SAPIENZA
UNIVERSITÀ DI ROMA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE, INFORMATICA E STATISTICA

Engineering in Computer Science

VISUAL ANALYTICS

OMNIVIZ: A UNIFIED DASHBOARD FOR MULTIVARIATE DATA ANALYSIS

Professors:
Giuseppe Santucci
Simone Lenti

Students:
Giuseppe Fosci
Emilio Martino
Alessio Rago

Academic Year 2024/2025

1 Introduction

The visual exploration of high-dimensional datasets presents a significant challenge in modern data analysis. The goal is to effectively map complex, multi attribute information onto a two-dimensional screen, enabling analysts to identify patterns, clusters, and outliers. This project addresses this challenge through the design and implementation of a modern web-based Visual Analytics dashboard built with **React** and **D3.js**.

Our application serves as a comparative environment for exploring multidimensional data, integrating four different and coordinated visualization types: a **Radviz Chart**, a **Stacked Bar Chart**, a **Pie Chart Grid**, and a **raster chart**. The architecture leverages the strengths of both technologies: React is used for its powerful component-based structure and efficient state management, which provides a modular and reactive user interface. D3.js, renowned for its data-driven approach to DOM manipulation, is used as the core engine to render each of the custom data visualizations, offering unparalleled control over the graphical representation of the data.

Our particular interest lies in evaluating the practical utility of Radviz, a nonlinear projection technique. As highlighted in the academic literature, such as "To RadViz or not to RadViz" by Ficorella et al. [1], Radviz can be a powerful tool but also presents interpretation challenges. To investigate this, our dashboard implements several key features:

- **Coordinated Highlighting (Brushing and Linking):** User interactions are linked across specific views to facilitate a multi perspective analysis. For example, selecting or hovering over a data point in the **Radviz Chart** immediately highlights the corresponding bar in the **Stacked Bar Chart** and **Piechart**. This focused coordination guides the user's analytical workflow from a high-level overview to individual detail.

- **Advanced Radviz Implementation:** The Radviz component incorporates the "Effectiveness Error Minimization Heuristic" (EEMH), an algorithm suggested by the literature to optimize the arrangement of dimensional anchors and improve the clarity of the visualization.

- **Flexible Data Handling:** The application is designed to be robust, capable of parsing user-uploaded CSV files by dynamically identifying label and feature columns, and classifying the dataset's properties to inform the user.

This report will first detail the technical architecture of our application, with a focus on how React's state management and D3.js's rendering capabilities were integrated within each component. Then it will proceed with a use-case analysis, using the interactive nature of our dashboard to demonstrate how different analytical tasks are better supported by certain visualizations.

2 Component-Based Architecture

The application is built upon a modular architecture using the **React** library. This approach promotes code reusability, maintainability, and a clear separation of concerns. The structure is organized around a central "orchestrator" component that manages the application's logic and state, feeding data and callbacks to a series of specialized child components. The main components and their specific roles are detailed below.

2.1 App.jsx: The Central Orchestrator

`App.jsx` is the root component and the brain of the application. It holds the primary responsibility for the overall application logic and layout. It initializes and manages all critical state variables, such as the loaded dataset, user selections, and UI configurations. It defines the main visual structure, which is divided into a collapsible control panel and a container for the visualizations.

2.2 The Control Panel (Implemented in App.jsx)

Although not a separate file, the control panel is a distinct functional unit within `App.jsx`. Implemented using Material-UI's `Paper` and other input components, its purpose is to provide the user with a centralized interface for controlling the visualization. It houses all interactive elements for controlling the application.

2.3 RadvizChart.jsx

This component is a specialized wrapper responsible for rendering the Radviz visualization. It encapsulates all the complex D3.js logic required for the radial projection, including the calculation of point positions based on the "spring-force"

metaphor and the implementation of the Effectiveness Error Minimization Heuristic (EEMH). It receives data and a color scale from `App.jsx` and is designed to be highly interactive. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

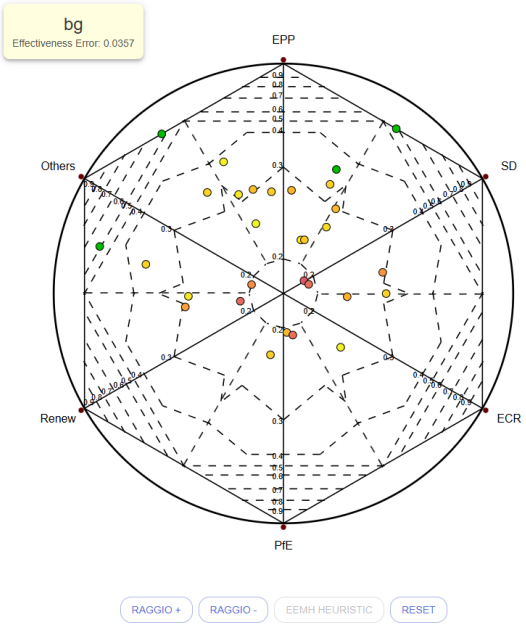

Figura 1: RadvizChart with Effectiveness Error (EE). When hovering over a node, a tooltip displays the node's name and its calculated EE, providing immediate feedback on the quality of the point's placement.

