Informatics

Vergleichende Evaluierung von Business Analytics und Visualisierungswerkzeugen und -anwendungen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

BSc Damir Dizdarevic
Matrikelnummer 12141497

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof.in Mag.a rer.soc.oec. Dr.in rer.soc.oec. Silvia Miksch
Mitwirkung: Univ.Lektorin Dipl.-Ing.in Dr.in techn. Johanna Schmidt

Wien, 2026-02-11

Damir Dizdarevic   Silvia Miksch

Informatics

Comparative Evaluation of Business Analytics and Visualization Tools and Applications

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

BSc Damir Dizdarevic
Registration Number 12141497

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof.in Mag.a rer.soc.oec. Dr.in rer.soc.oec. Silvia Miksch
Assistance: Univ.Lektorin Dipl.-Ing.in Dr.in techn. Johanna Schmidt

Vienna, 2026-02-11

Damir Dizdarevic Silvia Miksch

Erklärung zur Verfassung der Arbeit

BSc Damir Dizdarevic

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 2026-02-11

Damir Dizdarevic

Acknowledgements

I would like to thank my secondary supervisor, Univ.Lektorin Dipl.-Ing.in Dr.in techn. Johanna Schmidt, for her continuous support, patience and guidance during my research. Her insightful feedback and encouragement were invaluable.

I would also like to thank my main mentor, Univ.Prof.in Mag.a rer.soc.oec. Dr.in rer.soc.oec. Silvia Miksch, for her time, dedication and constructive criticism, which significantly improved the quality of my work.

Special thanks to the staff of the Vienna University of Technology for providing me with the knowledge and skills that I have applied in this Master's thesis.

I would like to thank my colleagues, some of whom I consider lifelong friends, for the good cooperation and helpful discussions.

Three years ago, I left my home country with the goal of completing my Master's degree in Vienna. This journey has been full of challenges and growth, and I am proud to have achieved the main goal of my departure.

I am very grateful for my family, as their support and encouragement during my time away from home has been a constant source of strength.

During the writing of this thesis, I found out that I will become a father to a beautiful son and that marks the biggest accomplishment and the most important event of my life. Alexander, I will forever be by your side and do everything to give you all the love and support you need and that one day you will proudly say that you have the best father in the world who was always there for you.

Most importantly, I am deeply grateful for my wonderful wife. Her constant support, patience, understanding, and love have been my driving force on this journey. Without her by my side, this achievement would not have been possible and as enjoyable as it is now, and I will cherish all the moments with her for the rest of my life.

Thank you all.

Kurzfassung

In den letzten Jahren ist die visuelle Analytik ein entscheidender Faktor für die datengetriebene Entscheidungsfindung in vielen Branchen geworden. Je mehr Daten zur strategischen Steuerung genutzt werden, desto wichtiger wird es, komplexe Informationen klar zu visualisieren. Eine Vielzahl von Werkzeugen und Bibliotheken, die unterschiedliche Nutzergruppen und Kompetenzniveaus bedienen, ist als Ergebnis dieses Trends entstanden.

Anwendungen mit benutzerfreundlichen Drag-and-Drop-Funktionalitäten wie Microsoft Power BI, Tableau und Excel werden immer beliebter. Ohne dass man programmieren können muss, erlauben sie es, aussagekräftige Visualisierungen zu erstellen; deshalb sind sie im Bereich der Business Intelligence für Anwender ohne technische Ausbildung besonders interessant. Im Gegensatz dazu sind Bibliotheken wie Plotly, GGPlot und D3 für Entwickler gedacht und bieten fortgeschrittene Funktionen für statistische Analysen, komplexe Datenmanipulationen und hochgradige Anpassungen von Visualisierungen.

Frühere Untersuchungen haben Unterschiede zwischen diesen Werkzeugen aufgezeigt, vor allem in Bezug auf die Unterstützung explorativer Analysen im Vergleich zur Ergebnispräsentation. Aber es gibt inzwischen Lösungen, die leistungsfähiger sind und mehr Funktionen bieten, um den neuen Anforderungen der Nutzer gerecht zu werden.

Die vorliegende Untersuchung hat das Ziel, eine vergleichende Analyse von ausgewählten, gängigen visuellen Analytikwerkzeugen zu bieten. Entscheidende Kriterien wie Benutzerfreundlichkeit, Anforderungen an die Datenaufbereitung und weitere relevante Eigenschaften werden bewertet. Ein Clustering-Ansatz, der funktionale Ähnlichkeiten berücksichtigt, wird zur Klassifikation der Werkzeuge angewandt. Um eine ausgewogene und umfassende Betrachtung zu garantieren, schließt die Analyse sowohl innovative, neuere Ansätze als auch etablierte Plattformen ein. Basierend auf den Ergebnissen sollen Empfehlungen für Praktiker und Wissenschaftler formuliert und Forschungsfragen identifiziert werden, die zur Weiterentwicklung von visuellen Analytiktechnologien beitragen – vor allem im Hinblick auf die Herausforderungen durch großskalige Datenmengen.

Abstract

In recent years, Visual Analytics has transformed how we approach data-driven decision-making across many industries. As organizations increasingly turn to data to guide their strategies, being able to clearly visualize complex information has become more important than ever. This rising demand has led to the creation of a wide variety of tools and libraries, each designed to suit different users and skill levels.

Because of their easy drag-and-drop functionality, programs like Microsoft Power BI, Tableau, and Excel are growing in popularity. Since they can generate insightful visuals without any programming, which is gaining a lot of attention in the field of Business Intelligence, these applications are especially helpful for professionals who lack technical skills. In contrast, Plotly, GGPlot and D3 are charting libraries that cater to programmers by offering options for more sophisticated statistical analysis, advanced levels of data manipulation, and even greater customization for sophisticated visualizations.

Prior conducted researches marked disparities among these tools, particularly with respect to how well their design supports data exploration in comparison to data presentation. Nevertheless, and as the field progresses, today's tools for Visual Analytics are more comprehensive and robust than in the past, providing new features and enhancements to keep pace with user demand.

This thesis sets out to carry out a detailed comparative analysis of some of the most widely used visualization tools in the field today. The goal is to evaluate their core strengths, limitations, and overall performance in the context of data visualization. Primary visualization criteria, including usability, data preparation requirements, and other pertinent aspects, will be the main focus of the evaluation. A clustering approach will be used to analyze the features of the tools, classifying them according to similarities in their features. To make sure the overview is thorough and balanced, both more recent, cutting-edge solutions and older, more established platforms will be covered. The goal of this thesis is, through these findings, provide guidance to the users regarding the tools that best address their specific requirements. Additionally, the analysis is anticipated to highlight questions that need further exploration and identify solutions aimed at improving visualization technologies in general, particularly as these technologies face new challenges posed by large-scale data.

Contents

CHAPTER 1

Introduction

Information visualization [1] and Visual Analytics [2] are research domains focused on facilitating the exploration, comprehension, and extraction of insights from data using visual methods. These fields emphasize the design of intuitive visual representations, including charts, graphs, and interactive interfaces, that convert complex datasets into accessible, actionable information. Information visualization and Visual Analytics apply principles from design, perception, and human-computer interaction to develop tools that enable users to detect patterns, identify trends, and support data-driven decision-making [3]. Both fields address the increasing challenges of modern datasets. As datasets increase in size and complexity, traditional analytical methods struggle to keep pace. Visualization and Visual Analytics provide structured ways to represent and interact with this data, helping users identify patterns, outliers, and trends that are difficult to discern through other means.

The significance of information visualization and Visual Analytics resides in their extensive influence. Research on information visualization and Visual Analytics has given the users both theoretical bases and practical methods for exploring data, coming up with hypotheses, and sharing results. Visualization and visual analytics techniques are now essential in various fields.

Application domains include business intelligence, where dashboards and interactive reports facilitate data-driven management and decision-making [4], healthcare and public health [5], where visual analytics aids clinicians and policymakers in monitoring, exploring, and interpreting complex clinical datasets [6], cybersecurity [7], where visual analytics is used to detect attacks, analyze network behaviour, and support incident response, and scientific research, where scientific visualization and interactive visual analysis are crucial for comprehending and conveying experimental and simulation results [8]. These improvements, especially the use of interactive visual interfaces with computational analysis methods, have made it possible to work well with large and complex datasets.

1.1 Motivation and Problem Statement

Research in information visualization and Visual Analytics has led to the design and usage of visualization tools that implement the researched principles and methods in everyday analytical practice. For example, Tableau, Microsoft Power BI, and Looker Studio are business-oriented applications that allows users to combine different types of data, do interactive analyses, and make dashboards and reports that can be shared without having to write any code. On the other hand, libraries made for developers, like D3.js, Plotly, and ggplot2, are very flexible and can be programmed to do custom analysis and advanced visualization tasks.

These visualization tools turn the results of academic research into solutions that both technical and non-technical users can use. To move visualization research forward, understanding how users use these tools is of particular interest. Studying how users interact with visualization tools helps researchers design more effective, intuitive, and impactful solutions in the future, ensuring that the newly developed tools actually meet real-world needs.

Based on this foundation, the thesis aims to compare and analyze existing visualization and Visual Analytics tools and applications, utilizing established evaluation frameworks from the information visualization and Visual Analytics literature. In this way, the thesis aims to build a basis to understand better how modern visualization packages incorporate research findings and how these packages are applied by users to help them analyzing different types of datasets.

In this thesis, we use the term **visualization tools** to refer to both *applications* and *charting libraries*. Applications denote standalone software platforms with graphical interfaces. Examples are Tableau by Salesforce and Microsoft Power BI. Charting libraries are code-based environments for visualization. Examples are Python Plotly, R GGPlot2, and D3.js.

In this thesis, we build on existing approaches to understand the tool landscape in information visualization and Visual Analytics. One very prominent comparative study by Behrisch et al. [9]. The authors systematically examined the feature sets of commercial Visual Analytics platforms. Their work provides a comprehensive taxonomy of functionalities, emphasizing distinctions in visualization types, analytical integration, and user guidance. This study is widely cited; however, it focuses only on commercial software packages and reflects the state of the art in 2018. In a study even earlier, Lisa Charlotte Rost [10] adopted a pragmatic approach to compare different applications and charting libraries. In her study, she recreated the same chart in 24 different visualization tools. Her work compellingly illustrates the practical differences among visualization tools in terms of usability, visual quality, and learning curve. The results of Rost's evaluation are shown in Figure 1.1, which shows that she rated charting libraries as more useful for analysis tasks than applications. Rost's study, however, was not academic in nature and focused on a single type of visualization, without linking her results to a structured, feature-based taxonomy.

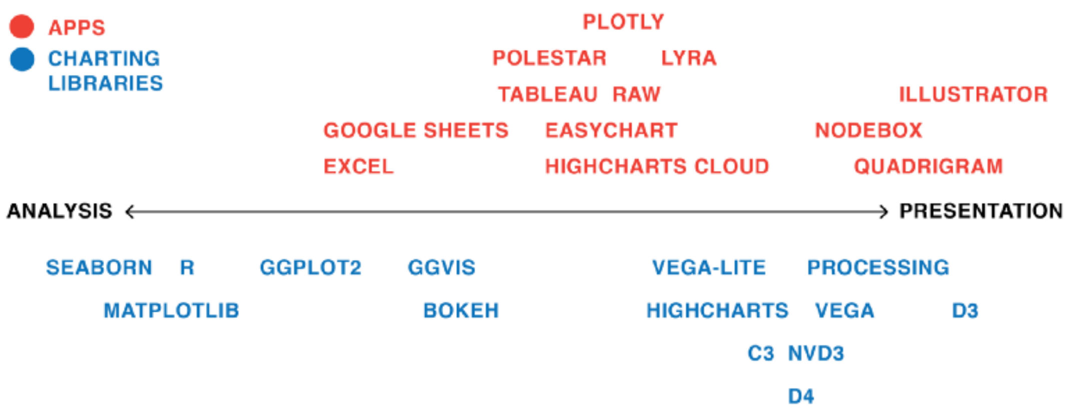


Figure 1.1: The figure shows visualization applications (red) and charting libraries (blue) arranged on a spectrum from analysis-oriented tools on the left to presentation-oriented tools on the right. Tools near the center (e.g., Tableau, Plotly, Vega-Lite) are positioned as hybrids that aim to support both data exploration and polished communication. Image taken from [10].

Summarizing existing approaches, we can rely on a structured feature-based analyses of commercial applications, but they are now out of date and do not include charting libraries. Conversely, the existing practitioner-focused, task-oriented comparisons are deficient in generalization and methodological rigor. Furthermore, no research has yet systematically integrated both commercial applications and charting libraries into a cohesive framework.

As a motivation for this thesis, therefore, we aim to **deliver a thorough evaluation of visualization tools as of 2025 by integrating the structured, feature-level assessment methodology of Behrisch et al. with the pragmatic, task-level reproduction strategy of Rost**. This work aims to provide researchers and practitioners with updated insights into the current status of visualization tools, clarify the application of Visual Analytics methods in practice, and formulate guidelines for the selection of the most suitable tool based on context and user profile.

1.2 Research Questions

Building on the motivation and identified gaps, this thesis aims to investigate the current state of visualization tools by focusing on three central research questions. These questions are designed to clarify what is known, what is missing, and how this work contributes beyond prior studies.

RQ1: What is the current landscape of visualization tools?

While previous studies have compared certain visualization tools, they were either confined to commercial applications or limited to specific case studies. There is still no complete, current survey of visualization tools. This research question is therefore formulated to address this gap by guiding a systematic mapping and description of currently available visualization tools in 2025.

RQ2: What commonalities and differences exist between visualization tools?

Most current evaluations focus on just one tool or a small part of the visualization ecosystem, so they don't give us much information about how tools compare to each other in terms of features and capabilities. This research question is examined through a comparative feature analysis of the chosen tools: their functional capabilities are methodically recorded, and the tools are categorized based on the types of features they offer (for example data preparation, interaction, supported visualization types, or analytical integration). Previous research has identified the advantages and disadvantages of specific tools; however, there is a lack of systematic evidence regarding their performance in standard visualization tasks.

This research question is formulated to address that gap by looking at how different tasks affect tool capabilities and how those capabilities affect real-world results.

RQ3: What evidence-based guidelines can be developed to support users in integrating visualization tools into their analytical workflows?

Users working with data have varying needs and objectives and need to fulfill different tasks. This means that visualization tools and applications will not fit all types of users and tasks.

1.3 Goal and Expected Results

Based on the research questions, we identified four specific goals and results the thesis aims to provide.

G1: General picture of the visualization tool landscape

First, it will give a general picture of the current visualization tool landscape by showing the different features and user support that today's visualization tools offer. This overview addresses the limitations noted in the current literature, which either focus on only a narrow subset of visualization tools or no longer reflect the current state of the visualization tool landscape.

G2: Analysis and comparison of visualization tool features

Second, the thesis will provide a comparative analysis of visualization tools, highlighting similarities and differences between applications and charting libraries. This viewpoint helps us better understand how visualization tools relate to each other within the overall tool ecosystem by pointing out which capabilities of visualization tools are common and which are unique. The aim is to identify patterns of similarity and difference across tools and, on this basis, to derive meaningful groups of visualization tools within the current visualization landscape.

G3: Task-based evaluation of visualization tools

Third, the thesis will assess task-based variations by analyzing the performance of visualization tools in executing standard visualization tasks.

G4: Guidelines for visualization tool usage

Fourth, the anticipated outcome of this goal is a compilation of pragmatic guidelines that assist researchers and practitioners in selecting suitable tools based on their requirements, such as prioritizing ease of use, flexibility, interactivity, or seamless integration into existing workflows.

1.4 Methodological Approach

This thesis utilizes a mixed-methods research design that integrates feature-oriented clustering analysis with task-based tool evaluation. The methodology was based on two fundamental studies: the feature categorization approach by Behrisch et al. [9] and the tool-replication framework by Rost [10]. The thesis proceeded in three distinct phases:

1.4.1 Phase 1: Survey of the Visualization Tool Landscape

The first step was to do a thorough survey of the current state of visualization tools, which included both applications and charting libraries. This involved methodical market research to pinpoint prevalent, newly developed, and professionally relevant tools as of 2025. Rather than evaluating the tools in detail at this stage, the aim was to compile a broadly representative set of candidates. Selection was guided by factors such as accessibility, evidence of active development, documented functionality, and reported use in research or professional analytics workflows. In this phase we made sure that the thesis was based on a representative and up-to-date set of visualization tools.

1.4.2 Phase 2: Feature Mapping and Comparative Analysis

In the second phase, the visualization tools that were found in Phase 1 were looked at and compared based on their functional capabilities. This included figuring out what each tool can do in terms of visualization techniques, interactivity, analytical support, and user guidance. The goal was to find similarities and differences and put visualization tools into groups according to the types of features they offer, in order to reveal patterns of similarity and difference within the overall visualization tool ecosystem and to relate these patterns to the needs of different user groups.

1.4.3 Phase 3: Visualization Reproduction and Grading

This phase addressed the efficacy of visualization tools in facilitating standard visualization tasks, complementing the feature analysis done in Phase 2. The choice of visualization tasks was based on well-known taxonomies from the visualization literature, which emphasize that different types of user goals need different kinds of visualization support.

1.4.4 Phase 4: Formulating Guidelines

In the last phase of the thesis, results from Phase 2 and Phase 3 were combined to create an evidence-based set of guidelines for users, when entering the market of visualization tools, and having to decide on specific applications or charting libraries.

CHAPTER 2

Fundamentals and Related Work

This chapter gives the thesis the theoretical and conceptual background it needs to fit into the fields of Information Visualization and Visual Analytics. These research areas are the basis for understanding how to design, use, and evaluate visualization tools. The chapter begins by introducing information visualization and Visual Analytics. It outlines their history, basic concepts, main ideas, and recent progress. We will also analyze existing evaluations schemes for applications and libraries. In addition, this chapter introduces Principal Component Analysis (PCA), a dimensionality-reduction method that is later used in this thesis to analyse and visualize the feature space of visualization tools.

2.1 Information Visualization and Visual Analytics

Information visualization and Visual Analytics use interactive visual representations to help humans understand complex data and make better decisions. Information visualization focuses on designing visual encodings and interactive views (such as charts, networks, and maps) that turn abstract data into perceivable structures, enabling users to see patterns, outliers, and relationships at a glance. Visual Analytics extends this by tightly integrating interactive visual interfaces with automated analysis methods (including statistics, data mining, and machine learning) to support complex reasoning tasks. Visual Analytics keeps the human 'in the loop,' so people can guide algorithms, validate results, and iteratively refine hypotheses in a mixed-initiative process.

2.1.1 Information Visualization

Information visualization is the discipline focused on the creation and application of interactive visual representations of abstract data to enhance human cognition. Card et al. [11] defined information visualization as *'the use of computer-supported, interactive, visual representations of abstract data to amplify cognition'*. Information visualization is different from scientific visualization because it focuses on non-spatial data, like financial figures, social networks, or text corpora [3]. Scientific visualization focuses on physical or spatial phenomena, like medical imaging or computational fluid dynamics.

The main idea is that when visual representation is combined with interactivity, it lets the users use their perceptual and cognitive strengths to find patterns, spot anomalies, and make sense of things [1]. Information visualization is a way to show insights from data in a visual way. By turning data into graphs, users can see patterns in the data sets that they might not be able to see just by looking at the numbers and statistical representations [12].

William Playfair and Florence Nightingale were the first people to use graphs and charts in the 18th and 19th centuries [13]. When computers came out in the late 20th century, researchers started looking into ways to use computers to explore data interactively. Interactive systems and the first visualization conferences (i.e., IEEE Visualization and IEEE Information Visualization) started to appear and made information visualization its own field of study. Since then, information visualization has become a well-established field, with its own journals and conferences.

Information visualization is based on a number of theoretical and methodological principles. A fundamental principle is graphical perception, as, for example, defined by Cleveland and McGill [14]. Graphical perception empirically illustrated the most effective visual encodings (e.g., position, length, angle, color) for quantitative comparisons. Guidelines for designing charts are still used in nowadays information visualization applications. The grammar of graphics as defined by Wilkinson [15] constitutes a systematic framework for describing and constructing visualizations which provides a structured, composable approach to designing visual representations. The grammar allows to break down visualizations into fundamental components, such as data mappings, scales, and geometric objects. Since it provides consistency and reproducibility, it is still used on modern information visualization libraries (e.g., Vega-Lite [16]). The basic concept of generating a visualization from a dataset is summarized in the information visualization pipeline, which is explained in detail in the next part.

Information visualization pipeline

The information visualization pipeline [17] suggests a series of steps for turning raw data into pictures. It has steps that take in data, change it into visual formats, and show it in ways that are easy for users to understand. The pipeline is outlined in Figure 2.1. This framework has since become a key idea in information visualization, helping to shape the design and use of useful data visualization tools. The information visualization pipeline describes how raw data is transformed into images through a sequence of stages, typically moving from data acquisition and preprocessing to mapping data attributes to visual encodings, and finally rendering interactive visual displays. The pipeline includes steps like importing and cleaning the data, filtering and enriching it as needed, and encoding it into graphical forms (such as positions, shapes, and colors).

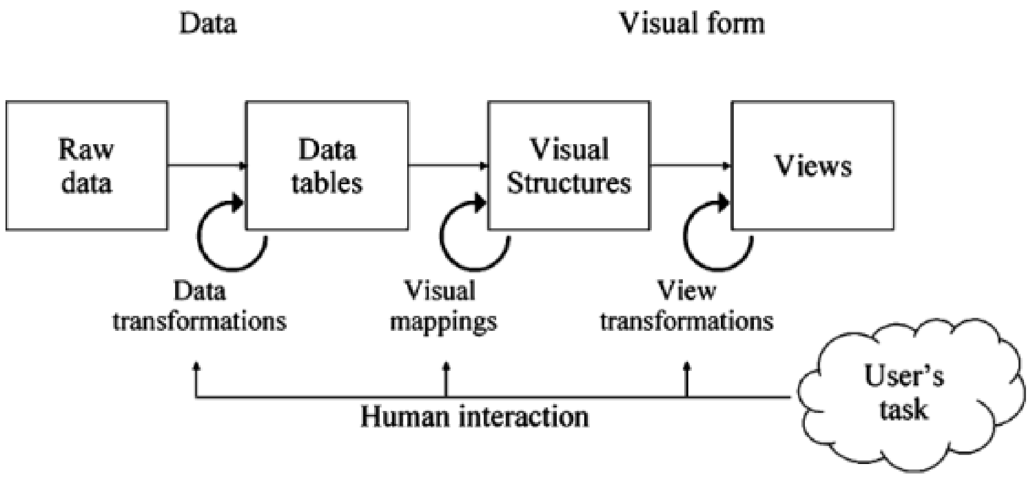


Figure 2.1: The information visualization pipeline. The main steps are turning raw data into structured tables, mapping these tables into abstract visual structures, and finally making views that users can interact with. At different points in the pipeline, users can affect the process by interaction. Image taken from Card et al. [11].

Data Transformation

The first step in the pipeline is data transformation, which means changing the data into a format that is structured and organized so that it can be visualized. This step is usually meant to both cut down on and improve the information in raw data. According to Card et al. [11], there are four types of data transformations.

First, transformations can use mathematics to get new values from old ones, like finding the mean or sum. Most of the time, these operations are used to make the information in raw values better. Second, data transformations can also change the way the data is structured, like putting it into tables that can be used to compare or group things. This step is more about changing how data is displayed and making it easier to see.

Third, we can use the structures we made to make new derived values. This could mean taking out groups or ranges from the well-organized data structures. These transformations take structured data and change it into higher-level information that cannot be seen in the raw data or the values that come from it. Finally, transformations can make new structures from old ones. These new structures can show us the data in a different way or bring out different relationships. For example, a table of ranges could be turned it into a binary table. In this case, existing ranges are used to set the rules for binarization. Each cell in the binary table shows whether a specific data point falls within a certain range or category.

All data transformation changes try to make sure that the data is in a so-called *idiosyncratic format*. This means that they are set up to work best in the next stages of the visualization pipeline.

Visual Mapping

The main part of the visualization process is visual mapping. At this point, a specific function F is used to map the derived structures and values in data tables to visual glyphs. During this process, raw data is mapped to visual elements such as points, lines, or colors. The mapping is based on defined rules, enabling users to perceive patterns, relationships, and structures that would otherwise remain hidden.

This function F takes structured data as input and gives back a picture of it as output. The goal is to make a visual form that is easy for users to understand. A well-designed visual mapping function must be able to be computed, which means it can be run by an algorithm. It also needs to be able to be inverted, so that users can get the data back from the visual representation. Also, users should be able to understand the function or its inverse, and the visual representation should try to make it as easy as possible for them to understand.

View Transformation

After the visual forms are made, they are put into views. Views show these visual forms on the screen and let users change the perspective by zooming, panning, and rotating them. View transformations allows users to see different parts of the data and concentrate on specific details. This process is in accordance with the visual information seeking mantra by Shneiderman [18], which suggest to always visualize data as *overview first, zoom and filter, then details on demand.*

Human Interaction

Interaction is an important part of the visualization pipeline because it lets users change the visualization (i.e., apply view transformations) and to find meaningful patterns by data exploration [19]. Selecting, filtering, linking, and rearranging or remapping are all common ways to interact.

Selecting When a user selects something, he/she marks certain data entities or subsets so that he/she can see more information or do more analysis.

Filtering Filtering makes the display show less data, which lets users focus on the information that matters to them.

Dynamic queries With dynamic queries, users can change the parameters of a query and see the filtered results right away.

Linking Linking connects information from different views to show users how choices in one view relate to data in another.

Rearranging and remapping Rearranging and remapping gives users the ability to change the visual mapping or pick different mappings to look at different parts of the data.

Techniques and Categories

Different types of data need different ways of visualizing them to get the point across. Information visualization techniques describe different views, or chart types, that can be employed to display data in a visual way. Picking the right charts and visual mappings for display data is a very important and complex decision in information visualization [12]. There are many ways to group different types of visualization techniques and categories. The most common ways to make distinctions are by:

- Problem - This method groups visualizations by the type of problem they are meant to solve, like communication, exploration, or confirmation.
- Type of Data - We can group visualizations by the kind of data they work with, like categorical, numerical, or time-series data.
- Dimensions - This tells us how many dimensions (univariate, bivariate, multivariate) the data visualization shows.
- Structure of the data: Visualizations can be grouped by the way the data is structured (linear, hierarchical, or circular).
- Interaction: This group divides visualizations into groups based on how interactive they are, like static, interactive, or dynamic visualizations.

In the following part, we summarize and illustrates some of the most prominent and well-known information visualization techniques, which are available in most of the available applications and charting libraries.

Heatmaps

A heatmap is a way to show two-dimensional data by using color to encode numbers in a matrix of rows and columns. The color intensity of each cell shows how big the value it represents is. This makes it a great way to find patterns, trends, and outliers in large datasets. The modern heatmap was first made in computer graphics in the 1990s. It came from shaded data matrices that were used as early as the 19th century [20]. Users like heatmaps because they can quickly show high-level information by using color gradients to highlight "hotspots" and "coldspots", as it is shown in Figure 2.2.

Improvements to standard heatmaps have made them more useful in both academic and practical settings. Clustered heatmaps use hierarchical clustering on both rows and columns to help us find patterns in the data, like groups of similar observations or variables. In bioinformatics, heatmaps are still the best way to see gene expression data. Clusters can show genes that are co-regulated or functional pathways that are shared. Gu [22], for example, identified heatmaps as a proper way to find similar patterns that are shared by groups of rows and columns in complicated datasets. Heatmaps are also a well-known tool for identifying patterns in the data, like missing items [23].

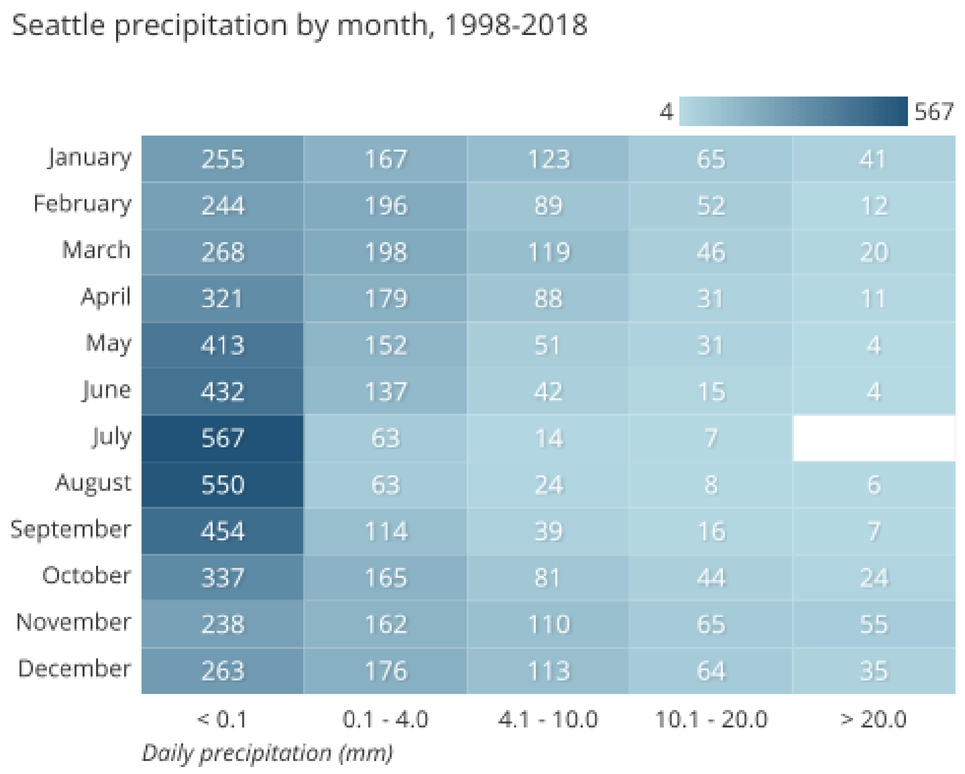


Figure 2.2: Representation of the daily precipitation in Seattle by Heatmap, darker colors indicate periods with higher precipitation. Image taken from Atlassian [21].

