Title
Designing Automated Assistants for Visual Data Exploration

Author
Lee, Doris Jung-Lin

Publication Date
2021-07-01

Designing Automated Assistants for Visual Data Exploration

by

Jung-Lin Lee

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Information Management and Systems

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Aditya G. Parameswaran, Chair
Professor Marti A. Hearst
Professor Joseph M. Hellerstein

Summer 2021

Designing Automated Assistants for Visual Data Exploration

Abstract

Designing Automated Assistants for Visual Data Exploration

by

Jung-Lin Lee

Doctor of Philosophy in Information Management and Systems

University of California, Berkeley

Professor Aditya G. Parameswaran, Chair

Visual data exploration enables analysts to identify trends and patterns, generate and verify hypotheses, and detect outliers and anomalies. However, the overwhelming number of decisions required in visual data exploration presents a barrier to discovering useful, actionable insights from data. To address this challenge, in this dissertation, we investigate how automated assistance via tooling aids visual data exploration.

We introduce four systems to survey the design space of visual exploration assistants across different analytical tasks and interface modalities. We first describe VisPilot and Zenvisage++, two novel visual exploration assistants that accelerate the data exploration process for individual visual analysis tasks: drill-down analysis and pattern search. Next, we examine visual exploration assistants aimed at supporting multiple types of visual analysis tasks. We introduce Frontier, a general-purpose visual exploration assistant within a GUI-based charting tool that recommends potential next steps in a mixed-initiative visual analysis workflow. We further develop Lux, a general-purpose visual exploration assistant situated within a computational notebook that provides proactive, always-on recommendations within an exploratory programming workflow. Findings from this dissertation contribute towards designing an intelligent visual exploration assistant that suggests helpful tailored feedback based on user's analytical needs and seamlessly guides users towards data-driven insights.

To Mom and Dad

Contents

List of Figures

List of Tables

Acknowledgments

This dissertation is the culmination of the support and guidance from many mentors, colleagues, friends, and family.

First, I am deeply thankful for the mentorship and support that my advisor Aditya Parameswaran provided over the past five years. Our collaboration began when he took a leap of faith in me when I approached him with a far-fetched research idea as a clueless first-year grad student. Through his continued mentorship, he encouraged my pursuit of a human-centered, interdisciplinary research agenda. Across all our projects, he challenged me to think about the practical implications and impact of my work. Beyond his technical tutelage, Aditya leads by example and is deeply invested in supporting the well-being of his students. I aspired to learn from him as I began to mentor students on my own. I could not have asked for a better role model during this period of my professional development.

I wanted to thank my committee members: Joe Hellerstein and Marti Hearst. Your feedback has been incredibly inspiring and valuable. Joe, thank you for sharing your wealth of knowledge and lived experience in building data management systems. Each and every one of our conversations left me with key questions and design decisions that have propelled this work. Marti, I thoroughly enjoyed our experience working together. I admire how you never failed to shed light on ways that we can improve the end-user experience in Lux and deliver this feedback with such precision and clarity. I am thankful for the research and design critiques you generously provided, which have been especially helpful through times when I felt stuck on the project.

At UIUC, I was fortunate to work with Karrie Karahalios at the start of my Ph.D. Karrie, thank you for all your thoughtful comments and teachings around how to conduct HCI research, from working with end-users to evaluating visualization systems to articulating research contributions. These early lessons have not only enriched our collaboration, but were also immensely influential to the subsequent projects explored in this dissertation.

My internship at Tableau played a significant role in shaping the way I conducted visualization research. During my time at Tableau, I learned from many amazing researchers and mentors, including Vidya Setlur and Melanie Tory. Melanie's valuable feedback on study design and analysis taught me how to be more rigorous in designing and conducting user studies. Vidya's deep expertise in natural language interfaces inspired many ideas and research questions about how visualization recommendations fit within an analytical workflow. Vidya, thank you for offering your continued support and unfiltered advice on research and life. From our impromptu Slack sync ups to our late-night sessions before paper deadlines, working alongside you has been one of the highlights of my time in grad school.

I am lucky to have been part of many communities across different groups and institutions that have supported my personal and professional growth. At UIUC, I am grateful to have learned from fellow students who were older and wiser than me: Mangesh Bendre, Himel Dev, Biplab Deka, Jinda Han, Silu Huang, Stephen Macke, Sajjadur Rahman, Akash Das Sarma, Tarique Siddiqui, Kristen Vaccaro, Tana Wattanawaroon, Doris Xin, Liqi Xu.

Midway through my Ph.D., many have contributed to making my move from UIUC back to Berkeley wonderfully seamless and welcoming. During this time, the I School faculty, and staff, including Coye Cheshire and Inessa Lee, and the incredible Ph.D. student community, have been nothing short of supportive. I also appreciate reconnecting with familiar faces from the lab where I spent the bulk of my undergrad years at, Eric Paulos, Chris Myers, Christie Dierk, Sarah Sterman, Kevin Tian.

From time to time, I've had the privilege of interacting with some world-class experts in their respective fields: Azza Abouzied, Sarah Chasin, Alvin Cheung, Joey Gonzalez, Björn Hartmann, Anthony Joseph, Alark Joshi, Eser Kandogan, Dominik Moritz, Fatma Ozcan, Abdul Quamar, Niloufar Salehi, Koushik Sen, and many others, who have been incredibly kind in sharing their thoughtful perspectives that have fueled this interdisciplinary research. My days at Berkeley was filled with exciting conversations and productive collaborations with friends and colleagues across RISE Lab and BiD: Rohan Bavishi, Rolando Garcia, Andrew Head, Forrest Huang, Charles Lin, Kevin Lin, Alyssa Morrow, Ananya Nandy, Devin Petersohn, Rishabh Poddar, Eldon Schoop, Vikram Sreekanti, Dixin Tang, Jeremy Warner, Michael Whittaker, Chenggang Wu, Yifan Wu.

Throughout my Ph.D., I am thankful to have the opportunity to mentor and work with many bright young minds who have worked relentlessly in contributing to the systems presented in this dissertation: Kunal Agarwal, Thyne Boonmark, Caitlyn Chen, Huizi Hu, Jintao Jiang, Jake Kang, Jaewoo Kim, Angela Lee, Ujjaini Mukhopadhyay, Jerry Song, Renxuan Wang, Chaoran Wang, Eva Wu, Edward Xue, Xu Yang, Micah Yong, Zhiwei Zhang, Jared Zhao. Thank you all for being on this journey with me.

Outside of academia, thank you to my friends who kept me grounded and sane: Andrew, Christine, David, Debbie, Gabby, Jessica, Jessie, Joanne, John, Kelly, Leah, Lydia, Mahesh, Mike, Nancy, Nir, Patty, Tess, Tina. Special thanks to Alan for being my closest confidant and for always being there for me. I cherished every moment of growing together with you over the last five years. Thank you, Cuong and Huong, for taking me in during the pandemic and treating me like your own family. I appreciate all that you have done for me.

This journey would not have been possible without the support of my family. Thank you, Anita, Kevin, Mom, Dad, for putting up with me during the highs and the lows. Mom and Dad, I am deeply thankful for all sacrifices you made in putting our education first, even if that meant leaving behind what was familiar and comfortable to you. Day by day, I have grown to appreciate the important ethics and values you instilled in me. Thank you for being my biggest believer and for encouraging me to pursue my dreams. I am forever grateful for all your patience, love, and support throughout this journey.

Chapter 1

Introduction

In our growing informational economy, data is at the center of decision-making across organizations, from retail, construction, and entertainment, to manufacturing, science, and healthcare [166, 206, 73]. To discover valuable insights from data to inform decisions, analysts perform exploratory data analysis (EDA) [214] to browse and inspect data, visualize trends and patterns, as well as transform, manipulate, clean, and prepare datasets for analysis. EDA occupies a large fraction of the workflows of typical data scientists; a 2019 survey cites that over 75% of respondents reported understanding and analyzing data as the most common activity in their job [206]. Visual data exploration involves the use of visualizations for identifying trends and patterns, generating and verifying hypotheses, and detecting outliers and anomalies during the process of exploratory analyses [104, 105, 54]. Studies have shown that visual data exploration is not only useful in its own right, but also serves to inform other parts of typical data science workflows, including cleaning, extraction, debugging, and modeling [15, 100, 238].

Despite its promise, the current process of visual data exploration is rife with challenges. To perform visual data exploration, analysts often employ GUI-based visualization tools such as Tableau [209] or visualization libraries such as matplotlib [89] to generate individual visualizations one at a time. However, analysts are often overwhelmed by the large number of analysis decisions they have to make to find relevant insights. For instance, to generate a single visualization, analysts need to select the subset of data to operate on, as well as one or more attributes to visualize. Moreover, with the programming-based tools, there is a need to write a lot of code to even get to a single visualization. The problem is exacerbated by the fact that analysts often do not have a clear notion of what visualization could potentially lead to valuable insights, leading to substantial time and effort wasted on unfruitful analyses, or worse, a lack of motivation to experiment with new hypotheses. These challenges present a barrier to insight discovery. As a result, visual data exploration remains out of the reach of most end-users without substantial programming or data expertise.

Given the adverse impact on productivity and high barrier-of-entry for the growing data science workforce, there is a pressing need for systems that guide analysts during the process of visual data exploration. We call such systems *visual exploration assistants* (also commonly

known as *visualization recommendation systems*), providing automated guidance during visual data exploration. Automation offers one potential approach to bridge this gap between the largely manual, existing practices and effortless insights. This dissertation investigates how automation can be incorporated via intelligent, highly-usable, and accessible assistants to help analysts in exploratory analysis workflows. As we will see in this dissertation, there is a range of research challenges in designing such visual exploration assistants. For instance, how do we determine which visualizations may be interesting to recommend to analysts? How can analysts make use of their domain knowledge to steer the assistant towards what they deem relevant? How does such an assistant fit into the analyst's existing workflow and practices?

Our vision for visual exploration assistants is to accelerate exploratory analyses similar to how search engines have democratized the ability for laypeople to access the web by accelerating and simplifying information search and retrieval. Similar to how search engines bridge the gap between a given user's search intent and desired tasks or webpages, visual exploration assistants effectively guide analysts towards their high-level analysis goals by automatically surfacing useful and relevant insights.

1.1 Overview of Dissertation

This dissertation contributes to the design of visual exploration assistants for exploratory data analysis. Our thesis is:

> *Analysts can perform visual data exploration more effectively with automated assistance throughout an exploratory analysis workflow.*

The dissertation covers a set of visual exploration assistants that we have developed across different interface modalities and analysis tasks supported. Here, we provide a brief overview of each of the chapters in this dissertation.

Background (Chapter 2):

We begin by surveying the existing work and background related to visual exploration assistants. This chapter presents an overview of the existing practices and challenges around visual data exploration. Then, we survey research on emerging visual exploration assistants. Throughout the chapter, we introduce key terminology used across the dissertation. A more detailed roadmap of the dissertation can be found in Section 2.3 in Chapter 2.

In the first part of this thesis, we contribute two novel systems aimed at helping users accelerate visual data exploration for a single visual analytical task, namely drill-down exploration and visual querying, described below.

Assistance during Drill-down Exploration with VisPilot *(Chapter 3):*

The task of navigating through a large, multidimensional dataset is a common challenge in exploratory analysis. Not only is manual drill-down and roll-up on data subsets tedious and inefficient for the analyst, the massive space of data subsets, lack of interesting patterns

in most data subsets, as well as fallacies and pitfalls stemming from spurious correlations and statistical paradoxes, all call for a systematic and effective way for analysts to make sense of and navigate through the large space of possible visualizations. This chapter presents VisPilot, an interactive visualization recommendation system that automatically identifies a dashboard of connected visualizations that summarizes the interesting and informative trends in the dataset. VisPilot intelligently explores the lattice of equivalent visualizations across data subsets, and recommends visualizations based on an intuitive user-expectation model.

Assistance during Visual Querying with Zenvisage++ *(Chapter 4):*

Next, we investigate a class of systems known as visual query systems (VQSs) that empower users to interactively search for line charts with desired visual patterns. We discovered that despite decades of past work on VQSs, these research efforts have not translated to adoption in practice. To address this gap in adoption, we collaborated with experts from three diverse scientific domains via a year-long user-centered design process to develop a VQS that supports their workflow and analytical needs, and evaluate how these improved VQSs can be used in practice. This chapter summarizes our design study findings and presents design guidelines for improving the usability and adoption of next-generation VQSs.

In the second part of this thesis, we explore how visual exploration assistants provide proactive visualization guidance for general-purpose analytic tasks across different modalities, from GUI-based charting tools to exploratory programming workflows.

Assistance in GUI-based Charting Tools with Frontier *(Chapter 5):*

Existing GUI-based charting tools such as Tableau [209] often provide recommendations by suggesting potentially interesting next steps during exploratory data analysis. These recommendations are typically organized into categories based on their analytical actions, i.e., operations employed to transition from the current exploration state to a recommended visualization. Existing systems often implement a small number of bespoke categories without a deep understanding of the utility of such categories in analytical workflows. This chapter explores the efficacy of recommendation categories by formalizing a taxonomy of common categories via a system, Frontier, that implements these categories. We evaluate workflow strategies adopted by users and how categories influence those strategies.

Assistance in Computational Notebooks with Lux *(Chapter 6):*

In recent years, visual data exploration largely happens in computational notebooks using a dataframe API, such as `pandas`, which provides a flexible means to transform, clean, and analyze data. Yet, visually exploring data in dataframes remains tedious, requiring substantial programming effort for visualization and mental effort to determine what analysis to perform next. This chapter presents Lux, an *always-on* framework for accelerating visual insight discovery in dataframe workflows. When users print a dataframe in their notebooks, Lux recommends visualizations to provide a quick overview of the patterns and trends and suggests promising analysis directions. The chapter presents careful system design decisions aimed at supporting seamless exploration, including a high-level language for specifying user

intent, and series of system optimizations to ensure interactive feedback. Finally, we present a set of performance and usability experiments demonstrating how Lux encourages rapid visual experimentation with data.

Lux's viral adoption and success can be attributed to the lessons learned from the earlier chapters in building and deploying visual exploration assistants. Beyond the research contributions, Lux has demonstrated significant real-world adoption in the data science community: Lux has already been embraced by practitioners, with over 2.6k stars on Github and 25k total downloads on PyPI as of August 2021. Multiple industry users have created tweets, blog posts, or YouTube videos extolling the virtues of Lux [57, 48, 163, 232, 135, 136]. Our vibrant community of data science users has been eager to share their successful use cases of Lux and its impact, spanning various industries from pharmaceutical to education to finance.

Conclusion (Chapter 7): Finally, we conclude by outlining the key takeaways and findings from our work, and describe future research directions in this space.

1.2 Research Methodology

The dissertation draws from an interdisciplinary body of work and techniques from human-computer interaction (HCI), data management, and visualization research. In each chapter, we investigate usability or adoption challenges in visual exploration assistants and identify key questions to bridge these missing gaps in the existing research landscape.

In Chapter 3, we address the problem of understanding data subsets via drill-down exploration. We designed VisPilot as a *system prototype* [254] to demonstrate the value of a visual exploration assistants that guides analysts through drill-down exploration. A lab experiment was designed to evaluate and quantify the hypothesized effect.

In Chapter 4, we observed that visual query systems for line chart exploration lacked practical adoption among practitioners. As a result, we performed a *user-centric, longitudinal* study to engage participants in need-finding, collaborative prototyping, and grounded evaluation [24, 83, 94, 198]. This unique approach enabled us to compare and contrast the differences and similarities across real-world use cases and domains and study how such tools are used in their situated environment, revealing generalizable challenges around tool adoption.

In Chapter 5, we investigated the effects of different types of recommendation categories in a visual exploration assistant, categorized under our taxonomy of analytical actions. We develop Frontier as a *design probe* [82] to explore how analysts leveraged recommendation categories in their analysis workflows. Frontier implements ten most-common recommendation categories based on our synthesis of existing systems, enabling us to systematically explore and compare the efficacy of these categories.

In Chapter 6, we design a visual exploration assistant that supports analysts within their dataframe workflow. We applied iterative design principles to create Lux as a high-fidelity system that was released in the open-source community. We evaluated the system

usage and effectiveness through a controlled first-use study [74] and post-deployment field trials [29] with early adopters. In addition, we performed system performance experiements to quantitatively evaluate the scalability of Lux.

Our choice of research methodology for each chapter is tightly coupled with the research questions that we aimed to address. We advocate that careful selection of research methods is crucial for understanding how visual exploration assistants can be put into practice.

1.3 Prior Publication and Authorship

Chapter 3 was published in the ACM Conference on Intelligent User Interfaces (IUI) in 2019 [122]. This work was done in collaboration with Huizi Hu and Himel Dev, who helped with the development of the Frontier system. The project was advised by Hazem Elmeleegy and Aditya Parameswaran.

Chapter 4 was published in the IEEE Conference on Visual Analytics Science and Technology (VAST) in 2019 [127]. The paper builds off of the Zenvisage system developed by John Lee and Tarique Siddiqui. In our year-long design study, many students from the University of Illinois contributed to the development of Zenvisage++, including Jintao Jiang, Jaewoo Kim, Chaoran Wang, Renxuan Wang, Edward Xue, Xu Yang, and Zhiwei Zhang. Our scientific collaborators have generously provided their valuable time and feedback that guided this work. The project was advised by Karrie Karahalios and Aditya Parameswaran.

Chapter 5 was published in the IEEE Transactions on Visualization and Computer Graphics (TVCG) in 2021 [124]. This work started as an internship project at Tableau Research, but has evolved substantially through the guidance and feedback from Vidya Setlur, Melanie Tory, Karrie Karahalios and Aditya Parameswaran.

Chapter 6 represents a paper currently under submission. The project was advised by Marti Hearst and Aditya Parameswaran. Dixin Tang helped brainstormed the design of the performance experiments and made substantial contributions to the paper writing. Lux would not be possible without the dedicated team of students who contributed to the open-source development of the system, including Kunal Agarwal, Thyne Boonmark, Caitlyn Chen, Jake Kang, Jaewoo Kim, Ujjaini Mukhopadhyay, Jerry Song, Micah Yong, and Jared Zhao. A full list of contributors can be found here: `https://github.com/lux-org/lux/graphs/contributors`.

All co-authors of these articles have consented for the aforementioned work to be reproduced in this dissertation.

Chapter 2

Background

This chapter begins with an overview of the present-day practices around visual data exploration. Then, we outline the existing landscape of visual exploration assistants. Finally, we present a vision for next-generation visual exploration assistants and describe the design space explored in this thesis. Throughout the chapter, we define key terms and their specific definitions (📖) as used in this dissertation.

2.1 Visual Data Exploration: People and Practice

What is visual data exploration?

Exploratory data analysis (EDA) fulfills an essential part of the information foraging (searching, filtering, and extracting) and sensemaking process (building a mental model) [113, 171] — ultimately leading analysts to formulate a course of action to further their analyses. While the definition of exploration is multifaceted, several established qualities and characteristics of data exploration have emerged from studies of exploration practices [5, 17].

First, the process of EDA is highly variable, spanning an unguided "*futzing and moseying*" [5] around the data to more directed querying guided by a high-level analysis goal [159, 17]. Statistician John Tukey describes exploratory data analysis as "*an attitude, a state of flexibility, a willingness to look for*" patterns and trends in the data [214]. A curiosity-driven style of exploration means that analysts do not always have well-defined hypotheses or questions in mind. Instead, Tukey describes EDA as similar to doing detective work, wherein analysts elaborate and incrementally build on their previous analysis by manually browsing and inspecting the data, computing statistics, querying, and visualizing data. Often, this unstructured exploration of the data is guided by high-level, open-ended inquiries and goals, such as determining if the data is clean enough for downstream analyses (profiling) or discovering new insights or hypotheses (discovery) [238, 5].

Second, exploratory analysis is often a result of iterative deductive inquiries resulting from incremental refinements of hypotheses [117, 185, 171, 113]. During EDA, data scientists

traverse through a space of analyses to determine potentially relevant insights and hypotheses worthy of investigating further. In practice, EDA involves browsing, querying, visualizing, and inspecting the data. More recently, exploratory programming has also emerged as a paradigm for rapid iteration, prototyping, and debugging of data through code, largely through an interactive environment, such as computational notebooks [75, 21, 106].

Why visualizations? Visualization is one of the most effective and widely-used techniques for understanding data. Visualizations help analysts discover trends and patterns, identify outliers and anomalies, characterize distributions, and inspect relationships and comparisons [27, 59]. Moreover, visualizations tell compelling and intuitive stories about our data, often summarizing complex, underlying processes that are otherwise hidden behind individual numbers.

The use of visualizations has largely been categorized into two goals [153, 26]: presenting information that the user already understands (i.e., storytelling [116, 190, 88]) or discovering information that the user is unaware of or has not fully understood (i.e., exploration [235, 121, 39, 145]). This latter process of performing exploration and analysis through visualizations is often known as *visual data exploration*. Preliminary insights derived from visualizations are especially important in the exploratory stages of the data science workflow to help catalyze new questions, hypotheses, and actions on what to do next. These insights also inform decisions in other parts of the data science workflow, including cleaning, debugging, and model-building.

Visual data exploration (or visual analysis) is the process of using visualizations for discovering insights during exploratory data analysis.

Who performs visual data exploration?

Data analysts often perform visual data exploration as part of their day-to-day work. These individuals encompass an interdisciplinary, diverse set with varying levels of data-centric and programming expertise, job roles and functions, and preferred toolstack. In particular, here, we describe two characteristics of these individuals that are relevant to this dissertation.

On Job Function: Due to the nascency and rapidly-evolving nature of the data analysis field, data work across different organizations can entail very different day-to-day tasks and professional functions. Existing surveys of industry practitioners have investigated data science practices in enterprise organizations and identified key user archetypes [100, 110, 73]. These personae include business analysts who compile reports using SQL and Excel, as well as data scientists who develop machine learning models to predict business metrics. Specifically, studies on data exploration practices report that EDA is performed by individuals across a variety of domains and roles [5, 238]. In fact, many domain experts may not even have the words "data scientist" or "analyst" in their job titles. To capture this nuanced meaning, we use the more general term *analyst* throughout this dissertation to refer to anyone who engages with data analysis and exploration as part of their work.

An **analyst** performs data analysis and exploration, which includes, but is not limited to, data scientists, business analysts, or other domain/subject-matter experts.

On Domain and Programming Expertise: According to a 2019 survey done by Kaggle [206], while data scientists often have advanced degrees, only about a third have had more than five years of programming experience. Data scientists often do not have a traditional degree in computer science (CS). They are often trained in a non-CS field where they developed a strong analytical approach to problem solving [110, 73]. In fact, their wealth of domain knowledge and quantitative skills [166] are key drivers for the more opportunistic open-ended style of exploration [5, 17]. The non-CS background for these analysts has two implications for tool design.

First, analysts are often described as "hackers" [110, 100], who stitch together variety of tools and packages well-suited for their particular task, rather than relying on a single end-to-end tool to encompass their entire workflow. In addition, Harris et al. (2013) observed a common "T-shaped" archetype for data scientists, describing someone with general skills across the five areas (programming, statistics, math, business, ML/big data) and a specific strength in one [73]. Thus, while some analysts are code-proficient, they are often less acquainted with standard software engineering practices, such as testing or debugging [7, 110]. As a result, analysts often opt for makeshift solutions that "get the job done", but these approaches become less sustainable for analyzing larger or more complex data.

Second, as data becomes more prevalent in a variety of domains, there is a growing number of analysts from non-CS fields that bring a wealth of domain expertise, knowledge, and perspectives to data-driven problems. Given the T-shaped characteristics described earlier, many domain-expert analysts may not have the programming or data expertise required to interact with data effectively. While the visualization evaluation literature has long proposed the need for a human liaison [203] or translator [189] that bridges the knowledge gap between domain scientists and visualization experts, we take an alternative approach to understand how improved tooling with automated assistance can lower the present barriers and invite domain-expert analysts to explore data on their own. In this dissertation, we describe how automated assistants democratize the visual data exploration process by empowering analysts to leverage domain expertise to advance their data-driven questions.

How do analysts currently perform visual data exploration?

Analysts employ various types of analysis tools to perform visual data exploration on their data. In particular, the different levels of programming expertise of analysts have led to two distinct classes of analytics tools: GUI-based and programming tools.

Interaction: GUI-based interfaces are highly accessible and easy-to-use interfaces that allow users to interact with their data through a series of point-and-click interactions. These interactive interfaces are built upon the principles of direct manipulation [195, 90], where end-users interact with an object representation of the data, such as cells in a spreadsheet [148,

20, 14] or attribute shelves in a chart construction interface [207, 176, 134]. These systems support users in the creation of visualizations by interacting with interface elements, such as buttons, dropdown, and option panes. One attractive aspect of such interfaces is the abilty to rapidly manipulate the objects in an incremental, reversible manner. For example, the ability for users to dynamically query data on-the-fly through these direct manipulation interfaces is conducive to visual data exploration and facilitating rapid hypothesis generation based on interactive visual feedback [12, 196, 79]. As evidence to the success of this approach, commercial visualization tools offering rich interactive charting experiences, e.g., Tableau [209], Microsoft Excel [148], or PowerBI [172], have been overwhelmingly popular among business analysts without substantial coding expertise.

Programming: Programming languages, such as Python and R, have been a popular choice among many analysts for data analysis and exploration. The community adoption around these languages has led to an associated ecosystem of libraries and APIs. Analysts are often attracted to the plethora of convenient capabilities that abstract common subroutines in specific parts of the data science workflow, e.g., pandas [210]/dplyr [230] for data manipulation, or matplotlib [89]/ggplot [229] for visualization. In addition, statistical analysis tools, such as SPSS [91], SAS [93], and Matlab [143], provide integrated visual environments and domain-specific languages (DSLs) for data preparation, statistical analysis, visualization, and modeling. To perform visual data exploration with code, analysts typically develop an analysis script that composes these data processing and visualization subroutines into a pipeline. In recent years, analysts often make use of IDEs, such as computational notebooks, to more easily iterate on and experiment with their analysis.

Analysts with coding expertise often favor the level of transparency and control that these largely open-source programming tools afford. However, the extreme flexibility of these languages also means that analysts need to expend substantial effort to make decisions around what analysis steps to take. For example, programmatic visualization toolkits like ggplot, D3 [25], and matplotlib, require analysts to explictly specify the encoding details of each individual visualization the user is interested in plotting. As we will see in Chapter 6, the high programming cost associated with writing such visualization code presents a barrier to visual data exploration.

2.2 Landscape of Visual Analytics Systems

To transform data into insight, the analyst needs to do more than simply create a single visualization to explore their data. With the ever-increasing complexity and size of datasets, analysts often suffer from the problem of information overload. As described by Keim et al. (2008), visual analytics systems aim to "*identify methods and models, which can turn the data into reliable and provable knowledge*" [105]. Endert et al. (2014) portray visual analytics as "*marry[ing] the big data processing capabilities of analytics with the human intuitive capabilities of interactive visualization*" [54]. The goal of visual analytics is to

facilitate an interactive and continuous dialogue between the human analyst and the system to advance knowledge discovery [33, 53, 102, 103].

Visual analytics systems employ a mixed-initiative approach where human analysts and systems work collaboratively to forage, synthesize, and make sense of the complexity and scale present in visual data exploration workloads.

In this section, we survey the landscape of existing research literature on visual analytics systems, with a particular focus on the *levels of assistance* that these tools provide. We organize the sections by introducing three classes of related techniques and systems offering increasing levels of automated assistance.

In the *specify* setting, the onus is placed on the analyst to manually provide an exact and complete specification of each decision in their analysis, such as whether a visualization should be a bar chart or a scatterplot. In the *search* setting, the analyst composes a set of high-level instructions or a *query* regarding what their desired analysis goals are, but not necessary *how* to perform them. Based on these queries, the system assists users by operationalizing their request. Finally, in the *suggest* setting, the analyst provides little to no specification of their analysis goals. The analyst leaves it up to the system to proactively recommend what may potentially be useful for their analysis.

Specify*, *search*, and *suggest are three interaction settings that characterize the level of automated assistance a visual analytics system provides. The different settings correspond to the division of work between the analyst and the system in accomplishing the analysis goals.

While we introduce each level in isolation, in practice, these techniques fall on a spectrum: it is not uncommon for visual analytics systems to offer interaction modalities that span multiple levels of assistance. What we will see is that as we increase the level of assistance from *specify* to *suggest*, it becomes more challenging for the system to automatically infer and interpret what the analyst might have in mind.

Specify: Languages and Frameworks for Visual Data Exploration

To visually explore data, analysts often need to operate on the data and create (i.e., design) a visualization. These tasks are often performed via different classes of tools, including data analysis frameworks and visualization design languages.

Frameworks for Data Analysis

Analysts often leverage data analysis frameworks and libraries for a diverse range of analytical needs, from data manipulation to statistical modelling. In exploratory analysis, data transformation choices (e.g., how do I reshape the data? what aggregation or filter should I apply?) are often intricately tied to statistical analysis and visualization decisions [5, 224].

Traditionally, data transformation can be performed in relational databases via SQL, including grouping and aggregation. Analysts commonly employ programming languages such as Python [218], R [174], SAS [93], or MATLAB [143] to more flexibly operate on their data. These programmming languages often come with an associated ecosystem of APIs and frameworks (e.g., `pandas` [210], or tidyverse [231]) for data manipulation, transformation, and summarization. Furthermore, analysts can programmatically explore and analyze their data through scripting or visual programming environements, such as R Studio [177] or Jupyter Notebooks [114].

Of late, dataframes have become a popular abstraction for data processing among data scientists [210, 168, 244]. Dataframes support a comprehensive set of operators that make it easy to do sophisticated data transformations, while also allowing rapid validation after each incremental step. However, programmatically working with dataframes for visual data exploration is challenging, requiring substantial programming and analytical know-how. Analysts often have to craft custom data processing code to experiment with different analyses — a tedious task that not only requires substantial analytical skills, but also intricate familiarity with disparate APIs and libraries. In Chapter 6, we present a seamless visual data exploration solution for working with dataframes.

Declarative Languages for Visualization Design

Information visualization makes use of graphical representations (i.e., marks) and visual properties (i.e., axes and channels) to encode selected information from the data [213, 59]. To design a visualization, the analyst defines a mapping from data to these graphical elements, such as the position and color of the marks, which is then rendered onto the digital screen. Numerous perceptual studies and principles have formalized guidelines and best practices for determining the most effective encoding for information visualization design [32, 213].

These visualization design principles have been codified into grammars, which defines the different components of specifying a visualization [234]. Visualization languages and libraries implement this grammar so that users without visualization design expertise can easily generate visualizations [187, 188, 229, 25, 207]. In recent years, there has been a trend towards visualization libraries with increasing levels of "declarativeness" to abstract away the low-level details of visualization design and programmatically synthesize effective graphical encodings [139, 138, 219, 188]. Modern interactive visual analytics systems, including the ones presented in this dissertation, build on top of these declarative visualization libraries [207, 186].

While the adoption of declarative visualization languages has been broad, specifying a visualization through these declarative languages suffers from the aforementioned complexity challenges — the need for painstaking manual specification of each possible visualization is time-consuming. Moreover, developments towards an extensible, general-purposed language for visual data exploration beyond visualization design is still elusive. A challenging aspect for language design is the need to support a range of analytical tasks that maybe useful for analysts, such as searching for outliers, explaining anomalies, comparing across visualiza-

tions, or getting an overview summary of the dataset. There has also been research into characterizing these visualization tasks into a framework or taxonomy [6, 79, 153, 27, 111], but future work remains to be done to operationalize these frameworks in the context of designing visual exploration assistants. In Chapter 5 of this dissertation, we present a taxonomy of common analytical actions corresponding to potentially interesting "next-steps" in an analyst's exploration workflow.

More generally, a declarative language for visual exploration would enable users to ask questions about their data at a high-level, with the system inferring the necessary steps to provide relevant feedback — effectively bridging the gulf of execution [158] between *how users think about data* and *how data exploration is actually performed*. Similarly, Heer (2019) [77] describes how domain-specific languages provide "*a shared medium in which both people and machines reason about and formulate actions*". Such a high-level language can serve as an intermediate layer between end-users and intelligent applications for *search* and *suggest*, described next.

Search: Accessible Query Interfaces

While the languages and frameworks in the *specify* setting are highly expressive, these tools require the analyst to provide a complete and exact description of the desired procedures for visual data exploration. However, composing programs based on specific query statements is often a challenging task for novice analysts and non-programmers. In practice, analysts often leverage their domain knowledge to guide high-level analysis decisions (e.g., explain the peak in sales in August), which can often be fuzzy and imprecise. Search interfaces assist users in the process of formulating a query through alternative modalities, including natural language, demonstration, and visual examples.

Natural Language Interfaces

Natural language serves as an intuitive and accessible modality for analysts to perform data exploration [96, 9]. Natural language interfaces for visual data exploration are largely categorized into systems for database querying [129, 45] and for data analytics [60, 193, 43, 56, 97].

Database querying is well-known to be a challenging task for non-technical or novice users as it requires an intimate understanding of the database schema and a level of mastery of the querying language, such as SQL [96, 92]. Natural language interfaces for databases build on prior work on keyword search to enable users to compose complex queries on relational or semistructured databases [128, 248]. These natural language querying systems often consist of a parser that constructs a query tree or graph, which is then mapped to portions of the query in the target DSL. Given that natural language queries can often be ambiguous, these systems often determine multiple between possible valid interpretations of the queries and return a ranked list of interpretations with query results to the user [131, 129, 130].

Similar to natural language interfaces to databases, natural language interfaces for visualization [60, 193], analysis [43], and machine learning [56, 97] also aim to democratize data access and capabilities to non-programmers. For visualizations, natural language is used to support dashboard construction by creating and modifying visualizations [11]. These systems leverage a custom grammar to extract relevant intents and entities and subsequently map them to a program or template written in terms of the underlying programming language [155]. Users can clarify ambiguous intents through the use of interactive widgets [60, 11] to refine and elaborate on their input utterances. Early prototypes for natural language visualization construction often lacked support for historical context, which is essential for composing complex queries. Subsequent systems such as Evizeon [85], Iris [56], and Ava [97], have moved towards a more *conversational* approach that leverages conversational principles to maintain and modify context across utterances [192]. While such systems have made it easier for novice data analysts to construct visualizations, users of natural language systems have expressed that they can often feel lost in the conversation due to the lack of feedback regarding how the system is interpreting the user query and what questions can be asked [193, 56], indicating the need for richer feedback mechanisms that can *suggest* follow-up questions as next-steps to guide the analysis [43, 204, 205].

Programming-by-demonstration

Programming-by-demonstration is a class of techniques in program synthesis that generates a valid program based on a task demonstrated by the user [68]. Programming-by-demonstration automates repetitive tasks by synthesizing programs for querying [255], data transformation [47, 18], and visualization [224, 181, 256]. Programming-by-example (PBE) refers to when the task is demonstrated through the a specification of the input and output examples. One of the most successful application of PBE is the Flash Fill feature in Excel spreadsheets, which takes in one or more cells containing example strings and synthesizes a program that performs the desired string transformation on additional cells [67].

While PBE alleviates the need to manually *specify* operations in the program, the specification of examples requires users to already know what the output should look like. Several programming-by-demonstration systems have employed other techniques that elicit additional sources of user input to meet users in the middle [251, 256, 47, 13]. For example, when a user demonstrates an interaction (e.g., brushing on a set of points) in a PBE-based visualization editor, the system offers a set of recommended previews of related interactions (e.g., change color or modify opacity) to help users refine their desired program [256]. Likewise, to alleviate the tedium of specifying tabular examples cell by cell, Wrex features an autocomplete feature that fills in the remaining unspecified row [47]. Programming-by-demonstration can also be combined with natural language to leverage the unique advantages of different interaction modalities [13].

Visual Query Systems

Visual query systems (VQSs) allow users to specify the desired pattern via some high-level specification language or interface, with the system returning recommendations of visualizations that match the specified pattern. VQSs can be considered a special class of query-by-example interfaces where users can directly search for visualizations by providing visual examples, such as a sketch of a pattern. While VQSs largely focus on line chart pattern search, in theory, VQSs could also be applied to other chart types, such as heatmaps, scatterplots [125], or images [30], based on different similarity metrics. For example, Earth mover's distance (EMD) is a common metric used for image search and retrieval [178].

Given that the shape of a line chart characterizes an intuitive relationship between the dependent variables, early work in this space focused on interfaces to search for time series with specific patterns. For example, TimeSearcher [81, 80] requires users to specify the query in the form of a rectangular selection box, with the system filtering out all of the time series that does not pass through the box. Other sketch-to-query interfaces allow users to sketch the desired shape of a visualization with the system returning visualizations that look similar [149, 226, 180]. Subsequent work recognized the ambiguity of sketching by studying how humans rank the similarity in patterns [49, 36, 142] and improved the expressiveness of sketched queries through finer-grained specification interfaces and pattern-matching algorithms [180, 84]. Some VQSs also support specifying patterns through boxed constraints for range-queries [80], regular expressions [250], and natural language [202]. While these systems have been extensively studied in the research literature, they have not been adopted in real-world use cases. In Chapter 4, we explore the challenges that we discovered around VQSs that hinder their adoption.

Suggest: Intelligent Recommendation Systems

Predictive, Mixed-initiative Interfaces

Across the design space of data tools, systems often differ in the level of query expressiveness and interface usability they enable. For instance, while query languages such as SQL are highly expressive, formulating SQL queries is challenging for non-programmers [96, 109]. As a result, visual query construction interfaces have been developed to address this issue by enabling direct manipulation of data through visual and tabular representations [148, 52, 255]. Some of these systems leverage *search* techniques discussed in the previous section to provide an intuitive interface for eliciting specific information that can be mapped onto a pre-defined query. However, many of these visually-lifted tools still require users to specify their desired result or outcome.

Predictive interaction is a mixed-initiative framework that builds on visual lifting, but further places the onus on the system to suggest a set of possible options based on the user's high-level, ambiguous input interaction [78]. Predictive recommendations close the gap between information foraging (i.e., *search* or *specify*) and exploration, reminiscent of browsing and searching behaviors on the Web [160]. Furthermore, predictive interaction

techniques have been successfully applied to enhance visual, direct manipulation interfaces for visualization construction [209] and data transformation [212]. Next, we describe how the principle of predictive interaction has been central to the design of mixed-initiative assistants for visual data exploration.

Visualization Recommendation System

Direct manipulation tools for visualization construction require users to manually specify the exact data aspects of interest and the visual encoding of a visualization [207, 188]—a tedious and overwhelming task during the early, exploratory stages of analysis, especially for analysts without substantial visualization design expertise [39, 241, 87]. In many cases, analysts may not have enough information to *specify* exactly what they are looking for, or even *search* based on a specific question in mind. As a result, there is a need for intelligent, predictive systems offers guidance at every step to ensure that user is never stuck or out of ideas at any point during analysis by helping users jump-start their exploration. Such recommendations help facilitate a smoother flow of analysis by suggesting potential alternate paths.

To address these challenges, visualization recommendation (VisRec) systems have been developed to suggest potentially interesting insights in the form of visualizations to help guide analysts in the visual data exploration process [237, 220, 191, 236, 41, 200, 122, 87, 107, 245, 51, 118, 62, 223, 132, 183]. Typically, visualization recommendations help accelerate the process of discovering interesting aspects of the data by encouraging breadth-oriented exploration [16, 185, 241]. Beyond exploration, visualization recommenders can also assist with data cleaning [101] and model development [227, 4]. Given that VisRec systems can be regarded as a special type of recommender system, existing systems have drawn inspiration from information retrieval to address the problem of information organization for search and querying [242, 39, 247, 55]. VisRec systems can be classified based on whether the data aspects of interest and the visual encoding are manually specified or suggested by the system [239]. The earliest VisRec systems assumed that the data attributes were already known by the user and focused on recommending visual encodings [138, 139].

The focus of this dissertation is on data-based recommenders that surface interesting portions of the data to visualize based on data or statistical properties [237, 220, 191, 236, 41, 200, 122]. Some of these systems are entirely automatic [237], whereas others leverage user interaction to guide the recommendations [41, 200]. Mixed-initiative recommendation systems combine manual specification with recommendations [87, 107, 245, 51, 118, 62, 223, 132]. For instance, both Voyager [241] and DIVE [87] allow users to select data attributes of interest. Voyager suggests visualizations based on iterating through possible attributes or encodings via the notion of wildcards, while DIVE creates groups of visualizations that cover subsets of the user-specified fields. In Chapter 5, we synthesized the existing literature on data-based VisRec systems into a taxonomy of common recommendation categories. A more comprehensive survey of the history of VisRec systems in general can be found in Lee (2020) [120].

While VisRec systems date back as early as Mackinlay's 1986 paper on APT [138], the adoption of VisRec systems have been limited, with encoding-based recommenders (e.g., ShowMe [139] in Tableau [209]) receiving more widespread adoption compared to data-based recommenders. This dissertation builds on top of this body of work and investigates adoption challenges of these tools as it pertains to the demands of modern data analysis practices and workflows.

Visualization recommendation (VisRec) systems proactively *suggest* potentially useful visualizations to analysts to guide them in the process of visual data exploration.

2.3 Organization and Roadmap

The work presented in this dissertation builds on research literature in visual analytics to further understand how automated assistance aids visual data exploration. In particular, we look at how to better design these systems to provide computational assistance to users to facilitate effective analysis.

	Analytical Task Supported	**Interface Modality**
Vispilot (Chapter 3)	Bar Chart Comparison	GUI
Zenvisage (Chapter 4)	Line Chart Search	GUI
Frontier (Chapter 5)	General-Purpose	GUI
Lux (Chapter 6)	General-Purpose	Mixed GUI/code

Table 2.1: Organization of the design space of visual exploration assistants explored in this dissertation, based on the type of analytical task supported and interface modality.

As outlined in Table 2.1, we organize the design space of visual exploration assistants along two axes. First, we distinguish between systems that support a single *type of analytical task* versus general-purpose systems that support multiple types of analytical tasks. Second, the *interface modality* dimension divides systems based on whether the input and output to the system involve programming or GUI-based interfaces.

The dissertation begins by first contributing to novel visual querying and recommendation systems that accelerate the data exploration process for a single visual analysis task: Chapter 3 introduces VisPilot, a system aimed at helping analysts perform bar chart comparisons during drill-down analysis; Chapter 4 introduces Zenvisage++, a visual query system focused on assisting analysts with line chart pattern search. These early projects revealed that analysts often have needs that go beyond a single analytical task and that context-switching to another tool often disrupts the flow of experimentation with data [126]. As a result, in subsequent chapters, we investigate how to better design general-purpose visual exploration assistants that guide analysts across a diverse set of tasks in a visual analysis workflow. Chapter 5 formalizes a taxonomy of analytical actions as potential next steps in

a GUI-based visual analysis workflow. Chapter 6 builds on this taxonomy and explores how recommendations seamlessly assist users in visualizing their dataframes during exploratory programming in a computational notebook.

The chapters of this dissertation sample the different areas of the design space to contribute to a deeper understanding of the benefits and pitfalls of visual exploration assistants. The aim of this dissertation is not to develop a one-size-fit-all visual exploration assistant that addresses the needs of all analysts. Instead, the dissertation should be seen as a series of explorations of designs that together contribute to understanding the role of automated assistance across analysts with different levels of expertise, tasks, and analytical needs. While more work certainly remains to be done, by contributing to a better understanding of how visual exploration assistants can be used across these different settings, this dissertation serves as a roadmap towards the broader adoption of visual exploration assistants for novel future use cases.

Chapter 3

Assistance during Drill-down Exploration with VisPilot

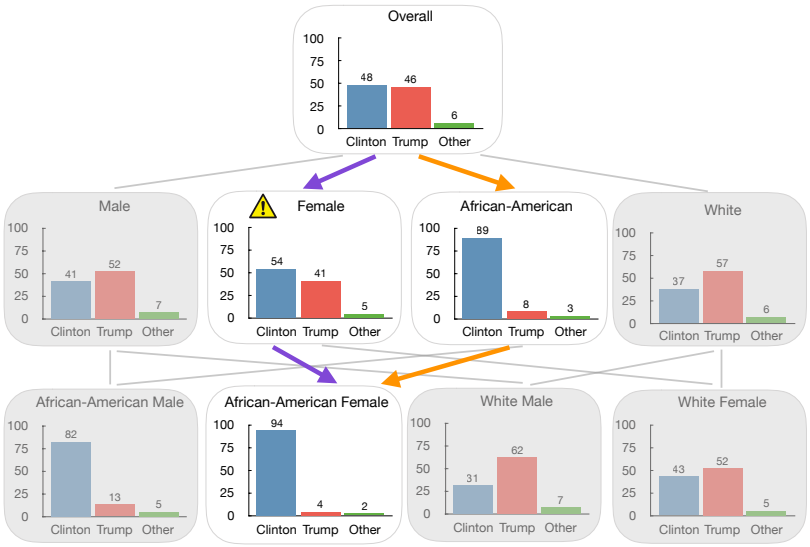


Figure 3.1: Example data subset lattice from the 2016 US election dataset illustrating the drill-down fallacy along the purple path as opposed to the informative orange path.

