COVER PAGE

# **Abstract**

Table of Contents

[Abstract 2](#_Toc75564317)

[1. Introduction 3](#_Toc75564318)

[**1.1 Real-time Dynamic Data Visualizations for Streaming Data** 4](#_Toc75564319)

[**1.2 Why do we need a Component that is general and flexible?** 4](#_Toc75564320)

[2. Background 5](#_Toc75564321)

[**2.1 What are Components?** 5](#_Toc75564322)

[**How do they help visualization and design?** 7](#_Toc75564323)

[**2.2 Literature Review** 7](#_Toc75564324)

[**2.2.1 Streaming Data** 7](#_Toc75564325)

[**2.2.2 Data Analysis** 9](#_Toc75564326)

[**2.2.3 User Interaction** 10](#_Toc75564327)

[**2.3 Review of Data Vis Tools** 12](#_Toc75564328)

[**2.3.1 HighCharts** 12](#_Toc75564329)

[**2.3.2. ECharts** 12](#_Toc75564330)

[**2.3.3.** **Google Chart** 13](#_Toc75564331)

[3. Approach 16](#_Toc75564332)

[**3.1. Component Architecture** 16](#_Toc75564333)

[**3.2. Customizing Components** 18](#_Toc75564334)

[**3.3.** **Animated Elements for Streaming data** 20](#_Toc75564335)

[References 22](#_Toc75564336)

# **Introduction**

Streaming data is flooding into small and large organizations, mostly coming from the web, social networks, machines, and devices. As demand for customer insights and faster analytics has grown, so has the need to extract business intelligence from data in real-time [1, 14, 15]. Operational intelligence relies heavily on streaming data. The data is usually automatically processed, and an alert notifies when anything goes outside of a defined threshold. Visualizing this information gives room for understanding better what is occurring and whether any automated decisions should be created, erased, or modified [1, 2].

## **1.1 Real-time Dynamic Data Visualizations for Streaming Data**

Visual elements are helpful when trying to find relationships among hundreds or thousands of variables to determine their relative importance or if they are essential at all. Using a dynamic visual above static visualization presents interactive ways of providing valuable insight into complex data [1, 5, 7]. The field of streaming data visualization is maturing quickly, with several techniques being developed for event detection, handling text streams, and analyzing communication network data [2]. However, there is a need to develop a deeper understanding of how human perception and cognition can cope with complex changes in continually evolving data streams. Despite our high perceptual bandwidth, human attention span is limited, implying that visualizations need to adapt to the fast rates of data streams and need to present and emphasize remarkable changes by updating the underlying data through optimal encoding strategies [8]. Furthermore, the visualization of streaming extensive data, given its scale and volume, poses a new challenge of lack of comprehensive visualization component tools, especially in rapidly evolving complex domains. There is, therefore, the need for flexible, interactive, and dynamic visualization techniques [10].

## **1.2 Why do we need a Component that is general and flexible?**

Components are the primary means to organize and create flexible visualizations. The struggle with customizing a pre-built chart in a way the original developers did not explicitly plan for can be very challenging. The only option is creating a hack or attempts to modify the component source code [9]. Several visualization tools present a one-of design, for instance, a template format that does not have to be recreated each time it is applied or used. Such visualization leaves little to no room for customization or creativity. Making components general and flexible will enhance their functionality and adaptability with other vis tools.

# **Background**

Designing visualization as a series of components makes it scalable, reusable and organized. They can either be used in tandem with another component, on their own or with any data visualization charts.

## **2.1 What are Components?**

A componentis a function object that takes a Selection instance as an argument and adds DOM elements to that Selection. The DOM tree modifications make components the most active tool in D3 because they are injectable into newly created elements in the DOM tree [3]. Thus, components are an integral mechanism for encapsulation and code reuse in D3. Components are likened to the principle of separation of concerns, where a visual design is separated into distinct parts such that each part handles separate concerns [11]. They are different chunks of code that work together and are integrated into a more extensive application.

D3 being a low-level declarative tool, does not have a plethora of inbuilt components. It, therefore, makes room for design control where creativity knows no bound. While exploring components in D3, Janerts [4] in his book used several illustrations and code snippets to describe how a component works. One such example is the sticker component. To achieve its reusability, create two separate components; one function handles the texts and borders, and the second function takes care of the sticker’s positioning.



Figure 2 A simple component continues

Figure 1 A simple component

To use the above component regardless of whether data is bound to the target selection or not, provide a second argument to the sticker function when called and use it as a label [4].

In addition, using the component with bound data, create a <g> element for each data point as usual, and then invoke the sticker() function using call(). It will execute the sticker() function while supplying the current Selection as the first argument [4].

Lastly, using the component without bound data, append a <g> element as the container for the sticker, invoke sticker() via call(), but this time you must explicitly supply a label. The call() facility will forward any arguments past the first one when invoking the supplied function [4].

### **How do they help visualization and design?**

Having described in detail what a component is, it is fitting to consider describing how it helps create visuals and their designs [11].

**Reusability:** A well-created component with the proper levels of abstraction can be used in varied places within an application or across a wide variety of projects while maintaining a single codebase.

**Flexibility:** Several high-level vis tools inhibit the full extent of customization but using D3 components promotes creativity, allowing better representation of complex datasets. It gives developers control over the design to create dynamic visualizations.

**Separation of concerns:** Creating several chunks of code that work together in the form of components removes the hassle of managing intermingled and complex codes.

**Adaptability:** Integrating a well-developed D3 component to other vis tools can enhance the functionality of those high-level vis tools, provided they are built using D3 as the foundation of the application. Trying to modify a high-level vis tool is futile because the code would become more fractured and brittle, never following a consistent workflow.

## **2.2 Literature Review**

### **2.2.1 Streaming Data**

We are in the data-driven era where data gets generated at a spontaneous rate ranging from simple academic records to complex network transmission producing data at continuous intervals [1, 14, 15]. According to the journal written by Kale Panoho [13], he described a study carried out by Deloitte stating that 49 percent of respondents said analyzing datasets helped them make informed decisions, 16 percent said it made key strategic initiatives more efficient, and 10 percent said it improved relationships with customers and peers. It is nevertheless vital to understand how to get the most out of data if the goal is to benefit from it entirely.

It is complex and challenging to extract patterns from streaming data using automation tools such as data mining and machine learning and visualize the patterns to communicate them to analysts [8]. Krstajic and Keim [5], in their article, researched the various tools used for visualizing static data and applied similar tools to visualizing dynamic data and discovered noticeable changes. In their observations, each is affected in some way by the loss of context, to a varying extent depending on the type of change in the data stream, and few of them have issues that also exist with static datasets, including overplotting and problem with visualizing data. Visual search tools effectively visualize data in an intuitive manner; thus, Chin et al. [12] observe that timelines, maps and tree diagrams are typical representations for temporal and geographic data. Nevertheless, if the visualization is to satisfy another requirement, another technique might be better for this purpose, although it would be less intuitive but have a higher information density. When presenting live data in real-time visuals, it is imperative to convey the information in a human-friendly manner to reduce perceptual complexity to support decision-making for the dynamic event stream. What differentiates streaming visualizations from visualizations of dynamic or time-oriented data lies in how the visualization is to be used for real-time change perception. Streaming visualizations are characterized by needing a more immediate decision or action from the user, which often rules out batch-oriented analyses and traditional exploratory tasks [8].

