Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
(Autonomous Institute Affiliated to University of Mumbai)
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India

Expt 8 : Designing Interactive Dashboards and Storytelling using D3.js on Environment/Forest Cover Dataset

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:

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

Expected Outcomes:

1. Ability to create various types of visualizations using D3.js.
2. Interactive dashboards demonstrating different types of charts. 3. Insights from the Environment/Forest cover dataset through visual storytelling.

Stepwise Procedure:

1. Setup Environment:
   - Install required tools (Node.js, npm, etc.).

Install D3.js library:
bash
Copy code
npm install d3

   -

2. Prepare the Dataset:
   - Use an Environment/Forest cover dataset. Ensure data includes numerical, categorical, and time-based information for comprehensive charting.

Load the dataset into your D3.js project:
javascript

Copy code

```
d3.csv("forest_cover.csv").then(function(data) {
        console.log(data);
});
```

3. Basic Charts:

Bar Chart - HDI by country

```
function createBarChart(data) {
    const svg = d3.select("#barChart").append("svg").attr("width",
500).attr("height", 300);
    addTitle(svg, "HDI by Country");

    const hdiData = data.map(d => ({ country: d.Country, hdi: +d.HDI }));
    const xScale = d3.scaleBand().domain(hdiData.map(d =>
d.country)).range([0, 480]).padding(0.2);
    const yScale = d3.scaleLinear().domain([0, d3.max(hdiData, d =>
d.hdi)]).range([280, 0]);

    svg.append("g").selectAll("rect")
        .data(hdiData).enter().append("rect")
        .attr("x", d => xScale(d.country))
        .attr("y", d => yScale(d.hdi))
        .attr("width", xScale.bandwidth())
        .attr("height", d => 280 - yScale(d.hdi))
        .attr("fill", "teal")
        .on("mouseover", (event, d) => {
            tooltip.style("visibility", "visible").text(`Country:
${d.country}, HDI: ${d.hdi}`);
        })
        .on("mousemove", (event) => {
            tooltip.style("top", (event.pageY - 10) + "px").style("left",
(event.pageX + 10) + "px");
        })
        .on("mouseout", () => tooltip.style("visibility", "hidden"));

    svg.append("g").attr("transform",
"translate(0,280)").call(d3.axisBottom(xScale).tickFormat(d =>
```

```
d.substring(0, 3)));
    svg.append("g").call(d3.axisLeft(yScale));
}
```

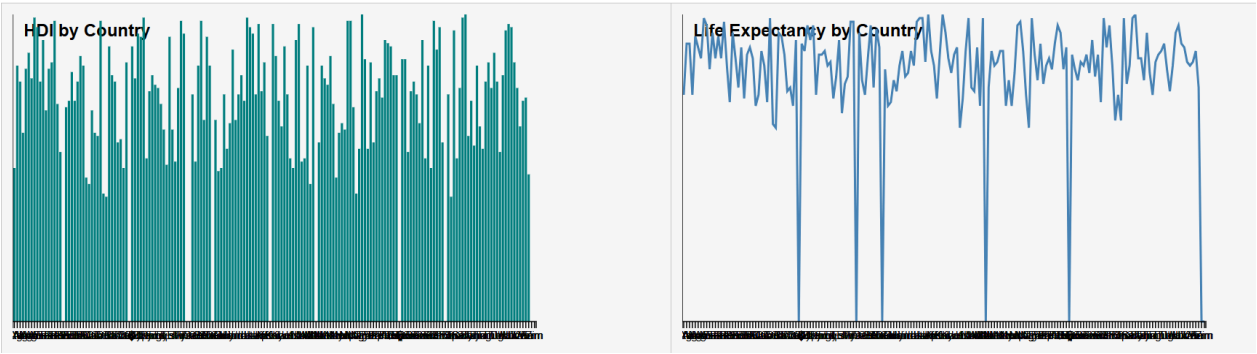
Line Chart- Life expectancy by country

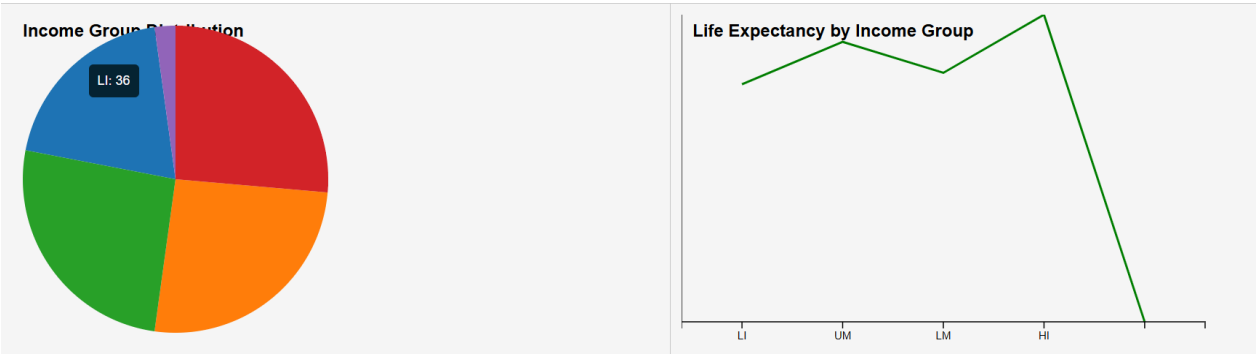
```
function createLineChart(data) {
    const svg = d3.select("#lineChart").append("svg").attr("width",
500).attr("height", 300);
    addTitle(svg, "Life Expectancy by Country");

    const lifeData = data.map(d => ({ country: d.Country, lifeExpectancy:
+d['Life Exectancy'] }));
    const xScale = d3.scaleBand().domain(lifeData.map(d =>
d.country)).range([0, 480]);
    const yScale = d3.scaleLinear().domain([0, d3.max(lifeData, d =>
d.lifeExpectancy)]).range([280, 0]);

    const line = d3.line()
        .x(d => xScale(d.country) + xScale.bandwidth() / 2)
        .y(d => yScale(d.lifeExpectancy));

    svg.append("path")
        .datum(lifeData)
        .attr("fill", "none")
        .attr("stroke", "steelblue")
        .attr("stroke-width", 2)
        .attr("d", line);

    svg.append("g").attr("transform",
"translate(0,280)").call(d3.axisBottom(xScale).tickFormat(d =>
d.substring(0, 3)));
    svg.append("g").call(d3.axisLeft(yScale));
}
```