During exploratory data analysis, an analyst may need to compare visualizations across different subsets of the data in order to understand multi-dimensional datasets. In particular, analysts often "drill-down" to different subsets of the data to examine the patterns that emerge from different subsets and compare across them. However, even for a moderately complex dataset, exploration of multi-dimensional datasets becomes challenging. In this chapter, we present the challenges and fallacies that come with drill-down analysis for multi-dimensional data, and design a visual exploration assistant called VisPilot, which guides analysts towards informative and interesting insights in their data during drill-down analysis.

3.1 Introduction

As we have described in Chapter 2, visual data exploration is the *de facto* first step in understanding multi-dimensional datasets. This exploration enables analysts to identify trends and patterns, generate and verify hypotheses, and detect outliers and anomalies. However, as datasets grow in size and complexity, visual data exploration becomes challenging. In particular, to understand how a global pattern came about, an analyst may need to explore different subsets of the data to see whether the same or different pattern manifests itself in these subsets. Unfortunately, manually generating and examining each visualization in this space of data subsets (which grows exponentially in the number of attributes) presents a major bottleneck during exploration.

One way of navigating this combinatorial space is to perform *drill-downs* on the space—a *lattice*—of data subsets. For example, a campaign manager who is interested in understanding voting patterns across different demographics (say, race, gender, or social class) using the 2016 US election exit polls [50] may first generate a bar chart for the entire population, where the x-axis shows the election candidates and the y-axis shows the percentage of votes for each of these candidates. In Figure 3.1, the visualization at the top of the lattice corresponds to the overall population. The analyst may then use their intuition to drill down to specific demographics of interest, say gender-based demographics, by generating bar charts for female voters by following the purple path, as shown in the second visualization at the second row of Figure 3.1, and then to the visualization corresponding to `African-American Female` voters in the third row.

Challenges with Manual Drill-down

There are three challenges associated with manual drill downs:

First, it is often not clear which attributes to drill-down on. Analysts may use their intuition to select the drill-down attribute, but such arbitrary exploration may lead to large portions of the lattice being *unexplored—leading to missed insights.*

Second, a path taken by analysts in an uninformed manner may lead to visualizations that are *not very surprising or insightful.* For example, an analyst may end up wasting effort by drilling down from the `African-American` visualization to the `African-American Female` one in Figure 3.1, since the two distributions are similar and therefore not very surprising.

Third, an analyst may encounter a *drill-down fallacy*—a new class of errors in reasoning we identify—where incomplete insights result from potentially confounding factors not explored along a drill-down path. As shown in Figure 3.1, an analyst can arrive at the `African-American Female` visualization via the purple or the orange drill-down path. An analyst who followed the purple path may be surprised at how drastically the `African-American Female` voting behavior differs from that of `Female`. However, this behavior is not surprising if the analyst had gone down the orange path that we saw earlier, where the proper reference (i.e., the distribution for `African-American`) explains the vote distribution for `African-American Female`. In other words, even though the vote distribution for

`African-American Female` is very different from that of `Female`, the phenomenon can be explained by a more general "root cause" attributed to the voting behavior for the `African-American` community as a whole. Attributing an overly specific cause to an effect, while ignoring the actual, more general cause, not only leads to less interpretable explanations for the observed visualizations, but can also lead to erroneous decision-making. For example, for the campaign manager, this could lead to incorrect allocation of campaign funds. To prevent analysts from falling prey to such drill-down fallacies—consisting of misleadingly "surprising" local deviations in trend during drill-down (`Female` → `African-American Female`)—it is important to preserve the proper parent reference (`African-American`) to contextualize the behavior of the visualization of interest (`African-American Female`). One approach to avoid this fallacy is to exhaustively explore all potential drill-down paths. Unfortunately, this approach does not scale.

While there have been a number of statistical reasoning fallacies that have been identified in visual analytics, including Simpson's paradox [69, 10], multiple comparisons [249], and selection bias [63], to the best of our knowledge, this work is the first to identify the drill-down fallacy, a common fallacy that appears during manual data exploration. There have also been efforts to develop visualization recommendation systems [121, 220] that assist or accelerate the process of visual data exploration [220, 200, 242, 101, 99, 22, 107, 183]. In particular, our approach is most closely related to Sarawagi et al.'s seminal work on discovery-driven OLAP cube exploration [183, 182, 184], where they develop a technique to identify interesting regions of the data by discovering cells that are maximally different from the expected values. Our work extends beyond this approach by developing the notion of informative comparisons with respect to a well-chosen reference, rather than the exhaustive enumeration of the entire combinatorial space of data subsets. As we will see in this chapter, this choice of informative reference and the recommendation of a k-connected subset of visualizations in our problem formulation enables users of our system to avoid drill-down fallacies.

VisPilot with Safety, Saliency, and Succinctness

We present a visual data exploration tool, titled VisPilot, that addresses the three aforementioned challenges of exploration by espousing three principles: (i) **Safety** (i.e., ensure that proper references are present to avoid drill-down fallacies), (ii) **Saliency** (i.e., identify interesting visualizations that convey new information or insights), and (iii) **Succinctness** (i.e., convey only the key insights in the dataset). To facilitate safety, we develop a notion of *informativeness*—the capability of a reference parent visualization to explain the visualization of interest. To facilitate saliency, we characterize the notion of *interestingness*—the difference between a visualization and its informative reference in terms of underlying data distribution. Finally, to facilitate succinctness, we embrace a collective measure of visualization utility by recommending a *compact* connected network of visualizations. Based on these three principles, VisPilot *automatically identifies a compact network of informative and interesting visualizations that collectively convey the key insights in a dataset.* Our user

study results demonstrate that VisPilot can help analysts gain a better understanding of the dataset and help them accomplish a variety of tasks.
Our contributions include:

- Identifying the notion of a *drill-down fallacy*;
- Introducing the concept of *informativeness* that helps identify insights that arise from something that holds in the data (as opposed to confounding local phenomena);
- Extending the concept of informativeness to a measure to quantify the benefit of a network of visualizations;
- Designing VisPilot, which efficiently and automatically identifies a network of visualizations conveying the key insights in a dataset; and
- Demonstrating the efficacy of VisPilot through a user study evaluation on how well users can retrieve interesting visualizations, judge the importance of attributes, and predict unseen visualizations, against two baselines.

3.2 Problem Formulation

In this section, we first describe how analysts manually explore the space of data subsets. We then introduce three design principles for a system that can automatically guide analysts to the key insights.

Manual Exploration: Approach and Challenges

During visual data exploration, an analyst may need to explore different subsets of the data that together form a combinatorial *lattice*. Figure 3.1 shows a partial lattice for the 2016 US election dataset. The lattice contains the overall visualization with no filter at the first level, all visualizations with a single filter at the second level (such as `Female`), all visualizations with two filters at third level, and so on. Analysts explore such a combinatorial lattice from top to bottom, by generating and examining visualizations with increasing levels of specificity. In particular, analysts perform *drill-downs* [66] to access data subsets at lower levels by adding one filter at a time (such as adding `African-American` to `Female` along the purple path) and visualize their measures of interest for each data subset—in this case the percentage of votes for each candidate. Further, as analysts perform drill-downs, they use the most recent visualization in the drill-down path—the *parent*—as a *reference* to establish what they expect to see in the next visualization in the path—the *child*. In Figure 3.1, the visualizations `Female` and `African-American` are the *parents* of the `African-American Female` visualization, explored along the purple and orange path respectively.

As we saw in the purple path in Figure 3.1, while performing drill-downs, analysts may detect a local deviation (we will formalize these and other notions subsequently) between

a parent and a child to be significant. For example, they may be surprised by the fact that the `Female` and `African-American Female` visualizations are very different from each other, and may find this to be a novel insight. However, this deviation is a result of `Female` not being an *informative* parent or reference for `African-American Female`—instead, it is a *deceptive* reference. Here, a different parent, `African-American`, is the most informative parent or reference of `African-American Female` because it is the parent that exhibits the least deviation relative to `African-American Female`. Here, the `African-American Female` visualization is not really all that surprising given the `African-American` visualization. We refer to this phenomenon of being deceived by a local difference or deviation relative to a deceptive reference as an instance of the *drill-down fallacy*. One way to avoid such fallacies is to ensure that one or more informative parents are present for each visualization so that analysts can contextualize the visualization accurately. While this fallacy is applicable to any chart type that can be described as a probability distribution over data (e.g., pie charts, heatmaps), we will limit our discussion to bar charts for brevity.

The "3S" Design Principles

Our goal is to help analysts discover the key insights in a dataset while avoiding drill-down fallacies. We outline three essential principles for finding such insights—the three S's: *safety*, *saliency*, and *succinctness*, and progressively layer these principles to formalize a measure of utility for a network of visualizations. We adopt these principles to develop a visual exploration tool that automatically generates a network of visualizations conveying the key insights in a multidimensional dataset.

Safety

To prevent drill-down fallacies, we ensure *safety*—by making sure that informative parents are present to accurately contextualize visualizations. A parent is said to be *informative* if its data distribution closely follows the child visualization's data distribution, since the presence of the parent allows the analyst to form an accurate mental model of what to expect from the child visualization. We compute the informativeness of the j^{th} parent V_i^j for a visualization V_i as the similarity between their data distributions measured using a distance function D. For bar charts, the data distribution refers to the height of bars assigned to the categories labeled by the x-axis, suitably normalized. Accordingly, the computed distance $D(V_i, V_i^j)$ refers to the sum of the distances between the normalized heights of bars across different categories. Quantifying deviation using distances between normalized versions of visualizations in this manner is not a novel idea—we leverage prior work for this [220, 200, 137, 44, 183].

The specific distance measure D is not important; while we use the Euclidean metric, we can easily work with other common distance metrics such as Kullback-Leibler Divergence

and Earth Mover's distance [220]. The most informative parent $V_i^\dagger$ for a visualization V_i is the one whose data distribution is most similar to V_i.

$$V_i^\dagger = \underset{V_i^j}{argmin}\ D(V_i, V_i^j) \tag{3.1}$$

Instead of insisting that the most informative parent is always present to contextualize a given child visualization, we relax our requirement somewhat: we don't need *the most* informative parent to be present, just *an* informative parent. We define a parent to be informative (denoted V_i^*) if its distance from the child falls within a threshold θ% of the most informative parent—the default is set to 90% and adjustable by the user.

Saliency

Simply ensuring that informative parents are present is insufficient; we also want to emphasize *saliency* by identifying visualizations that convey new information. In general, a visualization is deemed to be *interesting* if its underlying data distribution differs from that of its parents, and thus offers new unexpected information or insight. Such distance-based notions of interestingness have been explored in past work [37, 95, 220], where a large distance from some reference visualization indicates that the selected visualization is interesting. We deviate from this prior work in two ways: first, we concentrate on *informative* interestingness, where the interestingness of a child visualization is only defined with respect to informative parent references. Second, we weigh the interestingness by the proportion of the population captured by the child visualization. (That is, when a deviation is manifested in a larger population, it is deemed to be more significant and therefore more interesting.) Thus, we define the utility of a visualization V_i, $U(V_i)$ as follows:

$$U(V_i) = \begin{cases} \frac{|V_i|}{|V_i^*|} \cdot D(V_i, V_i^*) & \text{if } V_i^* \text{ is present} \\ -\infty & \text{otherwise} \end{cases}$$

That is, the utility or interestingness of a visualization is the distance between the visualization and its informative parent, if present[1]. To incorporate the effect of subpopulation size into our objective function, we multiply the distance $D(V_i, V_i^*)$ between an informative parent V_i^* and a child visualization V_i by the ratio of their sizes. Notice that the objective U has a minimax form [233], in that informativeness aims to minimize the distance between parent and child, while interestingness aims to maximize the resulting minimum distance. For convenience, we define $U(V_0)$, where V_0 is the overall visualization, to be 1, which is the maximum value that the expression $\frac{|V_i|}{|V_i^*|} \cdot D(V_i, V_i^*)$ can take, ensuring that the overall visualization is always valuable to include.

[1]If multiple informative parents, V_i^*, are present for a given visualization, V_i, then $U(V_i)$ is defined in terms of the most informative parent present.

Succinctness

We cannot possibly display all of the visualizations in the lattice of data subsets: this lattice scales exponentially in the number of attributes. Instead, we aim for *succinctness*, where we only select a subset S of size $|S| = k$ from all the visualizations. We define the utility of S as follows:

$$U(S) = \sum_{V_i \in S} U(V_i)$$

In this subset, for every visualization except for the overall visualization, one of its informative parents must be present (otherwise $U = -\infty$). Thus, this subset ends up being a connected network (a sub-graph of the overall lattice) rooted at the overall visualization, ensuring that for each visualization, there is an informative parent available for context. We can now formally define our problem statement.

PROBLEM. *Given a dataset and user-provided X, Y attributes, select a subset S of $|S| = k$ visualizations from the lattice of data subsets $\mathcal{L}$, such that $U(S)$ is maximized.*

Thanks to how we have defined U, S will include the overall visualization, corresponding to the entire dataset with no filter. And, for each visualization in S except the overall one, at least one of its informative parents will be present in S. This network of visualizations S can be displayed on a dashboard.

Since the edges between non-informative parents to children are not pertinent to the solution, we can remove those edges from the lattice, leaving only the edges from the informative parents to the children. Then, we are left with an arbitrary graph, from which we need to select a rooted subgraph of size k, with greatest utility U. For arbitrary distance metrics D, this problem can be viewed to be NP-HARD via a reduction from the NP-HARD problem of selecting items with prerequisites [165] (specifically, the AND graph variant). Next, we design an approximate algorithm to solve this problem.

3.3 VisPilot: Our Solution

We present our system, VISPILOT, by first providing a high-level overview of the underlying algorithm, and then describing the user interaction mechanisms.

Lattice Traversal Algorithm

For a given dataset and user-selected X and Y axes, we first enumerate all possible attribute-value combinations (i.e., filters) to construct the lattice upfront. Like we described in the previous section, we retain only the edges that correspond to informative parents. Then, we traverse this pruned lattice to select the connected subgraph S of k visualizations (or equivalently, nodes in the lattice) that maximizes the utility U. Our algorithm for traversing the lattice, titled *frontier-greedy*, is inspired by the notion of "externals" in Parameswaran et al. (2010) [165]. The algorithm incrementally grows a subgraph S' until k nodes are selected.

Throughout, the algorithm maintains a set of *frontier* nodes $\mathcal{F}$—nodes that are connected to the existing subgraph solution S' but have not yet been added. The frontier nodes includes all of the children of the nodes in S'. Given that our pruned lattice only retains edges between children and their informative parents, all frontier nodes are guaranteed to have an informative parent in the the existing solution and can be added to S without violating informativeness. At each iteration, the algorithm adds the node from the frontier nodes that leads to the greatest increase in the utility of S': i.e., the node V_n such that $U(S' \cup \{V_n\})$ is the largest. Figure 3.2 displays how the algorithm maintains the list of frontier nodes (in green), and the current S' (in blue), adding the node that leads to the greatest increase in utility (in yellow). Algorithm 1 provides the pseudocode.

Algorithm 1 Frontier Greedy Algorithm

```
1: procedure PickVisualizations(k, 𝓛)
2:     S' ← {V_0} /* adding the overall node */
3:     while |S'| < k do
4:         𝓕 ← getFrontier(S', 𝓛)
5:         bestUtility ← −∞
6:         for V_i ∈ 𝓕 do
7:             if U(S' ∪ {V_i}) >bestUtility then
8:                 maxNode ← V_i
9:                 bestUtility ← U(S' ∪ {V_i})
10:        S' ← S'∪ {maxNode}
       return S'
```

User Interaction

Given the visualizations in S', we can render these visualizations in a dashboard, where users can inspect the visualizations through panning and zooming with navigation buttons, mouse clicks, and key bindings. Users can also select the x and y axes of interest, aggregation function, and set the number of visualizations (k) to generate a dashboard. Figure 3.3 displays VisPilot in action on the Police stop dataset [170]. The dataset contains records of vehicle and pedestrian stops from law enforcement departments in Connecticut, dated from 2013 to 2015. In this case, the analyst is interested in the percentages of police stops (Y) that led to different outcomes (X), such as ticket, warning, or arrest. As shown in Figure 3.3a, the analyst may begin by generating a 7-visualization dashboard. They would learn that if a search is conducted (`search_conducted=t`), then the probability of being arrested increases from 6.2% to 42.1%. However, the probability goes down to 23.1% if the driver is Asian (`driver_race=Asian, search_conducted=t`). When examining these visualizations, the analyst can be confident that any deviations are both informative and interesting: that is, the informative parents are present for each child, making the takeaways more significant. Moreover, the analyst may learn that for drivers who had contraband found in the vehicle

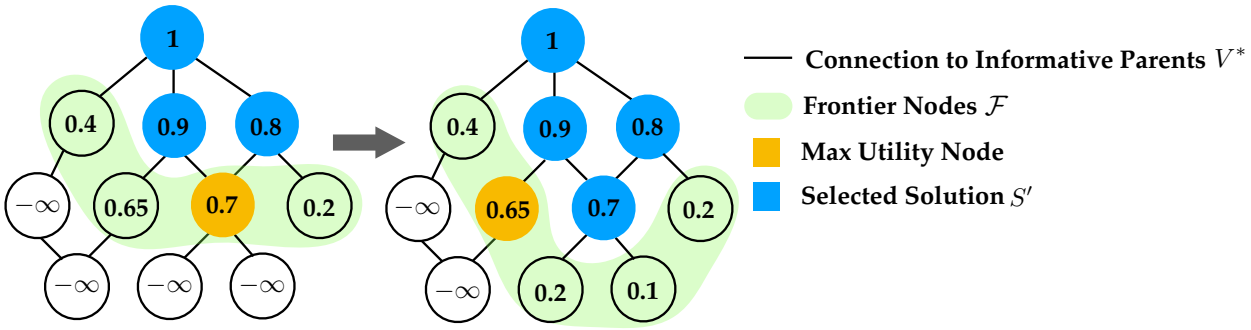


Figure 3.2: Example illustrating how the frontier greedy algorithm incrementally builds up the solution by selecting the node or visualization that leads to the highest gain in utility from the frontier at every step. Starting from a pruned lattice comprising only connections to informative parents (left) and three nodes in the existing solution (blue), we select the node with the highest utility gain (yellow) amongst the frontier nodes (green). The contribution to the utility of a node/visualization is depicted as the number within the node. On the right, the newly added node results in an updated frontier and the node leading to the highest utility gain is selected among them.

(`contraband_found=t`), the arrest rate for those who are 60 and over is surprisingly higher than usual, whereas for Asian drivers the arrest rate is lower.

After browsing through visualizations in the dashboard, the analyst may be interested in getting more information about a specific visualization. VisPilot allows analysts to perform additional drill-downs by requesting a new dashboard centered on a chosen visualization of interest as the new starting point (or equivalently, the root of the lattice) for analysis. Say the analyst is now interested in learning more about the other factor that contributes to high arrest rates: a long stop with `duration=30+min`. In Figure 3.3b, they can click on the corresponding visualization to request additional visualizations. Upon seeing the updated dashboard in Figure 3.3c, they learn that any visualization that involves the `duration=30+min` filter is likely to result in high ticketing and arrest rates. This implies that if a police stop lasts more than 30 minutes, the outcome would more or less be the same, independent of other factors such as the driver's race or age. To generate the expanded dashboard, VisPilot uses the same models and algorithms as before, except the selected visualization is set as the the overall visualization V_0 at the root node of the new lattice. This node expansion capability is motivated by the idea of *iterative view refinement* common in other visual analytics systems, which is essential for users to iterate on and explore different hypotheses [85, 242].

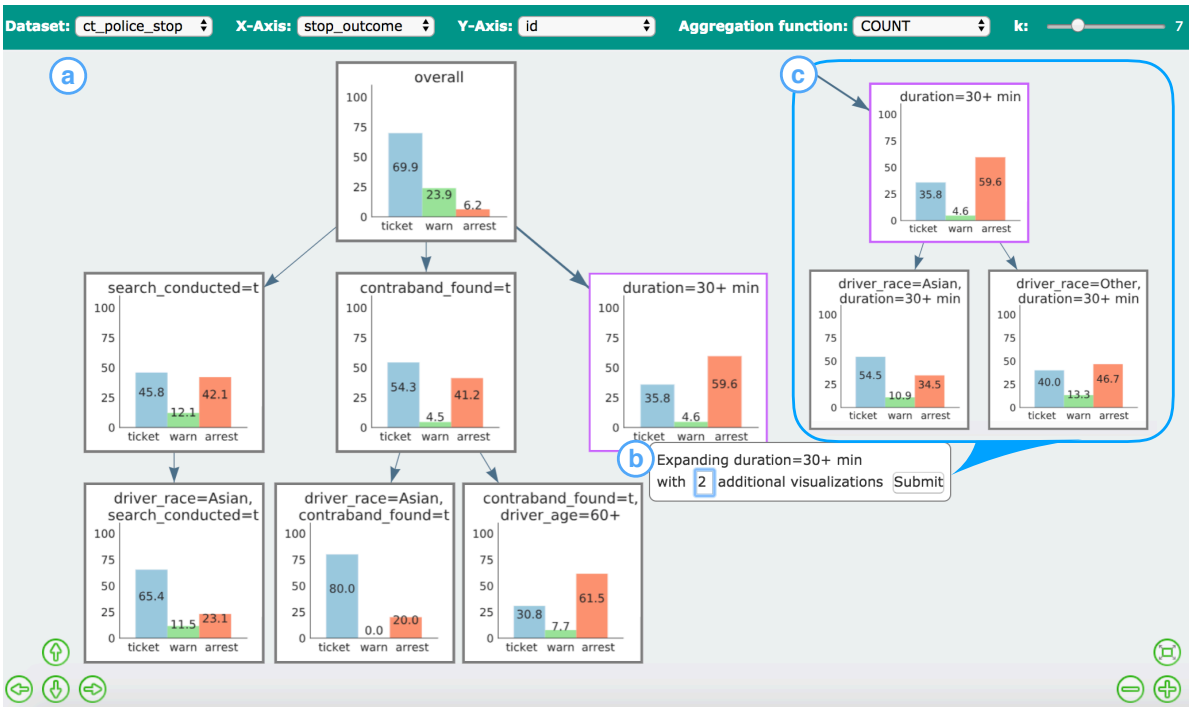


Figure 3.3: a) Overview of the VISPILOT interface for the Police Stop dataset. Users can select x, y axes, and aggregation function via the dropdown menu, to define the visualization space of interest, as well as adjusting dashboard parameters, such as the number of visualizations to show in the dashboard (k) via the sliders. b) User clicks on the duration=30+min visualization to request 2 additional visualizations. c) A preview of the added portion of the resulting dashboard is shown.

3.4 Evaluation Study Methods

In this section, we describe the methodology for a user study we conducted for evaluating the usefulness of VISPILOT for various exploratory analysis tasks. We aim to evaluate whether VISPILOT's "3S" design principles enables analysts to effortlessly identify insights in comparison with conventional approaches for multidimensional data exploration.

Participants and Conditions

We recruited 18 participants (10 Male; 8 Female) with prior experience in working with data. Participants included undergraduate and graduate students, researchers, and data scientists, with $1 - 14$ years of data analysis experience (average: 5.61). No participants reported prior experience in working with the two datasets used in the study (described below). Participants were randomly assigned two of the three types of dashboards with $k = 10$ visualizations generated via the following conditions.

VisPilot: The dashboards for this condition are generated by the aforementioned frontier greedy algorithm and displayed in a hierarchical layout as in Figure 3.3. To establish a fair comparison with the two other conditions, we deactivated the interactive node expansion capabilities.

BFS (short for breadth-first search): Starting from the visualization of the overall population, k visualizations are selected level-wise, traversing down the subset lattice, adding the visualizations at the first level with 1-filter combination one at a time, and then visualizations with 2-filter combinations, and so on, until k visualizations have been added. This baseline is designed to simulate a dashboard generated by a meticulous analyst who exhaustively inspects all visualizations (i.e., filter combinations) from the top down. These visualizations are then displayed in a 5×2 table.

Cluster: In this condition, k-means clustering is first performed on the data distributions of all of the visualizations in the lattice. This results in k clusters that cover the rest of the visualizations. For each cluster, we select the visualization with the least number of filter conditions as the cluster representative for interpretability and display them in a 5×2 table layout. This baseline is designed to showcase a diverse set of distributions within the dataset.

Dataset Descriptions. Each participant was assigned two different conditions on two different datasets (Police Stop and Autism, described below). The ordering of each condition was randomized to prevent confounding learning effects. The study began with a 5-minute tutorial using dashboards generated from the Titanic dataset [211] for each condition. To prevent bias across conditions, participants were not provided an explanation of how the dashboards were generated and why the visualizations were arranged in a particular way.

The first dataset in the study was the aforementioned Police Stop dataset. The attributes in the dataset include driver gender, age, race, stop time of day, stop outcome, whether a search was conducted, and whether contraband was found. We generated dashboards of bar chart visualizations with x-axis as the stop outcome (i.e., whether the police stop resulted in a ticket, warning, or arrest) and y-axis as the percentage of police stops that led to each outcome.

The second dataset in the study was the Autism dataset [58], describing the results of autism spectrum disorder screening for 704 adults. The attributes in the dataset are binary responses to 10 diagnostic questions as part of the screening process. This dataset serves as a data-agnostic condition, since there was no descriptions of the questions or answer labels provided to our study participants. We generated dashboard visualizations based on the percentage of adults that were diagnosed with autism.

Study Procedure

After the tutorial, for each dataset, participants were given some time to read through a worksheet containing the descriptions of the data attributes. Then, they were given an at-

tention check question where they were provided a verbal description of the visualization filter (i.e., data subset) and asked about the corresponding visualization in the dashboard. After understanding the dataset and chart schema, participants were asked to accomplish various tasks. Since VisPilot was developed based on a joint utility objective, it is impossible to design tasks that evaluate each of the "3S" principles individually. Instead, our tasks were selected to measure the overall efficacy and usefulness of the dashboards in helping a participant understand and become aware of different aspects of and insights within a dataset during drill-down analysis. These different aspects of dataset understanding can be roughly illustrated via Figure 3.2, from insights gained from *individual* displayed visualizations (blue selected nodes), to predicting behavior of *related* visualizations (green related nodes), to understanding *overall* attribute importance (entire lattice, a mix of green, blue, and unselected white nodes).

Labeling (Individual Assessment): Participants were asked to talk aloud as they interpreted the visualizations in the dashboard and label each one as interesting or not interesting, or leave it unselected. This subjective task measures how interesting *individual selected visualizations were* to participants.

Prediction (Related Assessment): Participants were given a separate worksheet and asked to sketch an estimate for a visualization that is not present in the dashboard. For every condition, the visualization to be estimated contained 2 filter combinations, with exactly one parent present in the given dashboard. After making the prediction, participants were shown the actual data distribution and asked to rate on a Likert scale of 10 how surprising the result was (1: not surprising and 10: very surprising). This task measured how well participants inferred the behavior of *related*, unobserved visualizations based on a limited set of selected dashboard visualizations.

Ranking (Overall Assessment): Participants were given a sheet of paper with all the attributes listed and asked to rank the attributes in order of importance in contributing to a particular outcome (e.g., factors leading to an arrest or autism diagnosis). Participants were allowed to assign equal ranks to more than one attribute or skip attributes that they were unable to infer importance for. Attribute ranking tasks are common in many data science use-cases, such as feature selection and key driver analysis. Since all dashboards were equal in size, our goal was to check whether this size limitation came at the cost of *overall* dataset understanding. Thus, the goal of this task was to study participant's overall dataset understanding by measuring how well participants judged the relative importance of each attribute.

At the end of the study, we asked two open-ended questions regarding the insights gained by participants and what they liked or disliked about each dashboard. On average, the study lasted around 48 minutes.

3.5 Study Results

We introduce the study findings for each task starting from the narrowest scope of *individual* visualizations to the widest scope of *overall* dataset understanding.

RQ1: How are *individual selected visualizations in the dashboard perceived subjectively by the users?*

Using click-stream data logged from the user study, we recorded whether a participant labeled each visualization in the dashboard as interesting, not interesting, or left the visualization unselected. Table 3.1 summarizes the counts of visualizations marked as interesting or not interesting aggregated across conditions. We also normalize the interestingness count by the total number of selected visualizations to account for variations in how some participants select more visualizations than others. The results indicate that participants who used VisPilot saw more visualizations that they found interesting compared to the BFS and Cluster conditions. While this task is inherently subjective, with many possible reasons why a participant may have marked a visualization as interesting, this result is indicative of the fact that the selected visualizations were deemed to be relevant by users. We will drill into the possible reasons why in the next section.

Condition	VisPilot	BFS	Cluster
Interesting	66	61	51
Not Interesting	10	20	22
Interesting (Normalized)	0.87	0.75	0.7

Table 3.1: Total counts of visualizations marked as interesting or not interesting across the different conditions. VisPilot leads to more visualizations marked as interesting and fewer visualizations marked as uninteresting.

RQ2: How well do dashboard visualizations provide users with an accurate understanding of *related visualizations?*

As discussed in Chapter 3.2, contextualizing visualizations correctly with informative references can help prevent users from falling prey to drill-down fallacies. To this end, the prediction task aims to assess whether users can employ visualizations in the dashboard to correctly predict unseen ones. Indeed, if the dashboard is constructed well, one would expect that visualizations that are not very surprising relative to their informative parents would be excluded from the dashboard (i.e., their deviation from their informative parents is not large).

The accuracy of participants' predictions is measured using the Euclidean distance between their predicted distributions and ground truth data distributions. As shown in Figure 3.4 (left), predictions made using VisPilot (highlighted in red) were closer to the actual distribution than compared to the baselines, as indicated by the smaller Euclidean distances. Figure 3.4 (right) also shows that VisPilot participants were able to more accurately reason

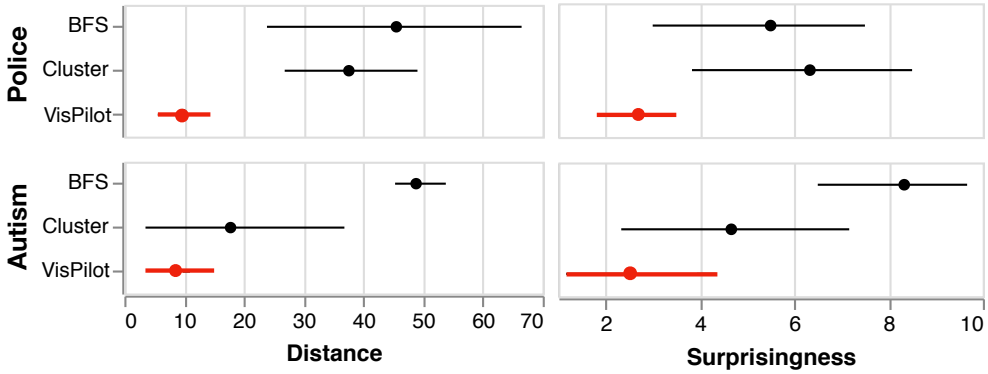


Figure 3.4: Left: Euclidean distance between predicted and ground truth. In general, predictions made using VISPILOT are closer to ground truth. Right: Surprisingness rating reported by users after seeing the actual visualizations on a Likert scale of 10. VISPILOT participants had a more accurate mental model of the unseen visualization and therefore reported less surprise than compared to the baselines.

about the expected properties of unseen data subsets (or visualizations), since they rated the resulting visualizations to be less surprising. CLUSTER may have performed better for the Police dataset than it did for the Autism one, for the same reason as in the attribute ranking task, where more univariate visualizations happened to be selected.

We also compute the variance of participants' predictions across the same condition. In this case, low variance implies that there is consistency or agreement between the predictions of participants who consumed the same dashboard, whereas high variance implies that the dashboard did not convey a clear data-driven story that could guide participants' predictions. So instead, participants had to rely on prior knowledge or guessing to inform their predictions. These trends can be observed in both Figure 3.4 and in more detail in Figure 3.5, where the prediction variance amongst participants who used VISPILOT is generally lower than the variance for the baselines. Overall, VISPILOT provides participants with a more accurate and consistent model of *related visualizations.*

RQ3: How well does the dashboard convey information regarding the *overall* dataset schema?

We use the common task of judging the relative importance of attributes as an indicator of the participants' overall understanding. To determine ground truth attribute importance, we computed the Cramer's V statistics between attributes to be ranked and the attributes of interest. Cramer's V is commonly used for determining the strength of association between categorical attributes [144]. We deem an attribute as important if it has one of the top-three[2] Cramer's V scores amongst all attributes of the dataset. For the list of rankings provided by each participant, we first remove attributes that participants chose not to rank. We compute the F-scores and average precision (AP) at k relative to the ground truth for various values

[2]This relevancy cutoff is visually-determined via the elbow method to indicate which rank the Cramer's V score drops off significantly.

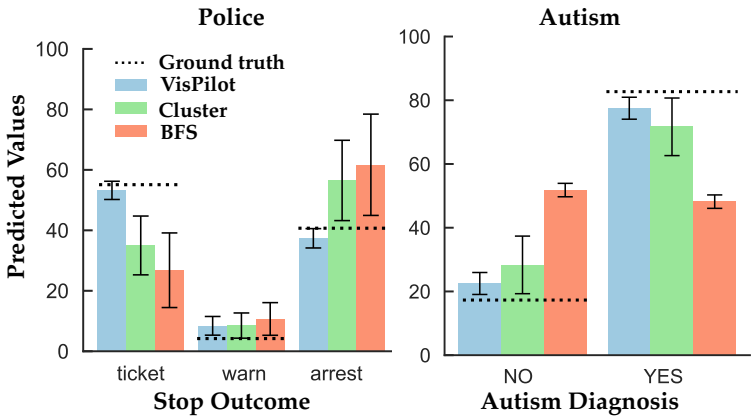


Figure 3.5: Mean and variance of predicted values. Predictions based on VISPILOT exhibit lower variance (error bars) and closer proximity to the ground truth values (dotted).

of k. Table 3.2 reports the average across participants in each condition, after picking the best performing k value for each participant based on F-score and AP respectively. Both measures capture how accurately participants were able to identify the three most important attributes for each dataset.

	Police		Autism	
Metric	F	AP	F	AP
VISPILOT	0.750	0.867	0.723	0.600
CLUSTER	0.739	0.691	0.725	0.665
BFS	0.739	0.592	0.222	0.200

Table 3.2: Best AP and F-scores for the attribute ranking task.

For this task, we expected BFS to have an inherent advantage, since BFS dashboards consist of all univariate distributions, providing more high-level, "global" information regarding each attribute. However, both VISPILOT and CLUSTER (which contained more "local" information) performed better than BFS. The problem with BFS is that given a limited dashboard budget of $k = 10$ visualizations that could be displayed, not all univariate distributions were shown. For the Police dataset, it happened to select several important attributes (related to contraband and search) to display in the first 10 visualizations. However, for Autism, only visualizations corresponding to binary diagnostic questions 1-4 fit in the dashboard. So the poor ranking behavior comes from the fact that the BFS generated dashboard failed to display the three most important attributes (questions 5, 6 and 9) given the limited budget. This demonstrates BFS's lack of consistency across different datasets, due to the fact that exhaustive exploration can only lead to limited understanding of the data.

We see that VISPILOT performs better than CLUSTER for the Police dataset and closely follows CLUSTER for the Autism dataset. It is not entirely surprising that CLUSTER did well,

since it is a well-established method for summarizing high-dimensional data [71]. For Autism, CLUSTER happened to pick the majority of visualizations (8/10) as univariate distributions that exhibited high-skew and diversity, leading to more informed inference of attribute importance. Since clustering seeks visualizations that exhibit diversity in the shape of the data distributions, it could potentially result in visualizations with many filter combinations. For the police dataset, 6 out of 10 visualizations had more than 2 filters, making it difficult to interpret the visualization without an appropriate context to compare against.

Overall, both BFS and CLUSTER do not provide consistent guarantees for highlighting important visualizations across different datasets. In general, our results indicate that participants gain a better *overall dataset* understanding regarding attribute importance using VISPILOT, with only a few targeted visualizations that tell the "entire story". This is without VISPILOT being explicitly optimized for the ranking task.

3.6 Discussion of Study Results

To further understand how participants made use of the recommended visualizations during their analysis, we analyzed the user study transcripts through an open coding process [151] by two of the authors. For each task in our study, we assigned a binary-valued code to indicate whether or not a participant engaged in a particular action or thought process. Table 3.4 highlights results from thematic coding discussed in this section. We will use the notation [Participant.DatasetAlgorithm] to refer to a participant engaging with a dashboard created by an algorithm={1,2,3}={VISPILOT, CLUSTER, BFS} on a dataset ={A,B}={Police, Autism}.

The Choice of Contextual References

As discussed earlier, analysts often make use of related visualizations to form their expectation or mental model for unseen visualizations. We refer to the visualizations used for such purposes as *contextual references*. The appropriate choice of a contextual reference (such as an informative parent) is necessary to ensure the *safety* of insights derived through drill-downs. To understand how "safe" the dashboards generated from each condition were, we examined the visualizations that participants compared against to inform unseen visualizations. In particular, we thematically encoded the participants' use of contextual references based on their verbal explanations for justifying their prediction task responses. As shown in Table 3.3, we find that participants make more comparisons in total using VISPILOT than CLUSTER and BFS.

Participants can (and often do) make comparisons against more than one type of contextual references to obtain their prediction. We uncovered four main classes of contextual references, described below using the example visualization V_i=`gender=F,age=21-30` (in the order of most to least similar to V_i):

1. **Parent** : Comparison against a visualization with one filter removed (e.g., `gender=F`)

Algorithm	Parent	Sibling	Relative	Overall	Total
VisPilot	12	8	0	11	31
Cluster	4	0	7	8	19
BFS	0	5	1	8	14

Table 3.3: Out of 12 participants, the number of participants who made use of each contextual reference across the two datasets. Participant behavior shows a similar trend in individual datasets. VisPilot participants made more comparisons in general and against parents compared to the baselines.

2. **Sibling** : Comparison against a visualization that shares the same parent. In other words, the filtered attributes are the same, but one filter has a different value. (e.g., `gender=F,age=60+`)

3. **Relative** : Comparison against a visualization that shares some common ancestor (excluding overall), but not necessarily the same parent. These visualizations share at least one common filter, but with more than one filter or filter value being different. (e.g., `gender=F,age=60+,race=White`)

4. **Overall** : Comparison against the distribution that describes the overall population (no filters applied).

Studying the participants' use of contextual references reveals inherent challenges that arise from using the BFS and Cluster dashboards. For Cluster, participants mainly compared against relatives and overall visualizations. Since Cluster optimizes the diversity of distributions amongst the selected visualizations, these visualizations had up to 4 filters and were disconnected from each other. For this reason, in many cases, participants could only rely on relatives and the overall visualization as contextual references. For example, P4.A2 pointed at a 4-filter visualization with extreme values (100% for warning; 0% for arrest and ticket) and indicated how "*a lot of* [the visualizations] *are far too specific. This is not very helpful. You can't really hypothesize that all people are* [sic] *going to be warned, because it is such a specific category, it might just be one person*". He further explained how he "*would not want to see the intersections* [(i.e., visualizations with many filters)] *at first and would want to see all the bases* [(i.e., univariate summaries)] *then dig in from there.*" The lack of informative contextual references in the Cluster dashboard is also reflected in how analysts exhibited high variance and deviation in their prediction responses.

Furthermore, improper comparisons against contextual references often make it difficult to interpret displayed visualizations. In particular, when visualizations composed of multiple filter conditions were shown in Cluster dashboards, 25% of the participants had trouble making sense of the meaning of a filter for at least one of the datasets (e.g., understanding that `gender=F AND age=60+ corresponds to female drivers with ages larger than 60 years old) at some point during the study`. In contrast, as shown in Table 3.4, this confusion only happened once for BFS and none for VisPilot. This is due to the fact that

CLUSTER dashboards seemed random to the users, making it challenging to find "close" contextual references to compare against. In contrast, the linear ordering of BFS and hierarchical ordering of VISPILOT were natural and interpretable for participants.

	VISPILOT	CLUSTER	BFS
Difficulty Interpreting Visualizations	0	3	1
Misjudged Significance of Population Size	0	4	1
Interpretable "Human-like" Dashboard	5	1	0
Number of Insights (Police)	11	8	9
Number of Insights (Autism)	16	6	11

Table 3.4: Summary of qualitative insights from thematic coding. We record the total number of insights based on overall dataset findings that were independently discovered by more than two different participants. For each participant, we coded the absence or presence of 7 such insights for the Police dataset and 6 insights for the Autism dataset.

For BFS, most comparisons were based on the overall visualization and siblings. Due to the sequential level-wise picking approach, the overall visualization corresponded to the immediate parent of all of the dashboard visualizations generated by BFS (all of which are univariate distributions for $k = 10$), so they are not explicitly recorded as a parent. While the overall and sibling comparisons can be informative, the incomplete comparisons, due to the limited number of first-level visualizations displayed, can result in flawed reasoning, as observed in the Autism prediction task. In contrast, for VISPILOT, almost all users compared against the overall one and parents, while some also exploited sibling comparisons to make weaker guesses for less-frequently observed attributes (e.g., using a 2-filter sibling visualization involving `driver_age` to infer another 2-filter visualization involving `driver_age` with a different parent.)

Interpretability of Hierarchical Layouts

In the post-study interviews, participants cited hierarchical layout as a key reason for why they preferred VISPILOT recommendations. Even though participants were never explicitly told what the edge connections between the visualizations meant during the study, they were able to interpret the meaning of the dashboards effortlessly through VISPILOT's hierarchical layout. For example, P1.A1 stated that "*the hierarchical nature* [is] *a very natural flow...so when you are comparing, you don't have to be making those comparisons in your head, visually that is very pleasing and easy to follow.*" Likewise, P9 described how VISPILOT's hierarchical layout for the Autism dataset was a lot easier to follow than the Police dataset shown in the table layout for CLUSTER:

> *If I had to look at this dataset in the format of the other one, this would be much more difficult. It was pretty hard for me to tell in the other one how to organize*

> *the tree, if there was even a tree to be organized. I like this layout much better, I think this layout allows me to approach it in a more meaningful way. I can decide, what do I think matters more: the overall trend? or the super detailed trends? and I know where to look to start, in the other one, every time I go back to it, I would say, where's the top level, where's the second level? I mentally did this. Like when you asked me that first question, it took much longer to find it, because I literally have to put every chart in a space in my head and that took a lot longer than knowing how to look at it.*

At the end of the study, some participants who were assigned dashboard conditions with 5×2 table layouts (i.e., BFS and CLUSTER conditions) sketched and explained how they would like the layout of the visualizations to be done. These participants expressed that they wanted "groupings" or layouts that arranged visualizations with the same attribute together. Other participants advocated for isolating the overall visualization outside of the dashboard table for facilitating easier comparisons. Both of these suggestions provide further motivation for our hierarchical organization of visualizations. Our findings echo prior work on visualization sequences and storytelling [88, 190, 26, 111] in that analysts prefer visualization sequences structured hierarchically based on shared data properties, such as ordering by increasing levels of aggregation.

Since we did not inform participants about how the dashboards were generated, it was surprising to see that some participants presumed that certain dashboards were hand-picked by a human analyst and hypothesized what this fictitious analyst's intentions were (e.g., "*It seems like the researcher who created this dashboard was specifically looking at people of Asian descent and people who are 60 or older.*" [P7.A1]). Table 3.4 shows how 5 out of 12 participants referred to the VISPILOT dashboards as if they were generated by a human, whereas only 1 participant for CLUSTER and none for BFS made such remarks. We encoded this phenomenon by looking at instances where a participant either explicitly referred to a person who picked out the dashboard or implicitly described their intentions through personal pronouns. At the end of the study, many were surprised to learn that the VISPILOT dashboard was actually picked out by an algorithm, indicating that VISPILOT could automatically generate convincing dashboards similar to ones that were authored with human intention. The interpretability of VISPILOT dashboards may have contributed to the increased number of insights discovered in both datasets compared to the two baselines, as summarized in Table 3.4.

