chart" width="600" height="450"></svg> <!-- Fixed width -->

</div>



<div class="chart-container">

  <svg id="pie-chart" width="300" height="350"></svg> <!-- Fixed width -->

</div>



<div class="chart-container">

  <svg id="box-plot" width="500" height="400"></svg> <!-- Fixed width -->

</div>



<div class="chart-container">

  <svg id="line-chart" width="600" height="450"></svg> <!-- Fixed width -->

</div>



<div class="chart-container">

  <svg id="histogram" width="600" height="400"></svg> <!-- Fixed width -->
```



```
</div>

  </div>

  <script>
    d3.csv("forest_data.csv")
      .then(function(data) {
        // Log raw data for debugging
        console.log("Raw data:", data);

        // Filter out any entries with "Total" in the State/UTs column
        data = data.filter(d => d["State/UTs"] !== "Total");

        // Parse numerical values from the CSV
        data.forEach(function(d) {
          d.Total_Forest_Area = +d["Recorded Forest Area - Total"];
          d.Reserved_Forest = +d["Recorded Forest Area - Reserved Forests"];
          d.Protected_Forest = +d["Recorded Forest Area - Protected Forests"];
          d.Unclassed_Forest = +d["Recorded Forest Area - Unclassed Forests"];
          d.Percentage_Area = +d["Percentage of Geographical Area"]; // New column
        });

        // Filter out any entries with NaN values
        data = data.filter(d => !isNaN(d.Total_Forest_Area) && d.Total_Forest_Area > 0);

        // Inspect the filtered data for debugging
        console.log("Filtered data:", data);
```

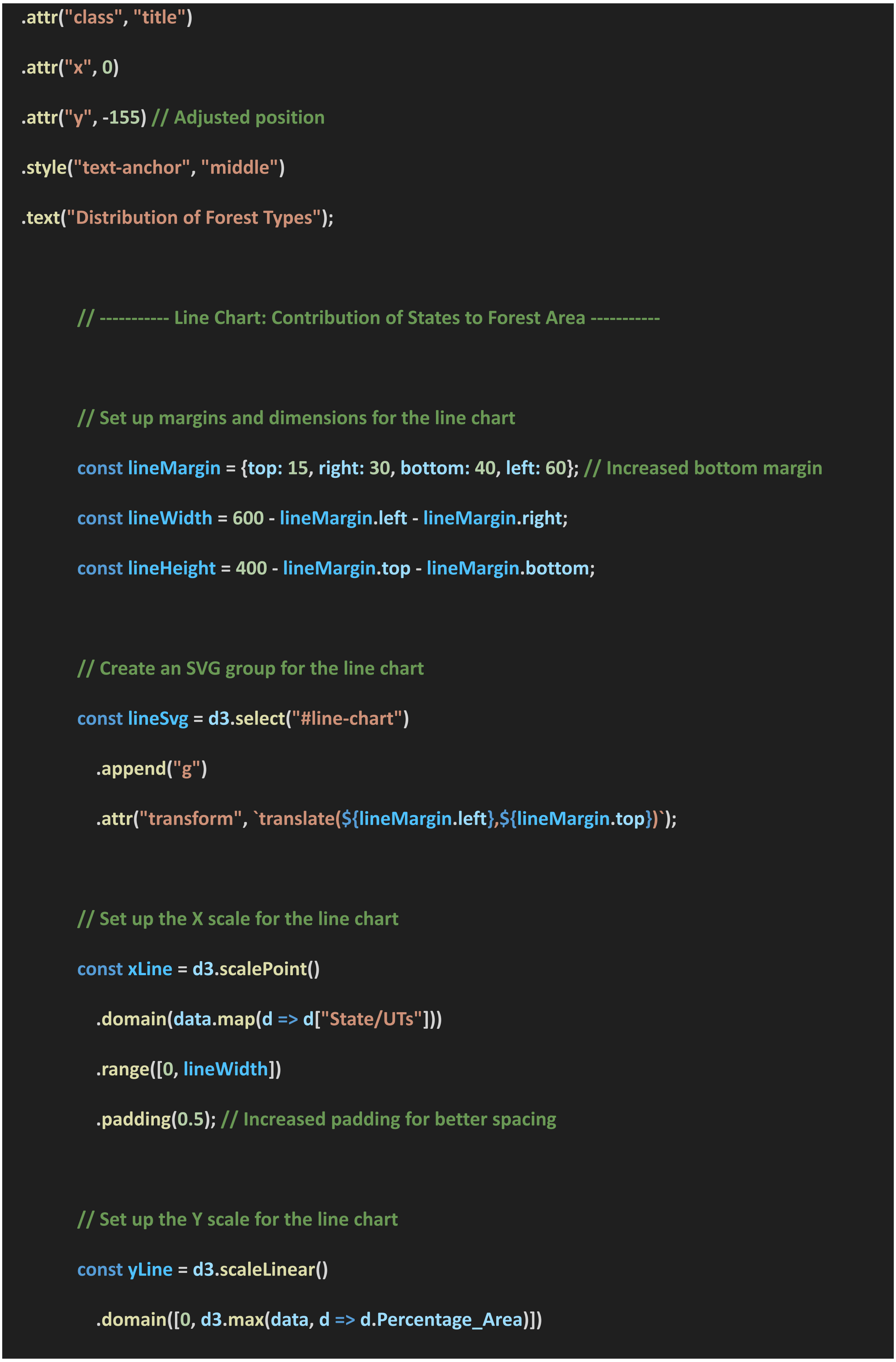


