Abstract of "Towards Accessible Data Analysis" by Emanuel Albert Errol Zgraggen, Ph.D., Brown University, April 2018.

In today's world data is ubiquitous. Increasingly large and complex datasets are gathered across many domains. Data analysis - making sense of all this data - is exploratory by nature, demanding rapid iterations, and all but the simplest analysis tasks require humans in the loop to effectively steer the process. Current tools that support this process are built for an elite set of individuals: highly trained analysts or data scientists who have strong mathematics and computer science skills. This however presents a bottleneck. Qualified data scientists are scarce and expensive which makes it often unfeasible to inform decisions with data. How do we empower data enthusiasts, stakeholders or subject matter experts, who are not statisticians or programmers, to directly tease out insights from data? This thesis presents work towards making data analysis more accessible. We invent a set of user experiences with approachable visual metaphors where building blocks are directly manipulatable and incrementally composable to support common data analysis tasks at the pace that matches the thought process of a humans.

First, we develop a system for back-of-the-envelope calculations that revolves around handwriting recognition - all data is represented as digital ink - and gestural commands. Second, we introduce a novel pen & touch system for data exploration and analysis which is based on four core interaction concepts. The combination and interplay between those concepts supports a wide range of common analytical tasks. The interface allows for incremental and piecewise query specification where intermediate visualizations serve as feedback as well as interactive handles to adjust query parameters. Third, we present a visual query interface for event sequence data. This touch-based interface exposes the full expressive power of regular expressions in an approachable way and interleaves query specification with result visualizations. Fourth, we present the results of an experiment where we analyze how progressive visualizations affect exploratory analysis. Based on these results, which suggest that progressive visualizations are a viable solution to achieve scalability in data exploration systems, we develop a system entirely based on progressive computation that allows users to interactively build complex analytics workflows. And finally, we discuss and experimentally show that using visual analysis tools might inflate false discovery rates among user-extracted insights and suggest ways of ameliorating this problem.

Towards Accessible Data Analysis

by

Emanuel Albert Errol Zgraggen

Fachhochschul Diplom, Hochschule für Technik Rapperswil, 2007

Sc. M., Brown University, 2012

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island

April 2018

This dissertation by Emanuel Albert Errol Zgraggen is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____

Andries van Dam, Advisor

Recommended to the Graduate Council

Date _____ _____

Tim Kraska, Reader
Brown University

Date _____ _____

Steven M. Drucker, Reader
Microsoft Research

Approved by the Graduate Council

Date _____ _____

Andrew G. Campbell
Dean of the Graduate School

Vita

Emanuel Albert Errol Zgraggen was born and raised in Switzerland. He finished a four year apprenticeship in Software Engineering and Business at Credit Suisse in 2002 after which he attended the Hochschule für Technik Rapperswil (HSR). He received a Fachhochschuldiplom in Computer Science from HSR in 2007. He then worked for several years as a full stack developer for web and mobile in Zurich before starting a Master's degree at Brown University in Providence in 2010. After earning his Master of Science degree in Computer Science in 2012 he started studying for his Ph.D under the advisement of Professor Andries van Dam (and later co-advised by Professor Tim Kraska). During his time as a Ph.D student, Emanuel interned twice at Microsoft Research in Redmond. He is a recipient of the Design Award Switzerland, a scholarship Hasler Stiftung and the Andries van Dam Graduate Fellowship. He will join the Computer Science and Artificial Intelligence Laboratory (CSAIL) at Massachusetts Institute of Technology in Cambridge in 2018 where he will be working as a postdoc in Professor Tim Kraka's group.

Acknowledgements

Foremost, I would like to thank Andy van Dam for his guidance, insights and freedom that he has provided over the years. I will always be grateful that he suggested I continue on as a Ph.D. student after finishing the Master's program. Similarly, I want to thank Tim Kraska, who co-advised my thesis work, for his mentorship, for always having great ideas and for pushing me to extend my work towards databases and machine learning. Many thanks to Steven Drucker for being a invaluable mentor and a great boss during my two internships at Microsoft Research and for taking me under his wings during my first research conferences.

Very special thanks to Bob Zeleznik, for his guidance, advice, help and feedback in research, for being a great friend and office mate, for answering my occasional questions about American culture, for informing me when there was free food and for many great discussions and debates about everything and nothing.

I am very grateful to the Brown Computer Science community. I would like to thank David Laidlaw and his research group for offering advice and feedback throughout my time at Brown. Thanks to all the amazing folks from the Database Group. Big thanks to tstaff and astaff, particularly Dawn Reed, Eugenia DeGouveia, Laura Dobler, Lauren Clarke and Lisa Manekofsky. And many thanks to my friends and peers in the department: Alex, Alexandra, Andrew, Conor, Evgenios, Hua, Nedi, Olga, Sam, Steve, Trent, Yeounoh and Zeyuang.

Alex Galakatos, Andrew Crotty, Bob Zeleznik, Carsten Binnig, Danyel Fisher, Eli Upfal, Jean-Daniel Fekete, Lorenzo De Stefani, Philipp Eichmann, Robert DeLine, Steven Drucker, Tim Kraska and Zheguang Zhao have been amazing collaborators and this work would not have been possible without them.

This endeavour would not have been possible without the support of the friends I made while at Brown: Cristina, Emily, Manisha, Martin, Nick, Pellumb, Philipp, Tak and Yuki; my friends

back home (thanks for making me feel like I never left whenever I visit): Fabio, Giama, Mäce and Thomas; my extended American family: Nancy, Toto and Tom; and my family and loved ones: Mimi, Püpel, Issmi, Gök, Angi, Monique, Boby, Samantha. Thank you all!

This research is funded in part by DARPA Award 16-43-D3M-FP-040, NSF Award IIS-1514491 and IIS-1562657, Andries van Dam Graduate Fellowship, the Intel Science and Technology Center for Big Data and gifts from Adobe, Google, Mellanox, Microsoft, Oracle, Sharp and VMware. All opinions, findings, conclusions, or recommendations expressed in this document are those of the author(s) and do not necessarily reflect the views of the sponsoring agencies.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Thesis Statement: Most people who rely on data to make informed decisions are not statisticians or computer scientists. However, tools that support data driven descision making are either targeted towards such advanced users or limited in functionality or scale. We aim to make data analysis more accessible by inventing a set of approachable user experiences that support data analysis tasks ranging from simple spreadsheet calculations to complex machine learning workflows on large datasets.

1.1 Motivation and Problem Statement

Data is everywhere. Companies store customer and sales information, researchers collect data by running experiments and application or website developers store interaction logs. But all this data is useless without the means to analyze it. Extracting actionable insights from data has been left to highly trained individuals who have strong mathematics and computer science skills. They have the background to query databases to create insightful reports and visualizations, develop statistical models and implement scalable infrastructures to process large and complex data. For example, it is common practice for corporations to employ teams of data scientists that assist stakeholders in finding qualitative, data-driven insights to inform possible business decisions. Having such a high-entry bar to data analysis however presents several challenges. For one, it presents a bottleneck. While research is trying to understand and promote visualization and data literacy [9, 32] and educational institutions are ramping up their data science curricula there is still a shortage of

skilled data scientists. And second, and more importantly, restricting data analysis to those with a computational background creates an inequality [23]. Small business owners without programming skills or research domains where computational background might not be as prevalent are at a disadvantage as they can not capitalize on the power of data.

We, as others [98, 63], believe that there is an opportunity for tool builders to create systems for *data enthusiasts* - people who are "not mathematicians or programmers, and only know a bit of statistics" [78]. Making sense of data is an exploratory process that demands for rapid iterations. This work is about creating visual interfaces that support this process at a pace that matches the thought process of human analysts and in ways that users can focus on applying their domain knowledge without requiring programming skills.

This dissertation starts out by showing how simple analysis tasks on small datasets - back back-of-the-envelope or spreadsheet-like calculations - can be exposed through a pen and paper like interface that leverages pre-learned skills such as handwritten math notation instead of specialized formula scripting languages, while still offering computational support. We then extend this style of direct manipulation to data that is stored in databases by presenting an approachable visual language that allows users to pose questions without writing SQL queries.

SQL databases are optimized for storing tabular data but are a notoriously bad fit for event sequence data: a type of data found in many domains ranging from electronic health records to telemetry and log data. We show a visual language that supports queries over such data through direct manipulation. Our system exposes the power of regular expressions and enables *data enthusiasts* to answer complex questions over event sequences.

The *fluid* [54] interaction style exhibited in all of our systems is designed to promote "flow" - staying immersed in the current activity and not being distracted by the user interface - and relies on prompt feedback and response times. However, datasets are often too large to process within interactive time-frames. We empirically show that progressive visualizations and computations show great promise as a viable solution for this problem. We build a progressive data processing backend and a user interface that supports visualizations and more complex analytical workflows such as building machine learning models to demonstrate the feasibility of this approach.

Empowering novice users to directly analyze data also comes with drawbacks. It exposes them to "the pitfalls that scientists are trained to avoid" [63]. We study and discuss one such pitfall - the multiple comparisons problem - and present some insights and techniques on how to address it.

1.2 Thesis Organization and Contributions

This thesis includes eight chapters, six of which contain primary contributions. We summarize these chapters and highlight their contributions in the following paragraphs. The bulk of this thesis has been previously published in journal and conference papers. Disclaimers at the beginning of each chapter highlight my personal contributions and the relevant publications are referenced here as well as at the beginning of each chapter.

Chapter 2: Spreadsheet-like Calculations through Digital Ink

In this chapter we present Tableur, a spreadsheet-like pen- & touch-based system. The need for back-of-the-envelope calculations, such as rough projections or simple budget estimations, occurs frequently and oftentimes while being away from desktop computers. While major software vendors have optimized their spreadsheet applications for mobile environments their generality still makes them heavyweight for such tasks. We have built Tableur targeted towards these use cases. Our design revolves around handwriting recognition - all data is represented as digital ink - and gestural commands. Through a rethought cell referencing system and by incorporating standard math notation recognition Tableur allows for simple formula creation and we experiment with techniques that support pattern-based prefilling of cells (*Smart Fill*) and exploration of what-if scenarios (*Reverse Editing*).

Emanuel Zgraggen, Robert Zeleznik, and Philipp Eichmann

Tableur: Handwritten Spreadsheets

In Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems

Chapter 3: Visual Data Exploration and Analysis through Pen & Touch

Interactively exploring multidimensional datasets requires frequent switching among a range of distinct but inter-related tasks (e.g., producing different visuals based on different column sets, calculating new variables, and observing the interactions between sets of data). Existing approaches either target specific different problem domains (e.g., data-transformation or data-presentation) or expose only limited aspects of the general exploratory process; in either case, users are forced to

adopt coping strategies (e.g., arranging windows or using undo as a mechanism for comparison instead of using side-by-side displays) to compensate for the lack of an integrated suite of exploratory tools. In this chapter we introduce PanoramicData (PD), a system which addresses these problems by unifying a comprehensive set of tools for visual data exploration into a hybrid pen and touch system designed to exploit the visualization advantages of large interactive displays. PD goes beyond just familiar visualizations by including direct UI support for data transformation and aggregation, filtering and brushing. Leveraging an unbounded whiteboard metaphor, users can combine these tools like building blocks to create detailed interactive visual display networks in which each visualization can act as a filter for others. Further, by operating directly on relational-databases, PD provides an approachable visual language that exposes a broad set of the expressive power of SQL, including functionally complete logic filtering, computation of aggregates and natural table joins. To understand the implications of this novel approach, we conducted a formative user study with both data and visualization experts. The results indicated that the system provided a fluid and natural user experience for probing multi-dimensional data and was able to cover the full range of queries that the users wanted to pose.

Emanuel Zgraggen, Robert Zeleznik, and Steven M Drucker

Panoramicdata: Data Analysis through Pen & Touch

IEEE Transactions on Visualization and Computer Graphics (InfoVis), 2014

Chapter 4: Visual Regular Expressions for Querying and Exploring Event Sequences

Many different domains collect event sequence data and rely on finding and analyzing patterns within it to gain meaningful insights. Current systems that support such queries either provide limited expressiveness, hinder exploratory workflows or present interaction and visualization models which do not scale well to large and multi-faceted data sets. In this paper we present (s|qu)eries (pronounced "Squeries"), a visual query interface for creating queries on sequences (series) of data, based on regular expressions. (s|qu)eries is a touch-based system that exposes the full expressive power of regular expressions in an approachable way and interleaves query specification with result visualizations. Being able to visually investigate the results of different query-parts supports debugging and encourages iterative query-building as well as exploratory work-flows. We validate our

design and implementation through a set of informal interviews with data scientists that analyze event sequences on a daily basis.

Emanuel Zgraggen, Steven M. Drucker, Danyel Fisher, and Robert DeLine
(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, CHI '15, 2015

Chapter 5: The Case for Progressive Visualizations

The stated goal for visual data exploration is to operate at a rate that matches the pace of human data analysts, but the ever increasing amount of data has led to a fundamental problem: datasets are often too large to process within interactive time frames. Progressive analytics and visualizations have been proposed as potential solutions to this issue. By processing data incrementally in small chunks, progressive systems provide approximate query answers at interactive speeds that are then refined over time with increasing precision. In this chapter we study how progressive visualizations affect users in exploratory settings in an experiment where we capture user behavior and knowledge discovery through interaction logs and think-aloud protocols. Our experiment includes three visualization conditions and different simulated dataset sizes. The visualization conditions are: (1) *blocking*, where results are displayed only after the entire dataset has been processed; (2) *instantaneous*, a hypothetical condition where results are shown almost immediately; and (3) *progressive*, where approximate results are displayed quickly and then refined over time. We analyze the data collected in our experiment and observe that users perform equally well with either instantaneous or progressive visualizations in key metrics, such as insight discovery rates and dataset coverage, while blocking visualizations have detrimental effects.

Emanuel Zgraggen, Alex Galakatos, Andrew Crotty, Jean-Daniel Fekete, and Tim Kraska
How Progressive Visualizations Affect Exploratory Analysis
IEEE Transactions on Visualization and Computer Graphics, 2016

Chapter 6: A System for Progressive Visualizations and Computations

In this chapter we present Vizdom, an interactive visual analytics system that scales to large datasets. Vizdom's design is informed by the findings from Chapter 5 which suggest that progressive visualizations are a viable solution to achieve scalability in visual data exploration systems. However, Vizdom goes beyond "just" visualizations and extends the concept of progressiveness to other types of reoccurring data analysis tasks such as building machine learning models. Vizdom scales seamlessly, and transparent to the user, across dataset sizes ranging from thousands to hundreds of millions of records. We present our system design, show how various data analysis tasks are supported within our system and discuss the limitations and advantages of progressive computations in the context of data analytics.

Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska
Vizdom: Interactive Analytics Through Pen & Touch
Proceedings of the VLDB Endowment, 2015

Chapter 7: Investigating the Effect of the Multiple Comparison Problem in Visual Analysis

The goal of a visualization system is to facilitate dataset-driven insight discovery. But what if the insights are spurious? Features or patterns in visualizations can be perceived as relevant insights, even though they may arise from noise. We often compare visualizations to a mental image of what we are interested in: a particular trend, distribution or an unusual pattern. As more visualizations are examined and more comparisons are made, the probability of discovering spurious insights increases. This problem is well-known in Statistics as the multiple comparisons problem (MCP) but overlooked in visual analysis. We present a way to evaluate MCP in visualization tools by measuring the accuracy of user reported insights on synthetic datasets with known ground truth labels. In our experiment, over 60% of user insights were false. We show how a confirmatory analysis approach that accounts for all visual comparisons, insights and non-insights, can achieve similar results as one that requires a validation dataset.

Emanuel Zgraggen, Zheguang Zhao, Robert Zeleznik, and Tim Kraska
Investigating the Effect of the Multiple Comparison Problem in Visual Analysis In Proceedings of the 36th Annual ACM Conference on Human Factors in Computing Systems, CHI '18, 2018

Zheguang Zhao, Emanuel Zgraggen, Lorenzo De Stefani, Carsten Binnig, Eli Upfal, and Tim Kraska

Safe visual data exploration

In Proceedings of the 2017 ACM International Conference on Management of Data

Chapter 2

Spreadsheet-like Calculations through Digital Ink

This chapter presents a system called Tableur, a spreadsheet-like pen- & touch-based system that revolves around handwriting recognition. This chapter is substantially similar to [177], where I was the first author and was responsible for the research direction, implementation and a majority of the writing.

2.1 Introduction

Spreadsheet applications are programs that allow users to enter, manipulate and analyze data that is represented in tabular form and through formulas that reference values in cells. They are widely used for different tasks and support advanced data analysis through built-in scripting languages. The need for simple calculations oftentimes occur while being in situations where a desktop computer is unavailable such as during meeting or while being at lunch with a colleague. Although spreadsheets are extremely powerful their complexity can be overwhelming especially while being in such mobile environments.

In this paper we present Tableur a novel gesture-based pen & touch system that offers spreadsheet-like functionality and is targeted towards back-of-the-envelope calculations on devices found in away-from-desk situations such as phablets, tablets and interactive whiteboards. We designed our system

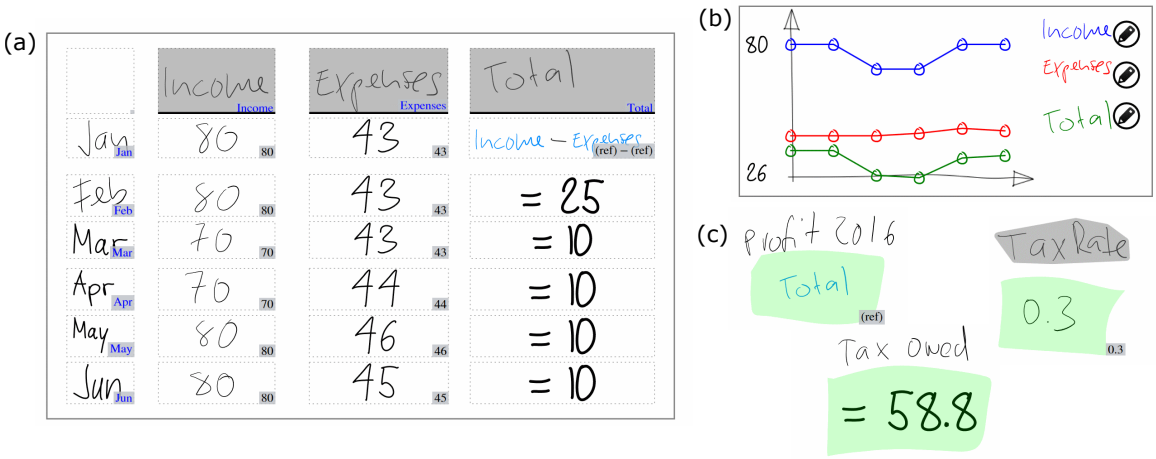

Figure 2.1: Screenshot of the application depicting the resulting view assembled by Eve throughout the introductory use case. a) Table with rows per month of estimates for income and expenses as well as formulas to compute the total. b) Line chart of "Income", "Expenses" and "Total" column. c) *Freeform Cells* that compute the sum over the "Total" column as well as taxes owed based on a constant "Tax Rate".

by following the *fluid* design guidelines [54] which promote direct manipulation, minimization of user interface clutter and advocate the integration of interface components directly into the visual representation. In Tableur all data is represented, created and edited as handwritten ink. We offer a set of gestures that manipulate and organize ink and incorporate handwriting recognition such that formulas can be created through standard handwritten math notation. We present a label-based design for referencing content in formulas and discuss techniques that support fast creation of patterns (*Smart Fill*), promote freeform workflows (*Freeform Cells*) and what-if scenarios (*Reverse Editing*).

2.2 Related Work

Numerous authors [27, 108, 109, 49, 107, 123] have emphasized the benefits of pen- & touch-based interfaces with regards to data exploration and analysis. Drucker et. al. [49] for example find that users not only subjectively prefer a touch interface over a traditional WIMP interface but also perform better with it for certain data related tasks. Others built pen- & touch-based systems that simplify story telling with data [108], allow users to gesturally create, label, filter and transform charts [27, 109] or enter calculated fields through handwriting [176]. However all of these systems focus on exploring and analyzing existing data and do not support standard spreadsheet functionality like creating and editing new data or referenced-based formulas. In other domains, such as diagram

drawing [174] or mathematical eduction [173, 172, 106], researchers have ported the immediacy and fluidity of paper-based pencil-drawing to digital media and augmented it with computational power. We adopted some of these approaches and techniques to the domain of spreadsheet calculations.

2.3 Tableur

We motivate our system through a use case throughout which we point to the appropriate sections that describe the techniques in more detail. Figure 2.1 shows the resulting spreadsheet that is assembled throughout the use case.

2.3.1 Use Case

Eve is the owner of a small business and wants to get a rough idea how her company will be performing over the next few months. She decides to use Tableur to assist her with this task. Tableur offers an unbounded zoom- and pan-able 2D canvas where ink can be placed anywhere with the use of a digital stylus. Eve starts by writing down an outline of her tabular data with the column headers "Income", "Expenses" and "Total" and rows labeled "Jan" and "Feb" indicating months she wants to analyze. She then performs a gesture (*Gestural System*) to signal that her ink should be interpreted as a tabular structure (*Ink Segmentation*). After filling out the individual cells with estimates of the respective incomes and expenses she realizes that she wants to expand her spreadsheet beyond just January and February. Another gesture prefills the cells with ink up until June (*Smart Fill*). Eve now wants to create a formula that computes the total (income minus expense) for each month. She does so by combining references to the "Income" and "Expenses" labels in the first row's "Total" column through drag and drop and handwrites a minus sign in between (*Formulas*). Propagating this formula to all cells of the "Total" column seems tedious so Eve opts to use another gesture to do so (*Smart Fill*). She fills out the rest of the spreadsheet with estimates for the respective months and then creates a free floating cell next to her table that she labels "Profit 2016" (*Freeform Cell*). She drops the "Total" label into that new cell and the system automatically calculates the sum over that column for her. In order to reinforce the numbers she just wrote down and calculated Eve decided to visualize them. She drags all the column labels out to free area of the canvas and the system shows her a simple line chart. A new supplier that Eve was in contact with offers very competitive prices that would cut down her monthly expenses from

roughly $60 to $40. To analyze this what-if scenario she edits the "Expenses" line in the chart by over-drawing a new horizontal line (expenses are more or less constant across months) at roughly the 40 mark (*Reverse Editing*). The system updates the numbers in her spreadsheet as well as her "Profit 2016" cell and she realizes that she could more than double her profit by switching to this new vendor. Eve finalizes her analysis by creating more *Freeform Cells*. She uses those to calculate how much in taxes she estimates to owe for the first half of 2016 based on a "Tax Rate" that she sets to 30%.

2.3.2 Gestural System

All commands within Tableur are triggered through gestures ranging from simple drag and drop operations to ink-based scribble-erase gestures [174] that remove unwanted content. To distinguish regular ink from gestures and to be able to overload gestures with multiple outcomes we implemented a two-step gesturizer system. All new ink gets analyzed and as soon as a potential gesture is recognized it gets highlighted and a pop-up menu is displayed. Selecting an option then triggers the gesture while ignoring it or tapping outside fades the gesture stroke back to normal ink. The top image in figure 2.2 illustrates this concept. The circular stroke has been recognized as a potential lasso gesture and is highlighted in red. The pop-up menu in the lower right corner informs the user that their gesture has two possible different outcomes. We are planning to incorporate techniques, like the one presented in GestureBar [25], in future versions of our prototype to enhance discoverability of our gesture set.

2.3.3 Ink Segmentation

Ink on the canvas is only interpreted and analyzed if the user chooses to transform it into an active object. Tableur offers two types of such objects: tables and *Freeform Cells*. A lasso gesture around existing ink (or a rectangular gestures for empty objects) is used to create such objects (figure 2.2 top). The system runs a segmentation algorithm that decides how to break the ink into a row and column structure. The algorithm looks at the distribution of ink among the X and Y axis and outputs a list of bounding boxes that represent individual cells. All ink is then assigned to its corresponding cell and the system invokes handwriting recognition on the content of each cell (figure 2.2 center). Note that such tables can easily be edited and extended by either writing more ink onto

Figure 2.2: Screenshots from the application. Top: Unsegmented ink outside of an active object with ongoing lasso gesture. Center: Table after ink has been segmented and assigned to corresponding cells. Bottom a) Expanded table to the right after adding ink. b) Ongoing expansion of table to the bottom.

them (figure 2.2 bottom) or by a set of gestures that allow to correct the automatic segmentation (splitting or merging of cells and columns).

2.3.4 Smart Fill

Standard spreadsheet applications usually offer functionality that helps users avoid doing repetitive and tedious tasks such as propagating formulas to different cells or expanding number or date patterns across columns or rows. Tableur incorporates similar functionality. A horizontal or vertical gesture across multiple cells activates this smart filling. By analyzing the cells under the gesture-stroke that contain content the system tries to extrapolate how to fill the remaining cells where the length of the gesture-stroke determines which existing cells need to be filled or how many additional cells need to be added. We currently support simple patterns such as dates (figure 2.3 left) or ascending and descending numbers as well as propagation of formulas where reference to rows or columns are updated accordingly (figure 2.3 right). Note that the added synthesized content is just like any other regular ink: it can be manipulate or deleted with the same commands. We sample letters or digits from a prerecorded alphabet that can be customized to match a user's handwriting.

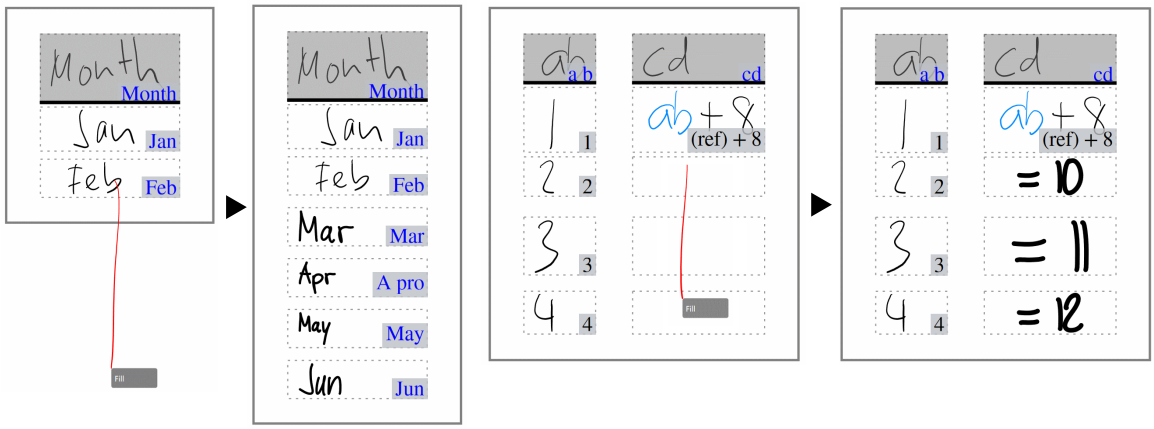

Figure 2.3: Two examples of *Smart Fill*. Left: Propagation of a date pattern. Right: Propagation of formulas with corresponding updates to row references.

2.3.5 Formulas

Formulas in Tableur live within cells of tables or *Freeform Cells* and consist of handwritten math, references to other cells or combinations thereof. The content of all cells is analyzed automatically as soon as it is inputed by the user. This step involves invoking two separate recognizers: a standard handwriting recognizer to detect textual input and a custom math recognizer. Tableur's math recognition engine is based on prior work developed by colleagues at Brown University [173, 172, 106]. It supports most common math handwriting notations (see figure 2.4 d for some examples) and offers a variety of customization settings to accommodate for user preferences and different handwriting styles. If the system determines that a cell contains a formula, either with just constants (figure 2.4 a) or with references to other cells (figure 2.4 b), the user can toggle between formula view (figure 2.4 b) and result view (figure 2.4 c) by tapping on it. Tableur's evaluation engine computes results of formulas by either directly interpreting ink or by following references over possibly multiple hops. Note that in case of cycles in formulas, illegal operations (e.g., division by 0) or invalid inputs (e.g., a reference to text) the system will display appropriate error messages.