Limitations of VisPilot

As described earlier, since the details of how the dashboards were obtained were not explained to the users during the study, some users expressed that they were initially confused by VISPILOT as not all variables were present in the dashboard. Others also found it confusing that the addition of filters did not always correspond to the same variables. For example, P2.A1 felt that the dashboard was intentionally biased:

> *I feel like this one, not all the data is here, so we are already telling a story, you are trying to steer the viewer to look at certain things. And the focus seems to be on where the arrest rate is high. You probably could have found other things that led to ticket being high, but you didn't pull those out. You are trying to see if there are other factors that lead to more arrests.*

This sentiment is related to participants' desire to perform their own ad-hoc querying alongside the dashboard to inspect other related visualizations for verifying their hypothesis. For example, P7.A1 wanted to inspect all other first-level visualizations for driver's race to assess its influence. P7.A1 expressed that while he had learned many insights from the dashboard, "*the only thing I don't like is I cannot control the types of filter, which is fixed.*" Since our current goal was to simply provide an informative dashboard and evaluate its utility, the present version of VisPilot is limited in its interactivity and the extent of free-form data exploration it supports. This result also points to how VisPilot could serve as a helpful assistant alongside other conventional visualization tools, such as Tableau. Outside the context of the user study, it is essential to explain how VisPilot selects the visualizations in an easy and interpretable manner to establish a sense of the summarization objectives for the users and help them make better inferences with the dashboard.

Since the goal of our study is to evaluate whether VisPilot can assist users in drill-down exploration, our preliminary study is limited to comparisons against baselines stemming from conventional approaches for multidimensional data exploration. While we understand how the VisPilot study condition may confound the hierarchical layout with the algorithmic choice of visualizations, our intention for the baseline was to simulate how analysts generate a large number of visualizations individually, typically arranged in a table grid layout, rather than using a hierarchical layout. Further evaluation comparing how different hierarchically-displayed visualization selection algorithms assist users in drill-down exploration is a direction of future work.

3.7 Conclusion

Common analytics tasks, such as causal inference, feature selection, and outlier detection, require comparing data distributions across different data subsets [8, 79, 243, 88]. However, without knowing *what* subset of data contains an insightful distribution, manually exploring distributions from all possible data subsets can be tedious and inefficient. Moreover, when examining data subsets by adding one filter at a time, analysts can fall prey to the drill-down fallacy, where they mistakenly attribute the interestingness of a visualization to a "local difference", while overlooking a more general explanation for the root cause of the behavior. To address these issues, we presented VisPilot, an interactive visualization recommendation system that automatically selects a small set of informative and interesting visualizations to convey key distributions within a dataset. Our user study demonstrates that compared to two other baselines VisPilot can guide participants toward more informed decisions for retrieving interesting visualizations, judging the relative importance of attributes,

and predicting unseen visualizations. Study participants also find dashboards generated by VisPilot to be more interpretable and "human-like", leading to more discovered insights.

VisPilot is one of the first visual exploration assistants that guides analysts across the space of data subsets by summarizing key insights with safety guarantees. In the following chapter, we look at another visual exploration assistant in our toolbox designed with a different visual analysis task in mind, namely, to address the challenge of line chart pattern search.

Chapter 4

Assistance during Visual Querying with Zenvisage++

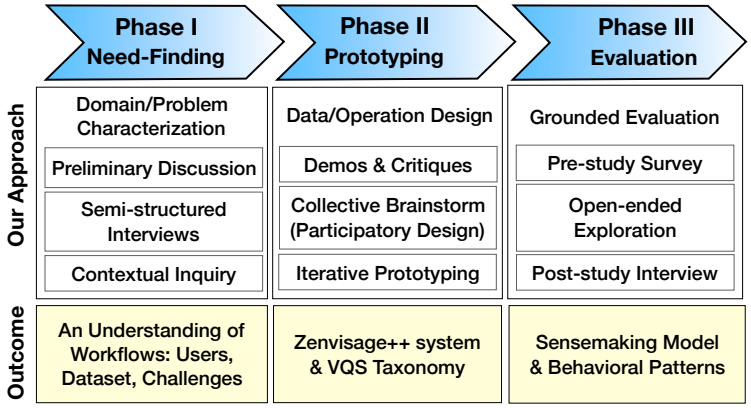


Figure 4.1: Lifecycle model summarizing our research approach and outcome of each phase.

Line chart exploration is a crucial process that helps analysts uncover insightful patterns and trends in their data. However, the current process of discovering line chart visualizations with actionable insights requires users to search through large numbers of visualizations manually, one at a time. This chapter describes our collaborative design process working with domain scientists across genetics, astronomy, and material science who experience this common pain point. Furthermore, we present design findings and guidelines from developing a visual query system for accelerating line chart pattern search.

4.1 Introduction

Line charts are commonly employed during data exploration—the intuitive connected patterns often illustrate complex underlying processes and yield interpretable and visually compelling data-driven narratives [59]. However, discovering line charts that display certain

meaningful patterns, trends, or characteristics of interest is often an overwhelming and error-prone process, consisting of manual examination of large numbers of line charts. For example, when trying to find supernovae, which exhibits a unique pattern of brightness over time (an initial peak followed by a long-tail decay), astronomers often have to manually construct and inspect thousands of line chart visualizations to find ones with their desired pattern. To address this exploration challenge, there has been a large number of papers dedicated to building *Visual Query Systems* (VQSs)—a term coined by Ryall et al. (2005) [180] to describe systems that allow users to specify and search for desired line chart patterns via visual interfaces [149, 80, 226, 199, 180, 36, 142, 49, 84][1]. These interfaces typically include a sketching canvas where users can draw a pattern of interest, with the system automatically traversing all potential visualization candidates to find those that match the specification.

While these intuitive specification interfaces were proposed as a promising solution to the problem of painful manual exploration of visualizations for time-series analysis [180, 226], to the best of our knowledge, VQSs have not lived up to these expectations and are not very commonly used in practice. One likely reason for the lack of VQS adoption may be attributed to how prior work has focused almost exclusively on optimizing the pattern-matching algorithms and interactions, with few invested in understanding actual user needs and how VQSs can be used for solving real-world problems. In this chapter, we seek to understand how VQSs can actually be used in practice, as a first step towards the broad adoption of VQSs in data analysis. Unlike prior work on VQSs, we set out to not only evaluate VQSs in-situ on real problem domains, but also involve participants from these domains in the VQS design. We present findings from a series of interviews, contextual inquiry, participatory design, and user studies with scientists from three different domains—astronomy, genetics, and material science—over the course of a year-long collaboration. The amount of time we invested in each of these three diverse domains surpasses the norm in this field and is key to uncovering the insights presented in this work. These domains were selected to capture a diverse set of goals and datasets wherein VQSs can help address important scientific questions, such as: How does a treatment affect the expression of a gene in a breast cancer cell-line? Which battery components have sustainable levels of energy-efficiency and are safe and cheap to manufacture in production?

In this work, we adapt methods from user-centered design (UCD) [158, 156, 64], such as interviews, contextual inquiry, and participatory design, into our design-implementation-evaluation cycle [194]; our methodology is summarized in Figure 4.1. Via contextual inquiry and interviews, we first identified challenges in existing data analysis workflows in these domains that could be potentially addressed by a VQS. Building on top of an existing open-source VQS, Zenvisage [200, 199], we iterated on the design of the VQS with participants over the course of a year to better compose data exploration workflows that lead to insight discovery. Rather than targeting a domain-specific solution, we engaged with multiple domains to observe differences and commonalities across domains and synthesize high-level insights regarding the use of VQSs. While conducting this multi-phased, mixed-methods

[1]We covered related work on VQSs in Section 2.2 in Chapter 2.

research agenda across three diverse use cases was challenging, this endeavor was necessary for addressing the qualitative, participant-centered research questions investigated.

We organize our design study findings into a taxonomy of VQS capabilities, involving three sensemaking processes inspired by Pirolli and Card's notional model of analyst sensemaking [171]. The sensemaking processes include *top-down pattern search* (translating a pattern "in-the-head" into a visual query), *bottom-up data-driven inquiries* (querying or recommending based on data), and *context-creation* (navigating across different collections of visualizations). We find that prior VQSs have focused on enabling top-down processes (via sketching capabilities), but have largely overlooked the two other processes that we found to be essential in all three domains. These missing capabilities partially explain why prior VQSs have not been widely adopted in practice.

We finally conducted an evaluation study with nine participants using our final VQS prototype to address their research questions on their own datasets. During this study, participants gained novel scientific insights, such as identifying a star that was known to harbor a Jupiter-sized planet, discovering a previously-unknown relationship between solvent properties, and finding characteristic gene expression profiles confirming the results of a related publication.

During this evaluation study, we were somewhat surprised to discover that sketching a pattern for querying is often ineffective on its own. This is due to the fact that sketching makes the assumption that users know the pattern that they want to sketch and are able to sketch it precisely. However, this is typically not the case in practice. For example, the geneticists from our study often did not have a preconceived knowledge of what to sketch for and relied heavily on VQS-recommended common and outlying patterns to jumpstart their queries. Likewise, while the material scientists from our study were interested in datapoints that fall within specific value-ranges, they did not have an apriori notion of what their desired patterns would look like. Overall, participants typically opted to combine sketching with other means of pattern specification—one common mechanism was to drag-and-drop a recommended pattern onto the canvas, and then modify it (e.g., by smoothing it out).

To further understand how participants engaged with VQSs in their analytical workflows, we constructed a Markov model to characterize how participants transitioned between different sensemaking processes during their analysis. We found that participants often constructed a diverse set of analytical workflows tailored to their domains by focusing around a primary sensemaking process, while iteratively interleaving their analysis with the two other processes. This finding points to how all three sensemaking processes, along with seamless transitions between them, are crucial for enabling the effective use and adoption of VQSs for addressing real-world challenges.

To the best of our knowledge, our study is the first to holistically examine how VQSs can be designed to fit the needs of real-world analysts, and how they are actually used in practice. Working with participants from multiple domains enabled us to compare the differences and commonalities across different domains, thereby identifying general VQS challenges and requirements for supporting common analytical goals. Our contributions include:

- a characterization of the problems addressable by VQSs through design studies with three different domains,
- a taxonomy of essential VQSs capabilities, leading to a sensemaking model for VQSs,
- an integrative VQS, ZENVISAGE++ capable of facilitating rapid hypothesis generation and insight discovery, resulting from iteration with end-users,
- study findings on how VQSs are used in practice, leading to the development of a novel sensemaking model for VQSs.

Our work not only opens up a new space of opportunities beyond the narrow use cases considered by prior studies, but also advocates common design guidelines and end-user considerations for building next-generation VQSs.

4.2 Methods

Via interviews and contextual inquiry in participants' normal work environments, we first identified the needs and challenges in participants' existing data analysis workflows. Given these challenges, we collaboratively designed VQS functionalities by engaging with experts from three different domains throughout the design process, leading to a final prototype ZENVISAGE++. After the design phase, we conducted an evaluation study to understand how VQSs are used in the real-world analytical workflows. Our research methodology is illustrated in Figure 4.1; we now describe the study procedure in more detail.

Phase I: Need-finding

We recruited participants by reaching out to research groups who have experienced challenges in data exploration, via email and word-of-mouth. Based on early conversations with analysts from 12 different potential application areas, we narrowed down to three use cases in astronomy, genetics, and material science through a process similar to the "*winnow*" stage in Sedlmair et al. (2012) [189]. The domains were chosen based on their suitability for VQSs as well as diversity in use cases. Six scientists, with extensive research experience in their respective fields, participated in the design process. Via interviews and contextual inquiries, we interviewed participants to learn about their dataset and research questions, shadowed participants in conducting their existing analysis workflows, and subsequently discussed the needs and challenges of their use cases. The interviews were semi-structured and focused on how the analytical tasks in their workflows relate to the scientific questions they were interested in.

	ID	Dataset	Participated in Design	Position	Years of Experience	Dataset Familiarity
Astronomy	A1	DES	✓	Researcher	10+	3
	A2	Kepler		Postdoc	8	5
	A3	Kepler		Postdoc	8	5
Genetics	G1	Mouse	✓	Grad Student	4	4
	G2	Cancer		Grad Student	2	2
	G3	Mouse	✓	Professor	10+	2
Material Science	M1	Solvent (8k)	✓	Postdoc	4	5
	M2	Solvent (Full)	✓	Professor	10+	5
	M3	Solvent (Full)	✓	Grad Student	3	5

Table 4.1: Participant information. The Likert scale used for dataset familiarity ranges from 1 (not familiar) to 5 (extremely familiar).

Phase II: Collaborative Prototyping

For iterative prototyping, we built on top of an existing open-source VQS, Zenvisage [200, 199], to create a functional prototype to showcase the capabilities of VQSs. The use of functional prototypes is a common and effective way of engaging with participants, by providing a starting point for collaborative design [31]. We collaborated with each team closely with approximately two 1-hour-long meetings per month, where we learned more about their datasets, objectives, and what additional VQS functionalities could help address their research questions. During these meetings, we collectively brainstormed with participants on the design of the prototype. Participants also had the opportunity to interact with the prototype through the help of a guided facilitator. Through these excercises, we elicited feedback from participants on how the VQS could better support their scientific goals and identified and incorporated several crucial capabilities into Zenvisage++. A summary timeline of our collaboration with participants over a year can be found in Figure 4.2.

Phase III: Grounded Evaluation

After the prototyping phase, we performed a qualitative evaluation to study how analysts interact with different VQS components in practice. Participants used datasets that they have a vested interest in exploring to address unanswered research questions (a total of six different datasets across nine participants). The evaluation study participants included the six scientists from Phase I and II, along with three additional "blank-slate" participants who had never encountered Zenvisage++ before[2] The use of all or a subset of the project stakeholders as evaluation participants is typical in participatory design [24]. While the small sample size of participants may be viewed as a limitation, this is a pervading challenge when recruiting domain-experts[15, 145], whose specific expertise and skills are rare and have

[2]Details regarding participants can be found in Table 4.1.

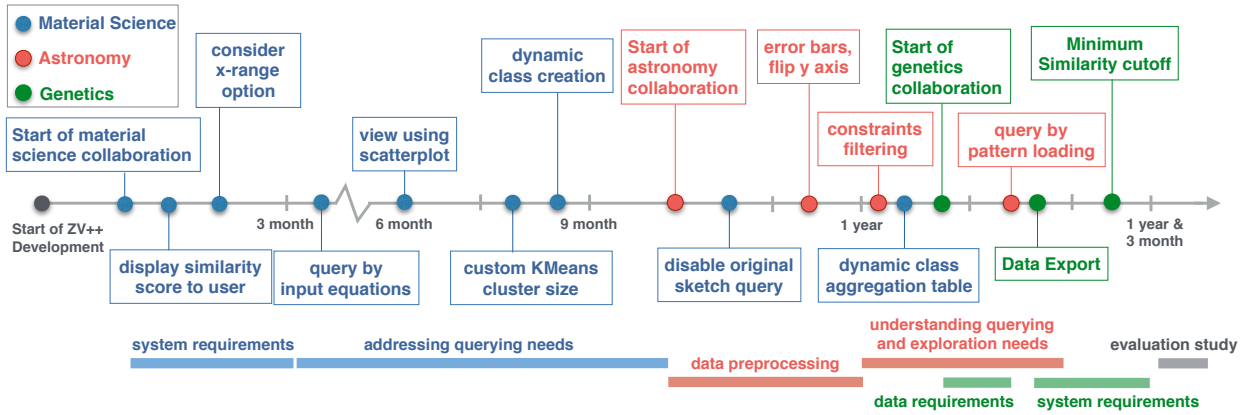


Figure 4.2: Timeline for progress in participatory design studies.

limited time due to their workplace demands relative to the general population. Nevertheless, even studies with a small group of domain experts involved are invaluable for understanding expert needs [189].

Evaluation study participants were recruited from each of the three aforementioned research groups, as well as domain-specific mailing lists. Prior to the study, we asked potential participants to fill out a pre-study survey to determine eligibility. Eligibility criteria included: being an active researcher in the subject area with more than one year of experience, and having worked on a research project involving data of the same nature used in the design phase. None of the participants received monetary compensation for the study, as this is not a common practice for collaborative design with stakeholders [161, 146].

At the start of the in-lab evaluation study, participants were provided with an interactive walk-through of ZENVISAGE++ and given approximately ten minutes for a guided exploration of a preloaded real-estate example dataset from Zillow [253]. This dataset contained housing data for various cities, metropolitan areas, and states in the U.S. from 2004-15. After familiarizing themselves with the tool, we loaded the participant's dataset and encouraged them to talk-aloud during data exploration, and use external resources as needed. If the participant was out of ideas, we suggested one of the main VQS functionalities[3] that they had not yet used. If this operation was not applicable to their specific dataset, they were allowed to skip the operation after having considered it. The user study lasted for about an hour and ended after they covered all the main functionalities. After the study, we asked participants open-ended questions about their experience.

[3]query by sketching, drag-and-drop, pattern loading, input equations, representative and outliers, narrow/ignore x-range options, filtering, data smoothing, creating dynamic classes, data export

4.3 Current Participant Workflows and Opportunities

In this section, we describe our study participants, their scientific goals, and their preferred analysis workflows, based on Phase I of our study. While we collaborated with each application domain in depth, we focus on the key findings from each domain to highlight their commonalities and differences, in order to provide a backdrop for our VQS findings described later on. Comparing and contrasting between the diverse set of questions, datasets, and challenges across these three use cases revealed new cross-disciplinary insights essential to better understand how VQSs can be extended for novel and unforeseen use cases.

Figure 4.3: Screenshots from contextual inquiry. Left: A1 performs data smoothing to clean the data and then examines a light curve manually using a Jupyter notebook. Right: G2 uses a domain-specific software to perform clustering and visualize the outputs.

Astronomy

Participants and Goals:

The Dark Energy Survey (DES) is a multi-institution project that surveys 300 million galaxies over 525 nights to study dark energy [46]. The telescope used to survey these galaxies also focuses on smaller patches of the sky on a weekly interval to discover astronomical transients, i.e., objects whose brightness changes dramatically as a function of time, such as supernovae or quasars. Their dataset consisted of a large collection of light curves: brightness observations over time, one associated with each astronomical object, plotted as a time series. Over five months, we worked closely with A1, an astronomer on the project's data management team at a supercomputing facility. Their scientific goal was to identify potential astronomical transients in order to study their properties, i.e., identify patterns in line charts. These insights can help further constrain physical models regarding the formation of these objects.

Existing Workflow and Design Opportunities:

Since astronomical datasets are often terabytes in scale, they are often processed and stored in highly specialized data management systems in supercomputing centers. As a preliminary step, the astronomer downloads a data sample to explore in a Jupyter notebook, performs data cleaning and wrangling, and verifies data fidelity by computing a set of relevant statistics. Then, to identify transients, the primary scientific goal of their exploration, the astronomer programmatically generates visualizations of candidate objects with `matplotlib` and visually examines each light curve. Figure 4.3 (left) shows an example of such a manually-generated light curve. If an object of interest is identified through visual analysis, the astronomer may inspect the image of the object for verifying that the significant change in brightness was not due to an imaging artifact. While experienced astronomers like A1 who have examined many transient light curves can often distinguish an interesting transient from noise by sight, manual searching for transients is still very time-consuming and error-prone, since the large majority of objects are false-positives. A1 immediately recognized the potential of VQSs, since he could use specific pattern search queries to directly identify these rare transients without cumbersome manual examination.

Genetics

Participants and Goals:

Gene expression is a common measurement in genetics obtained via microarray experiments [167]. We worked with a graduate student (G1) and professor (G3) at a research university who were using gene expression data to understand how genes are related to phenotypes expressed during early embryonic development. Their data consisted of a collection of gene expression profiles over time for mouse stem cells, aggregated over multiple experiments. Their scientific goal was to correlate gene function with their expression profiles (i.e., line charts) by gaining a high-level overview of the expression profile patterns.

Existing Workflow and Design Opportunities:

G1 often downloads the raw microarray data from a public database and preprocesses the data using a script written in R. Then, to explore this data, G1 loads the preprocessed gene expression data into a custom desktop application to visualize and cluster the gene expression profiles, as shown in Figure 4.3 (right). Prior to the study, G1 and G3 spent over a month searching for the "right" number of groups to cluster the profiles, by iteratively tuning the parameters on the clustering application and evaluating the output via a mix of application-provided visualizations and programmatically-generated statistics. While regenerating their results took no more than 15 minutes every time they made a change, the multi-step, segmented workflow meant that all changes had to be done offline, this is, they could only test out a few variations per week. When we first demonstrated the capabilities of a VQS in our introductory meeting, G3 was astonished to see that on performing an interaction, the recommended visualizations updated almost instantaneously, as opposed to waiting until the

next meeting for G1 to re-generate the results. They expressed an interest in VQSs, since the tool had the potential to dramatically speed up their collaborative analysis process.

Material Science

Participants and Goals:

We collaborated with material scientists at a research university who identify solvents for energy-efficient and safe batteries. These scientists worked on a large simulation dataset containing chemical properties for more than 280,000 solvents [108]. Each row of their dataset corresponded to a unique solvent with 25 different chemical attributes. We worked closely with a postdoctoral researcher (M1), professor (M2), and graduate student (M3) to design a sensible way of exploring their data. They wanted to use VQSs to discover solvents that not only have similar properties to known solvents, but are also more favorable (e.g., cheaper or safer to manufacture). To search for these solvents, they needed to understand how changes in certain chemical properties affect others (expressed as trends in line charts) under specific conditions.

Existing Workflow and Design Opportunities:

M1 typically starts his data exploration process by applying filters to a list of potential battery solvents using SQL queries (e.g., find solvents with boiling point over 300 Kelvins and lithium solvation energy under 10 kcal/mol). By iteratively applying and adjusting different (often complementary) sets of filters, he compares between different groups of solvents by observing their properties across a small sample. He manually examines the properties of each individual solvent by inspecting the 3D chemical structure of the solvent in a custom software, as well as gathering information regarding the solvent by cross-referencing an external chemical database and existing uses of this solvent in literature. The collected information, including cost, availability, and other physical properties, enabled researchers to select the final set of desirable solvents that could be feasibly experimented with in their lab. While M1 could identify potential solvents through manual lookups and comparisons, M2 and M1 saw the value in VQSs since it was often impossible to manually uncover hidden relationships between different attributes, such as how changes in one property affects the behavior of others for a class of solvents, across large numbers of solvents.

Themes Emerging From Need-finding Phase

Across the domains, several themes emerged around the bottlenecks that participants experienced in existing workflows.

- **Need for Expressive Querying:** While there is often a need to compare among large numbers of data instances, it is difficult to express and search for a desired shape-based pattern through programming languages like SQL or Python. And yet, none of the participants have heard of VQSs, let alone use them.

- **Need for Integrative Workflows:** Users often switched between different analytical tasks, including preprocessing, parameter specification, code execution, and visualization comparisons. The non-interactive nature of their segmented workflows impedes exploratory analysis and hinders collaboration.
- **Need for Faceted Exploration:** To deal with the large volume of data present, users have to select particular samples or subsets of data that are "worth investigating". Often, the choice of what criteria to apply as filters is also exploratory.

These themes seeded the collaborative feature discovery process, leading to the development of the system prototype, described next.

4.4 Design Process and System Overview

Given the need for a VQS, we further collaborated with participants to develop features to address their problems and challenges in Phase II of our study. We first provide a high-level system overview of the design product, ZENVISAGE++, then we reflect on our feature discovery process.

System Overview

The ZENVISAGE++ interface is organized into five major regions all of which dynamically update upon user interactions. Typically, participants begin their analysis by selecting the dataset and attributes to visualize in the *data selection panel* (Figure 4.4A). Then, they specify a pattern of interest as a query (hereafter referred to as *pattern query*), through either sketching, inputting an equation, uploading a data pattern, or dragging and dropping an existing visualization, displayed on the *query canvas* (Figure 4.4B). ZENVISAGE++ performs shape-matching between the queried pattern and other possible visualizations, and returns a ranked list of visualizations that are most similar to the queried pattern, displayed in the *results panel* (Figure 4.4C). At any point during the analysis, analysts can adjust various system-level settings through the *control panel* (Figure 4.4D) or browse through the list of *recommendations* provided by ZENVISAGE++ (Figure 4.4E). For comparison, the existing ZENVISAGE system (Figure 4.5) from Siddiqui et al. (2017) [200] allowed users to query via sketching or drag-and-drop and displayed representative and outlier pattern recommendations, but had limited capabilities to navigate across different data subsets and had few control settings. Our ZENVISAGE++ system is open source and available at: `http://github.com/zenvisage/zenvisage`; other details and documentation can be found at that link.

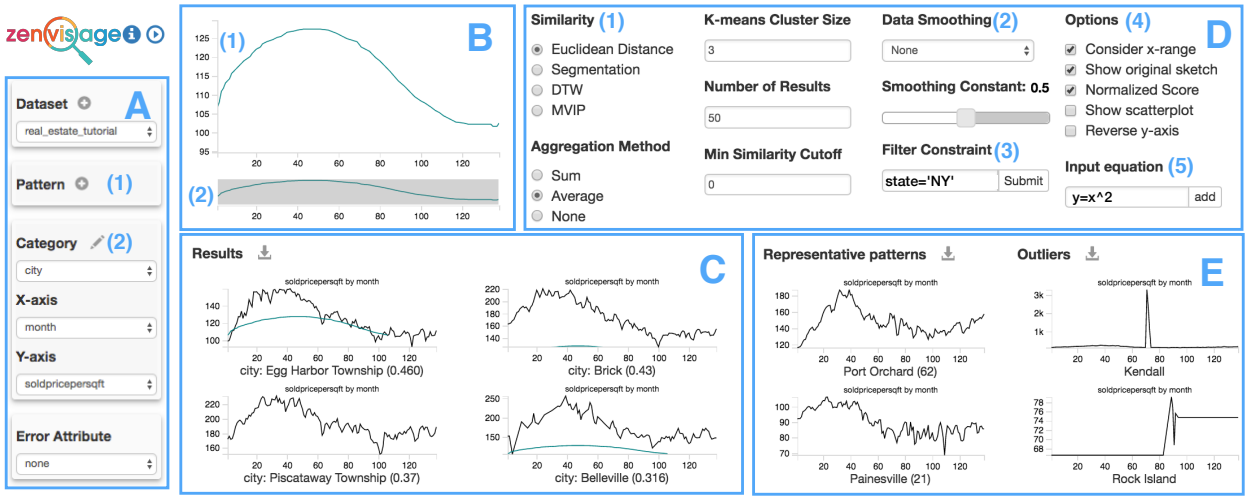


Figure 4.4: The ZENVISAGE++ system consists of : (A) data selection panel (where users can select visualized dataset and attributes), (B) query canvas (where the queried data pattern is submitted and displayed), (C) results panel (where the visualizations most similar to the queried pattern are displayed as a ranked list), (D) control panel (where users can adjust various system-level settings), and (E) recommendations (where the typical and outlying trends in the dataset is displayed).

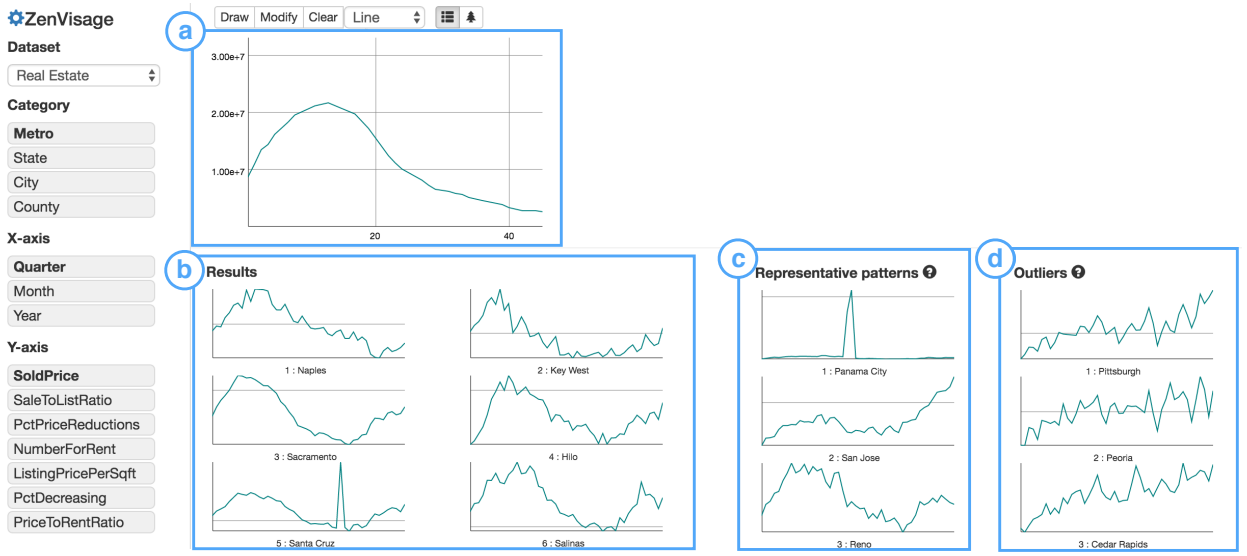


Figure 4.5: The existing ZENVISAGE prototype allowed users to sketch a pattern in (a), which would then return (b) results that had the closest Euclidean distance from the sketched pattern. The system also displays (c) representative patterns obtained through K-Means clustering and (d) outlier patterns to help the users gain an overview of the dataset.

The Collaborative Feature Discovery Process

Throughout the design process, we worked closely with participants to discover VQS capabilities that were essential for addressing their high-level domain challenges. We identified various subtasks based on the participant's workflows, designed sensible features for accomplishing these subtasks that could be used in conjunction with existing VQS capabilities, and elicited feedback on intermediate feature prototypes. Bodker et al. (1993) [23] cite the importance of encouraging user participation and creativity in cooperative design through different techniques, such as future workshops, critiques, and situational role-playing. Similarly, our objective was to collect as many feature proposals as possible, while being inclusive across different domains. We further organized these features we added to ZENVISAGE++ into Table 4.2 through an iterative coding process by one of the authors.

In grounded theory methods [150], researchers first create *open codes* to assign descriptive labels to raw data, followed by grouping open codes together by relationships or categories to form *axial codes*. Finally, *selective codes* are obtained by focusing on specific sets of axial codes to come up with a set of core emerging concepts. Inspired by grounded theory methods, we first collected the list of features, example usage scenarios, and similar capabilities in existing VQSs as open codes, corresponding to individual rows in Table 4.2. Then, we further organized this list into axial codes representing "components": core functionalities essential to VQSs (second column in Table 4.2). Finally, as we will describe in Chapter 4.5, the selective codes capture each of the sensemaking processes (leftmost column in Table 4.2). Instead of describing this table in detail, we present a typical example of how this table is organized. From right to left, consider the row corresponding to the Smoothing feature (column 3) in Table 4.2: one of the common challenges in astronomy and material science is that noise in the dataset can result in large numbers of false-positive matches. To address this issue, smoothing is a feature in ZENVISAGE++ that enables users to adjust data smoothing algorithms and parameters on-the-fly to both denoise the data and change the degree of shape approximation applied when performing pattern matching. Smoothing, along with range selection and range invariance (row 5 and 6), is part of the *match specification* component: VQS mechanisms for clarifying how matching should be performed. Both match specification and *pattern specification* (a description of what the pattern query should look like) are essential components for supporting the sensemaking process top-down pattern search (in blue, as labeled in the leftmost column).

It is important to note that while some of the proposed features in Table 4.2 (such as data filtering and view specification) are pervasive in general visual analytics (VA) systems [79, 6], they have not been incorporated in present-day VQSs. In fact, one of the key insights here is in recognizing the need for an *integrative* VQS whose sum is greater than its parts, that encourages analysts to rapidly generate hypotheses and discover insights by facilitating all three sensemaking processes. This finding is partially enabled by the unexpected benefits that come with collaborating with multiple groups of participants during the feature discovery process. Next, we reflect on what worked and what didn't work in the feature discovery process, to inform similar design studies for visual analytics systems.

Cross-pollination and Generalization via Parallel Use Cases.

Introducing the newly-added features to ZENVISAGE++ that addressed a particular domain often resulted in unexpected use cases for other domains. Considering feature proposals from multiple domains can also result in cross-pollination of feature designs, often leading to more generalized design choices. For example, around the same time when we spoke to astronomers who wanted to eliminate sparse time series from their search results, our material science collaborators also expressed a need for inspecting only solvents with properties above a certain threshold. Instead of developing separate domain-specific features, data filtering arose as a crucial, common operation that was later incorporated into ZENVISAGE++ to support this class of queries.

The Hidden Upfront Cost of Domain Integration.

While we expected to spend most of our collaborative design effort on figuring out the mechanics of visual query specification and matching, instead, preparing participant datasets for use in our system by meeting data and system requirements was the most time-consuming aspect of this phase. We provide a detailed timeline in Figure 4.2. Data requirements include gaining an understanding of the problem domain, understanding the types of data suitable for a VQS, and cleaning and loading of this data. System requirements include features required for the data to be visualized appropriately. Often, participants could only envision the types of queries to issue and how variations to the system could help better address their needs after seeing their data displayed for the first time in the prototype. We also found that the time it took us to satisfy the data and system requirements decreased as we progressed to the later domains, by leveraging existing features in our prototype to satisfy some of the upfront needs.

Build Connectors, not Swiss-Army Knives.

Participants often envisioned how VQSs can be used in conjunction with other resources that they are familiar with, including those used for reference, computing statistics, browsing related datasets, or examining other data attributes or visualization types not supported in the VQS (scatterplots, histograms). The prevalence of external tools for supporting analytical inquiries stems from how analysts often require multiple data sources or data attributes to further develop or verify their hypothesis. For example, to determine whether a particular gene belongs to a regulatory network, G2 not only needed to look at the expression data in the VQS, but also enrichment testing and knockout data. Likewise, others used specialized tools for visualizing telescope images and 3D chemical structures. Instead of forcing our VQS prototype into a swiss-army knife, we instead focused on building connectors that enable smoother transitions between tools. For example, our data upload and pattern upload feature invites participants to bring data from an external tool into ZENVISAGE++, while our data export feature allowed users to download the similarity, representative trend, and outlier results as csv files from ZENVISAGE++ into an external tool. For example, geneticists could export the clusters directly from ZENVISAGE++ as inputs to their downstream regression analysis.

The Art of Problem Selection.

While our collective brainstorming led to the cross-pollination and generalization of features, this technique can also lead to unnecessary features that result in wasted engineering effort. During co-design, there were numerous features proposed by participants, not all of which were incorporated. The reasons for not carrying a feature from design to implementation stage included:

- Nice-to-haves: One of the most common reasons for unincorporated features comes from participant's requests for nice-to-have features. We use two criteria (necessity and generality across domains) to judge whether to invest in developing a particular feature.
- "One-shot" operations: We decided not to include features that only needed to be performed once and remain fixed thereafter in the analysis workflow. For example, certain preprocessing operations such as filtering null values only needed to be performed once with an external tool, whereas data smoothing is a procedure that requires some degree of tuning and adjustments.
- Substantial research or engineering effort: Some proposed features did not make sense in the context of VQS or required a completely different set of research questions. For example, the question of how to properly compute similarity between time series with non-uniform number of datapoints arose in the astronomy and genetics use case, but requires the development of a novel distance metric and algorithm that is out of the scope of our design study objective.
- Underdeveloped ideas: Other feature requirements came from casual specification that was underspecified. For example, A1 wanted to look for objects that have a deficiency in one band and high emission in another band, but the scientific definition of "deficiency" in terms of brightness levels was ambiguous.

The decision of whether to invest in developing a feature requires a careful balance between promoting unforseen feature and wasted engineering efforts. Failure to identify these early signs may result in feature implementations that turn out not to be useful for the participants or result in feature bloat.

4.5 A Sensemaking Model for VQSs

To convey how features in ZENVISAGE++ address the analytical needs posed by each domain, we organize our PD findings into a sensemaking framework for VQSs. We now revisit Table 4.2 in an effort to contextualize our design findings using Pirolli and Card's sensemaking framework [171].

Pirolli and Card's sensemaking model for expert intelligence analysis distinguishes between information processing tasks that are *top-down* (from theory to data) and *bottom-up* (from data to theory). Correspondingly, in the context of VQSs, analysts can query either directly based on a pattern "in their head" [189] via *top-down pattern specification* or based on the data or visualizations presented to them by the system via *bottom-up data-driven inquiry*. In addition, when analysts do not know what attributes to visualize, *context creation*

helps analysts navigate across different collections of visualizations to seek visualization attributes of interest. In this section, we first describe the objectives of each sensemaking process, then we discuss how each sensemaking process is comprised of functional components that address the problem and dataset characteristics of each domain. For reference, the mapping between specific ZENVISAGE++ features and these components and processes can be found in the left two columns of Table 4.2.

Top-Down Pattern Search

Top-down processes are "*goal-oriented*" tasks that make use of "*analysis or re-evaluation of theories [and] hypotheses [to] generate new searches*" [171]. Applying this notion to the context of VQSs, the goal of top-down pattern search is to search for data instances that exhibit a specified pattern, based on analyst's intuition about how the desired patterns should look like "in theory" (including visualizations from past experience or abstract conceptions based on external knowledge). Based on this preconceived notion of what patterns to search for, the design challenge is to translate the pattern query from the analyst's head to a query executable by the VQS. This requires both components for specifying the pattern (*pattern specification*), as well as controls governing how the pattern-matching is performed (*match specification*).

Pattern Specification interfaces allow users to submit exact descriptions of a pattern query. This is useful when the dataset contains *large numbers of potentially-relevant pattern instances*. Since it is often difficult to sketch precisely, additional shape characteristics of the pattern query (e.g., patterns containing a peak with a known amplitude, or expressible as a functional form) can be used to further winnow the list of undesired matches.

Match Specification addresses the well-known problem in VQSs where pattern queries are imprecise [36, 84, 49] by enabling users to clarify how pattern matching should be performed. Match specification is useful when the dataset is *noisy*. When the pattern query satisfies some additional constraints (e.g., the pattern is horizontally invariant), adjusting these knobs helps prune away matches that are false-positives to help analysts discover true desired candidates.

Usage Scenario: A1 knows intuitively what a supernovae pattern should look like and its detailed shape characteristics, such as the amplitude of the peak and the level of error tolerance for defining a match. He first performs top-down pattern search by querying for transient patterns through sketching, then adjusts the match criterion by choosing to ignore differences along the temporal dimension and changing the similarity metric for flexible matching.

	Component	Feature	Purpose	Task Example	Similar Features in Past VQSs
Top-Down	**Pattern Specification:** *What is the shape of the pattern query?*	Query by Sketch (Figure 4.4B1)	Freehand sketching for specifying pattern query.	**A:** Find patterns with a peak and long-tail decay that may be supernovae candidates.	All include sketch canvas except [80].
		Input Equation (Figure 4.4A1)	Specify a exact functional form as a pattern query (e.g., y=x^2).	**M:** Find patterns exhibiting inversely proportional chemical relationship.	—-
		Pattern Upload (Figure 4.4D2)	Upload a pattern consisting of a sequence of points as a query.	**A:** Find supernovae based on previously discovered sources.	Upload CSV [149]
		Smoothing (Figure 4.4D2)	Interactively adjusting the level of denoising on visualizations, effectively changing the degree of shape approximation when performing pattern matching.	**A, M:** Eliminate patterns matched to spurious noise.	Smoothing [142] Angular slope queries [80] Trend querylines [180]
	Match Specification: *How should the pattern query be matched with other visualizations?*	Range Selection (Figure 4.4B2, D4)	Restrict to query only in specific x/y ranges of interest through brushing selected x-range and filtering selected y-range.	**A:** Matching only around shape exhibiting a peak. **M:** Matching only around shape region that exhibit linear or exponential relationships	Text Entry [226, 142] Min/max boundaries [180] Range Brushing [81]
		Range Invariance (Figure 4.4D1,4)	Ignoring vertical or horizontal differences in pattern matching through option for x-range normalization and y-invariant similarity metrics .	**A:** Searching for existence of a peak above a certain amplitude. **G:** Searching for a "generally-rising" pattern.	Temporal invariants [36]
Context Creation	**View Specification:** *What data to visualize and how should it be displayed?*	Data selection (Figure 4.4A)	Changing the collection of visualizations to iterate over.	**M:** Explore tradeoffs and relationships between physical attributes.	—-
		Display control (Figure 4.4D4)	Changing the details of how visualizations should be displayed.	**M:** Non-time-series data should be displayed as scatterplot.	—-
	Slice-and-Dice: *How does navigating to another data subset change the query result?*	Filter (Figure 4.4D3)	Display and query only on data that satisfies the composed filter constraints.	**A:** Eliminate unlikely candidates by navigating to more probable data regions. **M, G:** Compare how overall patterns change when filtered to particular data subsets.	—-
		Dynamic Class (Figure 4.6)	Create custom classes of data that satisfies one or more specified range constraints. Display aggregate visualizations for separate data classes.	**A, M:** Examine aggregate patterns of different data classes.	—-
Bottom-Up	**Result Querying:** *What other visualizations "look similar" to the selected pattern?*	Drag-and-drop (Figure 4.4C, E)	Querying with any selected result visualization as pattern query (either from recommendations or results).	**A, G, M:** Find other objects that are similar to X; Examine what other objects similar to X look like overall.	Drag-and-drop [81] Double-Click [36]
	Recommendation: *What are the key patterns in this dataset?*	Representative and Outliers (Figure 4.4E)	Displaying visualizations of representative trends and outlier instances based on clustering.	**A:** Examine anomalies and debug data errors through outliers. **G, M:** Understand representative trends common to this dataset (or filtered subset).	—-

Table 4.2: Taxonomy of key capabilities essential to VQSs and major features incorporated via user-centered design. We organize each feature based on its functional component. From left to right, each of the three sensemaking processes (first column) is broken down into key functional components (second column) in VQSs. Each component addresses a pro-forma question from the system's perspective. Table cells are further colored according to the sensemaking process that each component corresponds to (Blue: Top-down, Yellow: Context creation, Green: Bottom-up). We list the functional purpose of each feature as implemented in ZENVISAGE++, example use cases from participatory design (**A:** astronomy, **M:** material science, **G:** genetics), and similar features incorporated in past VQSs.

Bottom-Up Data-Driven Inquiry

In Pirolli and Card's sensemaking model, bottom-up processes are "*data-driven*" tasks initiated by "*noticing something of interest in data*" [171]. Likewise in VQSs, bottom-up data-driven inquiry is a browsing-oriented sensemaking process that involves tasks that are inspired by system-generated visualizations or results. The design challenge for VQSs to support bottom-up inquiries is to develop the right set of "stimuli" through *recommendations* that could provoke further data-driven inquiries, as well as low-effort mechanisms to search via these pattern instances through *result querying*. As we will discuss later, this process is crucial but underexplored in past work on VQSs.

Recommendations display visualizations that may be of interest to users based on the current data context. In ZENVISAGE++, recommendations comprise of representative trends and outliers, which are useful for understanding common and outlying behaviors when a *small number of common patterns* is exhibited in the dataset.

Result querying enables users to query for patterns similar to a selected data pattern from the ranked list of results or recommendations. Typically, analysts select visualizations with *semantic or visual properties* of interest and make use of result querying to understand characteristic properties of similar instances.

Usage Scenario: G2 does not have an upfront knowledge of what to search for. She learns about the characteristic patterns that exist in the dataset through the representative trends, a form of bottom-up inquiry, as a means to jump-start further queries via result querying, as well as understand groups of data instances with shared characteristics.

Context Creation

While top-down and bottom-up processes operate on a collection of visualizations with fixed X and Y attributes, context creation operates in the regime where the analyst may be investigating the relationships between multiple different attributes or values of interest. Context creation enables analysts to navigate across different visualization collections to learn about patterns in different regions of the data. The design challenge of context creation is to help users visualize and compare how data changes between these different contexts by constructing visualization collections with different visual encodings (*view specification*) or different data subsets (*slice-and-dice*).

View specification settings alter the encoding for all of the visualizations on the VQS currently being examined. This ability to work with different collections of visualizations is useful when the dataset is *multidimensional* and the axes of interest are *unknown*. Modifying the view specification offers analysts different perspectives on the data to locate visualization collections of interest.

Slice-and-Dice empowers users to navigate and compare collections of visualizations constructed from different subsets of the data. Data navigation capabilities are essential when the dataset has *large numbers of "support attributes"* that may be related to the visualization attributes (e.g., geographical location may influence the time series pattern for housing prices). Analysts can either make use of pre-existing knowledge regarding these support attributes to navigate to a data region that is more likely to contain the desired pattern (e.g., filtering to suburbs to find cheaper housing) or discover unknown patterns and relationships between different data subsets (e.g., housing prices are lower in winter than compared to summer).

