CS661A 2022-23-II Scribe: Yash Uttamchandani (yashlu22@iitk.ac.in)
February 6, 2023 Department: CSE

Information Visualization

Contents

1 Summary 3

2 How To Say Nothing with Scientific Visualization 4

3 Information Visualization 5
3.1 What exactly is Information Visualization? 5
3.2 How Information Visualization is different from Scientific Visualization? 5
3.3 What makes an Information Visualization Successful? 6
3.4 A Brief Taxonomy of Information Visualization Techniques 9
3.4.1 Empirical Methods 9
3.4.2 Interaction 10
3.4.3 Frameworks/Systems 11
3.4.4 Applications 12
3.5 Big Data Aspects in Information Visualization 13
3.5.1 Common Objectives for Big data Visualization 13
3.5.2 Considerations for creating big data visualization systems . . . 13
3.5.3 Enhancing visualization 14
3.6 Issues and Risks in Information Visualization 14
3.7 Libraries for Data Analysis and Information Visualization 15
3.7.1 MatplotLib 15
3.7.2 Seaborn 16
3.7.3 Bokeh 16
3.7.4 Plotly 16
3.7.5 Plotly Dash 17
3.7.6 D3 Data-Driven Document 17
3.8 Plotly Python Library 18
3.8.1 Main Idea in Plotly 19
3.8.2 Plotly Express Module 20
3.8.3 High Level Features of Plotly Express 22
3.8.4 Plotly Graph Objects Module 24
3.8.5 Figure Data Structure in Plotly: 24

3.9 Bokeh Library 25
3.9.1 Bokeh Features 25
3.9.2 Interfaces to Bokeh 26
3.9.3 Bokeh: Behind the Scenes 26
3.9.4 Bokeh Concepts 27
3.9.5 Example of Bokeh Plot + Interaction Tools 27

1 Summary

- Brief note about the various techniques that can be used to mislead or hide information in scientific visualizations.
- Information visualization is the process of presenting data in a visual format, such as graphs and charts, to help understand complex information quickly and effectively. It aims to reveal patterns and insights in data to support informed decision making.
- There are mainly two categories of methodologies for effective visualization, Namely, Model and Evaluation.
- Interaction is a key aspect of information visualization. It is categorized into two groups: WIMP and Post-WIMP.
- There exists many visualization systems and frameworks to help with the creation and customization of visualizations for various applications. These includes, Tableau,Improvise, The InfoVis Toolkit, D3, etc.
- Key considerations for creating effective big data visualization systems include using accurate and relevant source data, designing easy to understand visualization, and enhancing the visualization through the use of graphical overlays, reference structures, highlights, encodings, and summary statistics.
- Issues and risks in information visualization include imprecision and inaccuracy, oversimplification of complex data, security concerns and many more. It's important to ensure accuracy and quality of visualizations, minimize bias, and protect sensitive data.
- The most commonly used library for information visualization is Matplotlib, which is a basic and standard library in Python. Seaborn, built on top of Matplotlib, offers a simpler interface for creating statistical graphics. Bokeh is a strong tool for interactive visualizations on the web, while Plotly offers a vast array of options for animated graphics. Plotly Dash is a source-available library for building web-based analytical tools, and D3, a JavaScript library, is capable of manipulating and presenting data directly in a web browser.

2 How To Say Nothing with Scientific Visualization

These are the practices that **should** be kept taken care of in Scientific Visualization. [1] They are as follows:

1. Never include a color legend: A color legend makes it easier for the viewer to understand complex visualizations, particularly if multiple variables are being represented using different colors.

2. Avoid annotation: By adding annotations, the visualization becomes more effective in communicating complex information and making data insights easier to comprehend. It plays crucial role by providing additional context and information about data being presented.

3. Never mention error characteristics: Mentioning the characteristics of scientific visualization is important to ensure its accuracy, clarity, simplicity, and relevance in representing and communicating data insights.

4. When in doubt, smooth: Smoothing data when in doubt is not always a good idea as it can obscure important features or lead to incorrect interpretations. Over-smoothing can hide outliers, change the shape of the data distribution, or misrepresent the actual data.

5. Avoid providing performance data: Avoiding performance data in scientific visualization can reduce its accuracy and effectiveness in communicating data insights, and limit transparency and accountability. It is always recommended to provide performance data.

6. Never learn anything about the data or the discipline: Lacking knowledge about the data and the discipline can negatively impact the effectiveness and accuracy of scientific visualization. It is important to have a good understanding of the data and the discipline to create effective and accurate visualizations.

7. Never compare with others: Not comparing with others can limit the quality and effectiveness of scientific visualization by preventing learning from others' experiences and insights.

8. Never cite references of data: If you don't cite references then it can reduce the credibility and transparency of scientific visualization, making it difficult for others to trust the results. It is important to always cite references to ensure credibility and transparency.

9. Claim generalizability but show result on a single data: Claiming generalizability from results based on single data set can be misleading. To claim generalizability, multiple data sources should be considered.

10. Use view angle to hide shortcomings: Hiding limitations by using a specific view angle in undermines the accuracy and credibility of the results. It is important to be transparent and address limitations to enhance the accuracy and trust in the visualization.

11. 'This is easily extended to 3D': It is not always easy to extend to 3D. Claiming a 2D visualization is easily extended to 3D without proper consideration may result in a less effective and less accurate visualization.

3 Information Visualization

3.1 What exactly is Information Visualization?

Information visualization is an interdisciplinary field that deals with the graphic representation of data and information. It is a particularly efficient way of communicating when the data or information is numerous as for example a time series. It is also the study of visual representations of abstract data to reinforce human cognition. The abstract data include both numerical and non-numerical data, such as text and geographic information. It is related to infographics and scientific visualization. One distinction is that it's information visualization when the spatial representation (e.g., the page layout of a graphic design) is chosen, whereas it's scientific visualization when the spatial representation is given. [2]

3.2 How Information Visualization is different from Scientific Visualization?

The visual representation of data is the focus of both information visualization and scientific visualization, but they differ in their emphasis and objective. Information visualization primarily aims to present data in a comprehensible and attractive way to a wider audience, particularly in fields such as journalism, social sciences, and business. Its goal is to facilitate the discovery of patterns, trends, and relationships in the data. Scientific visualization, on the other hand, is a subset of computer graphics and primarily focuses on the representation of complex scientific and technical information, including medical imaging, simulations, and engineering designs. Its main objective is to assist scientists and researchers in comprehending and examining complex data by

creating visual depictions that bring out crucial features and patterns.
Thus, while both approaches deal with the visual representation of data, information visualization emphasizes communication and comprehension, while scientific visualization prioritizes analysis and discovery.