Dense-pixel Visualizations

Dense-pixel visualization, also called pixel-oriented or dense pixel displays, is a powerful way to show very large, multidimensional datasets. Dense-pixel based representations map each individual data value to a single colored pixel arranged in a compact layout, allowing very large multivariate datasets to be shown at once so that global patterns and correlations become visible, even though individual values are not easily readable. This visualization technique makes the most of the display space and handles datasets with hundreds of thousands or even millions of values without losing detail or overlapping, as we can observe on the Figure 2.3.

Pixels are usually arranged in a systematic way, either independently of the data (query-independent) or dynamically based on similarity metrics (query-dependent). They use color encoding to show patterns, clusters, or outliers at a glance. Keim [24] formalized the pixel-oriented approach, examining how design choices-such as pixel arrangement strategies and color mapping-can be framed as optimization problems intended to enhance effective visual data exploration.

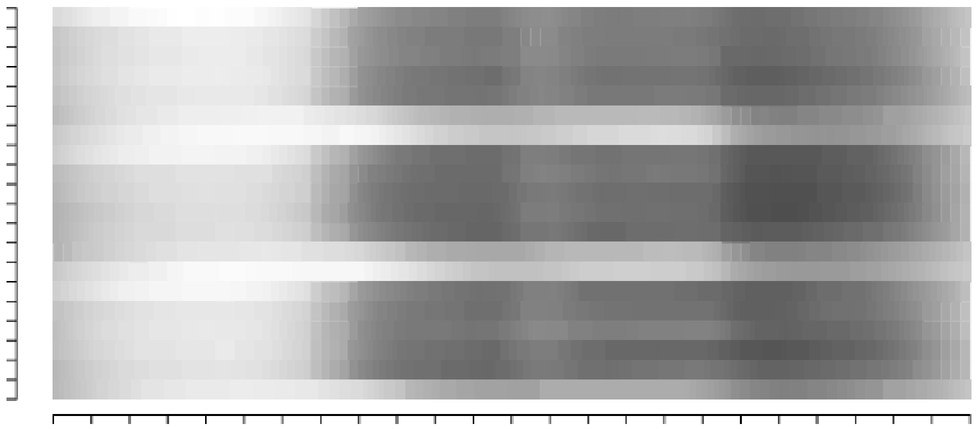


Figure 2.3: In this example of Dense-pixel based visualization, one value is shown each 15 minutes, producing a 96×20 pixmap. Image taken from Rodrigues and Gama [25].

Glyph-based Small Multiples

Glyph-based small multiples visualizations use glyphs, which are small visual objects that hold a lot of information, and the small multiples layout principle, which says that each glyph should be in its own cell within a consistent grid. This method lets viewers look at a lot of data points at once, making it easier to visually compare multivariate patterns without making things too complicated. Each glyph independently communicates various dimensions via visual variables including shape, color, size, or orientation, while the consistent grid layout-small multiples-guarantees visual stability and comparability throughout the dataset.

Academically, glyph-based small multiples have been examined and enhanced in various visualization fields. Borgo et al. [26] gave an overview of glyph-based visualization, including basic ideas, design rules, and application methods. Their summary shows how glyphs can effectively and meaningfully represent multivariate data. Recent endeavors in visualization design have investigated generative methodologies. Brehmer et al. [27] came up with the Diatoms technique, which automates the creation of glyph designs by taking samples from palettes of shapes, encoding channels, and spatial arrangements. This method shows these options using small multiples or a small permutables gallery. This lets designers look at several glyph designs at once and pick the best one based on how it looks and how creative they are.

Horizon Graphs

Horizon graphs are a space-saving way to show time series data. They do this by cutting a regular area chart into horizontal bands and stacking them on top of each other, often using color gradients to show intensity. This layered approach lets the graph take up much less vertical space while still showing trends and extremes visually. This lets users compare up to fifty or more time series in one view without too much clutter. An example if shown in Figure 2.4.

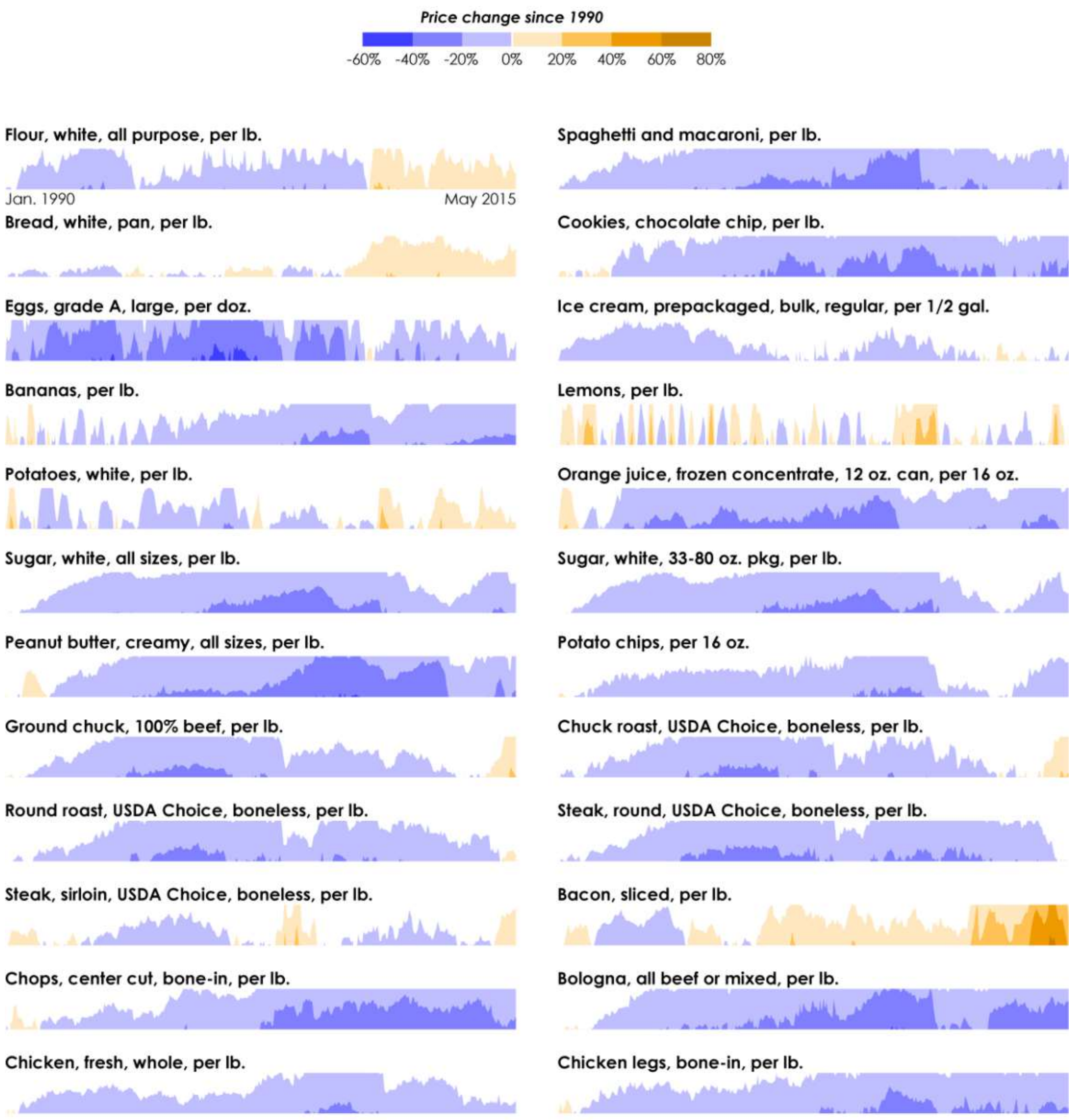


Figure 2.4: Visualization shows the percentage change in price for select food items, since 1990. Estimates are from the Bureau of Labor Statistics and adjusted for inflation. Image taken from FlowingData [28].

Dahnert et al. [29] looked at different types of collapsed horizon graphs, which improve horizontal resolution and readability when we need to look at multiple trends in space (like in geographic contexts). They discovered that standard horizon graphs are great for highlighting extreme values, but collapsed versions are better for spotting trends when looking at adjacent sequences. More recently, Braun et al. [30] introduced the order of magnitude horizon graph, tailored for time-series data with wide value ranges. Their study shows that this variation matches or outperforms traditional horizon graphs in tasks such as identification, discrimination, and estimation-and even exceeds them in certain contexts.

Parallel Coordinate Plots

One of the most well-known ways to show multivariate data is with parallel coordinates plots. Inselberg [31] first introduced parallel coordinates plots, and they became more popular in real-world situations [32]. Each variable is shown as a parallel vertical axis. Polylines that cross these axes at their corresponding values are used to draw individual data records. This visual encoding lets the users show relationships in more than two dimensions in a two-dimensional space. A Highcharts demo of a parallel coordinates plot is shown in Figure 2.5.

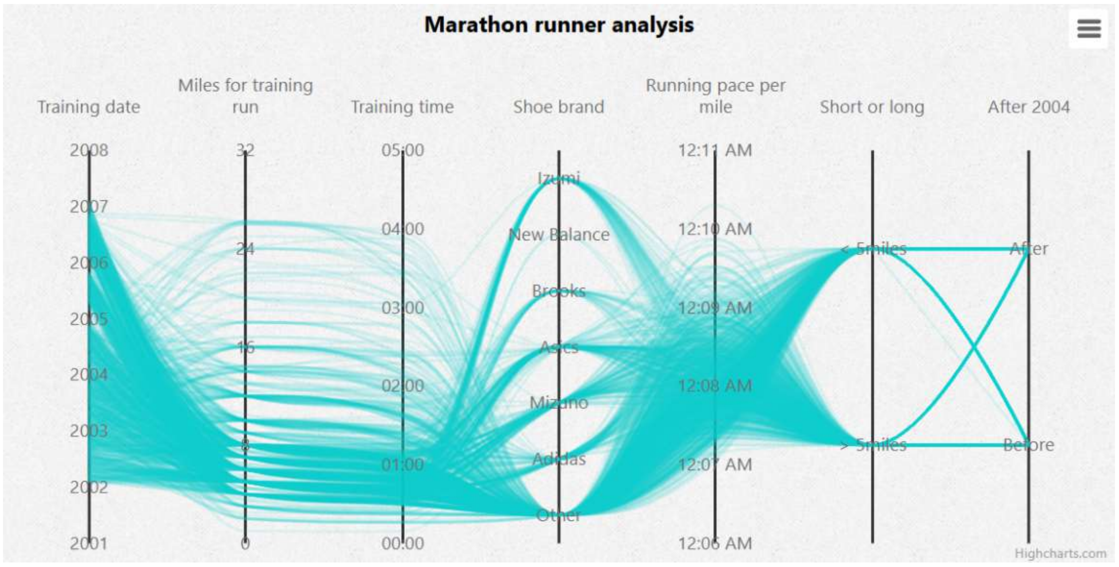


Figure 2.5: Chart showing an example of a parallel coordinate plot. This type of visualization is commonly used to show multivariate data, and can help analyze large datasets to find patterns and correlations. Image taken from Highcharts [33]

.

The main advantage of parallel coordinates plots is that they can show how different variables are related to each other. When polylines cross axes at similar points and stay almost parallel, it suggests a positive correlation. On the other hand, sharp X-shaped crossings usually mean a negative correlation. For instance, in the Iris dataset, coloring by species shows that the observations group into separate bundles of lines, which makes it possible to see class separation.

The main drawback of parallel coordinate plots is visual clutter, which results merely due to lines being drawn on the screen. Visual clutter can greatly reduce the effectiveness of this technique on large data, for which researchers tried to find clutter reduction techniques. One example is clustering and optimized edge arrangement [34], or edge bundling [35].

Sunburst Charts

A sunburst chart is a way to show hierarchical data using concentric circles. The chart visualizes hierarchical data spanning outwards radially from root to leaves. The hierarchy's root is in the middle, and each level extends outward in rings. The angle and color of each segment shows how much that category is worth compared to its siblings.

This radial layout makes it easy to see multiple levels of nested categories in a small, easy-to-understand structure. This is especially helpful for showing complicated relationships more clearly than traditional pie charts or decision trees, as also visualised in Figure 2.6.

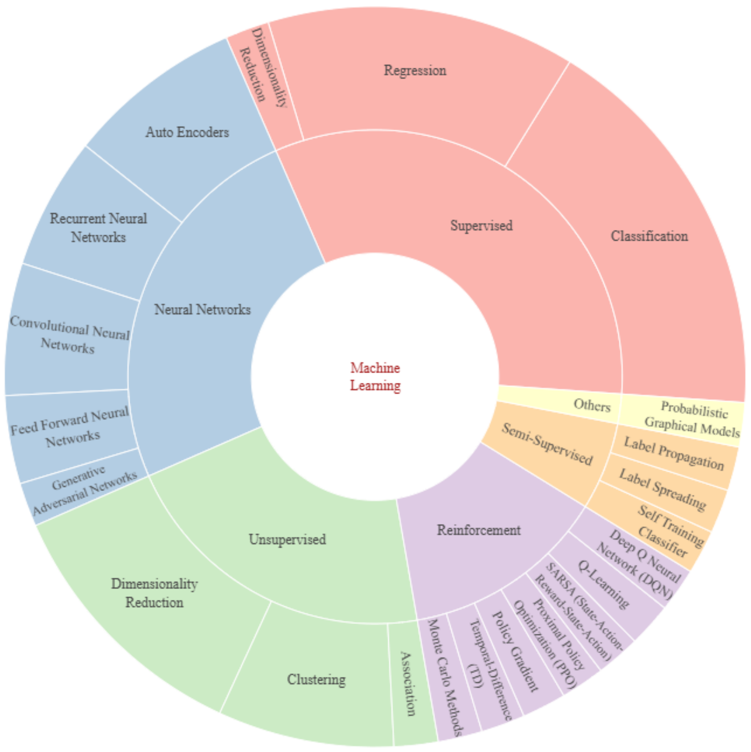


Figure 2.6: Sunburst chart showing hierarchical data in concentric circles. This chart represents categories better than standard pie chart or decision tree. Image taken from Dang [36].

Sunburst charts have been utilized in both empirical studies and practical implementations. For example, Taylor et al. [37] looked at sunburst and icicle visualizations in the eMouse Atlas of Gene Expression (EMAGE) platform. Their research demonstrated that the sunburst, due to its space-efficient radial configuration, offered a comprehensive overview of gene families or pathways and facilitated users in navigating and comparing associated gene expression data. In a comparison of hierarchical visualization techniques, such as treemaps, icicle plots, and sunburst charts, users generally preferred icicle plots for clarity. However, sunburst charts were still useful, especially when participants were already somewhat familiar with the layout, as demonstrated by Woodburn et al. [38]. Recently, Rastogi et al. [39] presented a method to automatically extract data from sunburst charts. The algorithm includes a chart classification, a component extraction, and a hierarchical data organization module.

Treemaps

A treemap is a way to fill space with a series of nested rectangles that show hierarchical information. The area of each rectangle is proportional to a quantitative attribute of the data it shows, as depicted in Figure 2.7. Treemaps were invented by Johnson and Shneiderman [40].

A treemap makes ideal use of display space by mapping hierarchical relationships into two-dimensional areas that are next to each other and take up less space. The way the visuals are arranged makes it easy for viewers to see complicated structures at a glance. The use of color coding also makes it possible to encode a second variable, whether it is categorical or quantitative, at the same time, which makes it easier to find patterns in more than one dimension.

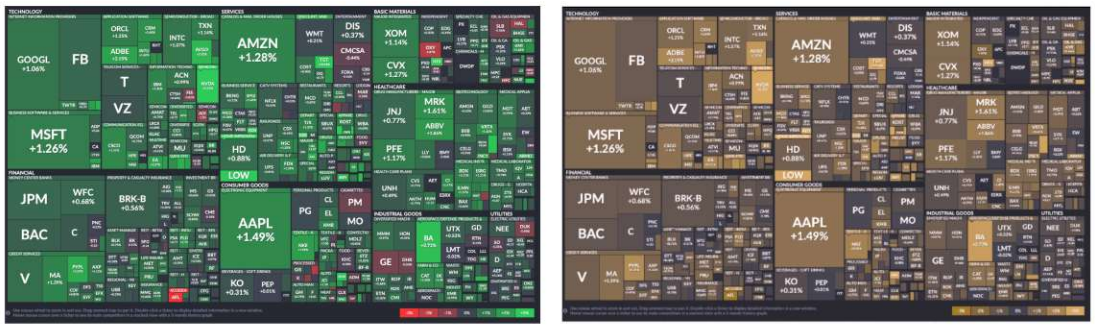


Figure 2.7: An example of the Treemap, visualizing the US stock market. Colors indicate stock market prices. The two images shown here are the same. On the left side, the original colors are shown. On the right side, a representation of the colors as seen by a colorblind person is depicted. Image taken from Laubheimer [41].

Treemaps have undergone extensive research and refinement in the academic realm. Pang et al. [42] performed a user study comparing treemaps and map-like hierarchical visualizations. They found that nested treemaps were faster for completing tasks, but in some cases, other options were more accurate and easier to use. The results were later confirmed by Firat et al. [43], who also concluded that We the properties of treemaps can hinder data visualization literacy and cognition. Increasingly, treemaps were also used for time-varying data. Tu and Shen [44] could show that treemaps work with hierarchical data that changes over time. Later, Vernier et al. [45] did a quantitative comparison of time-dependent treemaps, looking at several algorithms in terms of visual quality and temporal stability. This helped professionals choose the best treemap algorithms for datasets that change over time.

Node-link Diagrams

A node-link diagram, which is also called a network graph, is a basic way to show how things are connected. In these diagrams, the points that make up each entity are called nodes or vertices, and the lines that connect them are called links or edges. These kinds of representations are very easy to understand because they make connections between things like social networks, biological interactions, or system dependencies clear, which can be observed in Figure 2.8. Arrowheads, line thickness, or color are often used to encode extra information, like direction, weight, or type of connection [46].

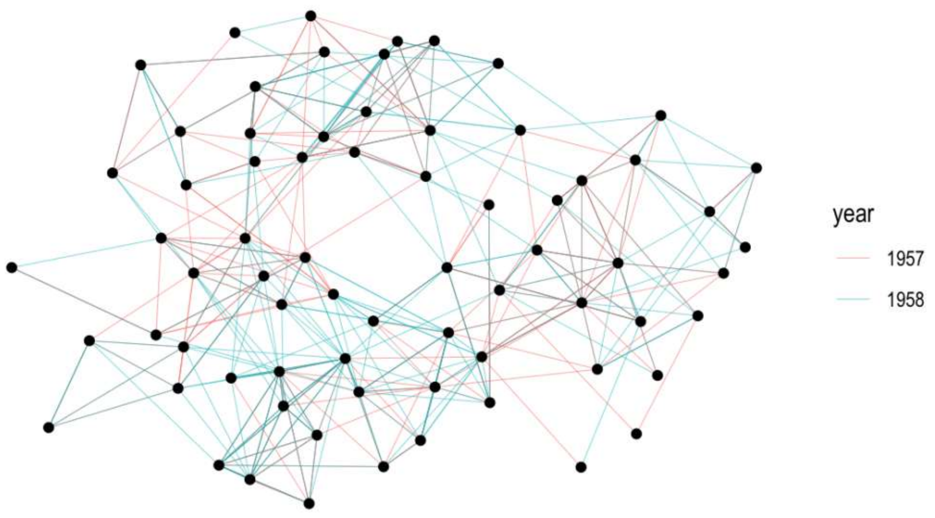


Figure 2.8: Representation of a node-link diagram with nodes represented as circles and edges represented as lines. Image taken from Sankaran [47].

Matrix-based Techniques

Matrix-based techniques show data in a grid or table format, with rows and columns usually representing different entities. The cells that cross each other show how these entities are related or what their values are, often using color. In network visualization, the adjacency matrix is a common type of matrix. Cell shading or color shows the presence, weight, or direction of connections between nodes. Users prefer these visualizations because they are small and can work with big datasets [48]. This is especially true when we look at other diagrams, which can get messy as more connections are made [49].

Academic research validates the effectiveness of matrix representations in various analytical tasks. For example, Abdelaal et al. [50] conducted a crowdsourced evaluation comparing node-link diagrams, adjacency matrices, and bipartite layouts. They found that adjacency matrices were usually the most reliable for tasks like finding clusters and estimating density in large or directed networks.

Geographic Maps

Geographic map-based visualizations use the natural spatial dimension of data to give the users easy-to-understand information about patterns that depend on location. Choropleth maps, which color-code regions based on data values, and proportional-symbol maps, which use the size of symbols to show magnitude, are two common types of these visualizations. Flow maps show how things move between areas, while heat or dot-density maps show how strong the distribution is. Cartograms change the size of geographic areas so that regions look proportional to a certain variable (like population). This shows spatial differences in a more data-driven way. These kinds of methods work well for making decisions, especially when the insight is based on geospatial context. For example, a mixed-methods study showed that health domain experts prefer map-based visualizations for analyzing data related to vitality [51]. This shows how maps can help with spatial reasoning and expert decision-making.

Fung et al. [52] performed an experimental study examining the impact of legends and grid lines on users' accuracy, confidence, and speed in interpreting contiguous area cartograms. Their findings showed that while these aids made estimates more consistent, they also added bias and made it harder to make decisions. This showed that there are important design trade-offs to consider when showing geographic maps. The field of geovisualization also looks at interactive and analytical geospatial techniques. It looks at how dynamic mapping, layering, and visual mining of spatial data can help the users learn new things and make decisions about space [53].

2.1.2 Visual Analytics

Visual Analytics is a cross-disciplinary area of study that combines interactive visualization, computational analysis, and human reasoning to help the users understand and explore complex data. Visual Analytics is defined as *'the science of analytical reasoning facilitated by interactive visual interfaces'* [2]. Visual Analytics builds on the methods and techniques from information visualization by adding interactive access for users to statistical methods to interactive systems. Heer and Shneidermann [54] specifically named interactivity and exploration as key parts of supporting the sense-making process with visualization. The idea of a *'Human in the Loop'* [55] is what explains why interactivity works. The Visual Analytics process is depicted in Figure 2.9. The need for human involvement emphasizes how important human judgment and knowledge are to the analytical process. Human-in-the-Loop systems let users and the system talk to each other all the time. Users improve their data analysis and visualization over time as they learn more about the data they have.

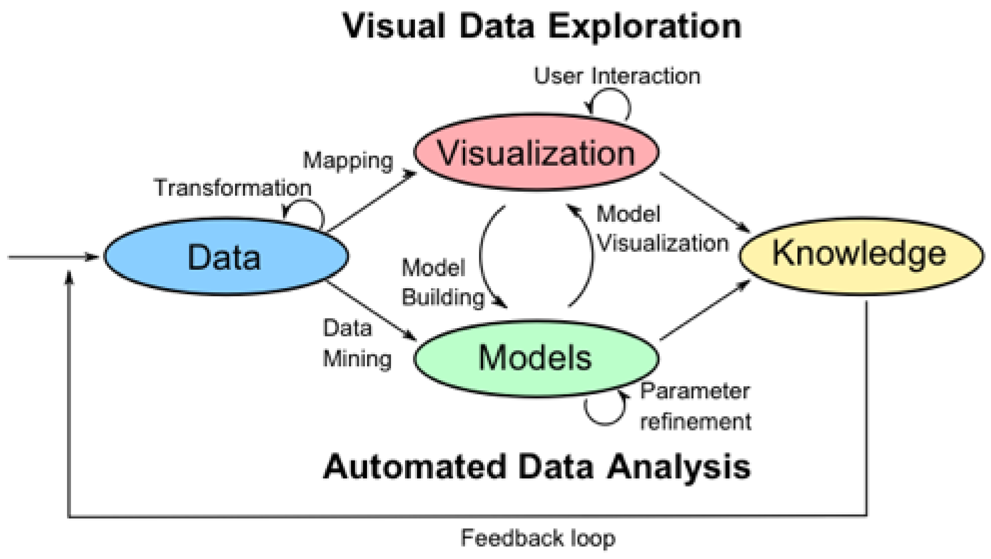


Figure 2.9: The Visual Analytics workflow. The model depicts how Visual Analytics combines automatic and visualization methods with a tight coupling, keeping the human in the loop. Image taken from Keim et al. [2].

In many application scenarios, heterogeneous data sources need to be integrated before visual or automatic analysis methods can be applied. Therefore, the first step is often to preprocess and transform the data to derive different representations for further exploration (as indicated by the Transformation arrow in the figure). Other typical preprocessing tasks include data cleaning, normalization, grouping, or integration of heterogeneous data sources.

After the transformation, the analyst may choose between applying visual or automatic analysis methods. If an automated analysis is used first, data mining methods are applied to generate models of the original data. Once a model is created the analyst has to evaluate and refine the models, which can best be done by interacting with the data. Visualizations allow the analysts to interact with the automatic methods by modifying parameters or selecting other analysis algorithms. Model visualization can then be used to evaluate the findings of the generated models. Alternating between visual and automatic methods is characteristic for the Visual Analytics process and leads to a continuous refinement and verification of preliminary results. Misleading results in an intermediate step can thus be discovered at an early stage, leading to better results and a higher confidence.

If a visual data exploration is performed first, the user has to confirm the generated hypotheses by an automated analysis. User interaction with the visualization is needed to reveal insightful information, for instance by zooming in on different data areas or by considering different visual views on the data. Findings in the visualizations can be used to steer model building in the automatic analysis. In summary, in the Visual Analytics Process knowledge can be gained from visualization, automatic analysis, as well as the preceding interactions between visualizations, models, and the human analysts.

As such, Visual Analytics combines the power of computers with the unique skills of the users in terms of perception, interpretation, and domain expertise. Algorithms create summaries, patterns, or reductions in dimensionality, while the users guide the exploration, make sense of unclear results, and improve hypotheses. Techniques like clustering, anomaly detection, and dimensionality reduction (like PCA, t-SNE, or UMAP) are often built into Visual Analytics systems to show patterns in high-dimensional data and make visual representations that are easy to understand [56]. The analysis process is inherently iterative and exploratory, frequently characterized by sensemaking cycles wherein analysts formulate, evaluate, and enhance hypotheses utilizing external visual representations [57]. Visual Analytics systems often use coordinated multiple views, which connect different visualizations so that users can look at multivariate or temporal-spatial data from different angles [58]. There has been a growing interest in guidance mechanisms in Visual Analytics systems lately [59]. Adaptive guidance seeks to assist users in intricate analyses by suggesting viable subsequent actions or identifying possible analytical oversights, while maintaining the analyst's control and autonomy.

Visual analytics has gained prominence in many different domains. For example, in healthcare [5], Visual Analytics methods support users to understand complex, time-varying data. In climate modeling [60], Visual Analytics methods are applied to understand complex model difference. For network security [61], Visual Analytics methods leverage network forensic and analysis tasks.

2.1.3 Current Challenges

Research in information visualization and Visual Analytics has expanded in multiple directions, not only advancing theoretical and technical foundations but also broadening its practical applications across diverse domains such as science, business, healthcare, and public policy. Information visualization and Visual Analytics research are facing several different challenges nowadays.

One important goal has been **scalability**, which addresses the need to make techniques and methods useable for vast amounts of data. Techniques like dimensionality reduction (t-SNE, UMAP, etc.) [62] have become very important for making large-scale interactive exploration possible [63]. Researchers have suggested distributed computation, streaming architectures, and progressive visualization techniques [64] to keep interactivity as the amount, dimensionality, and speed of data have all grown.

Storytelling and communication are other areas where progress has been made. Hullman and Diakopoulos [65] looked at how to use narrative structures in visualizations to get their points across clearly. This line of work has had an impact on journalism and education, where visualizations are used to explore and to persuade and explain. For example, Schuster et al. [66] showed how climate change can be communicated by interactive visualizations. Collaborative dashboards and multi-user environments help teams make sense of information that is spread out [67].

Explainable artificial intelligence is another active area of research [68]. Visual Analytics offers interactive visual tools that help the users understand and trust the results of machine learning models. Systems in this domain enable analysts to investigate model behavior, evaluate feature significance, or create counterfactual scenarios [69]. Information visualization research has changed a lot in the last few years because of its connection to machine learning. Generative AI models have proven to be very usefule for generating visualizations and performing exploratory analysis, which made researchers think about a shift how users will be using charts in the future [70].

Alongside these advancements, researchers have examined immersive and collaborative analytics. Improvements in **virtual and augmented reality** make it possible to fully explore high-dimensional data [71]. Researchers have looked into using virtual reality and augmented reality environments for exploring multidimensional data [72].

2.2 Evaluating Visualization Tools

Research in information visualization and Visual Analytics has led to many modern visualization tools being developed, which are now used in everyday analysis. To better understand the field of visualization tools, a substantial body of research has compared visualization tools. Many of these studies concentrated on business-oriented analytics applications or on specific use cases. The most important feature-level assessment of applications is the in-depth review by Behrisch et al. [9]. Their research methodically enumerated the functionalities of leading commercial platforms across various dimensions, including supported visualization methods, guided analytics, data transformation, and collaboration. Its main strengths are its methodological rigor and the fact that it covers a wide range of business applications, which makes it easy to see the differences between features and design trade-offs. The authors also came up with a classification of the inspected toolkits, and a categorization of the user types the tools might be useful for. The analysis is shown in Figure 2.10.