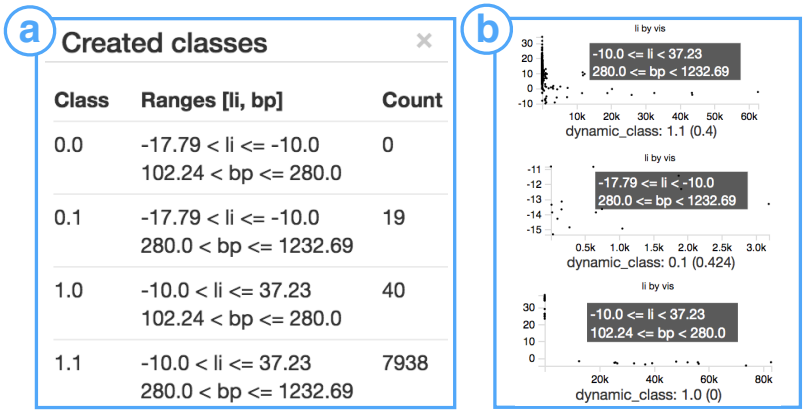


Figure 4.6: Example of dynamic classes. (a) Four different classes with different Lithium solvation energies (li) and boiling point (bp) attributes based on user-defined data ranges. (b) Users can hover over the visualizations for each dynamic class to see the corresponding attribute ranges for each class. The visualizations of dynamic classes are aggregate across all the visualizations that lie in that class based on the user-selected aggregation method.

Usage Scenario: M1 recognizes salient trends in his dataset such as inverse or linear correlations, but does not have fixed attributes that he wants to visualize or a pattern in mind to query with. Given a list of physical properties of potential interest, he performs context creation by switching between different visualized attributes to understand the dataset from alternative perspectives. He can also dynamically create different classes of data (e.g., solvents with low solubility or have high capacity) to examine their aggregate patterns, as shown in Figure 4.6.

The three aforementioned sensemaking processes are akin to the well-studied sensemaking paradigms of search (top-down), browse (bottom-up), and faceted navigation (context creation) on the Web [76, 160]. Due to each of their advantages and limitations given different information seeking tasks, search interfaces have been designed to support all three complementary acts and transition smoothly between them to combine the strength of all three sensemaking processes. Similarly for VQSs, our design objective is to enable all three sensemaking processes in ZENVISAGE++. Our Chapter 4.6 evaluation study reveals that

this integrative approach not only accelerates the process of visualization discovery, but also encourages hypotheses generation and experimentation.

Existing VQSs	Pattern Specification	Match Specification	View Specification	Slice-and-Dice	Result Querying	Recommendation	Open-Source
	Process & Component: Top-Down	Top-Down	Context Creation	Bottom-Up	Bottom-Up	Bottom-Up	
TimeSearcher		✓	✓		✓		
QuerySketch	✓	✓					
QueryLines	✓	✓					
SoftSelect	✓	✓					
Google Correlate	✓	✓					
TimeSketch	✓	✓					
SketchQuery	✓	✓			✓		✓
Zenvisage	✓	✓			✓	✓	✓
Qetch	✓	✓					✓
Zenvisage ++	✓	✓	✓	✓	✓	✓	✓

Table 4.3: Table summarizing whether key functional components (columns) are covered by past systems (row, ordered by recency). Heavily-used features for context-creation and bottom-up inquiry are largely missing from prior VQSs.

4.6 Evaluation Study Findings

Based on audio, video screen capture, and click-stream logs recorded during our Phase III evaluation study, we performed thematic analysis via open coding to label every event with a descriptive code. Event codes included specific feature usage, insights, provoked actions, confusion, need for capabilities unaddressed by the system, and use of external tools. To characterize the usefulness of each feature, we further labeled whether each feature was useful to a particular participant's analysis. A feature was deemed *useful* if it was either used in a sensible and meaningful way to accomplish a task or address a question during the study, or has envisioned usage outside of the constrained time limit during the study (e.g., if data was available or downstream analysis was conducted). In this section, we will apply our thematic analysis results to understand how each sensemaking process occurs in practice.

Uncovering the Myth of Sketch-to-Insight

To understand the usefulness of different visual querying modalities, we analyzed their frequency of use in our evaluation study. To our surprise, despite the prevalence of sketch-to-query systems in the literature, we found that only two out of our nine participants found it useful to directly sketch a desired pattern onto the canvas. The reason why most participants did not find direct sketching useful was that they often do not start their analysis with a specific pattern in mind. Instead, their intuition about what to query is derived from other visualizations they encountered during exploration, in which case it makes more sense to query using those visualizations as examples directly (e.g., by dragging and dropping that visualization onto the canvas to submit the query). Even if a user has a pattern in mind, translating that pattern into a sketch is often hard to do. For example, A2 wanted to search for a highly-varying signal enveloped by a sinusoidal pattern indicating planetary rotation  , which was hard to draw by hand.

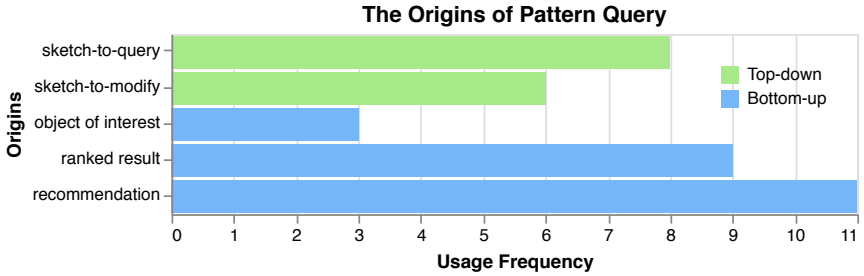


Figure 4.7: The number of times a pattern query originates from one of the workflows. Pattern queries are far more commonly generated via bottom-up than top-down processes.

We further investigated the processes that participants engaged in to construct pattern queries. Pattern queries can be generated by either top-down (sketching based on user's in-the-head pattern) or bottom-up (drag-and-drop based on what user observes from data) processes. While our study is not intended as a quantitative study with different querying modalities as conditions, we wanted to get an estimate of the relative frequency of different mechanisms across users. We examined the sequence of interactions that led to each pattern query and labeled each one based on one of the five ways it can be generated—two top-down and three bottom-up ways[4] We find that *bottom-up processes are 40% more commonly used than top-down processes for generating a pattern query.* Within top-down processes, a pattern query could arise from users directly sketching a new pattern or by modifying an existing sketch. For example, M2 first sketched a pattern to find solvent classes with anticorrelated properties (pattern as a straight line with negative slope) without much success in finding a desired match. So he instead dragged and dropped one of the peripheral visualizations similar to his desired one and then smoothed out the noise in the visualization via sketching,

[4]Top-down: sketch-to-query, sketch-to-modify; Bottom-up: Result querying via object of interest, via ranked result, or via recommendations. See Figure 4.7 for more details.

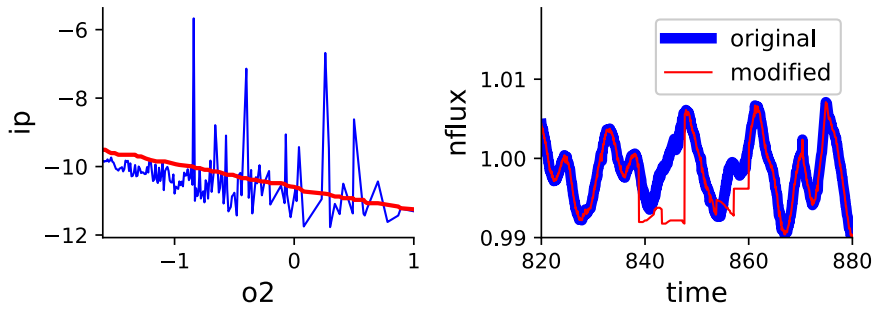


Figure 4.8: Example of sketch-to-modify, based on canvas traces from M2 (left) and A3 (right). The original drag-and-dropped query is shown in blue and sketch-modified queries in red.

yielding a straight line, as shown in Figure 4.8 (left). M2 repeated this workflow twice in separate occurrences during the study and was able to derive insights. Likewise, A3 was searching for pulsating stars characterized by dramatic changes in the amplitudes of the light curves. She knows that stellar hotspots also exhibit dramatic amplitude fluctuations, but unlike pulsating stars, the variations happen at regular intervals. Figure 4.8 (right) illustrates how A3 first picked out a regular pattern (suspected starspot), then modified it slightly so that the pattern looks more "irregular" (to find pulsating stars).

The infrequent use of top-down pattern specification was also reflected in the fact that none of the participants queried using an equation. In both astronomy and genetics, the visualization patterns resulted from complex physical processes that could not be written down as equations analytically. Even in the case of material science when analytical relationships do exist, it is challenging to formulate patterns as functional forms in a prescriptive manner.

We found that some users employed match specification to remedy undesired results from their top-down pattern queries. While we did not rigorously study the effects of different analytical parameter settings, we observed that more users refined their matches by adjusting the range and degree of approximation, rather than opting for a different similarity metric. This points to future work in developing more flexible and intuitive vocabularies for modifying the match along the research directions pursued in [36, 141] over incorporating additional complex, off-the-shelf matching objectives in VQSs.

Our findings suggest that while sketching is a useful construct for people to express their queries, the existing ad-hoc, sketch-only model for VQSs is insufficient on its own without data examples that can help analysts jumpstart their exploration. In fact, from Figure 4.7, we can see that sketch-to-query only accounted for about a fifth of the total number of visual queries performed during the study. This finding has profound implications on the design of future VQSs, since our comparison of VQS features across existing work (Table 4.3) suggests that past work has primarily focused on top-down process components, without considering how useful these features are in real-world analytic tasks. We suspect that these limitations may be why existing VQSs are not commonly adopted in practice. Note

that we are not advocating for removing the natural and intuitive sketch capabilities from future VQSs completely, but instead focusing future research and design efforts to examine other (often underexplored) VQS sensemaking processes. Such processes could be applied in conjunction with sketching to help analysts more flexibly express their analytical goals, described next.

Insights via Context Creation and Bottom-up Approaches

As alluded to earlier, bottom-up data-driven inquiries and context creation are far more commonly used than top-down pattern search when users have no desired patterns in mind, which is typically the case for exploratory data analysis. In particular, top-down approaches were only useful for 29% of the use cases, whereas they were useful for 70% of the use cases for bottom-up approaches and 67% for context creation. We now highlight some exemplary workflows demonstrating the efficacy of the latter two sensemaking processes.

Bottom-up pattern queries can come from either the ranked list of results, recommendations, or by selecting a particular object of interest as a drag-and-drop query. The most common use of bottom-up querying is via recommended visualizations. For example, G2 and G3 identified that the three representative patterns recommended in ZENVISAGE++ corresponded to the same three groups of genes discussed in a recent publication [61]: induced genes (profiles with expression levels going up ), repressed genes (starting high then decreasing ), and transients (rising first then dropping at another time point ). The clusters provoked G2 to generate a hypothesis regarding the properties of transients: *"Is that because all the transient groups get clustered together, or can I get sharp patterns that rise and ebb at different time points?"* To verify this hypothesis, G2 increased the parameter controlling the number of clusters and noticed that the clusters no longer exhibited the clean, intuitive patterns he had seen earlier. G3 expressed a similar sentiment and proceeded by inspecting the visualizations in the cluster via drag-and-drop. He found a group of genes that all transitioned at the same timestep, while others transitioned at different timesteps. By browsing through the ranked list of results, participants were also able to gain a peripheral overview of the data and spot anomalies during exploration. For example, A1 spotted time series that were too faint to look like stars after applying the filter CLASS_STAR=1, which led him to discover that all stars have been mislabeled with CLASS_STAR=0 as 1 during data cleaning.

Context creation in VQSs enables users to change the "lens" by which they look through the data when performing visual querying, thereby creating more opportunities to explore the data from different perspectives. Echoing the sentiment from past studies in visual analytics regarding the importance of designing features that enable users to select relevant subsets of data [196, 6, 79, 122], we found that all participants found at least one of the features in context creation to be useful.

Both A1 and A2 expressed that context creation through interactive filtering was a powerful way to dynamically test conditions and tune values that they would not have otherwise experimented with, effectively lowering the barrier between the iterative hypothesize-then-

compare cycle during sensemaking. During the study, participants used filtering to address questions such as: *Are there more genes similar to a known activator when we subselect only the differentially expressed genes?* (G2) and *Can I find more supernovae candidates if I query only on objects that are bright and classified as a star?* (A1). Three participants had also used filtering as a way to query with known individual objects of interest. For example, G2 set the filter as gene=9687 and explained that since "*this gene is regulated by the estrogen receptor, when we search for other genes that resemble this gene, we can find other genes that are potentially affected by the same factors.*"

While filtering enabled users to narrow down to a selected data subset, dynamic classes (buckets of data points that satisfies one or more range constraints) enabled users to compare relationships between multiple attributes and subgroups of data. For example, M2 divided solvents in the database into eight different categories based on voltage properties, state of matter, and viscosity levels, by dynamically setting the cutoff values on the quantitative variables to create these classes. By exploring these custom classes, M2 discovered that the relationship between viscosity and lithium solvation energy is independent of whether a solvent belongs to the class of high voltage or low voltage solvents. He cited that dynamic class creation was central to learning about this previously-unknown attribute properties:

> *This is really possible because of dynamic class creation, so this allows you to bucket your intuition and put that together. [...] I can now bucket things as high voltage stable, liquid stable, viscous, or not viscous and start doing this classification quickly and start to explore trends. Look how quickly we can do it!*

Combining Sensemaking Processes in VQS Workflows

Given our observations as to how participants make use of each sensemaking process in practice, we construct a Markov model to further investigate the interplay between these sensemaking processes in the context of an analysis workflow. Markov models have been used in the past by Reda et al. (2016) [175] in a similar manner to analyze interaction sequences from open-ended, exploratory analysis evaluation studies. The goal of such analysis is to quantitatively capture how users "*transitions between mental, interaction, and computational states*" to afford researchers to qualitatively characterize the processes and behavioral patterns "*essential to insight acquisition*" [175].

To compute the state transition probabilities in the Markov model, we make use of event sequences from the evaluation study, where each event consists of labels describing when specific features were used. Using the taxonomy in Table 4.2, we map each usage of a feature in ZENVISAGE++ to one of the three sensemaking processes. Each participant's event sequence is divided into sessions, each indicating a separate line of inquiry during the analysis. Based on these event sequences—one for each session, we compute the aggregate state transition probabilities (edge weight labels in Figure 4.10) to characterize how participants from each domain move between different sensemaking processes.

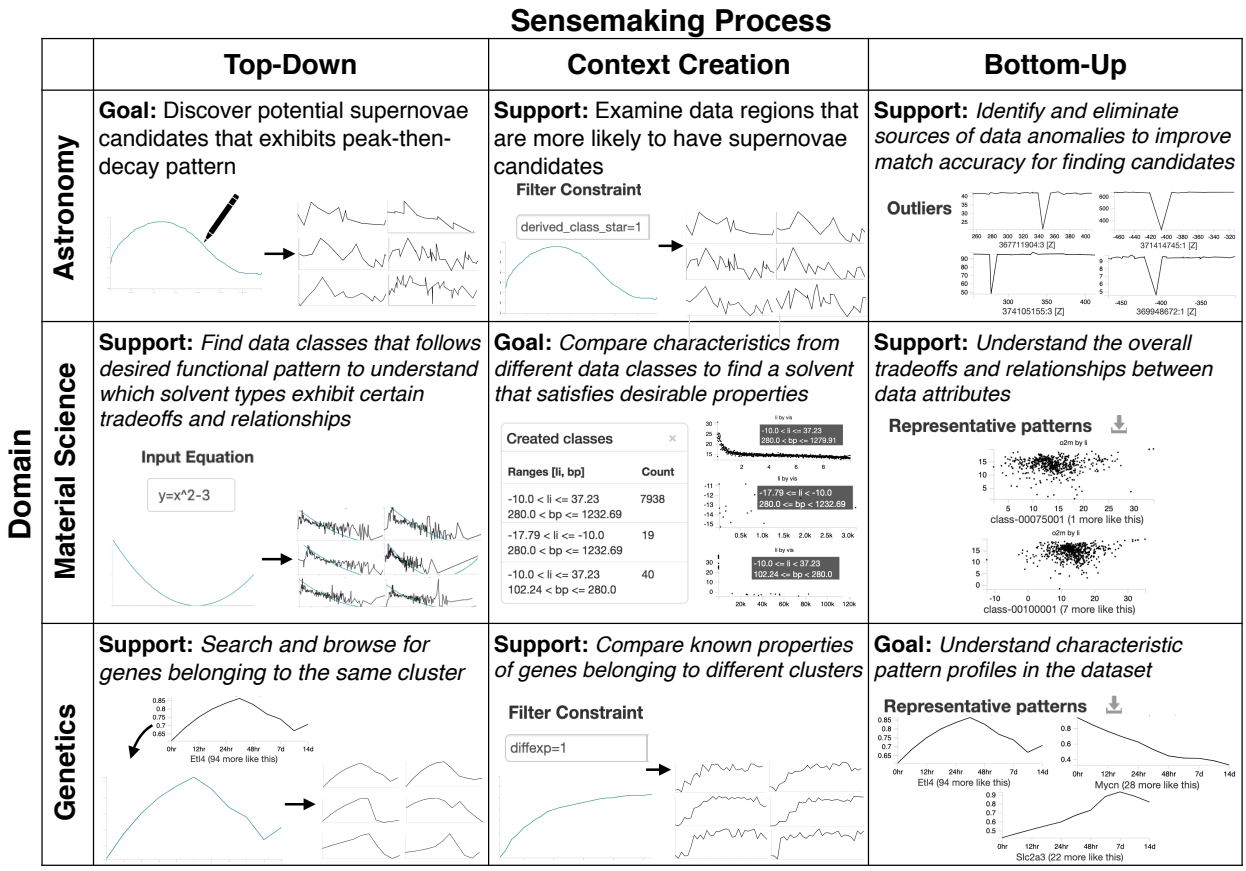


Figure 4.9: Table of example usage scenarios from each domain for each sensemaking process. Participants typically have one focused goal expressible through a single sensemaking process, but since their desired insights may not always be achievable with a single class of operation, they make use of the two other sensemaking processes to support (**Support**) them in accomplishing their main goal (**Goal**).

The transition probability represents the probability that an action from one class would be followed by one from the other. For example, in material science, 60% of events that started with bottom-up exploration lead to context creation and to top-down pattern search the rest of the time. Self-directed edges indicate the probability that the participant would continue with the same type of sensemaking process. For example, when an astronomer performs top-down pattern search, 64% of the transitions were followed by another top-down process and by context creation the rest of the time, but never followed by a bottom-up process. This high self-directed transition probability reflects how astronomers often need to iteratively refine their top-down query through pattern or match specification when looking for a specific pattern.

To study how important each sensemaking process is for participant's overall analysis, we compute the eigenvector centrality of each graph, displayed as node labels in Figure 4.10.

These values represent the percentage of time the participants spend in each of the sensemaking processes when the transition model has evolved to a steady state [169]. Given that nodes in Figure 4.10 are scaled by this value, in all domains, we observe that there is always a prominent node connected to two less prominent ones—but it is also clear that all three nodes are essential to all domains. Our observation demonstrates how participants often construct a central workflow around a main sensemaking process based on their analytical goals and interleave variations with the two other support processes as they iterate on the analytic task, as listed in Figure 4.9. For example, the material scientists focus on context creation 56% of the time, mainly through dynamic class creation, followed by bottom-up inquiries (such as drag-and-drop) and top-down pattern searches (such as sketch modification). The central process adopted by each domain is tightly coupled with the problem characteristics associated with each domain. For example, without an initial query in mind, geneticists relied heavily on bottom-up querying through recommendations to jumpstart their queries.

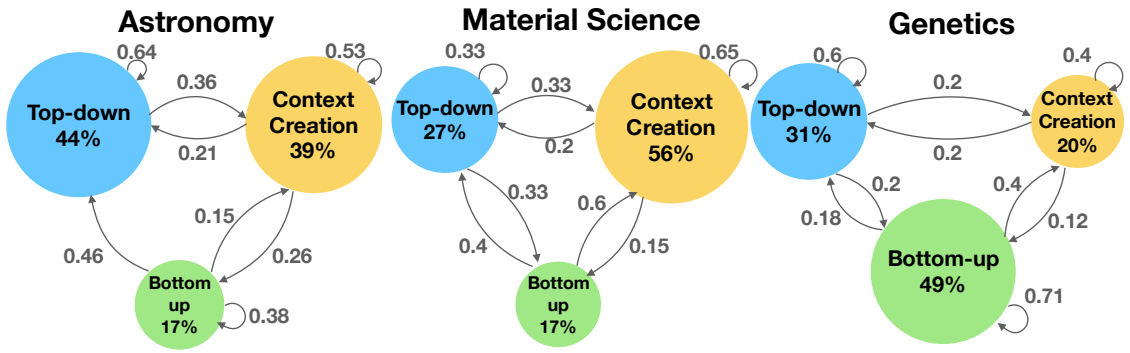


Figure 4.10: Markov models computed based on evaluation study event sequences, with edges denoting the probability that participant in the particular domain will go from one sensemaking process to the next. Nodes are scaled according to their eigenvector centrality, representing the percentage of time participants would spend in a particular sensemaking process in steady state. The data consists of 206 event actions taken by participants during the study (80 for astronomy, 65 for genetics, and 61 for material science).

The Markov transition model exemplifies how participants adopted a diverse set of workflows based on their unique set of research questions. The bi-directional and cyclical nature of the transition graphs in Figure 4.10 highlight how the three sensemaking processes do not simply follow a linear progression towards finding a single pattern or attribute of interest. Instead, the high connectivity of the transition model illustrates how these three equally-important processes form a sensemaking loop, representing iterative acts of dynamic foraging and hypothesis generation. This finding reinforces the importance of each sensemaking process and indicates that future VQSs need to be *integrative* in supporting all three sensemaking process to enable a diverse set of potential workflows for addressing a wide range of analytical inquiries.

Limitations

Although evidence from our evaluation study points to the infrequent use of direct sketch, we have not performed controlled studies with a sketch-only system as a baseline to validate this hypothesis. While we employed quantitative comparisons in various analysis throughout this section, our goal is to gain a formative understanding of VQS usage behavior across our small sample. Future studies with larger sample sizes and more representative samples are required to generalize these findings. The goal of our study is to uncover qualitative insights that might reveal why VQSs are not widely used in practice; further validation of specific findings is out of the scope of this work. While concerns regarding study results being focused on ZENVISAGE++ must be acknowledged, we note that ZENVISAGE++ is one of the most comprehensive VQSs to-date, covering many of the features from past systems and more (as evident from Table 4.3). We believe that our integrative VQS, ZENVISAGE++, can serve as a baseline for future research in VQS to evaluate against and build upon. Given that our work covered three design studies along with one evaluation study, we were unable to cover each domain to the level of detail typically found in a dedicated design study paper. Instead, our focus was to highlight the differences and similarities among these domains relevant to the capabilities required in VQS. Future longitudinal studies may also help alleviate the novelty effects that participants may have experienced during the evaluation study. While we have generalized our findings beyond existing work by employing three different and diverse domains, our case studies have so far been focused on scientific data analysis with domain-experts, as a first step towards greater adoption of VQSs. Other potential domains that could benefit from VQSs include: financial data for business intelligence, electronic medical records for healthcare, and personal data for quantified self. These different domains may each pose different sets of challenges (such as designing for novices) unaddressed by the findings in this work, pointing to a promising direction for future work.

4.7 Conclusion

While VQSs hold tremendous promise in accelerating data exploration, they are rarely used in practice. We worked closely with analysts from three diverse domains to characterize how VQSs can address their research challenges, to collaboratively design VQS capabilities, and to evaluate how VQSs are used in practice. Participants used our final system, ZENVISAGE++, to discover desired patterns, trends, and valuable insights to address unanswered research questions. Based on these experiences, we developed a sensemaking model for how analysts make use of VQSs. Contrary to past work, we found that sketch-to-query is not as effective in practice as past work may suggest. Beyond sketching, we find that each of the three sensemaking process (top-down, context-creation, and bottom-up) fulfills a central role in participants' analysis workflows to address their high-level research objectives. We advocate that future research on VQSs should invest in understanding and supporting all three sensemaking processes to effectively "close the loop" in how analysts interact and perform

sensemaking with VQSs.

Overall, while the integrative approach to visual querying appears to be a promising approach for accelerated line chart search, ZENVISAGE++ could not feasibly support general-purpose analysis tasks beyond our three presented use cases. Our collaborative design experience also suggests that the swiss-army knife approach to developing additional capabilities for ZENVISAGE++ would lead to feature bloat. As a result, we pivoted towards designing a flexible, general-purpose visual exploration assistant that supports a range of analytical tasks that is well-integrated with common data analysis workflows. These findings motivate the projects described in the second half of this dissertation, where we study how visual exploration assistants fit into GUI-based and programmatic data analysis workflows.

Chapter 5

Assistance in GUI-based Charting Tools with Frontier

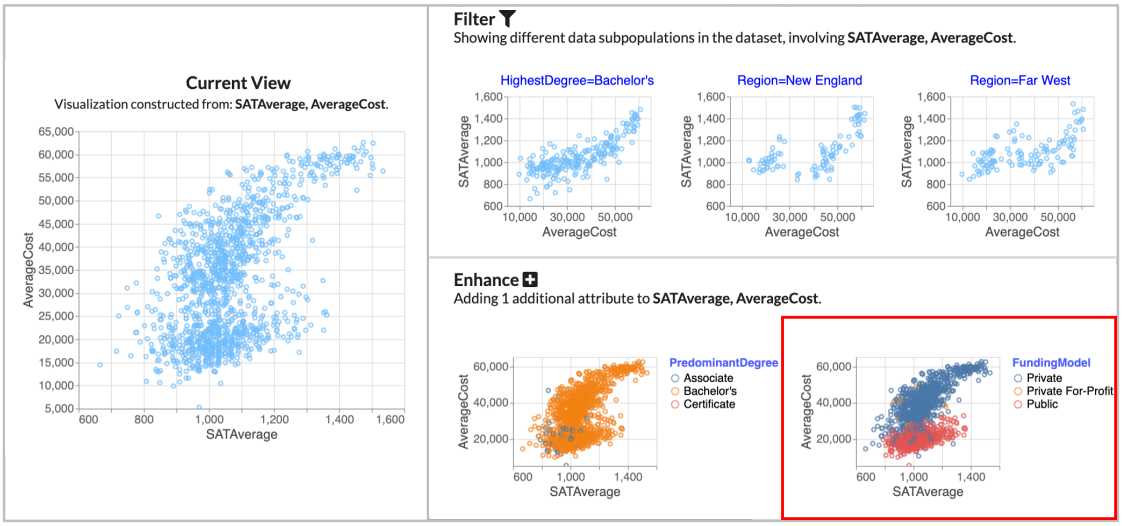


Figure 5.1: A screenshot of FRONTIER with a dataset containing college information. Starting from the Current View displaying a scatterplot of AverageCost versus SATAverage on the left, the user finds an interesting visualization recommended through the `Enhance` category highlighting the two distinct clusters for Private and Public FundingModels (shown with a red border). This recommendation is generated from the Current View, further "enhanced" by adding FundingModel to the color channel.

GUI-based charting tools, such as Tableau, provide accessible and intuitive ways for analysts to create visualizations through the ease of point-and-click interactions. Some of these charting tools make use of visualization recommendation (VisRec) to provide users with suggestions for potentially interesting and useful next steps during exploratory data analysis. These recommendations are typically organized into categories based on their analytical

actions, i.e., operations employed to transition from the current exploration state to a recommended visualization. However, the utility of the categories employed by these systems in analytical workflows has not been systematically investigated. This chapter explores the efficacy of recommendation categories in GUI-based charting tools by formalizing a taxonomy of common categories and developing a system, FRONTIER, that implements these categories. Using FRONTIER as a design probe for a general-purpose visual exploration assistant, we evaluate workflow strategies adopted by users and how categories influence those strategies.

5.1 Introduction

GUI-based charting tools provide an accessible means to ask questions about data through visualizations, where new questions often arise from unexpected observations. However, even though GUI-based charting tools makes it easy to design or create a visualization, challenges arise when the current choice of visualization or analyses does not yield interesting observations; this common pain point can cause users to feel stuck or overwhelmed, unsure of what question to ask next [65, 127]. Building on top of interactive charting tools, visualization recommendation (VisRec) systems guide users along their exploration journey by suggesting effective visual encodings [139, 242, 241] or potentially interesting visualizations [39, 87, 220, 237, 41, 191]. A review of related work on VisRec systems can be found in Chapter 2 and summarized in Table 5.1.

Recommendations are often organized into categories based on the *analytical actions* they embody. *Analytical actions* can be thought of as transitions between visualization states, corresponding to the operations performed to generate recommendations given the current visualization state. Example categories include `Filter`, displaying recommendations of sub-populations of the data, derived by adding filters to the current visualization, and `Enhance`, displaying recommendations by adding one additional attribute to the current visualization. Figure 5.1 illustrates how recommendation categories can support the analysis of a college dataset. A scatterplot of `SATAverage` and `AverageCost` can be *filtered* to specific `HighestDegrees` or `Regions` (right side, top) or *enhanced* by adding the `FundingModel` attribute to the color channel (right side, bottom). Users can explore their data via moves in the visualization space, selecting visualizations based on analytical actions that represent potential "next steps" in their analysis.

Most VisRec systems are single-purpose in the sense that they only display a small set of bespoke analytical-action-based recommendation categories. This limited selection of categories in existing systems stems from challenges in both *development* and *evaluation*. From an evaluation standpoint, determining the value of a given recommendation for a specific user goal is, in general, a challenge in recommender system design [147], but doing so for visual analysis tools is even harder. Unlike web search, where the typical goal is to find a single item (e.g. a movie to watch), this design is further complicated by the variety of user goals, ranging from specific inquiries to more complex open-ended objectives such as understanding relationships across attributes [152]. From a development standpoint, there is an

interface design and performance cost for dynamically computing large numbers of recommendations [41, 220]. As a result, most systems rely on a small set of fixed recommendation categories.

While prior work has certainly demonstrated the benefits of VisRec in supporting exploratory analysis [39, 242, 241], the design space of VisRec categories has not been thoroughly explored and evaluated. With new VisRec systems being introduced time and again in the visualization and HCI literature [120], there is a pressing need to take a step back to organize and make sense of the design space of analytical-action-based categories in VisRec and to further evaluate the benefits of multiple types of recommendation categories as a whole, and relative to each other. This evaluation is crucial for distilling design guidelines for next-generation VisRec systems and for enabling past, present, and future VisRec systems to be understood and compared in the context of an organization.

In this chapter, we deconstruct categorization in VisRec systems by comparing and evaluating the value of different recommendation categories in visual analytic workflows. We further investigate how analysis strategies are influenced by employing recommendation categories as well as the efficacy of various recommendation categories for different task and dataset characteristics. While recommendation categories such as `Filter` and `Enhance` are in fact present in prior systems [127, 122, 51, 242, 241, 87], there has been no systematic organization or comparison of recommendation categories and their underlying analytical actions. Another challenge is that no existing VisRec system comprehensively implements the space of possible categories to compare their effects on analytical workflows. This crucial gap in existing literature motivated the design of our general-purpose visual exploration assistant, FRONTIER[1]. We developed FRONTIER as an apparatus for investigating the merits and pitfalls of various recommendation categories in a single system.

Our contributions are summarized as follows:

- We present a taxonomy of common analytical action-based recommendation categories employed in visual analysis, synthesizing existing literature from VisRec and online analytical processing (OLAP). The taxonomy enables us to map out the design space of existing VisRec systems as well as future ones. (Chapter 5.2)
- We develop a design probe, FRONTIER, implementing ten recommendation categories from the taxonomy to explore the usage and impact of these categories in a visual analysis workflow. (Chapter 5.3)
- We present a mixed-methods user study to understand how recommendation categories support visual analysis and the relative efficacy of various recommendation categories. (Chapter 5.4, 5.5)

[1]The name FRONTIER is inspired by how the application of analytical actions offer next steps along potential exploration paths—enabling an explorer to expand the *frontier* of discovery. In the context of graph search algorithms, the term *frontier* refers to the nodes that lie between what has been discovered and those as yet undiscovered [34].

As part of this work, one of our goals was to take stock of and systematize research in the rapidly-evolving area of VisRec systems. Our findings validate prior conjectures about the value of categorizing recommendations [242, 41] and further reveal how there are substantial differences in the usage of various categories. For instance, participants indicated that recommendation categories `Enhance` (adding one attribute) and `Filter` (displaying data sub-populations) were most useful, while `Pivot` (swapping an attribute) was one of the least useful. Such findings point to the importance of comparative evaluation across categories and guidelines for improving category design in future visual exploration assistants.

5.2 Taxonomy of Recommendation Categories

Analytical actions correspond to transitions through the visualization space to generate categories of recommendations given a user's current visualization state. While various taxonomies [153, 27, 6] exist for describing the types of tasks (or *actions*) employed during visual analysis, as stated in Law et al. (2020) [119], we are not aware of any taxonomy that encompasses the types of data-based visualization recommendations that can be generated via analytical actions and are reflected in present-day VisRec systems. This section defines such a taxonomy, providing a common vocabulary for the organizing principles behind recommendation categories. The taxonomy arose through a systematic review of 20 VisRec systems. We first describe our approach to surveying existing VisRec system through a coding process. Then, we present the action in the taxonomy in detail.

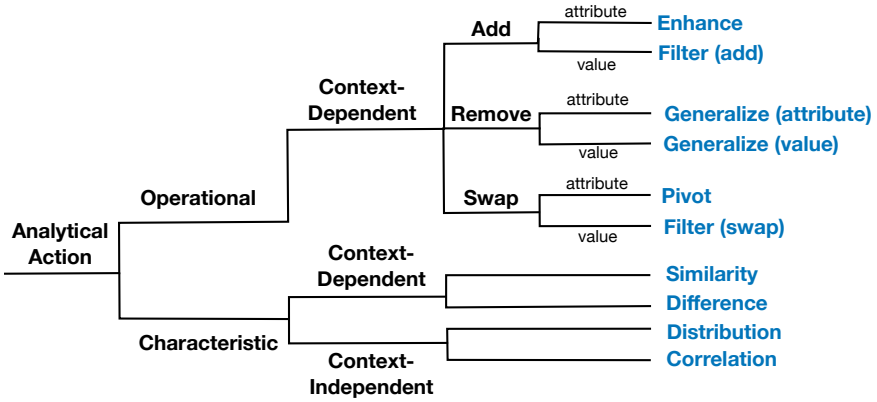


Figure 5.2: A taxonomy of common analytical actions used in recommending visualizations for visual analysis. The analytical actions are indicated in blue.

Literature Review and Coding Process

Through open and axial coding [208] by one of the authors, we listed the different types of recommendations supported across different systems and then iteratively grouped similar capabilities into categories. We surveyed the 25 research papers that introduces a system that recommends visualizations or visualization insights for the purpose of guiding users in the

Category	Related Work
Distribution	[87, 242, 241, 185, 237, 41, 191]
Correlation	[87, 241, 204, 237, 41, 40, 39]
Enhance	[87, 242, 241, 51]
Generalize (attribute)	[87, 132, 191]
Generalize (filter)	[87]
Pivot	[87, 51]
Filter (add)	[51, 8, 122, 243]
Filter (swap)	[51, 127, 142, 183]
Difference	[118, 220, 8]
Similarity	[118, 127, 142]

Table 5.1: Survey of recommendation categories from 20 VisRec systems. We included systems that recommend visualization(s) based on the result of some analytical action.

process of exploratory data analysis. We performed an extensive literature review spanning from visualization, HCI, and database venues to ensure a broad coverage of recommendation categories.

First, to better understand and group together related systems, we classified each work based on the type of recommendation system (data, encoding, or hybrid following the definition in Wongsuphasawat et al. [239]), depending on whether the system was mixed-initiative or fully-automatic, and the number of recommendation categories supported (single or multiple categories). Among the 25 papers, 14 were around data-based recommenders, 6 for hybrid, 5 for encoding-based recommenders. Given the focussed research question we were interested in, we filtered out the 5 encoding-based recommenders and focussed our analysis on the 20 data-based or hybrid recommenders.

Next, we listed all the features and capabilities supported supported across different systems. We were intentionally broad with what we included as recommendations to ensure that our analyses covered a range of capabilities. We then iteratively grouped similar capabilities into categories through multiple passes of the data, refining our definitions and codes. Finally, we presented the ten most common types of analytical actions used to generate and group visualization recommendations across these systems, summarized in Table 5.1.

Taxonomy of Analytical Actions

Our literature review uncovered ten most common types of analytical actions used to generate and group visualization recommendations in existing systems, summarized in Table 5.1. To keep the design space of recommendation categories tractable, we focused on data-based recommendations driven by the operational and characteristic transitions in the visualization design space. We codified these actions into a taxonomy as seen in Figure 5.2. Our taxonomy is not intended to encompass all analytical action types, but rather to synthesize the most

common ones used to categorize recommendations so that we can explore the interaction design space more deeply.

At its highest level, the taxonomy defines two main categories: *operational* and *characteristic*. The *operational* category describes analytical actions that navigate users through the visualization space via operations such as add, remove, and swap. The *characteristic* category describes actions that reveal certain characteristic patterns in the data, such as skewness and correlation. The taxonomy is further broken down into *context-dependent* and *context-independent* categories. Actions are *context-dependent* if they depend on the *current view* or visualization (i.e., the selected attributes, values, and visual encodings); they are *context-independent* if they do not depend on the *current view*.

Note that the operational actions described above overlaps with some of the categories in the chart transition model in GraphScape [111]. However, GraphScape provides a chart transition model that describes visualization edits, whereas our taxonomy describes actions in a VisRec context drawn from existing VisRec systems. Since our focus is less on encoding-based recommendations, we do not consider such aspects of the GraphScape taxonomy such as Scale and Mark. Likewise, GraphScape does not include characteristic actions described in this taxonomy.

Operational Analytical Actions

Operational actions apply data-oriented operations that transition the current view to a related, neighboring part of the visualization space. By definition, analytical actions that are operational must also be *context-dependent* as they operate on the current view. As seen in Figure 5.2, there are three broad categories of operational actions based on whether an attribute or value is added, removed, or swapped, leading to six (3×2) individual categories.

The example in Figure 5.3 demonstrates how operational actions can be thought of as moving along different paths in the *attribute* or *value* hierarchy. Every node in the attribute or value hierarchy represents a set of selected attributes or values. A user's current visualization is composed of their position on the attribute hierarchy (i.e., the space of all attribute combinations) and their position on the value hierarchy (i.e., the space of all filter value combinations). Movements through these hierarchies defines the set of possible operational actions. This conceptual model formalizes the space of possible visualizations that are one move away from the current visualization. The names of each action is based on the operations performed on user's current view. For example, users can either `Enhance` their current view by adding an attribute or `Generalize` to simplify the level of detail within their current view.

Our model draws on Online Analytical Processing (OLAP), a sub-field of data management that targets analytical querying of multi-dimensional data. However, unlike OLAP, which only considers the value hierarchy [66], we introduced the analogous attribute hierarchy to help capture common operations in visual analytics. Here are the six operational analytical action categories:

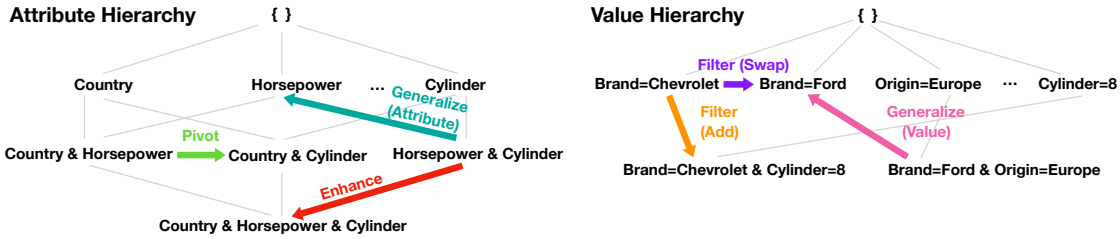


Figure 5.3: Operational actions represent transitions through the attribute and value hierarchies.

- **`Enhance`:** adds an additional attribute to the current view. If the user selects attributes A and B, `Enhance` displays visualizations involving attributes A, B, and C. This action corresponds to moving down the attribute hierarchy (Figure 5.3 red).

- **`Filter (add)`:** adds an additional filter to the current view. If the user selects attributes A and B, `Filter (add)` displays visualizations involving A, B, and a filter F. As described in Chapter 3, this is known as a *drill-down* in the value hierarchy in OLAP (Figure 5.3 orange).

- **`Filter (swap)`:** switches out the filter value to a different value, while keeping the filter attribute fixed. If the user selects attributes A and filter $F = V$, `Filter (swap)` displays visualizations involving A and an alternative filter $F = V'$. This action corresponds to moving horizontally across the value hierarchy to a node with the same filter attribute (Figure 5.3 purple).

- **`Generalize (attribute)`:** removes one attribute from the current view to display the more general trend. If the user selects attributes A and B, visualizations involving either A or B are displayed. This action corresponds to moving up the attribute hierarchy (Figure 5.3 turquoise).

- **`Generalize (value)`:** removes one filter from the current view to display the more general trend. If the user selects attributes A and a filter F, visualizations involving only A are displayed. In OLAP, the removal of a filter is known as a *roll-up* on the value hierarchy (Figure 5.3 pink).

- **`Pivot`:** displays visualizations that can be constructed if one of the attributes from the current view is replaced with another attribute[2]. If the user selects attributes A and B, `Pivot` displays visualizations involving either A and another attribute B', or B and another attribute A'. This action corresponds to moving horizontally along the attribute hierarchy (Figure 5.3 green).

[2]Note that the term *pivot* here should not be confused with the pivot table operation in spreadsheets.

Characteristic Analytical Actions

Characteristic analytical actions are designed to surface salient visual and statistical characteristics of the data[3], sorted based on an interestingess metric that is described in Chapter 5.2. Characteristic actions that are *context-independent* are designed with an overview intent, similar to "breadth-first" exploration strategies in web search [215] that are independent of the user's search query. We describe two types of independent actions that highlight patterns that may be of interest to the user:

- `Correlation:` highlights bivariate relationships between quantitative fields in the data through scatterplots of different combinations of quantitative attributes.
- `Distribution:` displays the possible univariate distributions in the dataset, with `COUNT` as the default measure. The visualization can either be a histogram, bar chart, or line chart depending on the data type of the attribute.

Characteristic actions that are *context-dependent* showcase salient visual characteristics based on the current view.

- `Similarity/Difference:` highlights data patterns that are visually similar or different from the current view.

Our taxonomy faithfully captures a diverse set of categories implemented in a variety of VisRec systems. As a result, there are overlaps between different actions in the taxonomy. For example, when a quantitative field is in the current view, `Enhance` displays a subset of `Correlation` that contains the selected field. Systems that implement such a taxonomy need to strike a balance between showing a comprehensive set of categories and avoiding multiple categories that might recommend the same visualizations. We will describe how FRONTIER selects categories to avoid duplicate recommendations in Chapter 5.3.

Ranking Objectives

Within each analytical action category, visualizations are often ranked using some interestingness objective. Given the different chart characteristics for different types of visualizations, the interestingness objective, even for a given action, may be different for visualization types. For example, a user may be interested in the degree of correlation in a scatterplot, while they may be interested in differences between the bar values in a bar chart. In this work, we consider commonly occurring basic chart types, including bar charts, histograms, line charts, and scatterplots, typically employed by existing VisRec systems. Even this set results in a considerable number of choices corresponding to every combination of action and

[3]In this taxonomy, we focus on multidimensional structured (i.e., relational) datasets as opposed to graph or semi-structured datasets.

visualization type. We identified a small number of classes of objectives that have been used in prior work for these combinations, which we catalog below.

For the *characteristic* actions, the objective typically captures the salient visual characteristics expressed by a visualization, such as the degree of correlation or skew. Visualizations are ranked from the `Correlation` action based on monotonicity [236], typically most to least correlated, while those from the `Distribution` action are ranked from most to least skewed. For the `Similarity` action, bar and line chart visualizations are ranked based on similarity to the *current view*, computed via the Euclidean distance between the measure values of the visualizations [200, 127].

For the *operational* actions, the objective used is typically determined by the visualization type of the recommended visualizations. These objectives capture perceptual characteristics generally associated with something unexpected or insightful in the visualization, including:

- **Non-uniformity**: For *bar/line charts and histograms without a filter*, visualizations are ranked highly if they are highly uneven, indicating the presence of outlying categories or shifts in distributions [41, 39].
- **Deviation**: For *bar/line charts and histograms with a filter*, the ranking is based on the deviation between the filtered and unfiltered (overall) distributions, based on the intuition that a visualization is potentially interesting if it differs greatly from some expected reference [122, 220].
- **Correlation**: For *uncolored scatterplots*, a visualization is ranked higher if it displays a high degree of dependence between the two measures, as measured by mutual information [154, 101] or Spearman's correlation [236].
- **Separability:** For *colored scatterplots*, a visualization is ranked higher if the colors for each category distinctly separate clusters of data points in the scatterplot [189, 39].