Unlike traditional spreadsheet systems that use an abstract notation for cell references in formulas we implemented a label-based approach in Tableur. Any content that a user wants to reference in a formula needs to have a user-defined handwritten label. Users can label *Freeform Cells* through gestures or convert row and column headers to label-handles. These handles (figure 2.4: ink with gray background) can then be dragged onto other cells to create references. The underlying evaluation engine interprets these references based on the origin of the label and the location of the formula.

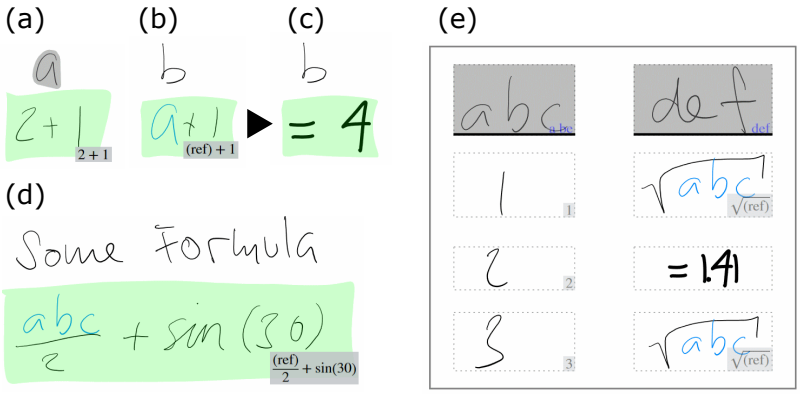

Figure 2.4: Screenshots from the application showing different examples of formulas. a) Labeled Formula in *Freeform Cell*. b) a) Formula including a reference in *Freeform Cell*. c) a) Formula in *Freeform Cell* showing its result view. d) a) Formula in *Freeform Cell* with more advanced handwritten math notation. e) Formulas in a table.

In tables and when created through label-handles of row or column headers the appropriate cells of the table are referenced (figure 2.4e: reference to column "abc") while the same reference in a *Freeform cell* gets evaluated as the sum of all values of the corresponding row or column (figure 2.4d: reference to column "abc"). All formulas are live. Changes to referenced values are directly reflected in the result view of a formula.

2.3.6 Freeform Cell

For many cases creating a full tabular structure feels too heavyweight. Examples include writing down a constant value that will get referenced in other formulas (e.g., π, "Tax Rate") or doing a simple single calculation (e.g., $38 * 140$). For these scenarios Tableur offers *Freeform Cells*, which are created through a simple rectangular gesture, can be placed anywhere on the 2D canvas and contain any kind of formula or value referenced by a label. Their freeform nature can be used to break out of the rigid tabular structure of standard spreadsheets and promote more free flowing working styles where the result of a tabular calculation can be summarized in a single *Freeform Cell* and then reused in other calculations downstream. Figure 2.1 c shows an example of this where multiple *Freeform Cells* are used to first capture an important result of a table (the sum of the "Total" column in the cell labeled "Profit 2016") and then calculate derivatives of it ("Tax Owed" is based on "Profit 2016" and "Tax Rate").

2.3.7 Reverse Editing

The idea of *Reverse Editing* stems from the notion that it is sometimes easier to sketch the shape of data rather than expressing it through other means. In the realm of data this could include statements like "I want my data to be linearly interpolated between a and b" or "Our income will increase in months with higher temperatures". These statements can be easily illustrated by drawing a straight line between a and b or by sketching a curve that starts low, increases constantly over the summer months and then plateaus again. We support this notion by allowing users to edit tabular data directly by drawing on top of line charts to express how they want their data to look like. Figure 2.5 shows an example of this technique. In Tableur dragging and dropping row or column label headers onto the 2D canvas creates simple low fidelity line charts of the corresponding data (figure 2.5 top). By tapping the edit button of a line series users enter the *reverse editing* mode in which lines drawn on top of the chart are interpreted as data-sketches. In the example (figure 2.5 middle) the user indicates that she wants the values of "b" to be linearly interpolated between 0 and 10. The system samples the ink the users drew at regular intervals and updates the values in column "b" of the underlying table. This sampling strategy supports any arbitrary curve to edit entire series as well as smaller scribbles or circles to move individual datapoints. While such sketches are relatively imprecise they still allow for rapid exploration of different what-if scenarios such as "What happens to our profit if our expenses are slightly higher or lower than expected?".

2.3.8 Implementation Details

The current prototype of Tableur supports all the functionality described in this paper, is implemented in C# and WPF (Windows Presentation Framework) and runs on pen- & touch-enabled Windows 8 or 10 devices.

2.4 Discussion and Future Work

We presented Tableur a gestural spreadsheet-like system that is optimized for pen & touch devices. Our system is based on digital ink where users input and edit data through handwriting and organize and annotate ink through a set of gestures. We use handwriting segmentation and recognition to transform digital ink into actionable and computable objects such as tables, constants and formulas. Furthermore we rethink how to fit existing spreadsheet functionality like *Smart Filling* and cell

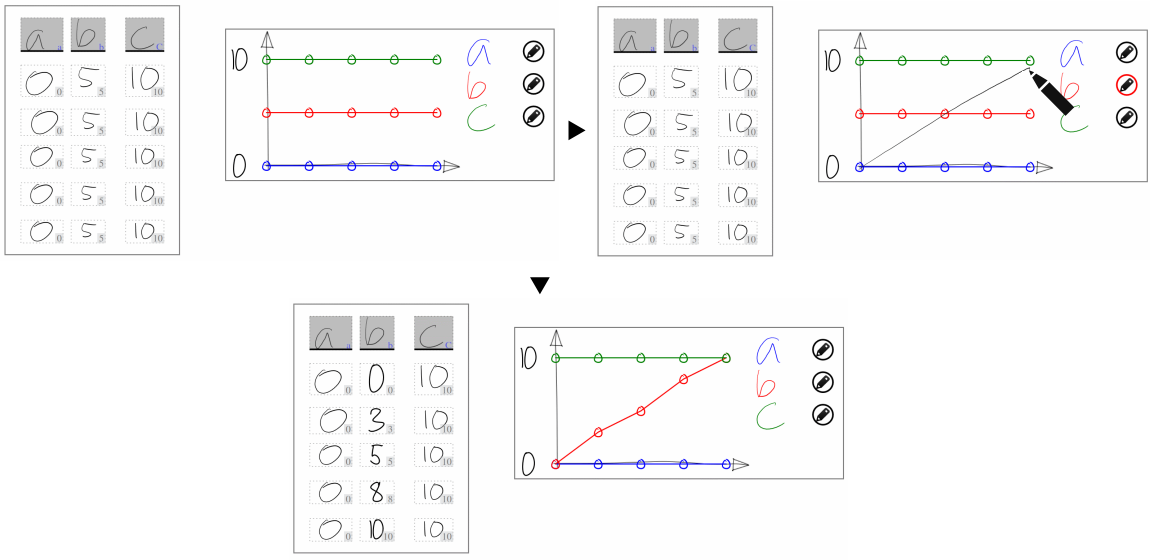

Figure 2.5: Screenshots from the application showing *Reverse Editing*. Top: Table with corresponding line chart. Center: The user wants the values of "b" to be linearly increasing from 0 to 10. She draws a line directly on top of the line chart that indicates this. Bottom: The system has sampled values along the user-drawn line and replaced the values of the "b" column correspondingly.

references for formulas into this new environment and describe novel functionality like *Reverse Editing* that exploits the benefits of a gestural and sketch-based application. Limited initial user testing hints at the benefits our approach might have over traditional WIMP interfaces especially on tablets and interactive whiteboards. However, thorough quantitative user studies are needed to analyze the effects in detail. We are currently planning a study where we will compare our prototype to a traditional spreadsheet system on tablet devices through user performance measures for adhoc back-of-the-envelope calculation tasks. Aside from evaluation work, we also intend to extend our system by including more *Smart Filling* patterns, offer more flexibility when creating formulas by supporting functions and labels to individual cells in tables and porting *Reverse Editing* to different chart types.

Chapter 3

Visual Data Exploration and Analysis through Pen & Touch

This chapter presents a system called PanoramicData, a novel pen & touch system for data exploration and analysis which. PanoramicData provides an approachable visual language that exposes a broad set of the expressive power of SQL, including functionally complete logic filtering, computation of aggregates and natural table joins. This chapter is substantially similar to [176], where I was the first author and was responsible for the research direction, implementation, running and analysis of the user study and a majority of the writing.

3.1 Introduction

Visual data analysis - gaining insights out of a dataset through visualizations is an interactive and iterative process where users need to switch frequently among a range of distinct but interrelated tasks. The set of tasks that recur in visual data analysis, as well as the tools that support them, is well understood [10, 83]. However, designing a system that supports this diversity of tasks in a unified, understandable and approachable way is a non-trivial challenge itself.

Design approaches for visual data analysis typically fall into either the rich-general or strong-specific categories. Rich general approaches are manifest in the form of programming languages, such as SQL or Python, possibly in combination with a general purpose tool like Excel. The syntactic,

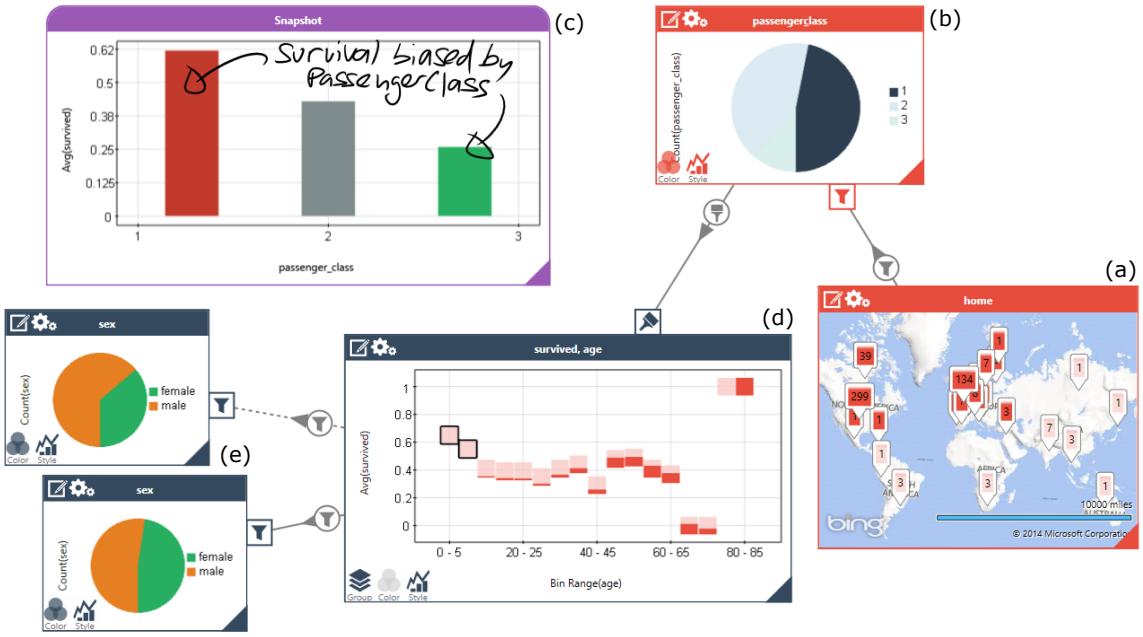

Figure 3.1: Data panorama of the Titanic passenger data-set. (a) Map-view of passenger home towns. (b) Pie-chart of passenger distribution from North-America and Europe (filtered by selection in (a)) across passenger classes. (c) Annotated snapshot of average survival rate by passenger class. (d) Average survival rate for passenger age bins. Brushed by selections in (a) and (b). (e, bottom) Gender distribution for passengers selected in (d). (e, Top) Gender distribution for passengers not selected in (d). Dashed line indicates inversion of selection.

programming nature of these rich general systems creates learning and performance barriers for all but the most dedicated users. Alternatively, strong specific approaches provide more accessible tools, albeit for some narrower problem domain, such as data-transformations [97], data presentations [108], or limited tasks within data exploration [150]. Unfortunately, this approach requires users to adopt cumbersome workarounds, such as exporting data to other programs, to complete many common tasks that fall outside the systems target domain.

Inspired by the metaphor of narrative panoramas, our research attempts an interactive rich-general approach by providing a small set of simple visualization primitives which can be linked in very direct, concrete ways through Boolean operations on an unbounded canvas to create sophisticated visualizations. These dynamic panoramas are expressive enough to represent interactive visualizations for the scope of tasks required for detailed exploratory analysis of multi-dimensional structured data. These dynamic panoramas are expressive enough to represent interactive visualizations for the scope of tasks required for detailed exploratory analysis of multi-dimensional structured data. Although this approach requires that users understand how to piecewise compose a complex visualization, we believe even beginning users can develop this mastery with minimal training because it corresponds to the incremental way in which questions generally form. For example, a complex query might germinate from a simple question about a dataset, such as "how many women died on the titanic?" Followed by, "show their age distribution," and then by "compare those distributions by berthing class," and so on. The totality of such a visualization chain can be a rather complex acyclic graph. However, the panoramic approach to visually representing this working set of composed queries is powerful [24, 51]. Not only is the step-by-step creation process approachable, but it also affords interactive "handles" at each query stage to explore tangential queries, such as switching from women to men in the example, or to simply verify that the question had been posed correctly. During our iterative design of PanoramicData (PD), our embodiment of this rich-general approach, we referred to Heer & Shneidermans [83] taxonomy of interactive dynamics for visual analytics as well as the analytical task taxonomy of Amar, Eagan & Stasko [10] as a guide to ensure that PD broadly covers exploration and analysis tasks. We observed that in order to support those tasks in a comprehensible and unified way our most effective design choices clustered around a set of four concepts (Derivable Visualizations, Exposing Expressive Data-Operations, Unbounded-Space, and Boolean Composition). These concepts, which all have been discussed to some extent in the literature, both guided and encapsulated the critical reasoning behind our design decisions. Although

we found that each of these concepts typically allowed for multiple different equivalent designs, omitting any concepts had a significant negative impact on either usability or visualization power. The strength of PD lies within the effective combination of those four concepts. We discuss these concepts in detail and describe how these concepts, and compounds of them, provide a framework that supports a wide range of data-exploration and analysis tasks.

A large part of our design efforts were targeted towards offering a fluid interaction model [54] which serves as the overarching structure of PD and unifies those four concepts in a comprehensible way. Inspired by recent work suggesting the benefits of pen and / or touch for analysis work [27, 49, 108, 165], we specifically designed PD for interactive whiteboards and pen-enabled tablets. By considering what can easily be accomplished through the use of pen and touch, we were able to combine the aforementioned concepts in a way that avoids explicit mode switching, minimizes UI cluttering, reduces indirection in the interface and provides effortless switching between the variety of interrelated tasks that data exploration and analysis require. Even though we emphasize the modalities of pen & touch, the entire UI could still be conveniently operated through standard mouse and keyboard interaction.

In this paper we present PD a novel pen & touch system for data exploration and analysis which is based on four core concepts, Derivable Visualizations, Exposing Expressive Data-Operations, Unbounded Space and Boolean Composition. The combination and interaction between those concepts presents an approachable interface that allows for incremental and piecewise query specification where intermediate visualizations serve as feedback as well as interactive handles to adjust query parameters. PD supports a wide range of common tasks.

We evaluated PD through a formative user study with both data and visualization experts. The results support our belief that PD, and the combination of the design- concepts it embodies, provides a fluid and intuitive user experience while still being expressive enough to answer the range of queries that users are likely to pose.

3.2 Related Work

We relate and contrast our work to research efforts in the areas of Pen & Touch Visualizations, Linked and Coordinated Views and Database Interaction.

3.2.1 Pen & Touch Visualization

Our work has been inspired by the research of Elmqvist et al. [54] and Lee et al. [107], which emphasize the importance of interaction to Information Visualization and propose to investigate new interaction models that go beyond traditional WIMP (windows, icons, menus, pointers) interfaces. More specifically, [27, 108, 165] have transposed insights about the usefulness of whiteboards in supporting thinking, collaborating and general problem solving processes [122, 164] to interactive whiteboards; an emerging class of hardware. SketchStory [108] focuses on the insight dissemination aspect of data-analysis. It supports presentation of pre-recorded data-related insights through a gestural pen & touch UI. Another pen & touch based whiteboard UI is presented in SketchVis [27]. It allows users to draw and label charts which the system fills in correspondingly and offers gestures to filter objects. Furthermore it offers a set of mnemonic gestures to specify data-transformations and mapping of visual elements. While similar to our work, we focus less on gestures for chart creation and more on gestural approaches to coordinate multiple visualizations.

Another direction of recent research investigates the usefulness of touch interaction for data exploration and analysis tasks. The TouchViz paper [49] presents FLUID, a touch interface for manipulating data-visualizations. The techniques presented focus on a single visualization rather than on a network of linked and coordinated views and filters.

3.2.2 Linked and Coordinated Views

The notion of coordinating visualizations manually in order to construct custom exploration interfaces has been introduced by [126]. SnapTogether Visualization allows users to coordinate views in order to support a set of common tasks, such as Brushing-and-linking, Overview and detail view, Drill-down, Synchronized scrolling and Details on demand. While this idea serves as a fundamental building block in PD, we expanded this concept through exposing finer-grained control in view coordination by allowing views to have multiple inputs that are combinable through Boolean operators and by propagating filtering operations across multiple hops. By rethinking the concept of snappable visualizations in terms of a gestural UI and by abstracting the underlying database-schema we are able to reduce the mental overhead that the heavy-weight view-coordination dialogs of SnapTogehter Visualization impose on users.

GraphTrail [51] is a system that allows exploration of large network datasets while preserving

exploration history. The system is optimized to work with network oriented data such as social networks and scientific collaborations. One of the core-concepts of Derivable Visualizations, being able to create visualizations out of existing ones in order to create query-chains, has been explored by GraphTrail. However, GraphTrail displays these chains statically whereas PD offers more flexibility through interactive modification of these chains and their elements. DataMeadow [55] presents an interactive visual analytics system based on DataRoses; a parallel coordinate starplot that exposes filtering along its axes and linking. We expand on their notion of linking by providing different types of links and base our implementation on familiar charts that allow for a higher degree of customizability and data-transformations.

Yuan et al [171] introduce a system that allows users to build up a visualization tree through a divide and conquer strategy. While this approach is similar to our piecewise query specification, it does not allow for manual rewiring of linked visualizations nor does it support data-transformations such as grouping or aggregation functions. Lark [155] links visual elements through a meta-visualizations which supports the creation of multiple variations of a view. However, linking between views from different data-attributes is not possible and it also supports limited data-types and operations. The Stacknflip UI presented in [151] uses links between visualization to show exploration histories and to guide users through an analysis based on a pre-defined setup model. VisLink [37] presents a visualization technique where 2D visualizations can be organized in 3D planes and relationships between views are displayed through edges. VisLink however does not address view-creation or data-transformation.

PDs filter-chains are comparable to dataflow networks. It is therefore similar to [4, 160], but offers specialized building-blocks which are targeted towards data-centered tasks.

3.2.3 Database Interaction

Tableau Software and its research predecessor Polaris [150] are systems to visually analyze large datasets. They offer support to create visually appealing and print-ready visualization of large datasets with a high degree of customizability, but have limited functionality to link multiple visualizations together.

dbTouch [89] presents a UI that enables users to touch and manipulate data intuitively and suggests the need for new database systems which are optimized for fluid interaction with data. While its UI is limited in its support for common analysis tasks and does not expose familiar

visualizations, it coins the term Schema-less Querying. PD supports this notion of abstracting the complexity of the underlying database schema by offering implicit table-joins.

Approaches to create visual query languages have been an active research topic [4, 170]. Those systems have limited ability to incorporate interactive visualizations or coordinate visualizations and often require users to understand the underlying database-schema in detail.

3.3 Core Concepts

PD's design emerged from a series of implementation iterations in which we explored the relative value of exposing different levels of data functionality, and of different interaction styles and techniques for creating customized, interactive and coordinated visualizations. Our intent was to create a system that was approachable and predictable for untrained users yet comprehensive enough to support the complex queries of advanced users. Over time, we observed that our more effective design choices clustered around a handful of concepts, all of which have been discussed to some degree in the literature, and that by being aware of these patterns, we were better able to identify and prune design alternatives. Ultimately, we resolved upon a set of four core concepts which we feel depict the fundamentally important characteristics of our approach.

3.3.1 Derivable Visualizations (C1)

Since visualizations are the focal components of any visual analysis system, users spend significant effort creating views of underlying data attributes. While several methods address this view-specification step [83], each with their own set of benefits and trade-offs, we believe, similar to approaches exposed in GraphTrail [51], that visualizations must additionally be derivable. Derivable visualizations are new, tangential visualizations made by directly referencing or reusing part of existing visualizations. By effortlessly deriving visualizations from existing ones, users can fluidly explore what-if scenarios of related data without disrupting the relevant context of their existing visualizations. In addition, derivable visualizations can reduce the complexity of creating an initial visualization, since modifying is typically easier than creating. In some ways, they become a form of "suggested visualization" [74]. An important consideration for derivable visualizations is to maintain visual consistency (e.g., same color or size) between data attributes that are shared in different visualizations.

3.3.2 Exposing Expressive Data-Operations (C2)

Manipulating data into a format capable of visualization requires data transformation (e.g., aggregation, grouping-operations) and derivation tools (e.g., calculating new field values from existing data) tools [150]. Further, these tools need to be available directly in the context where they are needed to avoid the cognitive disruption of searching. In the limit, every possible transformation could be needed at any point, requiring the full power of a complete scripting or programming language. Although such completeness should be a goal, providing a sufficiently large subset of possible data-operations may be sufficient, particularly if achieving completeness comes at the cost of obscuring or confounding the more frequent simple tasks.

3.3.3 Unbounded Space (C3)

Managing display space is a requirement of any interactive system. However, for cognitively heavy-weight tasks, the distraction of having to switch between views to see the components of a working set of information can significantly impact task performance [24]. By providing unbounded space including simple techniques for managing that space, several critical analytic tasks can be simplified to implicit, familiar activities. Multiple visualizations can simply be juxtaposed for comparison; reasoning chains can be reviewed to validate results; and maintaining prior queries in the periphery provides temporal context [17, 51].

3.3.4 Boolean Composition (C4)

They key concept for creating visualization networks is that each visualization's output must be combinable with the output of other visualizations [126]. We believe that having simple operations between visualizations affords complex results while imposing only the cognitive burden of understanding basic Boolean logic [143]. This requirement is not equivalent to typical data flow because what flows between visualizations is not data records themselves, but rather the data selection specifications used by the visualizations. That is, one visualization showing a filtered selection of data column "x", could be linked to another visualization showing data column "y". With data flow, this wouldn't make sense, but with our notion of composition, the second visualization would do the equivalent of a database join of column "y" with column "x" and then apply the "x" column filter before projecting to just column "y". This approach allows for multiple visualizations to be

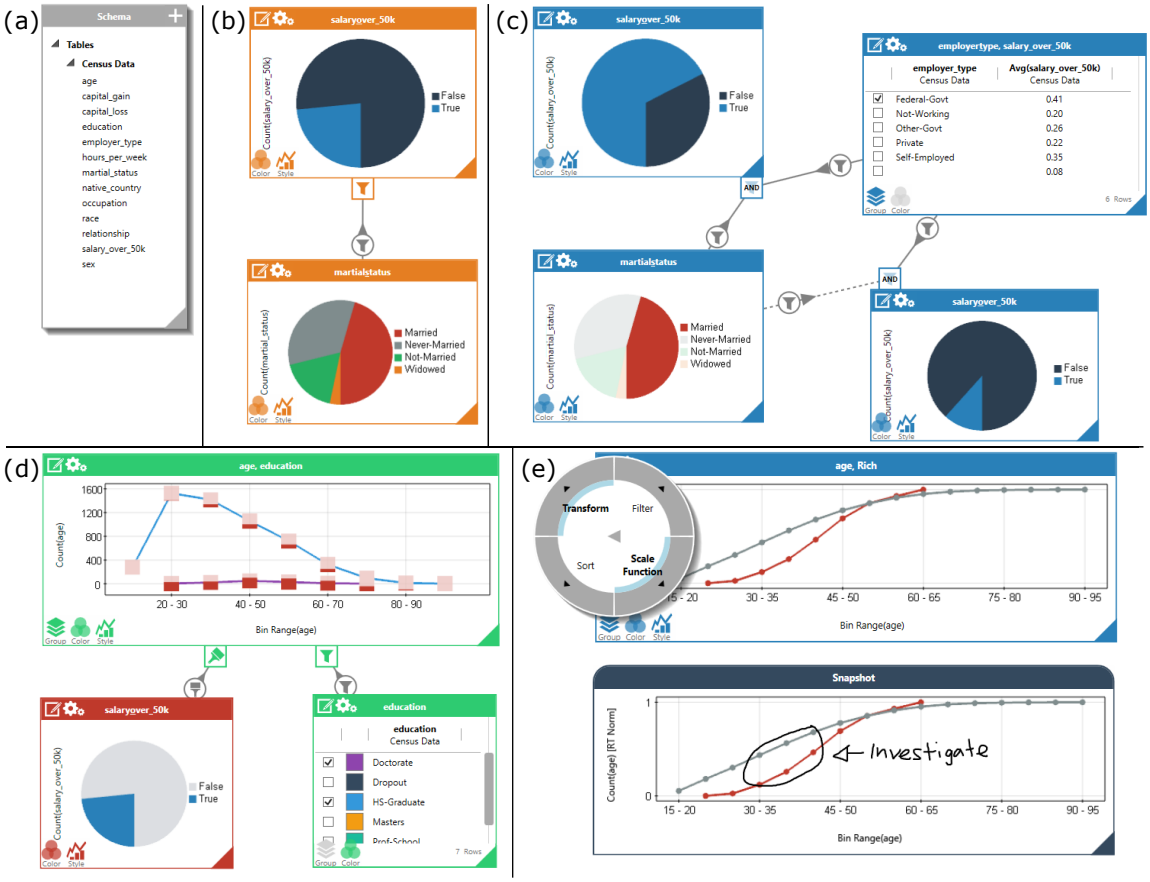

Figure 3.2: Different parts of a data panorama create by a user exploring Census data. The different parts are described in Section 3.4.1.

linked to a single new visualization, where Boolean logic operators determine which data records the new visualization receives. For generality and in particular to support the notion of visualization brushing, we extend this composition requirement to include an operator for preserving all the input data selection specifications as an array instead of always combining them with Boolean logic into single data selection specification.

3.4 The PanoramicData Prototype System

We developed PD, the embodiment of a rich-general approach to data analysis which unifies the aforementioned four design concepts in a comprehensible way, through an iterative process in which we steadily added and refined features in order to support a wide range of exploration and analysis tasks. We will motivate our approach through an introductory use-case and will then highlight

Data & View Specification	**Visualize** data by choosing visual encodings. **Filter** out data to focus on relevant items. **Sort** items to expose patterns. **Derive** values or models from source data.
View Manipulation	**Select** items to highlight, filter, or manipulate them. **Navigate** to examine high-level patterns and low-level detail. **Coordinate** views for linked, multi-dimensional exploration. **Organize** multiple windows and workspaces.
Process & Provenance	**Record** analysis histories for revisitation, review and sharing. **Annotate** patterns to document findings. **Share** views and annotations to enable collaboration. **Guide** users through analysis tasks or stories.