Dogadina and Voronin [73] compared Tableau and Power BI to D3.js and Matplotlib, focusing on the trade-off between usability and expressiveness. Applications tend to be faster to get insights and work together, while libraries gives the users more control, programmable workflows, and the ability to add new features. This comparison is useful for showing differences between categories, but the tools are few and the evaluation dimensions are not connected to a bigger picture, which makes it hard to draw conclusions about tools that are not in the examples or to compare newer tools to each other.

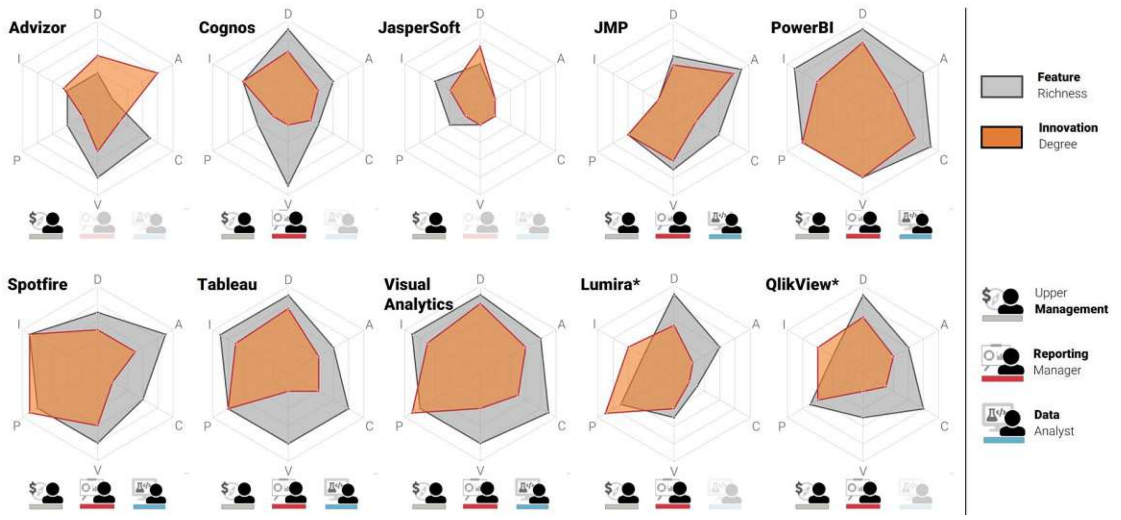


Figure 2.10: Commercial Visual Analytics applications can be categorized according to features like feature-richness, range of data types that are supported, supported visualization techniques, and performance. In addition, it can be shown that not all tools are useful for the same groups of users. For example, managers are interested in other parts of the data (i.e., more aggregated views) than users working directly with the data. Image taken from Behrisch et al. [9].

Parthe [74] compared the two visualization tools Tableau and Power BI for the usage of different Business Intelligence use cases. Similarily, Lousa et al. [75] compared the four applications Tableau, Microsoft Power BI, Sisense, and QlikView, to see how they fit into Business Intelligence workflows. Sabahath et al. [76] took a close look on the role of data visualization in business applications, identifying different types of visualizations that are currently used in business contexts.

Liu [77] analyzed in more detail, how the applications Tableau and Power BI can be employed for the time-consuming steps of data cleaning and data wrangling.

Domain-specific assessments offer supplementary evidence regarding tool appropriateness within actual constraints. Huang et al. [78] assessed dashboarding tools in healthcare, emphasizing usability, analytical integration, and data connectivity within clinical dashboards. Research of this nature reveals essential practical considerations-data governance, interoperability with domain systems, and workflow integration-that are vital in applied settings. Their findings are challenging to generalize to the broader context of visualization tools or to developer-focused libraries utilized beyond the healthcare sector.

Schmidt [79] explored how innovations from information visualization research are integrated into existing visualization toolkits, and could confirm earlier studies, that identified a gap between which techniques have already been researched and what is actually applied by the users [80].

In addition to these academic evaluations, comparisons made by practitioners highlight experiential differences that formal taxonomies might overlook. Rost [10] made a single line chart that showed effort, defaults, visual quality, and problems that the users had when using 24 different tools, including applications and libraries. The strength of this work is that it has a clear, tightly controlled reproduction task that shows where there are problems, where the users need to learn, and where design choices are subtle.

The literature collectively presents three principal insights and three corresponding deficiencies. First, feature-level surveys of commercial applications [9] offer comprehensive taxonomies but are now obsolete and do not include charting libraries. Second, cross-category comparisons [73] show the trade-off between applications and libraries, but they do not cover a wide range of tools or evaluations. Third, practitioner studies [10] reveal real-world friction via task reproduction but are methodologically constrained in their scope and generalizability.

2.3 Principal component analysis

In many fields, big datasets are becoming more common. To make sense of these kinds of datasets, analysts need to find ways to drastically cut down on their dimensionality while still keeping most of the information in the data. Principal component analysis (PCA) is one of the oldest and most popular methods for dimensionality reduction [81]. The idea behind it is simple: lower the number of dimensions in a dataset while keeping as much "variability" (statistical information) as possible. It is a statistical technique that has been used and sometimes reinvented in many different fields. Most of its development has been done by statisticians.

"Preserving as much variability as possible" means finding new variables that are linear functions of the ones in the original dataset, that successively maximize variance, and that are not related to each other. Finding these new variables, called principal components (PCs), is the same as solving an eigenvalue/eigenvector problem. Pearson [82] and Hotelling [83] were the first ones to introduce PCA. It was not until electronic computers became widely available decades later that it was possible to use it on datasets that were not too small. Since then, its use has grown rapidly, and many different versions have been made in many different fields. Since then, it has become very popular, and many different versions have been made in many different fields [84].

PCA can use either the covariance matrix or the correlation matrix as its basis. In either scenario, the new variables (the PCs) are contingent upon the dataset, rather than being predetermined basis functions, thus rendering them adaptive in a general sense. PCA is mostly used for descriptive purposes, not inferential ones. For inferential purposes, a multivariate normal (Gaussian) distribution of the dataset is typically presumed. However, PCA, as a descriptive tool, does not require distributional assumptions and is therefore a highly adaptive exploratory method applicable to various types of numerical data. Many adaptations of the fundamental methodology for various data types and structures have indeed been created.

PCA is a widely used and flexible tool for descriptive data analysis. It can also be adapted in many ways to work with a wide range of data types and situations in many fields. There have been suggestions to change PCA for binary data, ordinal data, compositional data, discrete data, symbolic data, or data with a special structure, like time series. The application to data with common covariance matrices is described by Flurry [85] and Hallin et al. [86]. PCA and PCA-related techniques have significantly influenced various statistical methodologies, including linear regression (notably principal component regression) and the concurrent clustering of both individuals and variables. Correspondence analysis, canonical correlation analysis, and linear discriminant analysis may only be loosely related to PCA, but they all use the same basic idea of breaking down certain matrices into factors. There is a lot of literature on PCA, and it covers a lot of ground. New adaptations, methodological outcomes, and applications continue to emerge.

CHAPTER 3

Methodology

The methodological approach of this thesis was divided into three phases. Structured market research and a survey based on the literature were used in Phase 1 to find and describe the current state of visualization tools. In Phase 2, a comparative feature analysis was done on the chosen tools. This involved writing down their functional capabilities and grouping them by the types of features they support. In Phase 3, each tool was used to reproduce a set of representative visualization tasks and was evaluated using a grading system, in order to assess how well the resulting visualizations support the communication of analytical findings. In Phase 4, guidelines for the usage of the visualization tools have been defined.

3.1 Phase 1: Survey of the Visualization Tool Landscape

The initial phase of the methodology involved executing a comprehensive survey of the existing visualization tool landscape. The survey was conducted as structured market research, encompassing both academic and professional spheres, to ascertain the currently pertinent tools in practice. By 2025, the ecosystem of visualization tools has grown a lot, with new tools coming out and big changes made to old ones.

After putting together a long list of visualization tools from academic sources, industry reports, and practitioner articles, a set of selection criteria was used to cut down the number of candidates. Some of these criteria were ease of use, how well they fit with the research goals, how well they worked in the past, whether they are open-source or require payment, and how well they were used in professional settings. Also, visualization tools were checked to see if they were still being developed, if they worked with modern data formats, and if they could be used in the workflows that users with programming experience and non-technical users use most of the time. The goal was to find a balance between established tools that still set the standard for best practices and new tools that offer new ways of doing things or make existing tools easier to use.

3.1.1 Tool Identification and Compilation

The first step in this thesis was to make a complete list of currently available visualizing tools. To do this, a wide range of academic papers and articles was reviewed, such as the article from Lu [87], the study by Behrisch et al. [9], industry reviews such as the one from Pyramid Analytics [88], and publicly available tool repositories.

The Gartner Magic QuadrantTM for Analytics and Business Intelligence Platforms [89] was a useful source for putting together the application list. Gartner Inc. puts out this report every year and is *'a tool that provides a graphical competitive positioning of technology providers to help you make smart investment decisions'* [89]. Gartner sorts visualization tools into four groups - *Leaders*, *Challengers*, *Visionaries*, and *Niche Players* - based on how well they can carry out their plans and how clear their vision is.

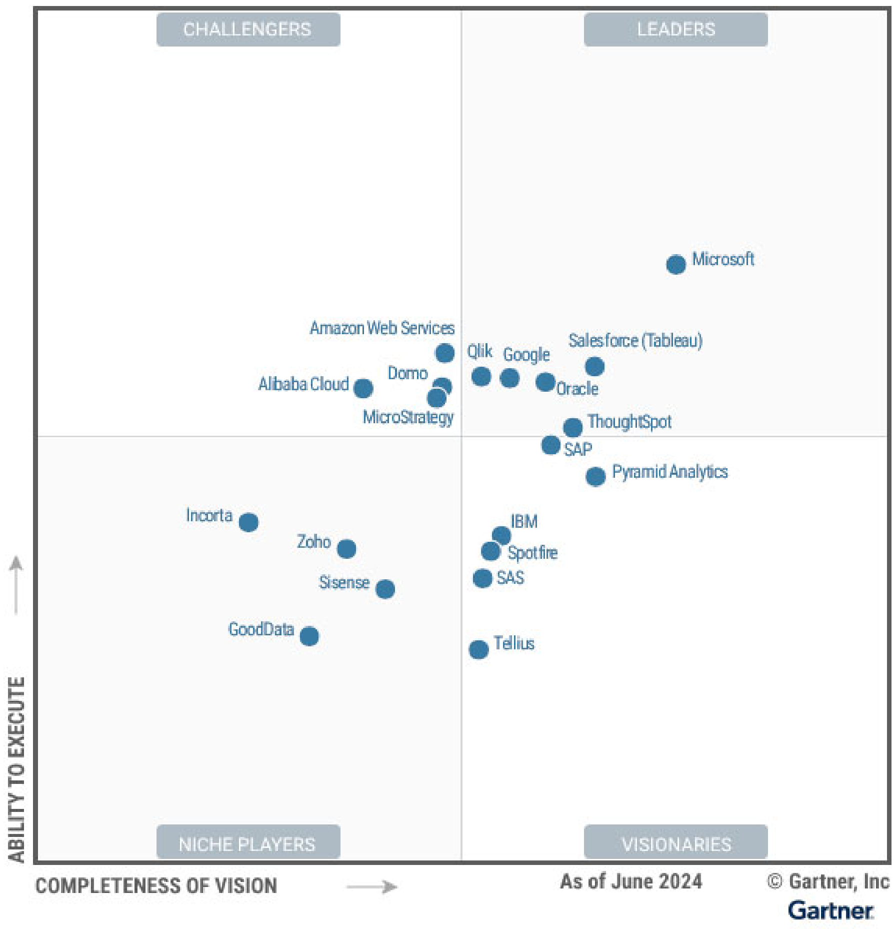


Figure 3.1: The 2024 Gartner Magic Quadrant for Analytics and Business Intelligence Platforms divides current market solutions into four groups. Most interesting are the *Leaders*, since these tools are considered the market leaders in the field. Image taken from Gartner [89].

Vendors who are called Leaders are those who can deliver their goods and services well and have a clear, long-term plan for how the market will change. Challengers are also good at getting things done, but might not have the same long-term vision as leaders. Visionaries, come up with new ways of doing things and disruptive ideas. Niche Players are businesses that focus on serving specific market segments or offering specialized features. The thesis used the 2024 version of the Magic Quadrant to find the top players, challengers, niche players, and visionaries in the field of BI and analytics, as can be seen in Figure 3.1.

The result of the survey and research was a long list of 56 visualization tools, which were divided into four groups: *Charting libraries (used by Rost)*, *New Charting Libraries*, *Applications (used by Rost)*, and *New Applications*. In this case, the reference Rost refers to the study by Rost [10]. The applications and their categorization is shown in Table 3.1. This longlist shows all the tools that were looked at at the start of the study.

Table 3.1: Initial longlist of visualization tools considered in this thesis, grouped into charting libraries and applications and indicating whether they were already used by Rost [10] or only considered in this work.

Charting Libraries (Rost)	**New Charting Libraries**	**Applications (Rost)**	**New Applications**
Seaborn	Pygal	Highcharts Cloud	Looker
Matplotlib	Streamlit	Lyra	QlikView / QlikSense
R	ZingChart	Quadrigram	Power BI
ggplot2	FusionCharts	EasyChart	Sisense
ggvis	ECharts	Plotly	GoodData
Bokeh	ApexCharts	Polestar	Infogram
Vega-lite	TauCharts	Raw	Flourish
Processing	ChartJS	Nodebox	iNZight
Highcharts	TOAST UI Charts	Illustrator (expensive)	Amazon QuickSight
Vega / Altair	Nivo	Tableau	ThoughtSpot
D3	HoloViews	Google Sheets	Axiis
D4	PyGWalker	Excel	BIRT
C3	PyCharts		ZOHO Analytics
NVD3	Plotnine		Chartio
	Leather		Looker Studio
			Domo Charts

3.1.2 Screening Criteria and Tool Exclusion

After the long list was created, it was filtered using a set of practical and strategic criteria. Filtering the long list of applications was important to focus the evaluation on a manageable, representative subset of tools.

We reduced the long list to make sure the remaining visualization tools can be properly analyzed in the upcoming steps. We selected certain criteria based on which we filtered the long list. The excluded visualization tools and the reason why they have been excluded can be found in Table 3.2 and Table 3.3. Visualization tools have been filtered out according to the following criteria:

Discontinued *Visualization tools that are no longer available or have been combined.* Some visualization tools are no longer available in their original form. For example, Atlassian bought Chartio in 2021 [90], and it stopped being a separate platform.

Experimental *Experimental, outdated and out-of-scope visualization tools.* Visualization tools like Lyra, Polestar, and Quadrigram were left out because they are still being tested, do not have current support, and do not have a lot of documentation. These visualization tools were useful for academic research, but they were not good enough for comparing things at a modern production level.

Scope *Redundancy and scope.* To cut down on redundancy, we left out visualization tools that had similar features to more stable or better-supported ones in some cases. For instance, older charting libraries like C3 and NVD3 were left out in favor of newer ones like Plotly and Vega-Lite.

Budget *Financial constraints.* Since this was an academic project, the budget was limited and some visualization tools were not possible to purchase. Amazon QuickSight and QlikView/QlikSense are two examples. They have strong business features, but they are too expensive for non-commercial research.

In summary, the visualization tools chosen for this thesis went through a set of filters to make sure that the sample accurately represents the current state of visualization. Visualization tools were considered if they were actively maintained in 2024–2025, had a measurable user base or market relevance, and provided a sufficient degree of documentation and accessibility for evaluation. These filters helped the thesis focus on visualization tools that are both useful in real life and show different ways of doing things. However, it is important to remember that the selection cannot cover all the different types of visualization technologies that are out there. The goal was to put together a diverse but manageable set that gives useful information about the state of the field.

Table 3.2: Applications excluded after screening, with the primary reason for exclusion.

Tool	**Primary reason for exclusion**
Amazon QuickSight	BUDGET (no suitable free/academic tier)
QlikView / Qlik Sense	BUDGET (no suitable free/academic tier)
Chartio	DISCONTINUED (integrated by Atlassian)
Lyra	EXPERIMENTAL (limited support & documentation)
Polestar	EXPERIMENTAL (limited support & documentation)
Quadrigram	EXPERIMENTAL (limited support & documentation)
Highcharts Cloud	SCOPE (superseded by other applications)
EasyChart	SCOPE (narrow in scope)
Raw	SCOPE (out of scope)
Nodebox	SCOPE (out of scope)
Illustrator	BUDGET (no suitable free/academic tier)
Google Sheets	SCOPE (superseded by other applications)
Looker	SCOPE (superseded by other applications)
Infogram	SCOPE (narrow in scope)
iNZight	SCOPE (narrow in scope)
Axiis	EXPERIMENTAL (limited support & documentation)
BIRT	SCOPE (narrow in scope)

3.2 Phase 2: Feature Mapping and Comparative Analysis

In the second part of our approach, we conducted a structured comparison of the visualization tools found in the short list. Phase 1 gave a general picture of the ecosystem, and in Phase 2, we looked at each visualization tool in more detail, focusing on its functional and visual features.

3.2.1 Feature Mapping

Feature mapping comprised looking at what kinds of visualization techniques were supported, how interactive the visualization tool were, how much analytical support was available (like filtering, aggregation, and statistical methods), and how much help the user received during the design process. By looking at these different aspects, we were able to show what visualization tools can do, and how they help the users do things in real life.

Table 3.3: Charting libraries excluded after screening, with the primary reason for exclusion.

Tool Primary reason for exclusion	
R	Scope (superseded by other applications)
Processing	Scope (superseded by other applications)
Highcharts	Budget (no suitable free/academic tier)
Vega / Altair	Scope (superseded by other applications)
D4	Scope (superseded by other applications)
Streamlit	Scope (superseded by other applications)
ZingChart	Scope (superseded by other applications)
FusionCharts	Scope (superseded by other applications)
ECharts	Scope (superseded by other applications)
TauCharts	Scope (superseded by other applications)
TOAST UI Charts	Scope (superseded by other applications)
Nivo	Scope (superseded by other applications)
PyGWalker	Scope (superseded by other applications)
PyCharts	Scope (superseded by other applications)
Plotnine	Scope (superseded by other applications)
Leather	Scope (superseded by other applications)

To make the comparison more systematic, the extracted features were put into groups based on their function, such as visualization support, interaction and customization, data preparation and transformation, and analytical integration. This grouping allowed us to find similarities and differences between tools at the level of feature groups instead of just at the level of individual options. Some tools, for example, had a lot of built-in support for quickly making standard charts and dashboards with little setup. Other tools focused on flexibility and expressiveness by letting users control encodings, layouts, and interaction logic in great detail using scripting or configuration languages.

Features were gathered into a structured format that could be used for clustering. We grouped the features into three feature categories: *visualization support* (e.g. availability of specific chart types), *automatic analysis* (e.g. built-in support for statistical or machine-learning operations), and *user guidance, perception, and cognition* (e.g. onboarding, recommendations, documentation, and LLM-based assistance). These groups are adapted from the categories proposed by Behrisch et al. [9], and grouped into three feature groups.

The first group, *Visualization*, looks at the different types of charts and visual techniques that a visualization tool can use, from simple bar charts to more complex ones like treemaps or parallel coordinates. The second group, *Automatic Analysis*, looks at how far visualization tools go beyond static charts by adding built-in analytical features like trend detection, statistical modeling, or AI-assisted insights. The third group, *User Perception, Guidance, and Cognition*, is about how visualization tools help users make sense of data by being interactive, giving suggestions, guiding workflows, or having features that make things easier for the brain. 10 visualization types, seven automatic analysis tasks, and 12 items for user guidance, perception, and cognition were used to organize the comparison within these three groups. For instance, in the category of visualization support, the criteria include the ability to support standard and non-standard chart types. In automatic analysis, the criteria look at whether statistical operations, forecasting, or finding anomalies are possible. Interactive filtering and automated chart recommendations are some of the features that are used to judge user guidance, perception, and cognition.

We used the official documentation of the visualization tools to get accurate information on each of these criteria. When necessary, we also did hands-on testing. The visualization tool names were kept in a separate column for later use, but they were not used as inputs for the clustering process. This was done to make sure that the grouping process was only based on measurable features and not on categorical labels. The references to the online documentations are as follows:

- Tableau: `https://help.tableau.com/current/pro/desktop/en-us/gettingstarted_overview.htm`
- PowerBI: `https://learn.microsoft.com/en-us/power-bi/`
- Looker Studio: `https://cloud.google.com/looker/docs/visualization-types`
- Excel: `https://support.microsoft.com/en-us/office/available-chart-types-in-office-a6187218-807e-4103-9e0a-27cdb19afb90`
- GoodData: `https://www.gooddata.com/docs/cloud/create-visualizations/visualization-types/`
- Flourish: `https://flourish.studio/examples/`
- ZOHO Analytics: `https://www.zoho.com/analytics/help/chart/chart-types.html`
- Domo Chart: `https://www.domo.com/learn/charts`
- Sisense: `https://docs.sisense.com/main/Default.htm`
- ThoughtSpot: `https://www.thoughtspot.com/data-trends/data-visualization/types-of-charts-graphs`

- Seaborn: `https://seaborn.pydata.org/`
- Matplotlib: `https://matplotlib.org/stable/index.html`
- Plotly: `https://plotly.com/python/`
- Bokeh: `https://docs.bokeh.org/en/latest/`
- Pygal: `https://www.pygal.org/en/3.0.0/documentation/index.htm l`
- HoloViews: `https://holoviews.org/user_guide/`
- ggplot2: `https://ggplot2.tidyverse.org/reference/index.html`
- ggvis: `https://ggvis.rstudio.com/ggvis-basics.html`
- Vega-Lite: `https://vega.github.io/vega-lite/docs/`
- D3: `https://devdocs.io/d3/`
- ChartJS: `https://www.chartjs.org/docs/latest/getting-started/`
- ApexCharts `https://apexcharts.com/docs/installation/`

3.2.2 Comparative Analysis

For comparing the visualization tools, we applied clustering to identify commonalities between visualization tools. We employed K-Means clustering [91], which is a commonly used and fast clustering approach. For K-Means, it is necessary to know the number of clusters beforehand, as this number defined the number of seed points. To assess the number of clusters, we used the Elbow Method [92].

For this, we ran K-Means for values of k from 1 to 10 and plotted the within-cluster sum of squares (WCSS) against k. The moment when more clusters only slightly lowered WCSS showed that $k = 2$ was the best choice. With this value set, K-Means was run with $k = 2$ using the k-means++ initialization method [93] to get stable cluster assignments. The dataset was then updated with the new cluster assignments.

The clustering analysis was mostly done with numbers, but Principal Component Analysis (PCA) [81] was also used to cut down on the number of dimensions in the feature space, making it possible to see clusters in two dimensions. This visual representation did not affect the results of the clustering, but it did make it easier to look at and understand the groupings.

3.3 Phase 3: Visualization Reproduction and Grading

In addition to the feature-based comparative analysis in Phase 2, we evaluated the visualization tools based on common visualization tasks. The aim of this evaluation step was to analyze how functionality correlates with analytical efficacy.

3.3.1 Visualization Task Types Identification

Visualization researchers made several attempts to categorize visualization tasks. Amar et al. [94] extended this perspective by identifying a taxonomy of low-level analytic tasks, such as retrieving values, finding extrema, filtering, and correlating variables, which visualizations should support. Brehmer and Munzner [95] proposed a why–how–what framework, connecting user goals (why), analytical actions (how), and data characteristics (what). Munzner [3] provides a structured framework for making decisions about visualization design, such as data abstraction, encoding channels, and interaction techniques.

In this thesis, we decided to base our task analysis on the task categorization proposed by Andrienko and Andrienko [96]. It provides a systematic coverage of tasks, starting from abstract data models and defining tasks by target information (i.e., what is unknown) and constraints (i.e., what is known). The categorization covers a fairly exhaustive space of exploratory tasks. Andrienko and Andrienko classify user activities into *Elementary Tasks* and *Synoptic Tasks*. This thesis picks five tasks from each task group that are typical of how users interact with data visualizations in a wide range of ways.

Elementary Tasks

- Lookup – Identifying a specific value (e.g., "What is the price of stock X on day Y?"). Evaluating tools on this task demonstrates how well they support straightforward and detail-oriented information access. In real life, this could mean looking at the current sales number for a product or the number of the users who went to the hospital on a certain day. Tables or key performance indicators (KPIs) are great for this kind of work because they focus on exact, item-level values instead of patterns.

- Comparison – Contrasting two or more values (e.g., "Compare stock prices between yesterday and today."). Grouped bar charts are a canonical way to facilitate such comparisons, since they allow side-by-side evaluation of magnitudes across categories. For instance, a business analyst might look at how sales are doing in different regions every three months. Tools that do well here should make it easy to visually align categories, offer the right scales, and let users interact with the comparison to make it better.

- Relation-seeking – Finding elements matching a condition (e.g., "Find countries where exports exceed imports."). Scatterplots are the most common way to show these kinds of relationships because they show linear, nonlinear, or clustered relationships between two quantitative dimensions. For example, we could look at whether spending more on marketing leads to more sales. The evaluation here shows how tools help with flexible axis selection, visual encoding (like color and size for a third variable), and interaction features like brushing and zooming, which are very important for relational analysis.

Synoptic Tasks

- Pattern Search – Detecting temporal or spatial trends (e.g., "Find intervals of increasing prices."). A 100 percent line chart, in which categories are normalized to proportions, lets users see how the distribution changes over time. For example, a researcher in public health might keep track of how the rates of different diseases change over time. Adding a reference line, like a target threshold, makes it even easier to find values that are different from what the users expect. This task checks how well tools can do normalization, annotation, and time-series interaction.

- Behavior Comparison – Contrasting patterns across groups (e.g., "Compare immigration trends across countries."). Geographic maps work best when the data has a spatial dimension, which lets analysts compare behaviors in different areas. For instance, mapping unemployment rates by country shows how they differ by location. Because of this, tools must support the integration of geographic data, flexible mapping projections, and visual encodings like choropleths and proportional symbols. This task tests whether tools lets the users make meaningful spatial comparisons and how well they prepare and show geospatial data.

The five different tasks were aligned with the dataset selected for this thesis. By basing each task type on analytical questions that are specific to the dataset, the evaluation stayed both theoretically sound and practically useful. We made sure that the tasks chosen are based on real-world situations and not just abstract examples.

3.3.2 Dataset and Task Mapping

To execute the tasks, we relied on a dataset that is large enough to challenge the visualization tools and to allow for a large range of tasks. A dataset for this thesis was especially defined and created. The tasks identified in Section 3.3.1 were then mapped to data questions with relation to the dataset.

Dataset

Two resources were used for this dataset. First, a dataset from the NBA (National Basketball Association) official website [97] comprising player bios was used. Second, a dataset from *Sports Reference* for basketball data was used [98]. Overall, we gathered data for the last 27 seasons - from the season 1996-97 up until the 2023-24 season. The final dataset we used in the thesis was created by joining the two different data sources. The names of the players were the key for joining of the tables. Although it worked in many cases, there were some players who had different namings in the datasets (for example, Robert Williams [97] was also named Robert Williams III [98]). In those cases, manual cleanup and change of the name in one of the datasets was required. A Python script was generated to join all the tables and create a unique data source with most comprehensive data on the NBA players bios and statistics. The dataset has been made available online on Kaggle [1].

The final dataset included 13, 391 records with 36 columns in total. There are no missing values, and every column is fully populated. Player information data like age, player height, player weight, country of origin, attended college, draft round and number were included [97], while all the statistics and advanced metrics were collected as well [98].

The players' weight and height were initially entered into the Imperial system, which was one of the dataset's problems. As a result, there were about 20 unique values for height and 150 unique values for weight, which is a very small quantity. This lack of variation in the data limited the granularity and possible insights that could be obtained from the study, which was troublesome, especially for a dataset with over 13, 000 players. In order to solve this, a technique was put into place that maintains the integrity of the data distribution by allocating randomized height and weight values within reasonable limits. Players were given a random height figure that fell within a realistic range that matched their height as reported in the Imperial system (for example, a player who was described as 6′7" to 6′8" would have a height between 200.66 cm and 203.2 cm). Furthermore, the majority of the data stayed grouped around realistic values because 80 percent of the randomized values fell within a smaller band (5% to 35% of the range). The dataset's average height increased by less than 1 cm as a result of this change, which is statistically insignificant. A similar method was used for weight, creating random weights within ± 0.22 kg of the initial amount based on the difference between two nearby pounds (0.44 kg). The average change in player weight across the dataset as a result of this modification was roughly 0.09 kg, which is likewise statistically insignificant.

Compared to the initial values, this approach significantly increases the dataset's granularity, even though it might not accurately capture the players' real physical measurements.

Task Mapping

We assigned all 5 task defined in Section 3.3.1 a related data analysis question and a visualization which was related to the final dataset created in Section 3.3.2. The data analysis questions were selected based on the task descriptions. The respective visualizations were assigned based on existing knowledge on data representation [99]. By basing each task type on analytical questions that are specific to the dataset, the evaluation stays both theoretically sound and practically useful. This makes sure that the tasks chosen are based on real-world situations and not just abstract examples. The tasks, corresponding data analysis question, and assigned visualization are shown in Table 3.4.

The three Elementary tasks focused on specific aspects of the dataset. For the Lookup task, a KPI-style table is perfect because it gives direct numerical output that makes it

[1] https://www.kaggle.com/datasets/damirdizdarevic/nba-dataset-eda-and-ml-compatible/data

Table 3.4: Task mapping. We mapped the tasks we defined in Section 3.3.1 to data analysis questions that were available in the data 3.3.2 and a visualization, which we intended to re-create with all the visualization tools in our evaluation.