5.3 The Frontier System

We introduce Frontier, a GUI-based charting tool that provides visualization recommendations across multiple analytical action-based categories. Frontier is a design probe that enables us to systematically explore and compare these categories. For brevity, we refer to the analytical action-based categories displayed in Frontier simply as *recommendation categories* henceforth. The Frontier interface is composed of four areas as illustrated in Figure 5.4. Starting from the left (Figure 5.4A), we have the Control Panel, a manual specification interface for specifying the visualization in the Current View (Figure 5.4B). The Control Panel lists measures and dimensions and allows users to add or remove attributes and values. The Specification Panel (Figure 5.4A top) allows users to fine-tune their visualizations by arranging attributes across specific encoding channels. Users can toggle on and off specific recommendation categories. If a category is not applicable for the given Current View, the cursor icon changes to a forbidden sign upon hover in the Category Menu

(Figure 5.4C). The recommendations are displayed row-by-row on the right (Figure 5.4D) analogous to faceted web search results to encourage browsing [247].

We drew inspiration from existing VisRec interfaces [241, 87, 41] and employed guidelines from mixed-initiative interfaces [86, 157] to balance interface usability with comprehensiveness in the display of recommendation categories. We iterated on the design with feedback from an interaction designer and made significant changes to the interface over a period of six months.

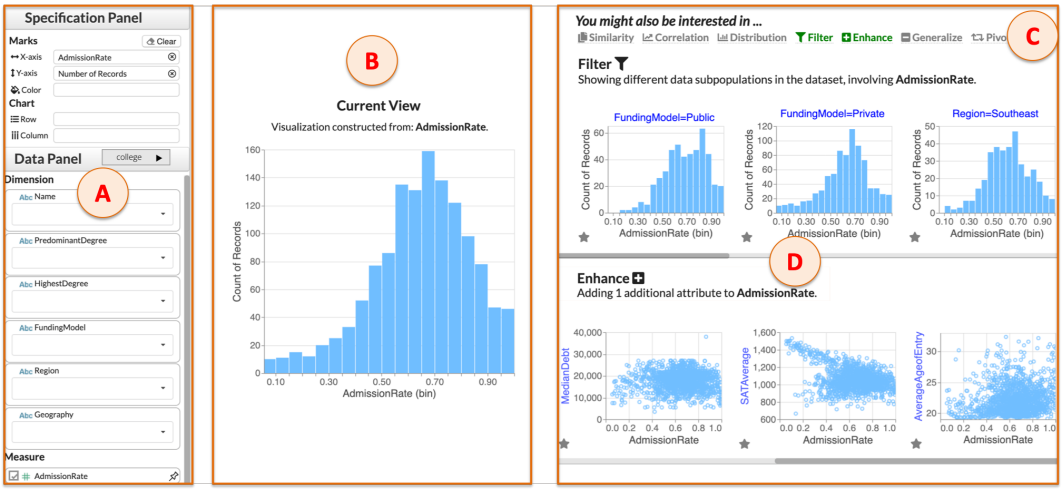


Figure 5.4: FRONTIER consists of four areas: Control Panel (A), Current View (B), Category Menu (C), and Recommendations Panel (D).

Design considerations

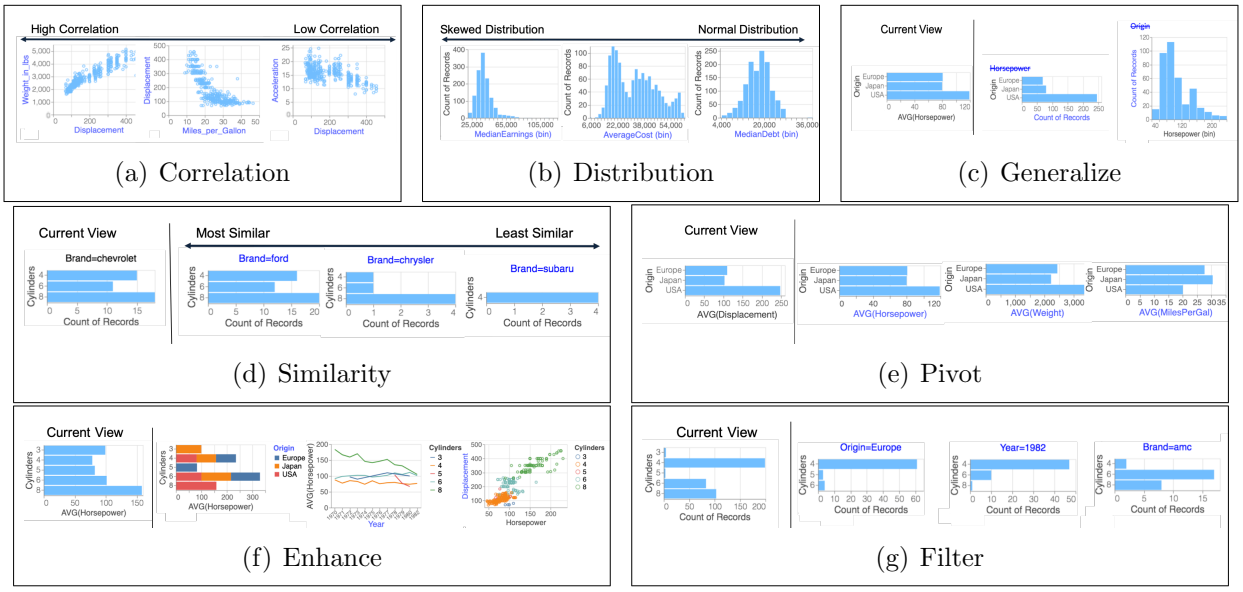
The following design considerations emerged while iteratively designing FRONTIER:

- C1: *Concise and informative.* Recommendation categories should provide a manageable set of options as "next-steps" in a user's analytical workflow. Users should never be shown an empty category nor should they be shown categories with overlapping recommendations.
- C2: *Coordinated and actionable.* Recommendation categories should be coordinated and consistent with other parts of the system, such as in the Category Menu and Current View. The user should be able to bring a recommended visualization into the Current View.
- C3: *Interpretable and visually discernible.* Recommendation categories should be self-explanatory and display visual indicators that convey their key characteristics or highlight how they differ from the Current View.

These requirements echo design considerations from prior work in mixed-initiative visual analytics systems [241, 204].

System Overview

FRONTIER is a web-based system with components described as follows. First, the *Data Manager* loads the dataset and metadata (i.e., the data type, data model, and default aggregation) and computes statistics (i.e., cardinality, correlation, minimum, maximum). The *Context Manager* maintains information about the attributes and values that the user has selected. Then the *Category Manager* determines which recommendation categories to display for a given Current View and maintains a list of categories. Finally, each *Category* contains information about specific recommendation categories, a sorted list of top-k recommended visualizations, and their associated scores.



(a) Correlation (b) Distribution (c) Generalize

(d) Similarity (e) Pivot

(f) Enhance (g) Filter

Figure 5.5: Examples of various recommendation categories implemented in FRONTIER. (A) **Correlation** generates scatterplots with bivariate relationships between quantitative fields ranging from high to low correlation. (B) **Distribution** shows the possible univariate distributions from the dataset ranging from skewed to normal distributions. In the following examples, the current view is shown on the left, with the corresponding recommendations shown on the right. (C) **Generalize** shows possible visualizations when one attribute or filter from the current view is removed (removed attributes shown with strikethroughs). (D) **Similarity** highlights data patterns ranging from most to least similar to the current view. (E) **Pivot** shows possible visualizations that can be constructed if one of the current attributes is changed to another (changed attributes shown in blue). (F) **Enhance** shows possible visualizations when an additional attribute is added to the current view (additional attributes shown in blue). (G) **Filter** displays the data subsets that can be constructed from the current view when a filter is applied.

Category Generation and Visualization Interaction

To select a manageable set of recommendation categories (C1) to display to users, we designed the following workflow. These rules are similar to the ones adopted in Voyager [242] and DIVE [87], which first provide an overview via univariate distributions, followed by more relevant visualizations based on subsequent user selection.

At the start of the analysis, when no attributes are selected, the `Correlation` and `Distribution` actions display univariate and bivariate visualizations, enabling users to get an overview of the dataset (Figure 5.5A, 5.5B). Operational categories evolve based on the Current View and come into play once any attribute is selected. To avoid redundancy across categories (C1), FRONTIER only shows context-independent categories when there are no attributes selected. For example, when a user selects a single quantitative attribute, `Enhance` (Figure 5.5F) generates a collection of scatterplots, similar to what is shown for `Correlation`. Similarly, when a categorical attribute is in the Current View, only `Pivot` recommendations (Figure 5.5E) are displayed, to avoid operating over the same collection of visualizations as the ones in `Distribution`.

To make the recommendation categories more succinct and manageable (C1), from Table 5.1, FRONTIER consolidates filter (add) and filter (swap) to give `Filter`, generalize (attribute) and generalize (value) to give `Generalize`, similarity and difference to give `Similarity`. FRONTIER's `Filter` adds an additional filter to the Current View on any categorical attribute[4] when there is no filter in the Current View (Figure 5.5G). When a filter is in place, `Filter` keeps the specified filter attribute, while swapping out one of the attribute values to showcase alternative data subsets for comparison. Note that any applied filter is always retained in all actions except in `Filter`. In `Generalize`, we display all possible visualizations by removing either one filter or one attribute that is in the Current View (Figure 5.5C). In `Similarity`, visualizations that look most similar to the Current View are ranked highest, but users can reverse the sort order to look at the most dissimilar visualizations (Figure 5.5D).

Users can double click any recommendation to bring the visualization into the Current View; this sets elements in the Control Panel to be consistent with that of the selected visualization (C2). We display the axis label of any element that differs between the Current View and the recommendation in blue, to ease comparison and highlight differences (C3).

5.4 Study Design

We conducted a mixed-methods study to explore how various recommendation categories impact visual analysis workflows. Our primary goal was to study the relative usefulness of various categories as well as how categorization influences the analysis workflow in general. To study these goals, our design probe, FRONTIER, implements the categories described in the previous section. Further, to understand the effect of categorization in mixed-initiative

[4]Note that FRONTIER only supports filters on categorical variables, although filters on quantitative variables, e.g., based on ranges are in indeed possible [99].

VisRec workflows, we included a mixed-initiative VisRec baseline, mirrored off of FRONTIER, that featured the same recommendations, but without any categorization. We describe this baseline later on. Overall, our exploratory study aimed to address the following research questions:

- RQ1: How do recommendation categories support and influence analytical workflows? What problem-solving and exploration strategies do users adopt when using recommendation categories in a mixed-initiative context?
- RQ2: What are the differences in user behavior across recommendation categories? What is the value and impact of individual recommendation categories and how does this vary across tasks and datasets?

Participants

We recruited 24 participants (10 female, 14 male) from within a software company. Nine were experienced users of Tableau, 13 had limited proficiency, and two had no experience. In a between-subjects design, participants were randomly assigned to use FRONTIER or *Baseline* with either the College [217] or Olympic Medals [1] dataset, with six participants per condition-dataset combination. Henceforth, we suffix $.F$ or $.B$ in the identifier to display whether the participant used FRONTIER or the *Baseline* condition.

Tasks and Data

There were two main parts to the study: closed-ended tasks and open-ended exploration.

Part 1: Closed-ended tasks

Closed-ended tasks were mainly intended to familiarize participants with the system while also providing some consistent objectives for task comparison. Participants completed four closed-ended questions that included common visual analytic tasks, including:

- Q1 (*Correlate*): find other measures that are linearly correlated with a selected attribute.
- Q2 (*Filter Compare*): compare bar charts across different data sub-populations.
- Q3 (*One v.s. All*): compare a filtered distribution with the overall distribution.
- Q4 (*Pattern*): compare the temporal trend across different measures.

For each task, participants answered a multiple choice question on a paper worksheet. They were instructed to use FRONTIER to answer the question, but were not told how to do so. All participants used the Cars dataset [222] for closed-ended tasks. This dataset was chosen because of its simple schema (five measures and five dimensions), clean insight patterns, and

because it is commonly used for demonstrating visualization systems [188, 241, 39], thus enabling comparisons.

Part 2: Open-ended exploration

Following the closed-ended tasks, participants completed an open-ended exploration task. This task enabled us to observe how people would choose to use (or not use) the recommendations in a natural analysis flow. Participants explored either the College or Olympic Medals dataset. Instructions were: "*We'd like you to explore this data to look for interesting insights. As you work, please let us know what questions or hypotheses you're trying to answer as well as any insights that you are learning about the data.*" Participants were instructed to talk aloud and star recommendations they found useful.

The two datasets for open-ended exploration were chosen due to their real-world and accessible nature. We chose datasets with different characteristics, enabling us to study a wider range of analytical inquiries: the College dataset has ten measures and six dimensions with low to medium cardinality, while the Medals dataset contains only three measures and twelve dimensions with medium to high cardinality.

Apparatus

In the FRONTIER condition, participants used the full version of FRONTIER with all of the recommendation categories. The ordering of recommendation categories within the interface was randomized for each user to minimize the preference for recommendations displayed at the top of the page.

To study the impact of categorization as a whole, we introduced a *Baseline* condition. This condition displayed the same set of recommendations except that the recommendation categories were removed so that all the recommendations appeared in a single, grid layout[5]. The goal of this *Baseline* is to establish a vanilla VisRec system that *(i)* eliminates the effects of recommendation categories, while preserving certain characteristics for a controlled comparison in that *(ii)* it is mixed-initiative and *(iii)* displays the same overall set of recommendations.

To understand this condition better, note that the organization of VisRec into categories is a result of both the labeling (i.e., interface elements such as dividers and textual descriptions of the categories) as well as the ranking within each category. To remove the effects of categorization *(i)*, we not only had to remove the category labels, but also had to shuffle the display order of the recommendations. In both conditions, participants can browse for more visualizations via horizontal scrolling (single scroll bar for the *Baseline*, one scroll bar per recommendation category for FRONTIER). To prevent preferential bias towards top-ranked visualizations, we ensured that the exact same set of visualizations appeared with and without scrolling across both conditions.

[5]The *Baseline* interface has a similar layout to several existing VisRec systems [237, 107]

One could potentially argue that the lack of organization in our baseline can be a confound. However, after considering alternative designs, including a no-recommendations baseline or no baseline at all, we concluded that our chosen baseline was the most appropriate option. We opted against comparing with manual specification tools without recommendations, given the general benefits of VisRec as shown in prior studies [39, 241, 242]. We also opted against comparing with existing VisRec systems that implement a subset of categories, as this would not allow us to tease apart the impact of categorization on analytic workflows. A baseline that only removes the category labels, but does not alter the display order, would only evaluate the effects of explicit category labels and is thus not a meaningfully different baseline. Since the goal of the study is not to demonstrate performance difference between Frontier and *Baseline*, we opted for a baseline with recommendations for investigating the research questions around recommendation categories.

Procedure

Sessions lasted approximately one hour, consisting of approximately five minutes of introduction and tutorial, 15 minutes of closed-ended tasks, 30 minutes of open-ended exploration, and 10 minutes of semi-structured interviewing. The tutorial video introduced the interface using the Cars dataset and stated that recommendations were selected based on an interestingness ranking and that the blue text indicated changes from the Current View. For Frontier, the video additionally described each recommendation category. The post-study interview included 7-point Likert scale questions (e.g., overall usability, recommendation usefulness) and open-ended questions on the system design and recommendations.

Analysis Approach

We employed a mixed-methods approach involving both qualitative and quantitative analyses. The primary focus of our work was a qualitative analysis of how recommendations of different categories influenced people's analytical workflows. We conducted a thematic analysis through open-coding of session videos, focusing on strategies participants took to answer their questions.

We thematically classified each participant based on how frequently they engaged with manual controls versus the recommendation panels. To obtain these classifications, we assigned separate labels for characterizing each participant's usage of the Control Panel and the recommendations (1: Majority of the time, 2: Sometimes, 3: Not often). Based on these labels, we grouped the participants by their relative frequency of use, where participants employed a *manual-oriented strategy* if they exhibited a higher usage of the Control Panel than recommendations, *balanced* if they had comparable usage of both, and *recommendation-oriented* if they exhibited a higher usage of recommendations than the Control Panel. Additionally, we define a visualization as *useful* if one or more of the following occurred: (a) the participant verbally described an insight, (b) the visualization was brought into view, (c) the visualization was starred, or (d) the participant expressed that it was useful or interesting.

We coded insights from the video recordings, reusing the definition of an insight from prior work [185, 133].

The quantitative analysis consisted of Likert question results from the interview as well as counts of expressed hypotheses, data insights, and recommendations participants found useful. We employed statistical testing where appropriate, but considered the quantitative analysis mainly as a complement to our qualitative findings. We adopted a 95% confidence interval for all statistical analyses. Our analysis approach is similar to other studies that employed mixed-methods to investigate analytical workflows [140, 185].

5.5 Study Findings

RQ1: How do recommendation categories support mixed-initiative analysis workflows?

To understand how recommendation categories support analytical workflows, we first examine the strategies participants adopted and understand their motivations for switching between different modes of exploration. Then we delve deeper to examine the specific benefits of recommendation categories and their affordances. Finally, we highlight how user's perceptions regarding the recommendation categories can evolve over the course of an analysis workflow.

Strategies in mixed-initiative recommendation workflows

Based on thematic analysis of how frequently participants engaged with manual control versus recommendation panels, we observe three major strategies that they employed across both the Frontier and *Baseline* conditions. We generally observe that participants were more inclined to use recommendations in their workflow when using Frontier than in the *Baseline*. We sought to better understand participants' motivations for opting for different analysis options. Participants employed a recommendation-oriented strategy for exploring unfamiliar attributes ($N_{F,B} = 6, 3$ participants[6]) during preliminary analysis ($N_{F,B} = 5, 4$) or when they were out of ideas on what to pursue further ($N_{F,B} = 5, 1$). We found a small group of participants ($N = 3$ for Frontier; $N = 1$ for *Baseline*) who relied almost entirely on the recommendations to drive their analyses and used the Control Panel only sparingly. Most of these participants either expressed that they had limited experience with creating visualizations or were unsure what to expect from the dataset. The sentiment expressed by these participants largely corresponded to the challenges that visualization novices face in translating abstract questions about their data to visualization specifications [65]. As

[6]We use the notation $N_{F,B}$ to report measurements for Frontier and *Baseline* respectively. In the example above, $N_{F,B} = 6, 3$ means that six participants using Frontier and three participants using *Baseline* used recommendations to explore unfamiliar attributes.

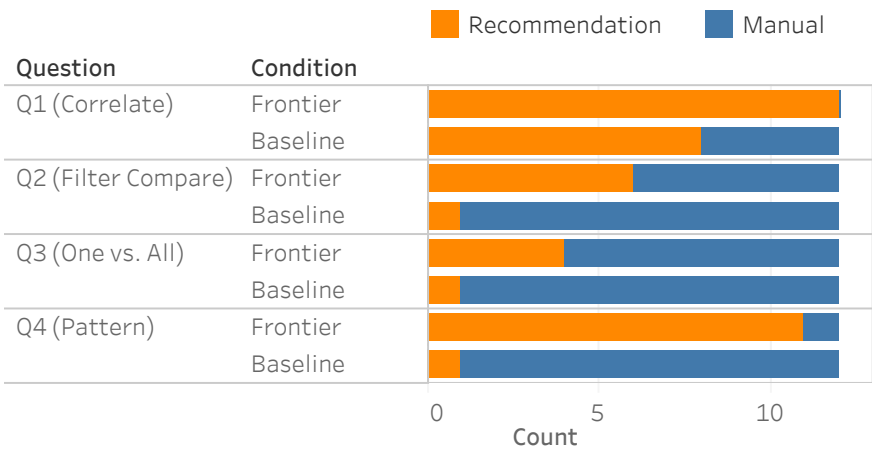


Figure 5.6: The number of participants who took a manual-oriented approach in solving the closed-ended question versus a recommendation-oriented approach.

$P6.F$ explained "*...the recommendations gave me a jumpstart [...] because if I didn't have the recommendations to begin with, I wouldn't even know where to start.*"

Participants also adopted a *balanced* strategy intermixing the use of the Control Panel and recommendations in unexpected ways. Three participants ($P7.F$, $P8.F$, $P17.B$) selected recommended visualizations that were "close enough" to what they wanted, then made minor tweaks using the Control Panel to attain their desired visualization. Participants also created familiar visualizations to trigger desired recommendations. For example, $P9.F$ wanted to look for linear trends in the data. They recalled seeing a clean linear trend between ACT and SAT scores previously, so they first created the same visualization via the Control Panel. Then they browsed through recommendations resulting from `Similarity` in order to find similar visualizations. Participants were able to leverage recommendations effectively in their workflow since the recommendation categories were transparent and interpretable, leading to predictable behavior.

Participants followed a *manual-oriented* strategy when the perceived cost of engaging in manual specification was lower than the effort it took to interact with the recommendations. This occurs when participants had a specific hypothesis in mind ($P12.F$, $P15.B$, $P16.B$, $P17.B$) or when participants expressed a preference for manual specification due to their familiarity with existing charting interfaces ($P10.F$, $P16.B$, $P22.B$). $P17.B$ explained the reason why they adopted a manual-oriented approach:

> *If the question that I want to answer is very clear, then I will go do it myself. There are two scenarios that I will switch from the left panel to recommendations. One thing is, I don't know what the next step is and I want insights. Second thing is, I don't know how to do it.*

As shown in Figure 5.6, a similar pattern is also observed for the strategies taken to solve the closed-ended task. In tasks where manual specification required significantly more work than

simply browsing the recommendations for answers (*Correlate*, *Filter Compare*), participants were more likely to adopt a recommendation-oriented strategy. On the other hand, in the *One vs. All* task where participants had to compare a filter and unfiltered visualization, participants opted for the manual-oriented strategy as it was fairly easy to remove a filter.

Participants also adopted a manual-oriented strategy when the perceived effort to interact with recommendations was higher than usual, such as when they were overwhelmed by the large, unorganized panel of recommendations in the *Baseline*. This is supported by the post-study Likert ratings, where participants reported recommendations in *Baseline* to be less useful ($\mu_{F,B}$=4.58, 5.50; $\sigma_{F,B}$=0.90, 0.79; U=33.5, p<0.05 via Mann-Whitney test) and more overwhelming ($\mu_{F,B}$=2.58, 1.76; $\sigma_{F,B}$=1.44, 1.76; U=58.0, p=0.21) than FRONTIER.

Value and impact of recommendation categories

We find that the presence of recommendation categories leads to richer and higher-utility exploration. During open-ended exploration, there were more insights generated via recommendations in FRONTIER than in the *Baseline* ($N_{F,B}$ = 171, 96; t=2.66, p<0.05). A similar trend was observed for the total number of useful visualizations generated via recommendations ($N_{F,B}$ = 149, 82; t=2.47, p<0.05), also shown in Figure 5.7 (left).

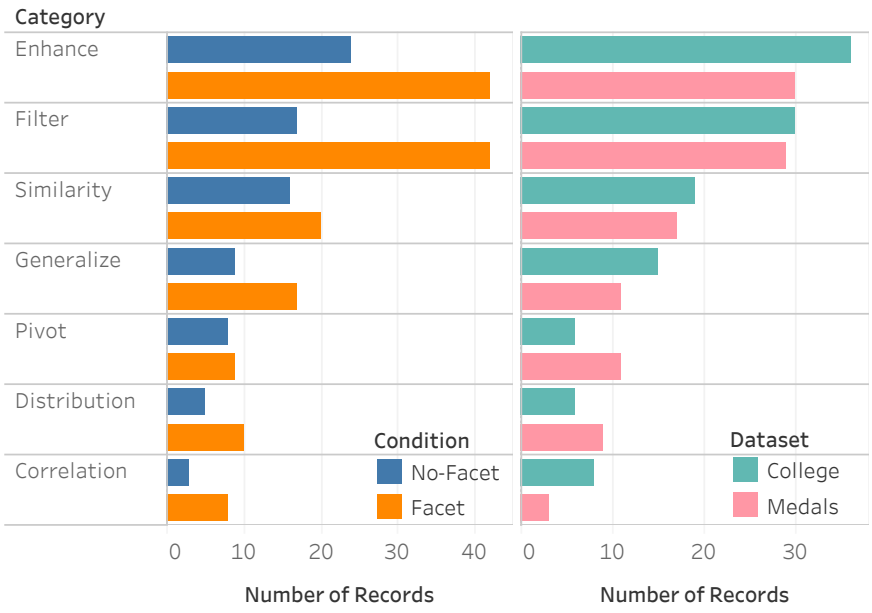


Figure 5.7: Number of useful visualizations for each recommendation category by condition (left) and by dataset (right) for open-ended tasks. Left: `Enhance` and `Filter` are most useful in FRONTIER, but to a lesser extent in *Baseline*. Right: `Pivot` and `Distribution` are more useful in dimension-heavy datasets (e.g., Medals), whereas `Correlation` is more useful in measure-heavy datasets (e.g., College).

Our observations suggest that recommendation categories reduced overhead associated

with interpreting the visualization recommendations. While we did not measure the visualization read-time directly due to the exploratory nature of the tasks, several qualitative observations support this idea. First, six out of 12 participants using FRONTIER expressed that they appreciated the organization. $P5$ noted: "*I like being able to really quickly visually inspect a bunch of things, because I can just slide a bunch of stuff past my eyes and be able to pick the stuff that jumped out.*" In contrast, many participants in the *Baseline* condition went back and forth multiple times between visualizations to make comparisons and ensure that they had the right answer when solving the closed-ended questions ($P18$, 15, 25, 26), which at times led to mistakes.

During the study, we noticed that some participants appeared to be "stuck" in their analyses if they either: a) verbally expressed that they were out of ideas, b) implicitly when they had hypothesizing time of greater than one minute, or c) expressed reluctance to explore further. In particular, only one out of 12 participants using FRONTIER got "stuck" (once) compared to three participants getting "stuck" (total of five occurrences) in the *Baseline* condition. This is partly attributed to how FRONTIER participants repurposed and adapted their workflows to take advantage of the diverse set of actions available through various recommendation categories.

Participants often leveraged categories with the same axes such as `Enhance` and `Filter` to attain insights involving comparisons across multiple visualizations. For example, $P10.F$ was interested in the age distribution of Russian athletes because of their highest medal count. They created a histogram distribution of age for Russia and browsed through the `Filter` action to see distributions for other countries. They exclaimed: "*Oh wow! Italy has some really old people for their medalists*". Seeing the Italy age distribution in the context of other age distributions highlighted its uniqueness; the visualization in isolation would have been uninteresting. Such comparisons across visualizations within an axes-consistent category are prevalent and often lead to better distributional awareness and understanding of the general patterns and trends in the dataset.

Evolving perceptions around recommendation categories

We found that participants came into the study with a diverse set of perceptions and expectations about recommendations that evolved throughout the course of the open-ended exploration session. For example, $P8.F$ explained that "*I feel like these suggestions require a lot more thought process in my head. So for the suggestions, because the one or two times that it doesn't seem a lot useful, I probably disregard it afterwards.*" $P4.F$ echoed a similar sentiment; several uninteresting visualizations early on deteriorated their confidence in `Correlation`: "*I thought that Correlation would be interesting. And it showed me like height and weight correlation and you heard me say I wasn't really interested in that. So it kind of made me nullify the entire Correlation panel together.*"

We also observed the reverse where participants with a negative initial impression of recommendations gained more trust and understanding over time. $P24.B$ expressed that they had a bias against recommendation systems and was reluctant to look at it. How-

ever, finding useful things from the recommendation encouraged them to adopt more of the recommendations in their workflow later on.

> *Before I even started the study, I have a bias about recommendation panels. Because most of the time recommendation panels do not show you what you want. So I'm already kind of wanting to do my own thing and do it myself, because my bias is that is more reliable than using a recommendation. [...] Once it started showing things, I was like, 'Oh, that is kind of interesting', or 'Oh, that is kind of relevant'. I started paying a little more attention to it. I had to keep reminding myself like, 'Oh yeah, this is the part of the screen I'm supposed to be looking at, it can actually be useful'.*

We also observed similar effects on a per-category level. For example, $P11.F$ discovered interesting insights based on `Filter` and noted in the subsequent analysis that they explicitly focused on the `Filter` category because they knew it would likely give something interesting. 11 out of the 24 participants also expressed that there was a learning curve in familiarizing themselves with the recommendation categories. A more longitudinal follow-up study is required to understand how users would interact with the recommendation categories when they become more familiar.

RQ2: What are the differences in usage and utility across recommendation categories?

Enhance and Filter were most useful, while Pivot least

As shown in Figure 5.7, some categories of recommendations were more useful than others. Somewhat surprisingly, while participants reported significantly more useful visualizations in FRONTIER than in *Baseline* especially for `Enhance` and `Filter`, we found that the relative ordering of usefulness for different recommendation types was largely the same independent of the condition. Note that while the categories were not explicitly shown in the *Baseline* interface, they were logged on the system side for the purpose of this analysis. As shown in Figure 5.7, `Enhance` and `Filter` are significantly more useful than `Pivot` (t=4.12, p<0.05; t=3.24, p<0.05 respectively via t-test).

In both conditions, we observed that participants manually performed analytical sequences that were similar to the analytical actions that produced the recommendations. We saw repeated patterns of participants manually performing the pivot operation on 14 separate occasions and the filtering operation on seven occasions. Given that `Pivot` was not actually regarded as very useful compared to the other categories (Figure 5.7), we suspect there may be a difference between the types of operations a user would like to perform manually, versus ones that they would prefer being recommended to them. Regardless, these interaction patterns indicate that `Filter` and `Pivot` do indeed resemble the natural, intuitive next steps in users' workflows.

Participants' post-study ratings of different categories in FRONTIER largely corresponded to the usefulness counts during the session in Figure 5.7. One exception is that `Correlation` and `Distribution` were perceived as easily understandable and useful by more than five out of the eight participants who provided a rating, but had low ranks relative to the usage frequencies captured in Figure 5.7. The discrepancy likely stems from the fact that these two context-independent categories are only displayed when the Current View is empty, so participants did not see them as frequently as the other categories.

Utility of recommendation categories across datasets

As shown in Figure 5.7 (right), the number of useful visualizations from certain categories depended on the dataset. While the usefulness of each category largely followed the trend observed in Figure 5.7 (left), the usage of `Pivot`, `Distribution`, and `Correlation` differed across datasets. Given that the College dataset contained large numbers of measures, while the Medals datasets had few measures and more dimensions, it was no surprise that there were more uses of `Correlation` in the College dataset ($N = 8$) than in the Medals dataset ($N = 3$).

On the other hand, `Distribution` was used more frequently with the Medals dataset ($N = 9$) than in the College dataset ($N = 6$), possibly because it also contained bar charts of the count distributions of dimensions (showing a surprising trend that Europe won significantly more medals than any other continent), whereas `Distribution` for the College dataset showed mostly histograms of measures. `Pivot` was also used more frequently in the Medals dataset ($N = 11$) than in the College dataset ($N = 6$), although the we were unable to determine the reason.

5.6 Study Limitations

While the analytical workflows that participants chose may be influenced by the category selection logic, we tried to minimize the effect of the display order on category preferences by randomizing the ordering of the various categories across users. We did not investigate the effects of recommendation ranking functions, but instead adopted standard data interestingness metrics from the literature [39, 122, 41, 40, 125]. Future work should explore how the interplay between ranking functions, visualization types, and category labels influences the usefulness of recommendations.

While we employed two different datasets to study task effects, future studies with more realistic dataset properties (large, higher-dimensional), larger sample sizes, different problem domains, and varying user expertise would be helpful. We have not explicitly controlled for visualization expertise, leading to more participants with limited proficiency. We also acknowledge potential novelty and unfamiliarity effects in our short one-hour study: most participants did not become fully fluent with all the categories and features provided in

Frontier. The correlation between the warm-up, closed-ended task and the participant behavior may also be a potential confound. As a result, participants exhibited a strong affinity towards the Control Panel due to preconceptions of and familiarity with existing charting tools. A longitudinal study that examines how categorized recommendations are used in practice is important future work.

5.7 Design Implications

From our study findings, we first describe the guiding principles for the design of VisRec categories. We then discuss interface considerations for recommendation categories and their potential pitfalls. Finally, we identify opportunities for supporting analytical actions in visual analysis.

Design guidelines for recommendation categories

Evidence from our study shows how recommendation categories can be powerful constructs that help situate users by establishing a mental framework for reasoning about recommendation results. The semantic grouping and visual affordances of recommendation categories "lift" the visual analysis to operate at the level of analytical actions. By observing how participants switched between manual specification and recommendations, we find that predictable categories reduce users' perceived cost of employing recommendations — crucial for establishing an effective mixed-initiative workflow where recommendations can be used seamlessly in conjunction with manual specification.

Furthermore, the success of `Enhance` and `Filter` lends an important lesson for future VisRec systems in designing simple and readily-accessible recommendation categories. In particular, our study suggests that transparency and interpretability are essential characteristics that lead to recommendation categories that are predictably useful.

One heuristic for evaluating the complexity of a recommendation is to check whether the underlying action addresses a question involving a single element, which can either be a descriptor of the visualization's characteristics or an element that differs from the current view. For example, `Correlation` answers the question: "*Which attributes are correlated?*" and `Enhance` answers: "*What visualizations can be generated by adding one additional attribute?*" Participants' failure to adopt `Pivot` may be partially due to multiple degrees of freedom in which attributes could be swapped. Further, there may be difficulty in articulating the analytical question that `Pivot` affords. This led to additional cognitive effort to reason about what was retained versus varied. Another contributor might be the drastic encoding change that can occur during swapping. This inconsistent behavior can be jarring and inhibits one's ability to compare across the collection. It remains an open question whether "anchoring" techniques that recommends appropriate encodings based on prior context [132] can be applied to recommended collections to offer a more consistent `Pivot`. Encoding inconsistency across collections is never an issue when swapping out values in `Filter` as the

visualized attributes are unchanged when we move across the value hierarchy. This may explain why `Filter` was useful in providing complementary views on sub-populations of data, often leading to unexpected insights.

While our category selection algorithm takes an overview-first [197] approach in showing context-independent categories at the beginning and context-dependent recommendations once a specified view exists, several users explicitly cleared their selection in order to get the overview from time-to-time. Additionally, two participants indicated that they hoped to find visualization recommendations that were more unrelated and surprising rather than simple alternatives to their Current View. The diversity-accuracy tradeoff is a classical problem in recommender system design [2]. Supporting a blend of both types of visualization recommendations is a first step towards assisting users with different information preferences and needs. Further research is needed to develop and evaluate these recommendations as well as to validate our proposed taxonomy.

Pitfalls of categorized recommendations

While recommendations are at most an annoyance when they are not interesting to the user, they can be potentially detrimental if they are used to draw conclusions without deeper examination. $P3.F$ summarized this tradeoff between exploration and exploitation: "*For recommendation, sometimes you get completely irrelevant things, things that are kind of normal. But then on the other side, you get this serendipitous discovery, which is very cool. [...] I mean, it's also dangerous, because you maybe see something where you should further investigate it if it's really an effect.*"

Over-reliance on recommendations could be problematic. For example, $P17.B$ said that they had built trust that the system would show something interesting. When the interface did not show anything interesting, they quickly moved onto the next hypothesis instead of digging deeper because they inferred that the system was telling them not to look there. We observed a similar effect during a closed-ended task, where participants were asked to select the data sub-populations that had more 8-cylinder cars (Q2). When Frontier displayed only visualizations for three out of the four multiple choice answer options in `Filter`, many participants who employed recommendations simply drew their conclusions based on the three visualizations included in the category, without verifying the remaining one.

The potential for erosion of creativity and critical thinking when interacting with an intelligent system is well-known [56, 39]. While this issue is not particular to recommendations based on analytical actions, but a more general phenomenon with recommendations [35], the ease of use and the apparent trust that users perceive from recommendation categories may exacerbate these issues. This challenge points to a need for future research in designing recommendations that provide some notion of coverage or inform users about what has or has not been examined.

Towards personalized, adaptive recommendations

Even though categorized visualization recommendations provide a means of organization, limited screen space makes it impossible to show all categories at once. Furthermore, users typically only peruse the first few items of a recommended list [252]. The diversity of preferences and individual strategies observed in our study suggests that personalized selection of recommendations may be worthwhile. For instance, while ten out of the 24 participants believed that there should be fewer recommendations, two participants (one from each condition) wanted to see more.

In post-study interviews, participants indicated that they wanted a more user-driven approach in creating their own organization. Three wanted the ability to extract selected recommendations into a separate dashboard, tab, or page and rearrange them freely into their own groups; eight wanted to hide some or all parts of the recommendation categories and retrieve them on demand. This points to an interesting future direction towards a hybrid human-recommender workflow. Similar to the variability of people's web search behavior [228], recommendations could be adaptive and personalized to tailor to users' preferences.

Personalization yields potential benefits beyond providing adaptive interfaces, namely, in providing optimization opportunities for system scalability. One of the challenges for visualization recommendations systems is the high computational cost associated with searching over a large search space of possible visualizations [221, 220, 39]. Given that there are preferences for certain categories over others for different users, datasets, and tasks, there is an opportunity to reduce the computational cost of an exhaustive search by pruning the recommendation search space.

5.8 Conclusion

The goal of recommender systems is to anticipate future user needs. In GUI-based charting tools, visualization recommendation categories help organize these possible futures as readily-available options to drive analytical workflows. In this chapter, we introduced a taxonomy based on prior literature to examine the usefulness of various recommendation categories based on the underlying analytical actions. We implemented Frontier as a design probe to better understand how a general-purpose visual exploration assistant can encourage users in the next steps of their exploration. Our user study confirmed that recommendation categories are indeed useful for facilitating data exploration, helping users understand and interpret the visualizations. While the general utility of categorization was not surprising, we more deeply explored *how* various categories of visualization recommendations were employed and the diverse workflow strategies that users adopted. Design implications stemming from this study provide unique opportunities for supporting general-purpose automated assistance in GUI-based charting tools. In the next chapter, we look at how this work on general-purpose visual exploration assistants can be applied to workflows beyond GUI-based charting tools, in particular, for exploratory programming workflows in computational notebooks.

Chapter 6

Assistance in Computational Notebooks with Lux

Figure 6.1: Top: When a user prints out a dataframe in Lux, the default `pandas` table is displayed. Bottom: Users can toggle to browse through a set of dataframe visualizations suggested by Lux.

In the last few chapters, we saw how visual exploration assistants can guide analysts when working with GUI-based visualization tools. Given that exploratory data science largely happens in computational notebooks, such as Jupyter [114], in this chapter, we study how visual exploration assistants can be integrated into the context of computational notebooks. Within notebooks, dataframe libraries, such as `pandas` [210], support flexible means to transform, clean, and analyze data. Yet, visually exploring data in dataframes incurs substantial programming effort and overhead. This chapter presents Lux, an *always-*

on framework for accelerating visual insight discovery in dataframe workflows. When users print a dataframe in their notebooks, Lux recommends visualizations to provide a quick overview of the patterns and trends and suggests promising analysis directions. Lux distills the lessons learned from the previous chapters to design a general-purpose, mixed-modality visual exploration assistant.

6.1 Introduction

As described in Chapter 2, apart from GUI-based charting tools, analysts (in particular data scientists) often leverage a dataframe library [168, 98], such as `pandas` [210], to perform exploratory data analysis. Dataframes offer a flexible and rich set of operators to transform, analyze, and clean tabular datasets. Data scientists typically manipulate dataframes within a computational notebook such as Jupyter, which offers a flexible medium to write and execute snippets of code; nearly 75% of data scientists use notebooks everyday [206]. In between these dataframe transformation operations, users visually inspect intermediate results, either by printing the dataframe contents as a table, or by using a visualization library to generate visual summaries. This visual inspection is *essential* to validate whether the prior operations had their desired effect and determine what needs to be done next. However, visualizing dataframes is a cumbersome and error-prone process, adding substantial friction to the fluid, iterative process of data science, for two reasons: cumbersome boilerplate code and challenges in determining the next steps.

Cumbersome Boilerplate Code. Substantial boilerplate code is necessary to simply generate a visualization from dataframes. In a formative study, we analyzed a sample of 587 publicly-available notebooks from Rule et al. (2018) [179] to understand current visualization practices. A surprising number of notebooks apply a series of *data processing* operations to wrangle the dataframe into a form amenable to visualization, followed by a set of highly-templatized *visualization specification* code snippets copy-and-pasted across the notebook. Our findings echo a recent study of 6386 Github notebooks [115], where visualization code was the most dominant category of duplicated code (21%). On top of the high cognitive cost when writing "glue code" to go from dataframes to visualizations [225, 5], users have to context-switch between thinking about data operations and visual elements. These barriers hinder exploratory visualizations and, as a result, users often only visualize during the "*late stages of [their] workflow*" [15, 100], rather than for experimenting with possible analyses—which is precisely when visualization is likely to be most useful.

Challenges in Determining Next Steps. Beyond writing code to generate a given visualization, there are challenges in determining which visualizations to generate in the first place. Dataframe APIs support datasets with thousands of records and hundreds of attributes, leading to many combinations of visualizations that can be generated. The many choices make it hard for the data scientist to determine what visualization to generate to advance analysis. They receive no automated guidance on what may be valuable visual-

izations to examine next. While there has been some work on automated visualization recommendation in the context of interactive visual analytics tools [242, 241, 200, 124], and as discussed in previous chapters, targeting identification of "interesting" patterns, trends, or insights, none of this work has impacted typical data science workflows in computational notebooks. The former is an easier problem since datasets are static; in a computational notebook, the dataframes are continuously evolving as data scientists perform data cleaning and transformation operations.

Always-On Visualization Recommendations with Lux. To address the above challenges, we introduce Lux[1], a seamless extension to `pandas` that retains its convenient and powerful API, but enhances the tabular outputs with automatically-generated visualizations highlighting interesting patterns and suggesting next-steps for analysis. Lux has already been adopted by data scientists from a diverse set of industries, and has gained traction in the open-source community, with over 2.6k stars on Github and 25k total downloads on PyPI as of August 2021. Lux is being used by data scientists across a range of industries spanning from healthcare to finance, with an active, organic community of enthusiasts creating blog posts, tweets, and videos explaining the value of Lux [57, 48, 163, 232, 135, 136].

Challenges of Always-On Visualization Recommendations. Prior work has examined supporting automatic recommendations of interesting summaries in an OLAP setting, e.g., [220, 200, 241, 221, 122, 243, 183, 99], and automatically picking the right visualization modality, given attributes of interest [242, 207, 138, 139]. However, providing always-on visualization recommendations while data scientists perform ad-hoc exploration of dataframes is non-trivial and presents its own unique research challenges:

What and how do we recommend? Data scientists using dataframes are unlikely to use a visualization tool that causes any disruption to their workflow. How do we make visualization recommendations as easy to peruse as the tabular view provided on printing the dataframe within a computational notebook? What types of useful visualization recommendations do we show? There are lots of visualizations that could be generated on a given dataset.

How do we support dataframe evolution? Unlike traditional visual analytics, dataframes are continually evolving over the course of data science. Operations involving pivots or grouping can drastically change the shape of the dataframe. How do we provide visualization recommendations as the dataframe metadata (cardinalities and data types for columns) is changing rapidly? The cost of updating the metadata and recommendations at every point in a dataframe workflow to keep the recommendations "always-on" is often high.

How can we be informed by the dataframe operations users are performing? How do we ensure that the visualization recommendations are relevant and useful, based on the operations that the users have performed? For example, if a user has just performed a grouping, it may indicate that the group-by column is of interest to them.

How do we allow users to steer the visualizations they want to see? Simply providing users the ability to passively receive visualization recommendations without any power to indicate

[1] `https://github.com/lux-org/lux`

their interests to Lux is not useful. How do we allow users to provide their "fuzzy intent" in a lightweight manner quickly and without having to write a lot of code—with the system filling in the gaps as needed?

How do we keep it interactive? Visualization recommendation involves traversing through a large search space of candidate visualizations to select ones that would be most interesting to the user. It is critical to provide interactive feedback—even seconds of latency substantially discourages users from visually inspecting their dataframes altogether. How do we ensure that the overhead of visualization recommendations are not substantial?

How do we allow users to export and edit? Often users want to be able to take the visualizations and further customize it to their needs. How do we enable users to export visualizations and edit them in their favorite visualization specification language?

How do we continue to support the rich `pandas` API? How do we provide this experience when continuing to support `pandas`' 200+ operators—without compromising the ease and flexibility of programmatic data transformation and preparation as is done presently?