Table 3.1: Taxonomy of interactive dynamics for visual analysis [83].

PDs key features in the context of their relation to the four concepts and how they are used to perform common exploration and analysis tasks. We will point directly to specific items of Heer & Shneiderman's (HS) taxonomy of interactive dynamics for visual analysis [83] (Table 3.1).

3.4.1 Introductory Use-Case

Figure 3.2 shows different parts of a rich, dynamic visualization panorama that a user constructed with PD while exploring and analyzing a random sample of census data (10,000 records). By double-tapping on the background the user opens up the schema-viewer which displays all the attributes of the data-set (Figure 3.2 (a)). The user drags two attributes, marital-status and salary-over-50K, out of the schema-viewer and drops them anywhere on the 2D canvas (C3). To analyze the relationship between the two attributes the user uses the pen to connect the two (C4) (Figure 3.2 (b)). Tapping on a slice in the marital-status pie-chart filters the second pie-chart to only show those records that satisfy the marital-status selected in the first chart. He toggles through the different marital statuses to observe if any of those have a higher chance of earning more than $50K annually than the rest. By dragging and dropping he derives two new visualizations and connects them again by using the pen (C1) (Figure 3.2 (c)). He has now arranged (C3, C4) a custom exploration interface that lets him analyze the correlation between employer-type and marital-status and how those reflect on people's annual salaries. He gains the insight that people who are married and work for the federal government have a significantly higher chance of earning more than $50K than other groups. By

flipping one of the visualizations around and using handwritten input he tags people in that group with the keyword "rich" (C2). This will facilitate later referral to this insight.

The user wants to explore some more attributes of this data-set; age and education-level (Figure 3.2 (d)). He draws an "L" type-shape (indicating X and Y axes) on the canvas and drops attributes from the schema-viewer to label the axes of his graph. By writing a "C" onto the label of the Y-axis he transforms the data to display the aggregate-count of the attribute instead of the raw value (C2). He changes the grouping context of this aggregate function to use a binning strategy instead of distinct values. He now sees the distribution of people's ages. By dropping another attribute onto the graph's color drop-target he splits the rendering into separated series. Now the chart visualizes age histograms for each education-level in a different color. To understand the meaning of the colors and to be able to filter on specific series he derives a legend-visualization by dragging off the color icon. Once again, he connects the visualizations with the pen so that he brushes the chart to see the probability of earning more than $50K individually for each data-point (C4). Doctorates are above this threshold even early-on in their career, while most High-school graduates are below it.

Our user creates another similar chart by first using a multi-touch copy gesture and then modifying it through drag and drop operations. He has colored this second chart by dropping his previously created "rich" tag onto the color-drop target. He now sees two age distributions, as a running total after applying a data-transformation through a radial menu accessible on the Y-axis (C2) (Figure 3.2 (e) top), one for people that match his tag and the second one for people who do not match his tag. By additionally selecting people who are self-employed in the tagged query he updates his definition of the "rich" tag and the age-histogram chart updates accordingly (C2, C4). To preserve an interesting insight seen in this chart he creates a static snapshot of it (C1) and annotates it with handwriting ((Figure 3.2 (e) bottom).

3.4.2 Pen & Touch and Gestural Interaction

Many analysis systems [150] offer a rich set of functionality that is exposed trough traditional WIMP (Windows, Icons, Menus, Pointer) interfaces and are prone to come with the drawbacks of this UI-metaphor [161]. The interaction design of a data analysis system should reduce the mental overhead and should not distract from the fundamental task. As others have argued for Information Visualization in general [54, 107], we believe that especially visual data analysis systems could heavily benefit from advances made in interaction technologies. Being able to offload user interface reasoning

(e.g., when searching for a tool) to sub-conscious natural interactions promises a significant benefit for data analysis work which is already cognitively overloaded. Gestural interactions for managing space with touch gestures are well known and effective. However, touch alone is not expressive enough to disambiguate certain interactions, such as performing a selection on data vs. moving the container of the data. By combining both pen and touch in the UI, we believe expressive power can be enhanced without increasing cognitive load since users can learn to subconsciously associate certain interaction, such as manipulation with touch, and others such as region selection with the pen. In addition, pen gestures offer the possibility of affording very efficient, easily remembered modeless shortcuts for abstract operations, such as writing a sigma symbol over a column of data to view its sum (C2). To achieve the cognitive offloading benefits of gesturing, interactions must be consistent across the entire system so that users will feel comfortable performing interactions with just muscle memory.

By considering what can easily be accomplished through the use of pen and touch, we were able to focus on an uninterrupted experience. Having two input modes eliminates the need for many mode changes and provides a consistent selection metaphor. Recognizing pen-input also obviates keyboard-input and reinforces PDs whiteboard metaphor. Using a pen to handwrite annotations on the background or on a snapshot of a visualization feels natural (HS "Annotate"). The pen is also used when precise control is needed, for example when selecting only a few data-points within a large scatterplot or when pointing to a narrow slice in a pie-chart (HS "Select", "Filter") We also use the pen to enable fluid interactions through shortcuts of commonly used actions within the system. Two visualizations can be linked together by drawing a line between them (C4, HS "Coordinate") or sorting within a table-view is performed by performing an up- or down-flick gesture (HS "Sort") similar to [133]. Additionally, selections within a visualization can be inverted by using a flick-gesture or transformations to data-attributes can be performed through a set of symbolic gestures (C4, HS "Filter", "Select", "Derive"). Similar to [27] PD offers pen-gestures to create new views. An "L"-shape drawing on the background is used to create scatter-, line- or bar-charts, a circle gesture for pie-charts and a rectangular gesture for tables. Finally, PD uses a scribble-erase gesture as a way to support deletion of unwanted content or links [174].

Touch allows for a fluid interaction with all visual elements within PD. For example all the drag & drop operations that PD expose can be performed through touch-interactions. We make extensive use of drag & drop operations in order to create, derive or modify visualizations (HS "Visualize").

Additional touch gestures include, a double-tap gesture anywhere on the 2D canvas go gain access to all attributes of the dataset, a press-and-hold gesture to get quick-previews of visualizations within a table-viewer or a two-finger gesture to create copies of visualizations. We also expose well-known direct-manipulation touch gestures to pan and zoom the 2D canvas or to change the viewports of visualizations (C3, HS "Navigate", "Organize").

3.4.3 SQL Mapping

PD operates directly on relational-databases. To make a dataset accessible within PD it needs to be annotated with a small set of meta-data. This includes explaining relationships between different tables (cardinality of relationship, primary and foreign keys), providing some information about the columns of the tables (data-types, preferred visualizer, aliases and human-understandable labels if needed) and meta-data about the tables themselves (aliases). All the data-related operations within PD are mapped to their corresponding SQL function. PD is therefore a visual language to SQL that exposes a broad set of its expressive power (C2, HS: "Derive"). Database-joins are done implicitly when needed and rely on the meta-data information (i.e., primary-foreign key relationships between tables). Users do not need to struggle with the complexity of SQL-joins, but this also limits PD to only expose part of the full expressive power of SQL (i.e., only natural joins). Different join types or joins on non-key-columns of tables are not supported yet. We plan to expose that functionality as part of our future modifications.

Within the system each visualization is represented as an abstracted model. This model includes the data-attributes needed to construct the visualization, the transformation applied to them, the incoming filtering or brushing relations, the items currently selected in the visualization, as well as a few visual properties that do not affect the SQL layer (e.g., rendering style or color mappings). It is important to note that no data flows between linked visualizations; instead they share information contained in their abstract visualization models. Triggering operations that affect the abstract model forces a particular visualization to refresh by generating and executing SQL queries. PD does not perform any data-related computations in memory; all computations are delegated to the underlying database system.

3.4.4 Schema-Viewer

Users need to be able to access attributes of the data-set. While being widely used and powerful, relational databases come with the potential caveat of having a complex schema that are hard to grasp for untrained users. Finding an intuitive and approachable way to expose the structure of the underlying database-schema has been a challenging aspect of the PDs design. Since database-schemas are essentially graphs, there could be multiple ways to "connect" two attributes from different tables. This ambiguity needs to be solved in order for the system to automatically assemble SQL joins. Consider the database-schema in Figure 3.3. Imagine a user wants to create a visualization that displays an attribute from the "Player" table and an attribute from the "Team" table. Without specifying a path that encapsulates how the two tables should be connected the system does not know which of the three possible options the user intended (Player Contract Team, Player Game Team (Home) or Player Game Team (Away)). Each of the paths has a semantically significant different meaning. The same problem arises and is further complicated when linking two visualizations together. In the case that a user is looking at two visualizations - one that displays a list of players and one that displays a list of teams - the user must connect the two together to perform a filter or brushing operation. Without further specification the meaning of this connection is again ambiguous.

In our current version we address this problem by presenting the database-schema in a tree-view that is always rooted at the same table. If a user now drags an attribute out of this tree-view the path of how to connect the root table to the table from which the attribute comes is unambiguous. The same applies for linked visualizations. Because they are initially created by dragging attributes from the schema tree-view we can always use the root-table as the lowest common denominator when performing SQL joins. This approach offloads the problem to choosing a root table. Databases that are laid out in star-schema form, with a single fact-table, are therefore a naturally good fit for PanoramicData. It intuitively makes sense to use such a fact-table as the starting point and therefore as the root element in our schema-viewer. In other cases, we let the user choose a root-table at the beginning of an exploration or analysis session. Our current prototype does not support re-rooting during a session and therefore diminishes some of the expressiveness of SQL in order to reduce the cognitive burden on users. The schema-viewer in PD (Figure 3.2 (a)) is brought up by a double-tap gesture anywhere on the 2D canvas. This allows users to quickly gain access to the

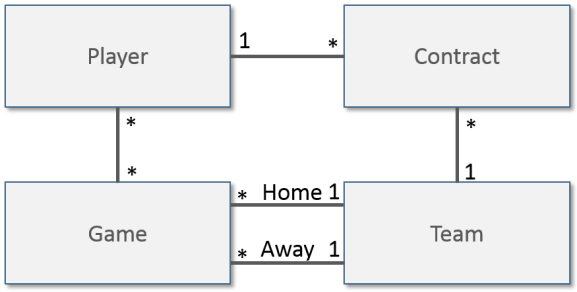

Figure 3.3: Simplified database-schema for sports statistics.

underlying dataset in whatever context they currently are. Dragging and dropping data-attributes from the schema-viewer is used throughout the system to create or modify visualizations.

3.4.5 Data-Transformers

Data-transformers, such as group-by aggregates (e.g., sum, count, max, min, avg) or sorting operators, are applied directly to every data-attribute in PD and can immediately update the rendering of a visualization (C2, HS: "Derive"). PD exposes data attributes wherever it makes intuitive sense for a given visualization. Scatter-plots for example place visual handles to their data-attributes on the X- and Y-axis, whereas table-viewers expose them as column-headers. Common transformers can be triggered through a set of gestures (see section 3.4.2) but are also accessible through a more traditional radial-menu for discoverability. Additionally each visualization exposes two drop targets: a group target and a color target. Dropping attributes on the group target specifies the context of aggregate calculations (e.g., what are we summing over?), whereas the color target is used to specify coloring of data-points within the visualization.

3.4.6 Calculated Fields

PD offers support to use well-known mathematical notations [173] to create calculated fields from any attribute (C2, HS: "Derive"). This functionality is exposed through the schema-viewer. A calculated field is no different than other attributes and can be used in exactly the same way as data-attributes to create or modify visualizations or to calculate group-by aggregates. PD further allows the user to create new attributes by transforming a chain of visualizations into a Boolean attribute that indicates if a data-record is part of this sub-set. This technique, described as tagging in the introduction use-case, is useful to create custom groupings within a visualization or to condense

a complicated filtering operation into a single attribute. Again, such set-attributes can be used as any other data-attribute, for example for coloring a graph or for filtering or brushing a visualization.

3.4.7 Zoomable Canvas

PD features an unbounded pan- and zoom 2D Canvas in which visual elements can be arranged in a free-form fashion (C3, HS: "Coordinate", "Organize"). Such a 2D canvas offers a couple of advantages. Firstly, users can manifest their elements in a way that matches their mental model. Secondly, logically corresponding elements can be arranged spatially near each-other without forcing them into a limited area. Furthermore, open space to fluidly explore what-if scenarios or tangentially related data-attributes is available within a set of simple pan and zoom gestures. Finally, this whiteboard metaphor offers an intuitive way to label findings or important parts of the exploration or analysis process by using handwritten annotations on the background (HS: "Annotate"). Figure 3.1 (c) provides examples of such annotations. This approach also offers a way to record the users exploration history. The filter-chains that are constructed during the exploration process can be conveniently revisited by locating them on the 2D canvas (HS: "Record"). While this is not as structured or automated as [82, 144] it enables users to decide which part of the exploration history they want to keep and allows them to layout and annotate exploration histories in a free-form fashion. Even though PD works well on pen-enabled tablets, it leverages this whiteboard metaphor best when used on an interactive whiteboard. In a single-user scenario the extra screen-real-estate is useful for exploring visualizations in full detail. Furthermore, in collaborative-scenarios a whiteboard offers a natural way of compiling or discussing data insights (HS: "Share") and it turns PD into a tool to disseminate knowledge in presentation scenarios while benefiting from its interactive nature to quickly answer questions from the audience similar to [108].

3.4.8 Creating Visualizations

PD's central visual elements are visualizations and creating them is one of the most common tasks within the visual data analysis process (HS: "Visualize"). We encourage fast and easy creation of visualization through drag & drop interaction (C1). Every visual element that represents a data-attribute can be dropped anywhere on the 2D canvas to create a default visualization of its values. Similar to [118], the default visualizations are based on simple heuristics such as the data-type of

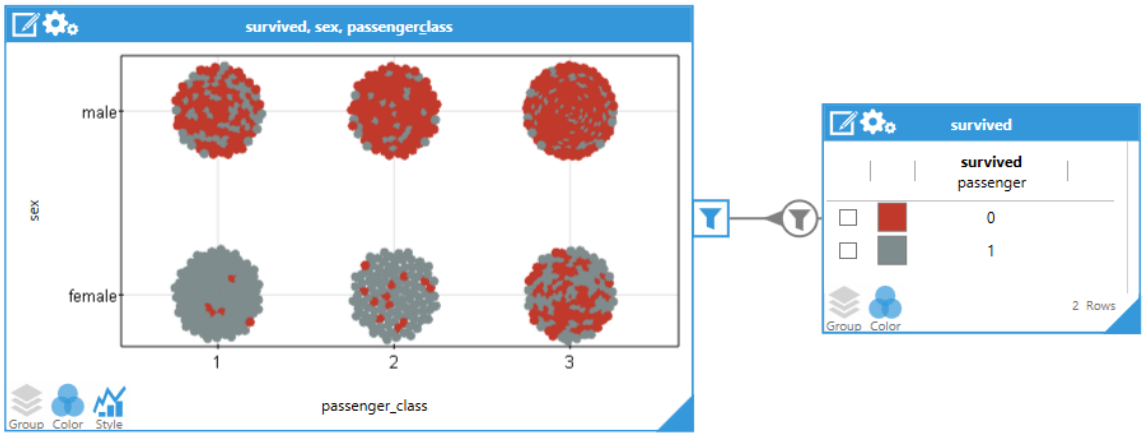

Figure 3.4: Scatter-plot with interactive legend directly derived from plot.

the attribute and the number of unique elements in the data and can be changed by adding hints to the pre-defined meta-data information. A geographic attribute, for example, is displayed in a map-visualization whereas a categorical or binary attribute gets rendered as a histogram showing the distribution of its unique values. PD supports a manageable set of familiar visualizations such as pie-, bar-, scatter- and line-charts as well as table- and map-viewers, but other visualizations could be included. Each visualization can be customized through dropping data-attributes on pre-defined drop targets. 2D plots for example expose drop targets on their X- and Y-axis or on a special color shelf. On the other hand, a table -viewer allows dropping of attributes anywhere on its column-headers to extend the columns they show.

Following our design concept of "Derivable Visualizations" (C1) led to some interesting design choices. For example we completely eliminate any sort of built-in legends within a visualization. Legends instead are visualizations themselves and can be derived from an existing visualization by dragging from its color drop-target. Legends are therefore fully interactive [135] and can be further customized if needed. This function works in part because PD offers consistent coloring of data-points across the system even for visualizations that are not linked. Figure 3.4 exemplifies such a legend.

Since Grammel et al. [74] showed that users of visualization systems often thought about data without any "processed" visual structures or visual attributes, we also include a gesture for deriving a table-view from any visualization (C1). Exposing the raw-data can serve as a valuable validation mechanism to avoid misinterpreting complex visualizations, and can be a more effective way to

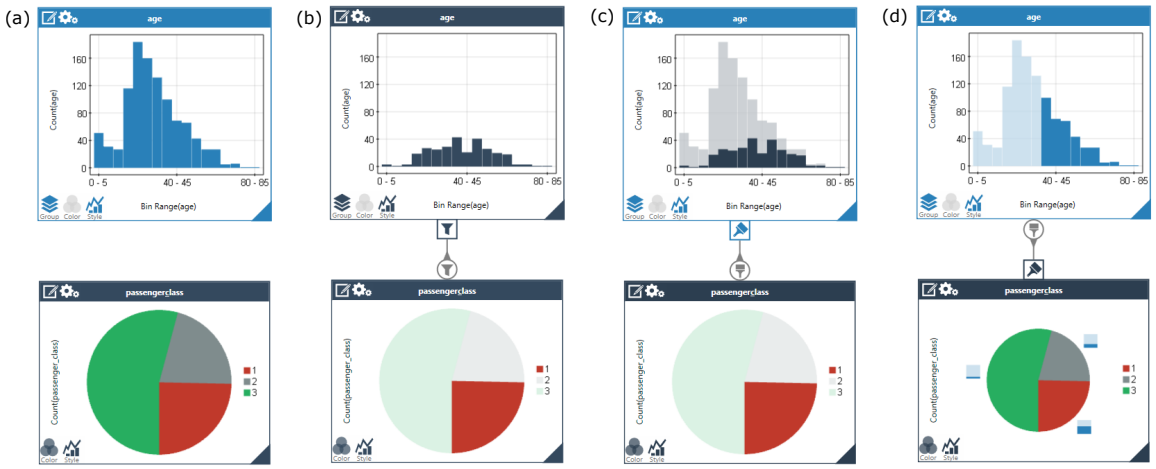

Figure 3.5: Four ways to compose two visualizations to show different relationships between attributes. (a) No relationship. (b) Top filters bottom. (c) Bottom brushes top. (d) Top brushes bottom.

identify certain patterns or to perform computations.

3.4.9 Linking

Being able to coordinate views is a powerful tool that addresses a lot of different common exploration and analysis tasks (C4, HS: "Coordinate") In PD two visualizations can be linked through a one-way directed connection. Links indicate that the target visualization is influenced in some form by the source visualizations. A link can be in one of two states: filtering or brushing. When using filter-links the target visualization is filtered to the set of items that is selected in the source visualization. A visualization can have multiple incoming filter-links and the user can choose how to combine filters (AND or OR). Additionally the output of any filter is invertible (NOT). Thus PD offers functionally complete visual filtering logic. In Figure 3.4.1, the middle pie-chart displays the distribution of passenger classes for passengers that are NOT over 30 (selection in the top left age-histogram and inverted filter-link visible by the dashed line) OR are males. Brushing-links are used to highlight the same data within a different context. In simple cases, for example when two scatter-plots showing the same data points but on different axes are linked together, this enables standard brushing and linking, where a selection in one chart highlights the corresponding data points in the other. Figure 3.4.1(e) shows a more complex example. The top plot is brushed by the purple pie-chart on the bottom. Notice that because the plot displays aggregated data-points (passengers are binned by

age ranges) brushing is more nuanced. It reveals what percentage of the passengers within the aggregation would fit the query specified by the purple filter. It is also important to mention that the purple visualization is fully interactive and that a selection change would automatically update the dependent visualization. A user can switch between those two link states (filter or brushing) through simple touch or pen gestures. This feature of PD provides a powerful tool for fluid data-analysis. Even just two simple visualizations of two attributes can be examined in a variety of ways. Consider a user trying to find an unknown relationship between two attributes, age and passenger-class, of passengers on the Titanic (Figure 3.5). The user starts with two visualizations (Figure 3.5 (a)), one showing the age histogram of all passengers and the other depicting their distribution according to passenger-classes. These are the default-visualizations for those attributes, which were attained by dragging the corresponding attributes from the schema-viewer. A pen gesture connects the two visualizations. Touching the slices of the pie-chart selects them and filters the age-histogram. After swiftly toggling through all the different passenger-classes the user thinks they noticed a slight shift in distributions for the first class (Figure 3.5 (b)). The user changes the link to a brushing-link and one now sees the age histogram of first class passengers relative to all passengers (Figure 3.5 (c)),. Indeed, it seems that there may be a shift in the distribution. With another pen-gesture the link flips in the opposite direction and the user pen lasso-select the passengers over 40. The little bars on the side of the pie-slices show the number of passengers within the class that the selection in the age histogram (Figure 3.5 (d)). It appears that older passengers were most likely in the first class.

Notice that PD allows configuration of these transformations within this scenario with one or two touch and pen gestures.

3.4.10 Copying and Snapshotting

PD offers two ways to create copies of visualizations. The first-one derives an exact live copy of the visualization that is fully interactive. The second-one takes a static snapshot of the visualization that can then be annotated (HS: “Annotate”, “Record”). We included this feature based on feedback we received through our user-study. In some cases user wanted to “keep” or “save” a visualization and disable automatic updating through filtering or selection changes.

3.4.11 Selections

All visualizations manage selection states. Selections in PD are represented as queries over the data (HS: "Select"). For 2D plots those queries are represented as ranges over the space of the data-attributes. This allows the system to keep selections even if the underlying data has changed (e.g., through a filtering operation). To speed up the calculation of those selection queries, especially for multi-dimensional-charts (X, Y and color) with a large number of data-points, we use spatial data-structures (i.e., Octrees). 2D chart types, such as scatter-plots or maps, allow free-form lasso selection with the pen to allow for fine-grained selection control

3.4.12 Scalability

Our system currently works best for databases up to 30,000 records. Performance of rendering and database querying, especially for large visualization networks across multiple database-tables, drops to a non-interactive level for data-sets with over 80,000 rows.

3.5 Evaluation

We evaluated PD through a formative user study with both data and visualization experts. The goal of this study was to understand PDs utility and its approachability, to gain some insight about how its features are used and to explore the type of queries user pose. The study involved five participants, three PhD students (conducting research in the fields of Databases or Visualization) and two advanced Undergraduates (Teaching Assistants for a Data Science class). None of the participants had any prior knowledge of our tool.

The results of this study support our belief that PD, and the design- concepts it embodies, provides a fluid and intuitive user experience while still being expressive enough to answer the range of queries that users are likely to want to pose. The study also suggests that users are able to use the system with a minimal training amount.

3.5.1 Procedure

Our participants were given a 10 minute introduction to PD. In this introduction we used a small example data-set and gave the participants a brief overview of all the features and gestures in the

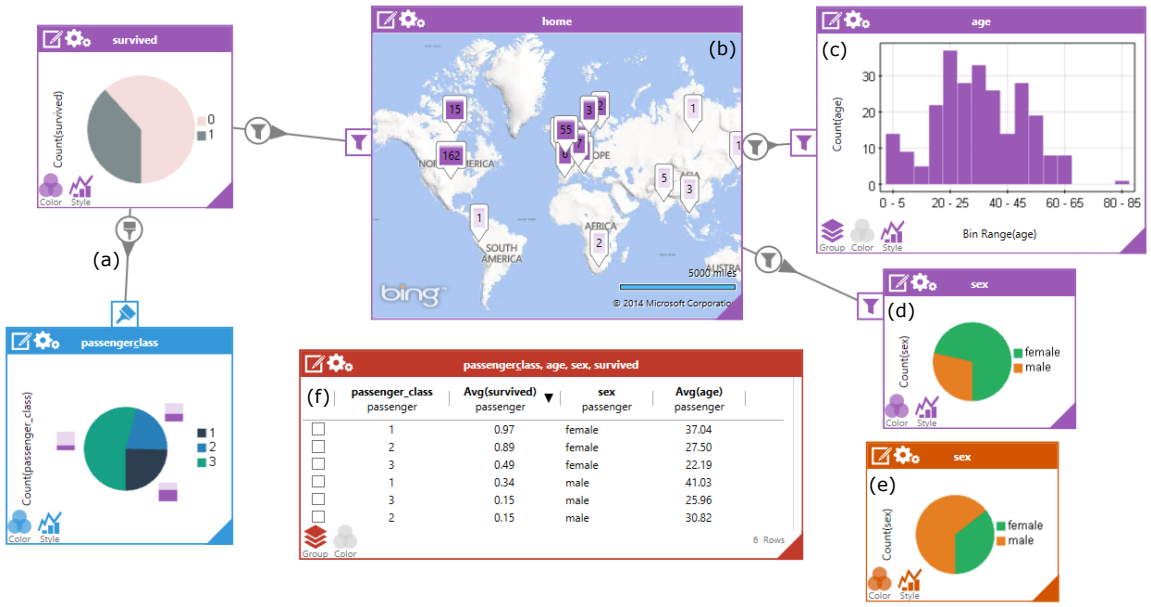

Figure 3.6: Data panorama that a user built up during our evaluation to investigate different aspects of the Titanic data-set. (a) The user started of exploring the relationship between passengers that survived and their passenger class. (b) He then tested different hypothesis (i.e., are there correlations between survival and home towns of passengers (b), their ages (c) and their gender (d). He kept an unfiltered distribution of gender (d) to compare the ratios. (f) In order to obtain more accurate numbers and to confirm what he visually inferred, he built a table containing average survival rates for each passenger class and gender combination.

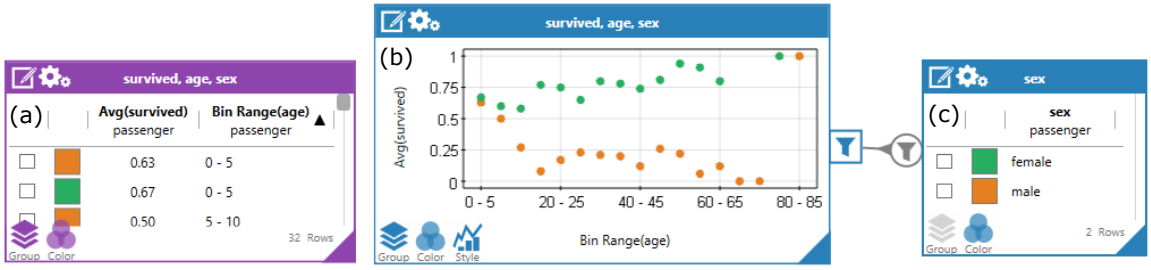

Figure 3.7: An example from our evaluation where a user first described the data he wanted to see in tabular form (a) and then created a chart by dragging and dropping the column headers to the x and y axis (b). To understand the colors he derived a legend (c) from his chart.

system. After that we briefly allowed the participants to familiarize themselves with the tool and the UI, before we exposed them to the Titanic dataset. This dataset contains 14 attributes about the passengers of the Titanic [86]. We instructed the participants to freely explore the dataset and to use a "think-aloud" protocol. This open-ended exploration lasted 40 minutes on average. The examiner provided suggestion in cases where the users did not know what aspects of the data to look at. The participants were also instructed to explain and present any interesting findings to the examiner. At the end we asked them to reflect on their experience with PD and to answer a set of questions, either targeted towards specific features in PD (e.g., How did you like the pen and touch interface?) or more free-form questions about their experience with data-analysis tasks (e.g., Do you have any data, work or private, that you analyze or explore? What tools do you currently use to do so?).