Task	**Data analysis question**	**Visualization**
	Elementary Tasks	
Lookup	*'How many points did player X average in season X?'*	Table
Comparison	*'How do averages of positions C and PF differ in the AST ratio over the last six years?'*	Grouped Bar Chart
Relation-seeking	*'Which players had a Player Efficiency Rating (PER) above 25 in the past 27 years?'*	Scatter Plot
	Synoptic Tasks	
Pattern Search	*'During which periods did the percentage of players taller than 205 cm drop below 30 percent?'*	Area Chart
Behavior Comparison	*'How did the average height of players vary by nationality over the 27-year period?'*	Choropleth Map

easy to find information quickly and accurately. For the Comparison task, a grouped bar chart makes it easy to see how group-based averages have changed over time. For the Relation-seeking task, a scatter plot was used to show many individual observations at once and lets the users highlight specific outliers with conditional filters. The two Synoptic tasks focused on getting a better overall picture of the dataset's structure. For the Relation-seeking task, an area chart was used, to highlight changes over time in a linear way. For the Behavior Comparison task, a choropleth map was created to see categories change over time and space.

3.3.3 Grading System

We required on a grading system to make the evaluation process more organized. In this case, the system is a set of scoring rules that lets the users compare different visualization tools in a clear and consistent way. There are five evaluation dimensions in the system, each of which looks at a different part of how well the tool works: ease of creation, visualization capabilities, customization options, data preparation requirements, and output and interactivity.

Based on recent research on multi-criteria evaluation of visualization systems [100] and mobile data visualizations [101], the grading system's dimensions were chosen and changed to fit the study's goals, which were to see how well different tools do at reproducing the chosen elementary and synoptic tasks. This grading system ensures that the task-based evaluation is both methodical and aligned with established approaches in the visualization evaluation literature.

- Ease of Creation (3 points)
  - 3 Points: The visualization can be created with minimal effort using intuitive, user-friendly interface. Common actions like drag-and-drop are supported with little to no need for code.
  - 2 Points: Visualization is possible through a well-documented process but requires some manual steps, including light coding or configuration.
  - 1 Point: The creation process is complex and not intuitive. Significant coding, use of external libraries, or reliance on documentation is necessary.
  - 0 Points: Visualization is not possible.
- Visualization Capabilities (4 points)
  - 4 Points: All elements of the desired visualization are rendered accurately and completely, without compromises.
  - 3 Points: The majority of the visualization is correctly rendered. Minor elements may be missing or slightly misrepresented.
  - 2 Points: The visualization can be produced, but with significant limitations or inaccuracies in how the data is displayed.
  - 1 Point: 1 Point: Only basic visual elements can be shown; advanced features or specific data representations are not possible.
  - 0 Points: Visualization is not possible.

- Customization Options (3 points)
  - 3 Points: Users can customize labels, colors, axes, scales, overlays, layout, and more. Extensive flexibility is provided.
  - 2 Points: Most common customization options are available, though certain aspects are fixed or limited.
  - 1 Point: Only a few visual properties can be modified.
  - 0 Points: Visualization is not possible.
- Data Preparation Required (2 points)
  - 2 Points: No need to make adjustments to the original dataset.
  - 0 Points: Original dataset had to be modified to a new one / visualization is not possible.
- Output and Interactivity (3 points)
  - 3 Points: The visualization supports dynamic interaction (e.g., filtering, tooltips, hover states) and can be exported in multiple formats.
  - 2 Points: Some interactivity is present (e.g., tooltips or zoom), but there are minor issues (not all options are available).
  - 1 Point: The visualization is static or offers minimal interactive elements. Export options are limited.
  - 0 Points: Visualization is not possible.

3.4 Phase 4: Formulating Guidelines

In the last phase, we mapped the clustering results from Phase 2 and the task-based evaluation results from Phase 3 to different user roles in data science, to identify how the visualization tools map current skill sets which are required for data science jobs. Previous studies focussed on the skills that are required for different data science jobs [102]. Other studies extracted roles like *Data Analyst* or *Data Engineer* from job advertisements using text analysis [103]. Behrisch et al. [9] defined three roles in Visual Analytics (Data Management, Reporting Manager, and Data Analyst) and analyzed which commercial applications fit which role.

Gunklach et al. [104] combined skill-based analysis and text-based analysis based on job advertisements to extract data science job roles. Since this analysis uses both skills and texts and represents a current state-of-the-art in data science, we based our analysis in this thesis on these job roles. The authors categorize data scientists into these nine roles:

Business Users Business users have application domain skills without having statistical, mathematical, or computer science skills. These users are characterized by different degrees of domain knowledge and disciplinary responsibility.

Business Analysts Users categorized as business analysts can be characterized as analytical business users, thus having strong domain knowledge. Business analysts have skills in both business intelligence and business analytics. Business analysts can make analytical decisions and communicate them to business users.

Data Analysts The third user group summarizes users that have less domain knowledge but more analytical skills in mathematics, computer science, statistics, and problem-solving. Data analysts communicate and present findings to business users by, for example, building dashboards. In contrast to business analysts, data analysts use simple AI approaches such as decision trees and k-means clustering in their analysis process.

Applied Data Scientists Users in this group have no specific business knowledge but rely strongly on machine learning, statistics, databases management, and data engineering.

Research Data Scientists Research data scientists take on the role of scientists, concerned not so much with applying existing algorithms as with modifying them or developing new ones.

Data Engineers The data engineers' job is to makes data available. They implement data pipelines to extract data from source systems into other systems such as data warehouses.

ML Engineers Users in this group are involved in diving deeper into the code and deploying models into productive solutions.

Software Developers Users identified as software developers have a background in computer science and are mostly concerned with programming tasks.

Data Science Architects Data science architects are similar to software developers in their background in computer science. They also have substantial experience with cloud architectures and management of databases.

CHAPTER 4

Results

The study in this thesis was conducted based on the methodology described in Chapter 3. In the upcoming sections, the results of the three analysis phases (Phase 1 - Section 4.1, Phase 2 - Section 4.2 & Section 4.3, and Phase 3 - Section 4.4) are described. Section 4.5 maps the analysis results to identified user groups in data science.

4.1 Landscape of Visualization Tools

After applying the screening and selection criteria to the long list (shown in Table 3.1), we ended up with a short or final list of visualization tools which was then used in the upcoming for the subsequent comparative and task-based analysis steps. The final list of visualization tools is shown in Table 4.1. We ended up with 22 visualization tools for our evaluation. 10 of them were applications and 12 were charting libraries.

4.1.1 Applications

Applications provide strategic insights across a range of business domains in addition to improving convenience and visualization accuracy. Applications help users identify patterns, monitor KPIs, and predict future trends in fields like financial reporting, marketing analytics, and sales forecasting. According to a study by Abdulla et al. [105], real-time visual feedback in these domains enhances operational efficacy and decision-making accuracy. Applications provide multi-tiered and multi-dimensional data perspectives, enabling users to interact with different data perspectives. Sun [106] states that interactive and hierarchical dashboards enable Business Intelligence visualization tools to assist organizations in recognizing market trends, categorizing consumer behavior, and enhancing internal operations.

Table 4.1: Final list of visualization tools selected for the comparative feature and task-based analysis.

Applications	**Charting Libraries**
Tableau	Seaborn
PowerBI	Matplotlib
Looker Studio	Plotly
Excel	Bokeh
GoodData	Pygal
Flourish	HoloViews
ZOHO Analytics	ggplot2
Domo Charts	ggvis
ThoughtSpot	Vega-Lite
Sisense	D3
	ChartJS
	ApexCharts

Tableau

Tableau is a popular business intelligence and data visualization tool that lets people make interactive dashboards, reports, and charts by dragging and dropping, and no programming skills are needed. It can connect to a wide range of data sources, such as spreadsheets, relational databases, and cloud platforms like Snowflake, Excel, and SAP HANA. Industry analyses and independent reviews describe Tableau as one of the best-known and most widely adopted business intelligence and analytics brands worldwide.

Tableau enables the users to explore data in dashboards interactively by giving users filters, grouping operations, calculated fields, and drill-down features as options. Users can create their own calculations, break down and combine data in different ways, and interactively improve the displayed subset of data. The software has many different types of standard visualizations, like bar charts, scatterplots, maps, histograms, and bullet charts. Users can put these types of visualizations together in dashboards to keep an eye on key performance indicators, find outliers, compare groups, and share the analytical results.

Several academic studies have examined how Tableau can be used to represent and explore complex or hierarchical data structures. A study conducted in 2022 by Hinson et al. [107] evaluated user interactions with different approaches to hierarchical data presentation in Tableau dashboards. Drill-down charts, expandable sections, and filter-based views were the three interface strategies that were assessed in the study. According to the results, drill-down interfaces improved accuracy and made navigation easier, especially for users who were unfamiliar with Tableau or the raw data structure. These results demonstrate Tableau's ability to support adaptable, user-directed exploration and point out areas for improving guided data discovery.

There are also limitations. For instance, Tableau's customization capabilities, although sophisticated, do not reach the level of detail available in more advanced code-based platforms such as D3.js or Plotly, as it was observed in [108]. Additionally, as a paid commercial product, its licensing fees can be prohibitive for some smaller teams. However, Tableau still offers, when compared to its closest competitors, an impressive power, usability, and presentation quality throughout visual data analysis.

PowerBI

Microsoft Power BI is an all-encompassing Business Intelligence suite that allows users to connect, analyse, and visualize data dynamically and at scale. According to the official Microsoft documentation, it is *'a collection of software services, apps, and connectors that work together to turn your unrelated sources of data into coherent, visually immersive, and interactive insights'* [109]. Power BI integrates with Excel, Azure services, SQL Server, and many other data sources, which makes it highly usable for both non-technical users and users with programming experience. It helps companies that are already using Microsoft products the most because SharePoint and Teams make it easy to share and import data.

The user interface of Power BI is focused on interactivity and usability. It has drag-and-drop features for making dashboards and DAX (Data Analysis Expressions) for making custom calculations. Users can make detailed reports, charts, and drill-down dashboards that automatically update when the data changes. Power BI is offered as a cloud-based service, meaning that reports and dashboards are hosted on Microsoft's online platform rather than on local machines, which lets users stream and work together on mobile devices in real time.

Recent academic studies have looked into how Power BI can be used in areas other than business, like education and the public sector. Dashmukh et al. [110] conducted a study that emphasized its role in improving data accessibility and usability for performance evaluation in educational systems, allowing stakeholders to engage with and interpret data without needing advanced technical expertise.

Power BI may have problems when working with very large data models or complex DAX expressions, which makes it necessary to have a deep understanding of data modeling. Even so, it is still one of the most powerful and affordable Business Intelligence tools on the market.

Looker Studio

Formerly Google Data Studio, Looker Studio is a free Google-developed web-based data visualization and dashboarding tool. Users may generate interactive reports and dashboards that draw information from several platforms including Google Analytics, Google Sheets, MySQL, BigQuery, and many others. With a focus on ease of use and collaboration, Looker Studio allows users to create visualizations in real-time without the need of advanced technical skills.

Looker Studio has tools for sharing analytical content, such as the ability to share dashboards or put reports on other websites. It also has live data connections, so dashboards can show updates from connected sources without needing to be refreshed manually. Users can make many different types of visualizations using the graphical interface instead of programming, which might make it easier for new users and for the non-technical users to get started. However, using pre-made visual parts and connectors might make it harder to be flexible when users need to do more advanced analysis.

Academic use cases have demonstrated the tool's effectiveness in visualizing institutional metrics and educational performance. For example, Dyon et al. [111] used Looker Studio to create a school-wide academic dashboard, and they reported 89% usability satisfaction for assisting in decisions based on teacher attendance and student grades.

Looker Studio's integration with the Google Ecosystem is one of the most prominent features. Using intuitive interface, users can create time series charts, scorecards, maps, bar graphs, and other advanced visuals. Filters and dynamic date ranges and other user generated data exploration techniques are also supported, making creating visuals much easier.

Excel

One of the most prominent and most commonly used spreadsheet software across the world, Microsoft Excel, has for a long period of time been used in business, finance, and analytics. It also offers powerful tools for data organization along with integrated visualization which makes it possible to use it for data analysis and presentation.

Unique features included are user friendly interface that allows for quick analysis of data, conditional formats, data validation; all-purpose chart generation with specific options as well as integration with Power BI for added capabilities.

A recent study, Mahmud et al. [112] investigated Excel's user-friendliness, visual appeal, and reputation in educational settings. The research found that Excel's intuitive interface, broad feature set, and positive reputation significantly influenced user preference and loyalty in data visualization tasks.

Unlike other statistical analysis programs, Excel's high-performance data processing and real time interactivity is not its speciality. It is also missing advanced methods of statistical analysis and machine learning algorithms. However, it is still a highly useful tool due to its value, ease of use, and ubiquity when it comes to presentations to non-technical audiences.

GoodData

GoodData is an advanced cloud-based analytics platform offering Artificial Intelligence Solutions for Business Intelligence operations; however, it specializes in the delivery of embedded analytics (which refers to integrating dashboards, reports, and data visualisations directly into existing business applications, so users can access insights within the tools they already use) and reporting services. GoodData is developed to plug and play seamlessly within existing business structures to enable the creation of custom dashboards and visualizations capable of being embedded into proprietary applications or customer portals.

GoodData has features for data governance and multidimensional modeling to help with complex data transformations. These features are useful in settings where data management is very important. The platform is made to work in big multi-tenant architectures, which means that different groups of users or clients can use the same system while keeping their data separate, which is used by many organizations that need to give a lot of users access to analytics while keeping security controls in place use this structure. The platform has ETL pipelines that lets users connect to many different data sources, and it has a web-based interface for making reports and dashboards. But how well these tools support advanced customization depends on how they are set up and how technically proficient the users are.

Nevertheless, individual analysts and small to medium enterprises are not the most likely users of GoodData services, as the platform is tailored more towards corporate developers and enterprise teams. Setting up user roles and custom data models involves considerable effort and time, making the learning curve steep. For organizations hoping to embed Best in class Artificial Intelligence Services, the complexity may not be an issue.

Flourish

Flourish is a cloud hosted data visualization tool, which allows users to make fully animated, interactive charts and stories, without the need of any coding skills. It is especially useful for analysts who want to add an impactful visual dimension to their presentations. Flourish offers an extensive catalogue of visualization templates including bar chart races, heatmaps, pictogram, 3D globes, and even network diagram. These templates are not standalone designs, but templates meant for the users to create narratives around them and guide the audience through the data.

One of the capabilities that makes Flourish stand out is ease of use. Users can upload spreadsheets, adjust the visuals through the editor, and publish the created visuals to the web or embed them in other web pages. In addition, it also offers a way to link data so that the charts can be kept updated with new source data. This is very useful for visual content that needs to be up to date but cannot be made fast enough by constantly changing it manually.

Although Flourish provides a robust set of capabilities for design and interactivity, it lacks analytical depth. Statistical analysis and data manipulation is not favourable within the design of this visualization tool. Hence, it is better suited as a front-end tool after the data has been cleaned and processed. Still, its design-oriented focus makes it an excellent tool for communicating insights in an engaging and accessible way.

ZOHO Analytics

ZOHO Analytics offers Self Service Business Intelligence and data reporting tools that use machine learning artificial intelligence technologies supporting voice commands to minimize the need for the user engaging any detailed coding skills. This visualization tool can integrate a variety of data sources from spreadsheets and databases to cloud storage and business software applications. It is custom designed for small, medium and enterprise level businesses on top of featuring easy to use data preparation and visualization design interfaces.

What distinguishes ZOHO Analytics is its built-in automation technology as well as its conversational analytics Zia, which enables users to ask questions in their local tongue and receive visualization-based answers. Also, it features predictive analytics, and trend forecasting along with drag and drop feature for customizing dashboards. With Zia giving direct answers, users are able to review previously acquired information and they are also put in a great position to plan and develop future strategies.

In a user study, Nunes et al. [113] evaluated the communicability of dashboards generated using three leading BI tools-Tableau, Power BI, and ZOHO Analytics-based on semiotic inspection. The results showed a greater number of communication disruptions in the interaction with Tableau and Power BI in relation to ZOHO Analytics, indicating that ZOHO Analytics allows the user a greater understanding of the result of the actions taken.

While it is difficult to contest the effectiveness of the Zia software, it's the effectiveness upon its completion that tends to miss the strength and scope of its predecessors. Users might also be troubled with lack of advanced customizations in geo or timezone targeted dashboards and other singularly focused data readers.

Domo Charts

Domo Charts are the visualization component of the Domo business intelligence platform, used to explore and communicate data through interactive visuals. Users can choose from more than 150 chart types, including specialized "data science" charts. The created charts can be customized by the users by controlling colors, labels, axes, formats, and annotations to tailor the appearance to specific audiences and reporting standards.

The created charts are interactive by default, supporting drill-downs, filtering, and hover tooltips that reveal detailed values, which enables users to move from high-level KPIs into granular records without leaving the dashboard context. Domo's chart engine is designed to handle large datasets efficiently, so even dashboards backed by millions of rows remain responsive.

ThoughtSpot

ThoughtSpot is a cloud-based Business Intelligence and analytics platform, with a focus on a search-driven and AI-enhanced user experience. Business users can ask questions about their data in natural language and get immediate, interactive visual representations. The platform eliminates the need for pre-made dashboards or SQL expertise by combining advanced indexing with AI analytics to deliver highly contextual responses from complex datasets. Its approach to "search analytics" sets it apart by providing a corporate data interface similar to Google.

ThoughtSpot interfaces with numerous cloud data warehouses, including Snowflake, BigQuery, Redshift, and Databricks. The SpotIQ engine finds patterns, outliers, and trends on its own and gives the users AI-generated insights in seconds. ThoughtSpot's interactive dashboard solution, Liveboards, lets users add analytics to apps or websites and customize visualizations. The platform is great for making quick decisions in fields like retail, healthcare, and financial services, where having access to in-depth information is important.

ThoughtSpot has several drawbacks despite its creative approach. The reliance on well-structured datasets is a major drawback; in order to enable smooth and accurate search experiences, users usually need a big back-end data preparation. This creates a dependency on data engineering or IT teams, especially in companies with complex data ecosystems. Also, the customization of visualizations is less flexible than that of tools like Tableau or Power BI, which offer more accuracy in terms of layout and appearance.

Sisense

Sisense is a cloud-native analytics platform that enables the integration of analytics into business processes, products, and services. Unlike traditional BI tools, Sisense focuses on embedded analytics and scalability, which is beneficial for product teams and providers of enterprise software. Users can add visual analytics to web portals, SaaS apps, and mobile devices with Sisense, making data-driven environments that are seamless and native.

The platform combines data preparation, modeling, and visualization into one interface. One of the most impressive things about it is the in-chip data engine, which lets data be processed quickly, even when it is large or complicated. Other functions supported by Sisense include integration of machine learning models, real-time data alerts, and customizable dashboards for different users.

The learning curve can be steep for teams without dedicated development headcount in place. While Sisense enables great flexibility, taking advantage of it requires significant skills in data modelling and embedding. Regardless, businesses aiming at providing advanced analytics within their products or services rapidly will find Sisense both effective and responsive to their needs.

4.1.2 Charting Libraries

Charting libraries are data visualization tools that emphasize precision, programmability, and versatility. Charting libraries cater to users with programming expertise seeking intricate control over data manipulation, visual aesthetics, and interactivity, in contrast to Business Applications, which prioritize usability. These libraries, extensively utilized in both industry and research for generating customized, high-fidelity visualizations, are generally incorporated into programming environments such as Python, R, or JavaScript.

Notable charting libraries are D3.js (JavaScript), Plotly (Python), and ggplot2 (R). D3.js focuses on data binding and custom SVG manipulation; Plotly specializes in interactive, browser-based visualizations; and ggplot2 utilizes the grammar of graphics model for statistical plotting. Dogadina and Voronin [73] recommend these libraries for their capacity to provide intricate analytical control, rendering them suitable for projects where data complexity and precision are critical.

The great degree of customization offered by charting libraries over business applications is one of their primary benefits. Nearly every element of the visualization is editable by users, including tooltips, animations, and axis labels and colors. In fields like science, finance, or technology where data must be accurately represented-often in non-standard formats-this level of detail is particularly crucial. This makes charting libraries a popular option for subject matter experts who feel at ease working in code environments, as noted by Lavanya et al. [114].

This flexibility entails a more challenging learning curve. Charting libraries generally necessitate proficiency in programming languages, syntactic structures, and data formats. Although visualization tools such as ggplot2 exhibit elegance in their grammar-based architecture, they may prove to be unintuitive for novices. Kruchten et al. [115] presented a metrics-based evaluation framework for visualization notations, revealing that libraries such as D3.js and ggplot2 compromise usability for functionality-a trade-off frequently embraced by proficient users in pursuit of control and accuracy.

A further advantage of charting libraries is their incorporation into data science workflows. These libraries are not independent applications but components of broader analytical ecosystems-like Jupyter notebooks, R Markdown, or web applications-rendering them highly extensible and appropriate for reproducible research. For example, Plotly effortlessly integrates with Dash to create interactive web applications, whereas D3.js can be incorporated into larger JavaScript frameworks for custom-made dashboards and narrative visualizations.

Comparing D3.js and Tableau, Nair et al. [108] discovered that while D3.js is more challenging to master, it offers superior control for visualizing extensive, intricate datasets, especially in big data contexts where conventional BI tools may prove inadequate. This shows a main idea in visualization: the conflict between being expressive and being easy to use.

Although they are technical, many charting libraries have extensive documentation, community support, and abstraction layers that make it easier for the users to familiarize themselves with the visualization tools. For instance, Vega-Lite and Plotly Express are high-level D3.js abstractions that make coding easier without losing important core features. A greater range of users can now more easily access advanced visualization capabilities thanks to these developments.

To summarise, charting libraries are crucial resources for users who appreciate accuracy, control, and integration in their analytical procedures, while also possessing programming knowledge. Their extensibility and customizability make them appropriate for advanced users handling complex data scenarios, despite the fact that they lack the convenience of business applications. These libraries will continue to be essential to data science and research workflows as the visualization tool ecosystem grows.

Seaborn

Seaborn is a high-level Python library built on top of Matplotlib that simplifies the creation of attractive, statistical visualizations. It integrates well with pandas and automates complex plot generation-such as correlation heatmaps, pair plots, and violin plots-making it especially useful for exploratory data analysis. With a basic understanding of Python syntax, users can create graphics using Seaborn's brief and innate syntax.

A key advantage of using Seaborn lies in its integration with pandas DataFrames, which is a cornerstone of data manipulation in Python. Specifically, users can create complex visualisations with short and simple function calls. This, in turn, removes a large portion of the repetitious code that is frequently needed in other libraries. On the other hand, it lacks interactivity options and the deep customization that Matplotlib possesses, which limits its use in real-time applications and web dashboards.

Matplotlib

Matplotlib is a *'comprehensive library for creating static, animated, and interactive visualizations in Python'* [116]. It has a flexible, low-level API that lets the user make all kinds of plots, including interactive, animated, and static ones. Because of how it was designed, it is great for the users who need a lot of customization. Matplotlib is a key part of the Python data science ecosystem because it is the base for more advanced visualization libraries like Seaborn and Pandas plotting tools.

Matplotlib has been widely used in academic contexts, particularly for creating publication-quality figures. Cao et al. [117] conducted a study that emphasized Matplotlib's efficacy in scientific, commercial, and engineering fields, highlighting its accuracy and clarity in depicting intricate visual structures. Kaestria et al. [118] used Matplotlib to show how useful it is for educational research by using line and scatter plots to show how digital device use affects academic performance. These studies confirm its utility for comprehensive, replicable visual analysis in academic workflows.

The main thing that makes Matplotlib so powerful is how flexible and controllable it is when it comes to visual output. This is important for custom and scientific plotting. It works perfectly with NumPy and Pandas, supports LaTeX for equations, and makes graphics with high resolution that can be used in print and online. But it is hard for beginners to learn, especially when compared to more advanced visualization tools like Seaborn or Plotly. The syntax can be long, and we cannot do much with it interactively unless we use it with other libraries like *ipywidgets* or Plotly.

Plotly

Plotly is a graphing library that offers interactive plotting functionalities in languages such as Python, R and JavaScript. It is particularly known for use when building web-based dashboards and real-time visualizations. Plotly is different from traditional static libraries in a way that a user can create charts that they can zoom into, hover over for details, or even export in different formats.

Academic research recognizes Plotly for its utility in interactive and exploratory data visualization. A 2023 study in the Advances in Systems Analysis series emphasized Plotly Express's high-level syntax and ability to generate insightful, publication-quality graphics with minimal code, especially in exploratory data analysis workflows [119].

Plotly is useful because it is interactive, works with web browsers, and can handle complicated visualizations like maps, 3D plots, and animations. It works with frameworks like Dash and Jupyter Notebooks, and the users often use it to make dashboards. But it might use more resources than static plotting libraries, and users sometimes have trouble making changes without learning how to use lower-level JSON-like syntax. Also, when working with large datasets or very customized visualizations, performance and flexibility may not be as good as they are with Matplotlib or D3.js.

Bokeh

Bokeh is a Python library for creating interactive visualizations for modern web browsers. Bokeh lets users create plots in a wide range of formats, from standalone HTML files to embedded dashboards and web apps. It also supports high-performance interactivity. It works well with NumPy, pandas, and Jupyter Notebooks, and it has both high-level and low-level APIs that let the user design and build things in different ways.

Bokeh has been employed in academic and applied research settings that require spatial or interactive data visualization. A notable example is the ST VISIONS tool, which builds on Bokeh to allow intuitive map-based visualizations of spatio-temporal datasets.

Researchers from the University of Piraeus showed that Bokeh allows for rich interactivity, such as filtering, coloring, and zooming, while still keeping a high-level interface for quick prototyping [120].

Bokeh is great for making interactive visualizations that work on the web and do not require much JavaScript knowledge. Because it can stream data in real time, integrate widgets, and support callbacks, it works especially well for dashboards and data apps. However, Bokeh may be harder to use for basic plotting than libraries like Seaborn, and large, complex plots may run slower if they are not optimized correctly. Also, even though its API is strong, it may seem complicated or broken when the user tries to do more advanced customization.

Pygal

Pygal is a Python charting library that focuses on generating vector graphics, particularly SVGs. It particularly well-suited for web-based applications. One advantage of Pygal is that is it beginner-friendly, since the user can quickly visualize data, without the need to worry about design details. The SVG output format ensures that graphics are scalable without any loss in resolution, making them ideal for responsive web pages or print-quality documents. Additionally, Pygal charts are interactivity (for example like tooltips or hover effects), which can be helpful when trying to present data in a more engaging way.

Pygal may not be the best choice for big or very customized visualizations, but it is great at what it does best: making clean, consistent charts that can be embedded. It is a useful tool for quick reporting because its syntax is simple and its default styles are nice. This is especially true when simplicity and clarity are more important than deep interactivity or customization.

HoloViews

HoloViews is an *'open-source Python library designed to make data analysis and visualization seamless and simple'* [121]. By focusing on the structure of the data instead of the details of plotting commands, it lets users make interactive, browser-based plots with very little code. HoloViews works well with backends like Bokeh and Matplotlib, and it works well in Jupyter Notebooks for exploratory analysis and research that can be repeated.

Stevens et al. [122] highlight HoloViews' utility in enabling reproducible scientific workflows. The authors explain how HoloViews reduces the need for custom plotting code by allowing users to pair their data with metadata structures, which are then rendered into interactive visualizations.

HoloViews does a good job of cutting down on duplicate code and making it easier to work with complex, multidimensional data. Its ability to easily create interactive visualizations makes it a popular choice for scientific dashboards and notebooks. But it has some problems, especially when users need to have very precise control over each plot element. In these cases, they might still need to use the syntax for Bokeh or Matplotlib. Also, beginners or the users who are used to simpler visualization libraries like Seaborn or Plotly may find it hard to learn because it has a steep learning curve and depends on a certain plotting backend.

ggplot2

ggplot2 is used in data visualization packages in R. It is built upon the principles of the Grammar of Graphics [15]. One of the main advantages of ggplot2 is the option to separate data from layers. This modularity lets users to incrementally build up their plots by adding components such as themes, facets, and scales. Also, ggplot2's syntax is easy to understand and consistent, so it is easy to learn and very useful for making graphs that show information.

ggplot2 is very important for teaching people about data and for making scientific graphs. Gould [123] emphasized that ggplot2 assists in the creation of consistent and high-quality plots. It is especially helpful for researchers in HCI and psychology who want visual outputs that can be changed and interacted with.

The biggest advantage of ggplot2 is that it has a method for plotting that is both organized and flexible, which allows a wide range of statistical graphics and layering options. It works well with the tidyverse ecosystem and can handle both small and large datasets without any problems. Still, ggplot2 can be hard to learn for the users who are new to the Grammar of Graphics framework.

ggvis

RStudio made ggvis, a R package that lets the user make interactive graphics using a syntax similar to ggplot2. It also lets the user interact with the graphics on the web. It was built on top of Vega and Shiny and let users make responsive plots that could be viewed in RStudio or embedded in web apps. The goal was to use the "Grammar of Graphics" ideas to make interactive web graphics. But the development of ggvis has slowed down a lot, and the package is now considered dormant.

The biggest advantage of ggvis was that it combined the familiar ggplot2-like syntax with Shiny interactivity, which made it easy to make sliders, tooltips, and layouts that worked on all devices. It was especially appealing to R users who wanted to try out interactive visualizations without having to learn JavaScript. But its main problems-no active development, an incomplete feature set, and limited integration with newer tidyverse tools-have made it mostly useless in modern R workflows.

Vega-Lite

Vega-Lite is a *'high-level grammar of interactive graphics. It provides a concise, declarative JSON syntax to create an expressive range of visualizations for data analysis'* [124].

Satyanarayan et al. [16] describe Vega-Lite as a system that connects statistical graphics and user interactivity. The authors present a grammar of interaction that makes it possible to define user-driven actions like selections and filters in a declarative way. Their evaluation shows that Vega-Lite lets the user make short, reusable visualization specifications while still being flexible and interactive.