The Lux Approach. We address the aforementioned challenges in developing Lux. Lux preserves all the functionalities of present-day dataframes, while augmenting the default tabular dataframe view with a toggle button to switch to visualization recommendations. Lux is a lightweight wrapper around `pandas` that intelligently caches and lazily evaluates the metadata and recommendations associated with a dataframe. At any point during the dataframe workflow, Lux offers an intuitive way of visualizing the dataframe. These include the types of visualizations common in past visualization recommendation systems, as well as novel dataframe visualizations based on structural (Series, Index) and history information. Lux additionally offers a powerful, intuitive and succinct intent language powered by a formal, expressive algebra that allows users to specify their fuzzy intent at a high-level. Lux implements an intent processing stack that compiles the declarative specification into appropriate visualization mappings. Overall, users can use Lux to quickly compose one or more visualizations, and get visualization recommendations for the next steps in their analysis. A naive implementation of recommendation on top of dataframes can be extremely costly incurring up to 575× slowdown relative to `pandas`. Lux ensures interactive visual feedback through a series of optimization strategies that minimize the overhead incurred on top of a dataframe workflow. Lux adds no more than two seconds of overhead on top of medium-to-large real-world datasets with characteristics covering around 98% of datasets in the UCI repository. Finally, Lux has intuitive ways to export one or more visualizations, as well as edit the underlying code for customization.

Our contributions are as follows:

- We show how Lux supports visual interactions with dataframes, and introduce a *dataframe interaction framework* (Chapter 6.2).
- We introduce *intent* as a high-level mechanism to convey aspects of interest to Lux, with a grammar and query language (Chapter 6.3).

- We introduce four classes of recommendations based on the metadata, intent, structure, and history. The latter two are dataframe-specific ones that have not been explored in prior work (Chapter 6.4).
- We develop a modular system, Lux, that interprets intent and generates recommendations (Chapter 6.5) with an efficient execution engine for metadata and visualization computation (Chapter 6.6).
- Finally, we evaluate the interactive performance of Lux (Chapter 6.7) and conduct usability studies with data scientists and early adopters (Chapter 6.8).

6.2 Visual Dataframe Workflows

We first demonstrate how always-on visualization support for dataframes accelerates exploration and discovery.

Lux Example Workfow

We present a workflow of Alice, a public policy analyst, exploring the relationship between world developmental indicators (such as life expectancy, inequality, and wellbeing) and the country's early effort in COVID-19 response. A live demo of the example notebook can be found here: `https://mybinder.org/v2/gh/lux-org/lux-binder/master?urlpath=tree/demo/hpi_covid_demo.ipynb`.

Always-on dataframe visualization. Alice opens up a Jupyter notebook and imports `pandas` and Lux. Using `pandas`'s `read_csv` command, Alice loads the Happy Planet Index (HPI) [72] dataset of country-level data on sustainability and well-being. To get an overview, Alice prints[2] the dataframe `df` and Lux displays the default `pandas` tabular view, as shown in Figure 6.2 (top, orange box). By clicking on the toggle button, Alice switches to the Lux view that displays a set of univariate and bivariate visualizations (bottom), including scatterplots, bar charts, and maps, showing an overview of the trends. Visualizations are organized into sets called *actions*, displayed as tabs. The one displayed currently is the `Geographic` action. By inspecting the `Correlation` tab in Figure 6.2 (not displayed here), she learns that there is a negative correlation between `AvrgLifeExpectancy` and `Inequality` (same chart as Figure 6.3 left); in other words, countries with higher levels of inequality also have a lower average life expectancy. She also examines the other tabs, which show the `Distribution` of quantitative attributes and the `Occurrence` of categorical attributes.

Steering analysis with intent. Next, Alice wants to investigate whether any country-level characteristics explain the observed negative correlation between inequality and life

[2]We refer to any operations that result in a dataframe in the output cell of a notebook as *printing the dataframe*, not the literal 'print (df)'.

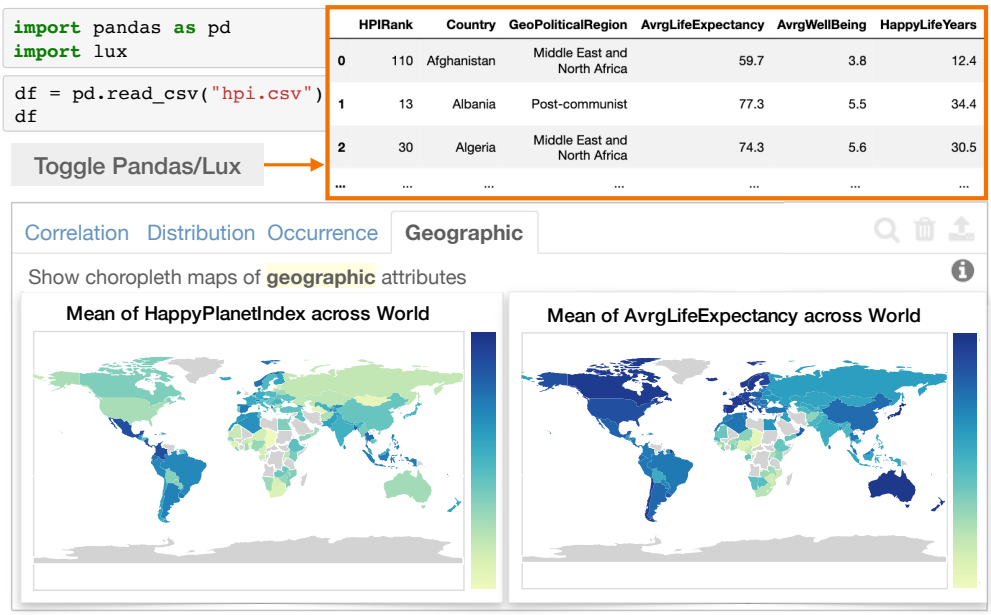


Figure 6.2: By printing out the dataframe, the default `pandas` tabular view is displayed (orange box) and users can toggle to browse through visualizations recommended by LUX.

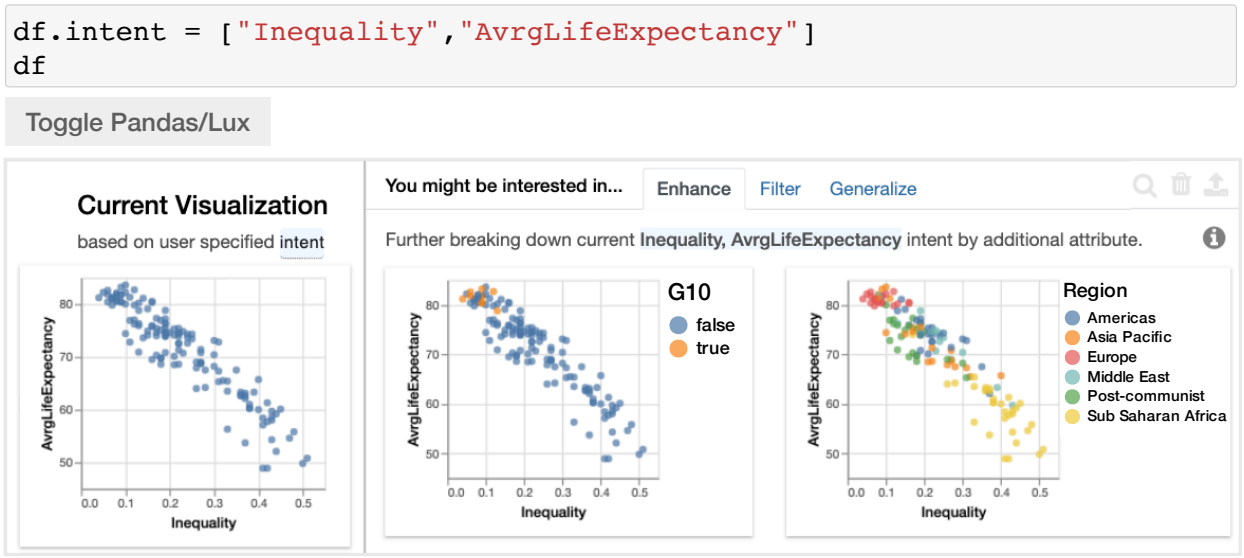


Figure 6.3: Alice sets the intent based on the attribute `AvrgLifeExpectancy` and `Inequality`, and LUX displays visualizations that are related to the intent.

expectancy. As in Figure 6.3, she specifies her analysis *intent* to LUX as: `df.intent = ["AvrgLifeExpectancy", "Inequality"]`. On printing the dataframe again, LUX employs the specified analysis intent to steer the recommendations towards what Alice might be

interested in. On the left, Alice sees the visualization based on her specified intent. On the right, Alice sees two sets of recommendations that add an additional attribute (`Enhance`) or add an additional filter (`Filter`) to her intent. By looking at the colored scatterplots in the `Enhance` action, she learns that most G10 industrialized countries (Figure 6.3 center) are on the upper left quadrant on the scatterplot (low inequality, high life expectancy). In the breakdown by `Region` (Figure 6.3 right), she finds countries in Sub-Saharan Africa (yellow points) tend to be on the bottom right, with lower life expectancy and higher inequality.

Seamless integration with cleaning and transformation. Alice is interested in how a country's development indicators relate to their early COVID-19 response as of March 11, 2020. To investigate this, she imports a new dataset that characterizes how strict a country's response is, via `stringency` [70], a number from 0-100, with 100 being the highest level of responses. As shown in Figure 6.4, (I) Alice loads and joins the newly-cleaned dataframe with the earlier HPI dataset. (II) When she sets the intent on `stringency`, she finds that China and Italy have the strictest measures (dark blue on map Figure 6.4 center). She also learns that the histogram of `stringency` is heavily right-skewed (Figure 6.4 left), revealing how many countries had low levels of early pandemic response. (III) To better discern country-level differences, Alice bins `stringency` values into a binary indicator, `stringency_level`, showing whether a country had `Low` or `High` levels of early response.

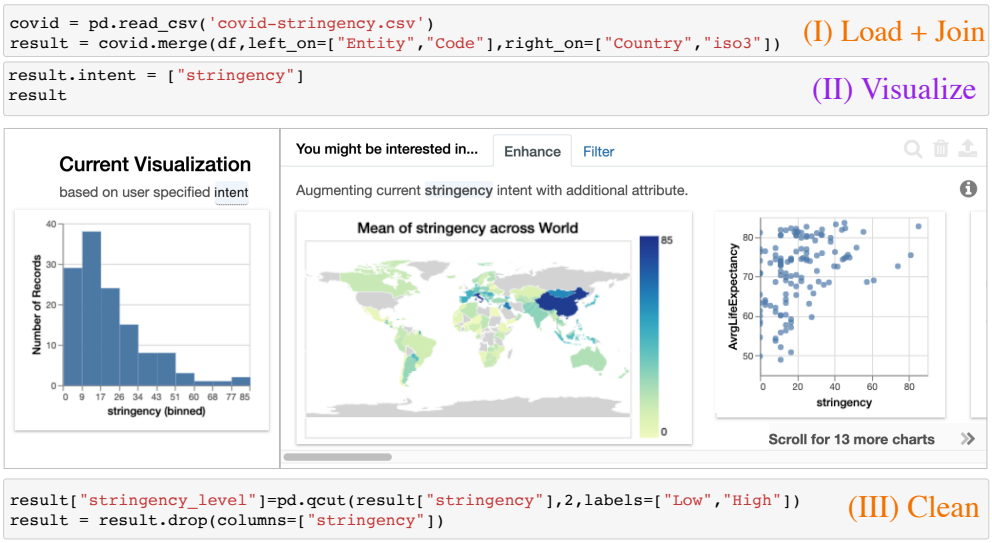


Figure 6.4: Tabular operations (orange, steps I & III) to load, clean, and transform the data, while visualizing with Lux (purple, step II).

With the modified dataframe, Alice revisits the negative correlation she observed previously by setting the intent as average life expectancy and inequality again. The resulting recommendations are similar to Figure 6.3, with one additional visualization showing the breakdown by `stringency_level` (Figure 6.5 right). Alice finds a strong separation showing how stricter countries (blue) corresponded to countries with higher life expectancy and

lower levels of inequality. This visualization indicates that these countries have a more well-developed public health infrastructure that promoted the early pandemic response. However, we observe three outliers (red arrow on Figure 6.5 right) that seem to defy this trend. When she filters the dataframe to learn more about these countries (Figure 6.5 left), she finds that these correspond to Afghanistan, Pakistan, and Rwanda—countries that were praised for their early pandemic response despite limited resources [3, 38, 19]. She clicks on the visualization in the Lux widget and the  button to export the visualization from the widget to a `Vis` object. Alice can access the exported `Vis` via the `df.exported` property and print it as code, following which she can tweak the plotting style before sharing Figure 6.5 (right) with her colleagues.

Overall, this example demonstrates the value of always-on visualization support within a dataframe workflow: the tight integration between Lux and dataframes enabled Alice to seamlessly perform data cleaning and transformation with her familiar `pandas` API and notebook environment.

```
result[(result["Inequality"]>0.35)&(result["stringency_level"]=="High")]
```

Toggle Pandas/Lux

	Entity	Code	Day	Country	GeoPoliticalRegion	AvrgLifeExpectancy
0	Afghanistan	AFG	2020-03-11	Afghanistan	Middle East and North Africa	59.7
87	Pakistan	PAK	2020-03-11	Pakistan	Asia Pacific	65.7
96	Rwanda	RWA	2020-03-11	Rwanda	Sub Saharan Africa	63.1

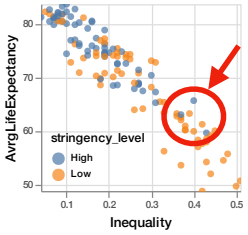


Figure 6.5: The scatterplot shows a separation between countries with high and low stringency in their COVID response. By filtering the dataframe (left), we see that Afghanistan, Pakistan, and Rwanda correspond to the three outliers (red boxed) that defies the trend.

Dataframe Interaction Framework

The demo illustrates the many flexible ways that users can interact with a dataframe to achieve their analytical goal. We outline this different interaction modalities in Figure 6.6. **Dataframe API** ①: Users can operate on the dataframe directly to perform any desired transformation or analysis. For example, Alice loaded the CSV, performed a join with another dataframe, and filtered to a data subset all via the familiar `pandas` dataframe API. **Intent** ②: Users can "attach" an intent to a dataframe to indicate aspects of the dataframe that they are interested in. The intent drives the actions and views that are generated in the levels above. In the demo, Alice indicated that she wants to learn more about `AvrgLifeExpectancy` and `Inequality`; Lux displayed visualizations related to the variable of interest. This intent is *virtual* in that as the dataframe changes, the intent can still be used to recompute visualizations on the updated dataframe; in some cases, this may result in a

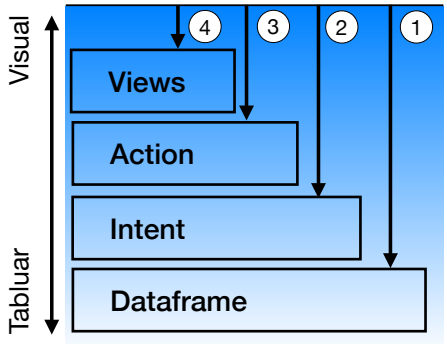


Figure 6.6: Visual dataframe interaction framework. ①-④ denotes four different modalities.

different visualization being computed, e.g., if the data type for a given intent column is modified. In Chapter 6.3, we describe a flexible intent language for specifying user interest.
Actions ③: Lux displays a default set of system-recommended actions that the users can interact with, e.g., `Enhance` or `Correlation`. Users can also register UDF-based actions for domain-specific needs. In either case, these actions are written in terms of the intent language but also leverage metadata and history. They instantiate a set of views displayed to users (described next).
Views ④: A *view* is an operationalization of intent when coupled with a specific dataframe instance. Users can directly create view(s) via `Vis`/`VisList` by specifying the intent applied to a given dataframe, resulting in one (or more) visualization(s). Actions instantiate one or more views—e.g., for a collection of visualizations (`VisList`) formed by plotting correlations across various attributes, each individual visualization (`Vis`) is an intent operating on a specific dataframe instance.
In this multi-tiered framework, changes in the bottom levels propagate to those above. Moreover, the settings at each level are retained across the session, so users can interact with the dataframe in a consistent and controllable manner. For example, when a user modifies the dataframe at the bottommost level, the same intent and actions are kept fixed and are used to update the views.

In addition to outlining different ways of interacting with a dataframe, the framework in Figure 6.6 from top to bottom spans a spectrum of interactions from visual-oriented to tabular-oriented ways of thinking, as exemplified by the orange and purple cells in Figure 6.3. During visual data exploration, some analytical tasks are better expressed as tabular operations (e.g., *convert the temperatures column to Fahrenheit*), while others are better expressed visually (e.g., *inspect correlation between sales and order volume* as a `Vis`). Many tasks are somewhere in between. Yet existing data querying languages and visualization grammars often create an artificial separation between the two, necessitating expensive "glue" code described earlier. By jointly considering and operating over visual and tabular aspects of dataframes, Lux supports a flexible and intuitive experience for interacting with data.

6.3 Intent Language Formalization

As shown in the framework in Figure 6.6, above the dataframe layer, users can specify their analysis intent, create custom actions, and generate desired views. This is all made possible through the intent language. The *intent language* is a lightweight, succinct way for users to programmatically and declaratively specify their high-level analysis interests and goals. Its capabilities are inspired by work on visualization query languages, such as ZQL [201] and CompassQL [239]. Unlike those languages, which are largely meant to be used internally within the corresponding interactive visual analytics systems (Zenvisage and Voyager) operating on static datasets, our intent language is tailored for programmatic specification coupled with a dynamically-evolving dataframe. In this section, we introduce the syntax of this intent language, and the underlying formal grammar. The grammar is decoupled from our specific implementation, which uses syntactic sugar for expressing the intent in a convenient Python-based API.

Intent Grammar

The *intent* grammar describes what the user is interested in within a dataframe. The intent is composed of one or more *clauses*, each of which is either an *axis* or a *filter* of interest.

$$\begin{aligned} \langle Intent \rangle &\rightarrow \langle Clause \rangle^{+} \\ \langle Clause \rangle &\rightarrow \langle Axis \rangle \;\mid\; \langle Filter \rangle \end{aligned} \tag{6.1}$$

An *axis* defines one or more attribute(s), mapped appropriately to a specific encoding or channel of the corresponding visualizations.

$$\langle Axis \rangle \rightarrow \langle attribute \rangle \langle channel \rangle^{?} \langle aggregation \rangle^{?} \langle bin\_size \rangle^{?} \tag{6.2}$$

For the axis, apart from the mandatory attribute(s), specified under $\langle attribute \rangle$, the remaining properties are optional—and can be automatically inferred. The axis construct is inspired by the grammar of graphics (GoG) [234] underlying visualization packages such as Vega-Lite [188] and ggplot [229]. Unlike GoG, our intent grammar doesn't require users to specify mark and channel properties. In GoG, users explicitly specify which encoding channel (e.g., x or y) each attribute is plotted—this is not necessary in our case.

Filters define a subset of data that the user is interested in. To specify a filter, the attribute being filtered, the operation, and the value, are required.

$$\langle Filter \rangle \rightarrow \langle attribute \rangle \;\; [=><\leq\geq\neq] \;\; \langle value \rangle \tag{6.3}$$

Consider the simple case when $\langle attribute \rangle$ refers to a single attribute and $\langle value \rangle$ refers to a single value in Equations 6.2 and 6.3; then, an intent with multiple clauses (axis or filter) represents a user preference to see each of the axis attributes visualized, for the subset of data corresponding to the conjunction of the filters.

In the more general case, $\langle attribute \rangle$ can correspond to a union of attributes, or a special wildcard value "?" (with an optional constraint to define the subset of attributes), while the $\langle value \rangle$ can refer to a union of values, or a special wildcard value "?".

$$\langle attribute \rangle \rightarrow \text{attribute} \cup \langle attribute \rangle^* \mid \text{"?"} \langle constraint \rangle^? \tag{6.4}$$

$$\langle value \rangle \rightarrow \text{value} \cup \langle value \rangle^* \quad \mid \quad \text{"?"} \tag{6.5}$$

The use of unions in either case (as well as "?" which implicitly is a union of all alternatives) admits a disjunction of options for the axis or filter clause. If there are $n_i \geq 1$ alternatives for the i^{th} clause, we can construct a collection of $n_1 \times n_2 \times \ldots \times n_k$ visualizations by taking the cross-product of alternatives per clause.

Specifying Intent

As described in Chapter 6.2, users can specify an intent indicating their analysis interests ②. Users can also create desired views by applying the intent to a specific dataframe ④. For the creation of actions ③, Lux makes use of the same view constructs as in ④ to enumerate one or more visualizations; however, the intent for these actions is often specified by Lux internally, instead of explicitly specified by the user.

Attaching an Intent to a Dataframe ②

Building on the grammar described above, within Lux, a `Clause` can specify one or more columns (i.e., `Axis`) or rows (i.e., `Filter`) of interest.
Query 1. To set `Age` and `Education` as columns of interest for a given dataframe `df`, one can state:

```
axis1 = lux.Clause(attribute="Age")
axis2 = lux.Clause(attribute="Education")
df.intent = [axis1,axis2]
```

Or one can also use the equivalent shortcut:

```
df.intent = ["Age", "Education"]
```

Once the intent is set, whenever `df` is printed, the Lux widget will use the intent to determine what visualizations to show to the user. Here, Lux would display visualizations related to attributes `Age` and `Education` from `df`. In the following, we will showcase the Lux intent syntax as part of `Vis` and `VisList`, but the syntax can also be used to simply set intent as in `df.intent` above.

Constructing a Single Intent-driven View ④

As mentioned in Chapter 6.2, a view operationalizes an intent on a dataframe. A view is specified using the `Vis` keyword within Lux, and results in `Vis` object that is rendered as a single visualization.
Query 2. Compare average `Age` across different `Education` levels.

```
axis1 = lux.Clause(attribute="Age")
axis2 = lux.Clause(attribute="Education")
Vis([axis1,axis2],df)
```

Query 2 is similar to Query 1, except that the intent is applied to the dataframe `df` to create a visualization via `Vis`, rather than changing the intent associated with the dataframe (to be used when the dataframe is eventually printed). Given that the intent involves one measure (`Age`) and one dimension (`Education`), Lux will display a bar chart. By default, average is the function used for aggregation.

Aggregation is one of three optional properties for `Axis` (Equation 6.2); others are channel and binning. If any of these are explicitly specified, they override Lux's defaults, as in the following query.

Query 3. Compare the variance of `MonthlyIncome` based on employee `Attrition`.

```
axis1 = lux.Clause("MonthlyIncome", aggregation=numpy.var)
axis2 = "Attrition"
Vis([axis1,axis2],df)
```

Finally, we can compose `Axis` and `Filter` together, as follows.

Query 4. Visualize the `Age`s for employees in the Sales `Department`.

```
axis = "MilesPerGal"
filter = "Department=Sales"
Vis([axis, filter],df)
```

Constructing Many Intent-driven Views ④

`VisList` represents a collection of visualizations, which can either be constructed indirectly by setting `df.intent` as in Chapter 6.3, or as an input intent to a `VisList`, as in the following query.

Query 5. Show how factors related to the rate of compensation differ for employees with different `EducationField`s.

```
rates = ["HourlyRate","DailyRate","MonthlyRate"]
VisList(["EducationField",rates],df)
```

Here, there is one `Vis` corresponding to `EducationField` combined with each of `HourlyRate`, `DailyRate`, and `MonthlyRate`. The wildcard character **`"?"`**, when used as part of an `Axis`, can be used to enumerate over *all* attributes in a dataframe; constraints may be used to restrict them to a certain type.

Query 6. Browse through relationships between any two quantitative columns in the dataframe.

```
any = lux.Clause("?",data_type = "quantitative")
VisList([any, any],df)
```

This `VisList` corresponds to the search space for the `Correlation` action; the `Correlation` action additionally ranks and sorts each `Vis` in the `VisList` based on their Pearson's correlation score.

`Filter` values can also be specified as a list or via wildcards across all possible values for a fixed filter attribute.
QUERY 7. Examine `Age` distributions across different `Countries`.

```
VisList([''Age'', ''Country''=''?''],df)
```

The generated `VisList` contains histograms of `Age`, one each for individuals where `Country` is `USA`, `Japan`, `Germany`, and so on.

6.4 Visual Recommendations

In the previous section, we have seen how users can either attach an intent to a dataframe, or this intent can be programmatically generated as part of LUX's recommendations. We discuss the latter in this section. In LUX, an *action* describes a set of visualization recommendations based on a predefined search space. LUX supports four major classes of actions. Metadata- and intent-based actions are akin to those used in past visualization recommendation systems [87, 240] as described in Chapter 5. We then introduce two novel classes based on the use of LUX within dataframe-based data science workflows, based on dataframe structure and history.

Metadata-based Recommendations. LUX maintains dataframe metadata, including attribute-level statistics such as min/max and cardinality to determine the semantic data type of each column and to automatically populate visualization settings. For example, based on data type, LUX can generate univariate and bivariate overviews. In Figure 6.2, Distribution, Occurrence, Temporal, and Geographical actions provide univariate overviews of columns, while the Correlation action provides bivariate overviews of all possible pairs of quantitative attributes, ranked based on Pearson's correlation.

Intent-based Recommendations. LUX displays recommendations based on the user-specified intent. On printing the dataframe, LUX displays a visualization based on the user-specified intent as in Figure 6.3, as the *Current Visualization*. In addition, LUX provides recommendations based on valuable next analysis steps starting from that visualization. For example, the `Enhance` action recommends visualizations formed by adding an additional attribute to the current visualization.

Structure-based recommendations. During the process of data science, data scientists often reshape their dataframes in ways that are more amenable to downstream analysis, discovery, and machine learning. Our formative study of existing notebooks indicates that the dataframe "structure" reveals strong signals for what the users subsequently visualize; LUX can use the same information to provide recommendations automatically:
Index-based visualizations: Dataframe indexes provide a natural way to order and label dataframe rows and columns. Indexes are typically created as a result of grouping and aggregation through operations such as `groupby`, `pivot`, `crosstab`. For any *pre-aggregated* dataframe (i.e., dataframes resulting from an aggregation operation), LUX creates visualizations by grouping the values either row or column-wise. For example, Figure 6.7 displays the result of a pivot operation, where each row is visualized as a time series line chart.

Series visualizations: Series are dataframes with a single column. Lux leverages the same dataframe visualization mechanism for Series, displaying univariate, metadata-based visualization, such as a bar chart for categorical and histogram for quantitative Series.

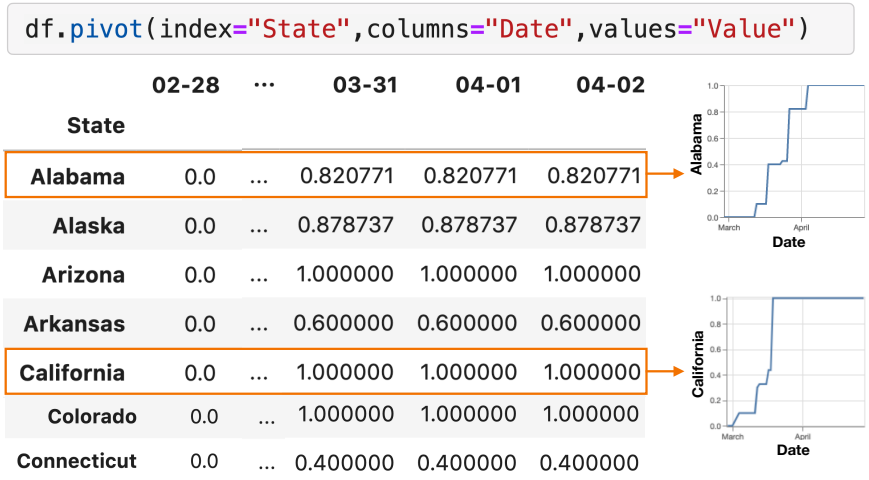


Figure 6.7: Row-wise index visualization displaying the normalized percentage of COVID-19 cases across different `States`.

History-based recommendations. Our formative study of notebooks also revealed that there is a strong connection between the operations performed by users and subsequent visualizations generated. For example, if the user cleaned up a particular column and renames it, it is likely that they would want to visualize the same column soon thereafter. Lux displays history-based recommendations based on whether the dataframe has been filtered or aggregated in its recent history. For example, when a filtering-based operation leads to a small dataframe (such as when a `head` or `tail` is performed), Lux visualizes the previous unfiltered dataframe since there are too few tuples for generating recommendations in the filtered dataframe. Lux also uses history to determine if an aggregation has been performed, helping identify the structure-based recommendations described earlier.

To collect this history, since Lux acts as a wrapper around `pandas` (described in the next section), we instrument each dataframe function and track each one with minimal overhead and store it as part of the dataframe, instead of requiring program analysis, which is prone to false positives [246]. Given that new dataframes or intermediate objects (e.g., `GroupBy`, `Series`) are often created when the user performs an operation, Lux propagates the history over to derived objects so that the history is not lost.

6.5 Lux System Description

Lux implements the visual dataframe framework described in Chapter 6.2, and is currently used by data scientists in real-world exploratory workloads.

Architecture

Lux employs a client-server model, leveraging computational notebooks as a frontend client. Lux currently supports Jupyter Notebooks, Jupyter Lab, Jupyter Hub, Microsoft Visual Studio Code, and Google Colab. The `ipywidgets` library is used for rendering an interactive HTML widget as the cell output. Once users import Lux, they can interact with a `LuxDataFrame` instead of a regular `pandas` dataframe. `LuxDataFrame` acts as a wrapper around `pandas`, and supports all existing `pandas` operations, while storing additional information, such as the intent, metadata, structure, and history, for generating visual recommendations.

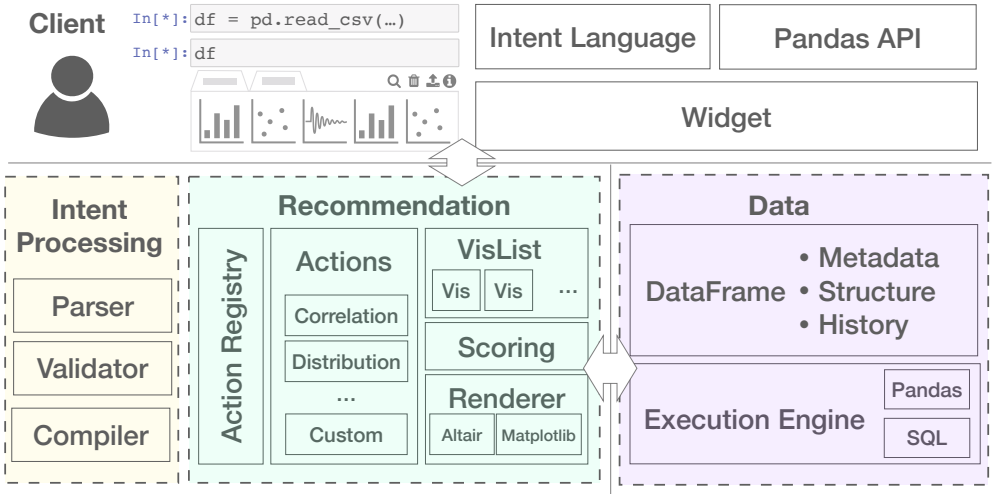


Figure 6.8: System architecture for Lux

As shown in Figure 6.8, the server side logic is largely separated into two distinct layers: 1) the *intent processing* layer is responsible for processing intent into executable instructions (Chapter 6.5) and 2) the *recommendation* layer is responsible for generating the displayed visualizations (Chapter 6.5). To generate the visualization recommendations, as well as compute metadata that is used in various stages, the execution engine performs the required data processing and optimization, either as a series of dataframe operations in `pandas` or equivalently in SQL queries executed in relational databases (Chapter 6.6).

The overall workflow is as follows:

> parse (6.5) → compute metadata (6.6) → validate (6.5) → compile (6.5) → select recs. (6.5) → compute recs. (6.6)

Metadata is memoized and only computed when needed. Finally, the system design is intended to be modular and extensible so that alternatives can be swapped in at different layers, e.g., Altair and Matplotlib visualization rendering libraries.

Intent Processing

Here, we discuss how Lux processes user intent to automatically infer missing details and determine appropriate visualization mappings. The intent processing layer parses, validates, and compiles the user's underspecified intent into complete specifications.

Parser and Validator

In Chapter 6.3, we saw how `Axis` and `Filter` can be be used to compose `Clause`s; the parser parses the user-inputted strings into an internal `Clause` representation. Subsequently, the validator checks for any inconsistencies between user-specified `Clause`s and the dataframe content. To do so, it leverages the dataframe's pre-computed metadata to verify the input intent. If the user's input does not align with the data present in the dataframe, the validator provides early warnings and suggests corrections to the input intent.

Compiler

During intent specification, users have the ability to omit certain optional details, making them *partial specifications*. Users also implicitly construct a collection of visualizations by using a union or wildcard character for `Axis` or `Filter`. Post validation, the compiler expands the `Clause`s into multiple visualizations and adds in defaults for the omitted details, making the `Clause`s complete. This transformation is performed in three steps.

1) Expand: If the input intent implicitly encodes multiple visualizations, the compiler "unrolls" these visualizations into individual `Vis` objects as a cross-product of the specified `Clause`s, leading to a `VisList` containing the resulting visualization specifications.

2) Lookup: For each `Vis` in the `VisList`, Lux populates the omitted details using the dataframe's pre-computed metadata. The compiler also removes any invalid visualizations generated that are either not supported in Lux or use ineffective encodings.

3) Infer: Finally, Lux infers the visualization encodings, including the marks, channels, and transforms (sort, aggregation, binning) required for generating the visualizations. The compiler implements rule-based heuristics drawn from best practices in design [59, 139].

After intent processing, Lux can now use the complete intent specification to either generate a `Vis` directly or generate a set of appropriate recommendations (described next).

Recommendation Generation

As described in the framework in Figure 6.6, actions organize collections of views into recommendations displayed to the users. The *action registry* in Lux keeps tracks of a list of possible actions that could be applicable for generating recommendations at any point in the analysis. On initialization, Lux registers a set of default actions (described in Chapter 6.4) applied to all dataframes. Users can also register their own custom actions programmatically by writing a Python-based UDF. The UDF generates a `VisList` of possible visualizations and optionally scores and ranks each `Vis`. The custom action is "triggered" whenever the dataframe satisfies the user-specified condition on when the action is applicable; Lux recommends visualizations based on the action.

6.6 Execution and Optimization

We now describe Lux's execution engine that is responsible for computing metadata and generating visualizations. We first describe the two major tasks performed by this execution engine. Then, we describe three optimizations aimed at speeding up these tasks.

Execution Engine

We now discuss how we compute metadata and visualizations.

Metadata Computation: The metadata computed includes attribute-level statistics and data types. The statistics include the list of unique values, cardinality, and min/max of the attribute. The unique values is used to determine the candidates generated by a wildcard for a filter on the column, or for validating filter input for the column, and for computing the cardinality. The cardinality information is used to determine the data type, while min/max is used for determining the limits on the visualization axes. Next, the execution engine infers the semantic data type based on the internal data type and cardinality information. Lux supports nominal, quantitative, geographic, and temporal data types. If the data type is misclassified, users can override the automatically-inferred data type.

Visualization Processing: After the user or system-specified intent has been transformed into one or more `Vis` objects with a complete specification, the execution engine translates each `Vis` to queries responsible for processing the data required for the visualizations. First, the engine applies any filters and retrieves relevant attributes. Next, the execution engine performs different visualization-specific operations depending on the mark type. For example, to process the data for a histogram, the engine bins an attribute into fixed-sized bins and performs a count aggregation for each bin. Table 6.1 summarizes the relational operations that corresponds to processing different visualization types.

Optimization

Next, we describe several optimizations aimed at minimizing the overhead incurred by Lux.

Intelligent workflow-based optimizations (wflow): During an analysis session, users constantly modify and operate on dataframes, which means that the metadata and associated recommendations can change throughout a session, especially during reshaping and type-modifying operations. Thus, unlike conventional visual analytics, where metadata can be computed upfront and stays fixed throughout, here, metadata needs to be constantly updated to ensure that recommendations are generated correctly. As a result, the computation associated with keeping the metadata "fresh" after each dataframe operation can be computationally expensive. We propose two techniques to reduce this overhead: 1) lazily compute the metadata and recommendations only when users explicitly print dataframes; 2) cache and reuse results later on in the session.

Since users often intersperse dataframe printing with several dataframe operations, it is likely that the computed metadata and recommendations would be outdated before users see the results. As a result, we can delay computation and compute the metadata and recommendations only after

the user has explicitly requested to print a dataframe. Each `LuxDataFrame` keeps track of how fresh the metadata and recommendations are and expires them when an operation makes a change to the dataframe. In particular, we leverage `pandas`'s internal functions that are triggered when:

- the dataframe is modified in place instead of returning a new dataframe, e.g., `df.dropna(inplace = True)`
- columns in the dataframe are updated, either through the bracket or dot notation, e.g., `df.Frac` or `df["Frac"] = df["value"]/100`
- the row or column labels are changed, e.g., `df.rename(columns={"val":"value"})`

Additionally, recommendations are expired when the intent is modified. On printing the dataframe, Lux recomputes the metadata as needed and generates the recommendations accordingly. This lazy strategy ensures no overhead on any non-print operations.

Lux further memoizes the metadata and recommendations so that any subsequent prints to an unmodified dataframe do not require recomputation. While this may sound like an overly specific use case, such operations are, in fact, very common. In dataframe sessions, users frequently perform "non-committal" operations that do not make changes to the dataframe to be used in subsequent analyses. These non-commital actions often involve printing dataframes as intermediate results to facilitate quick experimentation and debugging. As shown in In[3-5] in Figure 6.9, users may try to print out a column, perform grouping and aggregation, or print out descriptive summaries, all without modifying the original dataframe. In this case, when the user revisits the original dataframe, the memoized recommendations are immediately accessible to them.

```
# Modifying operation (Deferred Computation)                    [1]
df['review_date'] = pd.to_datetime(df["review_date"],format="%Y")
df.drop(columns=['Unnamed: 0'],inplace=True)

df # Recommendations computed for the first time here           [2]

df.groupby("company_location").mean() # Non-committal operation[3]

df.info() # Non-committal operation                             [4]

df # Memoized results, no extra work done!                     [5]
```

Figure 6.9: Example workflow demonstrating the applicability of WFLOW optimizations.

Approximate, early pruning of search space (prune): As described in Chapter 6.5, Lux searches through a `VisList` of candidates during recommendation generation phase to displays the most interesting visualizations to users. Dataframes that are wide or contain high-cardinality attributes can often result in large visualization search spaces. For instance, the Correlation action scales quadratically with the number of quantitative attributes in the dataframe. Given that Lux displays only the top-k ranked visualizations, there is a lot of time spent on generating the list of candidate visualizations that do not end up being displayed. With PRUNE, Lux first performs

Vis Type	Relational Operation
Scatterplot	Selection on 2 columns
Color Scatterplot	Selection on 3 columns
Line/Bar	Group-By aggregation
Colored Line/Bar	2D Group-By aggregation
Histogram	Binning + Count
Heatmap	2D Binning + Count
Color Heatmap	2D Binning + Count + Group-By aggregation

Table 6.1: Table summarizing the relational operations performed for processing different visualizations. Primary operations that accounts for the bulk of the visualization processing costs are listed.

a preliminary pass over `VisList` to approximate the score of each visualization and then proceeds to recompute the top-k selected visualizations in a second pass to process each of the displayed visualizations *exactly*.

Lux leverages a cached sample of the dataframe to approximate visualization scores (e.g., approximating correlation on a scatterplot by using only 30k rows on a dataframe with 1M rows), although other approximate query processing (AQP) methods could be applied.

Given that the prune optimization performs two passes over the `VisList` (first pass for pruning, followed by an exact recomputation for the top-k), the additional recomputation cost incurred can be higher than doing a single pass over the `VisList`. Therefore, this optimization should only be applied when the approximate savings are larger than the recomputation cost of the top k visualizations: $N \times t_{exact} \gg N \times t_{approx} + k \times t_{exact}$, where N represents the number of candidate visualizations, t_{exact} and t_{approx} are the cost of computing the exact and approximate scores, respectively. The cost of scoring a visualization is dominated by the relational operations for extracting the required visualization data (e.g., selecting two columns from a dataframe for scatterplots as shown in Table 6.1). Therefore, we calculate t_{exact} and t_{approx} using the estimated cost of these operations (described in Chapter 6.6).

Cost-based scheduling of actions (stream): We find that users generally spend an average of 28 seconds[3] skimming through the `pandas` table view before toggling to the Lux view. To ensure interactive responses, recommendation results can be streamed into the frontend widget as the computation for each action completes without having to wait for all of the actions to finish rendering. After compiling the visualizations for each action, we estimate the cost of the action as the sum of the visualization costs in the `VisList`, using the cost model describe next. This estimate is then used for scheduling the cheapest action to compute first, followed by computing the remaining in the background. In datasets where a few "laggard" actions dominate the overall recommendation generation (e.g., `Correlation` for a wide and highly quantitative dataset), the stream optimization provides users with early results and returns interactive control back to the user, instead of incurring a high wait time during their analysis session.

[3]Based on 514 collected logs of Lux usage, the time spent on the initial `pandas` table follows a long-tail distribution, with a median of 2.8 seconds and standard deviation of 183.4 seconds.

Cost Models for Visualization Types

We now discuss the latency cost estimation of different visualization types used for PRUNE and STREAM. The visualization cost is dominated by operations that LUX performs for each visualization type, as summarized in Table 6.1. We outline the functional form of the cost model and note that the coefficients A-D can be empirically fitted offline.

Scatterplots require selecting two columns (i.e., X/Y), so the cost of visualizing a scatterplot is linear in the number of points (N):

$$\text{cost(scatter)} = A \cdot N + B \tag{6.6}$$

In practice, the value of B can differ for different data types involved in the selected columns, since our analysis finds that a column's data types can significantly change the cost of selecting the column. Colored scatterplots select one additional color column, following Equation 6.6 but with different coefficients.

For bar charts, LUX essentially performs a group-by aggregation. The cost is therefore dependent on the number of unique values in group-by dimension (G_{bar}), i.e., the number of bars:

$$\text{cost(bar)} = A \cdot G_{bar} + B \cdot N + C \tag{6.7}$$

For colored bar charts, LUX performs group-by on both the bar dimension and the color attribute. Hence the dependence on the number of unique colors, G_{color}.

$$\text{cost(color bar)} = A \cdot G_{bar} + B \cdot G_{color} + C \cdot N + D \tag{6.8}$$

For histograms and heatmaps, LUX performs a binning of the data points into a number of bins or a two-dimension grid, followed by aggregation. For both chart types, the visualization cost is linear to the number of rows:

$$c\text{(histogram/heatmap)} = A \cdot N + B \tag{6.9}$$

where A, B are different for histograms and heatmaps.

The cost is largely independent of the number of bins or grid size because the number of total rows that the group-by aggregation needs to be processed is the same regardless of the number of buckets the data is divided among.[4] We also note that the heatmap's coefficients differs for different aggregation functions. For example, for heatmaps without color, the aggregation is based on the counts in each bin, and for quantitative colored heatmaps, the aggregation is an average of the data points that lie in the cell.

[4]Even though heatmaps and histograms can be an arbitrarily high resolution without affecting the processing speed in the execution engine, in practice, the bin resolution still needs to be capped at a reasonable limit, since increasing bin size impacts the rendering speed (i.e., more marks that needs to be drawn on the frontend).

6.7 Performance Evaluation

We evaluate Lux to measure its performance on large real-world datasets and notebook sessions, along the following dimensions:

- RQ1: What is the overall performance of Lux? Can Lux achieve interactive latency during a typical dataframe workflow?
- RQ2: What is the effect of the number of columns on Lux's performance?
- RQ3: How does the approximation-based PRUNE condition affect the quality of the recommendations relative to no approximation?

We focus on evaluating the interactive latency in this section; we describe the usability evaluation in the following section.

Data and Methodology

Data: We use two real-world datasets to evaluate the performance of Lux. The `Airbnb` dataset [42] contains 12 columns while the `Communities` [112] dataset contains 128 columns. For both datasets, we duplicated the dataset multiple times (up to 10M rows for `Airbnb` and up to 100k rows for `Communities`) to investigate the effects of scaling with the number of rows. After duplication, `Airbnb` exemplifies datasets with a moderate number of columns and a large number of rows, while `Communities` exemplifies those with a large number of columns. The upper limits on the two datasets cover around 98% of the datasets in the UCI repository [216].

Setup: All of our experiments were conducted on a Macbook Pro with 32GB of RAM and an Intel Core i9 processor running macOS 10.15.6. The experiments were run using Python 3.7.7, pandas 1.2.1, and a version of lux-api 0.2.3 adapted for purpose of the experiments. We used `papermill` [164] to programmatically execute each notebook cell. We set k for top k as 15 and apply PRUNE for any action where the number of visualizations exceeds k. For the sampling policy, we used cached random samples capped at 30k rows for approximating the visualization interestingness of dataframes over 30k rows (the choice of this parameter is justified in Chapter 6.7). For the runtimes reported, we exclude the frontend drawing time for each visualization given that it is constant and highly dependent on the chosen visualization library and frontend.