### **2.2.2 Data Analysis**

Visuals make it easy to carry out data analysis and data communication. It allows us to grasp intricate structures that cannot be grasped in any other way. Visualization helps us discover unimagined effects, and it challenges those that are already imagined [28]. It is impossible to see the bits and bytes in a file on a computer hard drive by themselves. We must visualize the data to make sense of it. Every new visual representation of our data offers insights into our data. Some might be already known, while others might be entirely new or even shocking. Some new findings indicate the beginning of an exciting story, while others could be errors in the data that can be discovered by visually viewing the data [16].

Different aspects of data get explored using different kinds of charts. Tables are excellent in handling a relatively small number of the dataset. They reveal labels and amounts in an organized and structured manner and become even more effective when it possesses features such as sorting and filtering through the data. Charts, on the other hand, map dimensions inside the data to visual elements of geometric shapes. While in a scatterplot, two dimensions are mapped to X and Y position, and a third dimension is applied when there is a need for size or colour. It is easier to visualize a temporal evolution with line charts, whereas bar charts are better suited for comparing categorical data [18, 28].

Though aesthetics plays a significant role in visualizations, creating and designing a visual representation that can capture and summarize the data enough to give the analyst a clearer picture of how exciting the data can be has been overlooked. At a minimum, the focus should be creating visuals representing the models and linking to the real-world entities and how they relate to the knowledge at hand [28]. An article written by Enrico Bertini [27] identified a poor visual representation called the multidimensional projections using algorithms such as t-SNE, and MDS was unbearably ambiguous.

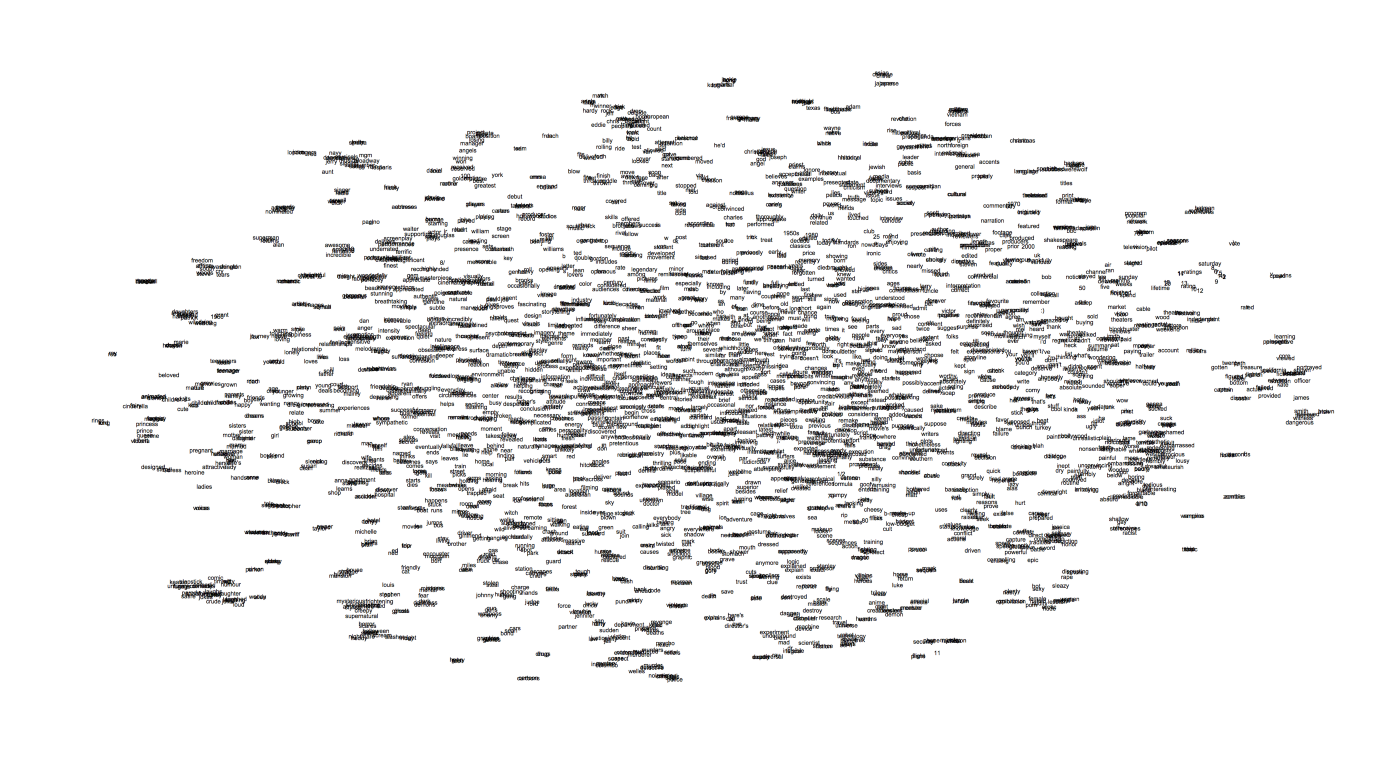


Figure 3 A poor visual representation. It is impossible to understand the representation.

Enrico Bertini [27] suggested that building amazing graphics is not the leading solution to data analysis rather, the focus should be on building tools to harness the full potential of data and visual representations.

### **2.2.3 User Interaction**

Understanding how users interact with visualizations and how those interactions contribute to the analytical process are the central challenges of information visualization [16, 24]. As interaction for visualization continues to grow, researchers advocate for visual designs that would give the end-users the full autonomy to personalize, restructure, remove elements that seem distracting and ultimately connect with visuals within physically appropriate settings [16, 22, 20, 23].

Yi et al. [25] have presented a study that categorizes various user interactions available in popular exploratory visualization tools. They categorize user interaction based on how users can select, explore, reconfigure, encode, abstract, elaborate, filter, and connect visual representations. Similarly, Green et al. [26] proposed that user interactions should be designed to not drive users out of their cognitive zone because their work emphasizes the need to maintain engagement with their task rather than become distracted with navigating a tool.

An article by Dimara [16] established that interaction occurs at every level of visualization, from processing raw data to creating visual representations. Because of this, it has been challenging to reach a consensus on the definition of interaction in visualization. The article further highlighted what is required when interacting with visual representations. For example, users interact with visuals using external entities such as the mouse, pen, keyboard, and other visual cues using speech, eye movement, and body movement. In addition, end-users can express and manipulate data through filtering and aggregation, changing the information of the representation and adjusting movable baseline to compare similarities and differences among various charts [16, 18, 21]. Finally, some visual designs are just difficult to explain in words, but something about how that page progressed logically creating an experience for the user that feels intuitive, how condensed content moved into expanded content, or how those images utilized three levels of hover effect to enable the user to explore the content naturally [16].

