CSE 578 Personal Reflection Report

AxiSketcher: Interactive Nonlinear Axis Mapping of Visualizations through User Drawings

Shubham Balte
smbalte@asu.edu
Arizona State University
Tempe, AZ, USA

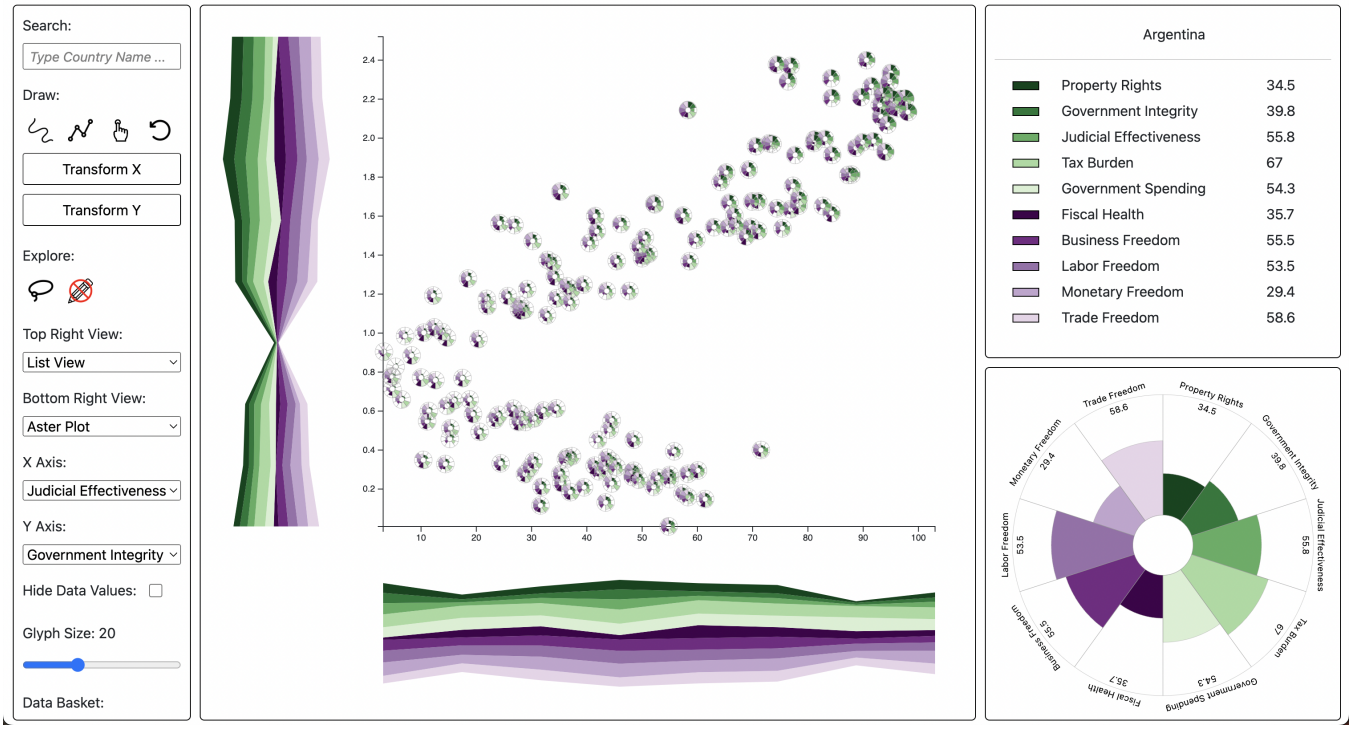

Figure 1: Visualization Overview

1 INTRODUCTION

For the Spring 2024 semester in Data Visualization, we were asked to replicate a visualization system that was previously released, with an extension to the visualization all in D3.js[1] . Me and my team chose to implement the AxiSketcher[3]. system from the list of the sample papers given by the professors. I suggested this choice to the team because it provided a unique way of nonlinear axis transformation, and could be applied to almost any dataset based around preferential data analysis, to which they agreed.

The use cases for AxiSketcher are numerous. The way I like to put it is "playing around with your preferences on a dataset" where the user performs exploratory data analysis to see how their preferences lie in the grand scheme of things, and potentially find other things similar to what they like.

2 DATASET

We decided to use a Kaggle dataset focusing on countries' Economic Freedom Index [4]. The reason we decided on this dataset was because the general use for this kind of a system would be exploratory data analysis, where users would want to discover new data points having similar attributes to the ones that they prefer. However, we wanted to showcase that the system can adapt to other more sophisticated use cases, like comparing different economies which might reveal some key attributes to the users.

Overall, the dataset gives a score out of 100 to each country in these key attributes:

- Property Rights
- Government Integrity
- Judicial Effectiveness
- Government Spending
- Tax Burden
- Fiscal Health
- Business Freedom
- Labor Freedom
- Monetary Freedom
- Trade Freedom

3 VISUALISATION DESIGN

In this section, I will give a brief overview of what the visualization system comprises of. Fig.?? is a basic overview of how the page looks while using the system.