Conditions: Our experiment measures the time it takes to execute every cell in the notebook across five different conditions:

- no-opt: Baseline condition with no optimization applied, representing a naive implementation of Lux where the results are explicitly computed at the end of every cell involving a reference to the dataframe. This condition is akin to the naive implementation in most visualization recommendation systems, where the results are updated whenever the dataset is operated on.
- WFLOW: Condition with the WFLOW optimization applied.
- WFLOW + PRUNE: Condition with WFLOW and PRUNE applied.
- all-opt: Condition with WFLOW, PRUNE, and STREAM applied, representing the best achievable performance within Lux.

- `pandas`: Condition with only `pandas` and *without* using Lux, representing the raw performance of dataframe workflows without the benefits of always-on visualizations.

Overall workflow performance (RQ1)

To evaluate the overall performance of Lux with a dataframe-based workflow, we measured the runtime for executing an example notebook involving `pandas`.

Workload: The workload is based on publicly available notebooks on Kaggle for `Airbnb` and `Communities`. These notebooks follow a typical exploratory analysis of a dataframe that includes loading, transformation, cleaning, computing statistics, and machine learning. We modified these notebooks to print out dataframes and series at various points in the notebook akin to what a user would typically do for validating the results of operations. In addition, we label each cell in the notebook as either a print of a dataframe, print of a series, or neither (i.e., any non-Lux Python command) to separately measure the runtime for different cell types. Table 6.2 shows the breakdown of the two notebook workloads by different cell types. We define *overhead* as the difference in runtime between the all-opt and `pandas` condition, i.e., the additional time required to support always-on visualizations via Lux.

	Airbnb		**Communities**		
	N	overhead [s]	N	overhead [s]	Distr.
Print df	14	21.18	14	1.41	
Print Series	7	0.61	4	0.07	
Non-Lux	17	0	25	0	

Table 6.2: Table reports the number of cells for each type (N), the additional time incurred on top of `pandas` for 10M `Airbnb` and 100k `Communities` (overhead), and the relative shape of the runtime distribution similar to Figure 6.11,6.10, (Distr.).

Overall runtime: To understand the overall performance of Lux on dataframes with varying sizes, we varied the dataframe size from 10k to 10M rows. Figure 6.11 displays the overall runtime averaged over all cells in the notebook. We find that the best achievable performance with Lux led to significant speedup with up to 11X improvement in overall runtime for the `Airbnb` dataset (and up to 345X for `Communities`) compared to the no-optimization baseline.

Printing Dataframes and Series: We measure the performance of each cell that prints a dataframe or series to understand the overheads associated with Lux. Figure 6.10 shows the average time it takes for printing a dataframe for `Airbnb` and `Communities`. In particular, the overhead of Lux for each print can be determined by comparing against the cost for a print in `pandas`. When the dataframe contains fewer than 1M rows for `Airbnb`, each print incurs no more than 2 seconds in addition to `pandas` (in the 10M case, each print incurred an overhead of 21 seconds). For `Communities`, the overhead was no more than 1.5 seconds.

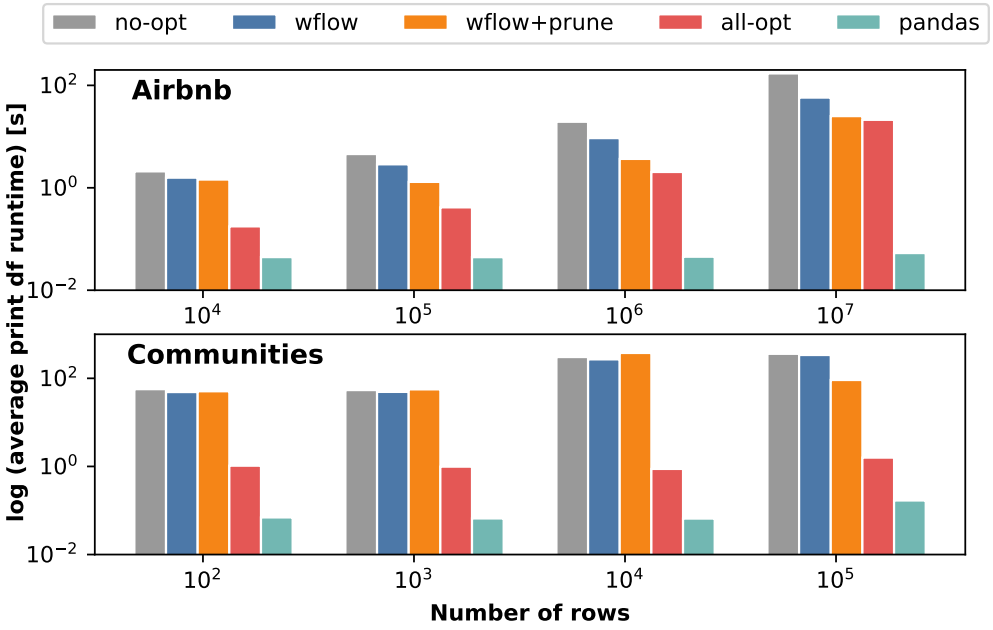


Figure 6.10: Average time for printing a single dataframe for different dataframe size and conditions.

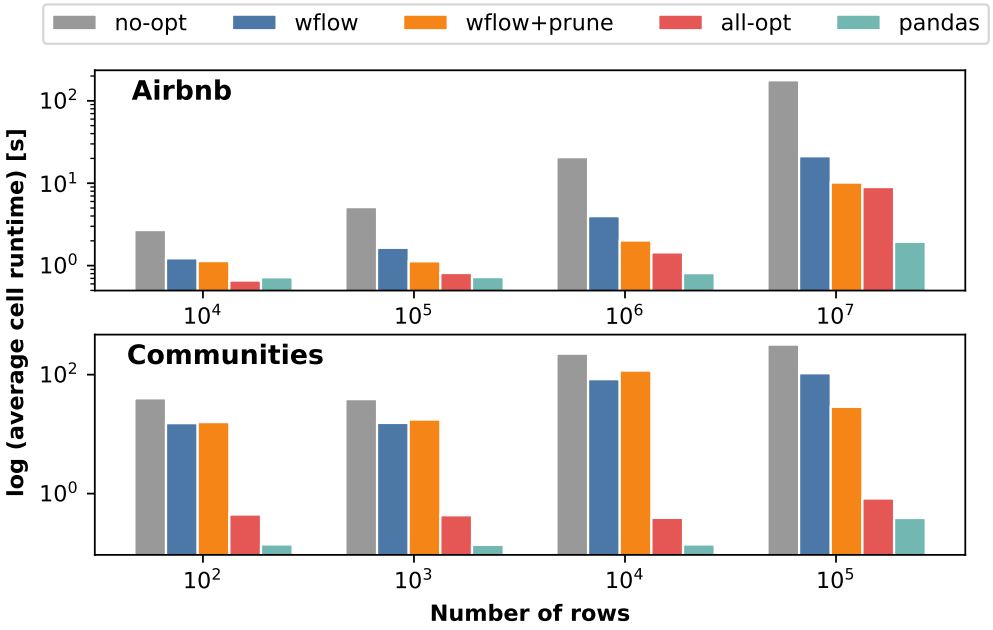


Figure 6.11: Average runtime of a notebook cell across the workload for different dataframe size and conditions.

As shown in the sparkline visualization in Table 6.2 row 2, the performance for printing series follows the same pattern as that of the dataframe. However, since series only involves a single column, it effectively avoids the costly procedure of traversing through a large search space. The overhead on top of `pandas` is no more than 1 second for each series print even on the largest datasets.

Non-Lux operations: Across all conditions except the baseline, the runtime for non-LUX operations (Table 6.2 row 3) is the same—demonstrating how LUX incurs zero overhead on any Python operations in a notebook session. When compared against the baseline, LUX is over 100X faster for 100k `Airbnb` and over 650X faster for 10M `Communities`. The performance improvement for non-LUX operations demonstrates how WFLOW's lazy evaluation strategy avoids unnecessary computation.

Effect of dataframe width (RQ2)

We investigate how the performance of LUX varies depending on the number of columns in the dataframe. To understand the effect of the width of a dataframe (w), we measure the processing time for a single dataframe print (after the metadata has already been precomputed). Given the dependence of actions on data types, we leverage a synthetic dataset to vary the number of columns in the dataframe, while fixing the proportion of data types. The simulated dataframe contains 100k rows with 78% quantitative columns, 20% nominal columns, and 2% as temporal. Across the quantitative columns, half of the columns are integers, while the other half are floats. For the nominal columns, we generate columns of strings with varying cardinalities chosen based on a geometric series between 1 to 10000.

Figure 6.12 left shows the runtime for different dataframe widths[5]. We note that the blue no-opt curve (power=2.53) scales exponentially with the number of columns. By applying the PRUNE and STREAM optimizations (red), LUX effectively lowers the cost of printing a dataframe by bringing the runtime closer to linear (power=1.07).

Effect on recommendation accuracy (RQ3)

To understand how the approximation-based PRUNE condition affects the recommended results, we experimented with different fractional sizes of the dataframe to be used in the sample and its effect on the recommendation ranking. We compared the list of recommendations generated with and without the optimization applied. We computed Recall@15 of the top k results against the ground truth rankings. We chose recall, instead of other rank position-dependent measures, because the top-k visualizations are computed exactly and re-ranked after selection, so the metric only needs to capture how accurately the top-k visualizations are retrieved.

The recall curves in Figure 6.12 right shows that for most actions 10% (5k rows) is required in the sample for achieving over 90% accuracy. For the 100k `Airbnb` dataset, the sample requirement is around 20-40% (i.e., 20-40k rows). As a result, we chose the sampling cap in our experiment to be 30k rows to reach an average of 90.5% on `Airbnb` dataset and near perfect ($\geq$ 95%) on

[5]We note that the no-opt condition is the same as WFLOW in this case since we are only measuring a single print dataframe cell.

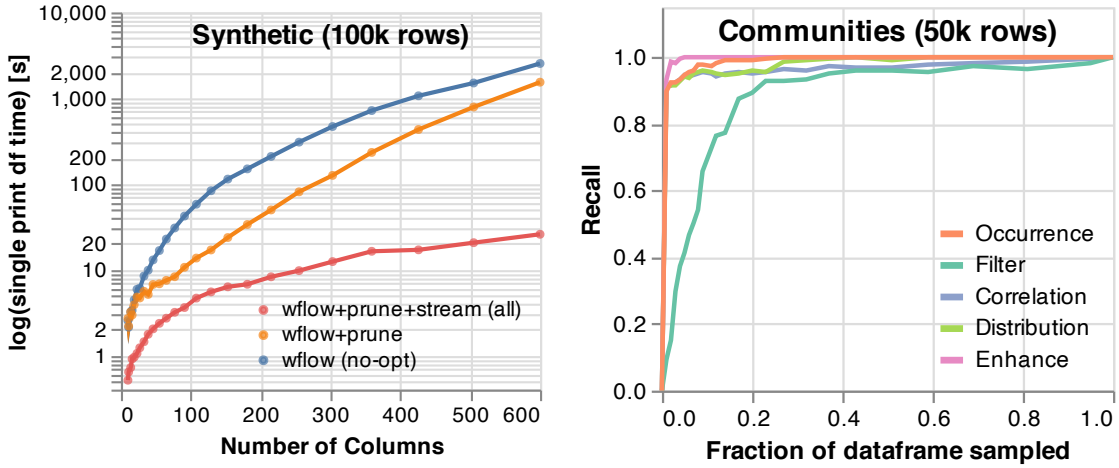


Figure 6.12: Left: Time spent for a single dataframe print varying the number of columns in a synthetic dataframe. Right: Recall curve for different actions varying fractional samples of rows in the 50k Communities dataset.

Communities. Compared to other actions, since Filter (light green in Figure 6.12 right) enumerates over data subsets, it requires more samples to ensure enough data points per stratum to achieve the same accuracy.

6.8 Assessments with Users

To evaluate the effectiveness of Lux in typical data science workflows, we performed two usability studies: a controlled study with new users and a field study with existing users of Lux.

First-use Controlled Study

We performed a study to understand participants' initial impressions of Lux and whether they are able to use Lux effectively in a controlled setting. This study was performed remotely from October to November 2020 using lux-api 0.2.0. This study was part of a 90-minute interactive session where participants were first introduced to the basics of Lux and guided through a set of hands-on exercises on how to use Lux. The study was conducted with two focus groups: the first was a bootcamp for industry data practitioners (N=20) and the second was an online lecture for students in a graduate-level data visualization course (N=15). Both groups engaged in the same set of instructions and tasks. The instructions and tasks were made available to participants via a web link to a live Jupyter notebook. Participants were led through three notebooks in sequence. Each notebook contained examples and exercises covering the key concepts in Lux using three datasets (College [217], Happy Planet Index [72], and Olympics [1]). Interactions on the Lux widget and actions performed on the notebook were logged via a custom extension [162]. The session concluded with a short survey documenting participants' experience. Due to the remote and unsupervised study setting, not all participants submitted survey responses or performed notebook operations that were logged.

Study Findings. We collected 16 survey responses (6 from bootcamp, 10 from lecture). The results were thematically coded and classified by one of the authors. In response to background questions regarding the existing exploration workflows of the participants, their concerns echoed the pain points that Lux aims to address, including difficulty in determining the "right" visualization to plot (5/16), modifying and iterating on visualizations (4/16), and determining where to begin an analysis (4/16). When asked to comment on aspects of Lux that they liked, 9/16 participants cited how the ability to print and visualize dataframes was the most useful. Participants also noted how the integration of Lux with their data science workflow was seamless and intuitive. When asked to comment on aspects of Lux that they found challenging, 8/16 participants described unfamiliarity and the learning curve associated with the intent syntax. When asked about what they would like to see most in future versions, participants were most interested in improving Lux's latency on large datasets (12/16)[6], followed by support for a wider and more useful set of recommendations (8/16) and making the intent language more customizable (7/16). At the end of the survey, 13/16 participants signed up for follow-ups and expressed interest in continuing to use Lux.

To evaluate whether participants were able to accomplish controlled tasks with Lux, we collected 23 unique logs of the participants' interaction with the notebooks. We qualitatively graded how well participants performed across the three exercises. The task success rate for the three exercises was 68% (for composing an intent indicating multiple views), 87% (for specifying a desired `Vis`), and 71% (for creating a `VisList`). By inspecting the trace of attempts, on average participants were able to obtain the first successful answer within their first five tries. Participants' most common mistakes involved confusion around the syntax for specifying multiple visualizations via union. Finally, participants were encouraged to try out one of the provided datasets for open-ended exploration. While participants successfully used Lux to print and visualize their dataframes, due to the setting and time constraints, their interactions with Lux were brief. The limited insight into how users performs open-ended exploration with Lux motivated the need for the following study.

Field Study Interviews

To understand how Lux is used in real-world analytical workflows, from December 2020 to January 2021, we conducted semi-structured interviews with participants who used Lux in their data science work. We interviewed two industry data scientists in an insurance (P1) and retail company (P3), and a researcher in education (P2). Given that participants had extended exposure to Lux, our questions largely focused on understanding how Lux fits into their existing workflows. Before the interviews, participants used Lux over the span of 1-2 months in their professional data science work. Their usage frequency varied: P1 used Lux daily, P2 used Lux once every one or two weeks, P3 used Lux around ten times in total. Unlike the first-use study where participants were led through instructions dedicated to how to create `Vis` and `VisList`, field study participants learned how to use Lux on their own through tutorials and documentation on our website. We performed a walk-through of real-world notebooks in which participants had used Lux.

[6]We note that the study was performed using the latest version of Lux at that point, which did not include many of the scalability improvements described earlier (WFLOW was included, but not STREAM and PRUNE).

To what extent do you find the following functionalities in Lux useful?	P1	P2	P3
Printing dataframe and inspecting recommended visualizations	Very useful	Very useful	Extremely Useful
Expressing analysis intent to steer recommendations	Extremely Useful	Extremely Useful	Very useful
Specifying visualization of interest via `Vis`	Moderately useful	N/A (Did not use)	Very useful
Specifying collections of visualizations of interest via `VisList`	Very useful	N/A (Did not use)	Very useful
Exporting selected visualizations from Jupyter widget	Very useful	Extremely Useful	N/A (Did not use)
Lux makes it easier to ...	**P1**	**P2**	**P3**
Visualize my data across different stages in the data science workflow	Agree	Strongly agree	Agree
Plot a single visualization that I have in mind	Strongly agree	Strongly agree	Strongly agree
Identify what aspects of data I should visualize	Strongly agree	Agree	Agree
Determine what to do next in my exploration	Agree	Somewhat agree	Neutral

Table 6.3: Table of Likert scale ratings across the three field study participants.

Study Findings. All three participants expressed that understanding their data was a challenge during exploration. In fact, two of the participants have developed their own homegrown solutions for past projects (echoing findings from Alspaugh et al. [5]), ranging from for loops across `matplotlib` charts in notebooks to VBA scripts that generate plots in Excel. In their existing workflows, P1 and P2 visualized their data programmatically via `matplotlib`, while P3 largely on Tableau's GUI for creating visualizations.

On dataframe visualizations: All three participants expressed that they appreciated how the automatic visualizations provided by Lux afforded them quick insight into their dataframes without the need for code. P2 typically examines over 100 columns of data as part of an educational course survey, and stated that Lux sped up the amount of time for EDA by at least two-fold: *" it really helps speed up my exploratory analysis. If not, it will take me forever to go through these many variables."* When asked about the scenarios for which they would toggle to the Lux view versus the default `pandas` table, most participants preferred seeing the Lux view for the purposes of EDA. Participants described how they only use the `pandas` table to quickly check if "*the data looks okay*" (P1) and rarely toggle back to it unless they observe anomalous trends in the visualizations. During the study, P2 adopted a workflow where they sampled a single row to display the `pandas` table in one notebook cell, then printed the Lux view in the cell below to check that the data falls in the expected ranges as displayed in the visualizations.

On dataframe intents: Participants indicated that the concept of intent was an intuitive way for steering the course of their analysis. P1 and P2 leveraged intent as a way of systematically exploring groups of variables they were interested in. To investigate their research questions, P2 listed groups of independent and dependent variables as their intent to explore each group one at a time. P1 and P3 used intent as a way of exploring predictive variables of interest, such as whether a customer purchased accessories alongside their orders, to help inform feature engineering for downstream machine learning. However, challenges in specification sometimes prevented them from making use of intent fully. In particular, P2 and P3 were both interested in exploring alternative data subsets for an attribute of interest (a query that is expressible in Lux's intent language); however, they were unaware that they could specify filter intent with wildcards. Improving the API for intent specification remains an important direction for future work.

On custom actions: Participants noted how the default Lux actions largely covered the basic sets of analyses that they would typically perform on their own. While most participants were unaware that Lux supported the ability to create custom actions, during various points in the interview, they described additional actions that they would find useful. For example, P3 described how they

wanted to create a custom action that lists the top ten dataframe columns with the most influence over a desired predictive variable. Other participants described actions that are similar to the default Lux actions, but with a different ranking. For example, P2 was interested in categorical variables that involved bar charts that looked very even, since that means that it has a closer-to-equal likelihood of being in either categories, so the trend is potentially interesting.

On user-specified views: Somewhat surprisingly, while `Vis` and `VisList` were highly favored in the first-use study, they were rarely brought up in the field study interviews. Possible explanations for their limited use include the unfamiliarity with these concepts and their usage of Lux in conjunction with other visualization tools. All participants used an existing visualization tool (matplolib or Tableau) while exploring their data with Lux. As a result, they simply defaulted to their familiar tools for specific visualizations when they knew exactly what to plot. To fully leverage `Vis` and `VisList` in their work, participants often asked for ways to extend or customize the visualization type for a user-specified view. For example, P3 explained how market share data was best visualized as a top-k pie chart, while P2 was interested in examining overlaid histogram distribution of different measures for binary variables, such as whether or not a course was open-ended. These findings indicate that increased flexibility in the intent language could afford the familiar visualization capabilities for users when creating specified views.

Usage of Lux *in data science workflows:* All three participants described using Lux explicitly in the exploration stage after data loading and cleaning, but before advanced analysis or modeling. P1 and P2 used Lux in conjunction with custom `matplotlib` code that they repurposed for their analysis. When asked why participants did not print the dataframe for visualizations during the data transformation and cleaning phase, P1 and P3 answered that since the dataframe prints resulted in a few seconds of latency, they were hesitant to do it until they were ready to *"chuck in [their] data and get the charts out"* (P3). Participants also described how Lux needed to be more robust in visualizing dirty or ill-formatted data.

Post-interview survey results: Table 6.3 details participants' Likert scale ratings of the functionalities and benefits of Lux. Participants found the use of intent and the ability to print and visualize dataframes to be the most useful features. Participants reported that they either did not make use of `Vis`, `VisList`, and export functionalities or found them to be less useful. Participants described how Lux made it easier to plot a single visualization that they had in mind, identify aspects of data they should visualize, perform visualization across different stages of the data science workflow, and determine what to do next. The average System Usability Scale (SUS) [28] score across participants is 70/100. All three participants were interested in continuing to use Lux in their data science work.

Limitations and future work: The discrepancy between the usage of views in the first-use and field study indicates that even though `Vis` and `VisList` are could be learned with a focused tutorial and exercise, they are not as discoverable and easy-to-use as the dataframe visualizations. Despite the enthusiasm around Lux, we find participants still attached to their existing visualization tool for this functionality. They shared concerns around customizability and the inability to express their desired visualizations in Lux, pointing to the need for improving the flexibility of the intent language. Given that our participants often work with data in commercial cloud data warehouses, it is not only important for Lux to speed up processing for recommendations, but also account for data that doesn't fit in memory in the future.

6.9 Conclusion

Building on lessons learned from our foray into designing various visual exploration assistants, this chapter describes Lux, a general-purpose, mixed-modality visual exploration assistant that provides always-on visualizations for dataframes within a computational notebook. Lux is a lightweight wrapper around dataframes that reduces the barrier of visualizing data and guides users in the process of determining next steps for analysis. To support automated visualizations, dataframes can be enriched with information directly or indirectly from the user, such as the user's intent and history, as well as the dataframe's structural information and metadata. We introduce a high-level query language for specifying a user's analysis interest, for experimenting with quick visualizations on-demand, and for working with large collections of visualizations. We develop and evaluate effective optimization strategies that intelligently cache and maintain metadata and recommendations. Lux's initial adoption and success points to the potential for general-purpose, mixed-modality visual exploration assistants for accelerating insight exploration and discovery.

Chapter 7

Conclusion

This dissertation investigates the design space of mixed-initiative assistants for visual data exploration. We developed research prototypes and systems to study how visual exploration assistants aid analysts to visualize and explore their data more effectively. In this chapter, we summarize the key findings of this thesis and outline future directions for visual exploration assistants.

7.1 Summary of Findings

We revisit the overview of the thesis as illustrated in Table 2.1 from Section 2.3 in Chapter 2 to summarize our findings (shown here as Table 7.1). Our findings largely correspond to our design space exploration across the two dimensions (labelled ①-② in Table 7.1) and our success with Lux as a general-purpose, mixed-modality visual exploration assistant (③).

	① **Analytical Task Supported**	② **Interface Modality**
Vispilot (Chapter 3)	Bar Chart Comparison	GUI
Zenvisage (Chapter 4)	Line Chart Search	GUI
Frontier (Chapter 5)	General-Purpose	GUI
③ **Lux (Chapter 6)**	General-Purpose	Mixed GUI/code

Table 7.1: Organization of the design space of visual exploration assistants explored in this dissertation, based on the type of analytical task supported and interface modality.

Finding ①: Visual exploration assistants help analysts perform visual data exploration more effectively across different analysis tasks.

In this dissertation, we presented both single and general-purpose visual exploration assistants that help analysts across different analysis tasks, from accelerating drill-down analysis and pattern search to suggesting a set of potential next steps for exploration.

In Chapter 3, VisPilot guides analysts towards informative and interesting subsets of the data to overcome the challenges of bar chart comparisons during drill-down analysis. Our evalua-

tion study shows that VisPilot helps analysts discover interesting visualizations, predict related visualizations, and rank the importance of attributes across the whole dataset.

In Chapter 4, Zenvisage++ helps analysts accelerate line chart pattern search to avoid the painstaking process of manually browsing through a considerable number of visualizations one at a time. Zenvisage++ support a core set of visual querying capabilities to help analysts answer common questions that arise during line chart exploration, including "what patterns look similar to my selected pattern?" and "what are the most common patterns in my dataset?".

While Zenvisage++ and VisPilot help analysts accelerate a single class of visual exploration tasks, real-world inquiries often consist of a diverse range of analytical tasks, each requiring different types of visualization recommendations and guidance. Chapter 5 explores how recommendation categories supporting different analytical tasks can be integrated within a general-purpose visual exploration assistant. In Chapter 6, Lux leverages an always-on visualization framework to provide general-purpose visual exploration assistance in a computational notebook. Lux offers natural and intuitive ways of looking at the dataframe beyond the tabular view, thereby reducing the barrier of writing code to visualize and explore data.

Finding ②: Visual exploration assistants can be situated across different interface modalities, including GUI-based, programmatic, or mixed, to facilitate a seamless workflow.

One of the novel contributions of this dissertation is considering not only *what* types of recommendations to suggest, but also *how* visual exploration assistants fit into the analyst's workflow. Workflow considerations are crucial, especially for the purpose of tool adoption, but as we saw in Chapter 4, they are often one of the most overlooked aspects in the design of visual exploration assistants.

Most existing visual exploration assistants we surveyed in Chapter 5.2 exist in GUI-based interfaces that are siloed from other tasks that an analyst may perform, such as data cleaning or model building. The cost of having to context-switch between different interfaces and modalities presents a barrier to exploration. Likewise, the systems described in Chapter 3-5 all operated on small, relatively clean datasets in a single CSV file with a relatively simple schema. From a tool adoption perspective, this means that there is often only a narrow set of scenarios where these tools are applicable. As a result, these GUI-based visual exploration assistants were limited in that they can only provide surface-level overviews of the datasets at the exploratory stages, but rarely lead to deep understanding of data or unexpected insights.

To address this challenge, one naive solution is simply to add all possible functionalities that a user may be interested in within a single GUI-based visual exploration assistant. We attempted this effort by adding interactive filtering, class creation, and smoothing capabilities alongside the core visual querying capabilities in Zenvisage++, but quickly realized that this effort was not sustainable due to the diversity of tasks and tools that analysts used in their workflow. Rather than trying to build a monolithic, feature-rich tool, we opted for a design that was as lightweight and minimal as possible when designing Lux, so that visual exploration can be seamless, transparent, and familiar to users. While Lux is one of the first visual exploration assistants designed to fit into a programmatic workflow, it also supports a GUI and interactions for working with the recommended visualizations. The seamless and fluid transition between GUI-based and programmatic workflows

in Lux demonstrates how there are indeed benefits for mixed-modality visual exploration assistants. Future work that integrates visual exploration assistants with other interface modalities (such as natural language) must carefully consider how the workflows afforded by these modalities fit in with the specific capabilities of the visual exploration assistant.

Finding ③: Flexibility across both task types and modalities in visual exploration assistants reduces workflow friction and encourages exploration.

As we have seen in this dissertation, both task type and modality are important design considerations for visual exploration assistants. Our experience working with real-world analysts revealed how removing friction in the analysis workflow is not just a quality-of-life improvement on top of an analyst's existing workflow, but often *determines* the degree to which an analyst explores their data. In our Zenvisage++ study, analysts described how they often have to switch between parameter specification, code execution, and visualization comparisons. Their segmented workflow made them more hesitant to visualize the results and experiment with different parameters. On the other hand, Zenvisage++ supported multiple sensemaking processes in a single interactive window, empowering analysts to significantly speed up their collaborative analysis process by interactively adjusting parameters and visually query on the fly. Likewise, in our Frontier study, manual specification appeared to be overwhelming or even daunting to participants with limited visualization experience. Analysts opted for the low-effort recommendation alternative to proceed with their analysis, leading them to insights that they would not have otherwise know to look for on their own.

Lux distills and improves upon the lessons we learned from our experience in building VisPilot, Zenvisage++, and Frontier. In particular, Lux is the first visual exploration assistant that is both general-purpose and mixed modality. Lux's success can be attributed to these design characteristics: it not only supports a diverse range of analytical tasks through the different types of recommendations, but also encourages expressive interactions through graphical and programmatic means. Given the success in adoption of Lux, we have learned that when designed appropriately, visual exploration assistants remove the friction in exploration and inspire new directions of investigation. Moreover, visual exploration assistants encourage analysts to rapidly experiment with large numbers of hypotheses interactively — a crucial step in the agile, creative process of discovering actionable insights.

7.2 Future Research Directions

Scalable Optimizations for Interactive Feedback

One of the challenges for visualization recommendations systems is the high computational cost associated with searching over a large search space of possible visualizations [221, 220, 39]. To ensure interactive feedback, we explored existing techniques from OLAP and approximate query processing in Chapter 6 to provide always-on visualizations for Lux. Optimizations employed by Lux merely scratch the surface of those we envision would be necessary for truly scalable visual exploration assistants.

Given that visual exploration assistants aim to address the problem of information overload by prioritizing important insights to display to users, the search component of the problem makes them more amenable to optimizations compared to standard query processing for a single visualization. For example, most recommendation categories employ a local ranking objective that involves scoring visualizations one at a time. Instead, data processing for all of the candidate visualizations can be performed in parallel. Likewise, given that visual exploration assistants often display only the top-k results to analysts, a scalable visual exploration assistant can prune the visualization search space before spending expensive compute cycles on visualizations that will not be shown. For approximate processing-based optimizations, future studies on the effects of these optimizations on the usefulness of suggested results are required.

Across recommendation categories, there are often overlapping parts of the search space that could benefit from materialization and reuse-based optimizations. For instance, when computing the deviation between the overall and filtered visualization in the `Filter` action (following the taxonomy in Section 5.2 in Chapter 5), the data corresponding to the overall visualization could be cached and reused. Moreover, for expensive filters, each column of data only needs to be fetched once to compose all the pairwise scatterplots in `Correlation`, and subsequently reused to compute aggregate visualizations for `Distribution` and `Occurrence`. These materialization techniques can be trivially applied to workflows where the data is not changing significantly, e.g., in Frontier's case. In scenarios where the data might be evolving constantly, there are optimization strategies that could partially recompute or cache parts of the precomputed data or metadata. For example, in the dataframe workflow described in Chapter 6, Lux could track the history of dataframe commands and identify read-only dataframe commands where the expensive recomputation of the metadata and recommendations is not required. Moreover, commands like `df.describe()` or `df.info()` generate an intermediate output dataframe that is rarely operated on again, so the results of the underlying dataframe can be cached and maintained. Even for simple update commands (e.g., when a column is dropped from a dataframe), there is opportunity to partially update previously recommended results by making slight modifications to the resulting metadata and recommendations (e.g., remove results involving the dropped column), instead of having to recompute everything from scratch.

Context-aware visual exploration assistants

Most programming and query languages require analysts to manually specify a series of operational steps or problem specifications that address their desired need. On the other hand, visual exploration assistants offer analysts the ability to specify vague, underspecified questions at a high level, as demonstrated with the intent language in Chapter 6. However, due to limitations in the expressiveness of existing specification languages, current visual exploration assistants tend to only support a narrow subset of these analytical inquiries; as a result, there is a range of possible intents that is not currently covered by these languages. In particular, one relatively underexplored direction is leveraging user's existing analytical context to inform and discover useful and relevant insights.

Analytical context captures the analyst's current state of data exploration. It is a generalization of user intent that not only includes the intent explicitly specified by the user, but also any contextual information implicitly inferred by the system. For example, task-level information can be embedded into the analytical context to inform visual exploration assistants how to appropri-

ately select relevant insights that align with the user's current state of data exploration. The visual exploration assistant can infer, based on a sequence of past operations, that an analyst may be interested in looking for specific outliers or anomalies to clean data, instead of generally browsing through distributions across the dataset.

Implicit, context-aware visual exploration assistants are especially important in analysis workflows where the analysis goals and tasks are highly exploratory and continuously evolving, such as when working with dataframes in computational notebooks or for conversational data analysis [123]. By developing a mechanism to capture the continuously changing context available, the visual exploration assistants can deliver a rich data exploration experience that is constantly up-to-date with the analyst's current state of data exploration, and surface insights relevant to the current context. While Section 6.4 in Chapter 6 outlines a limited use case for leveraging dataframe history for recommendations, many research challenges remain. First, the system designer must determine what aspect of the context to record, and the appropriate level of granularity to capture. For instance, if the context is too coarse-grained (e.g., the system detects that user is interested in getting an overview of the data), the context is not informative and does not provide much steering power for the visual exploration assistant. However, if the context is too fine-grained (e.g., the system detects that user just invoked the `dropna` command and determines all future data cleaning commands should be prioritized), the visual exploration assistant may appear to be too prescriptive since the low-level information does not fully capture the user's high-level goals.

A context-aware visual exploration assistant must also extract relevant context in a way that is interpretable to the user and explain how the context led to the recommended results (e.g., you might be interested in these outliers since you have performed these data cleaning operations). Given that any heuristics for inferring implicit intent could lead to false positives, the visual exploration assistant should surface the inferred context to the user and enable users to easily modify or correct any misinterpreted context.

Designing visual exploration assistants for real-world adoption

As described in Section 1.2 in Chapter 1, one of the central themes in this dissertation is our careful selection of research methodology with our research questions around how visual exploration assistants can be put into practice. By developing and evaluating visual exploration assistants to support real data analysis workflows across different areas of the design space, this dissertation serves as a roadmap for broader adoption of visual exploration assistants. From these experiences, we advocate that designers of visual exploration assistants should be aware of end-user considerations for adopting these tools in their existing workflows, such as the need for flexible ways to operate on the data.

There remain several research challenges around the adoption of visual exploration assistants around its design and evaluation. For instance, how do visual exploration assistants fit into different analyst personae and use cases? One axis of consideration is the role of expertise: in this dissertation, we saw how visual exploration assistants could benefit both non-programmer analysts and data scientists with substantial programming expertise. Could visual exploration assistants offer explanation or guidance to educate novices with limited data analysis or statistical knowledge to further democratize data exploration? When automated insights are presented to users in a way that is not appropriately contextualized or explained, there is danger for potential misinterpreta-

tion. (Chapter 3 describes one such fallacy.) How do real-world analysts make sense of the results provided by visual exploration assistants? How can visual exploration assistants help analysts validate and contextualize suggested results to prevent p-hacking [173, 35] and combat the problem of multiple comparisons [249]?

Another axis of consideration is how analyst's downstream goals of exploration impact the design of visual exploration assistants. We have seen how visual exploration assistants can help with creating a report or dashboard of insights and inform data cleaning or modeling decisions. Beyond exploratory use cases, could visual exploration assistants be used to support time-sensitive or mission-critical decisions in production settings? How can visual exploration assistants scalably support domain-specific suggestions that tailor to the specific needs of analysts in a particular domain? How do visual exploration assistants integrate with familiar, existing tools in the analyst's toolbox, such as spreadsheets and BI tools? Future studies that monitor the long-term deployment of visual exploration assistants across different user personae and use cases can shed light on these important questions around adoption.

In addition to investigating how visual exploration assistants can be useful to analysts, as system designers, it is equally important to ask the question: where do visual exploration assistants *not* work? For instance, while visual exploration assistants are well-suited for exploratory use cases, they are less well-suited for fine-grained presentational purposes, such as developing the powerful and expressive interactive visualizations that could be generated using D3 [25]. Visual exploration assistants also offer fewer adjustable settings and configurations compared to established BI tools such as Tableau. By identifying key strengths and weaknesses of visual exploration assistants, system designers can narrow down the design space and focus their efforts on integration with other tools that may provide complementary functionalities.

7.3 Final Remarks

As data becomes more central in the way that people make sense of the world and make decisions, we envision a future where anyone who wishes to understand and explore their data can do so with the help of visual exploration assistants. Visual exploration assistants empower analysts by helping them focus on leveraging their domain expertise and knowledge to ask better questions about their data, while hiding away the low-level operational details associated with visual data exploration. This dissertation demonstrates that visual exploration assistants can support analysts to discover valuable insights across different types of analytical tasks and modalities. In particular, mixed-initiative visual exploration assistants establish a productive collaboration between the human analyst and the assistant. With careful design and engineering, an intelligent visual exploration assistant should anticipate user intent, proactively seek opportunities to accelerate users towards insights, and offer feedback and guidance based on a user's analytical needs. Looking forward, future visual exploration assistants serve as an accessible means to guide, educate, and democratize the process of deriving value from data — bridging the gap between domain experts and data insights.

Bibliography

[1] *120 years of Olympic history: athletes and results*. https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results. Accessed: 2019-09-16. 2019.

[2] Gediminas Adomavicius and Young Ok Kwon. "Overcoming accuracy-diversity tradeoff in recommender systems: A variance-based approach". English (US). In: 2008 Workshop on Information Technologies and Systems, WITS 2008 ; Conference date: 13-12-2008 Through 14-12-2008. Jan. 2008, pp. 151–156.

[3] "Afghanistan: WHO mission reviews COVID-19 response". In: *World Health Organization* (2020).

[4] Google AI. *Facets: An Open Source Visualization Tool for Machine Learning Training Data*. URL: https://ai.googleblog.com/2017/07/facets-open-source-visualization-tool.html.

[5] Sara Alspaugh et al. "Futzing and Moseying: Interviews with Professional Data Analysts on Exploration Practices". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 22–31. DOI: 10.1109/TVCG.2018.2865040.

[6] Robert Amar, James Eagan, and John Stasko. "Low-level components of analytic activity in information visualization". In: *Proceedings - IEEE Symposium on Information Visualization* (2005), pp. 111–117. ISSN: 1522404X. DOI: 10.1109/INFVIS.2005.1532136.

[7] Saleema Amershi et al. "Software Engineering for Machine Learning: A Case Study". In: IEEE Computer Society, May 2019.

[8] Anushka Anand and Justin Talbot. "Automatic Selection of Partitioning Variables for Small Multiple Displays". In: *IEEE Transactions on Visualization and Computer Graphics* 2626.c (2015), pp. 669 –677. DOI: 10.1109/TVCG.2015.2467323.

[9] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. "Natural language interfaces to databases – an introduction". In: *Natural Language Engineering* 1.1 (1995), 29–81. DOI: 10.1017/S135132490000005X.

[10] Zan Armstrong and Martin Wattenberg. "Visualizing statistical mix effects and simpson's paradox". In: *IEEE transactions on visualization and computer graphics* 20.12 (2014), pp. 2132–2141.

[11] *Ask Data*. https://www.tableau.com/products/new-features/ask-data. Accessed: 2021-05-04. 2021.

[12] B. Shneiderman. "Creativity Support Tools: Accelerating Discovery and Innovation". In: *Communications of the ACM* 50.12 (2007), p. 20. DOI: 10.1145/1323688.1323689.

[13] Christopher Baik et al. "Duoquest: A Dual-Specification System for Expressive SQL Queries". In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. Portland, OR, USA: Association for Computing Machinery, 2020, 2319–2329. ISBN: 9781450367356. DOI: 10.1145/3318464.3389776.

[14] Eirik Bakke and David R Karger. "Expressive query construction through direct manipulation of nested relational results". In: *Proceedings of the 2016 International Conference on Management of Data*. ACM. 2016, pp. 1377–1392.

[15] Andrea Batch and Niklas Elmqvist. "The Interactive Visualization Gap in Initial Exploratory Data Analysis". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 278–287. ISSN: 10772626. DOI: 10.1109/TVCG.2017.2743990.

[16] Leilani Battle, Remco Chang, and Michael Stonebraker. "Dynamic Prefetching of Data Tiles for Interactive Visualization". In: (2016), pp. 1363–1375. ISSN: 07308078. DOI: 10.1145/2882903.2882919.

[17] Leilani Battle and Jeffrey Heer. "Characterizing Exploratory Visual Analysis: A Literature Review and Evaluation of Analytic Provenance in Tableau". In: *Eurographics Conference on Visualization (EuroVis) 2019* 38.3 (2019).

[18] Rohan Bavishi et al. "AutoPandas: neural-backed generators for program synthesis". In: *Proceedings of the ACM on Programming Languages* 3 (2019), pp. 1 –27.

[19] Jason Beaubien. "Why Rwanda Is Doing Better Than Ohio When It Comes To Controlling COVID-19". In: *NPR* (2020). URL: https://www.npr.org/sections/goatsandsoda/2020/07/15/889802561/a-covid-19-success-story-in-rwanda-free-testing-robot-caregivers.

[20] Mangesh Bendre et al. "DataSpread: Unifying Databases and Spreadsheets". In: *Proceedings of the VLDB Endowment* 8.12 (2015), pp. 2000–2003. ISSN: 2150-8097. DOI: 10.14778/2824032.2824121.

[21] Mary Beth Kery and Brad A. Myers. "Exploring exploratory programming". In: *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 2017, pp. 25–29. DOI: 10.1109/VLHCC.2017.8103446.

[22] Carsten Binnig et al. "Toward Sustainable Insights, or Why Polygamy is Bad for You". In: *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017*. 2017. URL: cidrdb.org/cidr2017/papers/p56-binnig-cidr17.pdf.

[23] Susanne Bodker, Kaj Gronbaek, and Morten Kyng. “Cooperative Design: Techniques and Experiences From the Scandinavian Scene”. In: ed. by Douglas Schuler and Aki Namioka. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1993. Chap. 8. ISBN: 0805809511.

[24] Claus Bossen, Christian Dindler, and Ole Sejer Iversen. “Evaluation in Participatory Design: A Literature Survey”. In: *Proceedings of the 14th Participatory Design Conference: Full Papers - Volume 1*. PDC ’16. Aarhus, Denmark: ACM, 2016, pp. 151–160. ISBN: 978-1-4503-4046-5. DOI: 10.1145/2940299.2940303.

[25] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. “D3: Data-Driven Documents”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (Dec. 2011), 2301–2309. ISSN: 1077-2626. DOI: 10.1109/TVCG.2011.185.

[26] Jeremy Boy, Francoise Detienne, and Jean-Daniel Fekete. “Storytelling in Information Visualizations: Does It Engage Users to Explore Data?” In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’15. Seoul, Republic of Korea: ACM, 2015, pp. 1449–1458. ISBN: 978-1-4503-3145-6. URL: doi.acm.org/10.1145/2702123.2702452.

[27] Matthew Brehmer and Tamara Munzner. “A Multi-Level Typology of Abstract Visualization Tasks”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2376–2385. ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.124.

[28] John Brooke. *”SUS-A quick and dirty usability scale.” Usability evaluation in industry*. ISBN: 9780748404605. CRC Press, 1996. URL: https://www.crcpress.com/product/isbn/9780748404605.

[29] Barry Brown, Stuart Reeves, and Scott Sherwood. “Into the Wild: Challenges and Opportunities for Field Trial Methods”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: Association for Computing Machinery, 2011, 1657–1666. ISBN: 9781450302289. DOI: 10.1145/1978942.1979185.

[30] Yang Cao et al. “Edgel index for large-scale sketch-based image search”. In: *CVPR 2011*. 2011, pp. 761–768. DOI: 10.1109/CVPR.2011.5995460.

[31] Luigina Ciolfi et al. “Articulating Co-Design in Museums: Reflections on Two Participatory Processes”. In: *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing - CSCW ’16* (2016), pp. 13–25. DOI: 10.1145/2818048.2819967.

[32] William S. Cleveland and Robert McGill. “Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods”. In: *Journal of the American Statistical Association* 79.387 (1984), pp. 531–554. DOI: 10.1080/01621459.1984.10478080.

[33] K. Cook et al. “Mixed-initiative visual analytics using task-driven recommendations”. In: *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*. Los Alamitos, CA, USA: IEEE Computer Society, 2015, pp. 9–16. DOI: 10.1109/VAST.2015.7347625.

[34] Thomas H. Cormen et al. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 0262033844, 9780262033848.

[35] Michael Correll. “Ethical Dimensions of Visualization Research”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Vol. 25. CHI ’19 1. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, 1–13. ISBN: 9781450359702. DOI: 10.1145/3290605.3300418. URL: https://doi.org/10.1145/3290605.3300418.

[36] Michael Correll and Michael Gleicher. “The Semantics of Sketch: Flexibility in Visual Query Systems for Time Series Data”. In: *Visual Analytics Science and Technology (VAST), 2016 IEEE Conference on*. IEEE. 2016, pp. 131–140. DOI: 10.1109/VAST.2016.7883519.

[37] Michael Correll and Jeffrey Heer. “Surprise! Bayesian Weighting for De-Biasing Thematic Maps”. In: *IEEE Transactions on Visualization and Computer Graphics* 2626.c (2016), pp. 1–1. ISSN: 1077-2626. URL: dx.doi.org/10.1109/TVCG.2016.2598618.

[38] “COVID-19 in Pakistan: WHO fighting tirelessly against the odds”. In: *World Health Organization* (2020). URL: https://www.who.int/news-room/feature-stories/detail/covid-19-in-pakistan-who-fighting-tirelessly-against-the-odds.