3.5.2 Results

During the opened-ended exploration phase of our study all the participants were able answer the questions they posed with the help of PD. They created sophisticated queries even within the limited amount of time (Figure 3.6, 3.7). All users mentioned that they particularly liked the fact that visualizations can be linked together to create filters. However, only two of our participants made use of brushing-links. Two of the users initially mentioned that they have problems distinguishing when to use the pen and when touch. This confusion was cleared up by telling them that touch is used to "move" objects while the pen allows for shortcut gestures or fine-grained selections. All of the users valued the fluid nature of the UI. More specifically, they thought that the UI was not distracting them from the question they wanted to answer and that it was easy to switch between different tasks (e.g., creating a visualization and linking / filtering). The mnemonic pen gestures for fast access of data-transformation operations were rarely used, while others, such as scribble-delete, the linking of visualization gesture or lasso-selection, were adopted instantaneously by all users. Three of the participants mentioned that they forgot how to perform those mnemonic gestures but would have liked to use them. Discoverability of gestures is a point that we would like to address in future versions of the system [25]. Four out of five participants indicated that they would use PD for their own analysis tasks. One user pointed out that the lack of statistical hypothesis-testing and machine learning methods lessens the value of PD for his own work.

We observed that none of the users had any issues with the touch-gestures. Even without prior

demonstration, most users would walk up to the whiteboard and naturally expected that they could drag visualizations around with their fingers or perform pinch-zoom gestures within graphs or maps. Users were able to decompose compound queries into sequences of simple linked visualization, which was a strategy we never explicitly explained. Users rarely created visualizations form scratch and instead modified visualizations that they derived from existing ones. The notion of deriving a visualization in order to display its legend was initially not clear to participants (e.g., "What do the colors in this graph mean?", "How can I display a legend for this graph?") but especially the reusability of those visualization-legends was appreciated after comprehending the concept ("Ah, now I can just use my legend to filter this other visualization.").

3.5.3 Anecdotal Insights

We would like to point to two anecdotal instances where participants used a table-view as a fallback solution. In one example our user tried to validate his hypothesis that people from less wealthy countries were more likely to be staying in the third class. He approached this by creating a map-visualization showing the passengers home countries, selected countries that he conceived as being less wealthy and linked it to a pie-chart showing the passenger distribution according to classes. The pie-chart showed that most of those passenger were staying in either the first or second class and therefore our user discarded his hypothesis. He noted that in the general population of all passengers is heavily biased towards the third class. Our participant wanted to find out where those third-class passengers came from. Different selections of countries still all offered a similar passenger class distribution (mostly first and second class). After quibbling with this for a while and expressing distrust in what the visualization was showing him, he decided to use a table-view to calculate the exact count of passengers per passenger-class and country. He then realized there was no record of where a majority of the passengers came from and that most of those unknowns were actually staying in the third class. We are planning to address this flaw (not dealing with unknown values properly) in a future version.

In a second example, this time from our pilot-study, a user wanted to display average survival rates for passengers in different age-bins. Furthermore he wanted to see two series that were colored differently for male and female passengers. He expressed concerns that he did not know how to create this plot. However, he said that he knew exactly what data he would like to have plotted. He started to create a table-view with the two columns and applied the appropriate operations to

transform the data and color the rows. After finishing, his table-view depicted the data he was interested in and he realized that it should be simple to plot it now ("Maybe I can actually just plot those two columns against each-other"). He used the axis-gesture, labeled the X- and Y-axis by dragging and dropping columns from his table and got the visualization he initially intended (Figure 3.6).

Those two anecdotes provide some interesting insight. They hint at a concept discussed by [74] and that we summarized as "Data Duality". Users frequently think about data in its raw form and tend to separate data from visual elements. In cases where they either do not trust a visualizations or have problem formulating a visualizations they use raw-data views as a fallback approach.

3.6 Conclusion

PD supports a limited set of chart types (bar-, pie-, scatter-, line-charts, table- and map-views), but it could be extend to include others. The current version PD also only exposes a subset of data-operations. Calculated fields in PD are based on mathematical expressions but there is no way to specify more involved operations (e.g. if, else statements). Additionally, users within PD are limited to compute aggregates over either distinct groups or evenly-spaced bins. More complex grouping schemes such as user-defined or data-type specific groups (e.g., grouping by state or county in geographic attributes or grouping by day, month or year in dates) are not supported yet. There are some exploration interface patterns (e.g., small-multiples of charts [157]) that are used commonly in the set of tasks that PD tries to support. While users in PD could technically layout such interfaces manually through a set of view-creation, linking and filtering operations, it would be a cumbersome task. We are planning to include shortcuts to such patterns in future version so PD.

While some more complete or commercial systems, like Tableau, do address some of the limitations mentioned above, they often lack support for simultaneous viewing and creation of custom exploration interfaces through flexible view-coordination. Furthermore they do not expose a model to incrementally build up complex and modifiable queries where intermediate views serve as interactive handles. In this paper we introduced PanoramicData (PD), a novel pen & touch interface for exploring and analyzing multi-dimensional data-sets. PD is based on a set of four core-concepts: Derivable Visualizations, Exposing Expressive Data-Operations, Unbounded Space and Boolean Composition. By unifying these concepts through a gestural pen & touch UI we are able support

a wide range of common exploration and analysis tasks in a way that avoids explicit mode switching, minimizes UI cluttering, reduces indirection in the interface and provides effortless switching between the variety of interrelated tasks that data exploration and analysis require. PD allows for incremental and piecewise query specification where intermediate visualizations serve as feedback as well as interactive handles to adjust query parameters. A formative user study indicates that our approach provides a fluid and intuitive user experience while exposing a comprehensive set of tools to answer a wide range of data-related questions.

Chapter 4

Visual Regular Expressions for Querying and Exploring Event Sequences

This chapter presents a system called (s|qu)eries (pronounced "Squeries"), a visual query interface for creating queries on sequences (series) of data, based on regular expressions. (s|qu)eries is a touch-based system that exposes the full expressive power of regular expressions in an approachable way and interleaves query specification with result visualizations. Being able to visually investigate the results of different query-parts supports debugging and encourages iterative query-building as well as exploratory work-flows. This chapter is based on [178], which is joint work with Microsoft Research - I was the first author and lead of the project.

4.1 Introduction

Event sequence data sets are collected throughout different domains: electronic health records in medical institutions, page hits on websites, personal data from "Quantified Self" technologies, eye tracking and input device logs during usability studies or program execution logs through telemetry systems to name just a few. The ability to efficiently explore and analyze such sequential data can

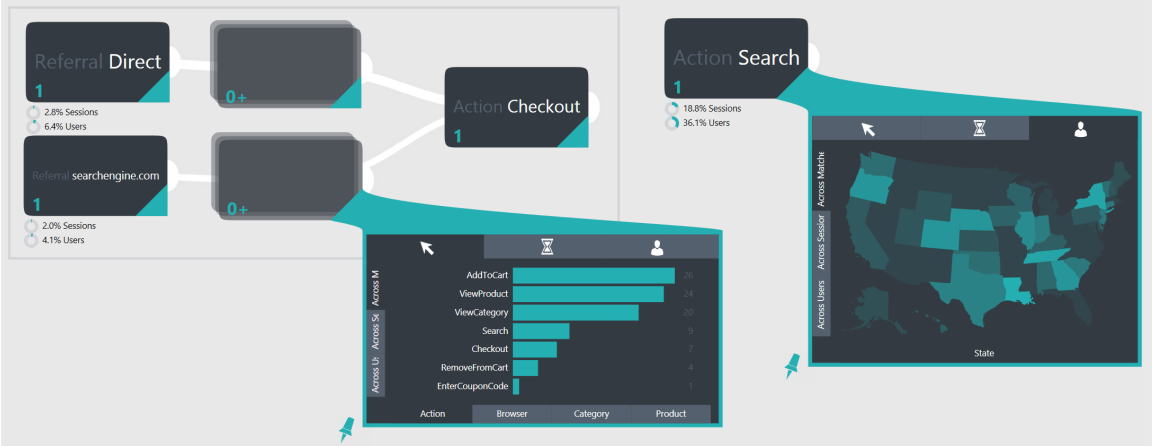

Figure 4.1: Two queries on a fictional shopping website web log. Left: Query to explore checkout behaviors of users depending on direct referral versus users that were referred from a specific website. Right: Query to view geographical location of customers that used the search feature.

generate deep insights and support informed decision making. Examples range from understanding application usage patterns, optimizing marketing campaigns, detecting anomalies, to testing of research driven hypotheses. HCI researchers and practitioners in particular can benefit from event sequence analysis, since usability issues and user performances can be identified and extracted from telemetry data or other logs [67]. The complexity and size of such data sets is continuously increasing and insights might be hidden within the natural ordering or relationships between events, at different levels of detail and in different facets of the data. For example, consider a smart phone application that logs user interactions. A usability manager might want to see high-level aggregations of user flows to uncover possible UX related issues, whereas an engineer might be interested in what led to the triggering of an obscure error code in the few specific sessions where it happened.

While the types of questions analysts and researchers might want to pose vary drastically across domains and scenarios, they often can be reduced to finding patterns of events and being able to inspect the results of such queries. Visualizations are a great way of leveraging the human visual system to support pattern finding. However, for large data sets it is hard to detect insights that are hidden within sub-populations or even single instances or to spot patterns with some degree of variation. Other systems that support querying more directly are either not well suited for this type of sequential data (i.e., SQL), offer limited expressiveness, are hard to learn, or discourage fluid, iterative and interactive exploration.

In this paper we introduce (s|qu)eries, a system that transforms the concepts of regular expressions into a visual query language for event sequences. By interleaving query specification with result visualizations, this integrated interface allows users to incorporate the output from an existing query into the parameters for the next query. We developed (s|qu)eries with a touch-first mindset which resulted in a design where visual elements are a direct component of the interface and indirection is kept to a minimum. Queries, such as the ones in Figure 4.1, can be built and manipulated fluidly with direct manipulation based touch gestures. While our sample application and users focused on telemetry data, the concepts translate to other domains where users need to both spot and investigate patterns.

We started off by analyzing the results of a recent internal study that interviewed 10 data analysts across different functions (UX designers and researchers, program managers, engineers and business stakeholders) from a telemetry group, discussing common formats for log data, understanding the tools that they currently use and, most importantly, finding the types of questions that they typically attempt to answer from their logs. From this point, we iteratively designed the (s|qu)eries system to ensure that it was flexible enough to answer their questions, but also fit into their current workflow.

We validated our approach through detailed interviews with members of five different data science groups within a large data-driven software company. All of the participants analyze event sequences frequently as part of their function within the company. During the interviews we asked the participants to explore their own data through (s|qu)eries. Even though questions, data and backgrounds varied greatly between those teams, we found that all of them were able to uncover previously unknown insights after minimal initial training.

The contributions of this paper are three-fold. First, we present a design for a visual query language for event sequences based on the full power of regular expressions. Second, we incorporated visualizations that allow users to interactively explore the results of both their overall queries as well as individual parts of the queries. These visualizations can be used, in turn, to further add to or refine their queries. Third, we investigate the effectiveness of the system through an informal study with five different data science groups within a large, data-driven software company.

4.2 Related Work

We relate and contrast this work to research efforts in the areas of Event Sequence analysis, Temporal Data Visualizations, Query Languages for Event Sequence & Temporal Data, and Touch-based Interfaces for Visual Analytics.

4.2.1 Event Sequence & Temporal Data Visualizations

Much of the work on event sequences visualization is in the domain of electronic health records. Work in this area is applicable to other event sequence domains like telemetry data, where the payloads of events are different but the visualization strategies are not. While in the medical domain, questions might be about what combination of treatments eventually lead to a stroke, in software telemetry, data scientists might pose similar questions about what events lead to a crash of the application. They both require users to find and analyze sequential patterns within event sequences.

LifeLines [131, 132] is seminal work on how to depict the medical history of a single patient. The tool allows users to visually spot trends and patterns within a single patient's history, or a session when looking at telemetry data. Many other systems [100, 79, 14] have used analogical time-line based representations to visualize single event sequences. By ordering pattern-parts from left to right, (s|qu)eries visualizes sequential patterns in a way that is similar to time-lines, but aggregates across multiple event sequences.

LifeLines2 [166] extends this concept to multiple patient records. Spotting patterns within the data can be achieved by aligning all records according to a sentinel event. However, users are still burdened with visually scanning the entire data set, which often makes pattern detection across large data sets, a cumbersome and error prone task. Other systems [169] address this issue by offering time or category based aggregation techniques. While these approaches scale better to moderately sized data sets, they might hide interesting outliers or pattern variations within the aggregations and are susceptible to noisy input. (s|qu)eries aggregates based on patterns and therefore allows users to explore event sequences at different levels of detail (i.e., from high-level aggregates to individual event sequences) and to capture and hide noise within single nodes.

Sankey diagrams [136] statically illustrate flows and sequential dependencies between different states, however aggregation and alignment techniques are needed to make Sankeys useful for spotting patterns and to reduce visual complexity. (s|qu)eries uses aspects of Sankeys (such as scaling the

width of links between nodes) to help visualize queries as well as the results.

Commercial products for web log analysis or log analysis in general such as Google Analytics[1] or Splunk[2] are built to scale to big data sets and to accommodate data from various sources. Their aggregated dashboard-like visualizations often hide patterns of sub populations and hinder exploratory and interactive workflows. SessionViewer [104] proposes a multiple coordinate-view approach to bridge between different levels of web log aggregations and supports state-based pattern matching through a textual language. However, SessionViewer's pattern language does not support simultaneous exploration of multiple queries nor is it directly coupled to result viewing.

4.2.2 Query Languages for Event Sequence & Temporal Data

While some systems rely on the human visual system to spot patterns in sequential data, others take a more query driven approach. The expressiveness of query systems varies greatly and is dependent on the type of supported data (point or interval events). Most work in this area is based on the concept of temporal logic proposed by Allen [8], which introduces 13 unique operators that describe temporal relationships between intervals. Currently, unlike other systems, (s|qu)eries focuses on sequences of point events (e.g., "a happens after b") instead of temporal queries (e.g., "a happens within x minutes of b").

Jensen et. al. [91] and Snodgrass et. al. [147] both propose extensions to SQL to support predicates for querying time periods. While offering a high degree of expressiveness, their command-based query languages come with a steep learning curve and are not well suited to exploratory workflows.

Jin et. al [92, 93] have addressed some of those problems by proposing a visual query language that is easily understood and specified by users. However, their comic strip metaphor is not well suited for expressing longer repetitions or parallel (alternate) paths. Numerous efforts [57, 105, 120, 121, 35] originated from work in the domain of electronic health records and present visual query languages with varying degrees of expressiveness and for different tasks. For example, PatternFinder [57] describes temporal patterns through events and time spans and visualizes the result in a ball-and-chain visualization. Others, such as Lan et. al. [105] propose extensions to Monroe's et. al. query language [120], including repetition and permutation constraints on point

[1] http://www.google.com/analytics
[2] http://www.splunk.com/

and interval events and absences thereof. Query specification in all of those systems is done through dialog heavy user interfaces and none of them directly interleaves query specification with result viewing.

In a complementary approach from the data mining community, work by Agrawal and Srikant [6, 148] in this domain introduces algorithms to automatically extract common sequential patterns from logs, rather than using user specified patterns to query the data. Such extracted patterns could serve as a starting points for further interactive exploration within (s|qu)eries.

4.2.3 Touch-based Interfaces for Visual Analytics

While we have not deeply explored the potentials for touch first interaction in this paper, their influence on the design came up numerous times. This 'touch-first' mindset (keep indirection at a minimum, animate changes to the UI, and make sure that interactive visual elements can be interacted with directly) was inspired by the work of Elmqvist et. al. [54] and Lee et. al. [107], who emphasize the importance of interaction for information visualization and visual analytics and propose to investigate new interaction models that go beyond traditional WIMP (windows, icons, menus, pointers) interfaces. "Fluid" interfaces, as proposed by [54], are not only subjectively preferred by users, but also enhance their performance (speed and accuracy) for certain data related tasks [49].

(s|q)ueries's visual query language makes extensive use of the direct manipulation paradigm [141]. Nodes, the main visual element in (s|qu)eries, can be manipulated to build queries as well as to analyze query results - they are direct handles to specify input as well as to interact with output. All interactions within the system are made through simple touch gestures. For example, in order to update a pattern, a user simply adjusts the physical layout of nodes by dragging on the node. Other such interactions include using the visualizations to constrain or build new queries or inspecting nodes to investigate the match set at various positions.

4.3 The (s|qu)eries System

We started the design process of (s|qu)eries by analyzing the results of a recent internal study at a large software firm. The study included 10 data analysts across different functions (UX designers and researchers, program managers, engineers and business stakeholders) from a telemetry group.

Question
What is the typical first interaction when users open the application?
What is the most/least often used component of the application?
What is the frequency with which the different features are accessed?
What is the frequency for feature X per user?
How many users perform workflow X?
How long does it take users to perform workflow X?
What are common technical problems?
What leads to common technical problems?
What are common attributes about users of the system that exploit a particular feature?

Table 4.1: Sample of common questions gathered by interviewing data scientists who explore software telemetry data.

It discuses common formats for log data, tools or combination of tools that are being used, and, most importantly, lists questions that stakeholders would like to answer from their logs. A portion of these are shown in Table 4.1.

The two key findings we extracted from analyzing questions like the ones in Table 4.1 are:

- Providing answers to questions is a two step process. First, the pattern of interest must be found across the full data set. Second, the relevant attributes of pattern matches need to be presented in an interpretable way.
- Question come in batches. Questions are often related to each other and / or are built upon answers from the previous ones.

These two findings manifest themselves directly within (s|q)ueries's design. (s|qu)eries exposes the power of regular expressions as a pattern finding language for sequential data through a visual language and interaction design such that users can interactively explore results of matches and use them as starting points for successive queries.

4.3.1 Introductory Use Case

We motivate our approach through an introductory use case that is based on some of the sample questions. Adam is a data scientist at a big e-commerce firm. The company just released a new smart phone application where users can search and browse for products and buy them directly. Adam wants to check if the company's expectations in terms of acquiring new customers were met and if there were any problems with the application. He got a dump of last week's client side telemetry data that the application logs. The sessions, or event sequences, within this data set look

something like this:

```
Session,User,TimeStamp,Attributes
4,3,8/1/14 3:01,action=AppStart
4,3,8/1/14 3:02,action=Search&query=Helmets
4,3,8/1/14 3:11,action=ViewProduct&query=Red Bottle
4,3,8/1/14 3:13,action=AppClose
```

Adam starts (s|qu)eries on his touch-based laptop, double-taps the background to create an unconstrained query node that matches any type of events and inspects the node by opening up its result view (Figure 4.2a). Adam sees a histogram showing the most common actions in the data set. Since he is interested in customers who buy products, he decides to drag out the "Checkout" action (Figure 4.2b). The system creates a new node pre-constrained on "Action = Checkout". Immediately, he sees that 14.2% of sessions resulted in successful checkouts.

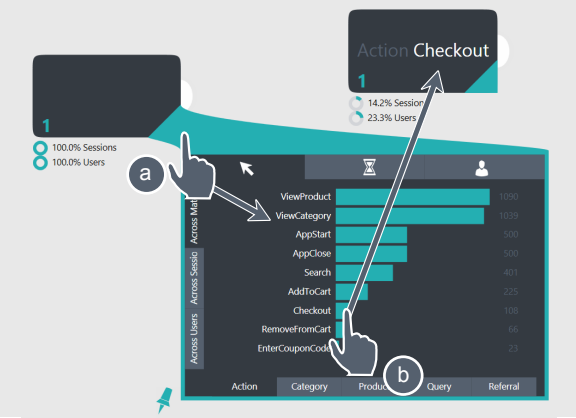

Figure 4.2: Opening up visualization view of a node and dragging from histogram to create a new constrained node.

Naturally he wants to know what led to users successfully checking out. He creates another node and links the two nodes together. This query allows him to look at all the events that happened immediately before people checked out.

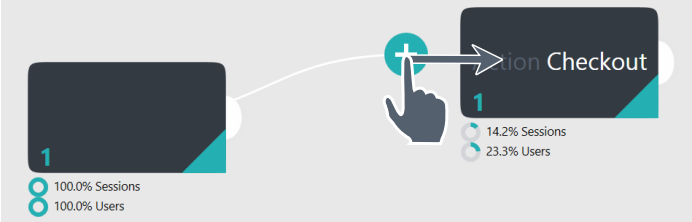

Figure 4.3: Linking of two nodes.

He opens up the visualization view again. Most commonly people added something to their shopping cart before they checked out which seems reasonable.

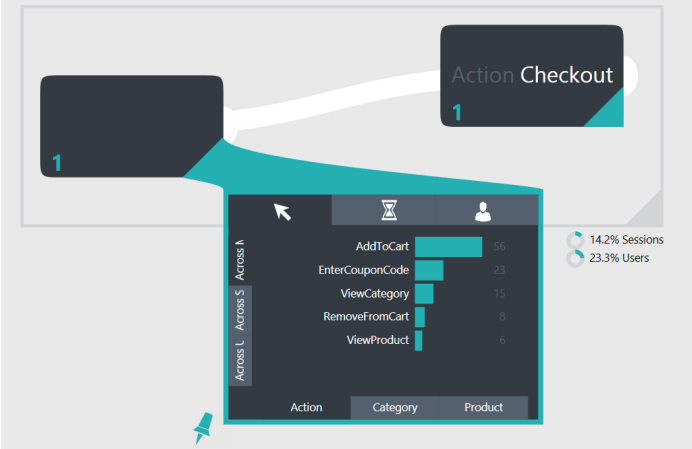

Figure 4.4: Inspecting part part of a sequential two node query.

But what about after checking out? Did users perform any other actions after they bought something? He switches the ordering of the two nodes and the visualization updates.

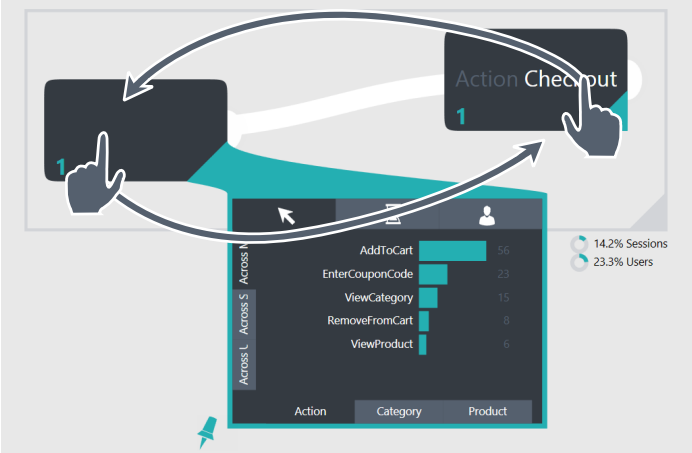

Figure 4.5: Rearranging nodes to create a new query.

Interestingly enough, there are some people that added other items to their cart even though they already checked out - that seems odd. He selects the "AddToCart" item, which in turn constrains the second node and allows Adam to further explore this unexpected user pattern to check if it is an error in the application or some other anomaly. This leads to more fruitful exploratory analyses.

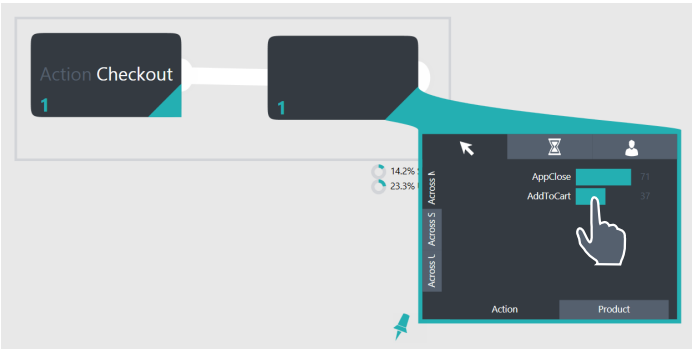

Figure 4.6: Selecting item in histogram to constrain a node.

4.3.2 Data Model

Like Splunk and other systems, (s|qu)eries operates on event sequences, broken into **sessions** by **user**. A single **user** reflects a logical user of the system: in a telemetry log, for example, a **user** might be a single login to the system. A user, however, may log in and out of the system; it can be valuable to break a users events into **sessions**, which represent the time from login to logout. A single **session**, then, is made of a sequence of **events**. Each event is associated with a **user id**,

a **session id**, a single **time-stamp**, and a series of **attribute-value pairs**, which describe that event.

In the use case above, the event sequence represents all the logged actions on the phone application; sessions begin when the user started the app, and end when they close it. An event holds all the information needed to perform its action. For example, when the user adds an item to the cart, the system might log the following three attribute-value pairs: ("action", "AddItemToCart"), ("timestamp", "12/21/2014 21:12:44"), ("item", "city bike"). (s|qu)eries builds its internal data model on the fly from the input file by aggregating along attribute-value pairs. While our current prototype expects a time-stamp, user and session information, which are particular to telemetry data, the data model is flexible enough to accommodate data from many domains as long as events can be represented as a collection of attribute-value pairs.

4.3.3 Query Language

In order to find answers to questions like those in Table 4.1 users need an efficient way to query sequential data. Such queries can be thought of as patterns over event sequences, where different parts of the pattern can be more or less constrained. Log data often contains a large amount of noise and a query language therefore needs to provide support to express variations and fuzziness.

The same applies to extracting patterns from character sequences. Regular expressions are widely used for that task. While they present a powerful language for finding sequential patterns, their text-based representation can be hard to understand and they do not offer support to explore results without additional tools.

By defining our patterns in terms of regular expressions, we leverage an extensive literature of what can and cannot be done within this framework and, if necessary, users can use the literature to form particularly complicated expressions. However, we designed the system such that users do not need to be intimately familiar with regular expressions in order to use the system effectively.

In the following sections we formally describe regular expressions and its base operations. We then discuss how these base operations manifest themselves in our visual query language and how users can specify and interact with them. Furthermore, we show how these visual primitives can be combined to create sophisticated queries over event sequences and serve as direct handles to view results and as starting points to incrementally build up and modify queries.

Regular Expressions

Regular expressions are a well known formal language, introduced by Kleene [101]. In popular tools like `grep`, regular expressions are used for matching patterns within text strings. Before describing how (s|qu)eries uses regular expressions to describe events, here is a short review of the constructs of regular expressions, in the familiar domain of text matching:

1. *Concatenation* : A way of specifying that character follows another character. For example, `abcd` matches "abcd", but not "abdc".
2. *Alternation*: A way of expressing choices. For example, `a|b` can match either the string "a" or "b".
3. *Grouping*: A way of defining the scope and precedence of operators. For example, `gray|grey` and `gr(a|e)y` are equivalent patterns which both describe the strings "gray" and "grey".
4. *Quantifiers*: A way of indicating how many of the previous element is allowed to occur. The most common quantifiers are:
    - The question mark `?` matches 0 or 1 of the preceding element, For example, `colou?r` matches both "color" and "colour".
    - The asterisk `*` matches 0 or more of the preceding element. For example, `ab*c` matches "ac", "abc", "abbc" and so on.
    - The plus sign `+` matches 1 or more of the preceding element. For example, `ab+c` matches "abc", "abbc", "abbbc", and so on, but not "ac".

In addition to these fundamental constructs, tools like `grep` also provide handy shorthand notations for wordy alternations.

1. *Wildcard*: A way of expressing a choice of any character. The wildcard `.` is shorthand for an alternation that includes every character in the alphabet.
2. *Ranges*: A way of expressing a choice based on the order of characters. For example, the wildcard `[a-z]` is shorthand for `a|b|...|z`. Negation (set complement) is also common, e.g. `[^0-9]` represents any character except a digit.
3. *Backreferences*: A way repeating previously matched patterns. For example, the wildcard `([a-c])X\1` is shorthand for `aXa|bXb|cXc`.

