| | |
|---|---|
| Name: | `Dhruvil Patel` |
| Class: | B.E. |
| Branch: | AIML |
| UID: | 2021600051 |
| Batch: | L |
| Experiment 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.

**Dataset:**

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

**Description:**

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.

**Attributes/Columns:**

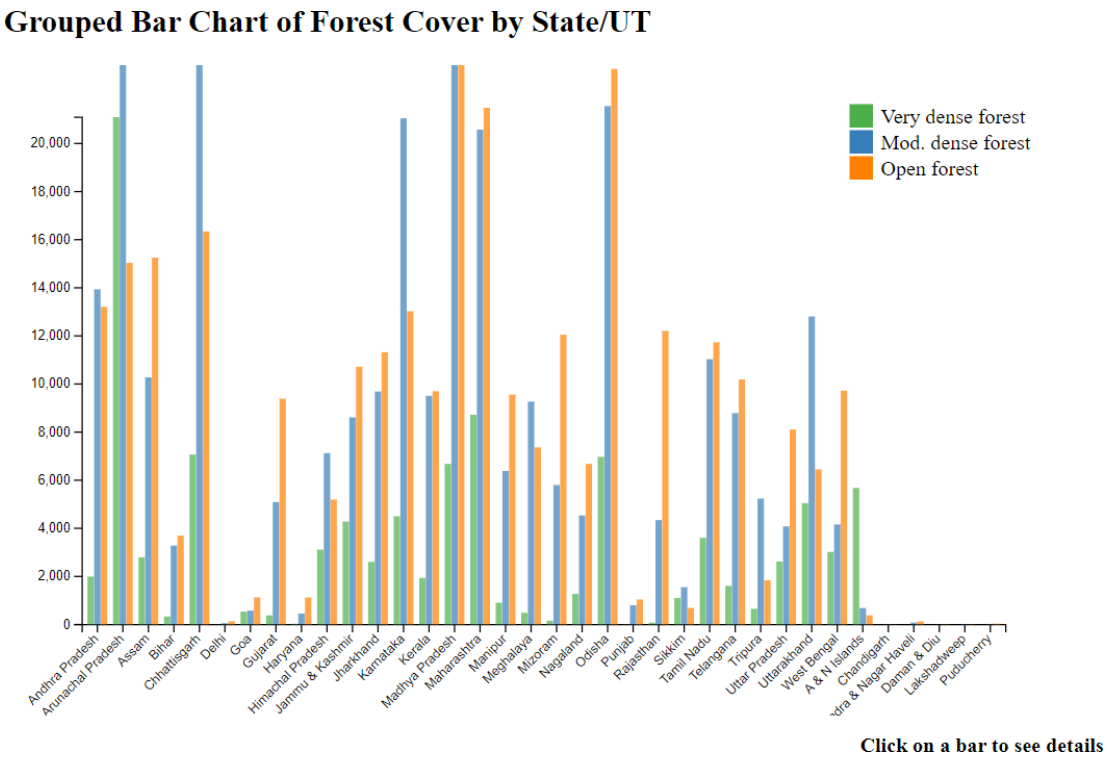
['State/UTs', '1987', '1989', '1991', '1993', '1995', '1997', '1999',
'2001', '2003', '2005', '2007', '2011', '2013']

