Linked Multiview Visualization using D3.js and React

Introduction

The project demonstrates a **multiview interactive visualization** using **React** and **D3.js**.

It plots the housing data using two parallel coordinates:

1. **The Scatterplot** displays the relationship between *Area* and *Price*.
2. **The Parallel Coordinates View** displays several numerical attributes (*Price*, *Area*, *Rooms*, *Stories*).

Both of views are **linked**: selecting data points in one plot highlights the corresponding data in the other.

The project supports **2D brushing**, **per-axis brushing** for interactive data exploration.

Data Description

The dataset used is `Housing.csv`, which contains details about houses such as price, area, number of bedrooms, and stories.

Preprocessing

Before visualization, the data is cleaned and converted to numeric values.

Each record is assigned a unique `index` for synchronization between visualizations.

```
const cleaned = response.data.map((d, i) => ({
  ...d,
  area: +d.area,
  price: +d.price,
  rooms: +d.bedrooms,
  stories: +d.stories,
  index: i
}));
setData(cleaned);
```

Attributes used:

- `price` – house price
- `area` – house area
- `rooms` – number of bedrooms
- `stories` – number of stories

System Design

The visualization is built using React components and D3 classes.

React Components

- `App.js`
  - Loads and preprocesses the dataset.
  - Maintains shared application state: `selectedItems`.
  - Connects both visualization components through controller methods.
- `ScatterplotContainer`
  - Mounts and manages the D3 scatterplot.
  - Handles point selection, brushing, and updating highlights.
- `ParallelCoordsContainer`
  - Mounts and manages the D3 parallel coordinates plot.
  - Handles axis brushing and synchronizes selection with scatterplot.

Implementation Details

Scatterplot View

The scatterplot shows the relationship between **Area (x-axis)** and **Price (y-axis)**.

Each point corresponds to one data record.

Features

- **2D brushing:** Drag to select multiple points.
- **Single-click selection:** Click to select a single point.
- **Dynamic highlighting:** Selected points are fully opaque; others fade.

Core Code Snippets

Render and Interaction

```
const points = this.matSvg.selectAll(".markerG").data(data, d => d.index);

points.join(
  enter => {
    const g = enter.append("g")
      .attr("class", "markerG")
      .style("opacity", this.defaultOpacity)
      .on("click", (event, d) => {
        event.stopPropagation();
        controllerMethods.updateSelectedItems([d]);
      });
```

```
        g.append("circle")
            .attr("class", "markerCircle")
            .attr("r", this.circleRadius)
            .attr("fill", "steelblue")
            .attr("stroke", "red");

        this.updateMarkers(g, xAttr, yAttr);
    }
);
```

2D Brushing

```
const brush = d3.brush()
    .extent([[0,0],[this.width,this.height]])
    .on("end", (event) => {
        if (!event.selection) {
            controllerMethods.updateSelectedItems([]);
            return;
        }
        const [[x0,y0],[x1,y1]] = event.selection;
        const selected = data.filter(d => {
            const x = this.xScale(d[xAttr]);
            const y = this.yScale(d[yAttr]);
            return x0 <= x && x <= x1 && y0 <= y && y <= y1;
        });
        controllerMethods.updateSelectedItems(selected);
    });
```

Highlighting

```
highlightSelectedItems(selectedItems) {
    const selectedSet = new Set(selectedItems.map(d => d.index));
    this.matSvg.selectAll(".markerG")
        .style("opacity", d => selectedSet.has(d.index) ? 1 : 0.3)
        .select(".markerCircle")
        .attr("stroke-width", d => selectedSet.has(d.index) ? 2 : 0);
}
```

Parallel Coordinates View

Each line represents one data record across multiple dimensions.

Axes correspond to attributes: `price`, `area`, `rooms`, and `stories`.

Features

- **Axis brushing:** Drag vertically on any axis to select a value range.
- **Linked highlighting:** Selections reflect in the scatterplot and vice versa.