3.3 What makes an Information Visualization Successful?

An information visualization is considered successful if it effectively communicates the intended message or insights in a clear, concise, and aesthetically pleasing manner. Some key factors that contribute to the success of an information visualization are:

- Relevance: The data being visualized should be relevant and meaningful to the audience.
- Clarity: The visualization must be easy to comprehend, and the information must be presented clearly.
- Accuracy: The visualization must correctly depict the data and not mislead the audience.
- Aesthetics: A well-designed visualization can make the information more engaging and easier to understand.
- Interactivity: Allowing the audience to interact with the visualization enhances their comprehension and increases engagement.

In conclusion, a successful information visualization must effectively communicate its message, captivate the audience, and provide insights that would not be possible through other means.

Some examples of Information Visualization

1) Information Visualization for Business Data
Through this visualization we can see that how the topic strength changes over the time.

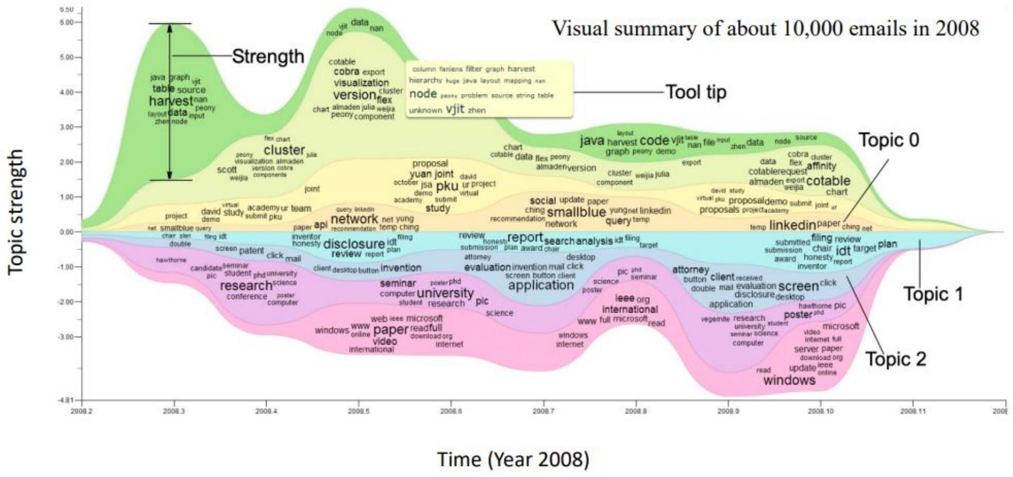


Figure 1: TIARA-created visual summary of about 10,000 emails in 2008. The x-axis encodes time, and the y-axis encodes topic strength. Each layer represents a topic and its evolution over time through a set of keywords distributed along the time line. The tool tip shows the overall, aggregated keywords of the topmost topic (green one).[3]

2) Information Visualization for Science Data

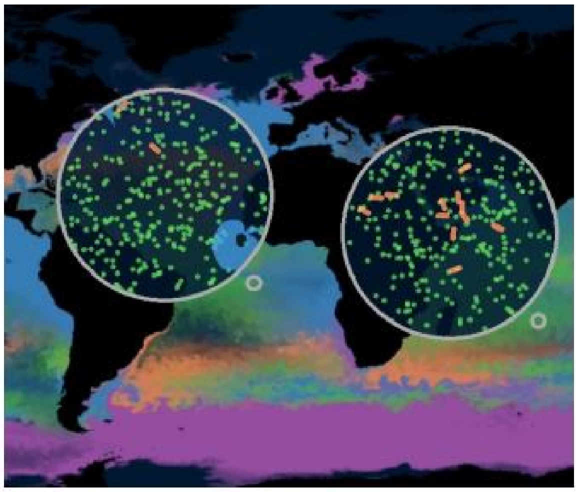


Figure 2: Circle viewers showing the relative abundance of phytoplankton type [4]

3) Information Visualization for ML Classifier Below figure describes a detailed evaluation of classifiers for model selection and debugging. An interactive, comparative, model agnostic visualization system.[1]

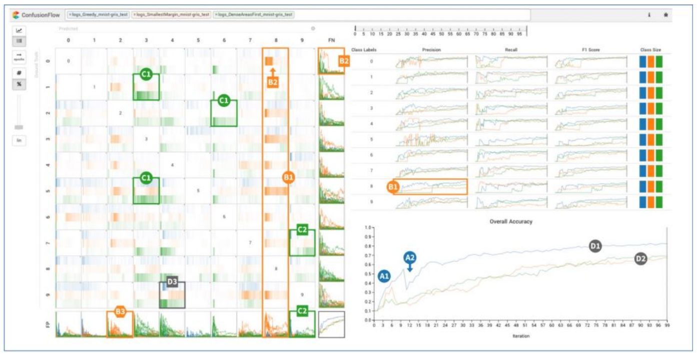


Figure 3: Visual comparison of instance selection strategies for effective data labeling. In an experiment, the collaborators tested three different labeling strategies: Greedy (Blue), Smallest Margin (orange), and Dense Areas First (green). Using ConfusionFlow, the collaborators made a series of findings regarding the overall performances (A, D1, D2) as well as the temporal progression of class confusions (B, C, D3) for the different strategies.[5]

4) Information Visualization for ML Model Explainability

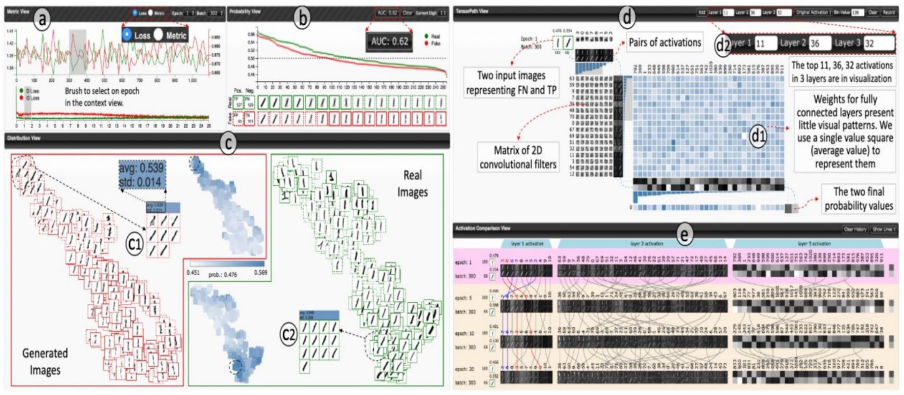

Brush to select on epoch in the context view.
AUC: 0.62
Two input images representing FN and TP
Pairs of activations
The top 11, 36, 32 activations in 3 layers are in visualization
Weights for fully connected layers present little visual patterns. We use a single value square (average value) to represent them
Matrix of 2D convolutional filters
The two final probability values
avg: 0.539
std: 0.014
Real Images
Generated Images