['State/UTs', 'Geographical area', 'Very dense forest',

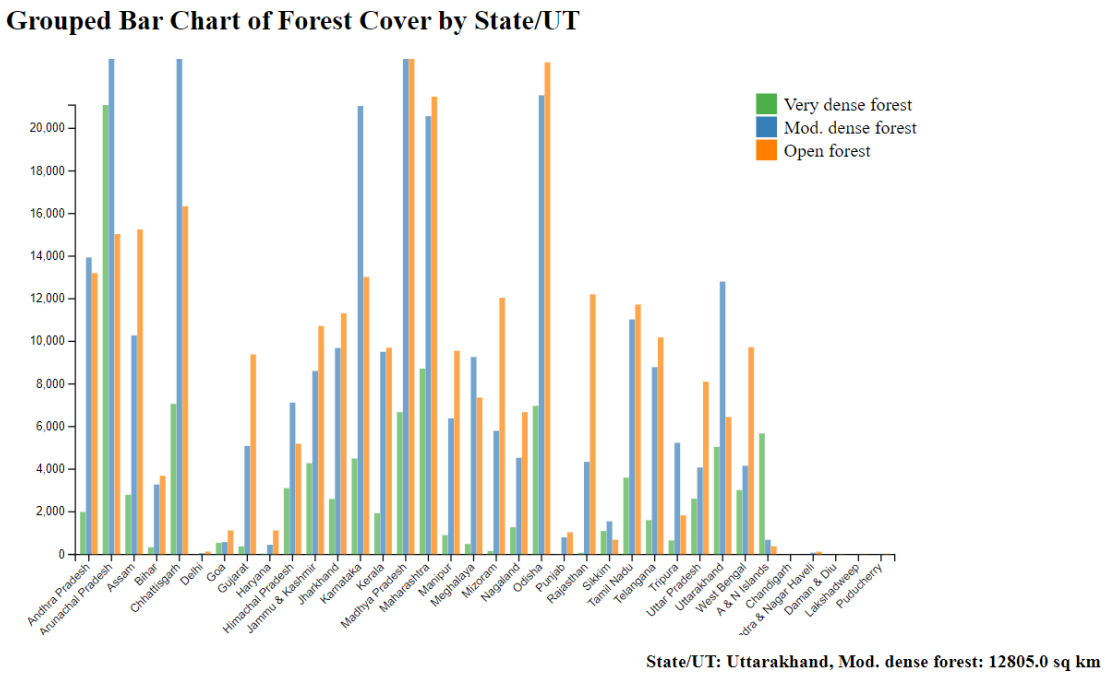
'Mod. dense forest', 'Open forest', 'Total forest',
'Percentage of geographical area', 'Scrub']

**Output / Plots:**

1. **Grouped Bar Chart**



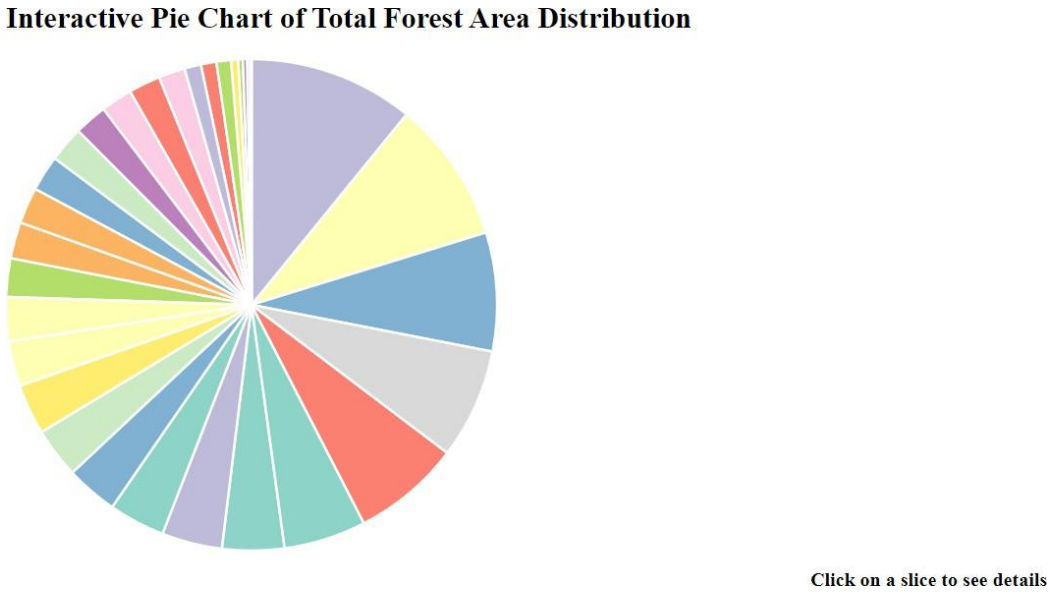

This grouped bar chart shows distribution of states based on Very dense Mod. dense and open forest cover areas.