For normal text matching, the alphabet contains all characters from a known set like ASCII or Unicode. Sophisticated text matching expressions can be built out of a combination of the above patterns. For example, the following expression matches email addresses:

```
[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+[.][a-zA-Z.]+
```

In (s|qu)eries, instead of directly matching characters in the log file, we represent each **event** as a unique character in a large alphabet based on attribute-value pairs. Each "character" in this alphabet is a fully described event with concrete values for every attribute. For example {("action", "search"), ("query", "city bike")} is a single character in the alphabet. An event's user id and session id are not part of its character representation; rather, they are metadata about that character, much as the line number and column number of a character X in a file are metadata about the X and not used in matching. The events' time-stamps define the ordering of event characters, just as textual characters appear in an order in a text file. Even though the characters of this event alphabet are more complicated than text characters, the regular expression constructs above are well defined for them.

Visual Query Language

Unlike in string-based regular expressions, (s|qu)eries uses a visual language to describe patterns. The main visual elements are nodes, drawn as rounded rectangles. In terms of regular expressions, a node is a character wildcard with a quantifier. The character wildcard is drawn in the middle of the rounded rectangle and the quantifier in the lower-left corner. We call a node *constrained* when some attribute-value pairs are specified and *unconstrained* otherwise. Figure 4.7b shows an example of an unconstrained node that matches any event; whereas Figure 4.7a shows a node that matches any event with the attribute-value pair ("Action", "Search"). Nodes can be constrained on multiple attributes and multiple values per attribute (Figure 4.7d). Constraints on nodes are specified either through a menu-like dialog by tapping on a node or through selections in visualizations.

Nodes can be linked together to create more complex patterns by dragging a node's link handle (white semicircle on the right side of a node) to another node. When linking nodes together, vertical and horizontal placement are both significant: left-to-right placement describes the concatenation of individual nodes; top-to-bottom placement expresses precedence in alternations. The pattern formed by Figure 4.7f shows both of these.

Figure 4.7: Shows nodes in different configurations, with their similar regular expression syntax. a) Constrained node that matches one event, with attribute Action = Search. b) Unconstrained node that matches none or one event of any type. c) Unconstrained node that matches 0 or more events of any type. d) Constrained node that matches one event, with not (white line indicates negation) attribute Action = Search and attribute Browser = Firefox 32.0.1 or IE 11.0. e) Matches sequences that start with one or more events with Action = Search, followed by a one wild card that matches one event. f) Matches event sequences that start with either an event where Action = Search or Action = ViewPromotion, followed by an event with Action = ViewProduct. This pattern is encapsulated in a group (gray box). g) Backreferencing: the chain icon indicates that this pattern matches sequences where some product was added to the cart and then immediately removed again.

Each node has a quantifier applied to it, whose notation is intended to be more intuitive than the standard quantifiers: "1" matches exactly one event (E), "0/1" matches none or one event ($E?$), "0+" matches zero or more events ($E*$) and "1+" matches one or more events ($E+$). Quantifiers are represented textually as well as visually (transparency and node-stacks). Users can change a node's quantifier by tapping on the quantifier label in the lower left corner. Figure 4.7 shows examples of these quantifiers and their visual style.

Additionally, (s|qu)eries supports two common shortcuts: negation and backreference. The former is used to match events that do not comply to the specified constraints (Figure 4.7d shows an example). The latter is used to find matches where an attribute has the same value across two

or more nodes (Figure 4.7g). Furthermore, nodes that are linked can be arranged into groups[3]. (s|qu)eries displays such groups with gray boxes around the node collections (Figure 4.7f).

(s|qu)eries features an unbounded pan and zoomable 2D canvas where users can lay out nodes in a free-form fashion. The concepts outlined above, such as constraints on nodes, quantifiers as well as linkage and position of nodes, can be combined to form arbitrary complex regular expressions over the input event sequence data to find patterns.

4.3.4 Result Visualization

The system runs patterns, formed by the visual language, against the underlying event sequence data set and will return event sequences or parts of event sequences that match it. For example, the pattern formed by Figure 4.7e will match all sub event sequences where one or more sequential events have an attribute-value pair of ("Action", "Search") and are followed by any other event. We will call this retrieved set of sub event sequences the match set.

Most regular expression implementations offer a concept called capturing groups. It enables extraction of the characters that matched a specific part of the entire regular expression pattern. We use this concept to assign events of the match set to nodes within the visual query pattern. Each node (and group) acts as a capture group and therefore knows which events of the match set it was responsible for matching. The nodes thus become visual handles to explore the results of the pattern query they build up. This represents a powerful tool for event sequence analysis, because users can now explore questions like what happens after a sequence of "Searches" by directly inspecting a node (rightmost node in Figure 4.7e).

The match set of each node (and group) can be inspected by pulling out its bottom right corner through a drag interaction (Figure 4.8a). Depending on the amount of pullout the user performed, the pullout view either shows a detailed view of match percentages and absolute numbers or it shows a full visualization view (Figure 4.8b and c).

[3]While our design supports grouping of nodes, it is not fully implemented in our current prototype. Within the prototype, groups around entire queries are supported, while nesting of groups is not.

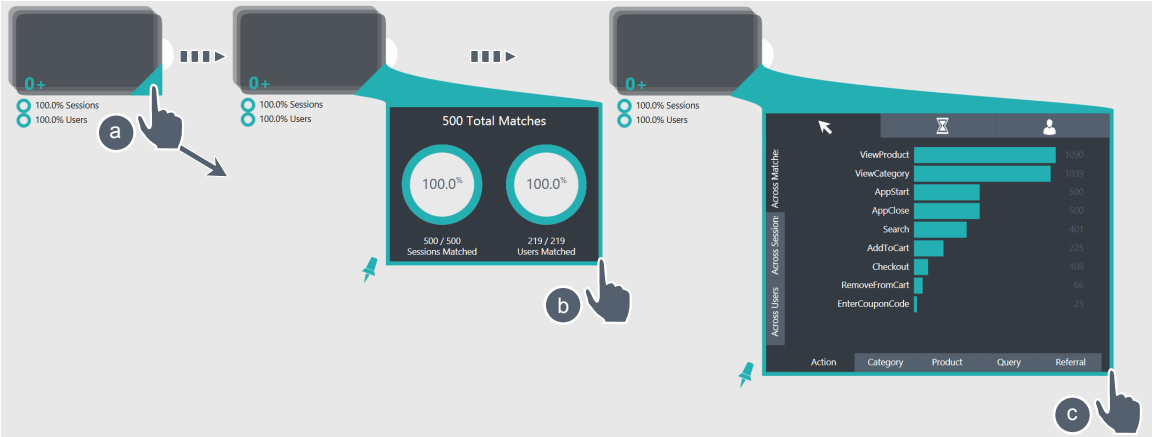

Figure 4.8: A node can be inspected to analyze its match set.

This second view acts as a portal to explore different dimensions of the match set. It is structured into three main tabs (Figure 4.9a) which allows users to switch between visualizations of attribute-value pairs, time dependent information and user information. The first one displays histograms of different and specific attribute-value pairs of the match set. The user can toggle between these histograms by using the attributes on the bottom of the view (Figure 4.9c). The number of attributes that are explorable is dependent on the payload of the matched events. A node that is constrained to the attribute-value pair ("Action", "Search"), will display different attributes than one that is constrained to ("Action", "AddToCart"). Events that match the former might have an associated query term, while the latter might have a product name. From analyzing the questions data scientists posed, we found that results oftentimes needed to be aggregated in different ways. A pattern can occur multiple times within a session and a user can be associated with multiple sessions. Therefore, aggregating either across matches, sessions or users (Figure 4.9b) can reveal different insights. An application feature might seem to be used frequently only because a few select users constantly use it, whereas across all users, it might not be that popular.

In order to encourage exploratory behavior and to allow users to seamlessly switch between query specification and result, these histograms are interactive. Tapping on an item in the histogram view constrains the node to the selected attribute-value pair, while dragging out an item creates a new pre-constrained node.

The second main tab displays time stamp related information of the match set, such as a heat map of hours of the day and day of the week of matches (Figure 4.9) or a histogram of match

durations (from time-stamp of the first event to time-stamp of the last event). It also shows views of the list of actual events sequences that were matched, grouped by length of the matched sequence. And finally, the third tab visualizes information about the associated users of matches like their home states (Figure 4.1 right) if geocoded data is available.

Additionally, some high level aggregates of the match set are overlaid right within the visual query language itself. Quick previews of match percentage are displayed at the bottom of nodes as well as line thicknesses are adjusted to show flow amongst branches (Figure 4.7f).

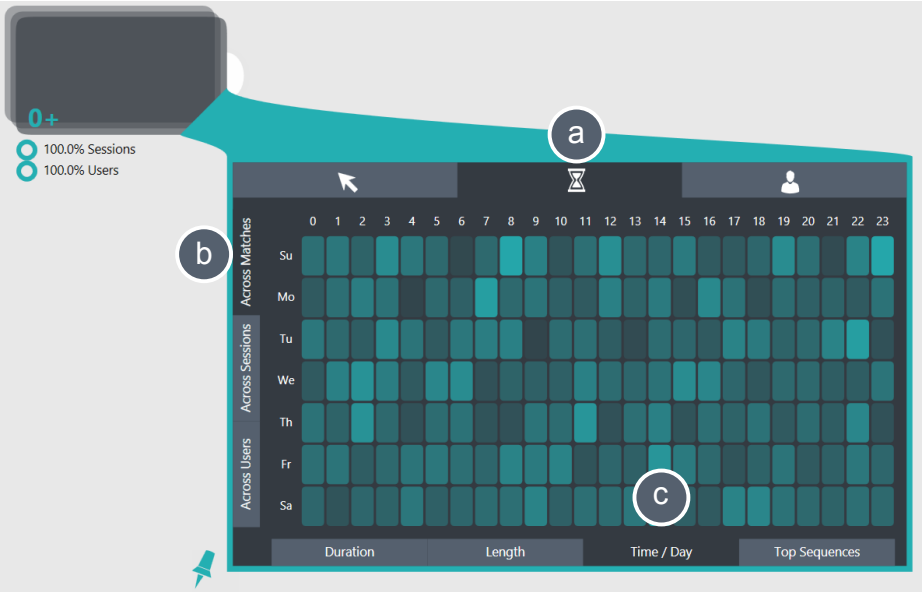

Figure 4.9: The visualization view has three tab navigators to inspect different attributes and aspects of the match set.

Visualizations are currently hard coded: any attribute can be interpreted as a histogram; "TimeStamp" and "UserLocation" can be interpreted for temporal views, timelines and map-views. For now, the source code must be updated to accommodate new visualization types which might be based on certain types of attribute-value pairs or to incorporate visualizations targeted towards a different data domain (e.g., electronic health records).

4.3.5 Implementation & Scalability

Our prototype is developed in C# and WPF and runs on both touch-enabled as well as mouse and keyboard Windows 8 devices. All computation and pattern matching is done in-memory. The system parses the visual pattern into text based regular expressions and utilizes C#'s regular expression

engine to extract pattern matches from the input data set. In order to do so, the system also converts all input sequences into long string representations. After extracting matches and linking them back to the in-memory representation of events, the system computes all aggregated statistics that are used for the visualizations. Pattern matching and aggregation is outsourced to a separate thread to guarantee interactivity while processing queries.

Our prototype works at interactive speeds for real world telemetry logs with up to $30,000$ sessions, with an average length of 150 events, each with payload of around 20 dimensions. While our in-memory approach does not scale to large data sets, the visual language and visualizations do, and the nature of the data makes the required computations an intrinsically good fit for parallelization. These are especially appropriate for map-reduce frameworks, where in the map-step we can compute the regular expression matches and then construct aggregations for the visualizations in the reduce step.

4.4 Evaluation

To evaluate our approach, we invited five people, all of whom analyze event sequences frequently, and encouraged them to bring a colleague along with them for detailed interview sessions. We felt that having colleagues there would help stimulate discussion about the tool. Two of them ended up bringing a colleague along with them. The participants were recruited from five different teams form a large software company. While all of the participants work with telemetry data, we ensured that we had broad coverage with different backgrounds as well as varying goals for potential insights from the tool. To create a familiar scenario and to test (s|q)ueries's ability to operate on different real-world data sets, we asked our participants to send us a sample of their data beforehand. We required the data to be sessionized in a form that made sense to them and limited the number of sessions to $1,500$ with no more than 200 events per session.

We initially gathered some background data on the subjects including their programming expertise, tools they currently exploit in their job and what kinds of insights they were trying to gain from the log data. These are summarized in Table 4.2.

After a 10 minute introduction and demo of the system on a synthetic test data set, we asked them to answer some questions by creating queries on the same test data set to familiarize themselves with the tool. Some example questions were of the form: What was the most common event that

User	Alice
Background/experience	A Product Manager comfortable with programming and familiar with regular expressions but not an expert programmer
Product	Online productivity application 1
Current Techniques	R, Python, SQL
Insights Sought	Test coverage of features within the application and current application shortcomings

User(s)	Brigitte and Charlie
Background/experience	Brigitte was a non-programmer while Charlie had a background in programming and Information Retrieval
Product	Online productivity application 2
Current Techniques	R, Python, SQL for matching sequences, Excel and R for visualization and modeling
Insights Sought	Reproducing crashes from the log data

User(s)	Dorinda and Eric
Background/experience	Data scientists without strong programming backgrounds
Product	Online source control repository
Current Techniques	SQL Services, SQL query analyzer, dashboard in SQL Server Reporting Services
Insights Sought	Understanding end-to-end usage, justifying business cases for separating out features

User	Frank
Background/experience	Developer on the backend server system supplying analytics for data scientists
Product	Social Media Analytics Service
Current Techniques	SQL, Hadoop, Sumologic
Insights Sought	Understanding performance problems in the system from telemetry data

User	Gabriella
Background/experience	Product manager with a little programming experience, not deeply familiar with regular expressions
Product	Web Analytics Platform
Current Techniques	C#, homegrown visualization tools
Insights Sought	Exploring navigation confusion in current offerings, trying to find features to add to their own web analytics platform

Table 4.2: Summary of users from evaluation

occurred immediately before "AddToCart"? How many people used "Search" more than once? What was the most common query term for "Search"?

In the second part of the interviews, we encouraged them to use the system to look at their own data. Since we were interested in seeing if they could create queries that reveal previously unknown insights, we instructed our participants to explain to us what kind of queries they were constructing and if the results confirmed some already known fact or not. This also ensured that we could check if their mental model was matching with what the queries actually expressed. All the sessions were voice-recorded and were conducted on a 55" vertical multi-touch display.

4.4.1 Results

Overall

All the users were able to grasp the system and answer questions after the 10 minute introduction. There was universal positive reaction to the tool with quotes like Charlie's, "This has big potential!" or Gabriella's, "Can we ship this now?". While people with strong programming background were quicker at learning the tool, even those without a programming background were able to use it to extract insights from the test data set as well as from their own data. One user (Brigitte) commented that anything that she was able to understand to this extent in 10 minutes was easy enough for her to use more often, unlike formulating raw queries in SQL or other tools which she tended to avoid using.

The users that came in pairs commented that they particularly enjoyed the touch, large display aspects of the tool since it was well suited to collaboratively exploring the data. Typically each would take a turn getting hands-on with the data queries while the other sat back and made suggestions.

All the users needed to be interrupted in the exploration of their own data after 30 minutes, and all requested to use the tool on their own.

Insights

There were several specific insights that users had while working with their own data.

Alice was able to use (s|qu)eries to **find out what preceded or followed events that were important to her**, for instance she noticed that one of the common events that happens before an "undo" was copy. That was not an undoable action, so she was concerned that this was not a

path that they had tested for. Another common action after "undo" was another "undo" so she went on to explore what were common circumstances that occurred before multiple "undos". This lead her to realize that pasting was often undone, even after several actions occurred. Both of these situations were surprising insights to her.

Bridget and Charlie **used constraints to isolate an exceptional case** - they showed that one feature crashed only on one version of the product. This confirmed a hypothesis that they had previously made. They explained that this had been recently added to that version of the produt and were not completely surprised.

Dorinda and Eric were able **to explore behaviors across longer, "end-to-end", sequences of actions.** They found that nearly half of the people used the "Work Item Tracker" without using the Version Control System. This was significantly more than they expected and was a good justification for unbundling the two features from each other. They had suspected this, but had not gotten clear proof of this before.

Gabriella **identified a potential broad audience of users** - web analytics managers, who ask questions like these frequently, but were often frustrated in trying to answer things beyond those put into standard, regular dashboards or reports.

Problems and Challenges

There were also some problems or challenges encountered.

Data sets were not always well formed for extracting insight. For instance, Brigitte and Charlie's log data included events that they wished to ignore. While this could have been preprocessed out of the file, they did not realize this until too late. Queries *could* be constructed that specifically ignored those events, but this was somewhat tedious using the current interface. They requested a feature that would allow them to selectively filter out events or combine certain multiple events into a single event.

Occasionally, regular expression rules could lead to some confusion over the results. Frank discovered that when a wildcard was used between two different events, he assumed that this meant that the event did *not* occur between the two events as opposed to matching the longest sequence possibile between the two events. For instance, if the pattern was A.*B, then this would match AAAAAB or ABBBBB or even ABCDAABCBAAB.

Everybody wanted the system to scale to support bigger event sets since most had data sets hundreds or thousands of times larger than those they explored in the tool.

4.5 Discussion & Future Work

While the current prototype has already proved itself useful both to practitioners in the informal user study and in continuing analysis efforts, there are a number of limitations that can be addressed in future work. Some of which involve small, iterative improvements in the design, and some which would involve more major changes.

Basing the system on regular expressions, which are well-understood in the literature, makes its capabilities and limitations predictable. Regular expressions, unlike push-down automata, can not keep track of state, and therefore certain questions can not be answered if they are not explicitly contained as payloads in the event stream. For instance, while a log might contain events where items are added to the cart, the user can not retrieve the total contents of a cart unless there is an explicit representation of the contents within the payload of the event stream. Furthermore, regular expressions themselves can be notoriously confusing. This is somewhat mitigated in the system by not requiring a cryptic textural stream, (e.g., the email matching regular expression mentioned previously). It also helps that the contents of various nodes in the query can be explicitly examined and the patterns that they match can be explored with the visualizations. In the (s|qu)eries system, the only confusions that came about were because of the greedy nature in the regular expression matching (as discussed in the results above). This also occurred when using alternations - when the top node was a wild card, by definition, no matches were left for nodes lower in the list. One user (Gabriella) did not even realize that the system was based on regular expressions until after we discussed it with her towards the end of the session. She was only vaguely familiar with them and was still completely capable of using the system to answer our test questions and questions on her own data.

The system at this time lacks the ability to add a temporal element to the queries. This would be an exciting and powerful additional capability but has not yet been addressed in the current prototype. Temporal constraints would interact with the regular expressions in an interesting fashion, but could probably be added as a further post-process on the results of a set of matched events.

From the interview with Dorinda and Eric, it became clear that logs often have erroneous or

lower level events that need to either be completely filtered out or combined into single, higher level events. Search and replace capability, similar to [105], could add great power to the (s|qu)eries system.

One of the current benefits of the (s|qu)eries system is that the results of the queries can easily be constrained to help create more queries. We want to further expand this interaction so that the system could either automatically display commonly pathways, or so that multiple paths can be interactively constructed by dragging out several results from a node to create separate queries.

Many, small usability improvements could also be made. People did not like starting with a blank canvas, so pre-populating the canvas with at least a single node that shows the most common events in the log would be useful. We wish to allow for saving and loading of queries so that sessions can be shared or continued.

Finally, expanding the capabilities of the visualizations can be done in many different ways. Specific vertical visualizations that explore certain payloads or combine the results of a query with other tables (like geocoding from IP lookup or heat maps of features used in a program) could be added. Several users wanted to contrast the results of multiple queries (e.g., What were the differences between what people added to a cart for those people that used search and those people that didn't?).

4.6 Conclusion

As sequential logs across numerous domains explode in availability, we see that there is great potential in enabling domain experts who may be non-programmers. Requiring a programmer in the loop greatly limits the exploratory possibilities that the readily available data facilitates. We plan on continuing to expand the capabilities of the (s|qu)eries system as we apply it to other users, other questions, and other domains.

We introduced (s|qu)eries, a touch-based system that features a visual query language building upon the full power of regular expressions. Our system incorporates visualizations that allow users to interactively explore the results of both their overall queries as well as individual parts of the queries. These visualizations can be used, in turn, to further add to or refine their queries. We investigate the effectiveness of the system through an informal study with five different data science groups within a large software company.

Chapter 5

The Case for Progressive Visualizations

In this chapter we study how progressive visualizations affect users in exploratory settings through an experiment where we capture user behavior and knowledge discovery through interaction logs and think-aloud protocols. This chapter is substantially similar to [175], where I was the first author and was responsible for the research direction, implementation, study design, analysis and the majority of the writing.

5.1 Introduction

The literature [139, 124, 31, 140] often states that a delay of one second is the upper bound for computer responses after which users lose focus on their current train of thought. In order to ensure a highly interactive environment for data analysis, a visual data exploration system should therefore strive to present some actionable and understandable artifact for any possible query over any dataset within a one second threshold. It is important to note that this artifact does not need to be the complete or most accurate answer, but it should be an answer that allows users to keep their attention on their current task.

Traditionally, visual data exploration systems employ a strategy whereby user-issued queries are offloaded to a database management system (DBMS), and the results are displayed once the

complete answer is computed. We call this the *blocking* approach: a user's current train of thought is blocked until the query result is fully computed. Even with modern hardware and state-of-the-art DBMSs that can process millions of data points per second, this traditional approach suffers from the basic issue that some datasets will still be too "big" to yield results for user queries within interactive thresholds.

A wide variety of strategies, including precomputation, prefetching, sampling, and progressive computation, have been proposed to overcome this fundamental limitation. However, each of these approaches comes with its own set of advantages and challenges. In particular, *progressive computation*, where data is processed incrementally in small chunks, offers an interesting tradeoff between result accuracy and computation speed. Moreover, unlike many of the other approaches, progressive computation also provides a number of natural opportunities to incorporate user feedback and computational steering. The research community has recently regained interest in progressive computation, attempting to analyze and exploit some of the peculiarities of this approach [65, 149, 62, 64, 60, 137]. However, the effects of progressive computation—and progressive visualization—on user behavior and knowledge discovery in exploratory settings have not been studied in detail.

The aim of this work is to investigate how progressive visualizations affect users in exploratory settings. To this end, we design and conduct an experiment where we compare three different visualization conditions: (1) instantaneous, (2) blocking, and (3) progressive. The instantaneous visualization condition acts as a stand-in for hypothetical systems where all queries, independent of the dataset size, return and display accurate results within a strict latency constraint. On the other hand, blocking visualizations simulate traditional systems where results are displayed only after the full dataset has been processed. Blocking visualizations are limited by the throughput of the underlying computation engine; that is, the wait time increases proportionally to the size of the dataset. Finally, progressive visualizations, where data is processed incrementally in small chunks, present approximate results to the user at different points during the computation. To compare each of these strategies, we capture interaction logs and verbal data from the participants using a think-aloud protocol. We then extract several knowledge discovery and user activity metrics from these recordings, and we analyze these metrics to test how progressive visualizations compare to the alternatives. For the purpose of our study, we picked simple uncertainty visualizations and update strategies and are specifically not testing different variations in those.

The main contributions of this paper are two-fold. First, we find that progressive visualizations significantly outperform blocking visualizations in almost all knowledge discovery and user activity metrics. Second, our results show that, surprisingly, progressive visualizations do not differ substantially from the best case scenario of instantaneous visualizations across many key metrics (e.g., insights per minute, insight originality, visualization coverage percentage). These findings suggest that progressive visualization is a viable solution to achieve scalability in visual data exploration systems.

5.2 Related Work

Our research builds upon related work in the areas of *Big Data Visual Analytics* and *Latency in Computer Systems*.

5.2.1 Big Data Visual Analytics

Visual data analysis, or the task of gaining insights from a dataset through visualizations, is an interactive and iterative process where users must frequently switch between a wide range of distinct but interrelated tasks. While the specific set of tasks that recur in visual data analysis, as well as the tools that support them, are relatively well understood [10, 83], the constantly increasing volume of data has forced the interaction paradigm away from interactive approaches back to large-scale batch processing [63].

Thus, we seek to address this fundamental conflict in visual exploratory data analysis. On one hand, we want to provide an experience where users can actively steer the exploratory process, allowing them to see results for each action without the distraction of a slow and unresponsive interface. On the other hand, constantly growing amounts of data and increasingly complex analysis techniques make it impossible to compute and deliver accurate responses within latency thresholds that are suitable to keep users on their current train of thought.

The research community has proposed several approaches to address this problem, each with its own set of advantages and challenges. We introduce a simple use case to illustrate these different approaches. Imagine a visual data exploration system that processes tabular data, allowing users to create simple aggregated histograms over this data by selecting different attributes. Furthermore, histograms are linked together, where selections in one visualization trigger filtering operations in

others, such as in GraphTrail [51], PanoramicData [176], and Vizdom [41]. Figure 5.1 shows an example where the user only wants to see the histogram of income for a specific age range.

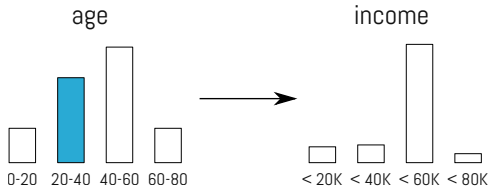

Figure 5.1: Two coordinated visualizations. Selections in the left filter the data shown in the right.

Conceptually, the simplest way to implement a data exploration system to support these interactions is through a *blocking* approach. After the user requests a particular histogram, the system scans the full dataset to perform the required aggregation and displays the result only after processing all of the data points. Using this approach, the user cannot see results until after the entire dataset has been processed. Response times of blocking systems are therefore proportional to the size of the dataset. Several commercially available visual analysis tools, including Tableau [2] (and its research predecessor Polaris [150]), Spotfire [3], and Microsoft Power Pivot [1], use a blocking approach.

Precomputation provides an opportunity to reduce the latencies of the blocking approach by performing some of the computation up front before a user begins exploration. The system can perform this precomputation during a loading phase and simply return a result from its cache when the user issues a request. This approach requires a substantial time and processing investment during the initial loading phase but can completely eliminate visualization latency during data exploration. However, a major drawback of precomputation is the potentially enormous number of both visualizations and exploration paths the user can take. In particular, the number of possibilities depends both on the characteristics of the data as well as the exploration tasks supported by the system. Permitting arbitrarily filtered histograms (e.g., as shown in Figure 5.1) drastically increases the number of required precomputations. Even with these issues, many systems have successfully used precomputation to improve the user experience. For instance, some commercial DBMSs support the creation of online analytical processing (OLAP) cubes [34], where the data is pre-aggregated along selected dimensions. Other research systems introduce improvements to these techniques using more advanced algorithms and data structures [113] or by exploiting modern hardware (e.g., GPUs) [116].