2.4 PieChart.jsx and the Grid Layout

The `PieChart.jsx` component is a simple, reusable module designed to render a single pie chart for a given data item. The more complex logic resides in `App.jsx`, which dynamically renders a grid of these `PieChart` instances, one for each row in the dataset. `App.jsx` also manages the grid's adaptive layout, calculating the optimal number of rows and columns to best fit the available space.

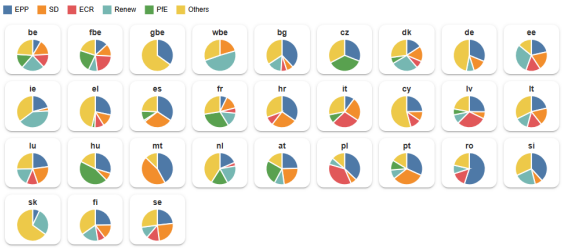

Figura 2: The Pie Chart Grid view, which displays an individual pie chart for each data point in the dataset. This component is designed for detailed inspection of the internal composition of each node.

2.5 StackedBarChart.jsx

The `StackedBarChart.jsx` component is dedicated to displaying the proportional composition of each data item. Its core logic involves normalizing the input data, using D3's stack layout generator (`d3.stack`) to calculate the geometry of each bar segment, and rendering the final chart. This component is a key part of the coordinated-view system, as it visually responds to selection and hover events initiated by Radviz chart. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

2.6 RadarChart.jsx

This component is responsible for creating a radar (or spider) chart to compare the profiles of multiple data points. It takes a subset of the data (typically the selected items) and maps their dimensional values onto a radial axis system. The implementation involves normalizing data to a common scale and using D3's radial line generator to draw the characteristic polygonal shapes. A key feature is its hover-interaction mechanism.

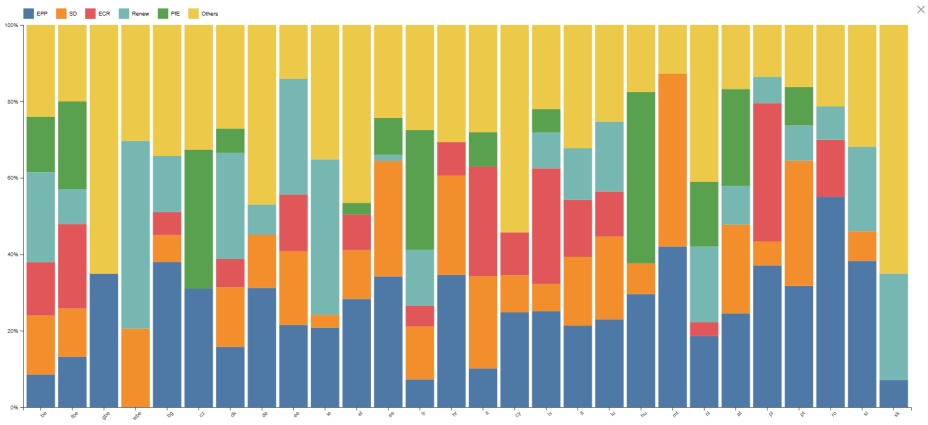

Figura 3: The interactive Stacked Bar Chart, demonstrating the "brushing and linking" feature. When a user clicks on a data point in the Radviz chart, the corresponding bar in this view is immediately highlighted with a distinct border, while other bars are faded. This allows the user to instantly isolate and analyze the compositional breakdown of a specific item of interest.

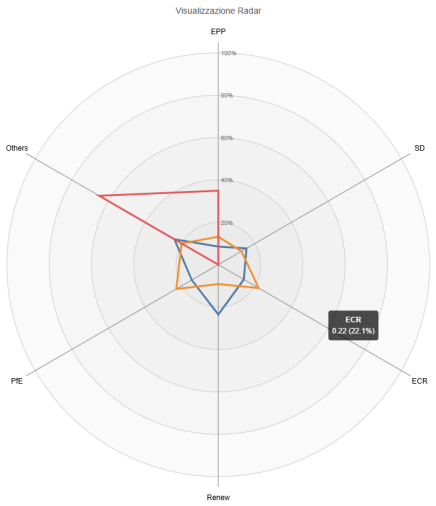

Figura 4: The Radar Chart, used for comparing the "profiles" of different data points. Hovering over a vertex on the line shows a tooltip with the precise value for that dimension, indicating its intensity.