3.1 System Components

3.1.1 Scatter Plot. The main view is a scatterplot that is made up of aster plots. This was from the original paper, which gives a brief detail at each datapoint apart from the selected axes, and also arranges them according to the axes attributes for comparison.

3.1.2 Dashboard. There is a dashboard panel on the left hand side of the page for the users to interact with the visualization. Here users can choose a method of drawing on the scatterplot, search for a specific data point by name, select the lasso tool to select datapoints, switch the axis attributes or side views, etc. When the user selects any items to be seen in detail on the side view panel, they appear in the data basket for giving the user feedback on their selection.

3.1.3 Side View Panels. There are 2 side view panels, which the user can choose to show any of the 5 provided views:

- **List View:** It shows the mean values of each attribute for the data points selected.
- **Table View:** It shows the values of each attribute for the data points selected in a scrollable detailed table.
- **Aster Plot:** It shows the mean value of each attribute in a radial bar chart (aka. aster plot)
- **Parallel Coordinates Plot:** It shows all the data points as lines going through vertical axes of attributes. It highlights the selected data points in red.
- **Radar Plot:** It shows two of the selected data points as spider charts overlapped on top of each other.

If the user has not selected any data points in the data basket, then the side views get updated every time a data point in the scatter plot is hovered upon, with its singular attributes. This helps to visualize the singular data points in more detail for comparisons or selections.

3.2 Interactions

3.2.1 Sketching Axes. The primary method of interaction with this system is by sketching to reconfigure axes. The user can select individual points, or draw a poly-line or free-form line to essentially provide a preferential order of points. Then, the system will "project" these preferences onto the axis the user selects by clicking "Transform X/Transform Y". This results in a stream-graph appearing below the axis which visualizes the projection of all attributes on the axis, and rearranges the scatter plot according to it. Fig.2 shows a user doing this using a poly-line. As you can guess, the line's vertices are *pseudo-points* which are approximated with the help of KNN[2] and an *inverted weighted sum* of distances.

3.2.2 Axis Rainbow. After the creation of the nonlinear axis, the user can further tweak their preferences on individual attributes at a point by hovering over the attribute on the axis rainbow and interacting with the aster plot tool-tip as shown in Fig. 3

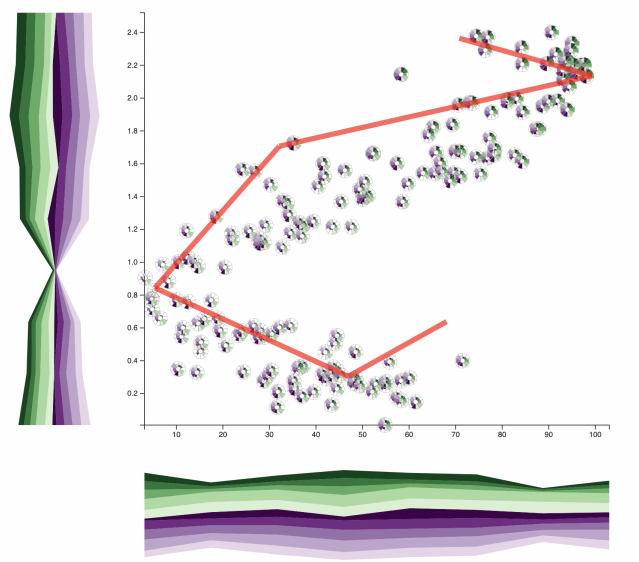

Figure 2: Sketching a poly-line

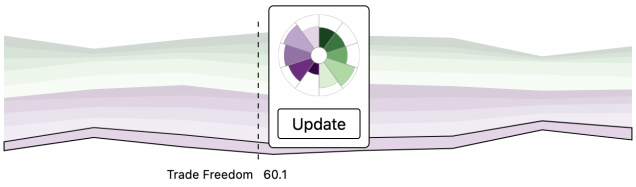

Figure 3: Interaction with axis rainbow

3.2.3 Lasso Selection. The user can either right-click on each data point and add it to the Data Basket, or use the Lasso tool shown in Fig. 4 to select data points for detailed analysis. The user can then press the "Update Side Views" button to update the side view panel with the selected data points and visualizations.

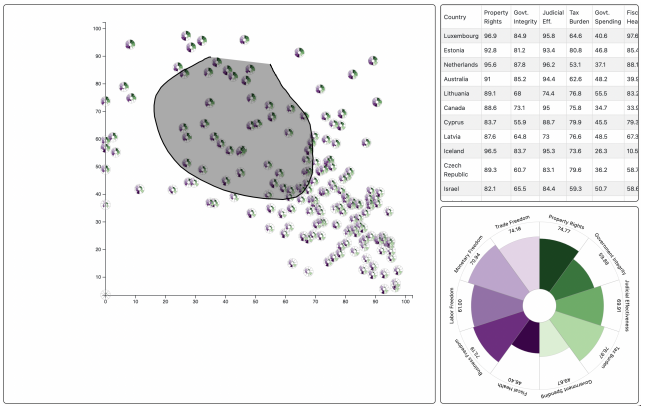

Figure 4: Lasso Selection

4 EXTENSION: RADAR PLOT

I implemented the extension for visualization system, which was a radar plot for comparing two data points in detail. The reason for implementing this as an extension was simple - as the use case for the visualization system is to play around with their own preferences in a dataset, they would eventually land on two data points that they want to compare. The overlapping radar plot is the perfect visualization to do just that. It presents the user with a polygon which is shaped according to each attribute value. The further away from the center the vertex in the polygon is, the higher its value for that attribute.