Vega-Lite's biggest strengths are its simple syntax, built-in interactivity, and ability to be used in web apps. It works well with responsive and accessible design, so it is great for dashboards, educational tools, and news articles. But it might be hard for the users who do not know how to code to learn, and its JSON-based specification can make it hard to make visualizations that are very unique.

D3

D3 (or D3.js) is a free, open-source JavaScript library for visualizing data. The library's *'low-level approach built on web standards offers unparalleled flexibility in authoring dynamic, data-driven graphics'* [125]. Many journalists, researchers, and business analysts use it to make custom visualizations that change based on user input and real-time data streams.

D3.js has been a key tool in academic research on visualization that involves a lot of interaction. Chen and Zhou [126] put forward a data-driven visualization model utilizing D3 for dynamic dashboards, demonstrating its superiority in fostering real-time insights and user engagement over static charting tools.

D3's greatest asset is that it is flexible in a way that users can make almost any kind of visual from scratch. It can handle animations, transitions, and interactivity better than most other libraries. But this power comes with a steep learning curve; to make visualizations, the users need to know JavaScript, how to change the DOM, and how to write functional code. For quick visualization tasks, it is not as good as tools like Plotly or Vega-Lite, and it might be too much for regular chart types.

ChartJS

Chart.js is an open-source JavaScript library that makes it easy to make responsive charts for web apps and other things. Chart.js is very popular with front-end developers and data scientists who are making dashboards or interactive reports because it is easy to set up and does not need much configuration.

Chart.js uses a declarative programming language, which makes it easy to use for beginners. Users can quickly create charts with only using a few lines of code. It also has built-in features like animations, tooltips, and legends that make charts more interactive without the need for extra libraries or plugins. Chart.js uses HTML canvas to render, which makes it suitable for large datasets.

Chart.js does not allow for as much customization as D3.js, but it is easy to use and looks good, so it is a good choice for a rapid prototyping and fast development cycles.

ApexCharts

ApexCharts is a modern JavaScript library for making charts that are very interactive, dynamic, and responsive. It has a lot of advanced chart types that are often needed in dashboards for tracking performance and finances. These include candlestick charts, heatmaps, radial bar charts, range area charts, and timelines. It was made with modern frameworks in mind and works well with React, Vue, and Angular.

ApexCharts is different because it lets us update data in real time and customize it in advanced ways through a clean, modular API. ApexCharts also makes it easy to export to SVG, PNG, and even PDF formats.

ApexCharts is great for situations where we need more than just still graphs. It does well in modern reporting tools that let us interact with them and dashboards that let the users watch things happen in real time. It has a lot of features and good documentation, which makes it a strong candidate for analytics projects that will be used in production.

4.2 Feature Mapping

The 22 visualization tools were evaluated and categorized according to the three groups *visualization support* (e.g. availability of specific chart types), *automatic analysis* (e.g. built-in support for statistical or machine-learning operations), and *user guidance, perception, and cognition*. The groups were described in more detail in Section 3.2.1.

To evaluate **visualization support**, we looked at how well each tool could handle a wide range of types of visualizations. Table 4.2 shows a summary of the visualizations tools in the study and which types of visualization techniques are supported by them. The visualization techniques have been described in detail in Section 2.1.1. There is a clear line between commercial software and open-source libraries. Many of the presented visualization techniques are well-supported by applications like Tableau and Power BI, and by charting libraries like Plotly. The hereby mentioned three visualization tools cover almost all of the chart types that were tested. On the other hand, applications like Looker Studio, Excel, and GoodData cover only some areas, usually only treemaps, sunburst charts, or geographic maps. Seaborn, Matplotlib, and ggplot2 are examples of charting libraries that are good at making traditional plots but do not have built-in support for more advanced techniques like horizon graphs. This means that while popular applications are great at providing a full set of visualizations, charting libraries often need extensions or user code to get the same level of coverage.

To evaluate **automatic analysis**, we looked at the features provided by the tools to support the generation of additional (i.e., not contained in the data) automatic analysis results. Table 4.3 shows a summary of the visualizations tools in the study and which types of analysis techniques are supported. Applications like Tableau, Power BI, Excel, and ZOHO Analytics come with built-in support for tasks like classification, clustering, regression analysis, and even predictive modelling. Charting libraries do not have such features built in. Instead, they leave statistical and machine learning tasks to be done in

Table 4.2: Visualization support. Binary matrix showing, for each visualization tool, whether it natively supports the indicated visualization types. The visualization techniques used here in the evaluation have been described in detail in Section 2.1.1.

		Parallel coordinates	Sunburst charts	Treemaps	Node-link diagrams	Matrix-based	Geographic maps	Heatmaps	Dense pixel-based	Glyph-based multiples	Horizon graphs
Applications	Tableau	✓	✓	✓	✓		✓	✓	✓	✓	✓
	Power BI	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Looker Studio		✓	✓			✓	✓			
	Excel		✓	✓			✓	✓			
	GoodData			✓				✓			
	Flourish		✓	✓	✓		✓	✓		✓	
	ZOHO Analytics		✓	✓			✓	✓			
	Domo Charts		✓	✓		✓	✓	✓			
	ThoughtSpot		✓	✓		✓	✓		✓		
	Sisense		✓	✓			✓	✓			
Charting Libraries	Seaborn					✓	✓		✓		✓
	Matplotlib					✓	✓	✓		✓	
	Plotly	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	Bokeh					✓		✓	✓	✓	
	Pygal										
	HoloViews				✓	✓	✓	✓		✓	
	ggplot2	✓		✓		✓	✓	✓	✓	✓	
	ggvis					✓				✓	
	Vega-Lite					✓	✓	✓		✓	
	D3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	ChartJS										
	Apex Charts		✓	✓				✓			

the surrounding programming environment. For this reason, charting libraries have not been included in Table 4.3 since all fields would be empty. The results show that there is a clear difference between applications and charting libraries. Charting libraries do not see the necessity to include automatic analysis into the library itself, since they are embedded in programming environments.

Table 4.3: Automatic analysis. Binary matrix showing the availability of built-in features for automatic analysis in each visualization tool, such as basic statistical summaries and model-based methods (e.g., regression or trend estimation). Charting libraries were left out, since the visualization libraries do not provide any automatic analysis features at all.

		Classification	Clustering	Outlier detection	Regression analysis	Time-series prediction	Normality tests (Shapiro)	Hypothesis tests (t-test)
Applications	Tableau	🟩	🟩	🟩	🟩	🟩	⬜	⬜
	Power BI	🟩	🟩	🟩	🟩	🟩	⬜	🟩
	Looker Studio	⬜	⬜	⬜	⬜	⬜	⬜	⬜
	Excel	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	GoodData	⬜	⬜	🟩	🟩	🟩	⬜	⬜
	Flourish	⬜	⬜	⬜	⬜	⬜	⬜	⬜
	ZOHO Analytics	🟩	🟩	🟩	🟩	🟩	⬜	⬜
	Domo Charts	⬜	⬜	⬜	🟩	🟩	⬜	⬜
	ThoughtSpot	⬜	⬜	⬜	⬜	⬜	⬜	⬜
	Sisense	⬜	🟩	🟩	🟩	🟩	⬜	⬜

To evaluate **user guidance, perception, and cognition**, we looked at the possibilities provided by the visualization tools to guide users, and to best support users' perception and cognition. Most visualization tools come with some kind of onboarding or documentation, and tutorials, forums, and usage manuals are almost always included. Table 4.4 shows a summary of the visualizations tools in the study and which types of guidance techniques and actions are supported. Tableau and Power BI are unique because they come with built-in suggestions for visualizations and parameters, as well as large support networks that include user and developer conferences. Almost all applications now support large language models (LLMs), which can be used by users to create visualizations and conduct additional analysis steps like data wrangling. What we were particularly interested about are the tools which have the LLM support integrated, such as Tableau's Einstein Copilot (now called Tableau Agent [127]), ZOHO Analytics is equipped with AI assistant Zia [128], and Sisense supports a generative AI to which any prompt can be given in a human form, and the dashboard will be generated [129]. This shows that AI-assisted guidance is becoming more popular. Open-source libraries, on the other hand, have a lot of documentation but do not have interactive help or community-driven events

Table 4.4: User guidance, perception, and cognition support. Binary overview of whether visualization tools offers guidance features that directly assist users during analysis, such as visualization recommendations, basic workflow guidance, tutorials, manuals, and community support.

		Visualization suggestions	Parameter suggestions	Getting-started tutorial	Advanced tutorials	Online forum	Online wiki	Usage manuals	Email support	User conference	Developer conference	Interactive help	LLM support
Applications	Tableau	🟥	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	⬜	🟥
	Power BI	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	Looker Studio	⬜	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	Excel	🟥	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	GoodData	⬜	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	⬜	🟥	⬜
	Flourish	⬜	⬜	🟥	🟥	🟥	⬜	🟥	🟥	⬜	⬜	🟥	⬜
	ZOHO Analytics	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	Domo Charts	⬜	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	⬜
	ThoughtSpot	🟥	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	Sisense	⬜	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Charting Libraries	Seaborn	⬜	⬜	🟥	🟥	⬜	🟥	🟥	⬜	⬜	⬜	⬜	⬜
	Matplotlib	⬜	⬜	🟥	🟥	⬜	🟥	🟥	⬜	⬜	⬜	⬜	⬜
	Plotly	⬜	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	⬜	⬜	⬜
	Bokeh	⬜	⬜	🟥	🟥	🟥	🟥	🟥	⬜	⬜	⬜	⬜	⬜
	Pygal	⬜	⬜	🟥	🟥	⬜	⬜	🟥	⬜	⬜	⬜	⬜	⬜
	HoloViews	⬜	⬜	🟥	🟥	⬜	🟥	🟥	⬜	⬜	⬜	⬜	⬜
	ggplot2	⬜	⬜	🟥	🟥	🟥	🟥	🟥	⬜	⬜	⬜	⬜	⬜
	ggvis	⬜	⬜	🟥	🟥	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
	Vega-Lite	⬜	⬜	🟥	🟥	⬜	🟥	🟥	⬜	⬜	⬜	⬜	⬜
	D3	⬜	⬜	🟥	🟥	⬜	⬜	🟥	⬜	⬜	⬜	⬜	⬜
	ChartJS	⬜	⬜	🟥	🟥	⬜	⬜	🟥	⬜	⬜	⬜	⬜	⬜
	Apex Charts	⬜	⬜	🟥	🟥	⬜	⬜	🟥	⬜	⬜	⬜	⬜	⬜

like commercial platforms do. The Python libraries (Plotly and Bokeh) stand out by providing the most support mechanisms - Plotly even has its own user conference.

4.3 Comparative Analysis

As outlined in Section 3.2.2, we used clustering (K-Means) to identify groups within the landscape of 22 visualization tools. The tables as shown in Section 4.2 were converted into a numerical feature matrices and then combined. The final feature matrix was subjected to the K-Means clustering analysis. While each column in the dataset correlated to a binary or scaled indicator of particular functionalities like statistical support, customization options, integration features, or interactivity, each row in the dataset represented a unique visualization tool. The algorithm grouped tools on the basis of their measured capabilities by eliminating the attribute labels from the clustering process. Plotting the within-cluster sum of squares (WCSS) for cluster counts ranging from one to ten was done using the Elbow Method to find the ideal number of clusters. The toolset appears to naturally divide into two main capability-based groups, as evidenced by the point at which adding more clusters led to diminishing returns at $k = 2$, (depicted in Figure 4.1)

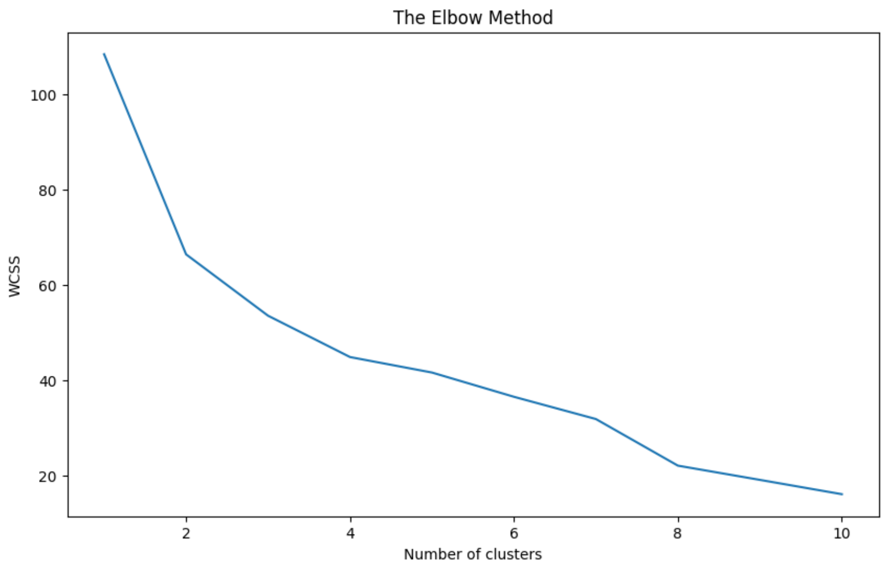


Figure 4.1: An elbow plot that shows the within-cluster sum of squares for different values of k in the K-Means analysis. The curve clearly bends at k=2, which means that two clusters are a good balance between making the model more complex and lowering the variance.

The K-Means algorithm was run with $k = 2$ using the k-means++ initialization method to guarantee stable convergence after the ideal cluster count was determined. Each visualization tool was given a cluster label based on how similar its functional features were, and the resulting clusters showed a clear division between the two groups. We used Principal Component Analysis (PCA) to map the visualization features from Section 4.2 to two dimensions, so that it was possible to show the clusters in one plot. The distribution of clusters for the 22 visualization tools is shown in Figure 4.2. The high-level division between low-code/no-code, presentation-focused platforms and code-centric, highly configurable libraries is probably consistent with this binary classification, which reveals a basic division in the current sample of visualization tools. This division is consistent with earlier findings in the literature, such as in the study by Rost [10], where she pointed out that applications typically favor programmatic interfaces for flexibility or ease of use with limited customization. The only exception we found is Plotly, which is a charting library, but ends up in Cluster 0 due to its unique feature set and community support. The cluster assignment for each visualization application can be seen in Table 4.5.

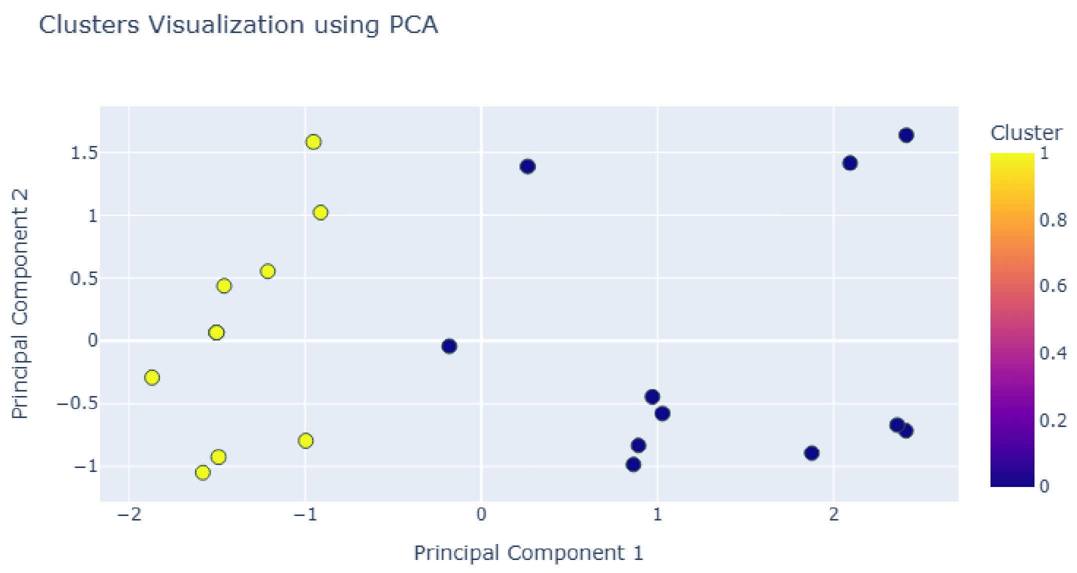


Figure 4.2: A two-dimensional PCA projection of all the visualization tools, with points color-coded based on their K-Means cluster assignment ($k = 2$). The plot makes it easy to see the difference between Cluster 0 (in blue) and Cluster 1 (in yellow). Cluster 0 comprises primarily applications. Cluster 1 includes exclusively code-centric charting libraries. The cluster distribution indicates an inherent difference in features between applications and charting libraries, with one exception (Plotly). Further, the cluster distribution shows more diverse features for the visualization tools in Cluster 0 than in Cluster 1, which means that the landscape of applications is more diverse than the landscape of charting libraries.

4.4 Task-Based Analysis

In the task-based evaluation, we wanted to find out which visualization tools are especially useful for the five tasks we selected in Section 3.3.1. For every task, we assigned an analysis question and a visualization (Section 3.3.2) according to the dataset we used

Table 4.5: Cluster assignments of the assessed visualization tools derived from the K-Means analysis of their feature profiles. There are two groups of tools: Cluster 0, which contains mostly applications, and Cluster 1, which contains exclusively charting libraries. Even though Plotly was identified as a charting library, it shows up in the application-dominated cluster (Cluster 0).

Cluster 0		*Cluster 1*	
Visualization tool	**Type**	**Visualization tool**	**Type**
Tableau	Application	Seaborn	Charting library
PowerBI	Application	Matplotlib	Charting library
Looker Studio	Application	Bokeh	Charting library
Excel	Application	Pygal	Charting library
GoodData	Application	HoloViews	Charting library
Flourish	Application	ggplot2	Charting library
ZOHO Analytics	Application	ggvis	Charting library
Domo Charts	Application	Vega-Lite	Charting library
ThoughtSpot	Application	D3	Charting library
Sisense	Application	ChartJS	Charting library
Plotly	Charting library	Apex Charts	Charting library

and suggestions for visualization chart types from current literature. We based the task-based evaluation on the visualization tool clusters that had been identified during the comparative analysis (Section 4.3). For every cluster, for every tasks, we show the ratings of the visualization tools. The ratings are defined according to our grading scheme (described in Section 3.3.3). The five grading criteria are *ease of creation*, *visualization capabilities*, *customization*, *data preparation*, and *output/interactivity*. For all criteria, we assigned points for each visualization tool. The evaluation gives an overview of how the visualization tools of the two different clusters score for the different grading criteria. For each task, 2–3 representative visualizations are shown and discussed in more detail, to outline why some tools score higher than others and how certain design traits and functionalities affect their performance.

4.4.1 Elementary Tasks

Elementary tasks are defined by Andrienko and Andrienko [96] as tasks that concern individual data elements, or single configurations of elements, rather than aggregates or whole datasets. These tasks refer to single data items or single instances of the dataset.

Lookup

The Lookup task describes the identification of an individual value in the dataset.

After grading all visualization tools, it could be seen that in **Cluster 0**, Tableau got the highest overall score. The scores are summarized in Figure 4.3. Tableau stands out with the highest overall score, while Plotly and ZOHO Analytics occupy the lower end of the range. All tools receive uniformly high ratings for visualization capabilities, ease of creation, and output and interactivity, indicating that, at a basic level, they are broadly comparable in how easily users can build charts and how well those charts support interaction.

The main differences arise in customization options, data preparation requirements, and, to a lesser degree, creation and interactivity scores, where Tableau achieves the maximum ratings and thereby exceeds competitors, especially Domo Charts, Excel, ThoughtSpot, Plotly, and ZOHO Analytics, which lose ground mainly on customization and sometimes on visualization strength. Mid-tier tools such as Flourish, Looker Studio, PowerBI, Sisense, and GoodData cluster closely together, suggesting they provide a balanced mix of visual quality and flexibility without clearly outperforming Tableau or underperforming the lower-scoring products.

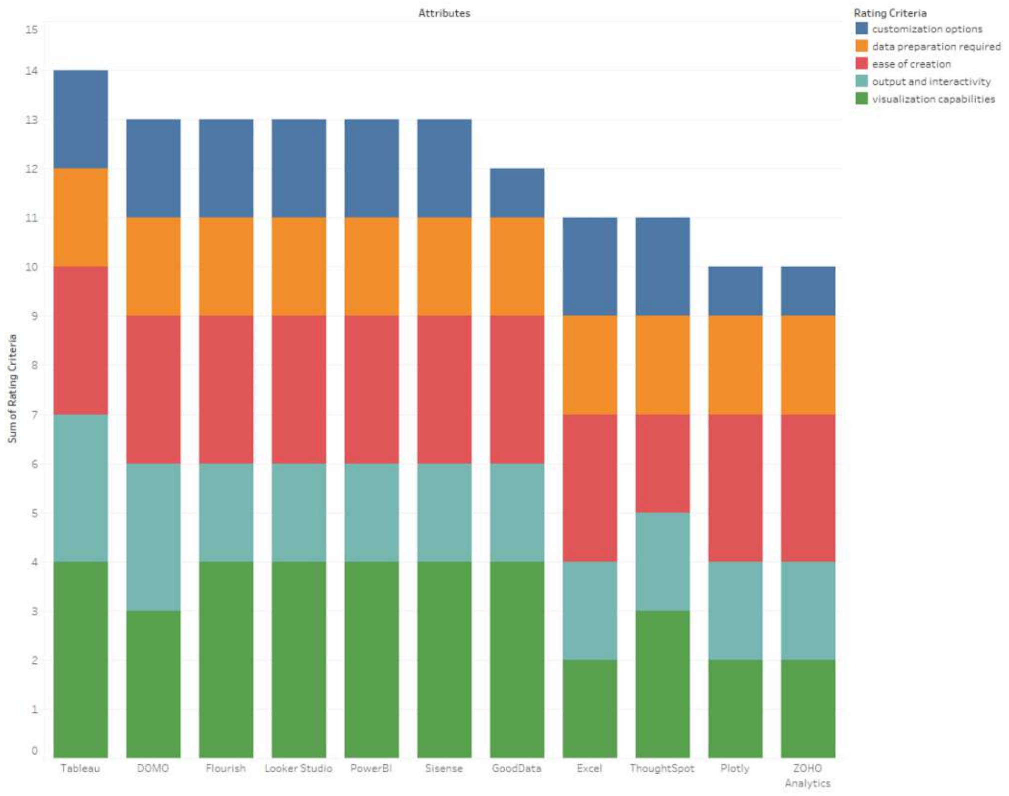
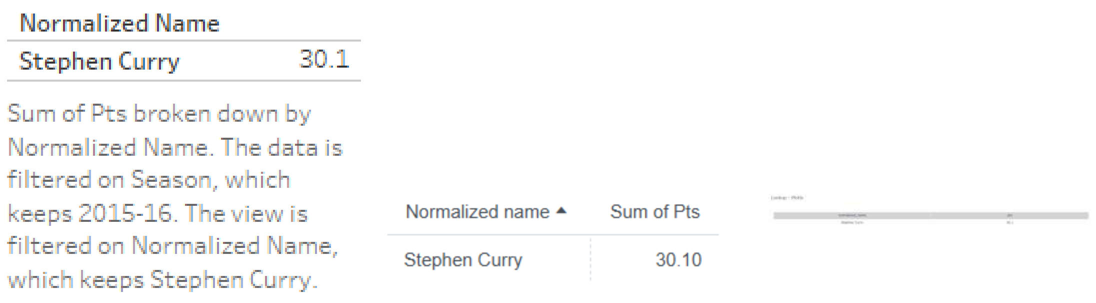


Figure 4.3: Scores of the visualization tools in **Cluster 0** for the **Lookup** task.

Selected charts generated during the evaluation are shown in Figure 4.4. In Tableau, the visualizations were easy to create, all charts could be customized by the user, no extra data preparation was needed, and Tableau charts by default offer interactivity. Tableau exports are a great tool for reporting. Tableau charts also offer additional explanations and guidance, as illustrated in Figure 4.4a. Tableau's KPI-style table to display the average points for a selected player and season. GoodData is one example for a visualization tools that scored less points than Tableau and the group of tools on second place. GoodData works well, but its customization options (e.g., how to handle titles) and export options are not as good as they could be. GoodData's visualization correctly displays the requested value but offers more limited options for formatting titles and exporting results. The visualization created with GoodData can be seen in Figure 4.4b. Plotly is an example for a visualization tool that scored the least points in Cluster 0. Plotly's table layout wastes space and does not let the users customize or export data as easily in this case. The result is shown in Figure 4.4c. Plotly returns the correct value in a table, but the layout wastes screen space and offers fewer options for formatting and exporting.



Normalized Name	
Stephen Curry	30.1

Sum of Pts broken down by Normalized Name. The data is filtered on Season, which keeps 2015-16. The view is filtered on Normalized Name, which keeps Stephen Curry.

Normalized name ▲	Sum of Pts
Stephen Curry	30.10

(a) Tableau (b) GoodData (c) Plotly

Figure 4.4: Lookup (Cluster 0) examples. Comparison between tables in Tableau, GoodData, and Plotly.

When performing the Lookup task for visualization tools in **Cluster 1**, the results were not as clear as for Cluster 0. The results are shown in Figure 4.5. In Cluster 1, the scores are closer together (between 9 and 11 points). There are two winners, ApexCharts and Vega-Lite, and a large group of visualization tools in the middle. ApexCharts Vega-Lite have consistently strong scores across customization options, ease of creation, interactivity, and visualization capabilities. Bokeh and Chart.js follow closely behind, with slightly lower totals driven mainly by somewhat weaker customization and visualization ratings, yet they still offer a solid overall balance of features.

D3 and ggplot2 sit in the middle of the pack: they achieve very strong marks for visualization expressiveness but are pulled down by lower ratings for ease of creation and, to a degree, data preparation, reflecting their more code-intensive workflows. HoloViews and Matplotlib exhibit similar totals, with respectable visualization and interactivity but limited customization convenience and more effort required to prepare and structure data for plotting. Seaborn trails this group with the lowest bar among those shown, suggesting that although it simplifies some aspects of statistical visualization, it lags behind in flexibility and interactive output.

Example charts generated during the evaluation can be seen in Figure 4.6. ApexCharts and Vega-Lite did the best job, while it was not possible to complete the task with ggvis

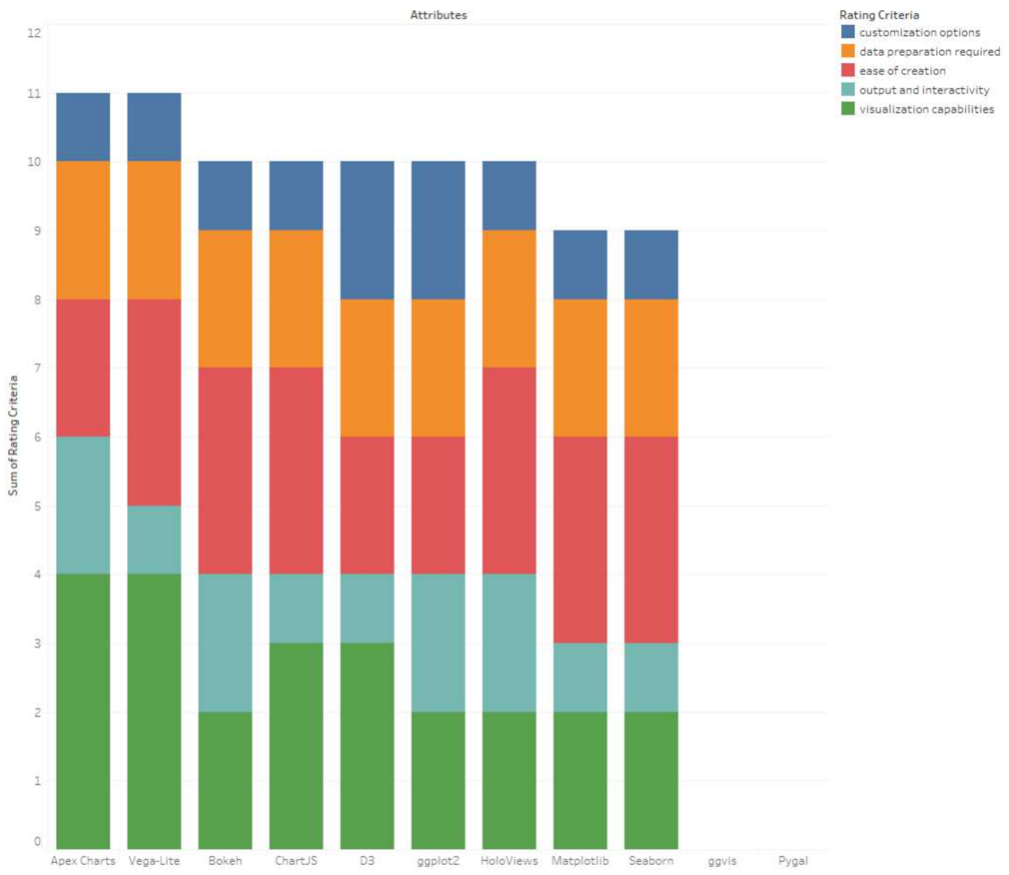
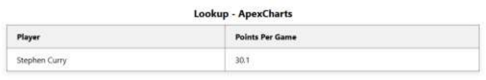


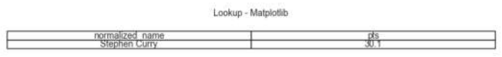
Figure 4.5: Scores of the visualization tools in **Cluster 1** on the **Lookup** task.