Figure 4: (a) the metric view with focus+context support; (b) the probability view shows the probability of real/fake images in a decreasingorder; (c) the distribution view reveals the feature distributions of real/fake images; (d) the TensorPath view highlights important activations, filtersand weights of D; (e) the activation comparison view tracks the changes among multiple pairs of images from different sources or timestamps.[6]

3.4 A Brief Taxonomy of Information Visualization Techniques

3.4.1 Empirical Methods

Empirical methods are data-driven techniques used to test and evaluate theories, products, or designs. In the context of information visualization, empirical methods involve using experiments and user studies to evaluate the effectiveness of different visual representations in conveying information. The goal of empirical methods is to provide evidence-based insights into the strengths and weaknesses of different visualizations, and to guide the design of effective visual representations of information.
Information visualization (InfoVis) researchers have put a significant effort into developing empirical methodologies to support the design and implementation of effective visualizations.
These methodologies can be broadly categorized into two categories:

- Model and Evaluation.

- The Model category consists of various approaches for creating visual representations of information, including:

  - Visual representation model: This approach focuses on creating visual representations of information that accurately capture the underlying data and patterns.

  - Data-driven model: This approach involves creating visual representations of information based on data-driven techniques, such as clustering, dimensionality reduction, and graph-based methods.

- The Evaluation category focuses on methods for measuring the effectiveness and performance of visual representations of information. The most commonly used evaluation method in InfoVis is:

  - User studies: User studies involve testing visualizations with real users and collecting data on their performance and feedback. This is a scientifically sound method for evaluating visualization performance, and it can provide valuable insights into the strengths and weaknesses of different visualizations.

  - Statistical analysis: This approach involves using statistical methods to analyze data from user studies and other sources, in order to make inferences about the effectiveness of different visualizations.

By using these empirical methodologies, InfoVis researchers are able to create and evaluate visualizations that are both novel and useful, providing a valuable contribution to the field of information visualization.

3.4.2 Interaction

Interaction is a key aspect of information visualization (InfoVis) techniques, as it allows users to interact with visual representations of data and gain insights into the underlying information. Interactions in InfoVis can be broadly categorized into two groups: WIMP and Post-WIMP.

- WIMP (Windows, Icons, Mouse, Pointer) interactions are based on traditional desktop interfaces, where users interact with the visualizations using a mouse and pointer to select, explore, and manipulate data.
- Post-WIMP comprises work on user interfaces, mostly graphical user interfaces, which attempt to go beyond the paradigm of windows, icons, menus and a pointing device, i.e. WIMP interfaces.[7] This category includes touch interfaces, such as those used on smartphones and tablets, which allow users to interact with visualizations using touch gestures, such as tapping and swiping.

Another way to categorize interactions in InfoVis is based on their function or operation, includes: [8]

1. Select: This interaction allows users to select interesting data elements within a visualization.
2. Explore: This interaction allows users to navigate through and explore different aspects of a visualization.
3. Reconfigure: This interaction allows users to change the layout, appearance, or other aspects of a visualization.
4. Encode: This interaction allows users to alter the fundamental visual representation of the data including visual appearance (e.g., color, size, and shape) of each data element.
5. Abstract/Elaborate: This interaction allows users to simplify or elaborate on a visualization, such as aggregating or disaggregating data.
6. Filter: This interaction allows users to filter data elements based on certain criteria.
7. Connect: This interaction allows users to connect data elements in a visualization, creating relationships or dependencies between them.

3.4.3 Frameworks/Systems

Researchers have developed a range of visualization systems and frameworks to help with the creation and customization of visualizations for various applications. These visualization systems/frameworks include:

1. Tableau: A powerful data visualization and business intelligence tool that enables users to create interactive dashboards, charts, and maps.

2. Improvise:Improvise is a visualization system that enables the creation of custom visualizations by combining existing visualization components. This system is designed to support the rapid exploration of alternative visualizations and to enable the rapid creation of custom visualizations.

3. The InfoVis Toolkit: It is a comprehensive visualization system that supports the development of visualizations for a wide range of applications. This system provides a rich set of visualization components and supports the integration of custom visualization components.

4. Prefuse: It is a software framework for creating dynamic visualizations of both structured and unstructured data.It provides theoretically-motivated abstractions for the design of a wide range of visualization applications, enabling programmers to string together desired components quickly to create and customize working visualizations.[9]

More Recently, a new web-based library D3 (Document-Driven Documents) has gained popularity in recent years as a toolkit for constructing interactive visualizations on the web. D3 is unique among visualization systems in that it uses a declarative approach to data visualization. This approach allows for the creation of complex visualizations with relatively simple code.

D3 provides a wide range of features for creating interactive visualizations, including support for creating charts, graphs, and maps. Additionally, D3 supports the use of user-defined interactions, such as mouse events and touch events, allowing users to create visualizations that respond to user inputs.

In summary, As of now D3 is a best web-based library that provides a powerful set of tools for creating interactive visualizations on the web. Its popularity, powerful features, and active community make it a great choice to create data visualizations for the web.

These are just a few examples of the many other visualization systems/frameworks such as **Highcharts**, **Leaflet**, **ggplot2** and many more that have been developed to support the creation and customization of visualizations for a wide range of applications.

3.4.4 Applications

The choice of visualization technique depends on the type of data being analyzed and the application domain. Different types of data and applications require different visualization techniques to effectively convey the information contained in the data. Here are four common types of data and applications and the visualization techniques used for each:

1. Graph data visualization: This type of visualization is used to represent relationships between data points. Graphs such as bar graphs, line graphs, and scatter plots are commonly used for graph data visualization.

2. Text data visualization: This type of visualization is used to represent large amounts of text data. Techniques such as word clouds, tag clouds, and sentiment analysis are commonly used for text data visualization.

3. Map data visualization: This type of visualization is used to represent geographical data. Techniques such as choropleth maps, heat maps, and dot density maps are commonly used for map data visualization.

4. Multivariate data visualization: This type of visualization is used to represent multiple variables in a single visualization. Techniques such as parallel coordinates plots, scatter plots, and matrix plots are commonly used for multivariate data visualization.