```javascript
// Set up margins and dimensions for the bar chart
const barMargin = {top: 20, right: 30, bottom: 40, left: 60};
const barWidth = 600 - barMargin.left - barMargin.right;
const barHeight = 400 - barMargin.top - barMargin.bottom;

// Create an SVG group for the bar chart
const barSvg = d3.select("#bar-chart")
   .append("g")
   .attr("transform", `translate(${barMargin.left},${barMargin.top})`);

// Set up the X scale for the bar chart
const xBar = d3.scaleBand()
   .domain(data.map(d => d["State/UTs"]))
   .range([0, barWidth])
   .padding(0.2);

// Set up the Y scale for the bar chart
const yBar = d3.scaleLinear()
   .domain([0, d3.max(data, d => d.Total_Forest_Area)])
   .range([barHeight, 0]);

// Add the X axis for the bar chart
barSvg.append("g")
   .attr("transform", `translate(0,${barHeight})`)
   .call(d3.axisBottom(xBar))
   .selectAll("text")
```



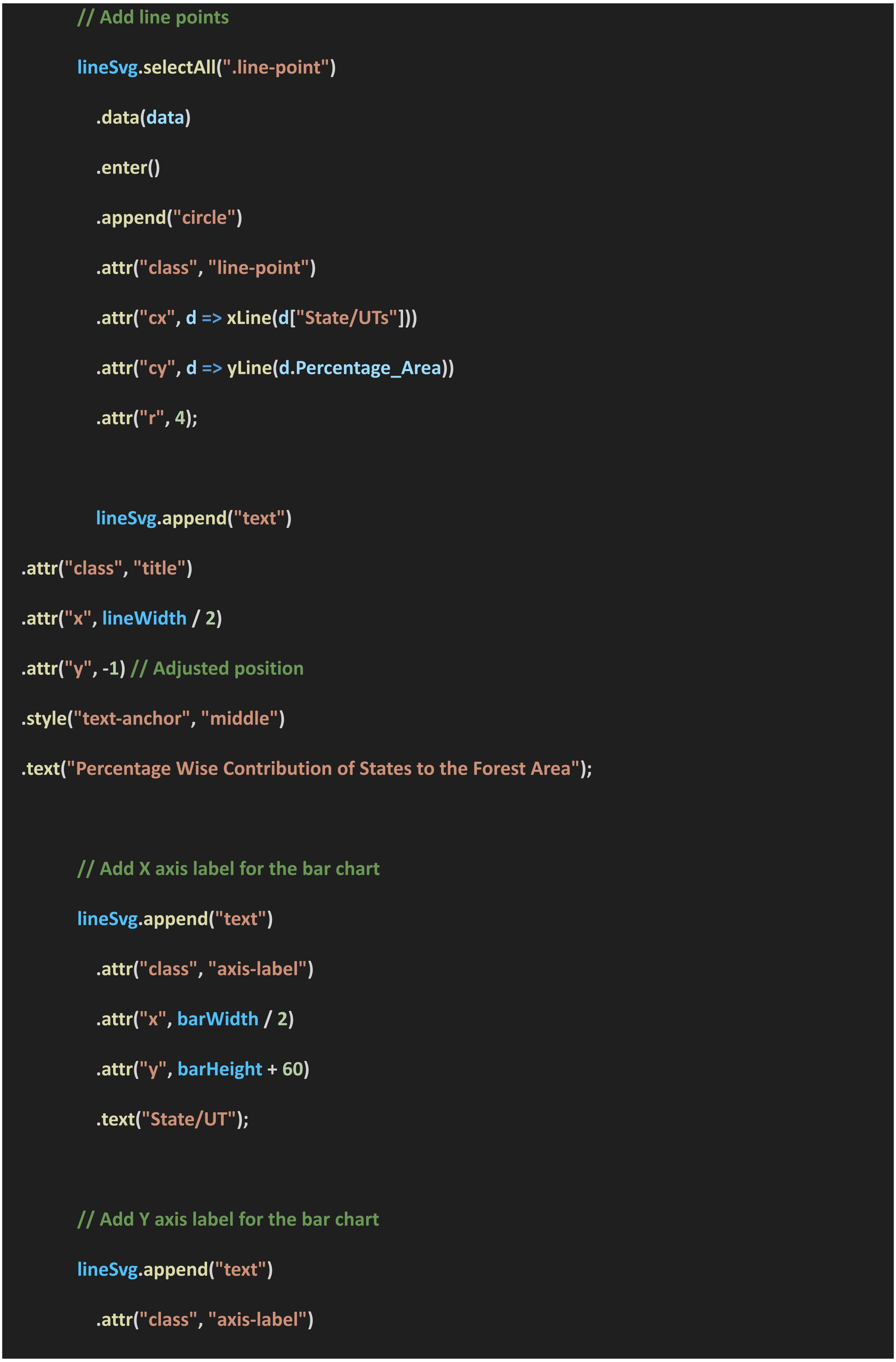
```
            .attr("transform", "rotate(-45)")
            .style("text-anchor", "end");

