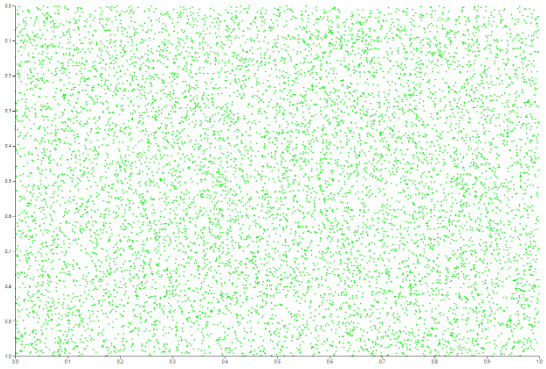
Assignment 1 - Data transformation

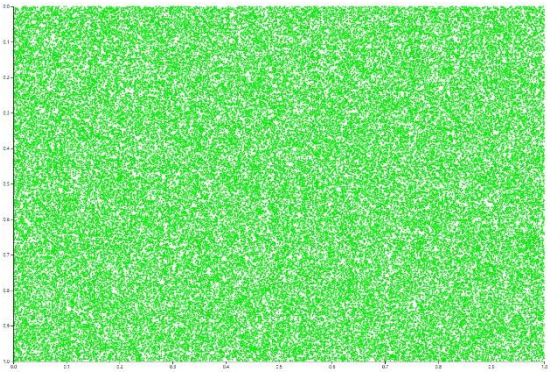
Weilin Ruan 50018083

Question 1: What problems do you encounter when the number of points increases? (30 points)

When increasing the number of points from the initial 10,000 to 1 million, 10 million, and even 100 million, the following issues arise:

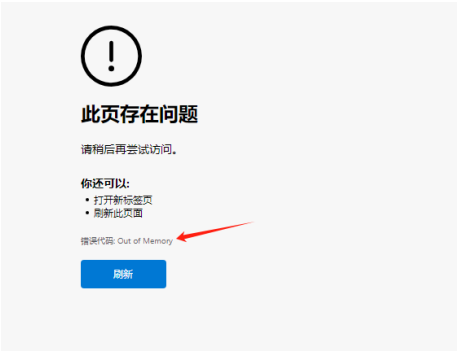


(a) scatterplot for 10,000 random points



(b) scatterplot for 100,000 random points

(c) scatterplot for 1,000,000 random points



(d) error happened when plotted 10,000,000 random points

1. **Performance Degradation:**

- At 1 million points, the browser struggles with responsiveness. The scatterplot takes longer to render, and the browser may freeze temporarily.
- At 10 million points, the lag becomes more substantial, and the browser may become unresponsive for extended periods before eventually rendering the plot.
- At 100 million points, the browser often crashes. This is due to the high computational load required for rendering and managing such many graphical elements.

2. **Memory Consumption:**

- For datasets in the range of tens or hundreds of millions of points, the browser's memory usage spikes significantly, often exceeding the available system memory, which can lead to crashes or slowdowns.

3. **Overplotting:**

- As the number of points increases, overplotting becomes a significant issue. The data points overlap to the extent that individual points are no longer distinguishable, making it difficult to extract meaningful insights from the visualization.

4. **Decreased Interactivity:**

- Interactive features like zooming, panning, and tooltips become sluggish or entirely unresponsive as the number of points increases. This severely limits the utility of the scatterplot for data exploration.

Question 2: Which data transformation method would you choose to address the problems? (30 points)

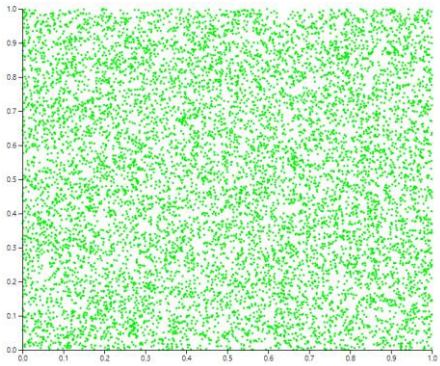
To mitigate the performance and visualization challenges, two effective data transformation methods can be applied:

1. **Aggregation:**

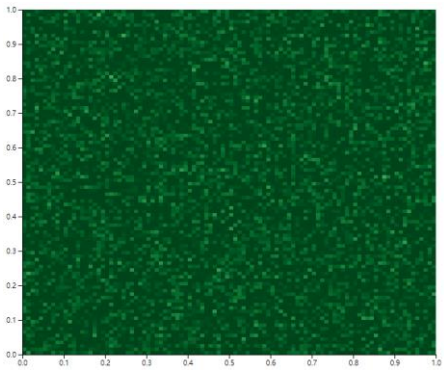
Aggregation involves grouping data points into bins or regions and representing them with a summary value (e.g., a heatmap that shows the density of points within a region). This drastically reduces the number of elements that need to be rendered.