Similar to precomputation, *prefetching* incrementally precomputes results during user exploration rather than computing all possible results a priori. Prefetching approaches typically either limit the degrees of freedom given to the user or employ an intelligent oracle that predicts the user's most likely subsequent actions, such as in a map application where users can perform only pan and zoom actions. Since the user can only transition to a very limited number of possible next states (i.e., panning left/right/up/down or zooming in/out), the system can therefore use the time that a user spends examining a specific region to prefetch neighboring tiles that might be requested next. Both ForeCache [16] and Semantic Windows [95] use prefetching by exploiting locality in the user's exploratory behavior to predictably prefetch chunks of data in anticipation of subsequent. Other prefetching based systems use models to estimate the most likely action the user will perform next. These predictive models can be built using a wide variety of different information. For example, Doshi et. al. [47] propose and compare different techniques that incorporate a user's interaction history, while Ottley et. al. [129] show that certain personality traits affect user exploration strategies. Prefetching systems need to provide a fallback strategy for cases when the prediction fails, most commonly by reverting to a blocking approach for cases when the user issues a query for which the result has not yet been prefetched. Furthermore, even with prefetching, the result might take too long to compute if the user only spends a short amount of time between interactions.

While all of the approaches discussed thus far are guaranteed to return completely accurate results for all queries, *sampling* takes a fundamentally different approach by trading speed for accuracy. Instead of computing the completely accurate result, the system uses a small sample of the data to compute the histogram and presents it to the user with some quality metric for the approximation (e.g., confidence intervals, error bars). Systems that use sampling can return results for all user queries within the specified interactivity time constraints without needing to precompute any query results. The data management community has heavily explored the development of approximate query engines. For example, BlinkDB [5] is a SQL engine that uses sampling to answer queries over large datasets and allows users to specify accuracy or response time requirements. DICE [96] similarly supports subsecond latencies for large datasets using a variety of optimizations including sampling.

Although sampling-based approaches can provide users with a quick overview of the dataset, they also introduce a completely new set of challenges. For instance, rare (but potentially important) datapoints might not be captured in the sample. Additionally, even experts sometimes have

trouble interpreting statistical accuracy metrics [43, 44]. Ferreira et. al. [61] introduced specialized visualizations that mitigate some of these issues, but further research is necessary in order to apply their findings to different visualization types and analysis tasks.

Progressive systems are an extension of sampling-based approaches that incrementally compute results over increasingly larger samples in order to provide more accurate results to the user over time. This concept is well known in the graphics domain and widely used on websites to improve user experience when loading high-resolution images [80]. A down-sampled, low-resolution image is displayed quickly and replaced with the high-resolution version when fully downloaded. This concept was previously explored in the data management community [84], where most of the subsequent research focused on creating progressive versions of well known DBMS operations (e.g., joins [77], aggregations [38]). More recently, progressive approaches have gained interest among the HCI and visualization communities [65, 149, 62, 64, 58, 137]. Progressive Insights [149] is an example of a progressive system that allows the user to analyze common patterns in event sequence medical data. Fisher et. al. [65] presented *sampleAction*, a tool that simulates progressive queries over large datasets, and discuss and analyze the implications of such queries and different confidence metrics through quantitative case studies. While this approach has similar drawbacks to sampling (e.g., bad for finding outliers, does not work with ordered data, requires statistical sophistication from users), many authors [62, 64, 58, 41] advocate for its usefulness in exploratory data analysis. For example, it allows users to decide what level of accuracy they need and provide opportunities to inject user-steerability into algorithms and computations. However, the effects of this approach in terms of user performance and behavior have not yet been analyzed in detail.

5.2.2 Latency in Computer Systems

As many have argued [83, 78], the goal of visual data exploration systems is to operate at a rate that matches the pace of data analysts. Systems should therefore attempt to keep query response times below thresholds that will make users lose focus on their current task. A study by Liu et. al. [114] shows that latencies of 500ms have significant effects on user performance in data exploration scenarios. In other domains, including web search [28] and video games [18], even smaller latencies (300ms and 100ms, respectively) can negatively influence user performance. Frameworks proposed by Nielsen and others [139, 124, 31, 140] suggest that responses within one second allow users to stay on their current train of thought, while response times over ten seconds exceed the average

attention span. We use these models to justify the different delay times used in our experiment.

5.3 Experimental Design

The aim of this work is to investigate how progressive visualizations influence users during exploratory data analysis, as well as how these techniques compare to blocking and instantaneous visualizations. We use a full-factorial 3 visualization conditions (blocking, instantaneous, progressive) $\times$ 2 dataset-delay conditions (6s, 12s) $\times$ 2 dataset-order conditions (123, 312) experiment. Our experiment is based on work by Liu et. al. [114] and Guo et. al. [76], which suggest using a hybrid evaluation approach that uses system logs and insight-based metrics coded from think-aloud protocols in order to analyze both (1) user interactions and (2) analysis performance. We expect that users will generate more insights per minute with instantaneous visualizations than with progressive ones ($H1$) and that users will generate more insights per minute with progressive visualizations than with blocking ones ($H2$). Furthermore, we anticipate user activity levels to be higher with instantaneous visualizations than with progressive ones ($H3$) as well as higher with progressive visualizations than with blocking ones ($H4$). This section provides a detailed description of the experimental design.

5.3.1 Visualization Conditions

In order to understand how progressive visualizations influence users in terms of knowledge discovery and interaction behavior, we compare them against two baseline visualization conditions: (1) blocking and (2) instantaneous. The instantaneous condition represents a hypothetical ideal scenario where the system always return query results within the time constraint regardless of dataset size. The blocking condition represents the other extreme, which is how many current visual data exploration systems operate. For blocking visualizations, query results are displayed only after the computation over the entire dataset concludes, with visualization latencies scaling with dataset size. The progressive condition bridges the instantaneous and blocking conditions by displaying an approximate result as soon as possible and then incrementally refining the results over time. Eventually, the full and completely accurate result is displayed, as in the blocking approach, but initial approximate results are still returned as soon as possible.

Figure 5.2 shows a schematic illustration of the differences between these three conditions. Assuming that a given hardware platform requires x seconds to compute and display the full query

result, then a system operating in the hypothetical instantaneous mode will return results immediately; a blocking system will use the full x seconds before displaying any results; and the progressive system will show inaccurate results that are incrementally refined throughout the x seconds.

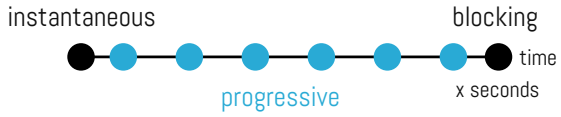

Figure 5.2: Schematic time scale showing when different visualization condition display results.

5.3.2 Datasets

We use three datasets from different domains in our experiment. The first dataset (DS1) contains information about cars [90] (8 attributes: 7 quantitative, 1 nominal), such as "acceleration" and "horsepower." The second one (DS2) includes data about wines (7 attributes: 5 quantitative, 2 nominal), with attributes including "type", "country of origin" and several different ratings. Finally, the third one (DS3) is a subset of the 1994 US census [90] (9 attributes: 3 quantitative, 6 nominal). We use a fourth dataset (Titanic) to introduce participants to the system.

The datasets contain 10,000 data points each. We introduce a dataset-delay factor that has two possible values (6s or 12s) and is used to artificially delay the computation of visualizations for the progressive and blocking conditions. In the blocking case, the user will have to wait either 6s or 12s before any visualization is displayed, while the computation spans the entire time period (6s or 12s) with 10 incremental updates in the progressive case. While 6s and 12s still represent relatively small times (it is common to have datasets large enough that computations can take hours), we chose these values for two reasons: (1) these delays are above and below the 10 second threshold that is often stated as the average attention span [124]; and (2) they are small enough to make an in-lab user study feasible.

5.3.3 System

We created an experimental system specifically for our study. The UI, shown in Figure 5.3, consists of a list of the current dataset's attributes (a) and four visualization panels ($v1$-$v4$). Users can drag and drop attributes onto the axes of the visualization panels, and tapping on an axis cycles through different aggregation functions (e.g., count, average). Our system supports two visualization types:

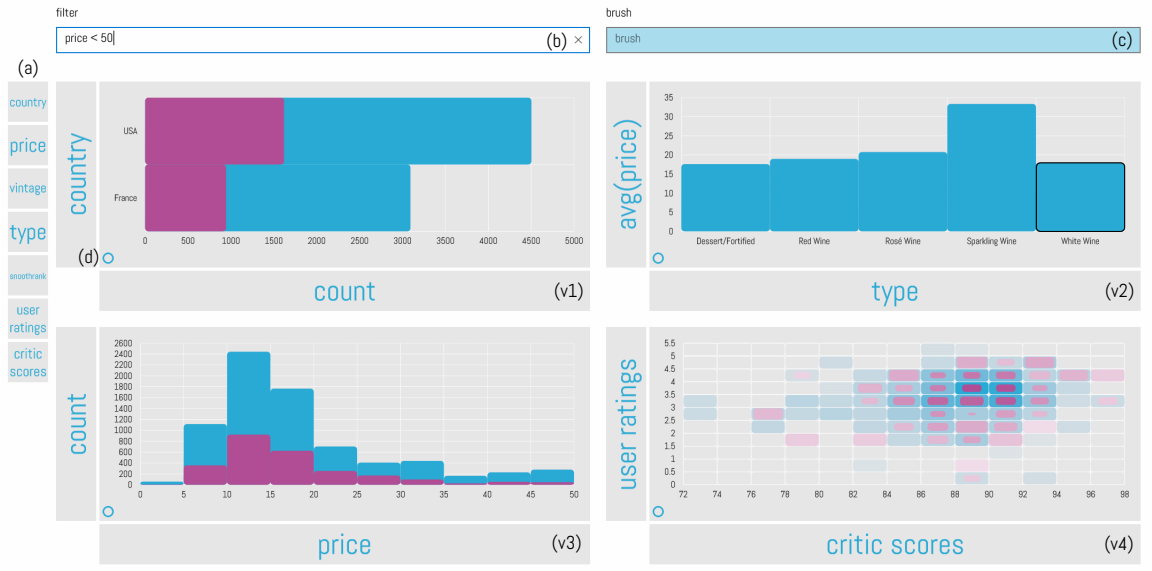

Figure 5.3: Screenshot of our experimental system.

(1) bar charts and (2) 2D-histograms. Visualizations are always binned and never show individual data points, which allows us to fix the time required to display a visualization, even for arbitrarily large datasets. That is, the number of visual elements to render is decoupled from the size of the underlying dataset.

Selecting a bin in a visualization triggers a brushing operation in which the selected data points are highlighted in all other visualizations. In the example (Figure 5.3), the user selected the rightmost bar in $v2$. All other visualizations now shade bins in two different colors: blue indicating the overall amount and purple the amount of data points that corresponds to the user's selection. Bar height and rectangle area are scaled to reflect the number of data points that match the selection. A textbox (c) permits finer-grained brushing control through arbitrary Boolean statements (e.g., highlight all wines where $price < 35$ *and* $vintage > 2010$). Similarly, a second textbox (b) supports filtering operations through Boolean statements to select a specific subset of the data. Brushing and filtering operations require all affected visualizations to recompute their results from scratch, regardless of the current visualization conditions. For example, selecting an additional bar in $v2$, and thereby changing the currently applied brush, will force all other visualizations (i.e., $v1$, $v3$, and $v4$) to recompute. Depending on the visualization condition, $v1$, $v3$, and $v4$ will either show the results of the new brushing action instantaneously, a loading animation until the full result is available, or incrementally updated progressive visualizations. The system does not perform any form of result

caching.

Upon startup, the system loads the selected dataset into memory. The system randomly shuffles the data points in order to avoid artifacts caused by the natural oder of the data and to improve convergence of progressive computations [46]. We approximate the instantaneous visualization condition by computing queries over entire dataset as quickly as possible. Initial microbenchmarks for a variety of queries over the small datasets used in the experiment yielded the following measurements: time to compute a result $\approx 100ms$ and time to render a visualization $\approx 30ms$. Although not truly instantaneous, a total delay of only $\approx 130ms$ is well below the guideline of one second, therefore allowing us to simulate the instantaneous condition. We simulate the blocking condition by artificially delaying the rendering of a visualization by the number of seconds specified through the dataset-delay factor. In other words, we synthetically prolong the time necessary to compute a result in order to simulate larger datasets. While a blocking computation is ongoing, we display a simple loading animation, but users can still interact with other visualization panels, change the ongoing computation (e.g, selecting a different subset), or replace the ongoing computation with a new query (e.g., changing an axis). Figure 5.4 (top) shows an example of a blocking visualization over time.

We implemented the progressive condition by processing the data in chunks of 1,000 data points at a time, with approximate results displayed after each chunk. In total, we refresh the progressive visualization 10 times, independent of the dataset-delay factor. We display the first visualization as quickly as possible, with subsequent updates appropriately delayed so that the final accurate visualization is displayed after the specified dataset-delay condition. Note that the initial min and max estimates might change afters seeing additional data, in which case we extend the visualization by adding bins of the same width to accommodate new incoming data. Throughout the incremental computation, we display a progress indication in the bottom left corner of a visualization (Figure 5.3 (d)). Progressive visualizations are augmented with error metrics indicating that the current view is only an approximation of the final result. Even though 95% confidence intervals based on the standard error have been shown to be problematic to comprehend in certain cases [39], we still opted to use them for bar charts due to their wide usage and familiarity. We render labels with margins of error (e.g., "±3%") in each bin of a 2D-histogram. Figure 5.4 (bottom) shows an example of a progressive visualization and how it changes over time. Note that the confidence intervals in the example are rather small, but their size can change significantly based on the dataset and query.

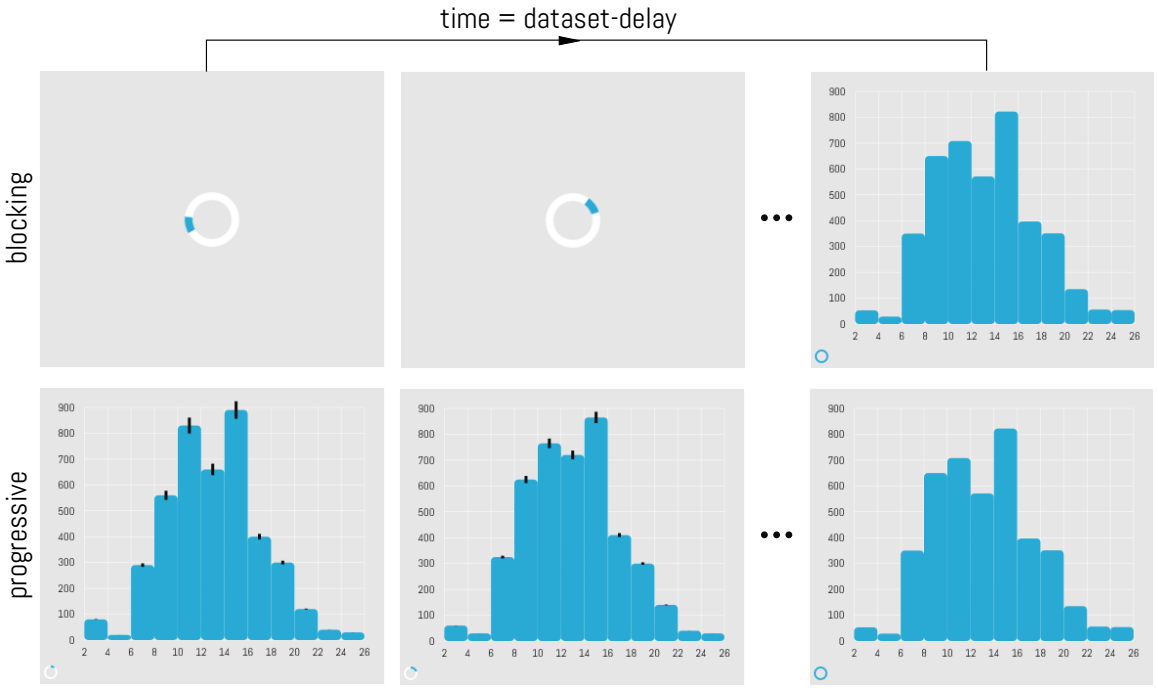

Figure 5.4: The blocking and progressive visualization conditions.

Our system is implemented in C# / Direct2D. We tested our system and ran all sessions on a quad-core 3.60GHz, 16GB RAM, Microsoft Windows 10 desktop machine with a 16:9 format, 1920x1080 pixel display.

5.3.4 Procedure

We recruited 24 participants from a research university in the US. All participants were students (22 undergraduate, 2 graduate), all of whom had some experience with data exploration or analysis tools (e.g., Excel, R, Pandas). 21 of the participants were currently enrolled in and halfway through an introductory data science course. Our experiment included visualization condition as a within-subject factor and dataset-delay and dataset-order as between-subject factors. Note that the dataset-delay has no direct influence on the instantaneous condition, but it is used as a between-subject factor to test if it affects the other visualization conditions. To control against ordering and learning effects, we fully randomized the sequence in which we presented the different visualization conditions to the user and counterbalanced across dataset-delay conditions. Instead of fully randomizing the ordering of datasets, we opted to create two predefined dataset-orderings and factored them into our analysis. The two possible dataset-ordering values were 123 and 312 (i.e., DS1 followed by DS2

followed by DS3 and DS3 followed by DS1 followed by DS2, respectively), and we again counterbalanced across dataset-ordering. We seed our system's random number generator differently for each session to account for possible effects in the progressive condition caused by the sequence in which data points are processed. While each participant did not experience all possible combinations of visualization, dataset-ordering, and dataset-delay conditions, all users saw all visualization conditions with one specific dataset-delay and dataset-ordering. For example, participant P14 was assigned dataset-ordering 312, dataset-delay 6s, and visualization condition ordering [blocking, instantaneous, progressive]. In total, this adds up to four trial sub-groups, each with six participants: (dataset-delay = 6s & dataset-order = 123), (dataset-delay = 12s & dataset-order = 123), (dataset-delay = 6s & dataset-order = 312) and (dataset-delay = 12s & dataset-order = 312). Within each subgroup, we fully randomized the order in which we presented the different visualization conditions.

After a 15 minute tutorial of the system, including a summary of the visualization conditions and how to interpret confidence intervals and margins of error, we instructed the participants to perform three exploration sessions. In the case of participant P14, these three sessions included (1) blocking visualizations on DS3, (2) instantaneous visualizations on DS1, and (3) progressive visualizations on DS2 all with 6s delay. Each session was open-ended and we asked the participants to explore the dataset at their own liking and pace. We allotted 12 minutes per session, but participants were free to stop earlier We used a think-aloud protocol [56] where participants were instructed to report anything they found interesting while exploring the dataset. Throughout each session, we captured both screen and audio recordings and logged low-level interactions (mouse events) as well as higher-level events ("axis changed", "aggregation changed", "textbox brush / filter", "visualization brush," and "visualization updated / stopped / completed"). An experimenter was present throughout the session, and participants were free to ask any technical questions or questions about the meaning of dataset attributes. At the end of the three sessions, we asked participants to give feedback about the tool and whether they had any thoughts regarding the different visualization conditions.

5.3.5 Statistical Analysis

Our study is designed as a full-factorial 3 (visualization conditions) $\times$ 2 (dataset-delay conditions) $\times$ 2 (dataset-order conditions) experiment. We applied mixed design analysis of variance tests (ANOVA) with visualization condition as the within-subject factor and dataset-delay and dataset-order as the between-subject factors to assess the effects of our factors on the various metrics

we computed. Note that the dataset-delay factor should have no influence on trials where the visualization condition is set to instantaneous.

We tested the assumption of sphericity using Mauchly's test, and we report results with corrected degrees of freedom using Greenhouse-Geisser estimates if violated. We report all significant (i.e., $p < 0.05$) main and interaction effects of these tests. For significant main effects, we conducted further analysis through Bonferroni corrected post hoc tests and for more nuanced interpretation, we opted to include Bayes factors for certain results and report BF_{10} factors along with corresponding significance labels [110]. Effect sizes are reported through Pearson's r coefficient, and significance levels are encoded in plots using the following notation: $*\ p < 0.05$; $**\ p < 0.01$; $***\ p < 0.001$.

5.4 Analysis of Verbal Data

Inspired by previous insight-based studies [72, 76, 114], we manually coded insights from the audio recordings and screen captures. An insight is a nugget of knowledge extracted from the data, such as "France produces more wines than the US". We followed the verbal data segmentation approach proposed by Liu et. al. [114] and adopted their coding process in which the first author did the bulk of the coding, but we iteratively revised finished codes with collaborators to reduce bias. In our case, we decided to count observations that are within the same visualization, have the same semantics, and are on the same level of granularity as one insight. For example: "It looks like country 1 makes the most cars, followed by country 2 and country 3" was coded as one single insight, whereas an observation across two visualizations (through brushing) such as "Country 1 makes the most cars and seems to have the heaviest cars" was counted as two separate insights. We did not categorize insights or assign any quality scores or weights to insights nor did we quantify accuracy or validity of insights. For our analysis, all insights were treated equally.

5.4.1 Number of Insights per Minute

In order to get a quantifiable metric for knowledge discovery, we normalized the total insight count for each session by the duration of the session. Figure 5.5 shows this resulting "number of insights per minute" metric across different factors. Our results show that this metric was significantly affected by the type of visualization condition ($F(1.452, 29.039) = 6.701, p < 0.01$).

Post hoc tests revealed that the instantaneous condition showed a slight increase of number of

insights per minute over the progressive condition (1.477 ± 0.568 vs. 1.361 ± 0.492, respectively), which was not statistically significant ($p = 1.0, r = 0.080$). However, the blocking condition reduces the number of insights per minute to 1.069 ± 0.408, which differed significantly from both the progressive ($p < 0.05, r = 0.292$) and instantaneous ($p < 0.001, r = 0.383$) conditions. A Bayesian Paired Samples T-Test that tested if $measure_1 < measure_2$ revealed strong evidence for an increase in insights per minute from the blocking condition to the progressive condition ($BF_{10} = 13.816$), extreme evidence for an increase from blocking to instantaneous ($BF_{10} = 134.561$), and moderate evidence for no change or a decrease between instantaneous and progressive ($BF_{10} = 0.130$). In summary, blocking visualizations produced the fewest number of insights per minute, whereas the instantaneous and progressive visualizations performed equally well.

5.4.2 Insight Originality

Similar to [76], we grouped insights that encode the same nugget of knowledge together and computed the originality of an insight as the inverse of the number of times that insight was reported by any participant. That is, insights reported more frequently by participants received lower overall originality scores. A participant received an insight originality score of 1 if he or she had only unique insights (i.e., insights found by no other users). The lower the score, the less original the insights were on average. We averaged originality scores across insights for each session and show plots for this insight originality metric across different factors in Figure 5.6. We did not find any significant effects that influence the insight originality metric, which indicates that the originality score seems unaffected by any of the factors for which we controlled.

5.5 Analysis of Interaction Logs

To analyze how our visualization conditions affect user behavior, we computed several metrics from either the mouse-movement events or the high-level system-specific events.

5.5.1 Visualization Coverage

We were interested in analyzing how much of the possible space of visualizations our participants covered. To create a metric for visualization coverage, we computed the set of unique visualizations possible within each dataset. We considered all attributes, both visualization types supported by our

system, and all possible aggregation functions. However, we ignored the axis-to-attribute mapping (e.g., a visualization with attribute A on the x-axis and attribute B on the y-axis is considered the same as if the axes were flipped). We then extracted and counted up all visualizations a participant created during a session from the interaction logs. Our final visualization coverage metric is the percentage of possible visualizations a participant created per minute.

Figure 5.7 plots this metric across different factors. Our analysis shows that the type of visualization condition significantly affects the percentage of the total visualizations that a participant covered per minute ($F(2, 40) = 9.847, p < 0.001$). Post hoc tests revealed that the instantaneous condition showed a slight increase in percentage over the progressive condition ($1.553\% \pm 0.690\%$ vs. $1.311\% \pm 0.651\%$, respectively), which was not statistically significant ($p = 0.226, r = 0.251$). However, the blocking condition reduces the percentage of total visualizations covered per minute to $0.845\% \pm 0.449\%$, which differed significantly from both the progressive ($p < 0.01, r = 0.545$) and instantaneous ($p < 0.001, r = 0.740$) conditions. Participants explored more visualizations per minute with the instantaneous and the progressive condition and there is no significant difference between the two.

Note that we did not assign any "importance" or "quality" scores to different visualizations. All visualizations have the same weight, even though some visualizations might be more informative than others or multiple visualizations might convey similar insights. Similarly, our visualization coverage metric is not designed to compare across different users or sessions, such that two users could have the exact same coverage score while looking at completely different parts of the dataset.

5.5.2 Number of Brush Interactions per Minute

While the coverage metric considers the number of possible static visualizations, it does not consider brushing. We measured the brushing interactions by counting the number of brush events from the interaction logs, which we then normalized by the duration of a session. The results, shown in Figure 5.8, demonstrate that the type of visualization condition significantly affected the number of brush interactions per minute ($F(1.335, 26.690) = 17.620, p < 0.0001$). Post hoc tests revealed that the instantaneous condition showed a significant increase in number of brush interactions per minute over the progressive condition (4.640 ± 4.095 vs. 2.108 ± 1.744, $p < 0.01, r = 0.316$), as well as over the blocking condition (1.190 ± 0.737 , $p < 0.001, r = 0.413$). Furthermore, brushing interactions per minute for the progressive condition were significantly different than for the blocking condition

($p < 0.05, r = 0.29$).

Additionally, our results showed a significant between-subject effect for dataset-order ($F(1, 20) = 4.568, p < 0.05$)). A post hoc test revealed that dataset-order 312 increased the number of brush interactions per minute over dataset-order 123 significantly (3.365 ± 3.284 vs. 1.927 ± 2.429, $p < 0.05, r = 0.242$). We hypothesize that this effect is due to the structure of DS3 (census dataset), which in turn leads to a learning effect. DS3, to which users where exposed first in dataset-order 321, has considerably fewer qualitative attributes than the other datasets. Users could not use strategies such as looking for trends in 2D histograms or compute averages across attributes and reverted to using the brushing functionality to correlate across different populations of the data. We often observed users manually cycle through one attribute and look for changes in another attribute. For example, users would create two histograms (e.g., one for "marital status" and one for "education") and then manually select different values in the first histogram (e.g., "married", "widowed") to determine whether the second histogram for that subpopulation differed from the overall population.

5.5.3 Visualizations Completed

We extracted the number of times a visualization was completed—the participant waited the full amount of time specified by the dataset delay until a visualization was completely accurate—from our log data. We then divided this count by the number of interactions that forced a visualization to recompute (e.g., axis change, brushing interaction). This gives us the percentage of times an interaction led to a fully accurate visualization. Figure 5.9 visualizes this metric for different factors. Note that, by definition, this metric is always 100% for the instantaneous condition and we therefore exclude it from post hoc tests. Our results show that the percentage of completed visualizations was significantly affected by the type of visualization condition ($F(2, 40) = 169.972, p < 0.0001$). Post hoc tests revealed that the blocking condition showed an increase of percentage of completed visualizations over the progressive condition ($56.22\% \pm 15.01\%$ vs. $45.99\% \pm 14.14\%$, respectively), which was statistically significant ($p < 0.05, r = 0.296$). In short, people often moved ahead without waiting for the full result.

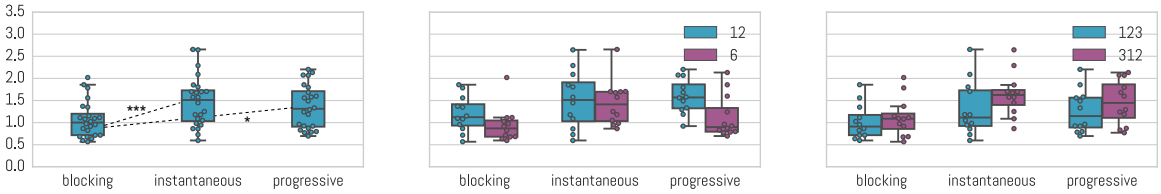

Figure 5.5: Insights per Minute: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for insights per minute (left) overall, (middle) by dataset-delay, and (right) by dataset-order. Higher values indicate better.