        // Add the Y axis for the bar chart
        barSvg.append("g")
            .call(d3.axisLeft(yBar));

        // Create the bars for the bar chart
        barSvg.selectAll(".bar")
            .data(data)
            .enter()
            .append("rect")
            .attr("class", "bar")
            .attr("x", d => xBar(d["State/UTs"]))
            .attr("y", d => yBar(d.Total_Forest_Area))
            .attr("width", xBar.bandwidth())
            .attr("height", d => barHeight - yBar(d.Total_Forest_Area));

            // Add title for the bar chart
        barSvg.append("text")
.attr("class", "title")
.attr("x", barWidth / 2)
.attr("y", -5) // Adjusted position
.style("text-anchor", "middle") // Center align
.text("Total Forest Area by State/UT");

        // Add X axis label for the bar chart
```

```javascript
barSvg.append("text")

    .attr("class", "axis-label")

    .attr("x", barWidth / 2)

    .attr("y", barHeight + 60)

    .text("State/UT");


// Add Y axis label for the bar chart

barSvg.append("text")

    .attr("class", "axis-label")

    .attr("transform", "rotate(-90)")

    .attr("y", -45)

    .attr("x", -barHeight /2)

    .text("Total Forest Area");


// ----------- Pie Chart: Distribution of Forest Types -----------


// Prepare the data for the pie chart

const pieData = [

    { type: "Reserved Forest", value: d3.sum(data, d => d.Reserved_Forest) },

    { type: "Protected Forest", value: d3.sum(data, d => d.Protected_Forest) },

    { type: "Unclassed Forest", value: d3.sum(data, d => d.Unclassed_Forest) }

];


// Set up dimensions and radius for the pie chart

const pieRadius = Math.min(300, 300) / 2;

const pieSvg = d3.select("#pie-chart")

    .append("g")
    .attr("transform", `translate(${pieRadius},${pieRadius})`);

// Create pie and arc generators
const pie = d3.pie().value(d => d.value);
const arc = d3.arc().innerRadius(0).outerRadius(pieRadius);

// Add the pie chart segments
pieSvg.selectAll("path")
    .data(pie(pieData))
    .enter()
    .append("path")
    .attr("d", arc)
    .attr("fill", (d, i) => d3.schemeCategory10[i])
    .attr("stroke", "white")
    .attr("stroke-width", 1);