Lookup - ApexCharts

Player	Points Per Game
Stephen Curry	30.1

(a) ApexCharts



Lookup - Matplotlib

normalized_name	pts
Stephen Curry	30.1

(b) GoodData

Figure 4.6: Lookup (Cluster 1) examples. Comparison between tables in ApexCharts and Matplotlib.

and Pygal. ApexCharts renders a compact, easy-to-read table with minimal preprocessing (shown in Figure 4.6a). With little preprocessing and basic interactivity/export, ApexCharts makes a small, easy-to-read table.

Matplotlib can technically show a table view, but it does not have any interactivity or deep customization, so it looks more like a static HTML table than a dedicated visualization. Matplotlib can display the required values in a table-like layout, but the result is essentially a static graph, as shown in Figure 4.6b.

Key takeaway from these visualizations is that Cluster 0 tools excel at fast, presentation-ready lookups with minimal setup, while Cluster 1 tools can reproduce the outcome but require more effort to achieve comparable polish and interactivity.

Comparison

The Comparison task contrasts two or more values in the dataset.

The visualization tools in **Cluster 0** show great variety. The tools scored between 7 and 15 points. Plotly and Tableau got a perfect score because it was easy to create the visualizations, they were accurately rendered, and all charts can be customized in many ways. Interactions are built-in for both tools by default. This is an interesting case, since Plotly, a charting library, and Tableau, an application, received a similar score.

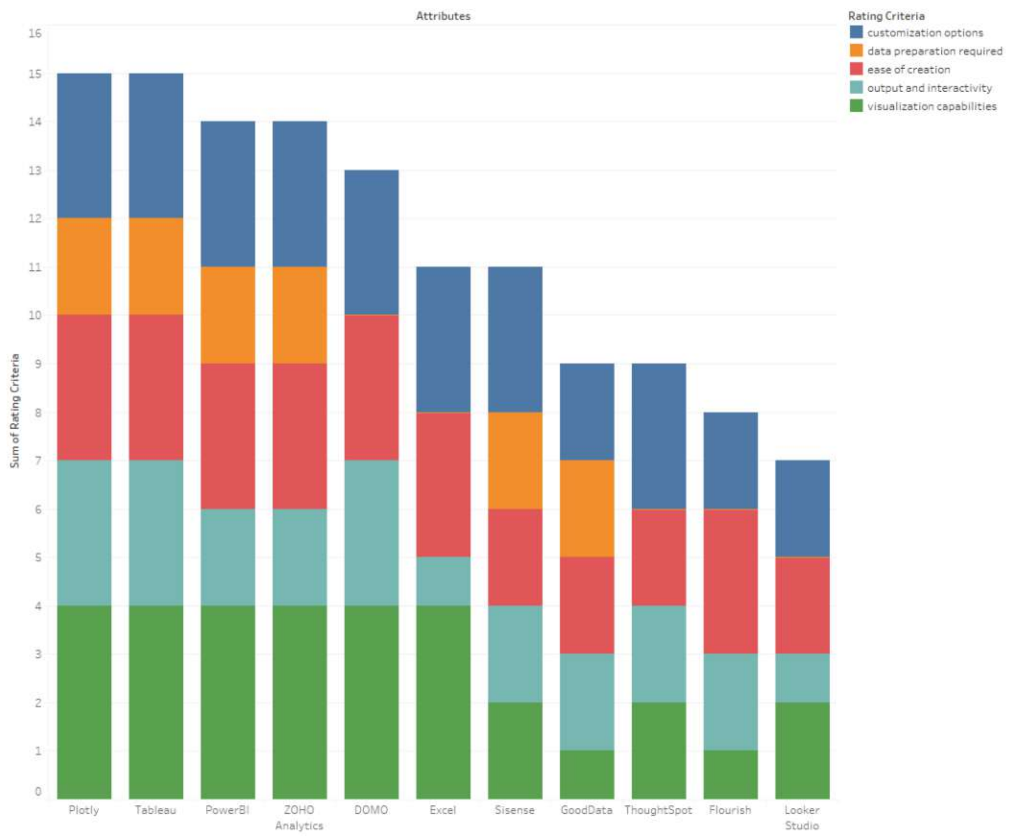
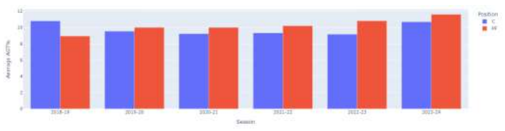
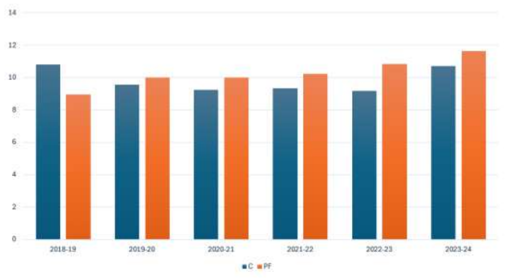


Figure 4.7: Scores of the visualization tools in **Cluster 0** for the **Comparison** task.

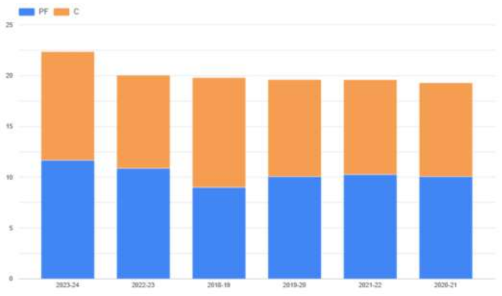
Power BI and ZOHO Analytics form a second tier: they match the leaders on visualization quality but score slightly lower, particularly on ease of creation and interactivity, which pulls their overall totals down a bit. Domo Charts and Excel occupy the middle of the distribution. Both tools provide solid visualization and interactivity, but Domo Charts loses some ground on data preparation and customization, whereas Excel is limited more by its interaction model and flexibility despite being easy to create charts in. Sisense, GoodData, ThoughtSpot, Flourish, and Looker Studio cluster toward the lower end of the scale, with shorter bars largely due to reduced scores in customization and sometimes in visualization capabilities, suggesting they are more constrained or require more work to reach the same level of expressiveness.



(a) Plotly



(b) Excel



(c) Looker Studio

Figure 4.8: Comparison (Cluster 0) examples. Comparison between bar charts in Plotly, Excel, and Looker Studio.

Example charts generated during the evaluation can be seen in Figure 4.8. Figure 4.8a shows the results created with Plotly. Plotly produces a clean grouped bar chart that closely matches the target specification and supports interactive features such as tooltips and highlighting. Excel can generate the required grouped bar chart and offers reasonable formatting options. Excel makes a nice-looking chart that can be customized, but it does not have much interactivity and needs some preprocessing. The example created with Excel is shown in Figure 4.8b.

Looker Studio does not work well because it cannot make a grouped bar chart. Looker Studio only supports stacked bar charts. Data preprocessing is needed, and Looker Studio only has limited functionality for sorting and customization. The visualization created with Looker Studio can be seen in Figure 4.8c.

In **Cluster 1**, the charting libraries received very similar scores. Score were distributed between 10 and 15 points. The results are shown in Figure 4.9. HoloViews, Matplotlib, and Pygal clearly lead the group, each achieving the maximum overall score by combining very strong visualization capabilities with rich customization, adequate ease of creation, and good interactive output.

ApexCharts and Bokeh follow just behind with slightly lower totals: they score highly on visualization expressiveness and interactivity but lose a point on data preparation or customization, suggesting that they still offer a powerful and reasonably convenient experience. D3 and ggplot2 fall into an upper-middle tier. Their scores show top marks for visualization power but reduced scores for ease of creation and, to a lesser extent, data preparation, which reflects their more code-centric and technically demanding workflows compared with the front-runners.

Chart.js and Vega-Lite form a balanced middle group: they provide solid visualization and interactivity but somewhat fewer customization options and slightly more effort around data preparation, yielding shorter overall bars than the leading tools. At the lower end, ggvis and Seaborn exhibit the smallest totals, primarily because of weaker visualization and interactivity scores and only moderate customization.

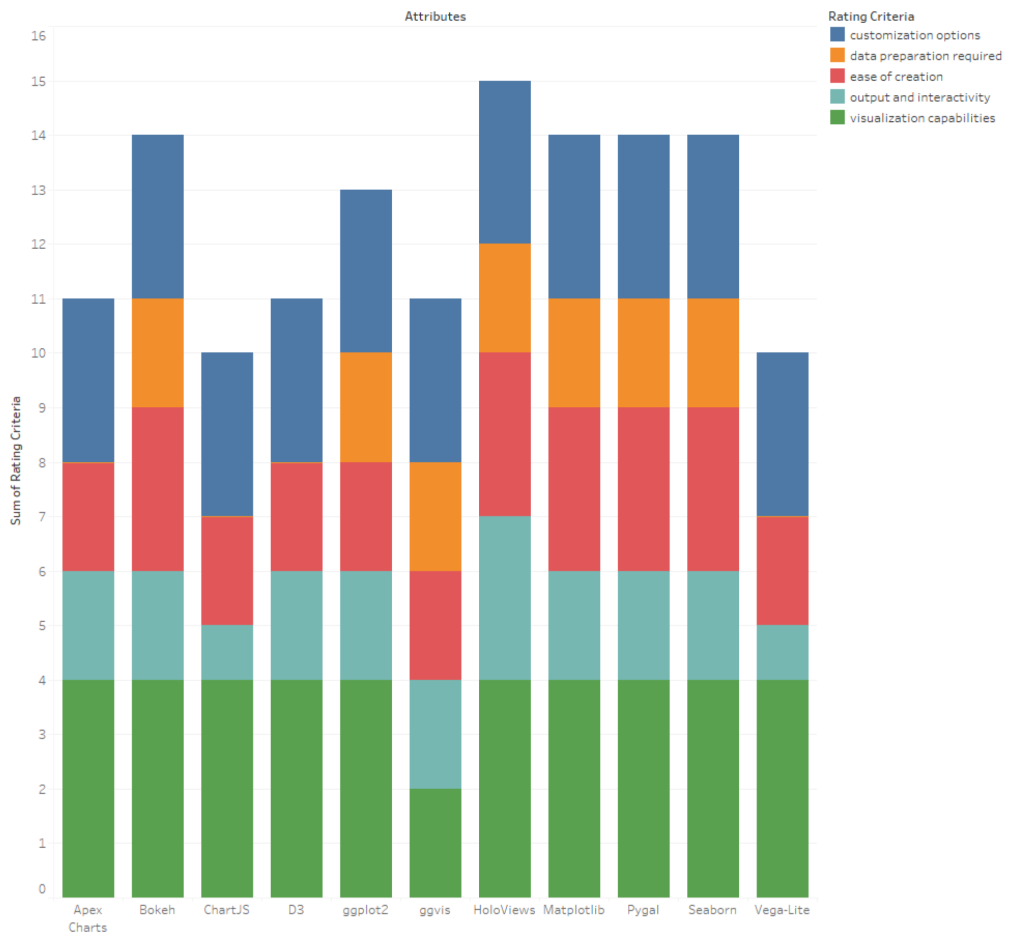
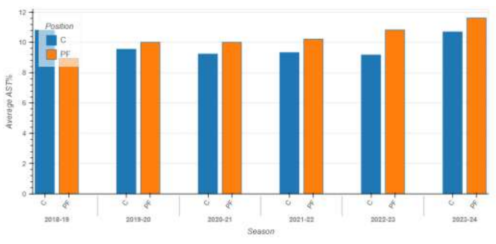


Figure 4.9: Scores of the visualization tools in **Cluster 1** for the **Comparison** task.

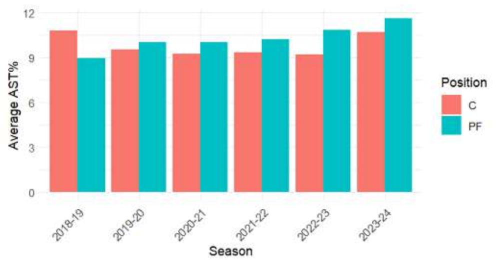
Example charts generated during the evaluation can be seen in Figure 4.10. The results created with Bokeh are shown in Figure 4.10a. Bokeh produces a clear grouped bar chart that closely matches the target specification. Bokeh produces a clear grouped bar chart that closely matches the target specification and offers good customization options, but provides only limited interactivity. The only problem we identified is that it does not allow for much interactivity in the setup that was tested.

ggplot2, results shown in Figure 4.10b, has a workflow that is similar but a little less intuitive, and it does not have any built-in interactivity. ggplot2 can reproduce the required grouped bar chart with high visual quality, yet its code-centric workflow is slightly less intuitive and it lacks built-in interactivity.

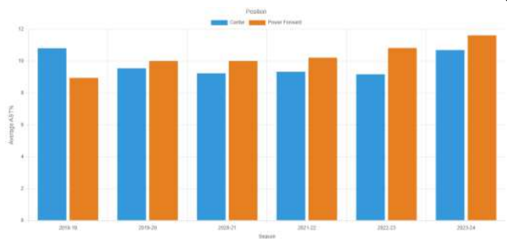
ChartJS is, in contrast, a JavaScript library. It only offers a limited set of visualizations. While ChartJS can display a grouped bar chart, it requires manual preprocessing for aggregation and offers limited interaction and export options. ChartJS, therefore, received the lowest score because it was necessary to preprocess the data and aggregate it, and ChartJS further only provides limited interactivity and export functionalities. Results created with ChartJS can be seen in Figure 4.10c.



(a) Bokeh



(b) ggplot2



(c) ChartJS

Figure 4.10: Comparison (Cluster 1) examples. Comparison between bar charts in Bokeh, ggplot2, and ChartJS.

When it comes to structured comparison tasks, Cluster 1 tools like Bokeh, HoloViews, and Matplotlib are very competitive. On the other hand, Cluster 0 tools vary depending on whether they support the exact chart specification.

Relation-seeking

When performing a Relation-seeking task, users try to find elements that match a certain condition.

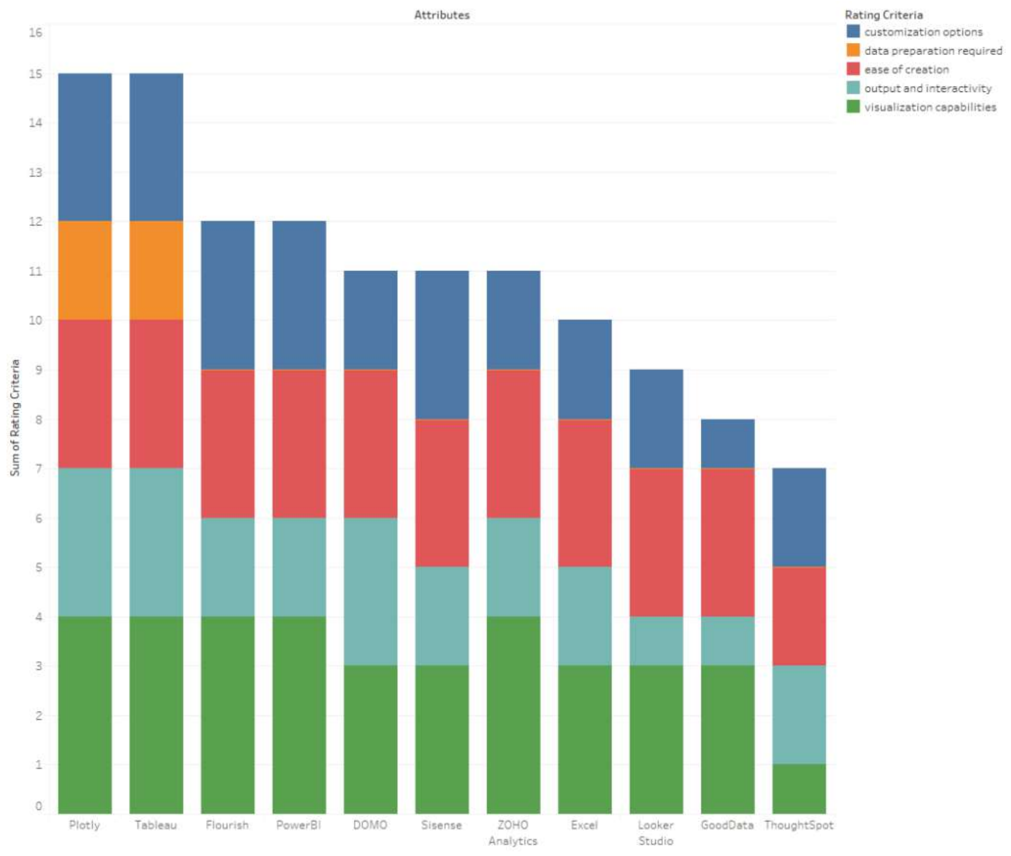


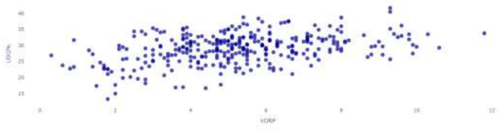
Figure 4.11: Scores of the visualization tools in **Cluster 0** for **Relation-seeking**.

When evaluating the visualization tools in **Cluster 0**, we could see that Plotly and Tableau both received high scores. They both have easy-to-use construction workflows, accurate encodings, rich tooltips, and smooth interaction, with no preprocessing needed. Flourish and Power BI form a second tier, scoring slightly lower overall but still offering a balanced mix of visual expressiveness, interactivity, and moderate effort for data handling and chart construction.

Domo Charts, Sisense, ZOHO Analytics, and Excel occupy the middle of the distribution. Their scores suggest that while they can support relation-seeking tasks, they either demand more work in data preparation or provide fewer customization options, limiting how deeply analysts can tailor views or interactions to specific questions. At the lower end, Looker Studio, GoodData, and particularly ThoughtSpot have noticeably shorter bars, driven by weaker ratings in visualization capabilities and interactivity.

Example charts generated during the evaluation can be seen in Figure 4.12. Plotly can produce interactive scatterplots with a lot of customization capabilities (see also Figure 4.12a). Plotly produces an interactive scatterplot that correctly highlights players with a Player Efficiency Rating (PER) above 25.

Although Sisense can display the relevant players, the visualization offers limited controls. Sisense is penalized for color-mapping friction and the need to prepare data. When many categories are mapped at once, the plot becomes cluttered, as seen in Figure 4.12b.


(a) Plotly

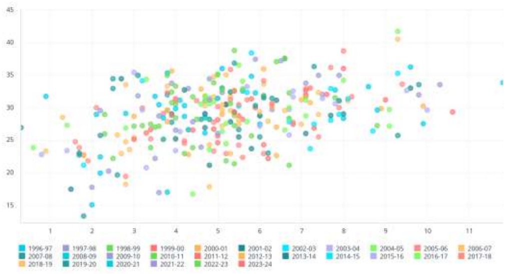

(b) Sisense

Figure 4.12: Relation-seeking (Cluster 0) examples. Comparison between scatter plots in Plotly and Sisense.

When evaluating the visualization tools in **Cluster 1**, we could see that, overall, libraries did very well. The results are shown in Figure 4.13. Bokeh, HoloViews, and Matplotlib sit at the top with the highest scores. Pygal forms a close second tier, only slightly behind the leaders, suggesting that it offers similarly strong visuals and interactivity but with modestly fewer customization options or a bit more effort in data handling.

ggplot2 and ggvis occupy the middle of the distribution. These tools score highly on visualization expressiveness but somewhat lower on ease of creation and data preparation, indicating that relation-seeking analyses are powerful but demand more coding expertise and setup. Seaborn and D3 cluster just below this middle group: Seaborn inherits good plotting defaults yet is limited by weaker customization and interactivity, while D3 provides immense potential but is penalized for its demanding data preparation and implementation effort, which constrains practical relation-seeking use for many users. At the lower end, Vega-Lite, Chart.js, and especially ApexCharts show noticeably shorter bars, driven mainly by reduced customization and interactive richness, implying that although they can visualize relationships, they offer less depth and flexibility for nuanced relation-seeking tasks compared with the leading libraries in this group.

Example charts generated during the evaluation can be seen in Figure 4.14. Without any preprocessing, HoloViews gives the users a clear, interactive plot with good export options, which can be seen in Figure 4.14a. HoloViews produces a clear, interactive scatterplot that directly highlights players with PER above 25 without any additional preprocessing and offers convenient export options.

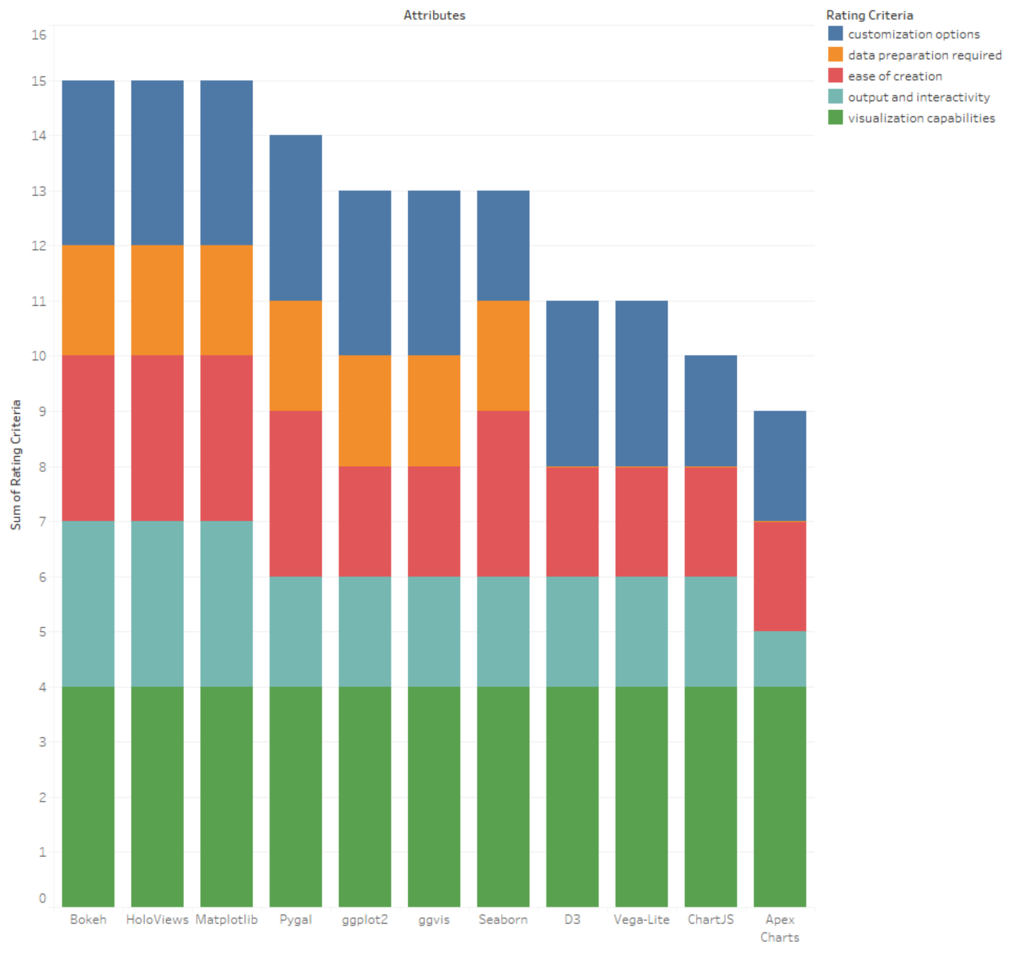
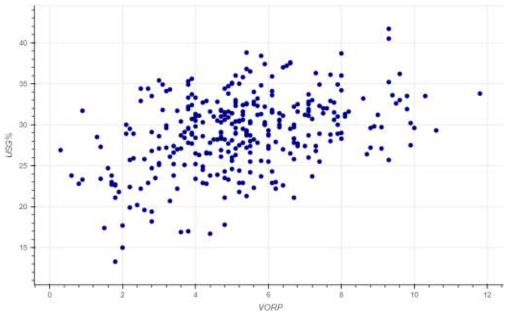


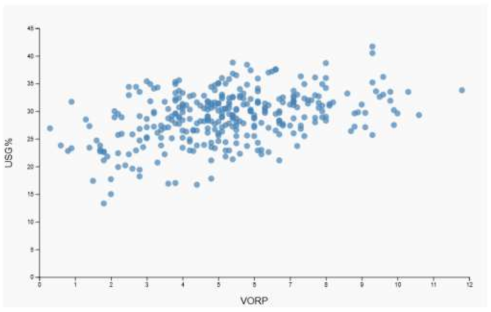
Figure 4.13: Scores of the visualization tools in **Cluster 1** for **Relation-seeking** task.

D3 gives a correct and easy-to-read result, but it needs preprocessing and does not allow for interactivity in the setup that was tested, shown in Figure 4.14b. D3 generates a correct and easy-to-read scatterplot, but in the evaluated setup the data had to be pre-aggregated and no interactive filtering or brushing was provided. ApexCharts needs preprocessing, allows for some interactivity, but does not allow for much export, so it gets a lower score, as depicted in Figure4.14c. ApexCharts is able to display the relevant players, but it requires manual preprocessing and offers only limited interaction.

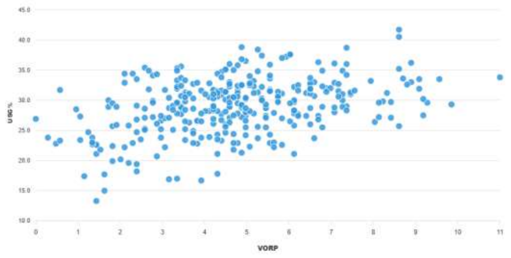
Both high-level Cluster 1 and leading Cluster 0 tools do a great job of finding relationships. When tools fail, the limiting factors are data-model constraints (aggregation) and interaction/export coverage.



(a) HoloViews



(b) D3



(c) ApexCharts

Figure 4.14: Relation-seeking (Cluster 1) examples. Comparison between scatter plots in HoloView, D3, and ApexCharts.

4.4.2 Synoptic Tasks

Synoptic tasks are defined by Andrienko and Andrienko [96] as tasks that involve subsets or the entire dataset, such as summarizing, comparing, or finding patterns over many elements at once.

Pattern Search

The Pattern Search tasks describes detecting temporal or spatial trends in the dataset. When evaluating the visualization tools in **Cluster 0**, we could see that Tableau and Plotly are receiving the highest score. The results are shown in Figure 4.15. Plotly and Tableau lead the group with the highest scores. Tableau pays a small penalty on data preparation effort compared with Plotly.

Flourish, Domo Charts, Power BI, and Sisense form a strong second tier. These tools match the leaders on visual richness but show slightly lower totals because of more constrained customization or additional work required to build and configure pattern-oriented views.

ThoughtSpot and ZOHO Analytics are located in the middle. These tools retain decent visualization and creation scores but have lower customization and, in ZOHO's case, more demanding data preparation, which makes them somewhat less flexible for complex pattern discovery workflows. Excel and GoodData fall further behind, with shorter bars driven by weaker visualization capabilities, more limited interactivity, and lower ease-of-creation scores, suggesting they are serviceable but not ideal for exploratory pattern search.

At the bottom, Looker Studio registers the lowest overall rating, indicating that its combination of visualization features, interactivity, customization, and data handling support is comparatively modest for users who need to search for intricate patterns in their data.

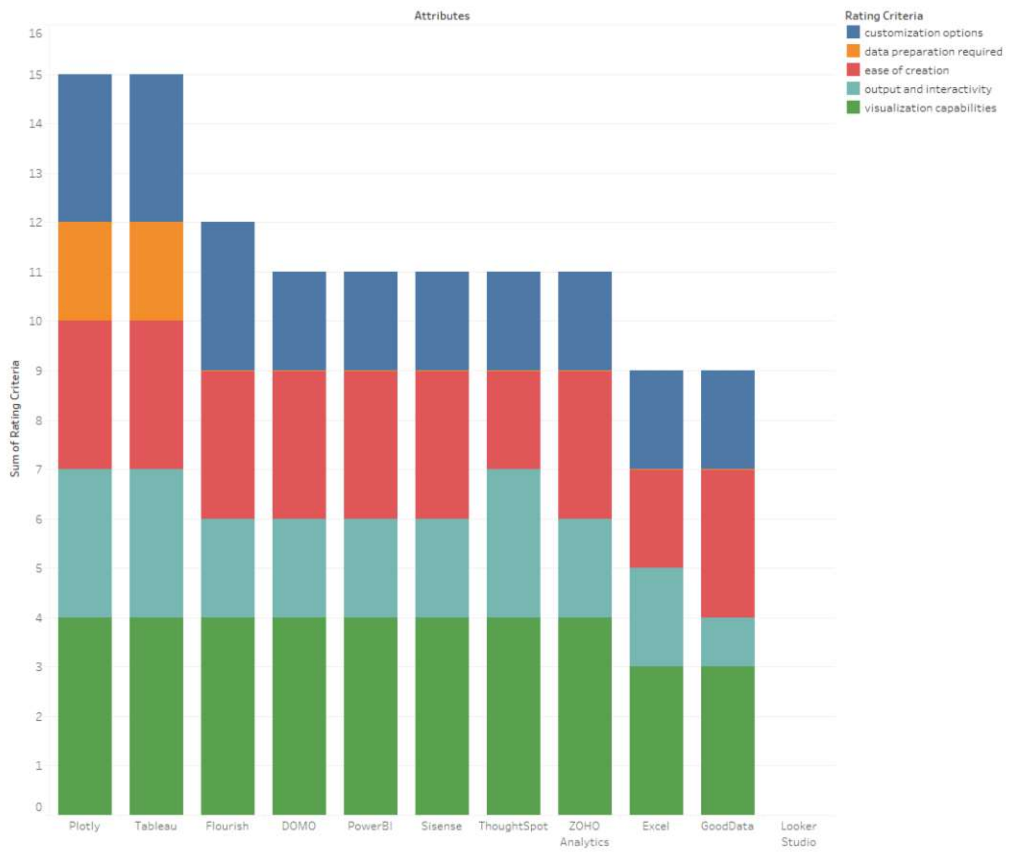
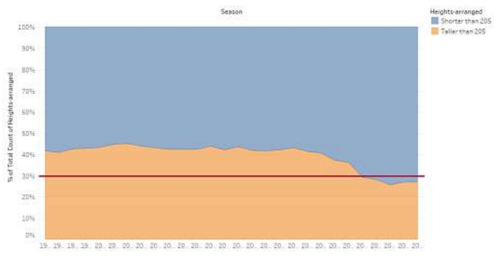
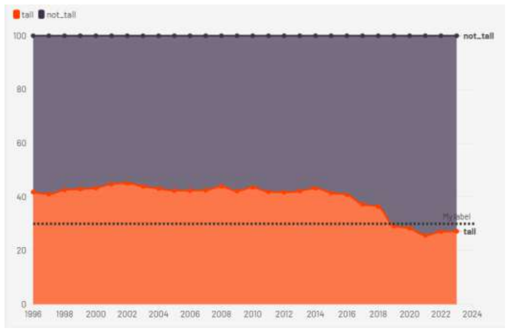


Figure 4.15: Scores of the visualization tools in **Cluster 0** for **Pattern Search**.