The radar is provided with 2 different colors for 2 data points, with a legend showing which is which. We experimented with 3 data points overlapping in one chart, but the resulting visualization was cluttered and hard to read, so we limited this to upto 2 data points. When the user hovers over one of the polygons, that polygon will darken and the other will lighten, and also show the datapoint name in the center to provide feedback of the selection. The user can also hover over the vertices of the polygons to see a tooltip of the exact attribute value for that vertex.

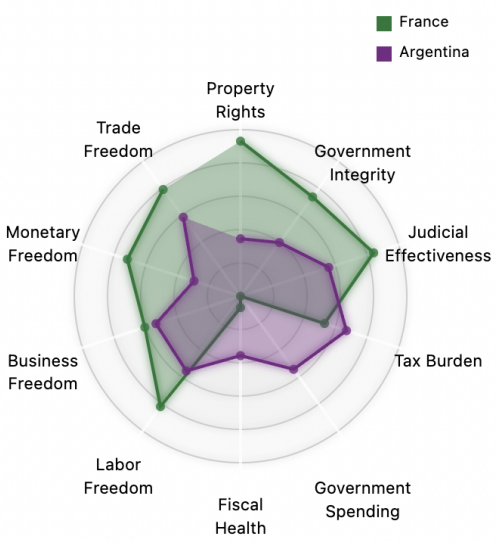

Figure 5: Radar plot

5 MY CONTRIBUTIONS AND LEARNINGS

In the team project, these were the sections where I was the primary contributor:

(1) Radar Plot (Extension)
(2) Table View
(3) List View
(4) Getting *pseudo-points* based on KNN and inverted weighted distance.

The radar plot is explained in detail in its own section. The main challenge with the radar plot was creating the area color according to the vertices. As the dataset attributes were already on the same scale of 0-100, I did not have to implement any normalization techniques for them. The radar chart is implemented as a module to have an "options" argument upon calling the function, which is used to customize different things about it like the number of inner circles shown, or the color scale or opacity to be used, size of the outline and the attribute dots, etc. This gave me an experience of creating a module for an existing code base which can be configured according to will.

The table and list views were pretty basic, but an interesting thing learned through the implementation of list view was modifying the CSS for an HTML <ul> element to show the accurately colored rectangles in place of bullet points.

Another interesting experience was the implementation of pseudo points. We are familiar of using KNN as a classification model in machine learning, but modifying it for our use case was fun. What we do is, we take the X and Y coordinates of the *pseudo-points* from the scatter plot. Then, we go through the dataset to find the K nearest neighbors (in my case, I set it to 3) by getting the points with the least Euclidean Distance calculated as:

$$\text{EuclideanDistance}(\mathbf{p}, \mathbf{q}) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \quad (1)$$

where:

- $\mathbf{p}$ and $\mathbf{q}$ are the two points in n-dimensional space,
- p_x and p_y are the x and y coordinates $\mathbf{p}$

Then, for each point, we used an inverted weighted distance to calculate the attribute values our pseudo point would have. The idea of this is essentially the closer points would be more similar in attributes, so their attributes should be weighed higher. To get the attribute A for the *pseudo-point*, we used

$$\text{A} = (\frac{d_0}{D})a_0 + (\frac{d_1}{D})a_1 + \ldots \quad (2)$$

where $\text{D} = (1/d_0) + (1/d_1) + \ldots$ and d_0, a_0 were the distance and attribute value of point 0. This now gives us a list of pseudo-points to use for the nonlinear axis transformation.

Overall, I had a great experience learning javascript, the nuances of data visualization and its applications in data science, UI/UX design and other various fields.

6 TEAM MEMBERS

These were the team members in my group project:

(1) Sharan Bajjuri
(2) Vamsi Deekshit Kanakavety
(3) Sathish Reddy Gathpa
(4) Manish Meher Edida
(5) Shubham Balte

REFERENCES

[1] [n. d.] D3 by observable | the javascript library for bespoke data visualization. (). https://d3js.org/.

[2] GeeksforGeeks. 2024. K-nearest neighbor(knn) algorithm. (Jan. 2024). https://www.geeksforgeeks.org/k-nearest-neighbours/.

[3] Bum Chul Kwon, Hannah Kim, Emily Wall, Jaegul Choo, Haesun Park, and Alex Endert. 2017. Axisketcher: interactive nonlinear axis mapping of visualizations through user drawings. *IEEE Transactions on Visualization and Computer Graphics*, 23, 1, (Jan. 2017), 221–230. DOI: 10.1109/tvcg.2016.2598446.

[4] MLIPPO. [n. d.] Freedom economic index. (). https://www.kaggle.com/datasets/mlippo/freedom-economic-index.