// Add labels
pieSvg.selectAll("text")
    .data(pie(pieData))
    .enter()
    .append("text")
    .attr("transform", d => `translate(${arc.centroid(d)})`)
    .attr("dy", "0.35em")
    .style("text-anchor", "middle")
    .text(d => d.data.type);

    pieSvg.append("text")
```



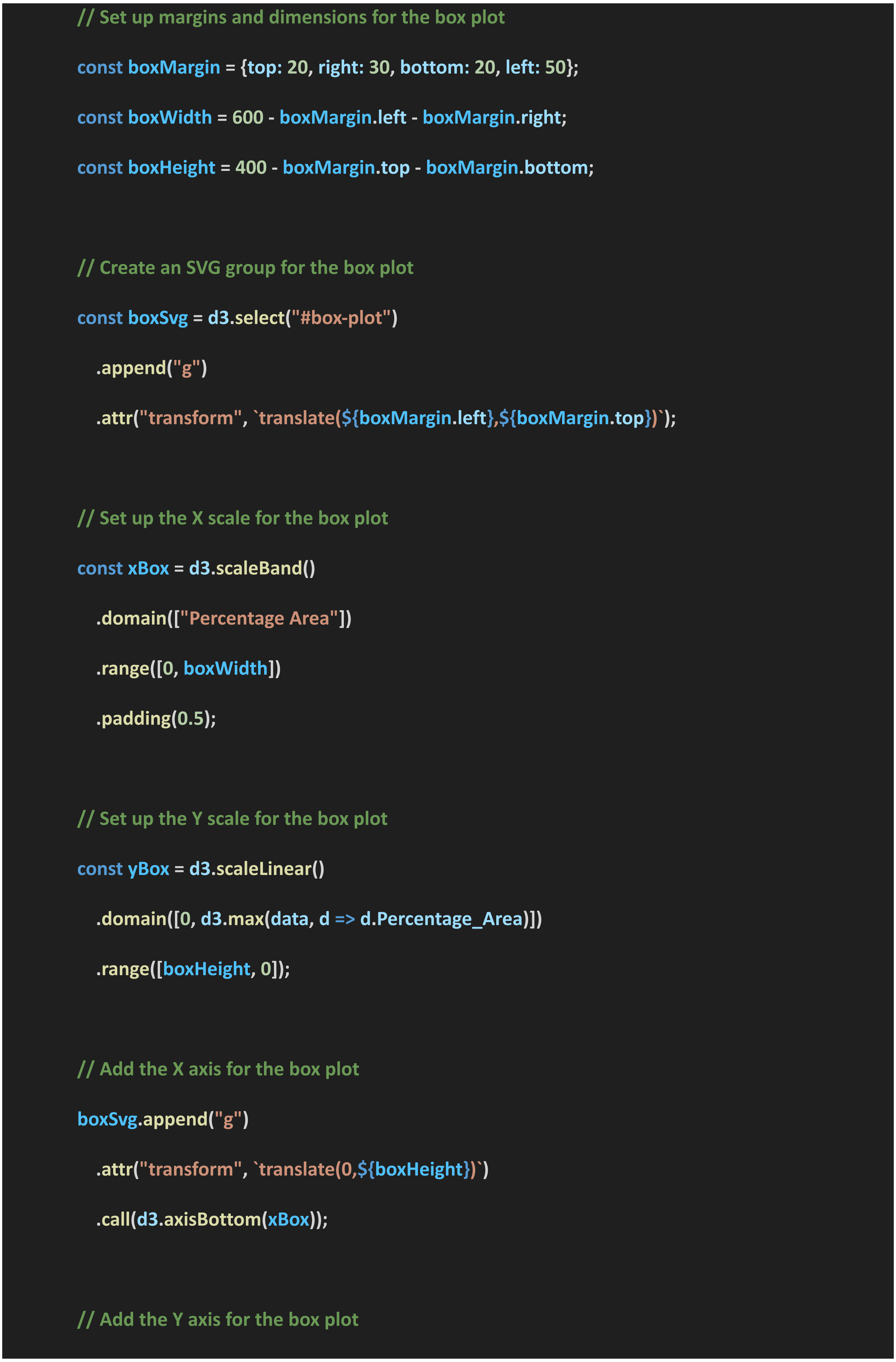
```
.attr("class", "title")

.attr("x", 0)

.attr("y", -155) // Adjusted position

.style("text-anchor", "middle")