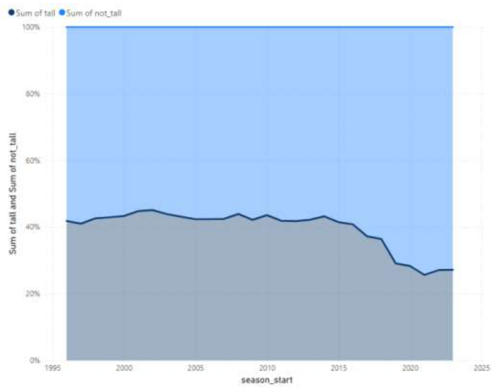
Example charts generated during the evaluation can be seen in Figure 4.16. Figure 4.16a shows the chart generated with Tableau. Tableau fully support reference line, annotation, and interaction. Flourish works well, but it needs some preprocessing, and the visualization is shown in Figure 4.16b. Flourish can reproduce the required time-series pattern and reference line, but it requires additional preprocessing of the data. Power BI supports the reference line, but it does not have the evaluated reference-line labeling and needs some cleaning steps, which lowers the score a little, as depicted in Figure 4.16c. These results resulted in lower scores for Power BI for this task.



(a) Tableau



(b) Flourish



(c) Power BI

Figure 4.16: Pattern Search (Cluster 0) examples. Comparison between area charts in Tableau, Flourish, and PowerBI.

When analyzing the visualization tools in **Cluster 1**, we could see that Python-based libraries like Seaborn did very well on all system criteria and get the highest scores. The results are shown in Figure 4.17. Bokeh, HoloViews, Matplotlib, and Pygal occupy the top tier with equally good scores. Seaborn and ggplot2 follow closely behind: they maintain strong visualization and solid interactivity but lose a little ground through slightly lower ratings for customization or more involved data preparation, suggesting that pattern search is powerful but demands more configuration or coding.

ggvis and D3 sit in the middle of the distribution. Both tools are visually expressive but show reduced ease-of-creation and data-preparation scores, reflecting the extra implementation work required before analysts can iteratively search for patterns. Chart.js and Vega-Lite fall into a lower tier, with shorter bars driven mainly by weaker customization and somewhat more limited interactivity, which constrains how finely users can tune views or combine multiple pattern-oriented displays.

In general, JavaScript-first libraries fall behind because they do not have all the features they need or because they are harder to use in the evaluation setup. At the bottom, ApexCharts has the lowest overall score despite decent visualization capabilities, because its comparatively modest customization and interaction support, together with less favorable ease-of-creation and data-preparation ratings, make it less suitable for advanced, exploratory pattern search.

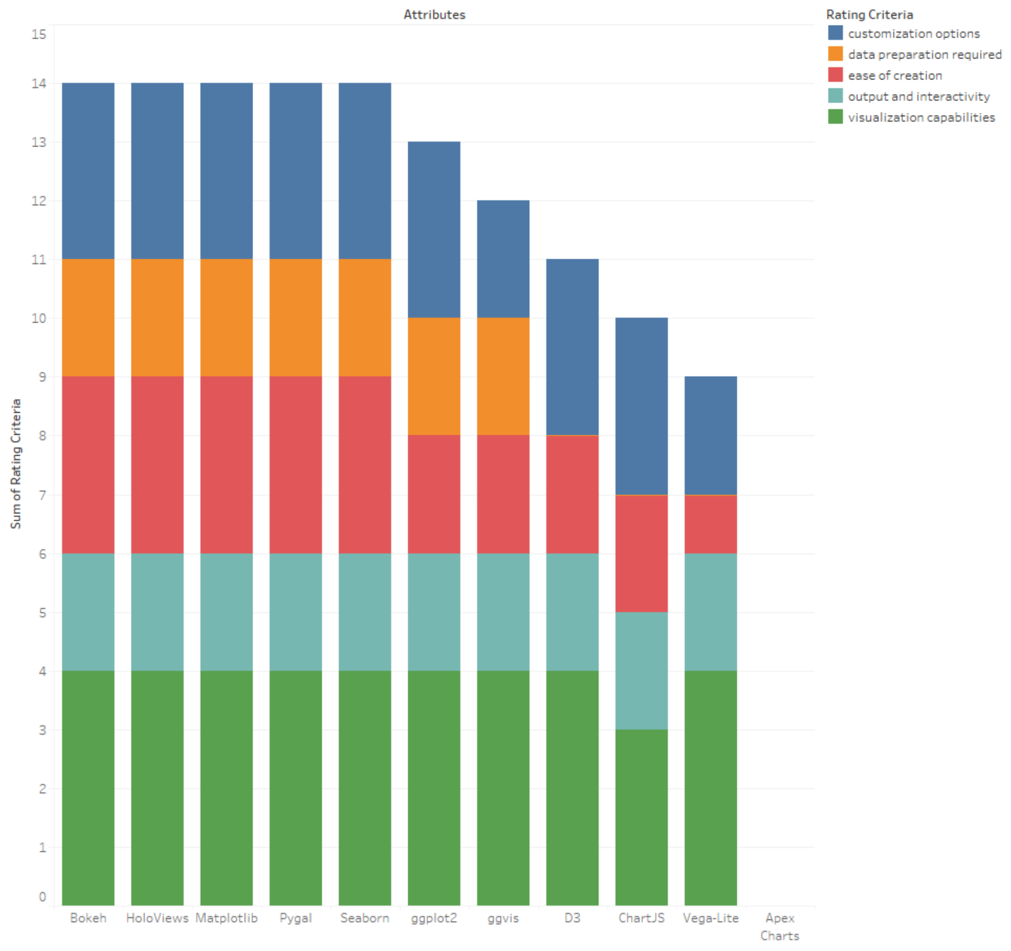
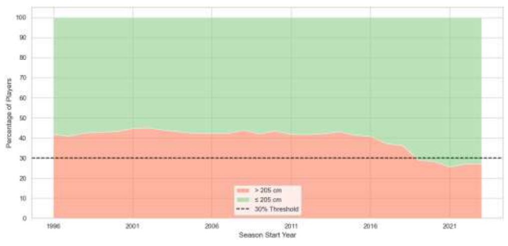


Figure 4.17: Scores of the visualization tools in **Cluster 1** for **Pattern Search**.

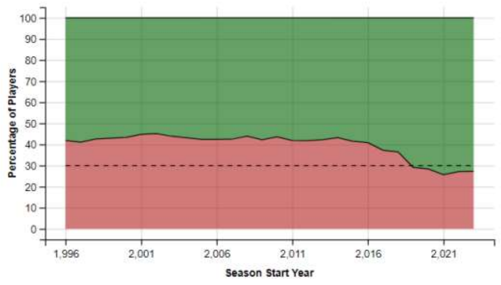
Example charts generated during the evaluation can be seen in Figure 4.18. Both area charts shown in this Figure have been created with charting libraries.

Seaborn produces a clear time-series area/line chart with the required normalization and reference line, and supports convenient styling and interaction in the evaluated setup. The results generated with Seaborn are shown in Figure 4.18a.

ggvis works well, but it loses points for having more complicated creation steps and not letting users change things as much as they want. ggvis can reproduce the required time-series pattern, but creating the visualization requires more manual steps and the available customization options are more limited.Results can be seen in Figure 4.18b.



(a) Seaborn



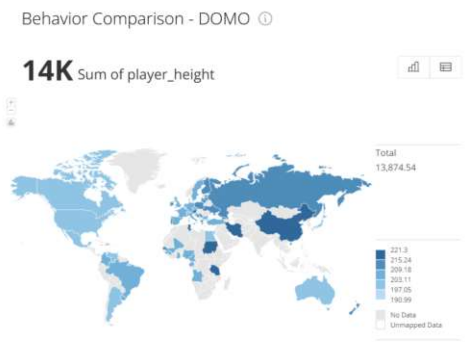
(b) ggvis

Figure 4.18: Pattern Search (Cluster 1) examples. Comparison between area charts in Seaborn and ggvis.

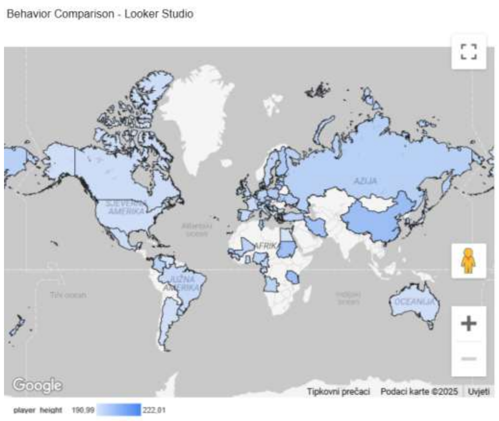
Behavior Comparison

When performing a Behavior Comparison tasks, users try to contrast patterns across groups in the dataset.

Examples for choropleth maps are shown in Figure 4.19. In this case, Domo Charts works well, but it needs to be preprocessed before geographic encoding, which lowers the score a little. Domo Charts produces a visually convincing choropleth map that correctly encodes average player height by country, but it requires substantial preprocessing before geographic encoding. The map that comes out of it is still visually convincing, as visible in Figure 4.19a. Looker Studio can generate a basic map for comparing average player height across countries, but the evaluated configuration offers limited interactivity and also depends on external preprocessing. Looker Studio gives the users a map that works but is not as interactive. The result is shown in Figure 4.19b.



(a) Domo Charts



(b) Looker Studio

Figure 4.19: Behavior Comparison (Cluster 0) examples. Comparison between choropleth maps in Domo Charts and Looker Studio.

When analyzing **Cluster 0**, we could see that, again, Tableau is the visualization tool that scores best. The results are shown in Figure 4.20. Tableau balances ease of use, map clarity, customization, and interaction. Tableau appears at the top with the tallest stacked bar, combining maximum visualization capabilities with strong customization options, high ease of creation, and solid interactive output, while its only minor weakness is a slightly higher data preparation effort.

Domo Charts and Plotly sit just below. Both tools provide robust visualization and interactivity, but Domo Charts is penalized by more demanding data preparation. Plotly trades some visualization strength for easier data handling and still offers rich customization. Excel, Flourish, Power BI, Sisense, ThoughtSpot, and ZOHO Analytics cluster in the middle. They show comparable total heights, indicating that they are generally competent for behavior comparison but tend to sacrifice either customization, interactivity, or ease of creation.

Looker Studio and GoodData occupy the lower end of the ranking with markedly shorter bars, primarily due to weaker visualization and interaction scores along with less favorable ease-of-creation ratings.

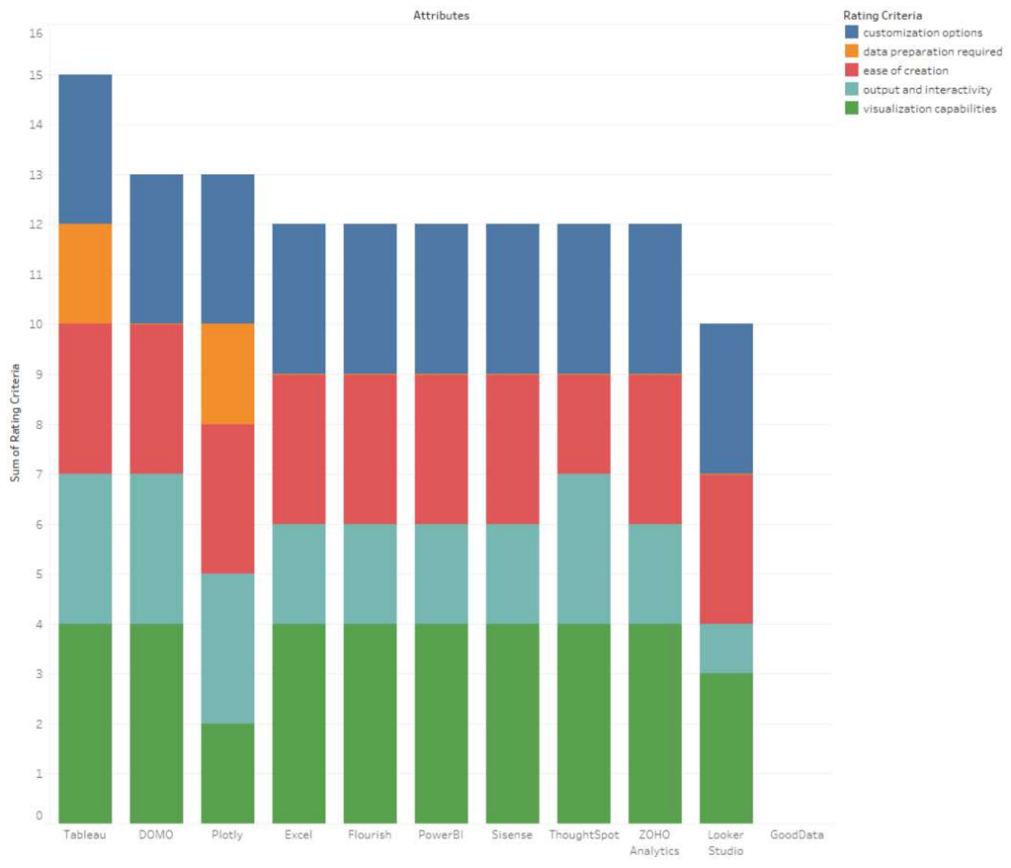


Figure 4.20: Scores of the visualization tools in **Cluster 0** for **Behavior Comparison**.

When analyzing visualization tools in Cluster 1, which contained exclusively charting libraries, we could see that this task was a difficult job for charting libraries. The results are shown in Figure 4.21. Five charting libraries could not create the needed geographic visualization under the evaluation conditions.

From the examples where it was possible (seen in Figure 4.22, we can see that ggplot2 produces a correct choropleth map that encodes average player height by country with appropriate colour mapping. Creating the visualization required multiple manual steps and the result is static. The map is shown in Figure 4.22a. However, the chart is not interactive and takes a lot of steps to make. HoloViews cannot create a heatmap-style choropleth as requested. HoloViews is unable to generate the requested choropleth and instead uses dots by coordinates, which does not match the target visualization idiom and offers limited support for comparing spatial patterns. The generated chart is shown in Figure 4.22b.

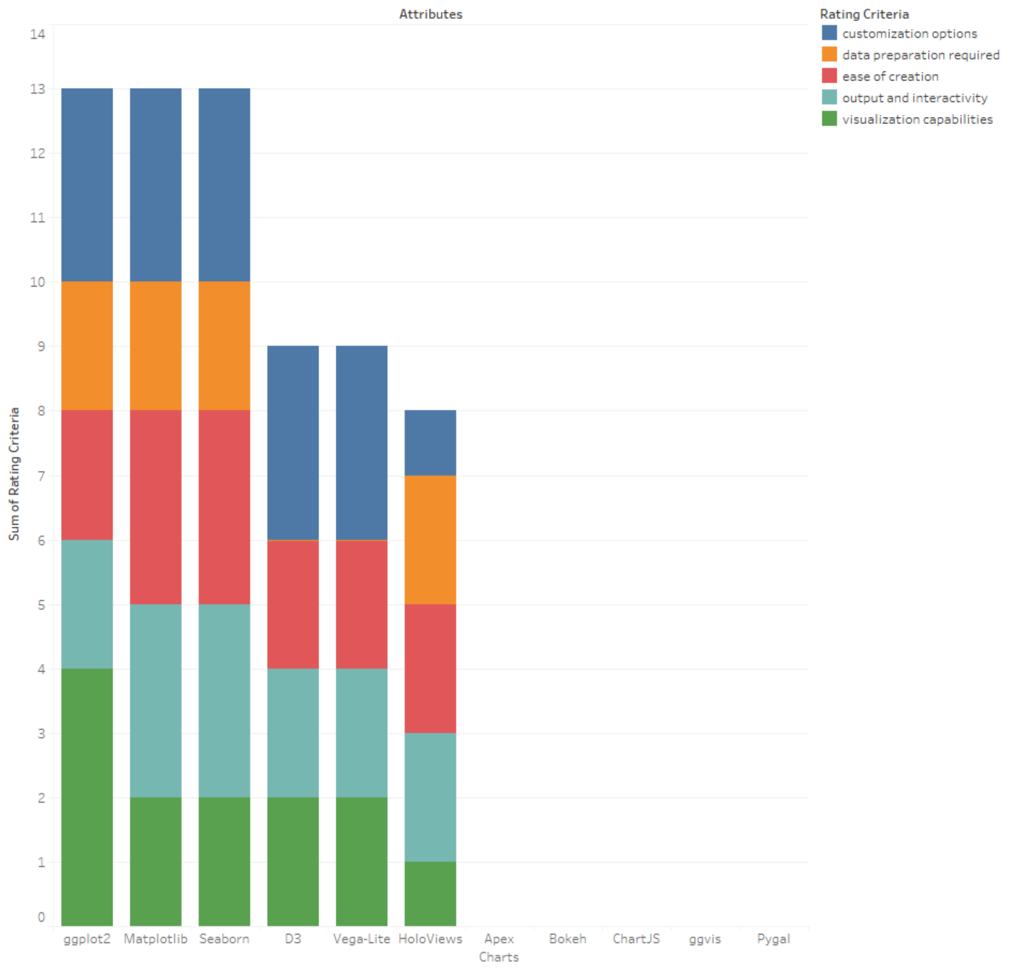
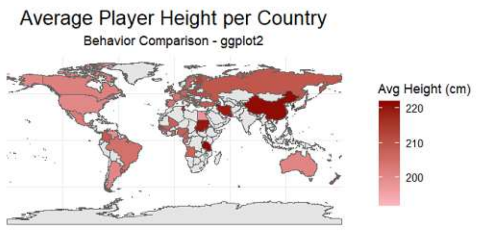
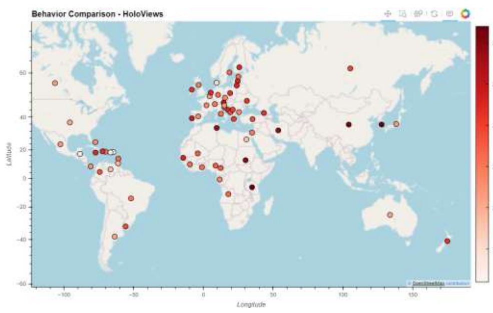


Figure 4.21: Scores of the visualization tools in **Cluster 1** for **Behavior Comparison**.


Average Player Height per Country
Behavior Comparison - ggplot2
Avg Height (cm)
220
210
200

(a) ggplot2



(b) HoloViews

Figure 4.22: Behavior Comparison (Cluster 1) examples. Comparison between choropleth maps in ggplot2 and HoloViews.

Applications still have an edge when it comes to comparing geospatial behavior because they come with built-in geocoding and map layers. Libraries, on the other hand, need additional setup and extensions to provide the same level of cartographic functionality.

4.5 User Guidelines

The task-based analysis in Section 4.4 revealed differences in the visualization tools for different tasks. We then used this information to map tasks to user groups. For user groups, we used the study by Gunklach et al. [104]. They used job advertisements and skill tables to identify nine different roles in data science. The roles are described in Section 3.4. Each role is categorized by different skills. Skills can be technical related to computer science (e.g., programming, software development, database management) or domain-specific (e.g., domain knowledge, process knowledge). In many cases statistics skills like probability, statistical modeling, and optimization are needed.

For creating guidelines, we mapped the skills required for every role to visualization tasks. For example, business-oriented skills like domain knowledge are rather related to quick tasks like Lookup or Comparison. More technical skills require more complex tasks like Pattern Search or Relation-seeking. From the compiled list of skills, we derived the importance of certain tasks for different user groups. The results can be seen in Table 4.6. The relevance of tasks for different user roles is quite diverse. We could identify some tasks, like Lookup and Behavior Comparison, that are of interest for multiple user roles. For two user roles, Software Developers and Data Science Architects, we could not identify any task that would fit their described skills.

The tasks Lookup and Comparison are primarily relevant for Business Users, Business Analysts, and Data Analysts. Relation-seeking, Pattern Search, and Behavior Comparison are of interest for Applied and Research Data Scientists, ML Engineers, Data Engineers, and Data Science Architects. The latter require stronger modeling, engineering, and architectural skills on top of analytic and domain knowledge.

Table 4.6: Data science roles mapped to the visualization tasks identified in this study. Red = high relevance, Yellow = middle relevance, Blue = low relevance.

	Business Users	Business Analysts	Data Analysts	Applied Data Scientists	Research Data Scientists	Data Engineers	ML Engineers	Software Developers	Data Science Architects
Lookup	🟥	🟥	🟥	🟦	⬜	🟦	🟦	🟦	🟦
Comparison	🟨	🟥	🟥	🟨	🟦	🟨	🟦	🟦	🟨
Relation-seeking	🟦	🟨	🟨	🟥	🟥	🟦	🟥	🟦	🟦
Pattern Search	🟦	🟦	🟨	🟥	🟥	🟨	🟥	🟨	🟦
Behavior Comparison	🟨	🟥	🟥	🟥	🟨	🟨	🟨	🟦	🟦

In summary, when mapping the relevance to the results of the task-based analysis in Section 4.4, we can summarize the following suggestions for tool usage for the different user groups. The suggestions are ranked according to the scores they received in the task-based analysis:

Business Users

1. Tableau (application)
2. Power BI (application)
3. Domo Charts (application), Sisense (application), ZOHO Analytics (application), ThoughtSpot (application)
4. Flourish (application)
5. Excel (application)

Business Analysts

1. Tableau (application)
2. Power BI (application)
3. Plotly (charting library)
4. Flourish (application)

Data Analysts

1. Tableau (application)
2. Plotly (charting library)
3. Power BI (application)
4. Matplotlib (charting library), Seaborn (charting library), ggplot2 (charting library)
5. Bokeh (charting library), HoloViews (charting library), Pygal (charting library)

Applied Data Scientists

1. Matplotlib (charting library), Seaborn (charting library), ggplot2 (charting library)
2. Plotly (charting library)
3. Bokeh (charting library), HoloViews (charting library)
4. Tableau (application)

Research Data Scientists

1. Matplotlib (charting library), Seaborn (charting library), ggplot2 (charting library)
2. Plotly (charting library)
3. Bokeh (charting library), HoloViews (charting library), Pygal (charting library)
4. D3 (charting library)

Data Engineers

No strong relevance for tasks or tools. According to the *middle* relevance for Comparison, Pattern Search, and Behavior Comparison tasks, visualization tools like Power BI (application) or Tableau (application) can be recommended.

ML Engineers

1. Matplotlib (charting library), Seaborn (charting library), ggplot2 (charting library)
2. Plotly (charting library)
3. Bokeh (charting library), HoloViews (charting library), Pygal (charting library)
4. Tableau (application), Power BI (application)

Software Developers

No strong relevance for tasks or tools. According to the *middle* relevance for Pattern Search tasks, visualization tools like Tableau (application) or Plotly (charting library) can be recommended.

Data Science Architects

No strong relevance for tasks or tools. According to the *middle* relevance for Comparison tasks, visualization tools like Plotly (charting library) or HoloViews (application) can be recommended.

CHAPTER 5

Discussion and Conclusion

In this thesis, we showed how the world of Visual Analytics has changed in the recent years, with many visualization tools offering different functionalities, use-cases, and possibilities to visualize data and to transform it into something meaningful. It can be expected that in the upcoming years this field will grow much more, especially in the field of LLM. Dashboards generated from the natural language will probably become more prevalent (such as the ones currently employed in Sisense), and the process of introducing the non-technical users will become easier than it has ever been.

5.1 Alignment with Research Questions and Goals

The thesis was based on three research question, and the aim to deliver four goals. The research questions and goals were outlined in Section 1.2 and Section 1.3. In this section, the results are aligned with the intended research directions.

5.1.1 RQ1: What is the current landscape of visualization tools?

We evaluated 22 visualization tools in the study. 10 were applications and 12 were charting libraries. Taken together, Tableau and Plotly could be identified as the current high-end application and charting library on the market. Other higher rated visualization tools were Bokeh (charting library), HoloViews (application), Matplotlib (charting library), and Pygal (charting library). The Python charting libraries clearly outstand in contrast to other, e.g., JavaScript-based libraries.

The field of mid-tier tools is larger than the top field. Mid-tier tools include Power BI (application), Flourish (application), Domo Charts (application), Sisense (application), ZOHO Analytics (application), Excel (application), ggplot2 (charting library), Seaborn (charting library), ggvis (charting library), and D3 (charting library). These tools typically excel in one or two dimensions (e.g., visual expressiveness or ease of creation), while lagging in others, especially customization convenience, interaction, and required preprocessing.

At the lower end, tools such as Looker Studio (application), GoodData (application), ThoughtSpot (application), Chart.js (charting library), Vega-Lite (charting library), and ApexCharts (charting library) can handle basic charting and simple exploratory questions. These tools offer less flexibility, weaker interaction, or more limited visual encodings.

In summary, the landscape is still bimodal. Applications focus on end-to-end workflows (connectors, data preparation, collaboration), while charting libraries focus on expressiveness, programmability, and close integration with analytical code ecosystems. There are hybrid tools, but most of the time, clusters still behave like two separate things.

With the complete review of the tool landscape, the goal **G1: General picture of the visualization tool landscape** could be fulfilled.

5.1.2 RQ2: What commonalities and differences exist between visualization tools?

The clustering method we created proved that tools can be really divided into 2 groups - a no-code, drag-and-drop Business Applications, and the code-dependent, highly configurable charting libraries. Understanding their differences is crucial when picking which kind of tool is needed for the certain visualization. Even though Plotly was predicted as Business Application, which is not true, another study relevant to this thesis also got the same results, so we can interpret this charting library as an anomaly. Clustering and PCA show a clear divide between tools that focus on presentation and those that focus on code. The first principal component is related to how easy it is to use and interact with something, and the second is related to how customizable and extensible it is.

Applications and Plotly (Cluster 0) work best in situations where the output's aesthetic appeal is important and where a high degree of interactivity can greatly improve the user experience. These kinds of tools work especially well for storytelling, presentations, and exploratory analysis, where dynamic, captivating visuals can encourage audience participation and deeper understanding. Charting libraries (Cluster 1), on the other hand, are more appropriate for situations that call for more intricate analytical features and smooth integration into pre-existing digital environments, like websites or unique applications. For developers and analysts looking to incorporate sophisticated visualizations into more comprehensive data-driven solutions, their adaptability, scalability, and compatibility with a range of programming frameworks make them a pragmatic option.

With the comparative analysis, the goal **G2: Analysis and comparison of visualization tool features** could be fulfilled.

The task-based analysis revealed differences when focussing on specific tasks. Lookup tasks are broadly well covered, since most tools and libraries could retrieve and display specific values or items. The main differences came from how easily users can customize the view and how interactive the lookup experience is. Comparison tasks are similarly well supported, but here richer customization and more expressive visual encodings start to matter more. The Relation-seeking task placed higher demands on interactivity (filtering, linking, brushing) and on the ability to combine multiple variables in a single or coordinated view. As a result, general-purpose applications and lighter-weight charting libraries slip behind, while tools that provide strong interaction models and multivariate views stand out. The Pattern Search task was well suited for tools that make it easy to iterate quickly over many views and handle more elaborate data preparation pipelines, which tended to favor the more powerful applications. Behavior Comparison sits somewhat between Comparison and Pattern Search, since this task required comparing how measures evolve across time, categories, or scenarios. Tools that can easily create and customize small multiples, layered time series, or trend views achieved higher scores. To summarize:

- Lookup and Comparison: Well covered by the tools. Applications get polished outputs faster, while charting libraries are better at controlling specifications and (for Comparison) often get the highest system scores.
- Relation-seeking: Gaps where tools force aggregation or limit interactivity and export.
- Pattern Search: Almost well covered. Differences in data preprocessing and annotation.
- Behavior Comparison: Applications as clear leaders, because they have integrated geospatial pipelines. Libraries often need extra packages or custom code to be equal.

With the comparative analysis, the goal **G3: Task-based evaluation of visualization tools** could be fulfilled.

5.1.3 RQ3: What evidence-based guidelines can be developed to support users in integrating visualization tools into their analytical workflows?

We aligned data science roles and their skills with the visualization tasks we used in the study. Based in the analysis we can conclude that Business Users and Business Analysts should primarily work in applications, since they offer more intuitive front-ends. More technical roles like Data Scientists, ML Engineers, and Data Engineers, should center their work in charting libraries, primarily Plotly or other Python libraries. Applications can be used for publishing layers. Data Engineers, Software Developers, and Data Science Architects, who work very closely on the technical data infrastructure, will hardly benefit from the usage of visualization tools.

With the user-based analysis, the goal **G4: Guidelines for visualization tool usage** could be fulfilled.