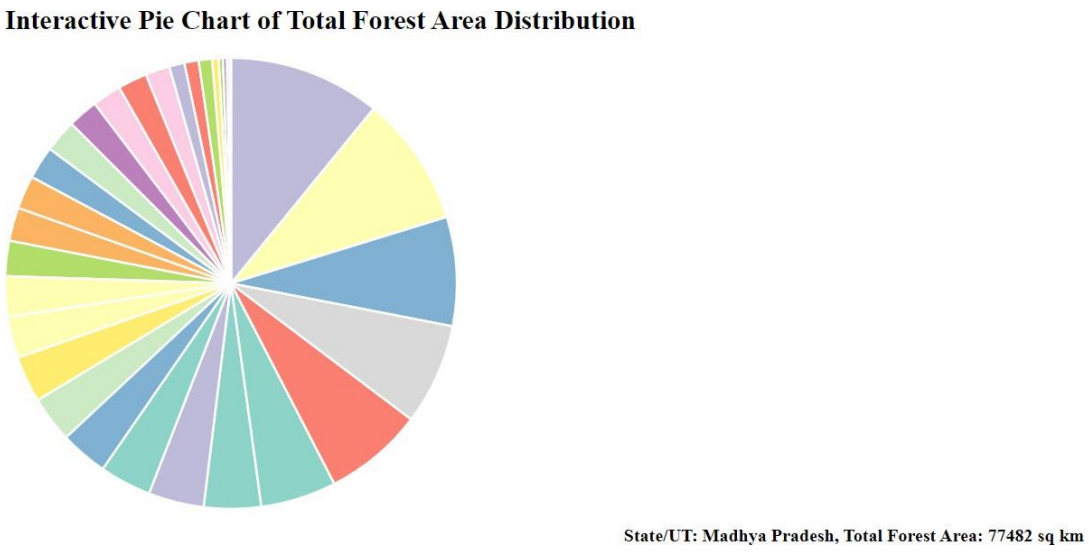

After clicking on a particular bar i can see the details of that bar below.

This chart helps in understanding various types of forest covers by comparing them in all states. Madhya Pradesh has the highest cover for Mod. dense and Open Forests whereas Arunachal Pradesh has the highest cover for Very dense forest.

## 2. Pie Chart

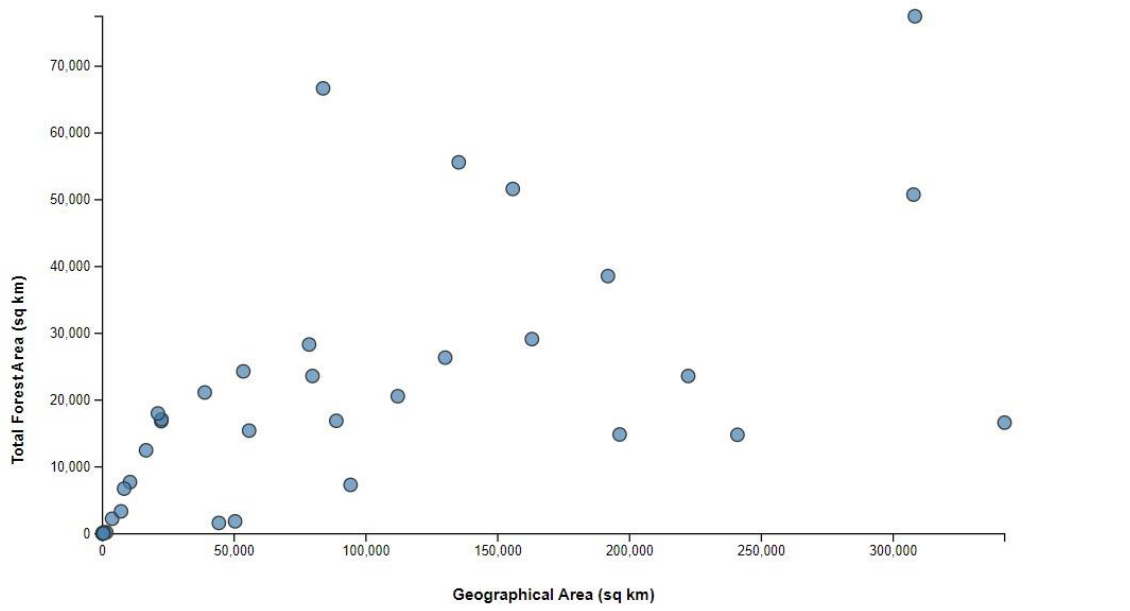


This pie chart shows the distribution of total forest covers of all the states.