.text("Distribution of Forest Types");


        // ---------- Line Chart: Contribution of States to Forest Area ----------


        // Set up margins and dimensions for the line chart

        const lineMargin = {top: 15, right: 30, bottom: 40, left: 60}; // Increased bottom margin

        const lineWidth = 600 - lineMargin.left - lineMargin.right;

        const lineHeight = 400 - lineMargin.top - lineMargin.bottom;


        // Create an SVG group for the line chart

        const lineSvg = d3.select("#line-chart")

          .append("g")

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


        // Set up the X scale for the line chart

        const xLine = d3.scalePoint()

          .domain(data.map(d => d["State/UTs"]))

          .range([0, lineWidth])

          .padding(0.5); // Increased padding for better spacing


        // Set up the Y scale for the line chart

        const yLine = d3.scaleLinear()

          .domain([0, d3.max(data, d => d.Percentage_Area)])
```

```
    .range([lineHeight, 0]);

// Add the X axis for the line chart
const xAxis = lineSvg.append("g")
    .attr("transform", `translate(0,${lineHeight})`)
    .call(d3.axisBottom(xLine));

// Rotate the x-axis labels
xAxis.selectAll("text")
    .attr("transform", "rotate(-45)") // Rotate labels by -45 degrees
    .style("text-anchor", "end"); // Adjust text anchor for better alignment

// Add the Y axis for the line chart
lineSvg.append("g")
    .call(d3.axisLeft(yLine));

// Create the line
const line = d3.line()
    .x(d => xLine(d["State/UTs"]))
    .y(d => yLine(d.Percentage_Area));

// Add the line to the line chart
lineSvg.append("path")
    .datum(data)
    .attr("class", "line")
    .attr("d", line);
```

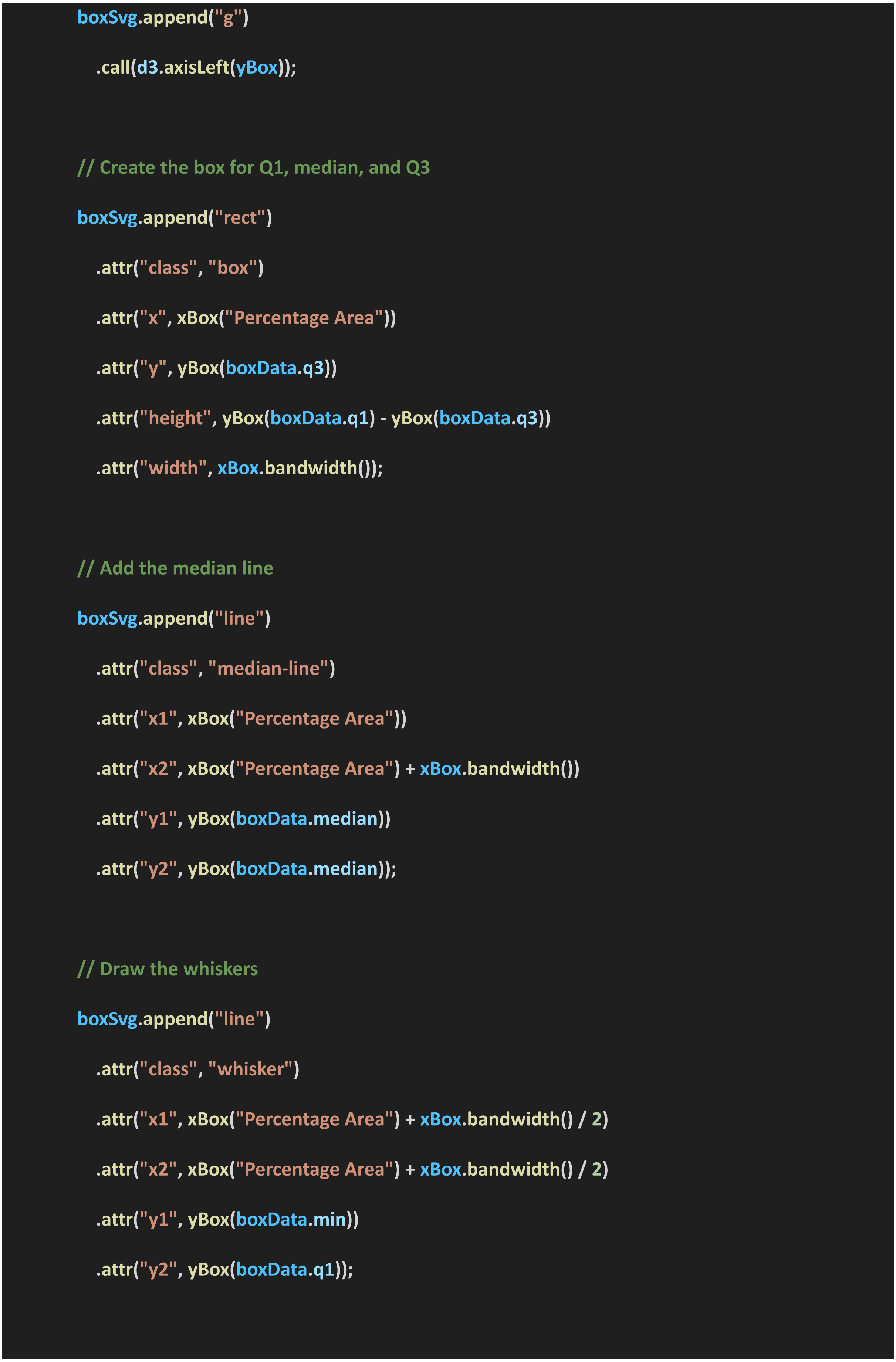