In conclusion, the type of visualization technique used depends on the underlying data and application domain. By choosing the appropriate visualization technique, One can effectively convey the information contained in the data to a wide range of audiences.

Exploratory Data Analysis

"The greatest value of a picture is when it forces us to notice what we never expected to see." - John Tukey It highlights the power of visualizations in presenting data. A well-designed visualization can reveal patterns and insights in data that might not be immediately apparent through other means, such as raw data tables. By noticing these insights, visualizations can help us make better decisions and understand complex data more easily.

3.5 Big Data Aspects in Information Visualization

3.5.1 Common Objectives for Big data Visualization

1. Decision initiation or modification: Big data visualizations can help identify trends, patterns, and anomalies in large datasets, and make informed decisions based on that information.

2. Enhancing understanding: Visualizing big data can help simplify complex information and make it more accessible to a wider audience, including stakeholders who may not be familiar with the data.

3. Communication expansion: Visualizing big data can also help facilitate communication between team members and departments, as well as between a company and its customers.

3.5.2 Considerations for creating big data visualization systems

1. Source data: It is important to ensure that the source data used for visualization is accurate, relevant, and up-to-date. This may involve cleaning the data, filling in missing values, and removing any outliers or irrelevant data points. The data should also be in a format that is suitable for visualization, such as a spreadsheet or database.

2. Information transfer to the audience: The visual representation of big data should be designed in a way that makes it easy for the target audience to understand the information being presented. This may involve choosing an appropriate visualization type, such as a bar chart or line graph, that best represents the data. The visualization should also be easy to read, with clear labels and annotations, and should provide relevant context and background information.

3. Design: The overall design of the visualization, including the use of colors, shapes, and labels, should be carefully considered to maximize its effectiveness. For example, the use of contrasting colors can help draw the audience's attention to important data points, while the use of shading can help distinguish between different data sets. It's also important to ensure that the visualization is visually appealing and engaging, as this can help keep the audience's attention and improve the overall impact of the visualization.

3.5.3 Enhancing visualization

1. Enhance visualization by graphical overlays: Graphical overlays, such as charts, graphs, and maps, can help enhance the visual representation of the data.

2. Reference structures: Including reference structures, such as scales or axes, can help provide context and make the data easier to understand.

3. Highlights: Highlighting key data points or trends can help draw the audience's attention to the most important information.

4. Encodings: The use of different encodings, such as size, color, and shape, can help convey different types of information.

5. Summary statistics: Including summary statistics, such as averages, medians, and standard deviations, can help provide a broader understanding of the data.

These are some of the key aspects of big data visualization to consider when creating visual representations of large datasets.

3.6 Issues and Risks in Information Visualization

1. Imprecision and Inaccuracy: Visualizations can sometimes be imprecise or inaccurate, which can lead to misinterpretation or miscommunication of the data. It's important to ensure that the visual representation accurately reflects the data and that any limitations or uncertainties are clearly indicated.

2. Display information at a lower level of precision and accuracy: Visualizations can sometimes display information at a lower level of precision and accuracy than numerical or tabular formats. This is because visualizations are designed to simplify complex data and make it easier to understand, which may involve summarizing or aggregating the data.

3. Optical Significance: Visualizations can be susceptible to optical significance, where the viewer perceives a difference or pattern as meaningful based on their perception, even without corresponding quantitative evidence to support this interpretation. This highlights the importance of carefully considering the design of the visualization and avoiding misleading or deceptive visual representations.

4. Visualization Oversaturation: The proliferation of data visualizations has both led to an increase in high-quality examples and a dramatic increase in deficient and flawed visualizations. This makes it important to carefully evaluate the

quality and accuracy of visualizations and to seek out examples that are well-designed and effectively communicate the data.

5. Bias: Visualizations can be subject to bias, either deliberately or unintentionally, leading to misleading or inaccurate representations of the data. It's important to be aware of potential sources of bias, such as sampling bias or selection bias, and to take steps to minimize their impact on the visualization.

6. Over-simplification: Visualizations can sometimes oversimplify complex data, leading to a loss of important information and context. This can result in misinterpretation or miscommunication of the data.

7. Data Security: Visualizing big data also raises security concerns, as the data may contain sensitive information that could be vulnerable to cyber-attacks or other forms of data theft. It's important to implement robust security measures to protect the data and minimize the risk of security breaches.

3.7 Libraries for Data Analysis and Information Visualization

3.7.1 MatplotLib

It is the most basic and Python's standard data visualization library. It is a 2D plotting library for Python, providing an extensive set of customizable visualizations and plots, including line plots, bar plots, scatter plots, histograms, and more. It is a low-level library, which means that it provides a lot of control over the appearance of visualizations. It also provides tools for creating animations and interactive visualizations. Additionally, it is a well-documented library with a large community of users. While Matplotlib is a powerful library for data visualization, it does have a few drawbacks:

1. Complexity: Matplotlib can be quite complex to use, especially for creating more advanced visualizations.

2. Limited Interactivity: Matplotlib is designed primarily for creating static visualizations, and while it does have tools for creating interactive visualizations, they are limited in comparison to other libraries.

3. Performance: Matplotlib can be slow when working with large datasets, especially when creating animations. This can make it unsuitable for real-time data visualization.

3.7.2 Seaborn

Seaborn is a data visualization library built on top of Matplotlib that provides a higher-level interface for drawing statistical graphics in Python. It is designed to make it easier to create visually appealing and informative plots, with a focus on aesthetics and interactivity.

One of the key features of Seaborn is its ability to create beautiful and visually appealing plots with just a few lines of code. It also has built-in support for working with categorical data and statistical models. It provides a number of interactive visualizations, including heatmaps, scatter plots, and pair plots. These interactive visualizations provide an intuitive way to explore and understand data.

Seaborn has limitations in customization, performance, 3D plotting, and handling complex statistical models.

3.7.3 Bokeh

Bokeh is a powerful library for creating interactive visualizations for modern web browsers, using Python. It offers a high-level interface for building beautiful graphics, including simple plots and complex dashboards, without the need to write any JavaScript code. Bokeh is designed to handle large and dynamic datasets and provides a range of interactive features, such as panning, zooming, and hover tools, to help users explore and understand their data. With Bokeh, visualizations can be easily created that can be embedded in web pages or shared online. As a result, Bokeh is widely used by data scientists, business analysts, and other professionals who want to create dynamic and engaging visualizations for the web.
Here it is discussed in greater detail.