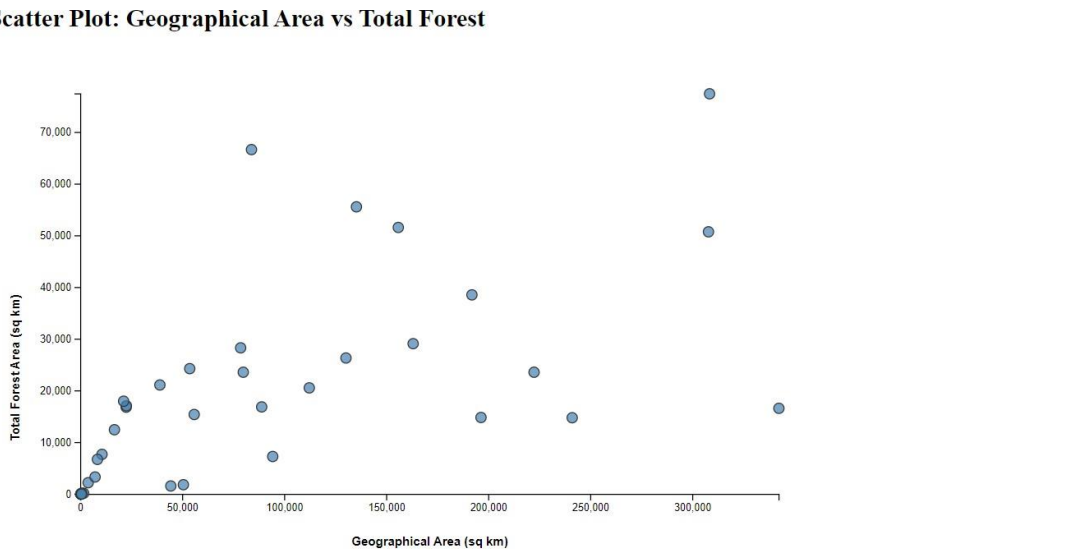



After clicking on a particular slice we can see the details below. Madhya Pradesh has the largest forest cover among all the states.

### 3. Scatter Plot



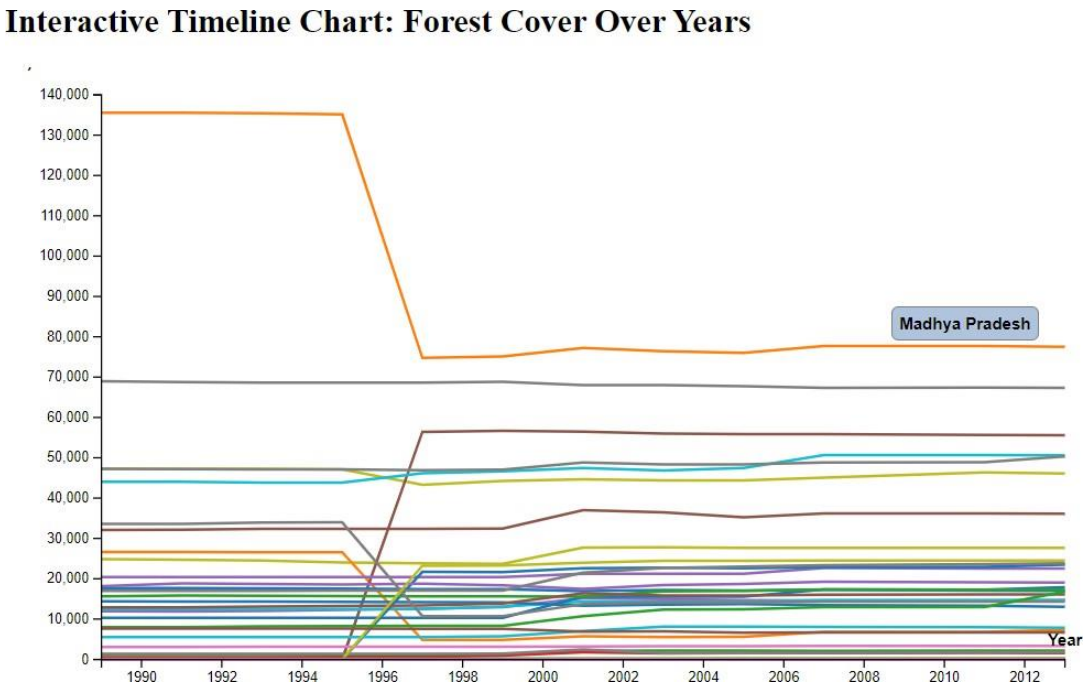

This is a Scatter plot showing the relationship between Geographical Area and Total Forest.




If we hover over a particular point we can see the details of that point below.