```
      // Add line points
      lineSvg.selectAll(".line-point")
        .data(data)
        .enter()
        .append("circle")
        .attr("class", "line-point")
        .attr("cx", d => xLine(d["State/UTs"]))
        .attr("cy", d => yLine(d.Percentage_Area))
        .attr("r", 4);


        lineSvg.append("text")
.attr("class", "title")
.attr("x", lineWidth / 2)
.attr("y", -1) // Adjusted position
.style("text-anchor", "middle")
.text("Percentage Wise Contribution of States to the Forest Area");


      // Add X axis label for the bar chart
      lineSvg.append("text")
        .attr("class", "axis-label")
        .attr("x", barWidth / 2)
        .attr("y", barHeight + 60)
        .text("State/UT");


      // Add Y axis label for the bar chart
      lineSvg.append("text")
        .attr("class", "axis-label")
```

```
    .attr("transform", "rotate(-90)")

    .attr("y", -45)

    .attr("x", -barHeight /2)

    .text("Forest Percentage");




// ---------- Box Plot: Percentage of Geographical Area ----------


// Calculate quartiles and outliers

const q1 = d3.quantile(data.map(d => d.Percentage_Area).sort(d3.ascending), 0.25);

const median = d3.median(data.map(d => d.Percentage_Area));

const q3 = d3.quantile(data.map(d => d.Percentage_Area).sort(d3.ascending), 0.75);

const iqr = q3 - q1;

const lowerBound = q1 - 1.5 * iqr;

const upperBound = q3 + 1.5 * iqr;


const boxData = {

    min: d3.min(data, d => d.Percentage_Area),

    q1: q1,

    median: median,

    q3: q3,

    max: d3.max(data, d => d.Percentage_Area),

    lowerBound: lowerBound,

    upperBound: upperBound,

    outliers: data.filter(d => d.Percentage_Area < lowerBound || d.Percentage_Area >
upperBound)