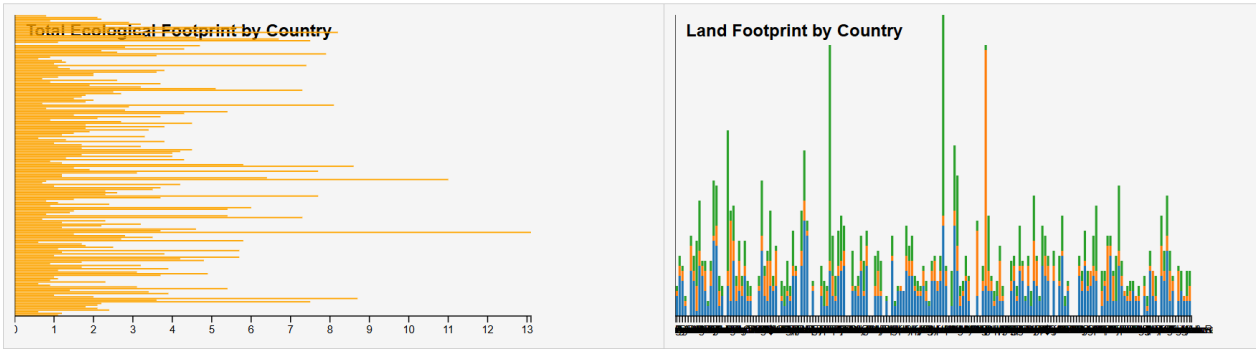
Pie Chart- Income group distribution

```
function createPieChart(data) {
    const svg = d3.select("#pieChart").append("svg").attr("width",
300).attr("height", 300);
    addTitle(svg, "Income Group Distribution");

    const incomeGroups = d3.rollups(data, v => v.length, d => d['Income
Group']);
    const pie = d3.pie().value(d => d[1]);
    const arc = d3.arc().innerRadius(0).outerRadius(140);

    svg.append("g").attr("transform", "translate(150,150)")
        .selectAll("path")
        .data(pie(incomeGroups)).enter().append("path")
        .attr("d", arc)
        .attr("fill", (d, i) => d3.schemeCategory10[i])
        .on("mouseover", (event, d) => {
            tooltip.style("visibility", "visible").text(`${d.data[0]}:
${d.data[1]}`);
        })
        .on("mousemove", (event) => {
            tooltip.style("top", (event.pageY - 10) + "px").style("left",
(event.pageX + 10) + "px");
        })
        .on("mouseout", () => tooltip.style("visibility", "hidden"));
}
```

Dashboard:







Interactive Data Visualization: The dashboard uses D3.js to create interactive visualizations that display ecological and economic indicators for various countries, helping users understand global sustainability metrics.

Charts and Graphs:

- **Bar Chart**: Shows HDI across countries.

- **Line Chart**: Visualizes trends in life expectancy.
- **Pie Chart**: Breaks down income groups by population.
- **Scatter Plot**: Maps SDG index against GDP per capita.
- **Map**: Plots carbon footprint by country.
- **Stacked Bar Chart**: Depicts land footprint types for each country.

Tooltips and Titles: Tooltips on each chart reveal detailed information when hovering over data points, and each chart includes a title for context.

Layout and Responsiveness: The CSS grid layout ensures each chart fits neatly, making the dashboard organized and easily navigable.

Purpose: The dashboard enables comparisons of countries' ecological footprints and economic indicators, providing insights into sustainability and resource use across regions.

Frequently Asked Questions (FAQ):

1. Q: How do I import a dataset into D3.js?
    - A: Use the d3.csv() or d3.json() function to load data.
2. Q: What is the best chart to compare forest cover across regions? ○ A: A bar chart is ideal for comparing categorical data like regions. 3. Q: How do I make the charts interactive?
    - A: You can add event listeners (e.g., onmouseover, onclick) for interactivity.
4. Q: Can I apply multiple types of charts on the same dataset?
    - A: Yes, different charts can highlight various aspects of the dataset.

Summary:

In this lab, we created an interactive dashboard using D3.js for data visualization on an Environment/Forest cover dataset. We explored both basic and advanced chart types and built a storytelling dashboard to provide insights into forest cover trends and distributions.

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.

Future Work:

1. Implement real-time data updates on the dashboard.
2. Add machine learning models to predict future forest cover and integrate them into visualizations.
3. Explore more complex D3.js visualizations like force-directed graphs and geographical maps.