We can say that this follows a linear relationship , i.e., if a state has more geographical area it is highly likely that the forest cover is also large.

4. **TimeLine Chart**




This timeline chart shows the forest cover areas over the areas for all states. On hovering over a particular line we can see the state which it belongs to. Forest Cover in Madhya Pradesh, Uttar Pradesh and Bihar seem to have reduced after the year 1994.

**Conclusion:**

D3.js is a powerful library for creating dynamic, interactive data visualizations. By implementing various chart types, we could extract meaningful insights about forest cover, trends, and patterns.

Line wrap

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Grouped Bar Chart and Interactive Pie Chart - Forest Cover</title>
    <script src="https://d3js.org/d3.v7.min.js"></script>
    <style>
        .bar {
            fill-opacity: 0.7;
        }

        .axis-label {
            font-size: 12px;
            font-weight: bold;
        }

        .legend text {
            font-size: 12px;
        }

        .pie-label {
            font-size: 12px;
            text-anchor: middle;
            fill: white;
        }

        .info-box {
            font-size: 16px;
            font-weight: bold;
            text-align: center;
            margin-top: 10px;
        }

        .scatter-dot {
            fill: steelblue;
            stroke: #333;
            stroke-width: 1px;
            fill-opacity: 0.7;
        }

        .line {
            fill: none;
            stroke-width: 2;
        }

        .tooltip {
            position: absolute;
            text-align: center;
            padding: 5px;
            font: 12px sans-serif;
            background: lightsteelblue;
            border: 1px solid gray;
            border-radius: 5px;
            pointer-events: none;
            opacity: 0;
        }
    </style>
</head>

<body>
    <h2>Grouped Bar Chart of Forest Cover by State/UT</h2>
    <div id="bar-chart"></div>
    <div class="info-box" id="info-box">Click on a bar to see details</div>

    <h2>Interactive Pie Chart of Total Forest Area Distribution</h2>
    <div id="pie-chart"></div>
    <div class="info-box" id="info-box2">Click on a slice to see details</div>

    <h2>Scatter Plot: Geographical Area vs Total Forest</h2>
    <div id="scatter-plot"></div>
    <div class="info-box" id="scatter-info">Hover over a point to see details</div>

    <h2>Interactive Timeline Chart: Forest Cover Over Years</h2>
    <div id="timeline-chart"></div>
    <div class="tooltip" id="tooltip"></div>

    <script>
        const margin = { top: 40, right: 30, bottom: 70, left: 60 };
        const width = 800 - margin.left - margin.right;
        const height = 500 - margin.top - margin.bottom;

        const pieWidth = 400;
        const pieHeight = 400;
        const radius = Math.min(pieWidth, pieHeight) / 2;