3.7.4 Plotly

Plotly is a Python-based graphing library for creating high-quality visualizations. It offers a wide range of features for creating interactive and animated graphics, including 2D and 3D plots, statistical charts, and maps. With Plotly, visually appealing and interactive visualizations, such as line plots, scatter plots, bar charts, histograms, and many more, can be easily created. Plotly also provides a JavaScript-based graphing library, Plotly.js, built on top of d3.js and stack.gl, which allows to create advanced visualizations that can be embedded in web pages or shared online.
Here it is discussed in greater detail.

3.7.5 Plotly Dash

Dash is an open-source Python library for creating reactive, web-based applications. It is built on top of Plotly.js and React.js and provides a user interface library for creating analytical web applications. Dash is often referred to as "React for Python" because it provides similar functionality to React, a JavaScript library for building user interfaces. It is designed to create web applications with interactive, data-driven visualizations. Dash provides several benefits for creating web applications:

1. Easy to use: Dash provides a simple and intuitive API for building web applications. With a few lines of code, users can create interactive visualizations and build web applications that can be shared with others.

2. Fast and responsive: Dash is built on top of Plotly.js, which provides fast and responsive visualizations, making it ideal for working with large datasets or real-time data.

3. Customizable: Dash provides a high level of customization. This includes the ability to create custom components and add custom JavaScript code for advanced functionality.

4. Scalable: Dash is designed to scale, making it suitable for large-scale web applications with multiple users. It is also designed to be extensible, allowing to add new functionality and components as needed.

Overall, Dash is a powerful and flexible library. Its combination of ease of use, fast performance, and customization options makes it an excellent choice for users who want to create engaging and interactive web applications with Python.

3.7.6 D3 Data-Driven Document

D3 (Data-Driven Documents) is a JavaScript library that allows users to manipulate and visualize data in the browser. It is built to work with web standards, such as HTML, SVG, and CSS, and provides a way to create dynamic, interactive visualizations without relying on proprietary frameworks. D3 is not a monolithic framework that seeks to provide every conceivable feature. Instead, D3 solves the crux of the problem: efficient manipulation of documents based on data.[10] D3 is widely used in data science, journalism, and information design, providing a flexible and powerful toolkit for creating a variety of visualizations. The library is designed to be highly customizable. One of the key strengths of D3 is its data-driven approach to manipulating the Document Object Model (DOM). This allows developers to bind data to elements in the browser and update the visualizations in real-time as the data changes. For example, a developer could bind data to the position, color, or shape of elements in a bar chart, and as the data changes, the visual representation of the chart would update dynamically. In addition to its data-driven approach, D3 provides a wide range of features and tools for working with data, including:

1. Data Binding: D3 provides a simple and intuitive data binding API that allows developers to associate data with elements in the browser.

2. Layouts: D3 includes a number of built-in layouts, such as pie charts, stacked bar charts, and treemaps, which make it easy to create complex visualizations.

3. Transitions: D3 provides a powerful transition system for creating smooth animations and visual effects.

4. Interactivity: D3 provides a flexible and intuitive API for creating interactive visualizations, allowing users to explore and interact with their data in new ways.

In addition to these Strong features, there are few downsides of D3:

1. Learning Curve: D3.js is a complex library and has a steep learning curve for beginners, especially for those without prior experience in web development or JavaScript. It requires a strong understanding of web standards and DOM manipulation, which can be challenging.

2. Performance: D3.js is a client-side library that runs in the browser, which can result in slow performance for large datasets or complex visualizations. This can be a concern for those who need real-time data visualization or who are working with large amounts of data.

3. Limited Mobile Support: D3.js is built for the web and may not work well on mobile devices, depending on the size and complexity of the visualization. For users who need to create visualizations that can be viewed on a wide range of devices, other libraries such as Plotly may be a better choice.

3.8 Plotly Python Library

Plotly is a powerful and versatile library for creating interactive visualizations in Python. It offers a wide range of chart types that can be used for data analysis, visual storytelling, and presentation. Plotly.js is the underlying library for Plotly in Python, providing all the necessary tools for data manipulation and visualization.

In addition to its standalone usage, Plotly can also be integrated with the Dash framework to create web applications. This allows for easy deployment and sharing of interactive visualizations with a wider audience.

3.8.1 Main Idea in Plotly

The main idea in Plotly is the combination of three main components: Data, Layout, and Figure.

- The Data object in Plotly is the primary component that specifies the data to be plotted. It contains information about the data points and how they should be displayed. Each set of data that is to be plotted is referred to as a “trace”. For example, if you were plotting two lines on a line chart, each line would be a separate trace.

- The Layout object defines the overall look and feel of the chart. It contains information about the chart title, axis labels, and other non-data related features. It also allows you to add annotations and shapes to the chart to highlight specific data points or areas. The Layout object can be customized to create charts that match the desired style and branding.

- The Figure object in Plotly is the final product that combines the Data and Layout objects. It defines the complete plot, including both the data and the overall look and feel of the chart.

In summary, the Data, Layout, and Figure objects in Plotly provide a flexible and intuitive way to create interactive visualizations. By breaking down the plotting process into these three parts, Plotly allows users to focus on the data and the desired display options, while taking care of the rest.

Plotly Python Modules

Plotly has several submodules that provide various levels of interface and functionality for data visualization. These submodules includes:

- Express: This is a high-level interface for data visualization in Plotly. It provides a simplified interface for creating common chart types, making it easy to create visualizations without having to delve into the details of the low-level Graph Objects.

- Graph Objects: This is the low-level interface to figures, traces, and layout in Plotly. It provides a more granular control over the elements of the plot, allowing users to fine-tune their visualizations.
- I/O: This submodule provides a low-level interface for displaying, reading, and writing figures in Plotly. It allows users to save their visualizations as JSON files, display them in Jupyter notebooks, or embed them in web pages.
- Subplots: This helper function provides a convenient way to lay out multiple plots in a single figure, making it easy to compare and visualize multiple datasets at once.
- Figure Factories: This is a collection of helper methods for building specific and complex chart types, such as candlestick charts, contour plots, and others.
- Plotly.colors: This submodule provides colorscales and utility functions for color management in Plotly visualizations.
- Plotly.data: This submodule contains built-in datasets for demonstration, educational, and testing purposes, allowing users to test Plotly's functionality and learn from examples.

Each of these submodules provides different levels of abstraction and control, making Plotly a highly flexible and versatile library for data visualization.

3.8.2 Plotly Express Module

Plotly Express is a high-level library that makes it easier and faster to create interactive figures. The library provides a collection of functions that allow you to create entire figures in just one line of code. This makes it ideal for rapid prototyping, exploration, and visualization of data.