5.2 Discussion and Limitation

An extensive study on visualization tools also has limitations. First of all, the scope of this thesis was inherently constrained by the choice of visualization tools, visualization types, and dataset employed. These components do not adequately represent the variety of methods, strategies, and datasets found in the larger visual analytics ecosystem, even though they were chosen to guarantee methodological clarity and viability. This means that while the results offer valuable and useful information about the effectiveness and usability of the systems under evaluation, they might not be universally applicable to all user types, application domains, or data complexity levels. This restriction emphasizes the necessity of exercising caution when extrapolating the findings to situations outside of the scope of this thesis.

Additionally, even though reproducibility is emphasized in the experimental design, some factors might still have an impact on the consistency of results over time. Potential variability in replication efforts is introduced by subjective aspects of grading as well as the ongoing development of the chosen tools and their features. Furthermore, it is impossible to completely rule out the possibility of bias resulting from participants' past familiarity with the systems under evaluation, since this familiarity may affect performance or perception in ways that are not evenly distributed throughout the sample. These factors emphasize how crucial it is to interpret the findings within the precise methodological and contextual parameters established by this investigation.

5.3 Future Work

Future work may expand the range of tools examined in this thesis. The selection in this study was limited to a feasible number of visualization applications and charting libraries that were actively maintained and available during the research period. Because the ecosystem is changing so quickly, it would be helpful to update the final list every now and then to include new tools and major version changes to old ones. Additionally, later research may incorporate more specialized tools, such as domain-specific visualization systems in healthcare, finance, or geospatial analysis, to examine whether the identified patterns extend beyond general-purpose applications and charting libraries. A long-term view could then show how capabilities, user guidance, and analytic integration change over time.

There are a number of methodological extensions that could be made. The comparative feature analysis in this thesis utilized binary coding of functionalities and employed unsupervised clustering techniques, specifically K-Means and PCA, for exploratory purposes. Future endeavors may incorporate more nuanced feature representations, including ordinal or weighted encodings that differentiate between fundamental and sophisticated implementations of the same function, or between native and workaround-based support. To see how strong the tool groups are, future work could also try different clustering and embedding methods, like hierarchical clustering or non-linear dimensionality reduction. Additionally, semi-automated feature extraction from documentation, APIs, or configuration files could diminish manual labor and potential bias in coding.

Finally, the evaluation based on tasks could be made more comprehensive and thorough. This thesis examined five representative elementary and synoptic tasks implemented on a singular NBA dataset, assessed from the viewpoint of an expert user replicating visualizations. Future work may integrate diverse task categories (e.g., collaboration, narrative construction, or model guidance), various datasets spanning multiple domains, and regulated user studies involving participants from specific target demographics, including data scientists, business analysts, and lay decision-makers. Researchers would be able to link feature-level capabilities more directly to user performance, error rates, and subjective experience. Together, these extensions would give a better and more detailed picture of how modern visualization tools help with real-world analytical work.

CHAPTER 6

Overview of Generative AI Tools Used

I am declaring that GenAI (Perplexity, ChatGPT) was used to familiarize myself with a syntax and quick examples of generating visualizations from the tools that I had not used before writing the thesis.

List of Figures

List of Tables

Bibliography

[1] A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, *Information Visualization*. Springer Berlin, Heidelberg, 2008.

[2] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, *Mastering the Information Age: Solving Problems with Visual Analytics*. Eurographics Association, 2010.

[3] T. Munzner, *Visualization Analysis and Design*. A K Peters/CRC Press, 2014.

[4] R. C. Roberts, "Visualising Business Data: A Survey," *Information*, vol. 9, no. 11, p. 285, 2018.

[5] B. Preim and K. Lawonn, "A Survey of Visual Analytics for Public Health," *Computer Graphics Forum*, vol. 39, no. 1, pp. 1–25, 2020.

[6] X. Tan, X. Suo, W. Li, L. Bi, and F. Yao, "Data Visualization in Healthcare and Medicine: A Survey," *The Visual Computer*, vol. 41, pp. 3037–3058, 2024.

[7] R. Damaševičius, J. Toldinas, A. Venčkauskas, v. Grigaliūnas, N. Morkevičius, and V. Jukavičius, "Visual Analytics for Cyber Security Domain: State-of-the-Art and Challenges," *Electronics*, vol. 8, no. 10, p. 1150, 2019.

[8] J. Wang, S. Hazarika, C. Li, and H.-W. Shen, "Visualization and Visual Analysis of Ensemble Data: A Survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 9, pp. 2853–2872, 2019.

[9] M. Behrisch, D. Streeb, F. Stoffel, D. Seebacher, B. Matejek, S. Weber, S. Mittelstädt, H. Pfister, and D. Keim, "Commercial Visual Analytics Systems: Advances in the Big Data Analytics Field," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 10, pp. 33 011–3031, 2018.

[10] L. C. Rost, "What I Learned Recreating One Chart Using 24 Tools," https://source.opennews.org/articles/what-i-learned-recreating-one-chart-using-24-tools/, 2016, [Accessed 2025-05-14].

[11] S. K. Card, J. D. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.

[12] R. Mazza, *Introduction to Information Visualization: A Systematic Approach.* Springer Science+Business Media, 2009.

[13] C.-h. Chen, W. Härdle, and A. Unwin, *Handbook of Data Visualization.* Springer Berlin Heidelberg, 2008.

[14] W. S. Cleveland and R. McGill, “Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods,” *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 531–554, 1984.

[15] L. Wilkinson, *The Grammar of Graphics.* 978-0-387-98774-3, 1999.

[16] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, “Vega-Lite: A Grammar of Interactive Graphics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 341–350, 2017.

[17] M. Ward, G. G. Grinstein, and D. Keim, *Interactive Data Visualization: Foundations, Techniques, and Application.* A K Peters, 2010.

[18] B. Shneiderman, “The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations,” in *Proceedings of the 1996 IEEE Symposium on Visual Languages*, ser. VL ’96, Boulder, CO, USA, Sep. 3-6 1996, p. 336.

[19] J. S. Yi, Y. a. Kang, J. Stasko, and J. Jacko, “Toward a Deeper Understanding of the Role of Interaction in Information Visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1224–1231, 2007.

[20] L. Wilkinson and M. Friendly, “The History of the Cluster Heat Map,” *The American Statistician*, vol. 63, no. 2, pp. 179–184, 2009.

[21] Atlassian, “A Complete Guide to Heatmaps,” https://www.atlassian.com/data/charts/heatmap-complete-guide, 2025, [Accessed 2025-07-09].

[22] Z. Gu, “Complex heatmap visualization,” *iMeta*, vol. 1, no. 3, p. e43, 2022.

[23] T. Gschwandtner and O. Erhart, “Know Your Enemy: Identifying Quality Problems of Time Series Data,” in *Proceedings of the 2018 IEEE Pacific Visualization Symposium*, ser. PacificVis ’18, Kobe, Japan, Apr. 10-13 2018, pp. 205–214.

[24] D. Keim, “Designing Pixel-Oriented Visualization Techniques: Theory and Applications,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 59–78, 2000.

[25] P. P. Rodrigues and J. Gama, “A Simple Dense Pixel Visualization for Mobile Sensor Data Mining,” in *Proceedings of the 2nd International Conference on Knowledge Discovery from Sensor Data*, ser. SensorKDD ’08, 2008, pp. 175–189.

[26] R. Borgo, J. Kehrer, D. H. S. Chung, E. Maguire, R. S. Laramee, H. Hauser, M. Ward, and M. Chen, “Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications,” in *Proceedings of Eurographics State-of-the-Art Reports*, 2013.

[27] M. Brehmer, R. Kosara, and C. Hull, “Generative Design Inspiration for Glyphs with Diatoms,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 389–399, 2022.

[28] FlowingData, “Horizon Graphs, with a Food Pricing Example,” https://flowingdata.com/2015/07/02/changing-price-of-food-items-and-horizon-graphs/, 2015, [Accessed 2025-08-09].

[29] M. Dahnert, A. Rind, W. Aigner, and J. Kehrer, “Looking beyond the horizon: Evaluation of four compact visualization techniques for time series in a spatial context,” arXiv:1906.07377, 2019. [Online]. Available: https://arxiv.org/abs/1906.07377

[30] D. Braun, R. Borgo, M. Sondag, and T. von Landesberger, “Reclaiming the Horizon: Novel Visualization Designs for Time-Series Data with Large Value Ranges,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 01, pp. 1161–1171, 2024.

[31] A. Inselberg, “The plane with parallel coordinates,” *The Visual Computer*, vol. 1, pp. 69–91, 1985.

[32] J. Heinrich and D. Weiskopf, “State of the Art of Parallel Coordinates,” in *Proceedings of Eurographics State of the Art Reports*, ser. EG ’13, Girona, Spain, May 6-10 2013.

[33] Highsoft, “Highcharts Visualizations: Parallel Coordinates,” https://www.highcharts.com/demo/highcharts/parallel-coordinates, 2025, [Accessed 2025-10-01].

[34] H. Zhou, X. Yuan, H. Qu, W. Cui, and B. Chen, “Visual clustering in parallel coordinates,” *Computer Graphics Forum*, vol. 27, no. 3, pp. 1047–1054, 2008.

[35] G. Palmas, M. Bachynskyi, A. Oulasvirta, H. P. Seidel, and T. Weinkauf, “An Edge-Bundling Layout for Interactive Parallel Coordinates,” in *Proceedings of the IEEE Pacific Visualization Symposium*, ser. PacificVis ’14, Yokohama, Japan, Mar. 4-7 2014, pp. 57–64.

[36] K. Dang, “Sunburst chart to visualize complex hierarchical data,” https://medium.com/@kirudang/sunburst-chart-to-visualize-complex-hierarchical-data-20cd5a2a308a, 2023, [Accessed 2025-07-09].

[37] A. Taylor, K. McLeod, C. Armit, R. Baldock, and A. Burger, “Visualization of gene expression information within the context of the mouse anatomy,” arXiv:1407.2117, 2014.

[38] L. Woodburn, Y. Yang, and K. Marriott, “Interactive Visualisation of Hierarchical Quantitative Data: An Evaluation,” in *Proceedings of the IEEE Visualization Conference*, ser. VIS ’19, Vancouver, BC, Canada, Oct. 20-25 2019, pp. 96–100.

[39] P. Rastogi, K. Singh, and J. Sreevalsan-Nair, “SunburstChartAnalyzer: Hierarchical Data Retrieval from Images of Sunburst Charts for Tree Visualization,” in *Proceedings of 41st Computer Graphics & Visual Computing Conference*, ser. CGVC ’23, Aberystwyth, Wales, UK, Sep. 14-15 2023.

[40] B. Johnson and B. Shneiderman, “Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures,” in *Proceedings of Visualization*, ser. VIS ’91, San Diego, CA, USA, Oct. 22-25 1991.

[41] P. Laubheimer, “Treemaps: Data Visualization of Complex Hierarchies,” https://www.nngroup.com/articles/treemaps/, 2019.

[42] P. C.-I. Pang, R. P. Biuk-Aghai, S. Fong, and Y.-W. Si, “An Experimental Comparison of Map-like Visualisations and Treemaps,” arXiv:1909.09351, 2019. [Online]. Available: https://arxiv.org/abs/1909.09351

[43] E. E. Firat and R. Denisova, Alena Laramee, “Treemap Literacy: A Classroom-Based Investigation,” in *Proceedings of the Eurographics Workshop on Education Papers*, Norrköping, Sweden, May 25-29 2020, pp. 29–38.

[44] Y. Tu and H.-W. Shen, “Visualizing changes of hierarchical data using treemaps,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, p. 1286–1293, 2007.

[45] E. Vernier, M. Sondag, J. Comba, B. Speckmann, A. Telea, and K. Verbeek, “Quantitative Comparison of Time-Dependent Treemaps,” *Computer Graphics Forum*, vol. 39, no. 3, pp. 393–404, 2020.

[46] V. Filipov, A. Arleo, and S. Miksch, “Are We There Yet? A Roadmap of Network Visualization from Surveys to Task Taxonomies,” *Computer Graphics Forum*, vol. 42, no. 6, p. e14794, 2023.

[47] K. Sankaran, “Node – Link Diagrams,” https://krisrs1128.github.io/stat479/posts/2021-03-06-week8-2/, 2021, [Accessed 2025-08-05].

[48] M. Okoe, R. Jianu, and S. Kobourov, “Revisited Experimental Comparison of Node-Link and Matrix Representations,” in *Proceedings of the 25th International Symposium on Graph Drawing and Network Visualization*, ser. GD ’17, Boston, MA, USA, Sep. 25-27 2017, pp. 287–302.

[49] F. Beck and S. Diehl, “Visual Comparison of Software Architectures,” in *Proceedings of the 5th International Symposium on Software Visualization*, ser. SoftVis ’10, Salt Lake City, UT, USA, Oct. 25-26 2010, pp. 183–192.

[50] M. Abdelaal, N. D. Schiele, K. Angerbauer, K. Kurzhals, M. Sedlmair, and D. Weiskopf, "Comparative Evaluation of Bipartite, Node-Link, and Matrix-Based Network Representations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 896–906, 2023.

[51] K. Wada, G. Wallner, and S. Vos, "Studying the Utilization of a Map-Based Visualization with Vitality Datasets by Domain Experts," *Geographies*, vol. 2, no. 3, pp. 379–396, 2022.

[52] K. L. T. Fung, S. T. Perrault, and M. T. Gastner, "Effectiveness of Area-to-Value Legends and Grid Lines in Contiguous Area Cartograms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 8, pp. 4631–4647, 2024.

[53] M. Nöllenburg, *Geographic Visualization.* Springer Berlin Heidelberg, 2007, pp. 257–294.

[54] J. Heer and B. Shneiderman, "Interactive dynamics for visual analysis," *Communications of the ACM*, vol. 55, no. 4, p. 45–54, 2012.

[55] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, and A. Fernández-Leal, "Human-in-the-Loop Machine Learning: A State of the Art," *Artificial Intelligence Review*, vol. 56, no. 4, pp. 3005–3054, 2023.

[56] A. Endert, W. Ribarsky, C. Turkay, B. L. W. Wong, I. T. Nabney, I. D. Blanco, and F. Rossi, "The State of the Art in Integrating Machine Learning into Visual Analytics," *Computer Graphics Forum*, vol. 36, no. 8, p. 458–486, 2017.

[57] P. Pirolli and S. Card, "The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis," in *Proceedings of the International Conference on Intelligence Analysis*, McLean, VA, USA, May 2-6 2005.

[58] J. C. Roberts, "State of the Art: Coordinated & Multiple Views in Exploratory Visualization," in *Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, ser. CMV '07, Zurich, Switzerland, July 2 2007, pp. 61–71.

[59] D. Ceneda, T. Gschwandtner, T. May, S. Miksch, H.-J. Schulz, M. Streit, and C. Tominski, "Characterizing Guidance in Visual Analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 111–120, 2017.

[60] J. Kehrer and H. Hauser, "Visualization and Visual Analysis of Multifaceted Scientific Data: A Survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 3, pp. 495–513, 2013.

[61] G. Shurkhovetskyy, A. Bahey, and M. Ghoniem, "Visual analytics for network security," in *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, ser. VAST '12, Seattle, WA, USA, Oct. 14-19 2012, pp. 301–302.

[62] N. Pezzotti, B. P. F. Lelieveldt, L. v. d. Maaten, T. Höllt, E. Eisemann, and A. Vilanova, “Approximated and User Steerable tSNE for Progressive Visual Analytics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 7, pp. 1739–1752, 2017.

[63] Z. Liu and J. Heer, “The Effects of Interactive Latency on Exploratory Visual Analysis,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2122–2131, 2014.

[64] J.-D. Fekete, D. Fisher, and M. Sedlmair, *Progressive Data Analysis.* The Eurographics Association, 2024.

[65] J. Hullman and N. Diakopoulos, “Visualization Rhetoric: Framing Effects in Narrative Visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2231–2240, 2011.

[66] R. Schuster, K. Gregory, T. Möller, and L. Koesten, “"Being Simple on Complex Issues" – Accounts on Visual Data Communication About Climate Change,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 9, pp. 6598–6611, 2024.

[67] P. Isenberg, N. Elmqvist, J. Scholtz, D. Cernea, K.-L. Ma, and H. Hagen, “Collaborative Visualization: Definition, Challenges, and Research Agenda,” *Information Visualization*, vol. 10, no. 4, pp. 310–326, 2011.

[68] A. Chatzimparmpas, K. Kucher, and A. Kerren, “Visualization for Trust in Machine Learning Revisited: The State of the Field in 2023,” *IEEE Computer Graphics and Applications*, vol. 44, no. 3, pp. 99–113, 2024.

[69] A. Chatzimparmpas, R. Martins, I. Jusufi, K. Kucher, F. Rossi, and A. Kerren, “The State of the Art in Enhancing Trust in Machine Learning Models with the Use of Visualizations,” *Computer Graphics Forum*, vol. 39, pp. 713–756, 2020.

[70] Y. Cui, L. W. Ge, Y. Ding, L. Harrison, F. Yang, and M. Kay, “Promises and Pitfalls: Using Large Language Models to Generate Visualization Items,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, no. 1, pp. 1094–1104, 2025.

[71] B. H. Thomas, “Virtual Reality for Information Visualization Might Just Work This Time,” *Frontiers in Robotics and AI*, vol. 6, p. 84, 2019.

[72] B. Bach, R. Dachselt, S. Carpendale, T. Dwyer, C. Collins, and B. Lee, “Immersive Analytics: Exploring Future Interaction and Visualization Technologies for Data Analytics,” in *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*, ser. ISS ’16, Niagara Falls, ON, Canada, Nov. 6-9 2016, pp. 529–533.

[73] V. V. Dogadina and A. V. Voronin, “Comparative Analysis of Data Visualization Tools,” *International Journal of Computing, Programming and Database Management*, vol. 5, no. 1, pp. 49–51, 2024.

[74] R. M. Parthe, “Comparative Analysis of Data Visualization Tools: Power BI and Tableau,” *Indian Scientific Journal of Research in Engineering and Management*, vol. 7, no. 10, 2023.

[75] A. Lousa, I. Pedrosa, and J. Bernardino, “Evaluation and Analysis of Business Intelligence Data Visualization Tools,” in *Proceedings of the 14th Iberian Conference on Information Systems and Technologies*, ser. CISTI ’19, Coimbra, Portugal, June 19-22 2019, pp. 1–6.

[76] A. Sabahath, A. Begum, and M. Aisha, “Understanding the Role of Data Visualization in Modern Business Applications,” *Advances in Business Information Systems and Analytics Book Series*, pp. 29–44, 2024.

[77] D. Liu, “Business Intelligence Tools for Data Extractions Transformations and Visualizations,” in *Proceedings of the International Symposium on Networks, Computers and Communications*, ser. ISNCC ’24, Washington DC, USA, Oct. 22-25 2024, pp. 1–4.

[78] H. Huang, S. Ravi, T. Warrington, H. Cui, C. Wang, M. McCreary, B. Lauffer, and C. Lu, “A Comparison of Data Visualization Tools: A Case Study in Health-Related Research,” *Information Visualization*, vol. 24, no. 1, 2024.

[79] J. Schmidt, “Usage of Visualization Techniques in Data Science Workflows,” in *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3*, ser. VISIGRAPP ’20, Valletta, Malta, Feb. 27-29 2020, pp. 309–316.

[80] A. Batch and N. Elmqvist, “The Interactive Visualization Gap in Initial Exploratory Data Analysis,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 278–287, 2018.

[81] I. T. Jolliffe, *Principal Component Analysis.* Springer New York, NY, 2011.

[82] K. Pearson, “LIII. On lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[83] H. Hotelling, “Analysis of a complex of statistical variables into principal components,” *Journal of Educational Psychology*, vol. 24, pp. 417–441, 498–520, 1933.

[84] K. Diamantaras, *Principal component neural networks theory and applications.* Wiley, 1996.

[85] B. Flury, "Common Principal Components and Related Multivariate Models," *Journal of the Royal Statistical Society Series C: Applied Statistics*, vol. 40, no. 1, pp. 180–181, 1991.

[86] M. Hallin, D. Paindaveine, and T. Verdebout, "Efficient R-estimation of principal and common principal components," *Journal of the American Statistical Association*, vol. 109, no. 2, pp. 1071–1083, 2014.

[87] Lu, Hugo, "Every single BI Tool Ever Ranked," https://medium.com/@hugolu87/every-single-bi-tool-ever-ranked-95630f92d7e6, 2024, [Accessed 2025-05-19].

[88] Pyramid Analytics, "Pyramid Named in 2024 Gartner Magic Quadrant for Analytics and BI Platforms," https://www.pyramidanalytics.com/gartner-2024-magic-quadrant-only/, 2024, [Accessed 2025-05-19].

[89] Gartner, Inc., "Magic Quadrant for Analytics and Business Intelligence Platforms," Gartner Report Document No. 5519595, 2024, [Accessed 2025-05-02].

[90] Atlassian, "Announcing Our Acquisition of Chartio," https://www.atlassian.com/blog/announcements/atlassian-acquires-chartio, 2021, [Accessed 2025-05-19].

[91] J. Wu, *Advances in K-means Clustering.* 978-3-642-29806-6, 2012.

[92] C. A. Willard, *Statistical Methods.* 978-0-367-20351-1, 2020.

[93] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "A local search approximation algorithm for k-means clustering," *Computational Geometry*, vol. 28, no. 2, pp. 89–112, 2004.

[94] R. A. Amar, J. R. Eagan, and J. T. Stasko, "Low-level components of analytic activity in information visualization," in *Proceedings of the IEEE Symposium on Information Visualization*, ser. INFOVIS '05, Minneapolis, MN, USA, Oct. 23-25 2005, pp. 111–117.

[95] M. Brehmer and T. Munzner, "A Multi-Level Typology of Abstract Visualization Tasks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2376–2385, 2013.

[96] N. Andrienko and G. Andrienko, *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach.* Springer Berlin, Heidelberg, 2006.

[97] NBA Media Ventures, LLC., "The official site of the NBA for the latest NBA Scores, Stats & News," https://www.nba.com/stats/players/bio, 2025, [Accessed 2025-01-04].

[98] Sports Reference, "Basketball Reference," https://www.basketball-reference.com/, 2025, [Accessed 2025-01-04].

[99] S. L. Franconeri, L. M. Padilla, P. Shah, J. M. Zacks, and J. Hullman, "The Science of Visual Data Communication: What Works," *Psychological Science in the Public Interest*, vol. 22, no. 3, pp. 110–161, 2021.

[100] H. W. Alomari, C. Vendome, and L. Rizkallah, "A Comprehensive Evaluation Framework of Software Visualizations Effectiveness," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 9, pp. 6056–6074, 2024.

[101] Y. Alshehhi, K. Ahmad, M. Abdelrazek, and A. Bonti, "6DVF: Data Visualisation Framework for mHealth Apps," in *Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering*, ser. ENASE '24, Angers, France, Apr. 28-29 2024.

[102] S. Debortoli, O. Müller, and J. vom Brocke, "Comparing Business Intelligence and Big Data Skills," *Business & Information Systems Engineering*, vol. 6, p. 289–300, 2014.

[103] S. Gottipati, K. J. Shim, and S. Sahoo, "Glassdoor Job Description Analytics – Analyzing Data Science Professional Roles and Skills," in *Proceedings of the IEEE Global Engineering Education Conference*, ser. EDUCON '21, Vienna, Austria, Apr. 21-23 2021, pp. 1329–1336.

[104] J. Gunklach, M. Nadj, S. Michalczyk, K. Jacob, C. Gröger, and A. Mädche, "Beyond the Unicorn? Job Roles in Data Science," *Business & Information Systems Engineering*, vol. 1, p. 15, 2025.

[105] A. Abdulla, A. Naim, A. Muniasamy, A. B. Mohammed, S. M. Bilfaqih, and A. Sabahath, "Optimizing Business Insights Data Visualization Applications in Sales Forecasting, Marketing Analytics, and Financial Reporting," *Data Visualization Tools for Business Applications*, p. 22, 2024.

[106] J. Sun, "Business Intelligence Visualization Technology and Its Application in Enterprise Management," in *Proceedings of the International Conference on Big Data Engineering and Technology*, ser. BDET '20, Singapore China, Jan. 3-5 2020, pp. 45–48.

[107] R. J. Hinson, A. J. Kinsella, and R. E. Propper, "So Much Information, So Little Screen Space: Assessing the Usability of Hierarchical Data Visualizations in Tableau," *Usability and User Experience*, vol. 39, pp. 310–318, 2022.

[108] L. R. Nair, S. D. Shetty, and S. D. Shetty, "Interactive Visual Analytics on Big Data: Tableau vs D3.js," *Journal of E-Learning and Knowledge Society*, vol. 12, no. 4, 2016.

[109] Microsoft Corporation, "Power BI Overview," https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview, 2024, [Accessed 2025-05-16].

[110] V. Deshmukh, J. Prithviraj, A. Rautkar, R. Agrawal, C. Dhule, and N. Chavhan, “Interactive Data Visualization Platform to Present Effective Teacher Performance with Power BI,” in *Proceedings of the International Conference on Artificial Intelligence, Communication, and Computational Technologies*, ser. ICAICCIT ’23, Faridabad, India, Nov. 23-24 2023, pp. 1335–1339.

[111] M. Dyon, K. Suryani, R. Widyastuti, and A. F. Rahmadani, “Designing Academic Data Visualization Dashboard Using Google Data Studio at SMPN 8 Pariaman,” *Journal of Applied Business and Technology*, vol. 5, no. 1, p. 154, 2024.

[112] M. M. Mahmud, S. F. Wong, A. Qazi, N. F. M. Ramli, S. F. Zakaria, and R. Rusli, “Excel-Ling in Data Visualization: Evaluating Microsoft Excel’s User-Friendliness, Visual Appeal, and Reputation Impact,” in *Proceedings of the International Conference on Innovative Engineering and Technology*, ser. ICIET ’24, Yamaguchi, Japan, Mar. 18-20 2024, pp. 507–513.

[113] F. Nunes, C. Correa, A. Jandrey, A. Barcelos, D. Reyes, M. Bernardes, A. Sales, and M. S. Silveira, “Data Visualization on Focus: Exploring Communicability of Dashboards Generated from BI Tools,” in *Proceedings of the XIX Brazilian Symposium on Human Factors in Computing Systems*, ser. IHC ’20, Diamantina, Brazil, Oct. 26-30 2020.

[114] A. Lavanya, S. Sindhuja, L. Gaurav, and W. Ali, “A Comprehensive Review of Data Visualization Tools: Features, Strengths, and Weaknesses,” *International Journal of Computer Engineering in Research Trends*, vol. 10, no. 1, p. 825, 2023.

[115] N. Kruchten, A. M. McNutt, and M. J. McGuffin, “Metrics-Based Evaluation and Comparison of Visualization Notations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, pp. 425–435, 2024.

[116] J. D. Hunter, “Matplotlib: Visualization with Python,” https://matplotlib.org/, 2024, [Accessed 2025-05-18].

[117] S. Cao, Y. Zeng, S. Yang, and S. Cao, “Research on Python Data Visualization Technology,” *Journal of Physics: Conference Series*, vol. 1757, p. 012122, 2021.

[118] R. Kaestria, E. F. Himmah, and R. Irawan, “Application of Matplotlib in Data Visualization for Analysis of the Relationship between Gadget Use and Learning Outcomes,” *Journal of Digital Business and Information Technology*, vol. 1, no. 1, pp. 29–39, 2024.

[119] G. Sunitha, A. V. Sriharsha, O. Yalgashev, and I. Mamatov, “Interactive Visualization With Plotly Express,” in *Advanced Applications of Python Data Structures and Algorithms*, 2023, pp. 182–206.

[120] A. Tritsarolis, C. Doulkeridis, N. Pelekis, and Y. Theodoridis, “ST_VISIONS: A Python Library for Interactive Visualization of Spatio-Temporal Data,” in *Proceedings of the 22nd IEEE International Conference on Mobile Data Management*, ser. MDM ’21, Toronto, ON, Canada, June 15-18 2021, pp. 244–247.

[121] HoloViews Developers, “HoloViews: Building Complex Visualizations Easily,” https://holoviews.org, 2024, [Accessed 2025-05-14].

[122] J.-L. R. Stevens, P. Rudiger, and J. A. Bednar, “HoloViews: Building Complex Visualizations Easily for Reproducible Science,” in *Proceedings of the 14th Python in Science Conference*, ser. SciPy ’15, Tacoma, WA, USA, July 7-13 2015, pp. 59–66.

[123] S. J. J. Gould, “Complex Data Visualisation Made Easy with R and ggplot2,” in *Proceedings of CHI Conference on Human Factors in Computing Systems Extended Abstracts*, ser. CHI EA ’22, New Orleans, LA, USA, Apr. 29 - May 5 2022, p. 127.

[124] Vega-Lite Developers, “Vega-Lite: A Grammar of Interactive Graphics,” https://vega.github.io/vega-lite/, 2024, [Accessed 2025-05-18].

[125] M. Bostock, “What is D3.js?” https://d3js.org/what-is-d3, 2024, [Accessed 2025-05-14].

[126] L. Chen and H. Zhou, “Research and Application of Dynamic and Interactive Data Visualization Based on D3,” in *Proceedings of the International Conference on Audio, Language and Image Processing*, ser. ICALIP ’16, Shanghai, China, July 11-12 2016, pp. 150–155.

[127] Salesforce, Inc, “AI in Tableau and Trust,” https://help.tableau.com/current/tableau/en-us/tableau_gai_einstein_trust.htm, 2025, [Accessed 2025-08-10].

[128] Zoho Corporation Pvt. Ltd., “Zia - AI-powered data analytics assistant,” https://www.zoho.com/analytics/zia/, 2025, [Accessed 2025-08-10].

[129] Sisense Ltd., “Generative AI Analytics - Smarter, Faster Insights,” https://www.sisense.com/ai-analytics-platform/generative-ai-analytics/, 2025, [Accessed 2025-08-10].