However, the authors pointed out that the goal to translate abstract information into visual representations that can be easily decoded is often overlooked. As a result, it is either too limited in visual effects or too ambiguous in that it loses sight of its purpose. In addition, the lack of flexibility in designs could inhibit the ability to perform complex queries or collaborate in real-time [16].

## **2.3 Review of Data Vis Tools**

Some programming experts believe that the declarative tool such as D3 often slows exploration and reuse of designs [29]. However, they proposed that high-level tools offer various visualization styles, are easy to use, customizable, and handle large data sets.

### **2.3.1 HighCharts**

HighCharts is a JavaScript library developed by Jane number technologies, whose underlying drawing technology is formed on SVG. HighCharts supports basic types of charts and rich types of charts such as stock charts and maps. HighCharts also allows users to customize interactive operations beyond the essential functions, and its underlying dependence on SVG drawing provides compelling drawing capabilities. For example, in HighCharts, users can also define the number and position of axes and support rotation, flip and other operations.

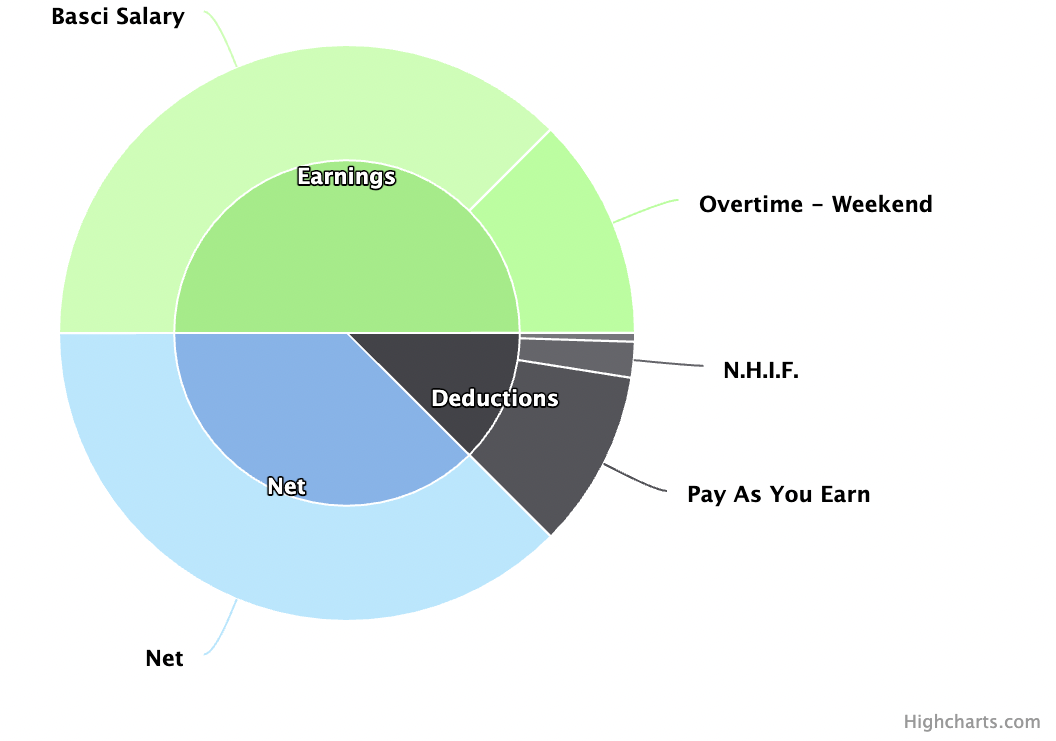
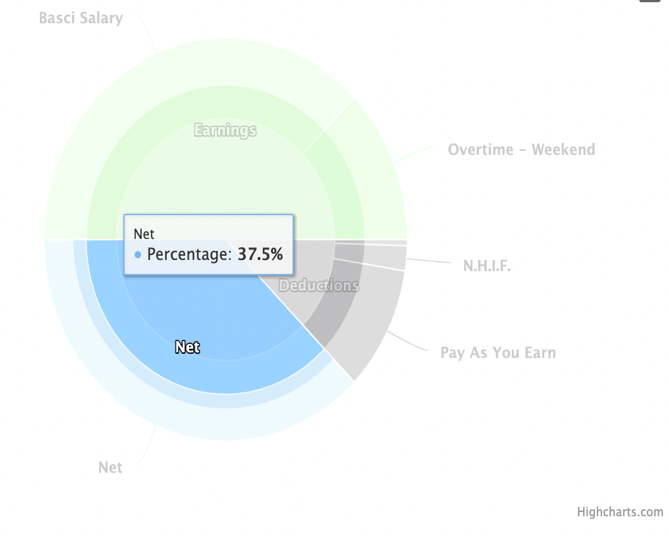


Figure 5 A hover effect for more insight

Figure 4 A Pay Slip Visualization

### **2.3.2. ECharts**

ECharts is a pure JavaScript library, its underlying drawing technology derived from Canvas technology, and the underlying algorithm relies on the ZRender class library. ECharts has a wide variety of chart types and supports rectangular, geographic, polar and other coordinate systems. ECharts is also very well supported on the mobile side, and features supported on the PC side are primarily supported on the mobile side. ECharts also has many advantages in data presentation. Because its underlying principle drew on Canvas drawing technology, it supports a tremendous amount of data, and it also supports multidimensional data above two dimensions and dynamic data display. ECharts also supports flashy visuals since its underlying reliance on Canvas drawing. Because Canvas drawing renders images pixel by pixel, it can draw smooth and beautiful charts and make gorgeous visual effects. In addition, ECharts is compatible with most major browsers [6].