The functions in Plotly Express use Plotly graph objects internally, and every function call returns an instance of the **plotly.graph_objects.figure**. This means that you can easily customize your plots using the underlying Plotly API, and you have full control over the appearance and behavior of your figures. Plotly Express currently includes the following functions:

- Basics:
  - Scatter: creates a scatter plot
  - Line: creates a line chart

  - Area: creates an area chart
  - Bar: creates a bar chart
  - Funnel: creates a funnel chart
  - Timeline: creates a timeline chart

- Part-of-Whole:
  - Pie: creates a pie chart
  - Sunburst: creates a sunburst chart
  - Treemap: creates a treemap chart
  - Icicle: creates an icicle chart
  - Funnel area: creates a funnel area chart

- 1D Distributions:
  - Histogram: creates a histogram
  - Box: creates a box plot
  - Violin: creates a violin plot
  - Strip: creates a strip plot
  - ECDF: creates an empirical cumulative distribution function plot

- 2D Distributions:
  - Density heatmap: creates a heatmap of density
  - Density contour: creates a contour plot of density

- Matrix or Image Input:
  - Imshow: creates an image show plot

- 3-Dimensional:
  - Scatter 3D: creates a 3D scatter plot
  - Line 3D: creates a 3D line chart

- Multidimensional:
  - Scatter matrix: creates a matrix of scatter plots
  - Parallel coordinates: creates a parallel coordinates plot

  - Parallel categories: creates a parallel categories plot

- Tile Maps:
  - Scatter mapbox: creates a scatter plot on a mapbox
  - Line mapbox: creates a line chart on a mapbox
  - Choropleth mapbox: creates a choropleth map on a mapbox
  - Density mapbox: creates a density map on a mapbox
- Outline Maps:
  - Scatter geo: creates a scatter plot on a geo map
  - Line geo: creates a line chart on a geo map
  - Choropleth: creates a choropleth map
- Polar Charts:
  - Scatter polar: creates a scatter plot in polar coordinates
  - Line polar: creates a line chart in polar coordinates
  - Bar polar: creates a bar chart in polar coordinates
- Ternary Charts:
  - Scatter ternary: creates a scatter plot in a ternary coordinate system
  - Line ternary: creates a line chart in a ternary coordinate system

3.8.3 High Level Features of Plotly Express

1. A single entry point into Plotly:It provides an easy entry point into Plotly for creating plots.
2. Sensible, Overridable Defaults: Plotly Express functions have sensible defaults that produce aesthetically pleasing plots. These defaults are fully customizable.
3. Flexible Input Formats:Plotly Express functions accept a variety of input formats, including Pandas dataframes, NumPy arrays, and JSON data.
4. Automatic Figure Labelling:It automatically labels the axes, legend, and title of your figures based on the data and arguments you provide.

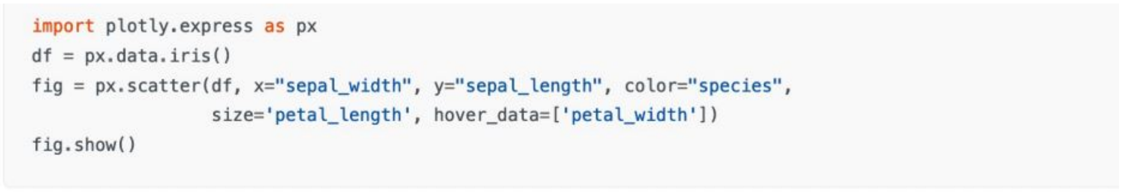
5. Automatic Hover Labels:It automatically generates hover labels, which provide additional information about your data when you hover over a specific point in the plot.

6. A Pandas backend: Plotly Express has a Pandas backend that makes it easy to work with Pandas dataframes and to take advantage of the many features of this popular library.

7. Automatic WebGL switching: When WebGL is not available, Plotly Express automatically switches to a non-WebGL mode, ensuring that you can create and view plots on a wide range of devices and platforms.

Example

Below is a simple example of scatter plot of Iris Dataset. The **px.scatter** function creates the scatter plot with “sepal_width” as the x-axis, “sepal_length” as the y-axis, “species” as the color of the markers, and ’petal_length’ as the size of the markers. The hover_data argument specifies what data to display when you hover over a marker, in this case it is ’petal_width’.



```python
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species",
                 size='petal_length', hover_data=['petal_width'])
fig.show()
```