2.7 DataClassifier.js: The Data Service

Though not a React component, `dataClassifier.js` is a critical part of the architecture. This service module contains all the "business logic" for data handling. Its functions are called by `App.jsx` to perform tasks such as fetching data from the server, parsing raw CSV text from user uploads, dynamically identifying name and feature columns, cleaning the data, and running the numerical classification algorithm. By abstracting this logic away from the UI components, the architecture remains clean, and the data processing functions can be tested independently.

3 Enhanced Interactive Features and Data Adaptation

3.1 Cross-Component Synchronization

Building upon the component-based architecture, the application implements a sophisticated **brushing and linking** mechanism that extends beyond the existing Radviz-StackedBarChart coordination. When a user interacts with any visualization component, the system now provides comprehensive visual feedback across all charts simultaneously.

3.1.1 Unified Selection States

The enhanced interaction system supports coordinated highlighting across three visualization types:

- **Node Selection**: Clicking on a data point in the Radviz chart triggers immediate visual feedback in both the stacked bar chart and the corresponding pie chart
- **Hover States**: Mouse hovering over elements provides real-time preview highlighting with distinct visual styling (gray borders) compared to selection states (black borders)
- **Automatic Navigation**: Selected pie charts are automatically scrolled into view, enhancing usability for large datasets

3.1.2 Visual Enhancement System

The pie chart grid now features dynamic visual emphasis through:

- Enhanced stroke width (3px) on pie segments for selected/hovered items
- External container borders with rounded corners for clear identification
- Opacity management to maintain focus on relevant data points

3.2 Intelligent Data Normalization

The application now incorporates an adaptive normalization strategy that automatically adjusts to different dataset characteristics, ensuring optimal visualization regardless of the underlying data distribution patterns.

3.2.1 Automatic Dataset Classification

The `dataClassifier.js` service has been enhanced to automatically categorize datasets into three distinct types:

`dominio_01` Pre-normalized datasets where all values fall within [0,1] range, such as the European election results where each value represents a percentage of votes

`partizionali` Partitional datasets where each row sums to 1.0, representing complete distributions

`generico` Generic datasets with heterogeneous value ranges requiring min-max normalization

3.2.2 Context-Aware Proportional Calculation

For pre-normalized data (e.g., election percentages), the system preserves semantic meaning by directly calculating proportions:

$$\text{proportion}_i = \frac{v_i}{\sum_{j=1}^{n} v_j} \tag{1}$$

For generic datasets with disparate ranges, a two-step normalization ensures fair representation:

$$\text{normalized}_i = \frac{v_i - \min_i}{\max_i - \min_i}, \quad \text{proportion}_i = \frac{\text{normalized}_i}{\sum_{j=1}^{n} \text{normalized}_j} \tag{2}$$

This approach ensures that in automotive datasets, for instance, engine displacement (999-3998cc) and safety ratings (3-5) contribute proportionally to the visualization rather than being dominated by scale differences.

3.3 Enhanced User Interface Controls

3.3.1 Adaptive Pie Chart Layout

The pie chart grid now features intelligent space utilization through:

- **Dynamic Title Management**: Toggle visibility of pie chart labels with automatic margin adjustment (25px with titles, 5px without)
- **Responsive Sizing**: Charts automatically expand to utilize available space when titles are hidden, maximizing data visualization area
- **Optimal Grid Calculation**: Real-time computation of grid dimensions to maintain square aspect ratios across varying dataset sizes

3.3.2 Seamless Integration

These enhancements integrate seamlessly with the existing control panel architecture, maintaining the centralized state management approach while extending the interactive capabilities. The `App.jsx` orchestrator continues to serve as the single source of truth, now managing additional state variables for cross-component synchronization and adaptive rendering preferences.

The result is a more cohesive and responsive visualization environment that adapts intelligently to different data types while providing enhanced interactive exploration capabilities across all visualization components.

4 Development

5 Conclusions

Riferimenti bibliografici

[1] M. Ficorella, S. Lenti, and G. Santucci. To radviz or not to radviz. In D. Archambault, R. Bujack, and T. Schreck, editors, *Eurographics Conference on Visualization (EuroVis) 2023*, volume 42 of *EuroVis*. Eurographics Association, 2023.

[2] Marco Angelini, Graziano Blasilli, Simone Lenti, Antonio Palleschi, and Giuseppe Santucci. Towards enhancing RadViz analysis and interpretation. In *2019 IEEE Visualization Conference (VIS)*, pages 226–230. IEEE.