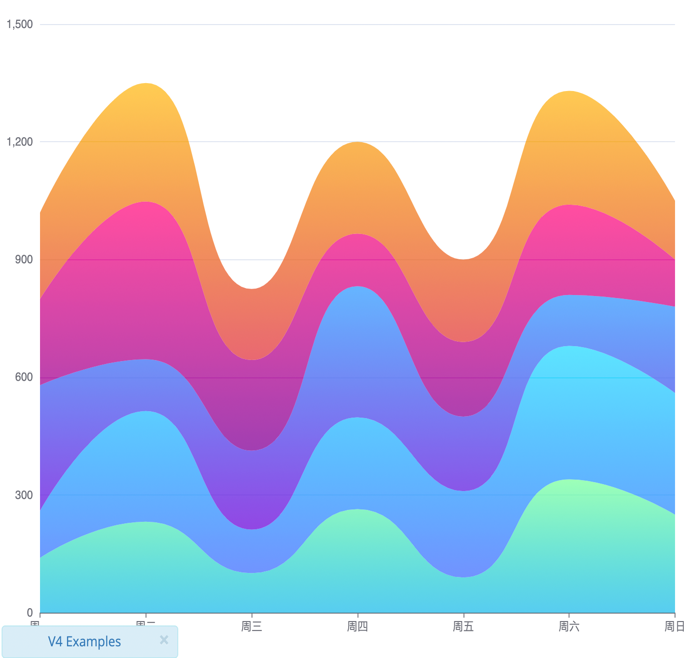
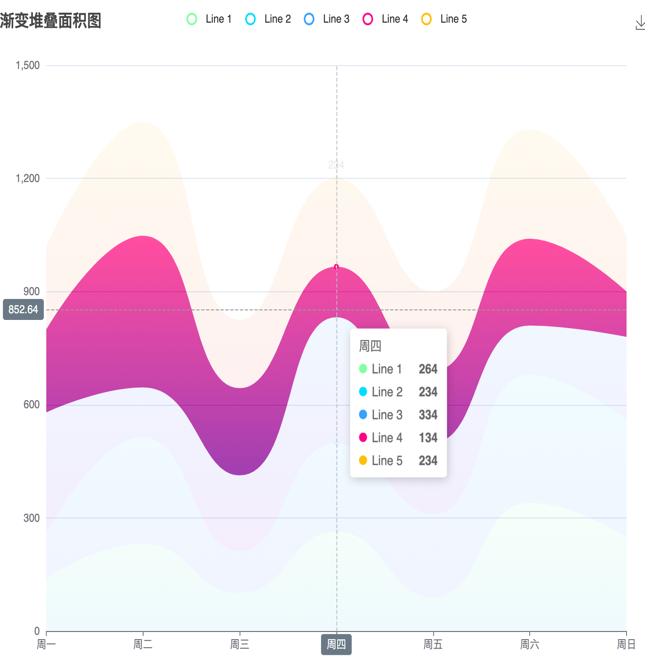


Figure 7 A hover effect for more details

Figure 6 A sample Visual Representation

### **Google Chart**

With Google Chart Tools, users can create charts from data sources and embed them in webpages. Google Charts is based on pure HTML5 and scalable vector-graphics technology; the charts can be displayed on various browsers and platforms, with no plug-ins required. Google Chart Tools support many chart types. There are 12 basic types: pie charts, scatter charts, gauge charts, geo charts, tables, treemaps, combo charts, line charts, bar charts, column charts, area charts, and candlestick charts. Users can create interactive charts. For example, when a user clicks on a visual entity, the chart can display more details about that entity. Google Charts also provide interface widgets such as category pickers, range sliders, or autocomplete. Users can combine charts and controls to create visualization dashboards. The obvious drawback with Google Charts is that it supports only 2D charts and does not support drawing graphs. They are suitable for customizing the chart types they already support but are not suitable for creating new charts [17, 19].

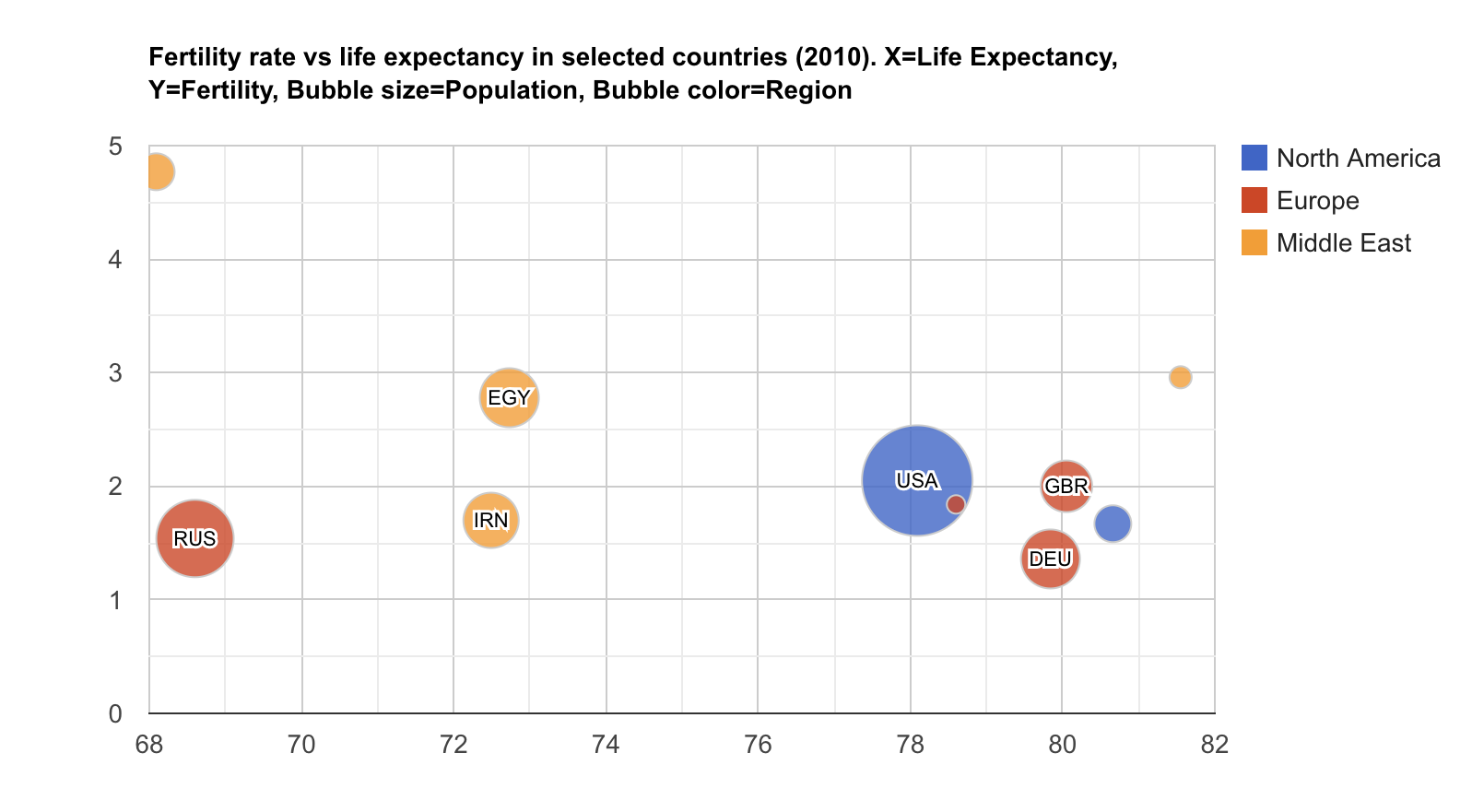


Figure 8 A visual display for fertility rate and life expectancy from different countries