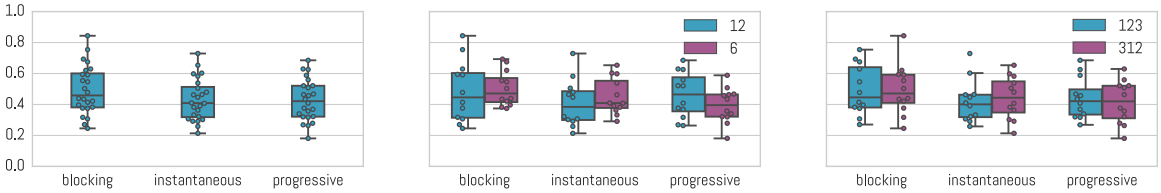

Figure 5.6: Insight Originality: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for insight originality (left) overall, (middle) by dataset-delay, and (right) by dataset-order. Higher values indicate more original insights.

5.5.4 Mouse Movement per Minute

Finally, to compute a simple metric of a participant's activity level, we calculated the distance in pixels the mouse was moved per minute and show the results in Figure 5.10. Our analysis shows that the type of visualization condition significantly affected the mouse movement per minute metric ($F(2, 40) = 5.431, p < 0.01$). Post hoc tests revealed that the instantaneous condition showed a slight decrease of mouse movement per minute over the progressive condition (303.799 ± 128.243 vs. 313.912 ± 139.1299, respectively), which was not statistically significant ($p = 1.0, r = 0.051$). However, the blocking condition reduces movement per minute to 258.060 ± 131.536, which differed significantly from both the progressive ($p < 0.05, r = 0.315$) and instantaneous ($p < 0.05, r = 0.245$) conditions. Mouse movement is a crude indication of a user's activity level, and our test shows that these levels are lowest with the blocking conditions. Again we find no difference between instantaneous and progressive visualizations.

5.6 Perception of Visualization Conditions

During our exit interview, we asked participants to provide feedback about the different visualization conditions they experienced. Most participants liked the instantaneous visualizations best, but

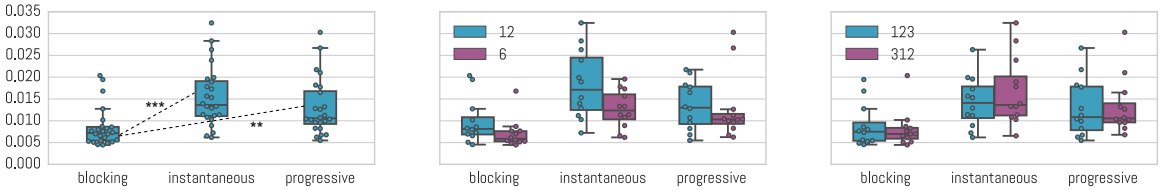

Figure 5.7: Visualization Coverage Percentage per Minute: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for for visualization coverage % per minute (left) overall, (middle) by dataset-delay, and (right) by dataset-order. Higher values are better.

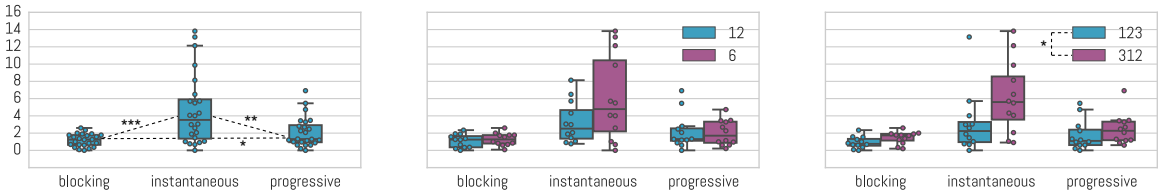

Figure 5.8: Brush Interactions per Minute: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for brush interactions per minute (left) overall, (middle) by dataset-delay, and (right) by dataset-order.

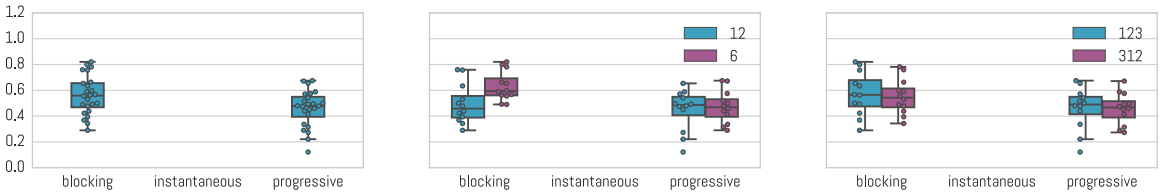

Figure 5.9: Visualizations Completed Percentage: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for for completed visualization % (left) overall, (middle) by dataset-delay, and (right) by dataset-order.

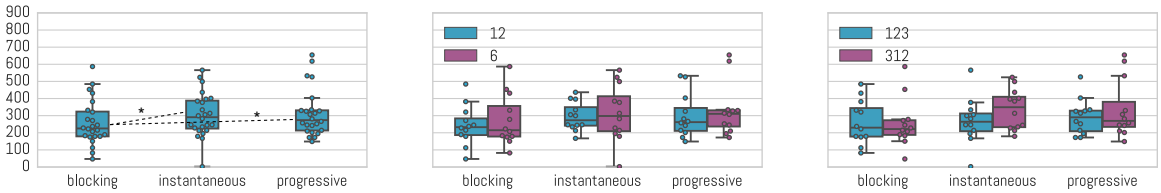

Figure 5.10: Mouse Movement per Second: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for mouse movement per minute (left) overall, (middle) by dataset-delay, and (right) by dataset-order.

preferred the progressive ones over blocking. Below are quotes from our participants that describe these preferences:

"I liked the progressive one better than blocking, but obviously the instantaneous one is best." "I initially felt the loading one [progressive] was weird because graphs change over time. But after seeing this one [blocking] I can appreciate the value of it [progressive]." "[Blocking] slows the process. But when it's one or the others [instantaneous or progressive], I can see one thing and then that naturally leads to the next thing I want to do. Here [blocking] I have to keep track of the last thing that I loaded while I do something else. It's [blocking] just more stilted and less continuous." "You see a rough picture in the beginning and then you can think about it while it's actually finishing. That's way better than just a loading animation."

A few participants expressed positive remarks towards blocking visualizations or commented that they adapted their strategies because of the wait-time:

"The slow one [blocking] made me feel more confident about what I saw, because there is lots of data behind it." "It [blocking] actually helped me to use the time to think. But it also might limit you from finding really interesting facts, because you're going in with an idea, you're using the loading time to come up with things that you think might be true. Without the loading time you could just randomly mess around with the data and find interesting things." "I used the loading time to do something else."

5.7 Discussion

Our analysis provides evidence that overlaps with findings in previous work, such as by Liu et. al. [114], which finds that even small latencies have an impact on user performance in exploratory settings. Our results show that knowledge discovery and user activity measures are negatively influenced by the blocking condition when compared to instantaneous visualizations. This intuitively makes sense and it is widely acknowledged that low latency leads to improved user experience and user performance. The more interesting evidence we present in this paper arises when we compare blocking to progressive visualizations or instantaneous to progressive visualizations.

The difference between the progressive and blocking conditions is that users can see approximate results while a query is ongoing rather than a loading animation. However, the overall delay until a final, 100% accurate visualization arrives is exactly the same for both conditions. Yet, our data

shows that users generated more insights per minute, had a higher visualization coverage percentage, and displayed higher levels of mouse movements when given progressive rather than blocking visualizations. Additionally, the percentage of completed visualizations (i.e., visualizations where users waited the full number of dataset-delay seconds to get the final answer) is higher in the blocking than the progressive condition. This result suggests that users are efficiently using these in-between and approximate visualizations to either pre-process information, extract insights early, or to decide that the result is not what they were looking for and then move on to the next visualization. A participant expressed it this way: *"It's much easier to look at a rough picture of the final data rather than just to see nothing at all. You can start to get an idea of relationships and things like that."*. Based on our analysis we accept hypothesis $H2$ (more insights with progressive than blocking) as well as $H4$ (higher user activity levels with progressive than blocking).

The instantaneous visualization condition represents a hypothetical system that can provide results to the user almost immediately for any query, minimizing interference to the user's thought process. While increasing amounts of data make systems that provide instantaneous visualizations practically infeasible, the evidence presented in our analysis shows that progressive visualizations might perform almost as well. We did not observe any significant differences across all of our metrics for the instantaneous and the progressive conditions except for the brush interactions per minute metric. We can therefore not accept $H1$ (more insights with instantaneous than progressive) or $H3$ (higher user activity levels with instantaneous than progressive).

However, there are some open questions that our study does not address. While our data suggests that approximate visualizations might help users grasp certain characteristics of a dataset rapidly, they simultaneously introduce a set of new challenges. For example, prior work in psychology [43, 44] has shown that even experts sometimes have trouble interpreting standard accuracy metrics (e.g., confidence intervals, error bars). While we did not observe any cases in our study where participants misinterpreted visualizations based on their uncertainty, these cases are also hard to capture. Our participants often reported high-level trends like *"these two categories seem about equal in counts"*, *"there is a slight negative correlation between these two variables"*. How much of a change would need to occur between the approximate and the final visualization before users would retract these insights? Progressive visualizations expose fundamental limitations when it comes to exploration sessions on datasets where it is important to capture insights that are based on outliers or small subsets of the data. It might take a long time to sample those rare datapoints, and visualizations

can therefore take a long time to converge to a view that shows those features prominently.

The second open question is how refresh rates for progressive visualizations affect user interaction and whether or not, in the extreme case, a simple one-sample visualization provides the same benefits. For example, techniques such as the one presented by Stolper et. al. [149] that allow users to decide when to update a visualization might be less distracting than our approach of constant regular updates. There are reasons to believe that progressive visualizations have advantages over approximate visualization that are based on a single sample. For example, users can decide individually when a visualization is accurate enough for their liking or their current task. We observed that some users tend to hold off with reporting insights until after several progressions, especially for visualizations where there was a high visual variance between updates. In other cases, participants waited for uncertainty metrics to be in a range small enough for them to make judgments. One of our participants stated: *"I didn't want to draw any conclusion right away. Especially in this one visualization where the error bar was across the whole graph, I decided to wait."*. While these anecdotal findings make a case for progressive visualizations over simple sampling-based ones, our study does not provide the means for a quantitative comparison between the two.

5.8 Conclusion

We investigated how progressive visualizations affect users in exploratory data analysis scenarios. Through a controlled experiment, we compared progressive visualizations to blocking and instantaneous visualizations and found significant differences in insight-based metrics and user activity levels across all three approaches. We observed that progressive visualizations outperform blocking visualizations in almost all metrics and that progressive visualizations do not significantly differ from the ideal scenario of instantaneous visualizations in terms of generated insights per minute, insight originality, or visualization coverage percentage.

Chapter 6

A System for Progressive Visualizations and Computations

In this chapter we we present Vizdom, an interactive visual analytics system that scales to large datasets through progressive computation. Part of this chapter is based on [41]. I designed and implemented the bulk of Vizdom, its visual language and its progressive computation backend (IDEA).

6.1 Introduction

While the set of tasks that recur in visual data analysis, as well as the tools that support them, are relatively well understood [10, 83], the constantly increasing amount of data has shifted the interaction paradigm away from an interactive approach to one based on batch processing. As Fisher et al. [63] argued, a common way to perform data analytics in the recent past was to use spreadsheet applications to analyze datasets small enough to fit completely in memory. Computations were therefore fast and results were available almost immediately. Users could perform analysis and explore different aspects of the data iteratively and refine findings interactively at a fast pace.

However, in the emerging "big data" era where one million records or more has been stated by the visualization community [159, 59] as a threshold, these conveniences are gone. Most traditional interactive visual analysis systems are not built to support this scale of data. Users need to give up real-time interaction and are forced to perform "big data" analytics through batch jobs written

in scripting or programming languages. These jobs run for minutes or hours, typically somewhere in the cloud, without providing insights on what goes on behind the scenes and without exposing intermediate results. This mainframe-like interaction paradigm is an inherently poor fit for data analytics. The work is exploratory by nature, demanding rapid iterations, and all but the simplest analysis tasks require humans in the loop to effectively steer the exploration process. Empowering analysts to use their domain knowledge to quickly explore different feature-sets or parameters for machine learning tasks, or supporting users to fast-reject unimportant aspects of the data, are crucial to produce high-confidence results quickly.

Multiple different strategies, such as precomputation, prefetching, sampling, and progressive computation, have been proposed to bring interactivity back to visual analytics. Each of these approaches comes with its individual set of advantages and limitations. For example, current large scale, state-of-the-art visual analytics systems [116, 113] achieve interactive frame-rates over large datasets for certain types of filtering queries, but simultaneously limit other analytical functionality at runtime. Brushing and filtering with arbitrary predicates, user-defined data transformations and interactive building of statistical models are all outside the scope of systems that rely solely on precomputation. Progressive computation, where data is processed incrementally in small batches, offers an interesting trade-off between accuracy of results and speed of computation while still supporting many common analytical tasks. The research community has recently regained interest in progressive computation and has attempted to analyze and exploit some of the peculiarities of this approach [65, 149, 62, 64, 58, 137, 41]. However, no attempts have been made to create a general purpose visual analysis system built on progressive computation.

We try to fill this void with Vizdom. Vizdom is an interactive visual analytics system that is entirely based on progressive computation and supports a wide range of reoccurring visual analysis tasks, including coordinating multiple visualizations to create custom exploratory interfaces and derive models through machine learning operators. Vizdom features a pen & touch-based user interface that was designed with a "touch-first" mindset in order to promote a fluid interaction style. Our system scales in both interaction model and computation across varying dataset sizes, ranging from thousands to hundreds of millions of records. In this paper, we show our system design in detail, present how our user interface design as well as our computation engine support analysis tasks in a progressive fashion, and discuss the limitations and advantages of progressive computation over other approaches in the context of visual analysis.

6.2 System Design

When designing Vizdom, we had two main goals: (1) keep frame-rates at an interactive level no matter how large the underlying dataset is and (2) every interaction needs to produce some actionable and understandable artifact within 500ms regardless of the dataset size. The reasoning behind these goals is that we would like to create an interactive environment in which analysts can chain operations together at a pace that matches their thought process without the system slowing down or diverting the users from their current task. Furthermore, we want to guarantee that users do not have to switch interaction strategies based on dataset size. Users should be able to operate at the same pace, with the same tools, and use the same strategies if their dataset has a thousand tuples or a few hundred millions of tuples. At a hight level, we achieve this by processing all queries in a progressive fashion combined with binned visualizations. In this section we first describe our system architecture in detail and then discuss how Vizdom supports various common analysis tasks. In our description, we particularly focus on the design changes we needed to make over a small-scale traditional blocking system (with datasets small enough to compute and visualize results quasi-instantaneous) and discuss advantages and limitations of a progressive approach over other approaches (e.g., precomputation).

6.2.1 Overview

Vizdom uses a client-server architecture where user-issued queries get submitted to the sever and results are then passed back to the client in order to render a visualization. Our client is a pen- & touch-based application with a visual language that consists of two main building blocks: visualizations and tasks. Visualizations are placeholders for views on the dataset and tasks are placeholders for operators that take data as input and either apply a transformation or build a model. Figure 6.1 shows examples of both: (b) a set of three coordinated visualizations, (c) a task operator computing a classification model based on some input. A menu bar on the left-hand side of the tool (Figure 6.1 (a)) allows users to reference attributes of the dataset “a”, create new visualizations “v”, or perform a task “t”. Vizdom features an unbounded zoom- and pan-able canvas where visual elements can be arranged in a freeform fashion. Users can combine the two building blocks on this canvas to support various analysis tasks such as creating a set of coordinated visualizations for brushing or filtering. All interactions on these building blocks are done through direct touch manipulation and data-entry is supported by using handwriting recognition. Each interaction that requires a data-computation

fires off a request to the backend.

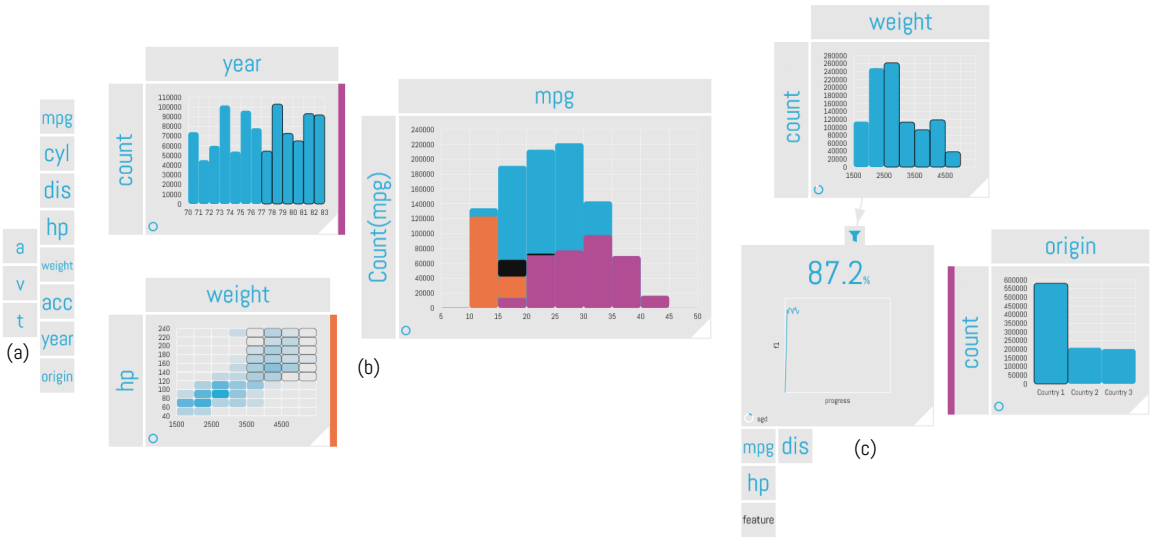

Figure 6.1: Screenshot of Vizdom showing a custom exploration interfaces created by coordinating multiple visualizations (left) and and an ongoing classification model building task (right).

6.2.2 Visual Analysis Tasks

Visual analysis is an iterative process where users frequently switch between a range of tasks (e.g., creating visualizations, filtering data, observing interactions between datasets, etc.). To ensure expressiveness and analytical power, a system needs to support a comprehensive set of such tasks. We use Heer & Shneiderman's taxonomy of interactive dynamics for visual analysis [83] as guideline for coverage in terms of analytical tasks and use the core concepts (derivable visualizations, unbounded space and boolean composition) of [176] as a design structure to expose these tasks in an understandable and approachable way. In this section we describe how common visual analysis tasks are supported in Vizdom.

Visualize

Being able to visually encode data is perhaps the most fundamental operation in visual analysis. Many common visualization types, (e.g., scatterplots) map data points directly to visual elements (one-to-one mapping). For large dataset sizes, this results in occlusion, cluttering, and slow rendering performance. Similar to [116], Vizdom uses binned visualizations to get around this problem. Figure 6.2 shows two such examples computed over a million datapoints each, (a) for a barchart and

(b) for a 2D scatterplot. Visualizations like these can be created through simple drag and drop operations, and tapping on an axis cycles through different aggregation functions (e.g., count, average, etc.). Bins within visualizations can be selected through tapping- or pen-based lasso-gestures.

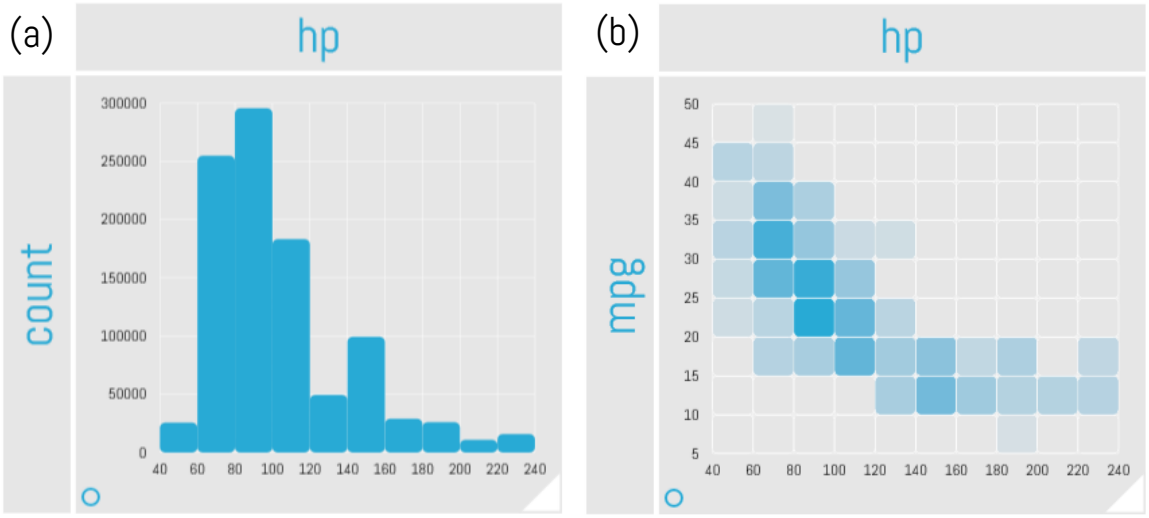

Figure 6.2: (a) barchart, (b) 2D binned scatterplot.

Vizdom computes all of these binned visualizations in a progressive fashion. An approximate result is displayed as fast as possible and is incrementally refined over time. The system computes and overlays visualizations with standard-error based 95% confidence intervals (bar-charts) or margins of error (2D scatterplot) and displays a progress indication in the bottom left corner of a visualization. One configurable parameter for binned visualizations is the number of bins to display. Because of the systems progressive nature, we do not know the minimum and maximum value of an attribute upfront, making it impossible to accurately calculate the bin-size at the start of a computation. Vizdom uses a binning algorithm that computes the bin-size according to the number of bins requested as well as the minimum and maximum estimations of a first chunk of data. If any subsequent chunk of data contains datapoints that lie outside of this initial estimation, we add bins of the same size to accommodate these new values. This algorithm relies on good initial minimum and maximum estimates. In the limit, the algorithm can produce histograms with a large number of bins if the dataset contains outliers that are far outside of the initial bounds estimate. In practice, and for the tested datasets, however, this algorithm is fairly robust. Our backend computes binned visualizations by processing one chunk of data at a time and stores intermediate results between iterations. The backend sends the increasingly more precise binned visualization to the client after each chunk

of data is processed. The client then renders the visualization.

By using binned visualizations coupled with progressive computation, we achieve perceptual and interactive scalability [116]. Binned visualizations reduce the amount of visual elements that need to be rendered while preserving overall patterns in the data as well as local features such as outliers. Additionally, progressive computation guarantees that Vizdom can display some actionable, although approximate, visualization independent of the size of the dataset within interactive timeframes.

Coordinate & Organize

Due to Vizdom's unbounded 2D canvas, organizing visualizations or tasks is as simple as arranging them on the screen through direct touch manipulation. We support two types of links to coordinate multiple views or tasks: persistent and transient links. Persistent links are created through a pen-gesture (drawing a line between two visual elements) and transient links are implicitly formed by proximity of elements. We map these link types, coupled with selections, to different types of operations such as filtering, brushing or specifying input parameters to tasks.

Filter

By coordinating visualizations with persistent links, users can create detailed interactive visual display networks in which each visualization can act as a filter for others. A set of linking operations turns the three visualizations in Figure 6.3 (a) into a custom filter interface (b). In the example, our analyst now uses this network to filter the displacement (dis) visualization to only include cars with a high miles per gallon (mpg) value and low horsepower (hp) along with cars that originate from "Country 1." Vizdom supports filtering over arbitrarily long chains of visualizations and includes complete boolean logic (and, or, not). Note that whenever selections in upstream visualizations change, downstream visualizations are recomputed. Vizdom's client passes the filtering predicate along to the server which then evaluates it for all tuples in a chunk of data to see if a specific tuple needs to be included in the current computation. We want to emphasize that through this technique, Vizdom supports filtering on arbitrary predicates at runtime unlike systems built using the precomputation approach.

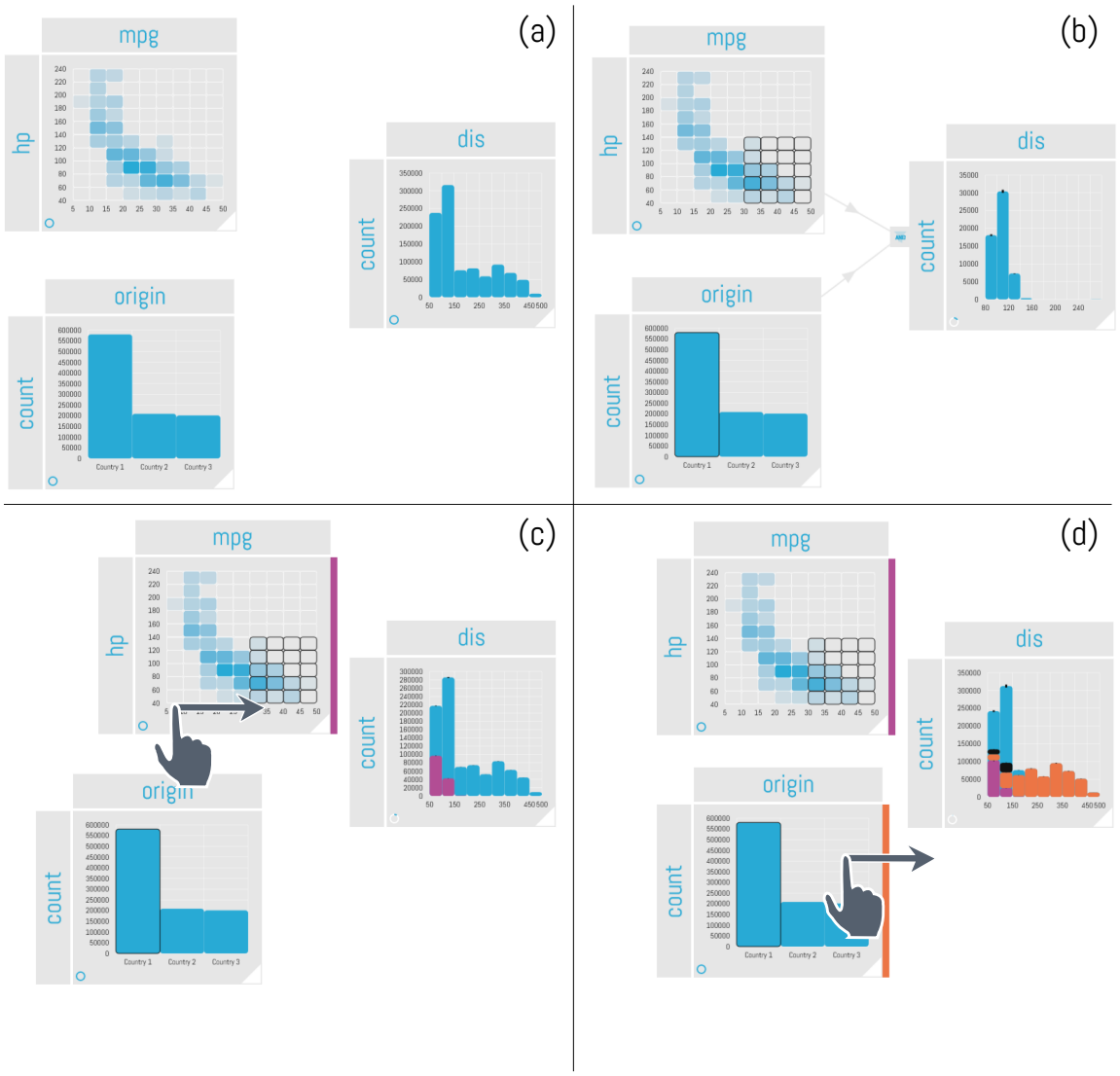

Figure 6.3: Examples of filtering and brushing. (a) Three uncoordinated visualizations. (b) Visualizations coordinated through persistent links where upstream selections serve as filter predicates for downstream visualizations. (c) Example of a brushing interaction through a transient link that is based on proximity. (d) Multiple brushes on the same visualization.