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

        const svgPie = d3.select("#pie-chart")
            .append("svg")
```

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

        const infoBox = d3.select("#info-box");
        const infoBox2 = d3.select("#info-box2");

        const scatterMargin = { top: 40, right: 30, bottom: 70, left: 90 };
        const scatterWidth = 800 - scatterMargin.left - scatterMargin.right;
        const scatterHeight = 500 - scatterMargin.top - scatterMargin.bottom;

        const svgScatter = d3.select("#scatter-plot")
            .append("svg")
            .attr("width", scatterWidth + scatterMargin.left + scatterMargin.right)
            .attr("height", scatterHeight + scatterMargin.top + scatterMargin.bottom)
            .append("g")
            .attr("transform", `translate(${scatterMargin.left},${scatterMargin.top})`);

        const scatterInfoBox = d3.select("#scatter-info");

        // Read the data
        d3.csv("Forest.csv").then(function (data) {
            const subgroups = ['Very dense forest', 'Mod. dense forest', 'Open forest'];

            const groups = data.map(d => d['State/UTs']);

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

            svgBar.append("g")
                .attr("transform", `translate(0,${height})`)
                .call(d3.axisBottom(x))
                .selectAll("text")
                .attr("transform", "translate(-10,0)rotate(-45)")
                .style("text-anchor", "end");

            const y = d3.scaleLinear()
                .domain([0, d3.max(data, d => +d['Very dense forest'])])
                .range([height, 0]);

            svgBar.append("g")
                .call(d3.axisLeft(y));

            const xSubgroup = d3.scaleBand()
                .domain(subgroups)
                .range([0, x.bandwidth()])
                .padding([0.05]);

            const color = d3.scaleOrdinal()
                .domain(subgroups)
                .range(['#4daf4a', '#377eb8', '#ff7f00']);

            svgBar.append("g")
                .selectAll("g")
                .data(data)
                .join("g")
                .attr("transform", d => `translate(${x(d['State/UTs'])}, 0)`)
                .selectAll("rect")
                .data(d => subgroups.map(key => ({ key: key, value: d[key], state: d['State/UTs'] }))) // Add state info
                .join("rect")
                .attr("x", d => xSubgroup(d.key))
                .attr("y", d => y(d.value))
                .attr("width", xSubgroup.bandwidth())
                .attr("height", d => height - y(d.value))
                .attr("fill", d => color(d.key))
                .attr("class", "bar")
                .on("click", function (event, d) {
                    infoBox.text(`State/UT: ${d.state}, ${d.key}: ${d.value} sq km`);
                });


            const legendBar = svgBar.append("g")
                .attr("transform", `translate(${width - 120}, -10)`);

            subgroups.forEach((subgroup, i) => {
                legendBar.append("rect")
                    .attr("x", 0)
                    .attr("y", i * 20)
                    .attr("width", 18)
                    .attr("height", 18)
                    .style("fill", color(subgroup));

                legendBar.append("text")
                    .attr("x", 24)
                    .attr("y", i * 20 + 9)
                    .attr("dy", "0.35em")
                    .text(subgroup);
            });

            const totalForestArea = data.map(d => ({
                state: d['State/UTs'],
                value: +d['Total forest']
            }));

            const pie = d3.pie()
```

```
                .value(d => d.value);

            const arc = d3.arc()
                .innerRadius(0)
                .outerRadius(radius);

            const pieColor = d3.scaleOrdinal()
                .domain(totalForestArea.map(d => d.state))
                .range(d3.schemeSet3);

            svgPie.selectAll('path')
                .data(pie(totalForestArea))
                .enter()
                .append('path')
                .attr('d', arc)
                .attr('fill', d => pieColor(d.data.state))
                .attr("stroke", "white")
                .style("stroke-width", "2px")
                .on("click", function (event, d) {
                    <!-- console.log("hei") -->
                    infoBox2.text(`State/UT: ${d.data.state}, Total Forest Area: ${d.data.value} sq km`);
                });


            const xScale = d3.scaleLinear()
                .domain([0, d3.max(data, d => +d['Geographical area'])])
                .range([0, scatterWidth]);

            const yScale = d3.scaleLinear()
                .domain([0, d3.max(data, d => +d['Total forest'])])
                .range([scatterHeight, 0]);

            svgScatter.append("g")
                .attr("transform", `translate(0,${scatterHeight})`)
                .call(d3.axisBottom(xScale))
                .append("text")
                .attr("x", scatterWidth / 2)
                .attr("y", 50)
                .attr("fill", "black")
                .attr("class", "axis-label")
                .text("Geographical Area (sq km)");

            svgScatter.append("g")
                .call(d3.axisLeft(yScale))
                .append("text")
                .attr("x", -scatterHeight / 2)
                .attr("y", -60)
                .attr("fill", "black")
                .attr("transform", "rotate(-90)")
                .attr("class", "axis-label")
                .text("Total Forest Area (sq km)");

            svgScatter.selectAll("circle")
                .data(data)
                .enter()
                .append("circle")
                .attr("cx", d => xScale(+d['Geographical area']))
                .attr("cy", d => yScale(+d['Total forest']))