[39] Zhe Cui et al. “DataSite: Proactive Visual Data Exploration with Computation of Insight-based Recommendations”. In: *CoRR* abs/1802.08621 (2018). arXiv: 1802.08621. URL: http://arxiv.org/abs/1802.08621.

[40] Tuan Nhon Dang and Leland Wilkinson. “ScagExplorer: Exploring Scatterplots by Their Scagnostics”. In: *Proceedings of the 2014 IEEE Pacific Visualization Symposium*. PACIFICVIS ’14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 73–80. ISBN: 978-1-4799-2874-3. DOI: 10.1109/PacificVis.2014.42.

[41] Çağatay Demiralp et al. “Foresight: Rapid data exploration through guideposts”. In: *arXiv preprint arXiv:1709.10513* (2017).

[42] Dgomonov. *Data Exploration on NYC Airbnb*. 2020. URL: https://www.kaggle.com/dgomonov/data-exploration-on-nyc-airbnb.

[43] Kedar Dhamdhere et al. “Analyza: Exploring Data with Conversation”. In: *Proceedings of the 22nd International Conference on Intelligent User Interfaces - IUI ’17*. 2017. ISBN: 9781450343480. DOI: 10.1145/3025171.3025227.

[44] Bolin Ding et al. "Sample + Seek: Approximating Aggregates with Distribution Precision Guarantee". In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD '16. San Francisco, California, USA: ACM, 2016, pp. 679–694. ISBN: 978-1-4503-3531-7. DOI: 10.1145/2882903.2915249. URL: http://doi.acm.org/10.1145/2882903.2915249.

[45] Diptikalyan Saha et al. "ATHENA: An Ontology-Driven System for Natural Language Querying over Relational Data Stores Diptikalyan". In: *Proceedings of the VLDB Endowment* 9.12 (2016), p. 1209. DOI: 10.4324/9780203932148.

[46] Drlica Wagner et al. "Dark Energy Survey Year 1 Results: The Photometric Data Set for Cosmology". In: *The Astrophysical Journal Supplement Series* 235.2 (Apr. 2018), p. 33. DOI: 10.3847/1538-4365/aab4f5. URL: https://doi.org/10.3847%2F1538-4365%2Faab4f5.

[47] Ian Drosos et al. "Wrex: A Unified Programming-by-Example Interaction for Synthesizing Readable Code for Data Scientists". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, 1–12. ISBN: 9781450367080. DOI: 10.1145/3313831.3376442.

[48] Pranavi Duvva. *Speed up EDA With the Intelligent Lux*. 2021. URL: https://pub.towardsai.net/speed-up-eda-with-the-intelligent-lux-37f96542527b.

[49] Philipp Eichmann and Emanuel Zgraggen. "Evaluating Subjective Accuracy in Time Series Pattern-Matching Using Human-Annotated Rankings". In: *Proceedings of the 20th International Conference on Intelligent User Interfaces - IUI '15* (2015), pp. 28–37. DOI: 10.1145/2678025.2701379.

[50] *Elections 2016 Exit Polls*. 2016. URL: http://edition.cnn.com/election/2016/results/exit-polls.

[51] Stef van den Elzen and Jarke J van Wijk. "Small multiples, large singles: A new approach for visual data exploration". In: *Computer Graphics Forum*. Vol. 32. 3pt2. Wiley Online Library. 2013, pp. 191–200.

[52] David W. Embley. "NFQL: The Natural Forms Query Language". In: *ACM Transactions on Database Systems (TODS)* 14.2 (June 1989), pp. 168–211. ISSN: 0362-5915. DOI: 10.1145/63500.64125.

[53] Alex Endert et al. "Semantic Interaction: Coupling Cognition and Computation through Usable Interactive Analytics". In: *IEEE Computer Graphics and Applications* 35.4 (2015), pp. 94–99. ISSN: 02721716. DOI: 10.1109/MCG.2015.91.

[54] Alex Endert et al. "The human is the loop: new directions for visual analytics". In: *Journal of Intelligent Information Systems* 43.3 (2014), pp. 411–435. ISSN: 1573-7675. DOI: 10.1007/s10844-014-0304-9.

[55] Jennifer English et al. "Flexible Search and Navigation using Faceted Metadata". In: (2002). URL: https://flamenco.berkeley.edu/papers/flamenco02.pdf.

[56] Ethan Fast et al. "Iris: A Conversational Agent for Complex Tasks". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18 (2018), 473:1–473:12. DOI: 10.1145/3173574.3174047.

[57] *Faster data exploration in Jupyter through Lux*. 2020. URL: blog.dominodatalab.com/faster-data-exploration-in-jupyter-through-the-lux-widget/.

[58] Fadi Fayez Thabtah. *Autism Screening Adult Data Set. UCI Machine Learning Repository*. 2017.

[59] Stephen Few. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. Analytics Press, 2012.

[60] Tong Gao et al. "DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*. UIST '15. Charlotte, NC, USA: Association for Computing Machinery, 2015, 489–500. ISBN: 9781450337793. DOI: 10.1145/2807442.2807478.

[61] Brian S. Gloss et al. "High resolution temporal transcriptomics of mouse embryoid body development reveals complex expression dynamics of coding and noncoding loci". In: *Scientific Reports* 7.1 (2017), p. 6731. ISSN: 2045-2322. DOI: 10.1038/s41598-017-06110-5.

[62] Google. "Explore in Google Sheets". In: (2015). URL: https://www.youtube.com/watch?v=9TiXR5wwqPs.

[63] David Gotz, Shun Sun, and Nan Cao. "Adaptive Contextualization: Combating Bias During High-Dimensional Visualization and Data Selection". In: *Proceedings of the 21st International Conference on Intelligent User Interfaces - IUI '16* (2016), pp. 85–95. DOI: 10.1145/2856767.2856779. URL: http://dl.acm.org/citation.cfm?doid=2856767.2856779.

[64] John D Gould and Clayton Lewis. "Designing for usability—key principles and what designers think". In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* 28.3 (1983), pp. 50–53. ISSN: 00010782. DOI: 10.1145/800045.801579.

[65] L. Grammel, M. Tory, and M. Storey. "How Information Visualization Novices Construct Visualizations". In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (Nov. 2010), pp. 943–952. ISSN: 2160-9306. DOI: 10.1109/TVCG.2010.164.

[66] Jim Gray et al. "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals". In: *Data Mining and Knowledge Discovery* 1.1 (Mar. 1997), pp. 29–53. ISSN: 1573-756X. URL: doi.org/10.1023/A:1009726021843.

[67] Sumit Gulwani, William R. Harris, and Rishabh Singh. "Spreadsheet data manipulation using examples". In: *Commun. ACM* 55 (2012), pp. 97–105.

[68] Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. "Program Synthesis". In: *Foundations and Trends in Programming Languages* 4.1-2 (2017), pp. 1–119. ISSN: 2325-1107. DOI: 10.1561/2500000010.

[69] Yue Guo et al. "What you see is not what you get ! Detecting Simpson ' s Paradoxes during Data Exploration". In: *HILDA 2017 - Proceedings of the Workshop on Human-In-the-Loop Data Analytics* (2017).

[70] Thomas Hale et al. "A global panel database of pandemic policies (Oxford COVID-19 Government Response Tracker)". In: *Nature Human Behaviour* (2021). ISSN: 2397-3374. DOI: 10.1038/s41562-021-01079-8.

[71] Jiawei Han. *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN: 1558609016.

[72] "Happy Planet Index". In: (2019).

[73] Harlan Harris, Sean Murphy, and Marck Vaisman. *Analyzing the Analyzers: An Introspective Survey of Data Scientists and Their Work*. O'Reilly Media, Inc., 2013. ISBN: 1449371760.

[74] Björn Hartmann et al. "Reflective physical prototyping through integrated design, test, and analysis". In: *UIST 2006: Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology* (2008), pp. 299–308. DOI: 10.1145/1166253.1166300.

[75] Andrew Head et al. "Managing Messes in Computational Notebooks". In: (2019).

[76] Marti A. Hearst. *Search User Interfaces*. 1st. New York, NY, USA: Cambridge University Press, 2009. ISBN: 0521113792, 9780521113793.

[77] Jeffrey Heer. "Agency plus automation: Designing artificial intelligence into interactive systems". In: *Proceedings of the National Academy of Sciences* 116.6 (2019), pp. 1844–1850. ISSN: 0027-8424. DOI: 10.1073/pnas.1807184115.

[78] Jeffrey Heer, Joseph M. Hellerstein, and Sean Kandel. "Predictive Interaction for Data Transformation". In: *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. 2015. URL: http://cidrdb.org/cidr2015/Papers/CIDR15_Paper27.pdf.

[79] Jeffrey Heer and Ben Shneiderman. "Interactive Dynamics for Visual Analysis". In: *Queue* 10.2 (2012), p. 30. ISSN: 15427730. DOI: 10.1145/2133416.2146416.

[80] Harry Hochheiser and Ben Shneiderman. "Dynamic query tools for time series data sets: Timebox widgets for interactive exploration". In: *Information Visualization* 3.1 (2004), pp. 1–18. ISSN: 1473-8716.

[81] Harry Hochheiser and Ben Shneiderman. "Interactive Exploration of Time Series Data". In: *Discovery Science*. Berlin, Heidelberg: Springer, 2001, pp. 441–446. ISBN: 978-3-540-45650-6.

[82] Fred Hohman et al. "Gamut: A Design Probe to Understand How Data Scientists Understand Machine Learning Models". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2019. DOI: 10.1145/3290605.3300809.

[83] Karen Holtzblatt and Sandra Jones. "Contextual Inquiry: A Participatory Technique for System Design". In: ed. by Douglas Schuler and Aki Namioka. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1993. Chap. 9. ISBN: 0805809511.

[84] Christian Holz and Steven Feiner. "Relaxed Selection Techniques for Querying Time-series Graphs". In: *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*. UIST '09. Victoria, BC, Canada: ACM, 2009, pp. 213–222. ISBN: 978-1-60558-745-5. DOI: 10.1145/1622176.1622217.

[85] Enamul Hoque et al. "Applying Pragmatics Principles for Interaction with Visual Analytics". In: *IEEE Transactions on Visualization and Computer Graphics* c (2017). ISSN: 10772626. DOI: 10.1109/TVCG.2017.2744684.

[86] Eric Horvitz. "Principles of mixed-initiative user interfaces". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '99 (1999), pp. 159–166. DOI: 10.1145/302979.303030.

[87] Kevin Hu, Diana Orghian, and César Hidalgo. "DIVE: A Mixed-Initiative System Supporting Integrated Data Exploration Workflows". In: *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. ACM. 2018, p. 5.

[88] Jessica Hullman, Robert Kosara, and Heidi Lam. "Finding a Clear Path: Structuring Strategies for Visualization Sequences". In: *Comput. Graph. Forum* 36.3 (June 2017), pp. 365–375. ISSN: 0167-7055.

[89] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[90] Edwin L Hutchins, James D Hollan, and Donald A Norman. "Direct manipulation interfaces". In: *Human–Computer Interaction* 1.4 (1985), pp. 311–338.

[91] IBM Corp. *IBM SPSS Statistics for Windows*. Version 25.0. Armonk, NY: IBM Corp, 2017. URL: https://hadoop.apache.org.

[92] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. "Overview of data exploration techniques". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM. 2015, pp. 277–281.

[93] SAS Institute. *SAS user's guide: Statistics*. Vol. 2. Sas Inst, 1985.

[94] Petra Isenberg et al. "Grounded Evaluation of Information Visualization". In: *Proceedings of the 2008 conference on BEyond time and errors novel evaLuation methods for Information Visualization - BELIV '08* (2008), p. 1. DOI: 10.1145/1377966.1377974.

[95] Laurent Itti and Pierre Baldi. "Bayesian surprise attracts human attention". In: *Vision Research* 49.10 (May 2009), pp. 1295–1306. ISSN: 0042-6989. URL: dx.doi.org/10.1016/j.visres.2008.09.007.

[96] H. V. Jagadish et al. "Making Database Systems Usable". In: SIGMOD '07. Beijing, China: ACM, 2007, pp. 13–24. ISBN: 978-1-59593-686-8. DOI: 10.1145/1247480.1247483.

[97] Rogers Jeffrey and Leo John. "Ava : From Data to Insights Through Conversation". In: *8th Biennial Conference on Innovative Data Systems Research (CIDR)* (2017).

[98] Alekh Jindal et al. "Magpie: Python at Speed and Scale using Cloud Backends". In: *CIDR*. 2021. URL: https://www.microsoft.com/en-us/research/publication/magpie-python-at-speed-and-scale-using-cloud-backends/.

[99] Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. "Smart Drill-Down : A New Data Exploration Operator". In: *Proceedings of the 41st International Conference on Very Large Data Bases* 8.12 (2015), pp. 1928–1931. ISSN: 21508097.

[100] Sean Kandel et al. "Enterprise Data Analysis and Visualization: An Interview Study." In: *IEEE transactions on visualization and computer graphics* 18.12 (2012), pp. 2917–26. ISSN: 1941-0506. DOI: 10.1109/TVCG.2012.219.

[101] Sean Kandel et al. "Profiler: Integrated statistical analysis and visualization for data quality assessment". In: *Proceedings of the International Working Conference on Advanced Visual Interfaces*. ACM. 2012, pp. 547–554. DOI: 10.1145/2254556.2254659.

[102] Eser Kandogan and Ulrich Engelke. "Agile visual analytics in data science systems". In: *Proceedings - 18th IEEE International Conference on High Performance Computing and Communications, 14th IEEE International Conference on Smart City and 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016* (2017), pp. 1512–1519. DOI: 10.1109/HPCC-SmartCity-DSS.2016.0215.

[103] Eser Kandogan and Ulrich Engelke. "Towards a Unified Representation of Insight in Human-in-the-Loop Analytics". In: (2018), pp. 1–7. DOI: 10.1145/3209900.3209912.

[104] D A Keim et al. "Challenges in Visual Data Analysis". In: *Tenth International Conference on Information Visualization, 2006* Iv 2006 (2006), pp. 9–16. DOI: 10.1109/IV.2006.31.

[105] Daniel Keim et al. "Visual analytics: Definition, process, and challenges". In: *Information Visualization: Human-Centered Issues and Perspectives*. 2008, pp. 154–175.

[106] Mary Beth Kery, Amber Horvath, and Brad A Myers. "Variolite: Supporting Exploratory Programming by Data Scientists." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2017, pp. 1265–1276. ISBN: 9781450346559. DOI: 10.1145/3025453.3025626.

[107] Alicia Key et al. "VizDeck". In: *Proceedings of the 2012 international conference on Management of Data - SIGMOD '12* (2012), p. 681. ISSN: 07308078. DOI: 10.1145/2213836.2213931.

[108] Abhishek Khetan, Dilip Krishnamurthy, and Venkatasubramanian Viswanathan. "Towards Synergistic Electrode-Electrolyte Design Principles for Nonaqueous Li-O2 batteries". In: *Topics in Current Chemistry* 376 (Apr. 2018).

[109] Nodira Khoussainova et al. "Snipsuggest: Context-aware autocompletion for sql". In: *Proceedings of the VLDB Endowment* 4.1 (2010), pp. 22–33. ISSN: 2150-8097. DOI: 10.14778/1880172.1880175.

[110] Miryung Kim et al. "The emerging role of data scientists on software development teams". In: *Proceedings - International Conference on Software Engineering* 14-22-May-2016 (2016), pp. 96–107. ISSN: 02705257. DOI: 10.1145/2884781.2884783.

[111] Younghoon Kim et al. "GraphScape: A Model for Automated Reasoning about Visualization Similarity and Sequencing". In: (2017). DOI: 10.1145/3025453.3025866.

[112] Kkanda. *Analyzing UCI Crime and Communities Dataset*. 2020. URL: https://www.kaggle.com/kkanda/analyzing-uci-crime-and-communities-dataset.

[113] Gary Klein et al. "A data-frame theory of sensemaking". In: *Expertise out of Context: Proceedings of the Sixth International Conference on Naturalistic Decision Making* (Jan. 2007), pp. 113–155.

[114] Thomas Kluyver et al. "Jupyter Notebooks - a publishing format for reproducible computational workflows". In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by Fernando Loizides and Birgit Scmidt. Netherlands: IOS Press, 2016, pp. 87–90. URL: https://eprints.soton.ac.uk/403913/.

[115] A. P. Koenzen, N. A. Ernst, and M. A. D. Storey. "Code Duplication and Reuse in Jupyter Notebooks". In: *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 2020, pp. 1–9. DOI: 10.1109/VL/HCC50065.2020.9127202.

[116] Robert Kosara and Jock Mackinlay. "Storytelling: The Next Step for Visualization". In: *Computer* 46.5 (2013), pp. 44–50. DOI: 10.1109/MC.2013.36.

[117] Heidi Lam. "A Framework of interaction costs in information visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1149–1156. ISSN: 10772626. DOI: 10.1109/TVCG.2008.109.

[118] Po-Ming Law, Rahul C Basole, and Yanhong Wu. "Duet: Helping Data Analysis Novices Conduct Pairwise Comparisons by Minimal Specification". In: *IEEE transactions on visualization and computer graphics* 25.1 (2019), pp. 427–437.

[119] Po-Ming Law, Alex Endert, and John Stasko. "Characterizing Automated Data Insights". In: *IEEE VIS*, arXiv:2008.13060 (Aug. 2020), arXiv:2008.13060. arXiv: 2008.13060 [cs.HC].

[120] Doris Jung-Lin Lee. *Insight Machines: The Past, Present, and Future of Visualization Recommendation*. Feb. 2020. URL: https://medium.com/multiple-views-visualization-research-explained/insight-machines-the-past-present-and-future-of-visualization-recommendation-2185c33a09aa.

[121] Doris Jung-Lin Lee and Aditya Parameswaran. "The Case for a Visual Discovery Assistant: A Holistic Solution for Accelerating Visual Data Exploration". In: *IEEE Bulletin of Technical Committee on Data Engineering* 41.3 (2018), pp. 3–14.

[122] Doris Jung-Lin Lee et al. "Avoiding Drill-down Fallacies with VisPilot: Assisted Exploration of Data Subsets". In: *Proceedings of the 24th International Conference on Intelligent User Interfaces*. IUI '19. Marina del Ray, California: ACM, 2019, pp. 186–196. ISBN: 978-1-4503-6272-6. DOI: 10.1145/3301275.3302307.

[123] Doris Jung-Lin Lee et al. "Boomerang: Proactive Insight-Based Recommendations for Guiding Conversational Data Analysis". In: *Proceedings of the 2021 International Conference on Management of Data*. SIGMOD/PODS '21. Virtual Event, China: Association for Computing Machinery, 2021, 2750–2754. ISBN: 9781450383431. DOI: 10.1145/3448016.3452748.

[124] Doris Jung-Lin Lee et al. "Deconstructing Categorization in Visualization Recommendation: A Taxonomy and Comparative Study". In: *IEEE Transactions on Visualization and Computer Graphics* (2021), pp. 1–15. ISSN: 19410506. DOI: 10.1109/TVCG.2021.3085751. eprint: 2102.07070.

[125] Doris Jung-Lin Lee et al. "SCATTERSEARCH: Visual Querying of Scatterplot Visualizations". In: *IEEE Symposium on Information Visualization, 2019 (Poster)*. InfoVis '19. 2019. URL: https://arxiv.org/abs/1907.11743.

[126] Doris Jung-Lin Lee et al. "Three Lessons from Accelerating Scientific Insight Discovery via Visual Querying". In: *Patterns* 1 (Oct. 2020), p. 100126. DOI: 10.1016/j.patter.2020.100126.

[127] Doris Jung-Lin Lee et al. "You can't always sketch what you want: Understanding Sensemaking in Visual Query Systems". In: *IEEE Transactions on Visualization and Computer Graphics* (2019), pp. 1–1. DOI: 10.1109/TVCG.2019.2934666.

[128] Jaehui Park Sang-goo Lee. "Keyword search in relational databases". In: *Knowledge and Information Systems* 26.2 (2011), pp. 175–193. DOI: 10.1007/s10115-010-0284-1.

[129] Fei Li and H. V. Jagadish. "Constructing an interactive natural language interface for relational databases". In: *Proceedings of the VLDB Endowment* 8.1 (2014), pp. 73–84. ISSN: 21508097. DOI: 10.14778/2735461.2735468.

[130] Yunyao Li, Huahai Yang, and H. V. Jagadish. "Constructing a Generic Natural Language Interface for an XML Database". In: *Proceedings of the 10th International Conference on Advances in Database Technology*. EDBT'06. Munich, Germany: Springer-Verlag, 2006, pp. 737–754. ISBN: 3-540-32960-9, 978-3-540-32960-2. DOI: 10.1007/11687238_44.

[131] Yunyao Li, Huahai Yang, and HV Jagadish. "NaLIX: an interactive natural language interface for querying XML". In: *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM. 2005, pp. 900–902.

[132] Halden Lin, Dominik Moritz, and Jeffrey Heer. "Dziban : Balancing Agency & Automation in Visualization Design via Anchored Recommendations". In: *Proceedings of the Conference on Human Factors in Computing Systems - CHI '20* (2020).

[133] Zhicheng Liu and Jeffrey Heer. "The effects of interactive latency on exploratory visual analysis". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2122–2131. ISSN: 10772626. DOI: 10.1109/TVCG.2014.2346452.

[134] Zhicheng Liu et al. "Data Illustrator: Augmenting Vector Design Tools with Lazy Data Binding for Expressive Visualization Authoring". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 2018, 1–13. ISBN: 9781450356206. DOI: 10.1145/3173574.3173697.

[135] *LUX Exploratory Data Analysis (EDA)*. 2020. URL: https://www.youtube.com/watch?v=00BEjUzKDmY.

[136] *LUX Library: Matplotlib replacer?* 2020. URL: https://www.youtube.com/watch?v=m41h4mGzwwE.

[137] Stephen Macke et al. "Adaptive Sampling for Rapidly Matching Histograms". In: *Proc. VLDB Endow.* 11.10 (June 2018), pp. 1262–1275. ISSN: 2150-8097. DOI: 10.14778/3231751.3231753.

[138] Jock Mackinlay. "Automating the design of graphical presentations of relational information". In: *ACM Transactions on Graphics* 5.2 (1986), pp. 110–141. ISSN: 07300301. DOI: 10.1145/22949.22950. URL: http://portal.acm.org/citation.cfm?doid=22949.22950.

[139] Jock D. Mackinlay, Pat Hanrahan, and Chris Stolte. "Show Me: Automatic presentation for visual analysis". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1137–1144. ISSN: 10772626. DOI: 10.1109/TVCG.2007.70594.

[140] Narges Mahyar and Melanie Tory. "Supporting communication and coordination in collaborative sensemaking". In: *IEEE transactions on visualization and computer graphics* 20.12 (2014), pp. 1633–1642.

[141] Miro Mannino and Azza Abouzeid. "Qetch: Time Series Querying with Expressive Sketches". In: *SIGMOD Conference*. 2018.

[142] Miro Mannino and Azza Abouzied. "Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches". In: CHI '18 (2018), pp. 1–12. DOI: 10.1145/3173574.3173962.

[143] MATLAB. Natick, Massachusetts: The MathWorks Inc., 2010.

[144] Mary L. McHugh. "The Chi-square test of independence". In: *Biochemia Medica* 23.2 (June 2013), pp. 143–149. ISSN: 1330-0962. URL: ncbi.nlm.nih.gov/pmc/articles/PMC3900058/.

[145] Peter Mclachlan, Tamara Munzner, and Florham Park. "LiveRAC : Interactive Visual Exploration of System Management Time-Series Data". In: *CHI 08 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2008), pp. 1483–1492.

[146] Brenna McNally et al. "Gains from Participatory Design Team Membership As Perceived by Child Alumni and Their Parents". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, 2017, pp. 5730–5741. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025622.

[147] Sean M. McNee, John Riedl, and Joseph A. Konstan. "Making Recommendations Better: An Analytic Model for Human-Recommender Interaction". In: *CHI '06 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '06. Montréal, Québec, Canada: Association for Computing Machinery, 2006, 1103–1108. ISBN: 1595932984. DOI: 10.1145/1125451.1125660.

[148] Microsoft Corporation. *Microsoft Excel*. Version 2019 (16.0). Sept. 24, 2018. URL: https://office.microsoft.com/excel.

[149] Matt Mohebbi et al. "Google correlate whitepaper". In: (2011).

[150] Michael J. Muller and Sandra Kogan. "Grounded Theory Method in HCI and CSCW". In: *Human Computer Interaction Handbook* (2012), pp. 1003–1024.

[151] Michael J. Muller and Sarah Kuhn. "Participatory Design". In: *Communications of the ACM* 36.6 (June 1993), pp. 24–28. ISSN: 0001-0782. DOI: 10.1145/153571.255960.

[152] T. Munzner. *Visualization Analysis and Design*. AK Peters Visualization Series. CRC Press, 2015. ISBN: 9781498759717. URL: https://books.google.de/books?id=NfkYCwAAQBAJ.

[153] Tamara Munzner. "Chapter 3. Why: Task Abstraction". In: *Visualization Analysis and Design*. 2014, pp. 42–65. ISBN: 978-1-4665-0891-0. DOI: 10.1200/JCO.2003.02.071.

[154] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN: 0262018020, 9780262018029.

[155] Arpit Narechania, Arjun Srinivasan, and John Stasko. "NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries". In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2020). DOI: 10.1109/TVCG.2020.3030378.

[156] Jakob Nielsen. "Usability Inspection Methods". In: *Conference Companion on Human Factors in Computing Systems*. CHI '94. Boston, Massachusetts, USA: ACM, 1994, pp. 413–414. ISBN: 0-89791-651-4. DOI: 10.1145/259963.260531.

[157] Jakob Nielsen and Rolf Molich. "Heuristic Evaluation of user interfaces". In: *CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* April (1990), pp. 249–256. ISSN: 1942-597X. DOI: 10.1145/97243.97281.

[158] Donald A. Norman and Stephen W. Draper. *User Centered System Design; New Perspectives on Human-Computer Interaction*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1986. ISBN: 0898597811.

[159] Chris North. "Toward measuring visualization insight". In: *IEEE Computer Graphics and Applications* 26.3 (2006), pp. 6–9. ISSN: 02721716. DOI: 10.1109/MCG.2006.70.

[160] Christopher Olston and Ed H. Chi. "ScentTrails". In: *ACM Transactions on Computer-Human Interaction* 10.3 (2003), pp. 177–197. ISSN: 10730516. DOI: 10.1145/937549.937550.

[161] Nils O. Ommen et al. "Toward a better understanding of stakeholder participation in the service innovation process: More than one path to success". In: *Journal of Business Research* 69.7 (2016), pp. 2409–2416. ISSN: 0148-2963. DOI: https://doi.org/10.1016/j.jbusres.2016.01.010.

[162] lux org. *lux-logger*. URL: https://github.com/lux-org/lux-logger.

[163] Parul Pandey. *Intelligent Visual Data Discovery with Lux: A Python library*. 2021. URL: https://towardsdatascience.com/intelligent-visual-data-discovery-with-lux-a-python-library-dc36a5742b2f.

[164] *papermill 2.3.3 documentation*. 2020. URL: https://papermill.readthedocs.io/.

[165] Aditya G. Parameswaran, Hector Garcia-Molina, and Jeffrey D. Ullman. "Evaluating, combining and generalizing recommendations with prerequisites". In: *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10* (2010), p. 919. URL: dx.doi.org/10.1145/1871437.1871555.

[166] DJ Patil. *Building data science teams: the skills, tools and perspectives behind great data science groups*. O'Reilly, 2011.

[167] Pei Chen Peng and Saurabh Sinha. "Quantitative modeling of gene expression using DNA shape features of binding sites". In: *Nucleic Acids Research* 44.13 (2016), e120. DOI: 10.1093/nar/gkw446.

[168] Devin Petersohn et al. "Towards Scalable Dataframe Systems". In: *Proc. VLDB Endow.* 13.12 (July 2020), 2033–2046. ISSN: 2150-8097. DOI: 10.14778/3407790.3407807.

[169] Bremaud Pierre. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer, 2011.

[170] E. Pierson et al. *A large-scale analysis of racial disparities in police stops across the United States*. 2017. URL: http://openpolicing.stanford.edu/data/.

[171] Peter Pirolli and Stuart Card. "The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis". In: *Proceedings of International Conference on Intelligence Analysis*. Vol. 5. 2005, pp. 2–4.

[172] *Power BI — Interactive Data Visualization BI Tools*. https://powerbi.microsoft.com/en-us/.

[173] X. Pu and M. Kay. "The Garden of Forking Paths in Visualization: A Design Space for Reliable Exploratory Visual Analytics : Position Paper". In: *2018 IEEE Evaluation and Beyond - Methodological Approaches for Visualization (BELIV)*. Oct. 2018, pp. 37–45. DOI: 10.1109/BELIV.2018.8634103.

[174] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2019. URL: https://www.R-project.org/.

[175] Khairi Reda et al. "Modeling and evaluating user behavior in exploratory visual analysis". In: *Information Visualization* 15.4 (2016), pp. 325–339. ISSN: 14738724. DOI: 10.1177/1473871616638546.

[176] Donghao Ren, Bongshin Lee, and Matthew Brehmer. "Charticulator: Interactive Construction of Bespoke Chart Layouts". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019), 789–799. ISSN: 1077-2626. DOI: 10.1109/TVCG.2018.2865158.

[177] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, PBC. Boston, MA, 2020. URL: http://www.rstudio.com/.

[178] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. "The Earth Mover's Distance as a Metric for Image Retrieval". In: *International Journal of Computer Vision* 40.2 (2000), pp. 99–121. ISSN: 1573-1405. DOI: 10.1023/A:1026543900054.

[179] Adam Rule and U C San Diego. "Aiding Collaborative Reuse of Computational Notebooks with Annotated Cell Folding". In: 2.November (2018).

[180] Kathy Ryall et al. "Querylines: approximate query for visual browsing". In: *CHI'05 Extended Abstracts on Human Factors in Computing Systems*. ACM. 2005, pp. 1765–1768. DOI: 10.1145/1056808.1057017.

[181] Bahador Saket et al. "Visualization by Demonstration: An Interaction Paradigm for Visual Data Exploration". In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 331–340. DOI: 10.1109/TVCG.2016.2598839.

[182] S. Sarawagi. "User-adaptive exploration of multidimensional data". In: *Proc of the 26th Intl Conference on Very Large* (2000), pp. 307–316. URL: citeseer.ist.psu.edu/sarawagi00useradaptive.html.

[183] S Sarawagi et al. “Discovery-driven exploration of OLAP data cubes”. In: *6th International Conference on Extending Database Technology (EDBT 98)*. EDBT ’98 (1998), pp. 168–182. ISSN: 03029743. URL: http://dl.acm.org/citation.cfm?id=645338.650401.

[184] Sunita Sarawagi. “User-Cognizant Multidimensional Analysis”. In: *The VLDB Journal* 10.2–3 (Sept. 2001), 224–239. ISSN: 1066-8888.

[185] Ali Sarvghad, Melanie Tory, and Narges Mahyar. “Visualizing Dimension Coverage to Support Exploratory Analysis”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 21–30. ISSN: 10772626. DOI: 10.1109/TVCG.2016.2598466.

[186] Arvind Satyanarayan and Jeffrey Heer. “Lyra: An interactive visualization design environment”. In: *Computer Graphics Forum*. Vol. 33. 3. Wiley Online Library. 2014, pp. 351–360.

[187] Arvind Satyanarayan et al. “Reactive vega: A streaming dataflow architecture for declarative interactive visualization”. In: *IEEE transactions on visualization and computer graphics* 22.1 (2016), pp. 659–668.

[188] Arvind Satyanarayan et al. “Vega-lite: A grammar of interactive graphics”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 341–350.

[189] M. Sedlmair et al. “A Taxonomy of Visual Cluster Separation Factors”. In: *Computer Graphics Forum* 31.3pt4 (Dec. 2012), pp. 1335–1344. ISSN: 1077-2626. DOI: 10.1111/j.1467-8659.2012.03125.x.

[190] Edward Segel and Jeffrey Heer. “Narrative visualization: Telling stories with data”. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1139–1148. ISSN: 10772626. URL: dx.doi.org/10.1109/TVCG.2010.179.

[191] Jinwook Seo and Ben Shneiderman. “A rank-by-feature framework for interactive exploration of multidimensional data”. In: *Information visualization* 4.2 (2005), pp. 96–113.

[192] Vidya Setlur, Melanie Tory, and Alex Djalali. “Inferencing Underspecified Natural Language Utterances in Visual Analysis”. In: *Proceedings of the 24th International Conference on Intelligent User Interfaces*. IUI ’19. Marina del Ray, California: Association for Computing Machinery, 2019, 40–51. ISBN: 9781450362726. DOI: 10.1145/3301275.3302270.

[193] Vidya Setlur et al. “Eviza: A Natural Language Interface for Visual Analysis”. In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST ’16* (2016), pp. 365–377. DOI: 10.1145/2984511.2984588. URL: http://doi.acm.org/10.1145/2984511.2984588.

[194] Helen Sharp, Yvonne Rogers, and Jenny Preece. *Interaction Design: Beyond Human Computer Interaction*. USA: John Wiley and Sons, Inc., 2007. ISBN: 0470018666.

[195] Ben Shneiderman. "Direct Manipulation: A Step Beyond Programming Languages." In: *IEEE Computer* 16.8 (1983), pp. 57–69. URL: http://dblp.uni-trier.de/db/journals/computer/computer16.html#Shneiderman83.

[196] Ben Shneiderman. "Dynamic queries for visual information seeking". In: *IEEE Software* 11.6 (1994), pp. 70–77. ISSN: 07407459. DOI: 10.1109/52.329404.

[197] Ben Shneiderman. "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations". In: *Proceedings of the 1996 IEEE Symposium on Visual Languages*. VL '96. IEEE. Washington, DC, USA: IEEE Computer Society, 1996, pp. 336–. ISBN: 0-8186-7508-X. URL: http://dl.acm.org/citation.cfm?id=832277.834354.

[198] Ben Shneiderman and Catherine Plaisant. "Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies". In: *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*. ACM. 2006, pp. 1–7. DOI: 10.1145/1168149.1168158.

[199] Tarique Siddiqui et al. "Effortless data exploration with zenvisage: an expressive and interactive visual analytics system". In: *Proceedings of the VLDB Endowment* 10.4 (2016), pp. 457–468. DOI: 10.14778/3025111.3025126.

[200] Tarique Siddiqui et al. "Fast-Forwarding to Desired Visualizations with Zenvisage". In: *The biennial Conference on Innovative Data Systems Research (CIDR)*. 2017.

[201] Tarique Siddiqui et al. "Fast-Forwarding to Desired Visualizations with Zenvisage." In: *CIDR*. 2017.

[202] Tarique Siddiqui et al. "ShapeSearch: Flexible Pattern-based Querying of Trend Line Visualizations". In: *Proceedings of the VLDB Endowment* (2019).

[203] Svenja Simon et al. "Bridging the Gap of Domain and Visualization Experts with a Liaison". In: *Eurographics Conference on Visualization (EuroVis) - Short Papers*. Ed. by E. Bertini, J. Kennedy, and E. Puppo. The Eurographics Association, 2015. DOI: 10.2312/eurovisshort.20151137.

[204] Arjun Srinivasan et al. "Augmenting Visualizations with Interactive Data Facts to Facilitate Interpretation and Communication". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 672 –681.

[205] Arjun Srinivasan et al. "Discovering Natural Language Commands in Multimodal Interfaces". In: *Proceedings of the 24th International Conference on Intelligent User Interfaces*. IUI '19. Marina del Ray, California: Association for Computing Machinery, 2019, 661–672. ISBN: 9781450362726. DOI: 10.1145/3301275.3302292.

[206] *State of Data Science and Machine Learning 2019*. URL: https://www.kaggle.com/kaggle-survey-2019.

[207] C. Stolte, D. Tang, and P. Hanrahan. "Polaris: a system for query, analysis, and visualization of\nmultidimensional relational databases". In: *IEEE Transactions on Visualization and Computer Graphics* 8.1 (2002), pp. 1–14. ISSN: 10772626. DOI: 10.1109/2945.981851.

[208] Anselm Strauss and Juliet Corbin. "Grounded theory methodology: An overview." In: Handbook of qualitative research. Thousand Oaks, CA, US: Sage Publications, Inc, 1994, pp. 273–285. ISBN: 0-8039-4679-1 (Hardcover).

[209] *Tableau*. https://www.tableau.com/. Accessed: 2019-09-11. 2019.

[210] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[211] *Titanic: Machine Learning from Disaster. Kaggle.* 2017. URL: http://www.kaggle.com/c/titanic.

[212] Trifacta Wrangler. *Trifacta*. Oct. 13, 2020. URL: https://www.trifacta.com/.

[213] Edward R. Tufte. *The Visual Display of Quantitative Information*. USA: Graphics Press, 1986. ISBN: 096139210X.

[214] John W Tukey. "Exploratory data analysis". In: 2 (1977).

[215] Daniel Tunkelang. *Faceted Search*. Morgan and Claypool Publishers, 2009. ISBN: 1598299999, 9781598299991.

[216] *UCI Machine Learning Repository*. 2021. URL: https://archive.ics.uci.edu/ml/datasets.php.

[217] *US Department of Education: College Scorecard Data*. https://collegescorecard.ed.gov/data/documentation/. Accessed: 2019-09-16. 2019.

[218] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[219] Jacob VanderPlas et al. "Altair: Interactive Statistical Visualizations for Python". In: *Journal of Open Source Software* 3.32 (2018), p. 1057. DOI: 10.21105/joss.01057.

[220] Manasi Vartak et al. "SeeDB: efficient data-driven visualization recommendations to support visual analytics". In: *Proceedings of the VLDB Endowment* 8.13 (2015), pp. 2182–2193. DOI: 10.14778/2831360.2831371.

[221] Manasi Vartak et al. "Towards Visualization Recommendation Systems". In: *SIGMOD Records* 45.4 (May 2017), pp. 34–39. ISSN: 0163-5808. DOI: 10.1145/3092931.3092937. URL: http://doi.acm.org/10.1145/3092931.3092937.

[222] *Vega-Dataset: Cars*. https://vega.github.io/vega-datasets/data/cars.json. Accessed: 2019-09-16. 2019.

[223] Fernanda Viegas et al. "Generating charts from data in a data table". In: *US 20180088753 A1* (2018).

[224] Chenglong Wang et al. "Falx: Synthesis-Powered Visualization Authoring". In: *ArXiv* abs/2102.01024 (2021).

[225] Chenglong Wang et al. "Visualization by Example". In: *Proc. ACM Program. Lang.* 4.POPL (Dec. 2019). DOI: 10.1145/3371117.

[226] Martin Wattenberg. "Sketching a graph to query a time-series database". In: *CHI'01 Extended Abstracts on Human factors in Computing Systems*. ACM. 2001, pp. 381–382. DOI: 10.1145/634067.634292.

[227] James Wexler et al. "The What-If Tool: Interactive Probing of Machine Learning Models". In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 56–65. DOI: 10.1109/TVCG.2019.2934619.

[228] Ryen W. White and Steven M. Drucker. "Investigating Behavioral Variability in Web Search". In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. Banff, Alberta, Canada: ACM, 2007, pp. 21–30. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242576.

[229] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. URL: https://ggplot2.tidyverse.org.

[230] Hadley Wickham et al. *dplyr: A Grammar of Data Manipulation*. R package version 0.7.6. 2018. URL: https://CRAN.R-project.org/package=dplyr.

[231] Hadley Wickham et al. "Welcome to the tidyverse". In: *Journal of Open Source Software* 4.43 (2019), p. 1686. DOI: 10.21105/joss.01686.

[232] Cornellius Yudha Wijaya. *Quick Recommendation-Based Data Exploration with Lux*. 2021. URL: https://towardsdatascience.com/quick-recommendation-based-data-exploration-with-lux-f4d0ccb68133.

[233] Wikipedia contributors. *Minimax — Wikipedia, The Free Encyclopedia*. [Online; accessed 30-December-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=Minimax&oldid=866945016.

[234] Leland Wilkinson. *The Grammar of Graphics (Statistics and Computing)*. Berlin, Heidelberg: Springer-Verlag, 2005. ISBN: 0387245448.

[235] Leland Wilkinson, Anushka Anand, and Robert Grossman. "High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions". In: *IEEE Transactions on Visualization and Computer Graphics* 12.6 (2006), pp. 1363–1372. ISSN: 10772626. DOI: 10.1109/TVCG.2006.94. eprint: 1077-2626.

[236] Leland Wilkinson, Anushka Anand, and Robert Grossman. "Graph-Theoretic Scagnostics". In: *IEEE Symposium on Information Visualization (INFOVIS)* (2005), pp. 157–164.

[237] Graham Wills and Leland Wilkinson. "Autovis: automatic visualization". In: *Information Visualization* 9.1 (2010), pp. 47–69.

[238] Kanit Wongsuphasawat, Y. Liu, and J. Heer. "Goals, Process, and Challenges of Exploratory Data Analysis: An Interview Study". In: *ArXiv* abs/1911.00568 (2019).

[239] Kanit Wongsuphasawat et al. "Towards a general-purpose query language for visualization recommendation". In: *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. ACM. 2016, p. 4.

[240] Kanit Wongsuphasawat et al. "Visualizing dataflow graphs of deep learning models in tensorflow". In: *IEEE transactions on visualization and computer graphics* 24.1 (2017), pp. 1–12. ISSN: 10772626. DOI: 10.1109/TVCG.2017.2744878.

[241] Kanit Wongsuphasawat et al. "Voyager 2 : Augmenting Visual Analysis with Partial View Specifications". In: (2017). DOI: 10.1145/3025453.3025768.

[242] Kanit Wongsuphasawat et al. "Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations". In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 649–658. ISSN: 10772626. DOI: 10.1109/TVCG.2015.2467191.

[243] Eugene Wu and Samuel Madden. "Scorpion: Explaining Away Outliers in Aggregate Queries". In: *Proceedings of the VLDB Endowment* 6.8 (2013), pp. 553–564. ISSN: 21508097. DOI: 10.14778/2536354.2536356.

[244] Yifan Wu. "Is a Dataframe Just a Table?" In: *PLATEAU Workshop at UIST*. 2019.

[245] Mehmet Adil Yalçın, Niklas Elmqvist, and Benjamin B Bederson. "Keshif: Rapid and expressive tabular data exploration for novices". In: *IEEE transactions on visualization and computer graphics* 24.8 (2018), pp. 2339–2352.

[246] Cong Yan and Yeye He. "Auto-Suggest: Learning-to-Recommend Data Preparation Steps Using Data Science Notebooks". In: *International Conference on Management of Data (SIGMOD)*. 2020.

[247] Ka-Ping Yee et al. "Faceted Metadata for Image Search and Browsing". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '03. Ft. Lauderdale, Florida, USA: ACM, 2003, pp. 401–408. ISBN: 1-58113-630-7. DOI: 10.1145/642611.642681.

[248] Jeffrey Xu Yu, Lu Qin, and Lijun Chang. "Keyword Search in Relational Databases : A Survey Schema-Based Keyword Search on Relational Databases". In: (2010), pp. 1–12.

[249] Emanuel Zgraggen et al. "Investigating the Effect of the Multiple Comparisons Problem in Visual Analysis". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: ACM, 2018, 479:1–479:12. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174053.

[250] Emanuel Zgraggen et al. "(s—qu)eries: Visual Regular Expressions for Querying and Exploring Event Sequences". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (2015), pp. 2683–2692. DOI: 10.1145/2702123.2702262.

[251] Tianyi Zhang et al. "Interactive Program Synthesis by Augmented Examples". In: *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. UIST '20. Virtual Event, USA: Association for Computing Machinery, 2020, 627–648. ISBN: 9781450375146. DOI: 10.1145/3379337.3415900.

[252] Tao Zhou et al. "Solving the apparent diversity-accuracy dilemma of recommender systems". In: *Proceedings of the National Academy of Sciences* 107.10 (2010), pp. 4511–4515. DOI: 10.1073/pnas.1000488107.

[253] *Zillow*. www.zillow.com. Accessed: February 1, 2016. 2016.

[254] John Zimmerman, Jodi Forlizzi, and Shelley Evenson. "Research through Design as a Method for Interaction Design Research in HCI". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. San Jose, California, USA: Association for Computing Machinery, 2007, 493–502. ISBN: 9781595935939. DOI: 10.1145/1240624.1240704.

[255] Moshé M Zloof. "Query by example". In: *Proceedings of the May 19-22, 1975, national computer conference and exposition*. ACM. 1975, pp. 431–438.

[256] Jonathan Zong et al. "Lyra 2: Designing Interactive Visualizations by Demonstration". In: *IEEE Transactions on Visualization and Computer Graphics* 27 (2021), pp. 304–314.