Interactive Multi-Dimensional Housing Data Visualization with Coordinated Views

Jan Esquivel Marxen
University of Luxembourg
Luxembourg
jan.esquivel.001@student.uni.lu

Abstract

This report presents an interactive housing data visualization combining synchronized scatterplot and violin plot views with multi-rectangle brushing. The left scatterplot shows area-price relationships, while the right view displays price distributions across bedrooms, bathrooms, and stories using kernel density estimation. The system uses React-D3.js patterns with parameterized violin rendering that supports n variables and persistent multi-rectangle selection for complex, non-linear queries. We discuss the visual encoding choices, the React-D3 architecture, and trade-offs of this design.

Note: Parts of this report were developed with AI assistance for code generation and documentation. All technical content was reviewed and validated by the author.

ACM Reference Format:
Jan Esquivel Marxen. 2025. Interactive Multi-Dimensional Housing Data Visualization with Coordinated Views. In *Proceedings of Data Visualization Course (Data Viz 25).* ACM, New York, NY, USA, 2 pages.

1 Introduction

Housing market data includes both continuous variables (area, price) and discrete structural characteristics (bedrooms, bathrooms, stories). While scatterplots effectively show bivariate relationships, they don't reveal distributional patterns across categories.

This project implements coordinated views combining a scatterplot with violin plots:

Left View - Scatterplot with 2D Brush:

- Area vs. Price with multi-rectangle persistent brushing
- Manual rectangle creation supporting complex queries
- Click selection for individual data point inspection

Right View - Violin Plots with Interaction:

- Price distributions for bedrooms, stories, bathrooms (1-5)
- Kernel density estimation with Epanechnikov kernel
- Jittered points overlay within violin contours
- Multi-rectangle brushing synchronized with left scatterplot
- Parameterized design supporting n categorical variables

Selections in either view highlight points in both views. The architecture separates React containers (state management) from D3 classes (rendering).

Section 2 justifies the visual encodings, Section 3 describes the implementation, and Section 4 discusses trade-offs.

2 Visual Design and Justification

2.1 Dataset Properties

The Housing.csv dataset has 545 records with:

- **Continuous:** price, area (sq. ft.)
- **Discrete ordinal:** bedrooms, bathrooms, stories (1-5)
- **Categorical:** furnishing status, and others

User Tasks: Find area-price correlations, compare price distributions across room counts, investigate outliers (which could benefit the user), and select complex subsets (e.g., "3-bedroom, 1 or 3 bathroom, 2 story houses priced 2M-4M").

2.2 Scatterplot Design (Left View)

Encoding:

- **X-axis (area):** Linear scale; position encodes continuous data effectively
- **Y-axis (price):** Linear scale shared with right view for comparison

Interaction:

- **Multi-rectangle brushing:** Draw persistent rectangles to select multiple regions (e.g., both low-price and high-price segments)
- **Click selection:** Single-click to inspect individual points
- **Clear button:** Remove all brushes

Pros: Familiar idiom; multi-rectangle selection enables complex queries; precise position encoding.

Data Viz 25, Luxembourg

Cons: Potential overplotting; no furnishing status encoding; non-standard brush interaction.

2.3 Violin Plot Design (Right View)

Encoding:

- **X-axis:** Categorical bands for counts 1-5; n violins per band
- **Y-axis:** Price (shared scale, axis hidden)
- **Violin contour:** Width encodes KDE probability density
- **Points (jittered):** Raw data overlaid within violins
- **Color:** Tableau10 palette distinguishes variables
- **Separators:** Dashed lines between categories

Interaction: Same multi-rectangle brushing as scatterplot. Selections update both views.

Pros: Shows full distributions with multimodal patterns; color-coding enables comparison; jittered points show raw data; parameterized for n variables.

Cons: Less familiar than box plots; KDE bandwidth needs tuning; requires scrolling for many variables; harder to read than histograms.

2.4 Design Alternatives Considered

Parallel Coordinates: Shows all variables but loses distributional detail and suffers from overplotting. Also a standard technique; we wanted something different.

Heatmap: Shows bivariate distributions but loses individual points and requires binning.

Histogram Small Multiples: Less compact than violins; requires bin tuning; can't easily overlay points.

Our design balances distributional insight, individual data access, and interactive exploration.

3 Implementation and Architecture

3.1 React-D3 Integration Pattern

We separate React (state management) from D3 (SVG rendering):

```
App.js (React root)
  +-- ScatterplotContainer (React)
  |     +-- Scatterplot-d3.js (D3 class)
  +-- ViolinScatterContainer (React)
        +-- ViolinScatter-d3.js (D3 class)
```

App.js: Loads CSV, computes shared `yDomain`, manages `selectedItems` state, provides callbacks.

Containers: Create D3 instances (`useEffect`), compute color palettes (`useMemo`), render legends and controls, call `highlightSelected` on selection changes.

D3 classes: `create()` builds SVG structure, `render()` computes scales and draws, `highlightSelected()` updates styling, `clear()` cleanup.

This avoids re-render loops by creating D3 instances once and keeping imperative logic isolated.

3.2 Multi-Rectangle Brushing

Unlike `d3.brush` (single area), we support persistent multi-rectangle selection:

```
svgElem.on('mousedown.multiRect', (event) => {
 const [mx,my] = d3.pointer(event, this.svg.node());
  this._creating = true;
  this._createStart = [mx,my];
  this._currentRect = this.svg.append('rect')
    .attr('class','multi-brush')
    .attr('x', mx).attr('y', my)
    .attr('width', 0).attr('height', 0);
});
```

On `mouseup`, we store the rectangle and compute the union of selected points. This triggers `highlightSelected()` in both views.

4 Discussion and Trade-offs

4.1 Multi-Brush Interaction Complexity

Pros: Enables complex queries; persistent rectangles show selection criteria visually.

Cons: Non-standard interaction; no per-rectangle delete (only global Clear); overlapping rectangles can be confusing.

Improvements: Add 'x' buttons per rectangle, shift-click to toggle, and zoom/pan for the scatterplot.

4.2 Color Encoding vs. Small Multiples

We color-code three variables and overlay them within each x-category.

Pros: Compact; easy comparison; clear legend.

Cons: Overlapping violins occlude; color harder to distinguish than position; limited palette (10 colors).

Alternative: Small multiples avoid occlusion but use more space and make comparison harder.

5 Conclusion

This project uses scatterplot and violin plot views with multi-rectangle brushing to explore housing data in an interactive, non-linear way. The coordinated views reveal bivariate relationships and distributional patterns across categories. The design balances distributional insight, raw data access, and interactive exploration. Future improvements could include zooming, per-rectangle deletion, and binary variable inclusion.