};
```

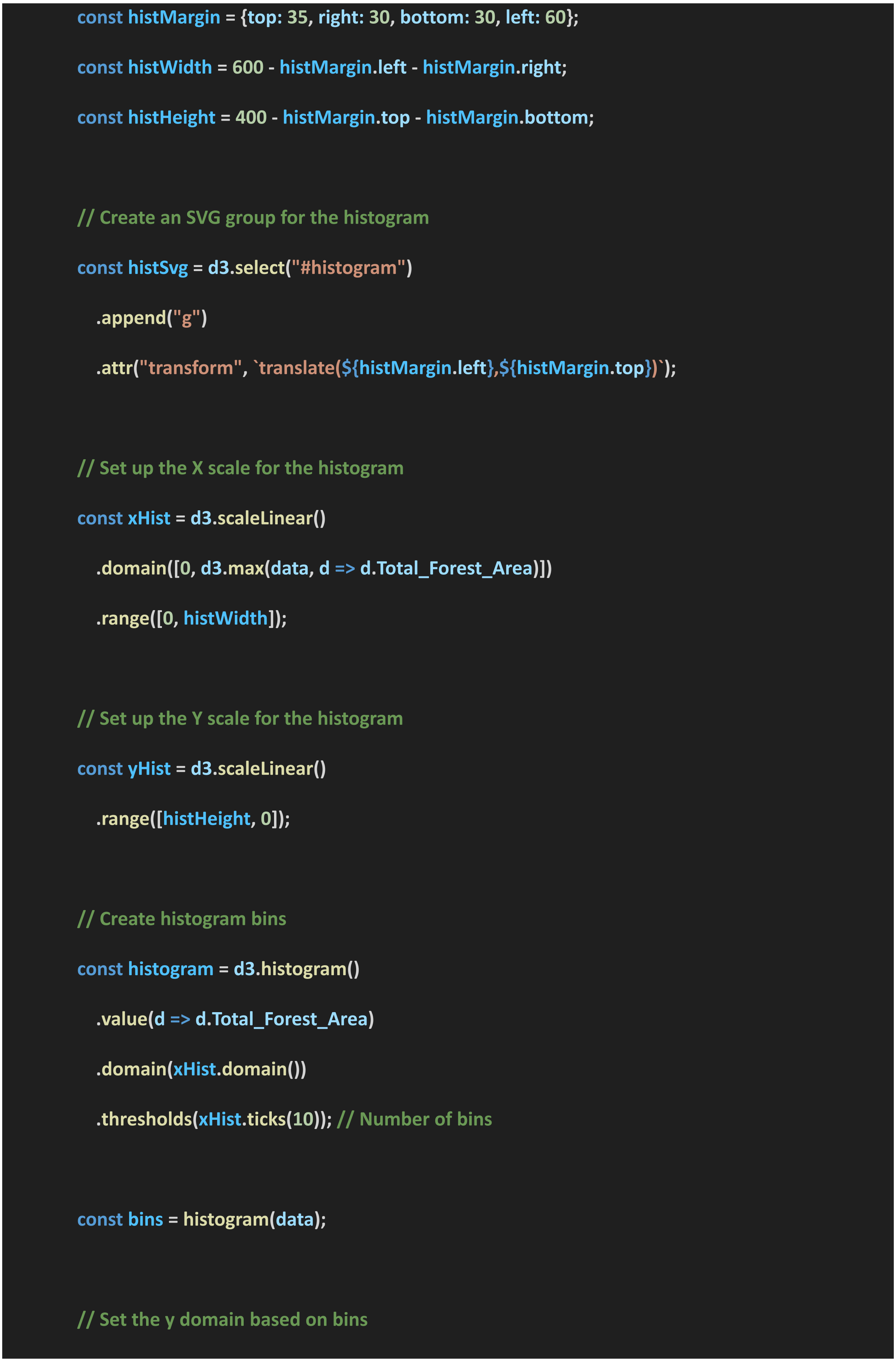


```
// Set up margins and dimensions for the box plot
const boxMargin = {top: 20, right: 30, bottom: 20, left: 50};
const boxWidth = 600 - boxMargin.left - boxMargin.right;
const boxHeight = 400 - boxMargin.top - boxMargin.bottom;

// Create an SVG group for the box plot
const boxSvg = d3.select("#box-plot")
  .append("g")
  .attr("transform", `translate(${boxMargin.left},${boxMargin.top})`);

// Set up the X scale for the box plot
const xBox = d3.scaleBand()
  .domain(["Percentage Area"])
  .range([0, boxWidth])
  .padding(0.5);

// Set up the Y scale for the box plot
const yBox = d3.scaleLinear()
  .domain([0, d3.max(data, d => d.Percentage_Area)])
  .range([boxHeight, 0]);

// Add the X axis for the box plot
boxSvg.append("g")
  .attr("transform", `translate(0,${boxHeight})`)
  .call(d3.axisBottom(xBox));

// Add the Y axis for the box plot
```



```javascript
boxSvg.append("g")
    .call(d3.axisLeft(yBox));

// Create the box for Q1, median, and Q3
boxSvg.append("rect")
    .attr("class", "box")
    .attr("x", xBox("Percentage Area"))
    .attr("y", yBox(boxData.q3))
    .attr("height", yBox(boxData.q1) - yBox(boxData.q3))
    .attr("width", xBox.bandwidth());