2. **Sampling:**

Sampling involves selecting a subset of the data for visualization, which reduces the number of points while still preserving the overall trends. Random sampling is an effective method to maintain a representative view of the data without overwhelming the browser.



(e) sampled data from 1 million points



(f) aggregation from 1 million points

Question 3: What are the benefits and drawbacks of each transformation method? (40 points)

1. Aggregation

- **Benefits:**
    - Improved Performance: By reducing the number of graphical elements to render, aggregation reduces the computational burden on the browser, allowing for a smoother visualization experience.
    - Density Representation: Aggregation can give insights into the distribution or density of points in different regions, which is useful for identifying clusters or patterns in the data that might not be apparent with individual points.
    - Overplotting Reduction: Aggregating data into bins or areas reduces overplotting, making the visualization more readable.
- **Drawbacks:**
    - Loss of Detail: Aggregation may obscure finer details or outliers in the data since individual points are no longer visualized, leading to a potential loss of critical information.
    - Choice of Binning Method: The effectiveness of aggregation depends on how the data is grouped. An inappropriate bin size or method could result in misleading patterns.

2. Sampling

- **Benefits:**
    - Performance: Sampling reduces the dataset size, making it possible to render the scatterplot quickly without overwhelming the browser.
    - Simplicity: Sampling is easier to implement compared to more complex aggregation methods and can still provide a representative overview of the data.
    - Outliers and Trends: Depending on the sampling method, important trends or outliers in the data can still be retained.
- **Drawbacks:**
    - Potential Bias: Random sampling might not always capture all important patterns or outliers, leading to a biased or incomplete view of the data.

- Loss of Precision: Since only a subset of the data is visualized, some finer details or less frequent patterns may be missed.

Question 4: Implement one of the data transformation methods and submit the code. (20 bonus points)

I have implemented data sampling as the chosen transformation method since it allows for a quicker and more straightforward reduction in data size while maintaining a representative visualization of trends. Below is the modified index.html file incorporating random sampling.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
  width, initial-scale=1.0">
    <title>Scatterplot with Sampling</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        #scatterplot {
            width: 100%;
            height: 90vh;
        }
    </style>
</head>
<body>
    <h1>Scatterplot with Sampling</h1>
    <div id="scatterplot"></div>

    <script src="https://d3js.org/d3.v7.min.js"></script>
    <script>
        // Function to generate random data points
        function generateRandomData(numPoints) {
            let data = [];
            for (let i = 0; i < numPoints; i++) {
                data.push({
                    x: Math.random(),
                    y: Math.random()
                });
            }
```

```
32.             return data;
33.         }
34.
35.         // Function to sample data
36.         function sampleData(data, sampleSize) {
37.             const sampledData = [];
38.             const totalData = data.length;
39.             const step = Math.floor(totalData / sampleSize)
    ;
40.             for (let i = 0; i < totalData; i += step) {
41.                 sampledData.push(data[i]);
42.             }
43.             return sampledData;
44.         }
45.
46.         // Generate 100 million points
47.         const totalPoints = 100000000;
48.         const data = generateRandomData(totalPoints);
49.
50.         // Sample data to reduce the number of points
51.         const sampleSize = 1000000;  // 1 million points
52.         const sampledData = sampleData(data, sampleSize);
53.
54.         // Set up the SVG and scatterplot dimensions
55.         const width = window.innerWidth;
56.         const height = window.innerHeight * 0.9;
57.         const margin = { top: 20, right: 30, bottom: 30, le
    ft: 40 };
58.
59.         const svg = d3.select("#scatterplot")
60.             .append("svg")
61.             .attr("width", width)
62.             .attr("height", height);
63.
64.         const x = d3.scaleLinear()
65.             .domain([0, 1])
66.             .range([margin.left, width - margin.right]);
67.
68.         const y = d3.scaleLinear()
69.             .domain([0, 1])
70.             .range([height - margin.bottom, margin.top]);
71.
72.         svg.append("g")
```

```
            .attr("transform", `translate(0, ${height - margin.bottom})`)
            .call(d3.axisBottom(x));

        svg.append("g")
            .attr("transform", `translate(${margin.left}, 0)`)
            .call(d3.axisLeft(y));

        // Plot the sampled data points
        svg.append("g")
            .selectAll("circle")
            .data(sampledData)
            .enter()
            .append("circle")
            .attr("cx", d => x(d.x))
            .attr("cy", d => y(d.y))
            .attr("r", 1)
            .style("fill", "steelblue");
    </script>
</body>
</html>
```