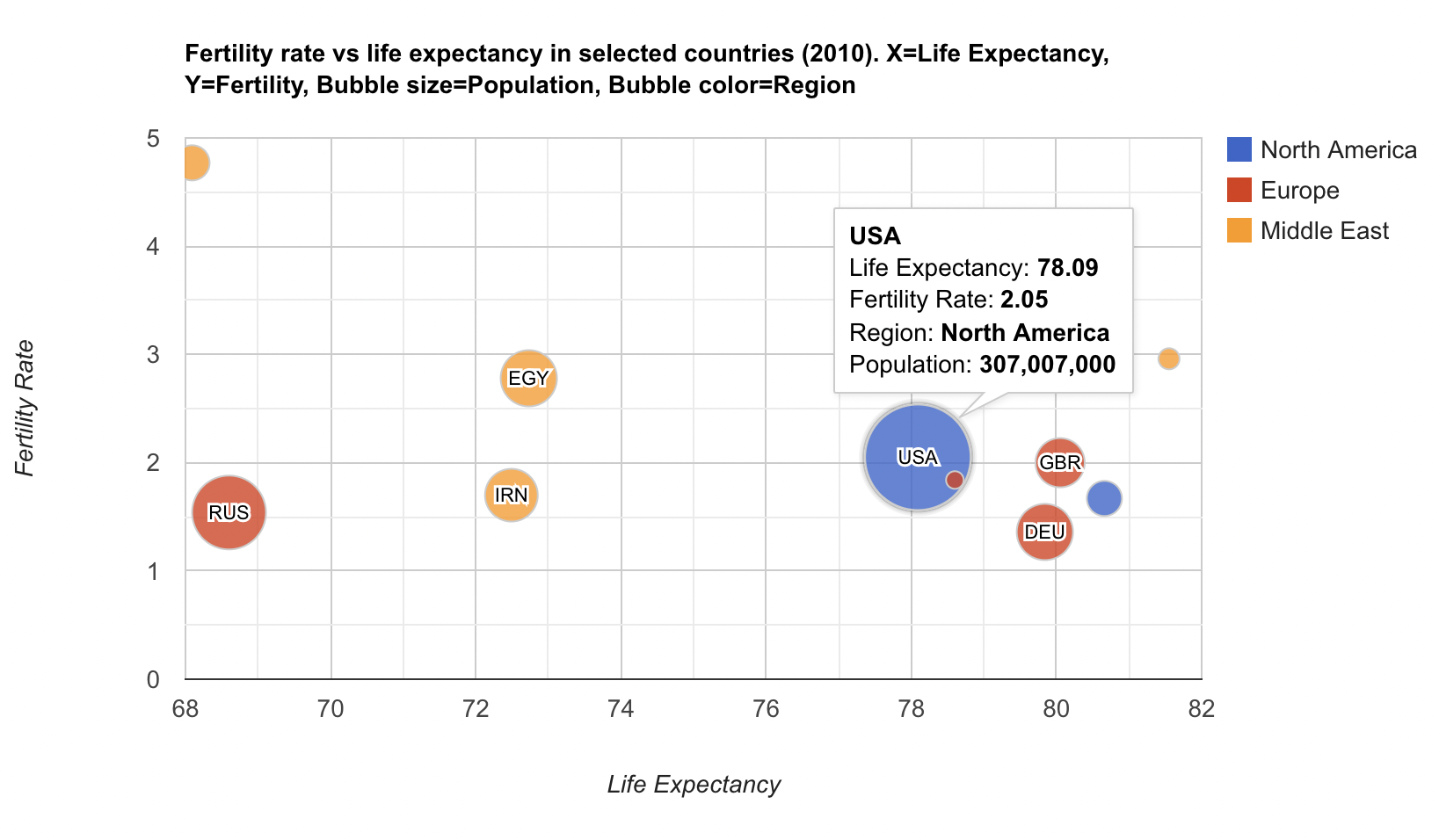


Figure 9 The hover effect displays the detailed summary for each country.

One major drawback with using other vis tools aside from D3 is the degree of freedom to create new charts or enhance their charts. Modifying the existing code to make changes to the visual design has been proven to be difficult and frustrating. However, integrating a new component created using D3 to other vis tools will not pose a challenge since most of those tools were built off D3.js.

# **Approach**

## **3.1. Component Architecture**

The components architecture is divided into three parts.

**Data:** Massive amount of data gets transmitted from different sources in real-time. As the streams of data flow, they are placed in the queue state, and each dataset is displayed on the visual representation one at a time for in-depth analysis. In addition, different operations can be performed on the datasets to give insight into data and watch out for outliers or anomalies.

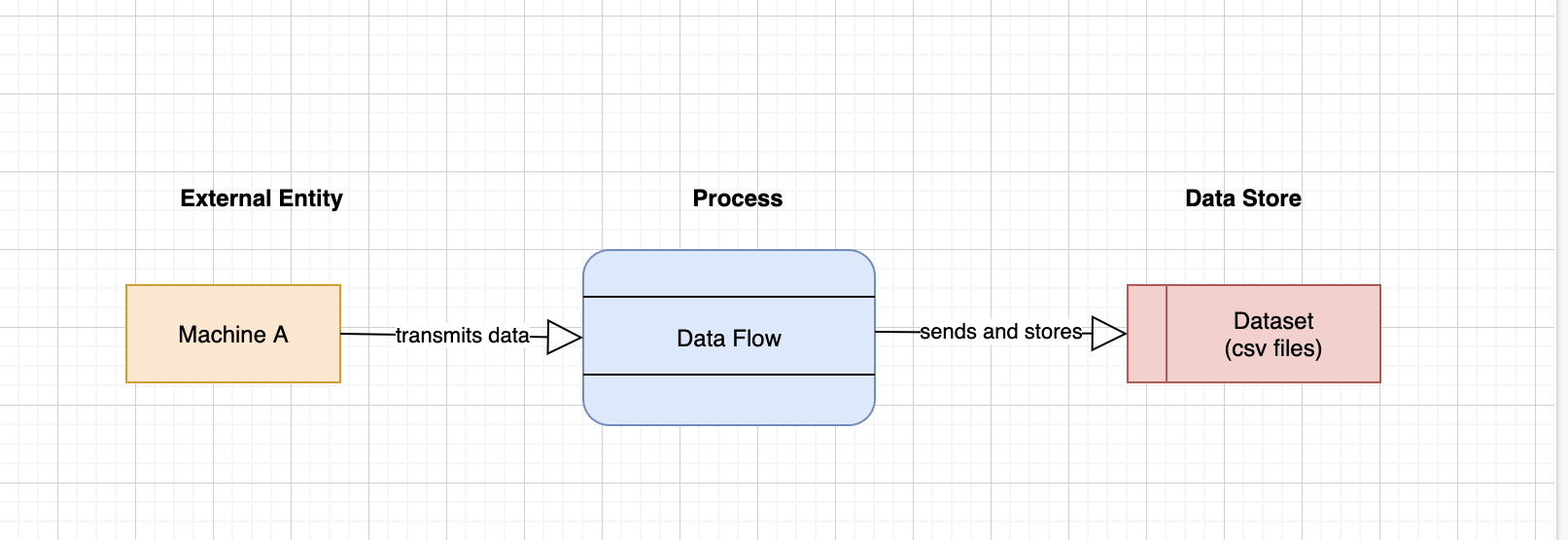


Figure 0 Data flow diagram

**Component:** Components are multifaceted and added to more than one application. In our approach, components can serve as a plug-in that enhances functionalities or libraries that offer an excellent visualization experience. These components are connected to achieve the same goal: to create different charts that summarize what the data is all about. Each serves a different purpose but is interconnected to produce the visual representations.