// Add the median line
boxSvg.append("line")
    .attr("class", "median-line")
    .attr("x1", xBox("Percentage Area"))
    .attr("x2", xBox("Percentage Area") + xBox.bandwidth())
    .attr("y1", yBox(boxData.median))
    .attr("y2", yBox(boxData.median));

// Draw the whiskers
boxSvg.append("line")
    .attr("class", "whisker")
    .attr("x1", xBox("Percentage Area") + xBox.bandwidth() / 2)
    .attr("x2", xBox("Percentage Area") + xBox.bandwidth() / 2)
    .attr("y1", yBox(boxData.min))
    .attr("y2", yBox(boxData.q1));
```

```
        boxSvg.append("line")

          .attr("class", "whisker")

          .attr("x1", xBox("Percentage Area") + xBox.bandwidth() / 2)

          .attr("x2", xBox("Percentage Area") + xBox.bandwidth() / 2)

          .attr("y1", yBox(boxData.q3))

          .attr("y2", yBox(boxData.max));


        // Plot outliers

        boxSvg.selectAll(".outlier")

          .data(boxData.outliers)

          .enter()

          .append("circle")

          .attr("class", "outlier")

          .attr("cx", xBox("Percentage Area") + xBox.bandwidth() / 2)

          .attr("cy", d => yBox(d.Percentage_Area))

          .attr("r", 4); // Radius of outlier points


          boxSvg.append("text")

.attr("class", "title")

.attr("x", boxWidth / 2)

.attr("y", -8) // Adjusted position to -10

.style("text-anchor", "middle")

.text("Box Plot of Percentage of Geographical Area");


        // ---------- Histogram: Distribution of Forest Areas -----------


        // Set up margins and dimensions for the histogram
```



```
const histMargin = {top: 35, right: 30, bottom: 30, left: 60};
const histWidth = 600 - histMargin.left - histMargin.right;
const histHeight = 400 - histMargin.top - histMargin.bottom;

// Create an SVG group for the histogram
const histSvg = d3.select("#histogram")
    .append("g")
    .attr("transform", `translate(${histMargin.left},${histMargin.top})`);

// Set up the X scale for the histogram
const xHist = d3.scaleLinear()
    .domain([0, d3.max(data, d => d.Total_Forest_Area)])
    .range([0, histWidth]);

// Set up the Y scale for the histogram
const yHist = d3.scaleLinear()
    .range([histHeight, 0]);

// Create histogram bins
const histogram = d3.histogram()
    .value(d => d.Total_Forest_Area)
    .domain(xHist.domain())
    .thresholds(xHist.ticks(10)); // Number of bins

const bins = histogram(data);

// Set the y domain based on bins
```

```
yHist.domain([0, d3.max(bins, d => d.length)]);


// Add the X axis for the histogram

histSvg.append("g")

  .attr("transform", `translate(0,${histHeight})`)

  .call(d3.axisBottom(xHist));


// Add the Y axis for the histogram

histSvg.append("g")

  .call(d3.axisLeft(yHist));


// Create the bars for the histogram

histSvg.selectAll(".bar")

  .data(bins)

  .enter()

  .append("rect")

  .attr("class", "bar")

  .attr("x", 1)

  .attr("transform", d => `translate(${xHist(d.x0)},${yHist(d.length)})`)

  .attr("width", d => xHist(d.x1) - xHist(d.x0) - 1) // Width based on bin range

  .attr("height", d => histHeight - yHist(d.length));


  histSvg.append("text")

  .attr("class", "title")

  .attr("x", barWidth / 2)

  .attr("y", -20)

  .text("Distribution of Forest Areas");
```

```
        // Add X axis label for the bar chart

        histSvg.append("text")

          .attr("class", "axis-label")

          .attr("x", barWidth / 2)

          .attr("y", barHeight + 60)

          .text("Forest Area");


        // Add Y axis label for the bar chart

        histSvg.append("text")

          .attr("class", "axis-label")

          .attr("transform", "rotate(-90)")

          .attr("y", -45)

          .attr("x", -barHeight /2)

          .text("Number of States/UT");


      });

  </script>

</body>

</html>
```

Conclusion:

Through this experiment, we gained valuable insights into D3.js and its powerful capabilities for data visualization. We explored how to effectively plot various types of graphs, including bar charts, line plots, histograms, and more. We learned how the various plots can be arranged into a dashboard in D3.js.