Implementation Highlights

```
_axisBrushed(event, controllerMethods, data) {
  const selections = [];
  this.axisLayer.selectAll('.axis').nodes().forEach((node,i)=>{
    const dim = this.dimensions[i];
    const brushNode = d3.select(node).select('.axis-brush').node();
    const sel = d3.brushSelection(brushNode);
    if(sel) selections.push({dim, range:[this.yScales[dim].invert(sel[1]), this.yScales[dim].invert
(sel[0])]});
  });

  const matched = selections.length===0 ? [] :
    data.filter(d => selections.every(s => d[s.dim]>=s.range[0] && d[s.dim]<=s.range[1]));

  controllerMethods.updateSelectedItems(matched);
}
```

Highlight Synchronization

```
highlightSelectedItems(selectedItems) {
  const selectedSet = new Set(selectedItems.map(d => d.index));
  this.lineLayer.selectAll('path')
    .attr('stroke', d => selectedSet.has(d.index) ? 'orange' : '#4682b4')
    .attr('stroke-width', d => selectedSet.has(d.index) ? 2 : 1)
    .attr('opacity', d => selectedSet.has(d.index) ? 1 : 0.12);
}
```

Linking Both Views

The link between the scatterplot and parallel coordinates is achieved through a shared React state variable, `selectedItems`.

```
const [selectedItems, setSelectedItems] = useState([]);
const updateSelectedItems = (items) => setSelectedItems(items);
```

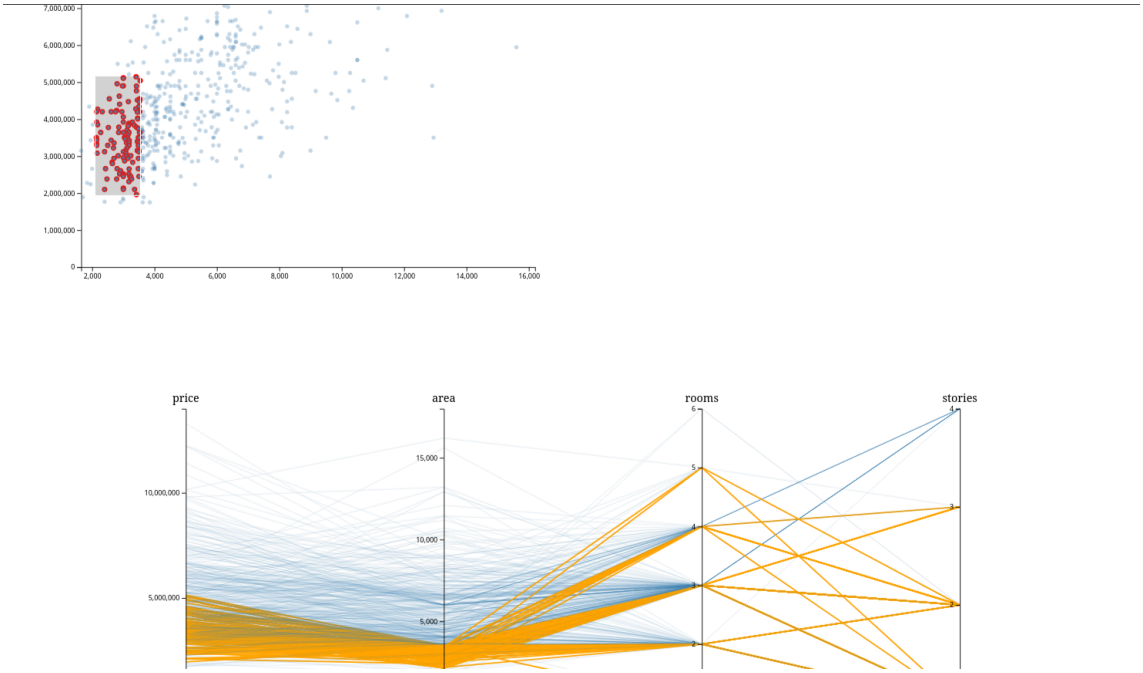
Both D3 views call `updateSelectedItems()` whenever the user interacts, and React re-renders each visualization with the new selection.

This ensures **bi-directional highlighting**:

- Selecting in the scatterplot updates the parallel coordinates.
- Selecting in the parallel coordinates updates the scatterplot.

Results and Observations

Interaction	Effect
Draw rectangle in scatterplot	All points within the region are selected and highlighted in parallel coordinates.
Brush vertically on an axis in parallel coordinates	Only the corresponding data lines and points remain highlighted in both views.



Conclusion

By employing D3.js and React we have created an interactuve multiview visualizatition using scatterplot and parallel coordinates plot. The implementation allows users to get insights into housing data by brushing and selecting subsets of points with correspondence elements being highlighted across both plots. Using event handling, opacity management and selection synchronization the visualization shows intuitive understanding. The result spotlights the coordinated multiple views of revealing patrns and relationships between data paving way to advanced visual analytics.