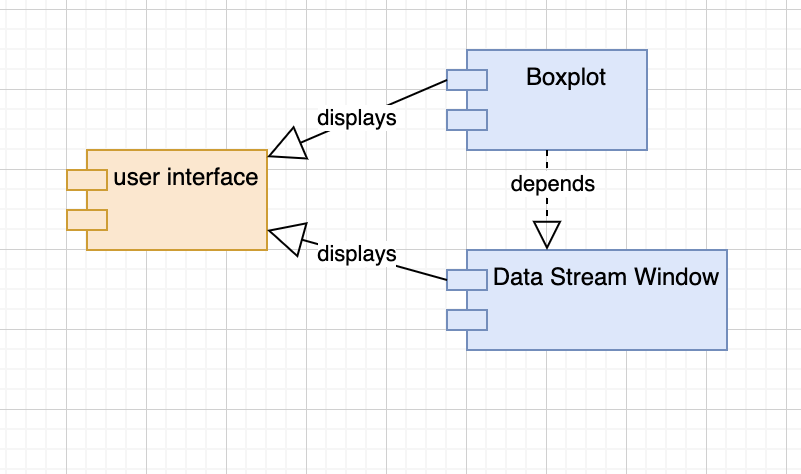


Figure 11 Components

**Visual representation:** The visual representation helps to monitor events or activities at a glance by providing insights from one or more charts on the window. It conveys real-time information by pulling complex data points directly from large datasets and, due to its dynamic nature, makes it easy to interact with by sorting or filtering through the datasets.

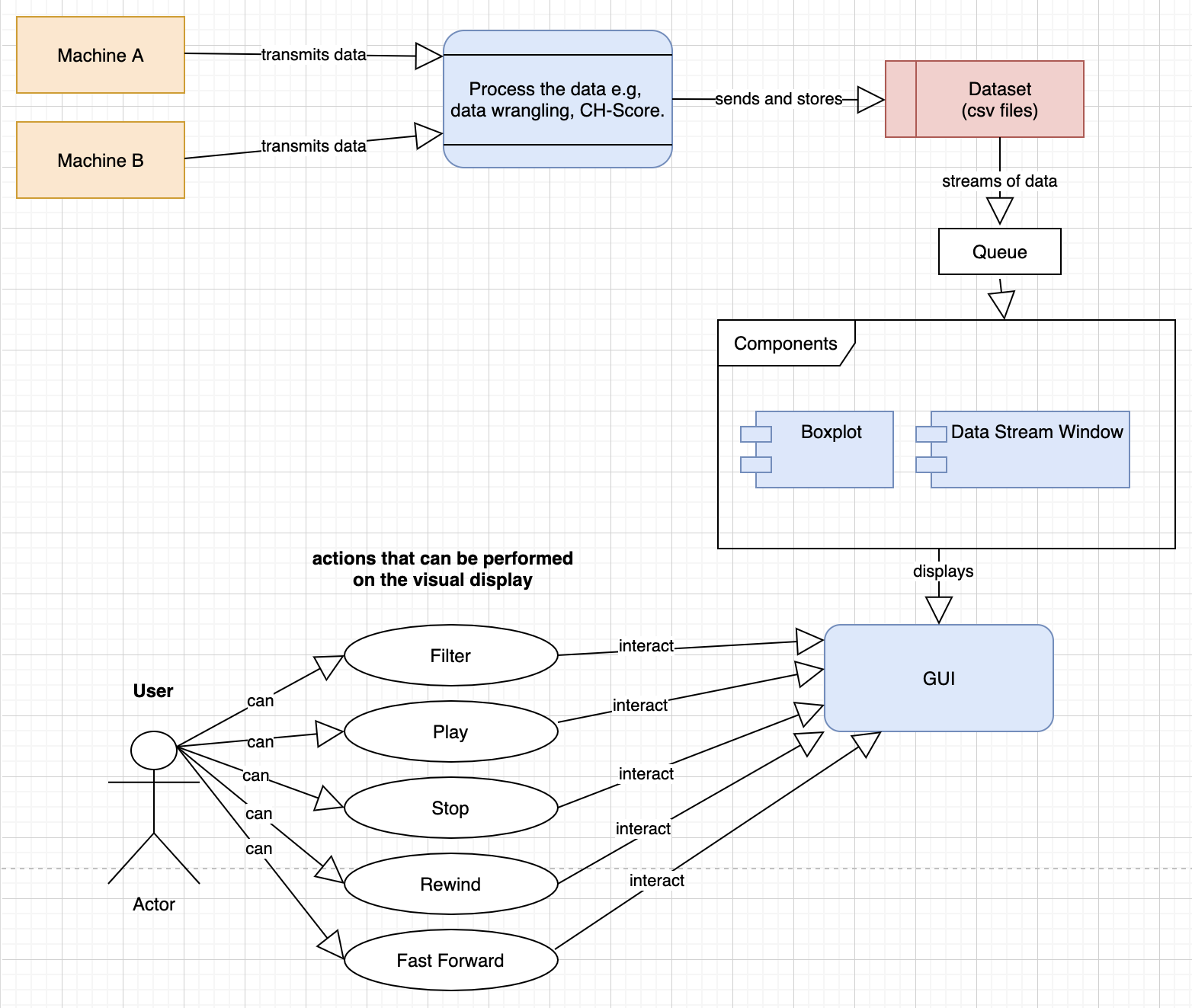


Figure 12 User Interaction Diagram

## **3.2. Customizing Components**

The challenge with other visualization tools aside from D3 is the inability to customize the pre-built charts. As a result, they inhibit creativity, and users are confined to the numerous custom charts provided. D3 does not provide predefined visualizations such as charts, graphs, and maps, but it offers developers unlimited opportunities to create custom visualizations unlike others found in high-level libraries.