                .attr("r", 5)
                .attr("class", "scatter-dot")
                .on("mouseover", function (event, d) {
                    scatterInfoBox.text(`State/UT: ${d['State/UTs']}, Geographical Area: ${d['Geographical area']} sq km, Total Forest: ${d['Total for
                })
                .on("mouseout", function () {
                    scatterInfoBox.text("Hover over a point to see details");
                });



        }).catch(error => {
            console.error("Error loading the CSV file: ", error);
        });



        const timeMargin = { top: 20, right: 30, bottom: 70, left: 70 };
        const timeWidth = 800 - timeMargin.left - timeMargin.right;
        const timeHeight = 500 - timeMargin.top - timeMargin.bottom;

        const svgTimeline = d3.select("#timeline-chart")
            .append("svg")
            .attr("width", timeWidth + timeMargin.left + timeMargin.right)
            .attr("height", timeHeight + timeMargin.top + timeMargin.bottom)
            .append("g")
            .attr("transform", `translate(${timeMargin.left},${timeMargin.top})`);

        // Read the data
        d3.csv("Forest2.csv").then(function (data) {
            console.log(data)
            const yearKeys = Object.keys(data[0]).slice(1).map(Number);
            const stateTimelineData = data.map(d => ({
                state: d['State/UTs'],
                values: yearKeys.map(year => ({ year, cover: +d[year] }))
            }));

            const xTimeScale = d3.scaleLinear()
                .domain(d3.extent(yearKeys))
                .range([0, timeWidth]);
```

```
293
294             const yTimeScale = d3.scaleLinear()
295                 .domain([0, d3.max(stateTimelineData, s => d3.max(s.values, v => v.cover))])
296                 .nice()
297                 .range([timeHeight, 0]);
298
299             const lineGenerator = d3.line()
300                 .x(d => xTimeScale(d.year))
301                 .y(d => yTimeScale(d.cover));
302
303             const lines = svgTimeline.selectAll(".line")
304                 .data(stateTimelineData)
305                 .enter()
306                 .append("path")
307                 .attr("class", "line")
308                 .attr("d", d => lineGenerator(d.values))
309                 .attr("stroke", () => d3.schemeCategory10[Math.floor(Math.random() * 10)])
310                 .on("mouseover", function (event, d) {
311                     d3.select("#tooltip")
312                         .style("opacity", 1)
313                         .html(`<strong>${d.state}</strong>`)
314                         .style("left", (event.pageX + 5) + "px")
315                         .style("top", (event.pageY - 28) + "px");
316                 })
317                 .on("mouseout", function () {
318                     d3.select("#tooltip").style("opacity", 0);
319                 });
320
321             svgTimeline.append("g")
322                 .attr("transform", `translate(0,${timeHeight})`)
323                 .call(d3.axisBottom(xTimeScale).tickFormat(d3.format("d")))
324                 .append("text")
325                 .attr("x", timeWidth)
326                 .attr("y", -10)
327                 .attr("fill", "black")
328                 .attr("class", "axis-label")
329                 .text("Year");
330
331             svgTimeline.append("g")
332                 .call(d3.axisLeft(yTimeScale))
333                 .append("text")
334                 .attr("x", -50)
335                 .attr("y", -20)
336                 .attr("fill", "black")
337                 .attr("class", "axis-label")
338                 .text("Forest Cover (sq km)");
339
340
341         }).catch(error => {
342             console.error("Error loading the CSV file: ", error);
343         });
344
345
346     </script>
347
348
349
350 <!-- Code injected by live-server -->
351 <script>
352     // <![CDATA[  <-- For SVG support
353     if ('WebSocket' in window) {
354         (function () {
355             function refreshCSS() {
356                 var sheets = [].slice.call(document.getElementsByTagName("link"));
357                 var head = document.getElementsByTagName("head")[0];
358                 for (var i = 0; i < sheets.length; ++i) {
359                     var elem = sheets[i];
360                     var parent = elem.parentElement || head;
361                     parent.removeChild(elem);
362                     var rel = elem.rel;
363                     if (elem.href && typeof rel != "string" || rel.length == 0 || rel.toLowerCase() == "stylesheet") {
364                         var url = elem.href.replace(/(&|\?)_cacheOverride=\d+/, '');
365                         elem.href = url + (url.indexOf('?') >= 0 ? '&' : '?') + '_cacheOverride=' + (new Date().valueOf());
366                     }
367                     parent.appendChild(elem);
368                 }
369             }
370             var protocol = window.location.protocol === 'http:' ? 'ws://' : 'wss://';
371             var address = protocol + window.location.host + window.location.pathname + '/ws';
372             var socket = new WebSocket(address);
373             socket.onmessage = function (msg) {
374                 if (msg.data == 'reload') window.location.reload();
375                 else if (msg.data == 'refreshcss') refreshCSS();
376             };
377             if (sessionStorage && !sessionStorage.getItem('IsThisFirstTime_Log_From_LiveServer')) {
378                 console.log('Live reload enabled.');
379                 sessionStorage.setItem('IsThisFirstTime_Log_From_LiveServer', true);
380             }
381         })();
382     }
383     else {
384         console.error('Upgrade your browser. This Browser is NOT supported WebSocket for Live-Reloading.');
385     }
386     // ]]>
387 </script>
388 </body>
389
390 </html>
```