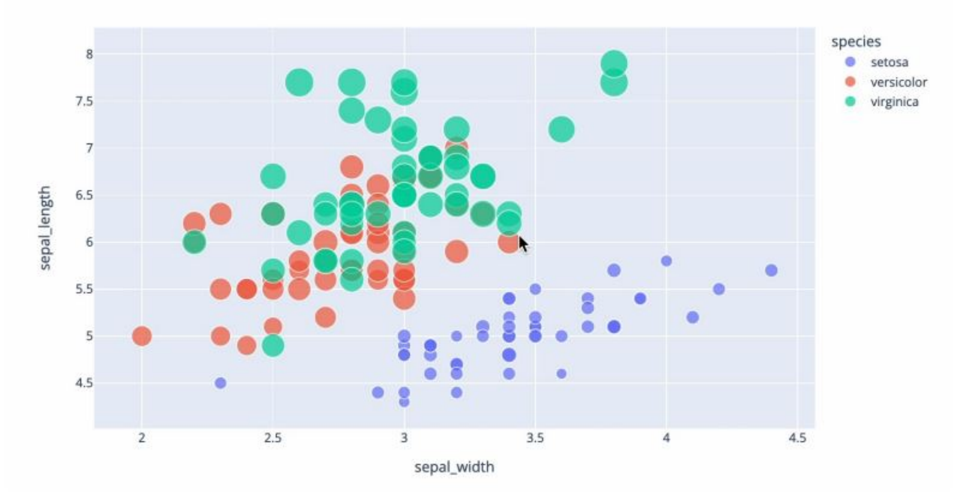


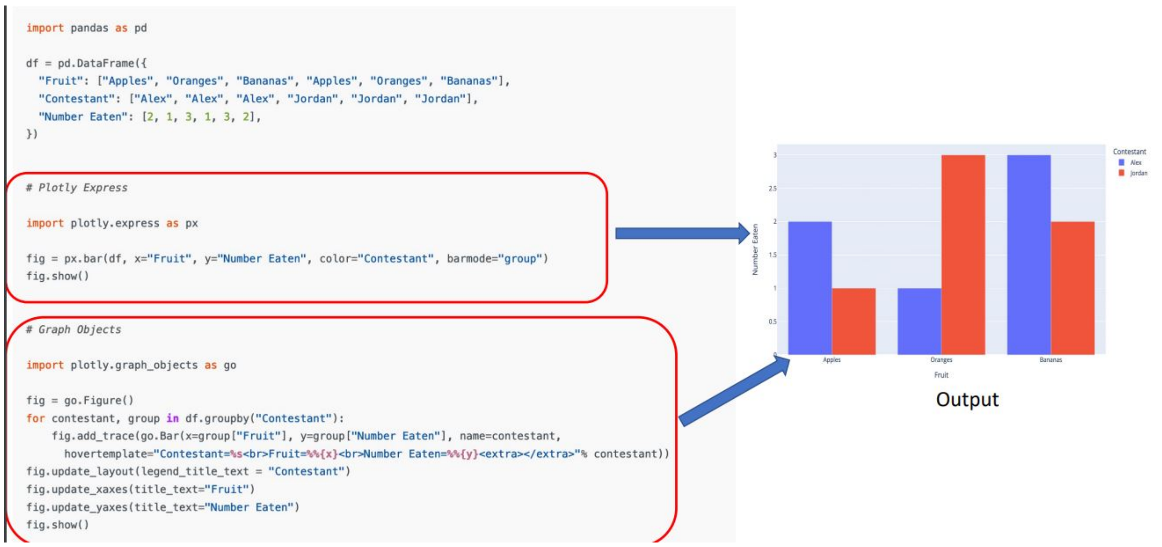
Figure 5: Scatter Plot using Plotly Express Module [1]

3.8.4 Plotly Graph Objects Module

Plotly Graph Objects Module is a Python module. The module provides a set of Python classes that correspond to various parts of a figure, such as the x-axis, y-axis, plot, and annotations. These classes can be used to directly specify the characteristics of the plot, such as the data to display, the axis labels, the markers' size and color, and the hover labels.

The figures created, manipulated and rendered by the Plotly Python library are represented by tree-like data structures that are automatically serialized to JSON for rendering by the Plotly.js JavaScript library. This enables the figures to be rendered and displayed in the browser, allowing for interactive, web-based visualizations.

The Graph objects in the Plotly Graph Objects module can be directly used to plot the data without needing to specify the details of the JSON serialization. So we can create, manipulate, and customize figures without having to worry about the underlying data structures and rendering mechanisms.
Below we can see the comparison between Plotly Express module and Graph Objects Module. In Plotly express we are getting the same result with just few lines of code, while in Graph Object the lines of code are more. With graph object module, we have more customization power but things are more complicated.

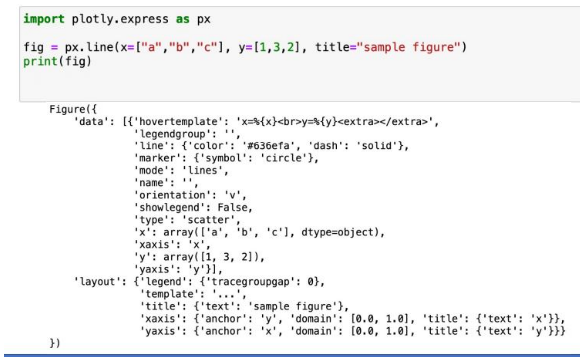


```
import pandas as pd

df = pd.DataFrame({
  "Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],
  "Contestant": ["Alex", "Alex", "Alex", "Jordan", "Jordan", "Jordan"],
  "Number Eaten": [2, 1, 3, 1, 3, 2],
})

# Plotly Express

import plotly.express as px

fig = px.bar(df, x="Fruit", y="Number Eaten", color="Contestant", barmode="group")
fig.show()

# Graph Objects

import plotly.graph_objects as go

fig = go.Figure()
for contestant, group in df.groupby("Contestant"):
    fig.add_trace(go.Bar(x=group["Fruit"], y=group["Number Eaten"], name=contestant,
      hovertemplate="Contestant=%s<br>Fruit=%%{x}<br>Number Eaten=%%{y}<extra></extra>"% contestant))
fig.update_layout(legend_title_text = "Contestant")
fig.update_xaxes(title_text="Fruit")
fig.update_yaxes(title_text="Number Eaten")
fig.show()
```

Figure 6: Comparison of Graph Objects and Express Module [1]

3.8.5 Figure Data Structure in Plotly:

It is very useful Data Structure. We can get all the details of figure. Below is one such example that shows how we can effectively use figure data structure.



```
import plotly.express as px

fig = px.line(x=["a","b","c"], y=[1,3,2], title="sample figure")
print(fig)
```

```
Figure({
    'data': [{'hovertemplate': 'x=%{x}<br>y=%{y}<extra></extra>',
              'legendgroup': '',
              'line': {'color': '#636efa', 'dash': 'solid'},
              'marker': {'symbol': 'circle'},
              'mode': 'lines',
              'name': '',
              'orientation': 'v',
              'showlegend': False,
              'type': 'scatter',
              'x': array(['a', 'b', 'c'], dtype=object),
              'xaxis': 'x',
              'y': array([1, 3, 2]),
              'yaxis': 'y'}],
    'layout': {'legend': {'tracegroupgap': 0},
               'template': '...',
               'title': {'text': 'sample figure'},
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'x'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'y'}}}
})
```

Figure 7: Figure Data Structure [1]

3.9 Bokeh Library

Bokeh: Bokeh is a visualization library for Python that allows users to create interactive, web-based visualizations. It uses HTML and JavaScript as the main rendering technologies, with the help of the BokehJS library in the background. Bokeh also provides useful features for web application development, such as support for embedding visualizations in web pages, server-side rendering, and the ability to connect visualizations to real-time data streams.

3.9.1 Bokeh Features

- Bokeh can seamlessly work with popular Python data tools such as Pandas and Jupyter Notebook, making it easy to work with existing data sources and quickly create visualizations.
- Bokeh plots are highly interactive. Unlike static plots produced by libraries such as Matplotlib and Seaborn, Bokeh plots provide a rich, interactive experience for the user. For example, users can pan and zoom the plot to better focus on specific areas of interest, or hover over points to see their data values.
- Bokeh allows the user to add custom JavaScript code to further extend and customize its functionality.
- Bokeh's ability to integrate with Flask or Django web applications makes it an ideal choice for building complex visual analytics systems. With Bokeh, it's possible to create dynamic and interactive visualizations that can be embedded directly into web pages, and connected to real-time data streams, making it easy to build powerful data analytics applications for a wide range of use cases.