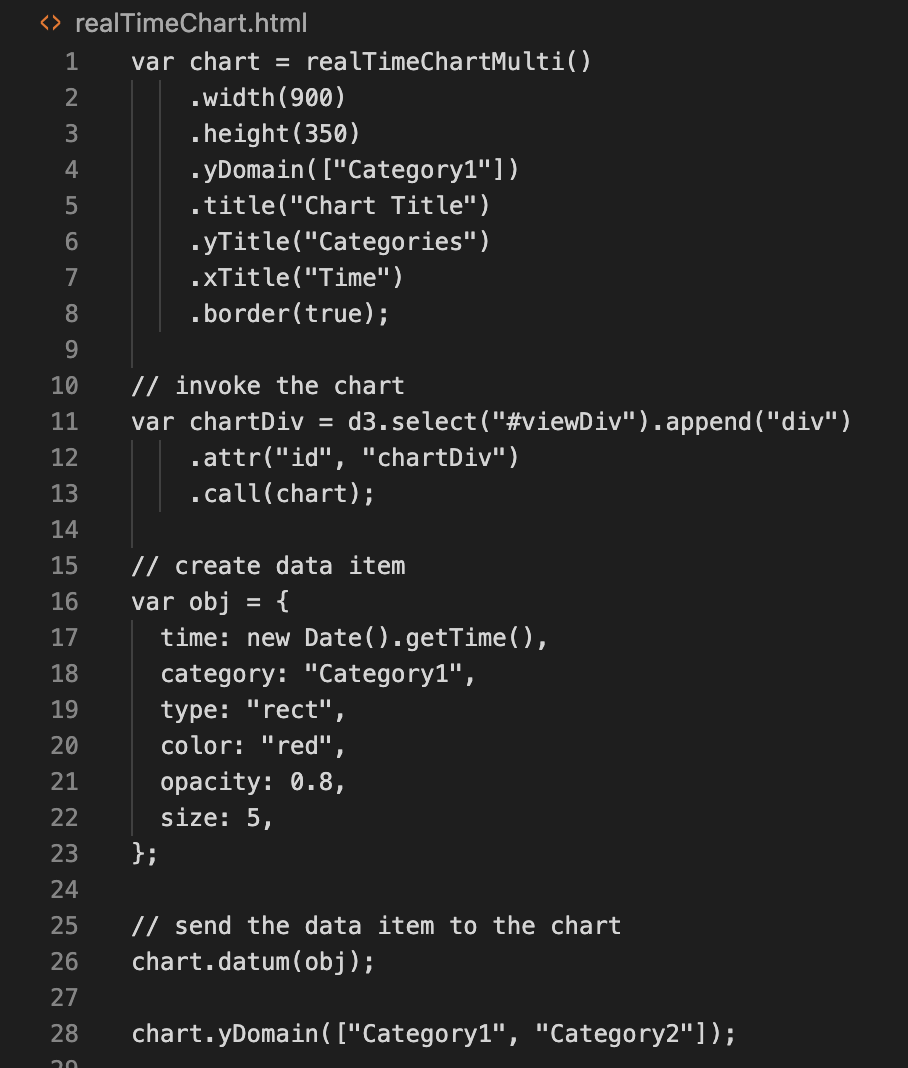


Figure 13 Real-Time Chart

The above code snippet is a real-time component created by Bo Ericsson [30], and below is the visual representation.

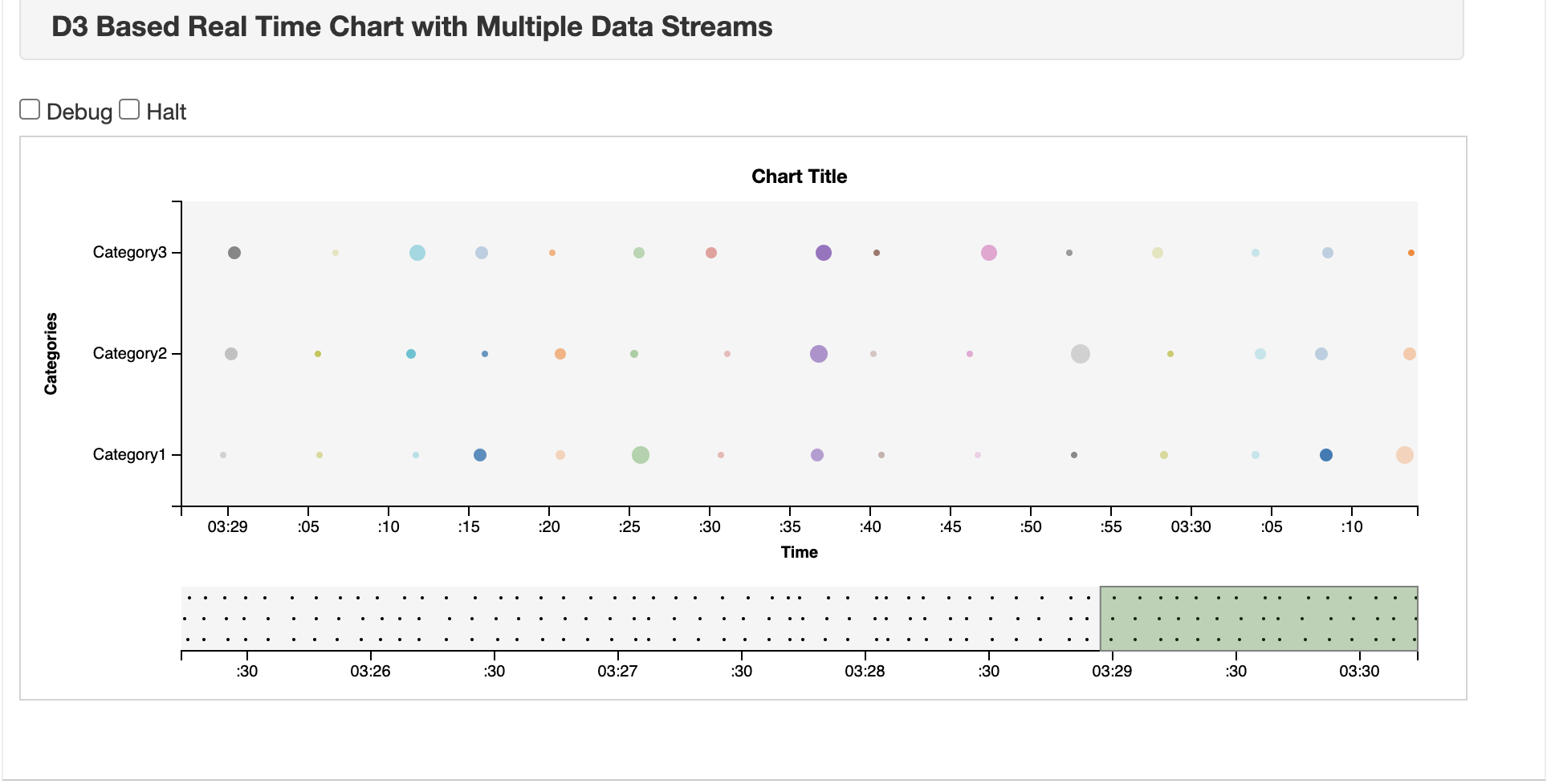


Figure 14 Data Stream

It is a reusable component that accepts data in real-time, and it shows numerous asynchronous data streams to be viewed, each in a horizontal band across the SVG. A new data stream gets added dynamically by calling the yDomain method with the new array of data series.

## **Animated Elements for Streaming data**

Animated data visualization is best described as layering up what the audience sees. Layering up a chart line using several lines and instead of putting all the lines on the chart at the same time, animate the lines and let it transition from one to the other, thereby walking the user through each data point or group. Animated elements create a sense of connection from the user to the dynamic chart, thereby tracking the changes in the visual display [31]. According to Tamara Munzner [32], “Animation is compelling when used for transitions between two dataset configurations because it helps the user to maintain context.”

The animation approach adopted in the component:

**Data Changed:** Observing how data changes from one to another allows the user to gain insights into past behaviour and make informed decisions. The best method is to use the filtering or sorting technique to select different values and see what the data looks like at that point. Another method that allows how much the dataset has existed over time is storytelling. It tells a story with the data and makes the target audience understand how much the data has evolved over the years or over a period. Animating changes in data is a great way to showcase trends, anomalies or patterns in the data.

**Scroll:** Monitoring the dataset scroll in a horizontal band across the SVG in real-time allows the user to respond to crises or report anomalies quicker.

