**CIS 568-7101 : Data Visualization**

# Major Assignment 2 - Multi line using D3JS

Piyush Patil

02184301

## GitHub Repository:

Repository Link: <https://github.com/Patil1707/DV>

## Code Implementation:

Below is the HTML and D3.js code.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Supply Chain Analysis</title>

<script src="https://d3js.org/d3.v6.min.js"></script>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

padding: 20px;

background-color: #f9f9f9;

}

h2 {

text-align: center;

margin-bottom: 20px;

}

.container {

display: flex;

flex-direction: row;

justify-content: space-between;

align-items: flex-start;

gap: 5px;

}

.table-container, .chart-container {

flex: 1;

background: white;

padding: 10px;

border-radius: 6px;

box-shadow: 0px 4px 8px rgba(0,0,0,0.1);

min-width: 35%;

}

table {

width: 100%;

border-collapse: collapse;

font-size: 14px;

margin-top: 10px;

}

th, td {

border: 1px solid #ddd;

padding: 8px;

text-align: center;

}

th {

background-color: #007BFF;

color: white;

font-weight: bold;

}

tr:nth-child(even) {

background-color: #f2f2f2;

}

svg {

width: 100%;

height: 450px;

}

.legend text {

font-size: 12px;

}

</style>

</head>

<body>

<h2>Supply Chain Analysis</h2>

<div class="container">

<div class="table-container">

<h3>Profit Table</h3>

<table id="profitTable">

<thead>

<tr >

<th>Date</th>

<th>Estimated Cost</th>

<th>Raw Material</th>

<th>Workmanship</th>

<th>Storage</th>

<th>Actual Cost</th>

<th>Sold Price</th>

<th>Margin of Profit</th>

</tr>

</thead>

<tbody></tbody>

</table>

</div>

<div class="chart-container">

<h3>Multi-line Chart</h3>

<svg id="chart" viewBox="0 0 600 500"></svg>

</div>

</div>

<script>

d3.csv("data\_sample.csv").then(function(data) {

const parseDate = d3.timeParse("%m/%d/%y");

const formatDate = d3.timeFormat("%b %d %Y");

data.forEach(d => {

d.date = parseDate(d.date);

d.EstimatedCost = +d.EstimatedCost;

d.RawMaterial = +d.RawMaterial;

d.Workmanship = +d.Workmanship;

d.StorageCost = +d.StorageCost;

d.ActualCost = d.RawMaterial + d.Workmanship + d.StorageCost;

d.SoldPrice = d.EstimatedCost \* 1.1;

d.MarginOfProfit = d.SoldPrice - d.ActualCost;

});

const tbody = d3.select("#profitTable tbody");

const rows = tbody.selectAll("tr")

.data(data)

.enter()

.append("tr");

rows.append("td").text(d => formatDate(d.date));

rows.append("td").text(d => `$${d.EstimatedCost.toFixed(2)}`);

rows.append("td").text(d => `$${d.RawMaterial.toFixed(2)}`);

rows.append("td").text(d => `$${d.Workmanship.toFixed(2)}`);

rows.append("td").text(d => `$${d.StorageCost.toFixed(2)}`);

rows.append("td").text(d => `$${d.ActualCost.toFixed(2)}`);

rows.append("td").text(d => `$${d.SoldPrice.toFixed(2)}`);

rows.append("td").text(d => `$${d.MarginOfProfit.toFixed(2)}`);

const margin = { top: 50, right: 50, bottom: 50, left: 20 },

width = 700 - margin.left - margin.right,

height = 450 - margin.top - margin.bottom;

const svg = d3.select("#chart")

.append("g")

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

const xScale = d3.scaleTime()

.domain(d3.extent(data, d => d.date))

.range([0, width]);

const yScale = d3.scaleLinear()

.domain([-500, 3500] )

.range([height, 0]);

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

const lineGenerator = metric => d3.line()

.x(d => xScale(d.date))

.y(d => yScale(d[metric]));

const metrics = [

{ key: "EstimatedCost", label: "Estimated Cost" },

{ key: "ActualCost", label: "Actual Cost" },

{ key: "SoldPrice", label: "Sold Price" },

{ key: "MarginOfProfit", label: "Margin of Profit" }

];

metrics.forEach((metric, i) => {

svg.append("path")

.datum(data)

.attr("class", "line")

.attr("d", lineGenerator(metric.key))

.attr("stroke", color(i))

.attr("fill", "none")

.attr("stroke-width", 2)

.style("opacity", 0)

.transition()

.duration(1000)

.style("opacity", 1);

});

svg.append("g")

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

.call(d3.axisBottom(xScale).ticks(6));

svg.append("g")

.call(d3.axisLeft(yScale));

const legend = svg.selectAll(".legend")

.data(metrics)

.enter().append("g")

.attr("class", "legend")

.attr("transform", (d, i) => `translate(${width - 100}, ${i \* 20})`);

legend.append("rect")

.attr("width", 12)

.attr("height", 12)

.attr("fill", (d, i) => color(i));

legend.append("text")

.attr("x", 20)

.attr("y", 10)

.text(d => d.label)

.attr("font-size", "12px");

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

</script>

</body>

</html>

## Output Screenshots:

A screenshot of a computer

Description automatically generated