3.9.2 Interfaces to Bokeh

- bokeh.plotting: "bokeh.plotting" is a high-level interface in the Bokeh library for plotting data. It contains the definition of the Bokeh's "Figure" class, which is the main building block for creating Bokeh plots.
  The Figure class allows users to plot vectorized graphics as "glyphs", which are the building blocks of Bokeh plots. Glyphs can include shapes such as lines, circles, rectangles, and other shapes. These shapes can be represented as both raster and vector graphics. Raster graphics are made up of pixels, while vector graphics are defined by mathematical equations and can be resized without losing quality, while Raster can lose quality when resized.

- bokeh.model: The "bokeh.models" interface in the Bokeh library provides low-level, customizable building blocks for creating Bokeh plots and visualizations. This interface offers a collection of classes and functions that allow users to create and customize the components of their plots, such as axes, grids, and annotations. It provides greater control and flexibility compared to the higher-level "bokeh.plotting" interface, making it ideal for advanced and complex visualization projects.

  But It has some demerits also: It generally requires more code compared to the higher-level "bokeh.plotting" interface, as users need to manually specify and customize each component of their plots. This can lead to longer and more complex code compared to using the higher-level interface.

3.9.3 Bokeh: Behind the Scenes

It consists of two main components: the BokehJS JavaScript library and the Bokeh Python library.

- The BokehJS JavaScript library runs in the browser and is responsible for handling the rendering and user interactions of the visualization. It takes a collection of declarative JSON objects as input and uses them as instructions for rendering the various aspects of the visualization. The JSON objects contain information about the plot, such as the data, the visual representation of the data (e.g. lines, circles, etc.), and the behavior of the plot in response to user interactions.
  The BokehJS library converts the JSON objects into BokehJS models and views, which are used to render the plot in the browser. The BokehJS models contain the underlying data and logic of the plot, while the views are responsible for rendering the plot and handling user interactions.

- The Bokeh Python library is responsible for generating the JSON objects that the BokehJS library uses to render the visualization. The Python library provides a high-level interface for creating plots and a low-level interface for more advanced users who want to customize their plots in more detail. At the lowest level, the Python library is built on a set of model classes that are reflected in the BokehJS models created in the browser.

3.9.4 Bokeh Concepts

- BokehJs: BokehJS is the backend JavaScript client library that takes care of rendering the visualizations and handling user interaction. It processes the JSON objects generated by Bokeh Python library to render the visualizations in a browser.

- Glyph: Glyphs are the basic visual building blocks of Bokeh plots. They are API objects that draw vectorized graphics to represent data and include elements such as lines, rectangles, squares, and wedges.

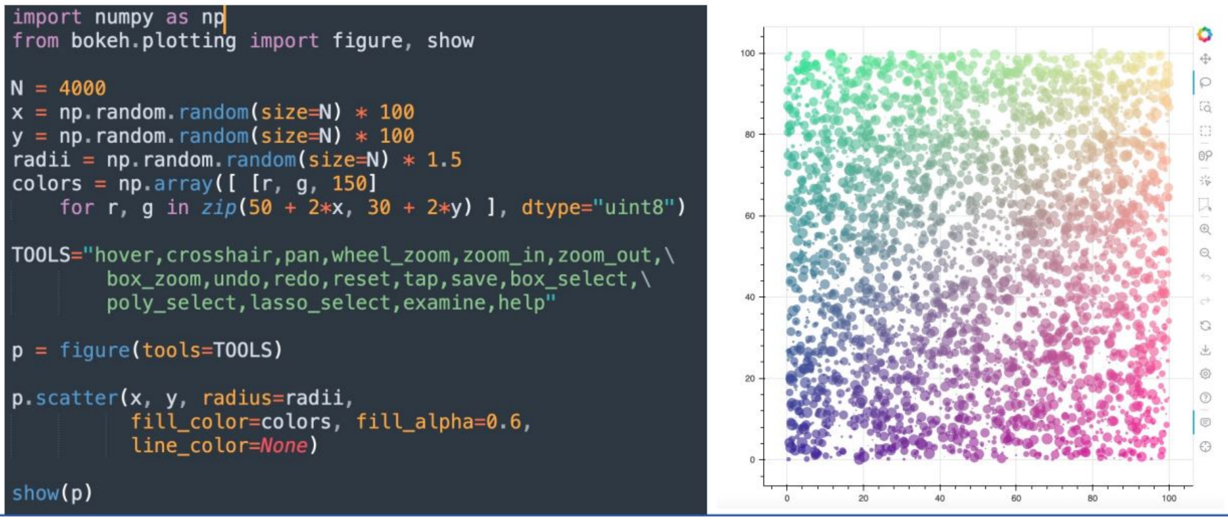
- Models: Models are the lowest-level objects that a Bokeh visualization consists of. They represent the data and properties of visual elements in the plot, such as axis, grids, etc.

- Layout: Layout is a collection of Bokeh objects, including plots and widgets. It provides a way to arrange multiple visual elements on a single page.

- Widgets are user interface elements that are not directly part of a Bokeh plot. Examples include sliders, drop-down menus, and buttons. They can be used to provide input to the Bokeh plot or to update it based on events and data.

3.9.5 Example of Bokeh Plot + Interaction Tools

The code generates and plots a scatter plot of circles with random x and y positions, sizes, and colors. The plot is interactive, and the specified tools can be used to navigate and interact with the plot.



```python
import numpy as np
from bokeh.plotting import figure, show

N = 4000
x = np.random.random(size=N) * 100
y = np.random.random(size=N) * 100
radii = np.random.random(size=N) * 1.5
colors = np.array([ [r, g, 150]
    for r, g in zip(50 + 2*x, 30 + 2*y) ], dtype="uint8")

TOOLS="hover,crosshair,pan,wheel_zoom,zoom_in,zoom_out,\
        box_zoom,undo,redo,reset,tap,save,box_select,\
        poly_select,lasso_select,examine,help"

p = figure(tools=TOOLS)

p.scatter(x, y, radius=radii,
          fill_color=colors, fill_alpha=0.6,
          line_color=None)

show(p)
```

Figure 8: Bokeh Plot Example

References

[1] https://hello.iitk.ac.in/sites/default/files/cs661asem223/5b439d9cfed2d52cae078e345b99df8ee363b2

[2] https://en.wikipedia.org/wiki/data_and_information_visualization.

[3] https://dl.acm.org/doi/pdf/10.1145/2089094.2089101.

[4] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6327286.

[5] https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9149790.

[6] Ganviz: A visual analytics approach to understand the adversarial game. 2018.

[7] https://en.wikipedia.org/wiki/post-wimp.

[8] https://ieeexplore.ieee.org/document/4376144.

[9] https://dl.acm.org/doi/abs/10.1145/1054972.1055031.

[10] https://d3js.org/.