**Mouseover:** It serves several purposes, from zoom in to display statistical ratios of the dataset. A dynamic boxplot with a hover effect gives a detailed summary (Max, Min, Median, First and Third Quantile) of each thread line. Simply following the progression of information from one frame to another provides engaging, detailed content.

# **References**

1. Pereira, T., Moreira, J., Mendes, D. and Goncalves, D., 2020. Evaluating Animated Transitions between Contiguous Visualizations for Streaming Big Data. *2020 IEEE Visualization Conference (VIS)*.
2. Mansmann, F., Krstajic, M., Fischer, F., & Bertini, E. (2011). StreamSqueeze: a dynamic stream visualization for monitoring of event data. *Visualization And Data Analysis 2012*. doi: 10.1117/12.912372
3. Meeks, E., 2017. *D3.js in Action, Second Edition*. Manning Publications.
4. Janert, P., 2019. *D3 for the impatient*. 1st ed. O'Reilly.
5. Krstajic, M. and Keim, D., 2013. *Visualization of Streaming Data: Observing Change and Context in Information Visualization Techniques*. *IEEE Transactions on Visualization and Computer Graphics*, pp.1-7.
6. Li, D., Mei, H., Shen, Y., Su, S., Zhang, W., Wang, J., Zu, M. and Chen, W., 2018. ECharts: A declarative framework for rapid construction of web-based visualization. *Visual Informatics*, 2(2), pp.136-146.
7. Agrawal, R., Kadadi, A., Dai, X. and Andres, F., 2015. Challenges and opportunities with big data visualization. *Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems*.
8. Dasgupta, A., Arendt, D., Franklin, L., Wong, P. and Cook, K., 2017. Human Factors in Streaming Data Analysis: Challenges and Opportunities for Information Visualization. *Computer Graphics Forum*, 37(1), pp.254-272.
9. Teller, S. (2021). Building Animated Components, or How React Makes D3 Better - SitePoint. Retrieved 21 June 2021, from https://www.sitepoint.com/how-react-makes-your-d3-better/
10. Ali, S., Gupta, N., Nayak, G. and Lenka, R., 2016. Big data visualization: Tools and challenges. *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*.
11. Gonzalez, T. (2021). Building Reusable Data Visualization Components for a Modern Web. Retrieved 21 June 2021, from https://medium.com/nightingale/building-reusable-data-visualization-components-for-a-modern-web-54e19f5863b4
12. Chin, G., Singhal, M., Nakamura, G., Gurumoorthi, V., & Freeman-Cadoret, N. (2009). Visual Analysis of Dynamic Data Streams. *Information Visualization*, *8*(3), 212-229. doi: 10.1057/ivs.2009.18
13. Panoho, K. (2021). Council Post: The Age Of Analytics And The Importance Of Data Quality. Retrieved 21 June 2021, from https://www.forbes.com/sites/forbesagencycouncil/2019/10/01/the-age-of-analytics-and-the-importance-of-data-quality/?sh=336badfd5c3c
14. Yadranjiaghdam, B., Yasrobi, S. and Tabrizi, N., 2017. Developing a Real-Time Data Analytics Framework for Twitter Streaming Data. *2017 IEEE International Congress on Big Data (BigData Congress)*.
15. Fischer, F., Mansmann, F. and Keim, D., 2012. Real-time visual analytics for event data streams. *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12*,.
16. Dimara, E. and Perin, C., 2020. What is Interaction for Data Visualization? *IEEE Transactions on Visualization and Computer Graphics*, 26(1), pp.119-129.
17. Supaartagorn, C., 2016. A Framework for Web-based Data Visualization Using Google Charts Based on MVC Pattern. *King Mongkut’s University of Technology North Bangkok International Journal of Applied Science and Technology*.
18. Xu, K., Ottley, A., Walchshofer, C., Streit, M., Chang, R. and Wenskovitch, J., 2020. Survey on the Analysis of User Interactions and Visualization Provenance. *Computer Graphics Forum*, 39(3), pp.757-783.
19. Zhu, Y., 2012. Introducing Google Chart Tools and Google Maps API in Data Visualization Courses. *IEEE Computer Graphics and Applications*, 32(6), pp.6-9.
20. Berinato, S. (2021). Visualizations That Really Work. Retrieved 21 June 2021, from https://hbr.org/2016/06/visualizations-that-really-work
21. Seinfeld, S., Feuchtner, T., Maselli, A., & Müller, J. (2020). User Representations in Human-Computer Interaction. *Human–Computer Interaction*, 1-39. doi: 10.1080/07370024.2020.1724790
22. Few, S. (2021). Data Visualization for Human Perception. Retrieved 21 June 2021, from https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/data-visualization-for-human-perception
23. Corstorphine, M. (2021). Interaction Design: Big Things Have Small Beginnings | aequilibrium. Retrieved 21 June 2021, from https://aequilibrium.com/article/interaction-design-big-things/
24. Endert, A., & North, C. (2012). *Interaction Junk: User Interaction-Based Evaluation of Visual Analytic Systems* [Ebook] (pp. 1-3). ACM. Retrieved from https://www.cc.gatech.edu/~aendert3/resources/Endert-BELIV2012.pdf
25. Yi, J., Kang, Y., Stasko, J., & Jacko, J. (2007). Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions On Visualization And Computer Graphics*, *13*(6), 1224-1231. doi: 10.1109/tvcg.2007.70515
26. Green, T., Ribarsky, W., & Fisher, B. (2009). Building and Applying a Human Cognition Model for Visual Analytics. *Information Visualization*, *8*(1), 1-13. doi: 10.1057/ivs.2008.28
27. Bertini, E. (2021). From Data Visualization to Interactive Data Analysis. Retrieved 21 June 2021, from https://medium.com/@FILWD/from-data-visualization-to-interactive-data-analysis-e24ae3751bf3
28. Aisch, G. (2021). Using Data Visualization to Find Insights in Data. Retrieved 21 June 2021, from https://datajournalism.com/read/handbook/one/understanding-data/using-data-visualization-to-find-insights-in-data
29. Kim, Y. and Heer, J., 2021. Gemini: A Grammar and Recommender System for Animated Transitions in Statistical Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 27(2), pp.485-494.
30. Ericsson, B. (2020). *D3 Real Time Chart with Multiple Data Streams*. Popular Blocks. https://bl.ocks.org/boeric/6a83de20f780b42fadb9. (https://github.com/boeric/d3RealTimeChartMulti/blob/master/realTimeChartMulti.js)
31. Data@Urban. (2019, July 23). *4 Observations on Animating Your Data Visualizations*. Medium. https://urban-institute.medium.com/4-observations-on-animating-your-data-visualizations-cf987b069c35.
32. Muzner, T. (2014, September 26). Visualization Analysis and Design. https://www.cs.ubc.ca/~tmm/vadbook/.
33. McNulty, K. (2019, December 12). *Animated storytelling using the Javascript D3 library*. Medium. https://towardsdatascience.com/animated-storytelling-using-the-javascript-d3-library-a1c2264142ad. \*\*