Brush

Dragging two visualizations close together implicitly creates a transient link between them. Moving the visualizations further apart again removes the linkage. In Vizdom, this interaction technique triggers a linking and brushing operation, where selected data points are highlighted in another visualization. In Figure 6.3 (c), the horsepower (hp) vs. miles per gallon (mpg) scatterplot is dragged close to the displacement (dis) histogram. This interaction reveals that cars with high miles per gallon values and low horsepower are all within the lower end of the displacement range. A second move interaction (d) reveals two things: (1) there is little overlap between cars with a high mpg/low horsepower and cars from "Country 1" and (1) cars with a high displacement value come all from "Country 1". Vizdom assigns a unique color to each incoming brushing-visualization and colors data points black if they are associated with two or more brushes. Heights of bars as and rectangle areas for 2D scatterplots are scaled accordingly to reflect the number of data points falling within a specific brush. Changes in brush-selections automatically force the system to recompute a downstream visualizations. Similar to filtering, the backend evaluates brushing predicates for each chunk of data to determine the number of tuples that should be highlighted. By combining filtering and brushing operations, users can create arbitrary complex boolean brushing predicates.

Navigate

Navigational patterns such as the widely cited "Information Seeking Mantra" [142] (i.e., overview-first, zoom and filter, details on demand) are directly supported through coordinated visualizations and filtering or brushing operations. Figure 6.4 shows an example where a user creates a filter-chain to stepwise zoom into a specific range of the datapoints.

Derive

When analyzing a dataset, analysts oftentimes create or derive new variables from existing values. Examples range from simple transformations (e.g., normalizations or log transforms) to computing complex statistical models over the dataset. Vizdom supports two types of such derived values: simple aggregation functions in visualizations and classification models through machine learning operators. We will focus on the latter in this section.

Binary classification tasks are created by dragging out one of the available tasks from the task

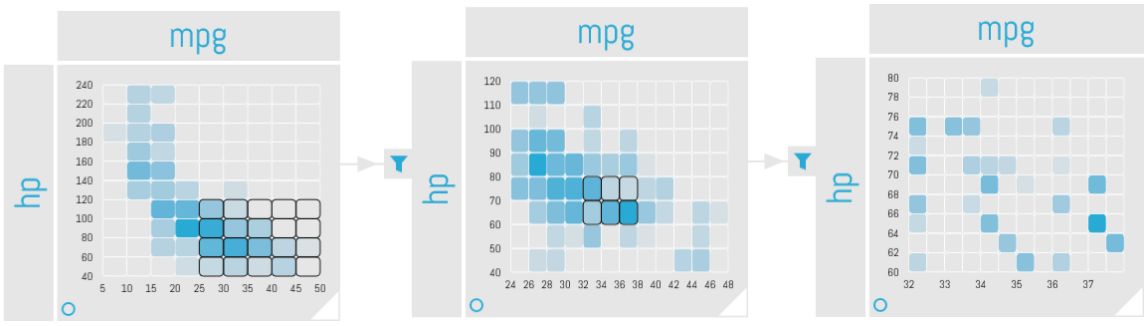

Figure 6.4: Stepwise zooming into a specific range of datapoints by using a set of linking and filtering operations.

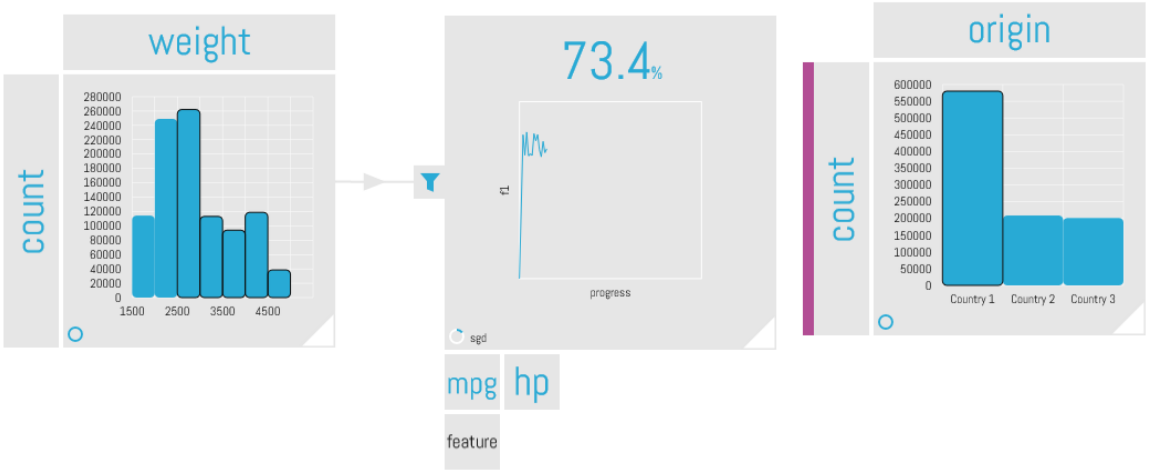

Figure 6.5: A binary classification task using a support vector machine based on stochastic gradient descent. The model is trained progressively based on the attributes mpg and hp and tries to predict if a car is from "Country 1". Furthermore, while building this model, only cars from a specific weight range are included. The task's view shows how accuracy improves while progressively training on more and more samples.

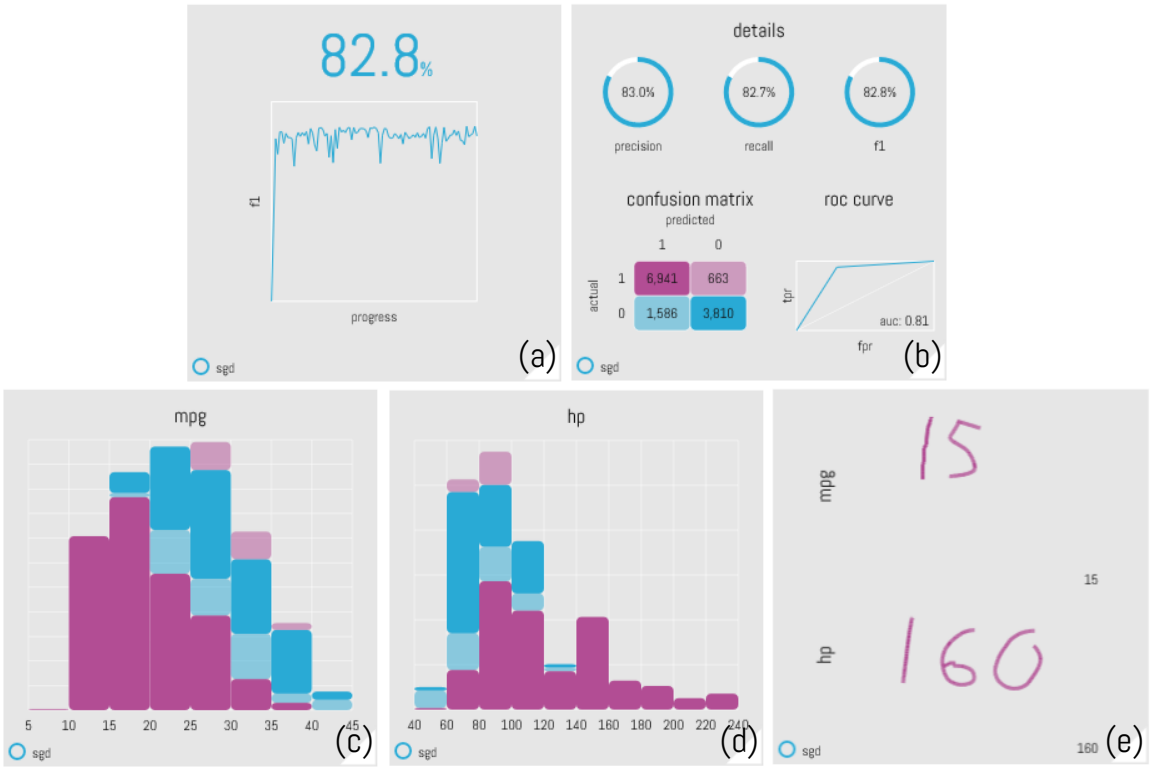

Figure 6.6: Five different views of a classification task. (a) F1 score over time. (b) Detailed performance statistics. (c,d) Histograms of features based on test data and shaded by classification result. (e) Testing panel to manually create samples (through handwritten recognition) that can be classifed by the current model.

list (Figure 6.1 (a) under item "t"). Such a task needs two inputs before it can start computing a model: (1) the set of features to train over and (2) a label function that indicates which training tuples are positive or negative examples. In Vizdom, features of a classification task can be specified by dragging and dropping attributes to its feature slot, and labels are specified through transient links and selections. Figure 6.5 shows an example. We created a binary classification task (middle) using a support vector machine based on stochastic gradient descent ("sgd") that tries to predict if a car comes from "Country 1" (right) based on its miles per gallon (mpg) and horespower (hp) values. Furthermore, we filter this task to only include cars from a specific weight range (left). Our system trains this model incrementally by feeding it more and more training samples in each iteration and notifies the user about the current state of the model after each chunk of data is processed. The task's current view shows that the model's current accuracy (F1 score) is at 73.4% and displays how this value has changed over time. The system retains a certain fraction of the tuples as testing data to evaluate the models performance and compute accuracy metrics.

Model building in machine learning is an iterative process were users frequently switch between analyzing the quality of a model and changing parameters or features of the training process. Similar to ModelTracker [11], Vizdom exposes some of the internals of classification operators to make this transition less disruptive to the users. Unlike ModelTracker, however, which shows information about individual training tuples, we again aggregate samples for these debugging views to accommodate for large datasets. Figure 6.6 shows the individual views of a classification task. Users can switch amongst views through tapping gestures. As seen in the previous example, the first view (a) shows how the testing accuracy of the model (F1 score) evolves over time and with increasing training data. The second view gives more detailed statistics about the classifier, such as precision and recall, ROC curve, and confusion matrix. Subsequent views are based on feature attributes. For each feature (in our case mpg (c) and hp (d)), the system depicts how the testing dataset was classified. The color coding follows that of the confusion matrix in (b). Dark purple represents true-positives, light purple represents false-negatives, dark blue represents true-negatives, and light blue represents false-positives. These views can help analysts understand were the errors in the model's classifications come from. In the example, we might read from (d) that most of the higher horsepower (hp) cars are classified correctly and almost all of the classification errors are made with lower end horsepower cars. The last view (e) serves as an input panel were users can manually test how certain combinations of features are classified. As shown in the example, we input values for

Figure 6.7: Examples of derive operators. (a) A user defined calculation. (b) A definition operator which defines a new derived attribute based on user specified brushes. (c) Visualization of the new attribute created in (b).

the two features through handwritten ink and see that our model classifies a car with 15 mpg and 160 hp as a car from "Country 1". We again use color coding here: the input strokes turn purple if classified as positive and blue if negative.

Other examples of derive operators include user defined calculations or definitions. Figure 6.7 (a) shows an example of the former, where a user creates a new derived value based on a computation involving "horspower" and "weight", and (b) the latter, where a user creates a new categorical variable based on selections in visualizations. The output of all operators can be renamed and used as any other attribute. By dragging the output of the definition operator in Figure 6.7 (b) a user can visualize the distribution of values of this new attribute (c).

6.2.3 Process & Provenance

Vizdom implements some simple annotation and recording tools through digital ink and since the system has an unbounded 2D canvas, past analysis results can be revisited by zooming and panning. However, we did not not focus on various other tasks, such as exporting or sharing of views for collaboration or storytelling, within "Process & Provenance" category of Heer & Shneiderman's taxonomy. We believe that well-known tools and techniques that support these tasks could be adapted to a progressive computation-based system with almost no changes.

6.2.4 Backend

In order to achieve low latency for the type of ad-hoc queries that Vizdom allows, our processing backend (called IDEA) relies on *approximate query processing* (AQP) techniques, which provide query result estimates with bounded errors. The current implementation of IDEA, which operates on raw textual files, requires a loading step were we randomly shuffle an imported dataset, if it's

Figure 6.8: Shows a summary report generated by IDEBench [53] comparing four different analytical database systems over three dataset sizes and a time requirement of 1s.

small enough, or we create an in-memory sample. This structure allows us to generated a random stream of tuples at runtime. We envision that in the future this approach can be extended to directly connect to existing data warehouses by using sampling strategies [127, 128] to access random streams of tuples. Each user-issued query is computed incrementally through chunks of data at a time where the size of a chunk is a configurable parameter. The server sends messages to the client with the current incremental result after each chunk has been processed. Additionally, complete and partial results are cached and shared across clients to maximize reuse opportunities.

An example illustrates this process for a simple aggregation query, where the results are computed in such a progressive or online fashion [85]. Let's assume we have a dataset with 1,000 numbers and our chunk-size is set to 100. A client creates a request to compute the average over this dataset. Our sever starts by getting a chunk of 10 tuples from the data-source, computes the average, and stores the average along with how many tuples have been processed so far in the system's cache. The server can then return the first incremental result to the client: an estimate of the average based on 10 samples. Note that we ignore computing error metrics in this example for simplicity. On the next iteration, the server requests the next chunk of 10 numbers and updates its currently stored incremental result and again notifies the client. If our random data-source operates without replacement, the server can stop its computation and send the final answer to the client after the 100^{th} iteration. At no point did the server have to hold more than a 10 input tuples in memory and it only had to store

two numbers (current average and number of tuples processed) to represent the current state of the computation. We extend this same principle to machine learning operations where we train models incrementally by feeding them more and more training samples in each iteration and present the current state of the model to the user after each chunk of data is processed.

In related work, where we argue for a new benchmark for interactive data exploration [53], we compare the performance of IDEA's progressive visualization computation engine against other leading systems. Figure 6.8 highlights this comparison. We find that IDEA is able, unlike a traditional database system like MonetDB [22], to return approximate answers for all tested queries within 1s (*Time Violated* is 0) independent of the dataset size (100 million, 500 million and 1 billion records). Additionally we note that for other metrics, such as the percentage of bins in a visualizations that are missing due to the approximation or the approximation error itself, IDEA is either comparable to other AQP systems (SystemX) or outperforms them (XDB [111]).

A more detailed and complete description of IDEA, its internals and background on AQP systems can be found in our related work [42, 68].

6.3 Conclusion & Future Work

Current large scale, state-of-the-art visual analytics systems (e.g., imMens [116], [113]) are based on precomputation. While these systems support real-time interaction for certain queries on datasets with billions of tuples, they compromise in the area of analytical functionality. Multi-level, ad-hoc brushing and filtering of visualizations, data transformations that are not defined at loading time, and interactive building of statistical models are all outside the scope of systems that rely on precomputation. With Vizdom, we designed a system based on interactive progressive computation that works on similarly sized datasets sizes without these typical restrictions.

However, progressive computation has its own set of limitations. Incremental and approximate answers, as implemented by Vizdom, can be hard to interpret. As shown in prior work [45, 43, 44], even experts make mistakes when interpreting standard statistical accuracy metrics. Advances in uncertainty visualizations are needed to fully mitigate some of these issues. Additionally, not all algorithms and analysis tasks lend themselves well to progressive computations. Some methods are highly susceptible to outliers, or are not guaranteed to converge to the true result when implemented progressively. For example, if an analyst needs to detect extremely rare outliers in a large dataset

or wants to compute the exact minimum or maximum value of an attribute, progressive systems do not offer speedups over blocking systems. In the future, we are planning to extend Vizdom by implementing more types of visualizations (e.g., maps), adding other types of machine learning operators, and supporting user-defined transformations, either through handwritten math notation or visual scripting interfaces.

In this chapter, we presented Vizdom, a progressive computation system that enables real-time visual analytics that scales across dataset sizes ranging from thousands to hundreds of millions of tuples. We support a wide range of common visual analysis tasks, such as creating coordinated views for filtering and brushing, and deriving models from the input data through machine learning operators.

Chapter 7

Investigating the Effect of the Multiple Comparison Problem in Visual Analysis

In this chapter we study the effect of the multiple comparisons problem in visual analysis by measuring accuracy of user reported insights on synthetic datasets. We provide some early insights and simple techniques as to how this problem could be mitigated. The bulk of this work is also substantially similar to [178] and a smaller part is based on [180]. For [178] I was the first author and responsible for the research direction, implementation, study design, analysis and the bulk of the writing. Zheguang Zhao contributed the mathematical methods and implementation on how to generate synthetic datasets (part of Section 7.4.2), how to run statistical tests (part of Section 7.6.4) and the background on the multiple comparisons problem in statistics (Section 7.3.3). For [180], I contributed the design and implementation of the visual language and user interface.

7.1 Introduction

Here is a thought experiment. Imagine a game where you roll a pair of dice and win if you get two sixes. The probability of winning is 1/36. Now let's change the game mechanics. Instead of just rolling once, you can continue rolling the dice. You might get a three and a four in your first roll.

You did not win, but you keep going. On your 100^{th} try you get two sixes and win. Everyone will win this game eventually. The probability of winning after an infinite number of rolls is 1. Even after just 100 rolls the chances of winning are over 94%.

Similar to others [29], we argue that the same thing happens when performing visual comparisons, but instead of winning, an analyst "loses" when they observe an interesting-looking random event (e.g., two sixes). Instead of being rewarded for persistence, an analyst actually increases their chances of losing by viewing more data visualizations. This concept is formally known as the *multiple comparisons problem* [19]. Think of each dice roll as a comparison between a visualization (the rolled outcome) and a mental model of something interesting (e.g., two sixes). As more comparisons are made, the probability rapidly increases of encountering interesting-looking (e.g., data trend, unexpected distribution, etc), but still random events. Treating such inevitable patterns as insights is a *false discovery* (or Type I error) and the analyst "loses" if they act on such insights. Because there was nothing to discover. The data was random.

Unlike the dice example, empirical datasets are an unknown weighted product of noise and signal and thus not totally random. The data analyst's "game" is to detect data patterns that are real and ignore those that are spurious. Unfortunately, the difference between the two may be small or non-existent. This creates a second way for an analyst to lose: it is a *false omission* when a real pattern is ignored because it looks uninteresting. False discoveries and omissions might be rare, but, due to the multiple comparisons problem, they are increasingly likely to occur as analysts look at more visualizations.

To further demonstrate the multiple comparison problem in exploratory visualization, we present a representative scenario using random data; we consider non-random data in our experiments (section 7.4).

Jean works at a small non profit organization. Every year they send their donors a small thank-you gift and want to repeat that this year. From past experience, the organization knows that only half of all new donors become recurring donors. Jean suspects there might be a relationship between retention rate and thank-you gift type. Maybe better-liked gifts trigger repeat donations. Jean uses his favorite visualization tool to explore data from the last 10 years. He first looks at the 2006 data and sees that slightly less than half of the new donors donated again (Figure 7.1 (a)). Then Jean inspects the 2007 data and sees the same result (Figure 7.1 (b)). After scanning through all the other years and coming to similar conclusions, Jean looks at 2016 (Figure 7.1 (f)). Instantly

he sees that this visualization is much different than the others, depicting a noticeable shift toward more repeat-donors. Jean gets excited. He believes he has figured out a way to improve the donor retention rate. People liked the USB-drive his company sent out that year so much that they stayed loyal to the organization. Even though this gift is the most expensive one, it is worth it to send it again this year. Long term donors bring in a lot of money.

Is Jean's insight correct? It is not. The dataset Jean looked at was generated by sampling from a uniform distribution. It was completely random. We controlled the process that generated this dataset and there was no signal that related gifts to donor retention rate in any form. Jean's false discovery led to a spurious insight. By doing ten comparisons he increased the likelihood of finding a seemingly interesting pattern in data that was the product of randomness.

The multiple comparison problem is well-known in statistics. There are various common approaches for following up Jean's exploratory analysis with statistical analysis. Each comes with its own pitfalls and disadvantages. We introduce these via co-workers with whom Jean shares his insight: Noemi (*confirmation; same dataset*), Hasan (*confirmation; validation dataset*) and Kendra (*mixing exploration and confirmation*).

Noemi transforms Jean's insight into something statistically testable. She defines a null hypothesis: becoming a repeat-donor is just as likely as being a onetime-donor. She tests if the 2016 data rejects this. The p-value turns out to be 0.028 indicating a significant effect (for a significance level of 0.05). Noemi arrives at the same, wrong, conclusion as Jean. By confirming a hypothesis on the same dataset that has informed that hypothesis, she introduced systemic bias.

Like Noemi, Hasan converts Jean's insight into a statistical hypothesis but tells Jean it's unsafe to test on the same dataset. They agree to send the USB-drive again this year and re-run the test after obtaining new retention data. The test comes out as not significant, refuting Jean's initial insight. Hasan got it right. Running confirmatory analysis on a validation dataset is statistically sound. However, obtaining new data is, in practice, often expensive, time-consuming or even impossible.

Kendra takes yet a different approach. She suspects that Jean probably did a bunch of visual comparisons prior to arriving at his insight. These comparisons need to be incorporated in any confirmatory analysis done on the same dataset to prevent errors caused by the multiple comparison problem. She asks Jean to meticulously recount what he did and maps all of Jeans visual comparisons to equivalent statistical hypotheses. Jean did a total of ten comparisons: one explicit for the year 2016, and nine implicit, unreported, for the years 2006 - 2015. She runs statistical tests on the original

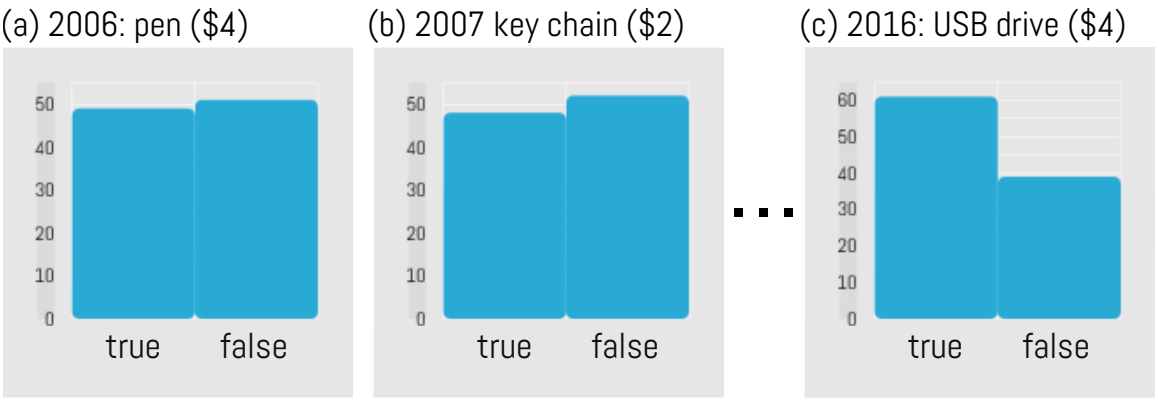

Figure 7.1: Examples of the multiple comparison problem in visualizations of a randomly generated dataset. A user inspects several graphs and wrongly flags (c) as an insight because it looks different than (a) and (b). All were generated from the same uniform distribution and are the "same". By viewing lots of visualizations, the chances increase of seeing an apparent insight that is actually the product of random noise.

dataset using the Benjamini-Hochberg [20] procedure to control for such a multiple comparisons scenario. The corrected p-value for the year 2016 equals 0.306. Kendra deems the test insignificant and informs Jean that his insight is likely due to random noise in the data. While Kendra's approach (*mixing exploration and confirmation*) requires Jean to remember his visual analysis session in detail, it also allows for statistically valid confirmation of his findings using the same dataset.

This paper presents an experiment that quantifies the accuracy of user reported insights, where we define insights as observations, hypotheses and generalizations directly extracted from the data. We acknowledge this definition is narrow. Insights gained from visualizations can be much broader and multifaceted. Visualizations help users gain a deep understanding of a specific problem domain. However, we purposefully limit ourselves to this subset of insights because there is no ambiguity of what correctness means. By using synthetic datasets with known ground truth labels, we can assign a binary score to each insight: it is either true or not. We then compute an accuracy score for each participant by dividing the count of correct insights by the number of all insights.

We follow up by manually mapping insights to corresponding statistical tests and evaluate the three different confirmatory approaches just illustrated. We report and discuss how an approach that validates user insights on the same dataset as used during exploration inflates the false discovery rate due to the multiple comparisons problem. We show that these errors are dramatically reduced by validating on a separate dataset. Finally, we demonstrate that by accounting for all visual comparisons done by a user during exploration, the approach of *mixing exploration and confirmation*,

can achieve similar results to using a separate dataset.

7.2 Why the visualization community should care

In theory, there is a clear divide between exploratory and confirmatory data analysis methods [158]. The goal of the former is to browse through data letting visualizations trigger potential insights and hypotheses. The latter extends this process with a directed search intended to confirm or reject insights and hypotheses given a priori [156]. Within this realm, mistakes are acceptable in the exploratory phase because it is expected that the confirmatory phase will correct them.

In practice, however, the distinction between the two methods can be blurry. Oftentimes users will unwittingly switch between the two and "convert results from investigative analysis into evidence" [99]. There are also pitfalls associated with this approach that are unobvious to non-statisticians; for example doing confirmatory data analysis on the same dataset as the exploratory analysis introduces systemic bias known as data dredging or p-hacking [81]. While splitting a dataset into exploratory and confirmatory parts gets around that problem, it significantly lowers the power of any test due to smaller sample sizes. And without using advanced controlling procedures for multiple hypothesis error that allow for incremental testing [179], iterative switching between exploration and confirmation can not be done. Standard procedures such as Bonferroni [50] can only be applied once per dataset.

The blurring of the lines between exploratory and confirmatory analysis is arguably magnified by how both commercial visualization systems and research prototypes are advertised: "...uncover hidden insights on the fly...", "...harnesses people's natural ability to spot visual patterns quickly, revealing everyday opportunities and eureka moments alike..." [153], "...no expertise required..", "...more powerful approach to data exploration gets you to answers faster..." [154], "...an interactive data exploration system tailored towards "fast-forwarding" to desired trends, patterns, or insights, without much effort from the user..." [146]. We believe that such statements instill a false sense of confidence and reliability into visualizations and exploratory data analysis and the insights obtained through them. This might be especially true for tools that target *data enthusiasts* - people who are "not mathematicians or programmers, and only know a bit of statistics" [78].

7.3 Related Work

We relate and compare our work to prior art in the areas of *Insight-based Evaluation*, *Visual Inference and Randomness* and *Multiple Comparisons Problem in Statistics*.

7.3.1 Insight-based Evaluation

Many have argued that the primary goal of information visualizations is to provide insights [30, 125, 33], and, unsurprisingly, the visualization community has increasingly opted to use insight-based evaluation methods [115, 72, 76, 175, 138]. In addition to measuring directly how well systems achieve this main goal, these methods also allow for ecologically valid comparisons between designs. But as Plaisant has argued [130], evaluation methods should not only help estimate how efficiently a new technique reveals trends or phenomena from data, but also provide estimates of the potential risk for errors. Insight-based evaluation methods clearly address the former but have for the most part ignored the latter. Amongst other risks for misinterpretations [26], visualizations are subjective and can be misleading [162]. A visual feature can be perceived by users even though it is merely the product of random noise in the data. And the risk for this increases with the number of visualizations that are being inspected.

Van Vijk [162] introduces an economic model that equates the investments associated with a visualization (e.g., initial cost to create a visualization, perception and exploration costs) with the return on those investments (i.e., the total knowledge gained by a user). We want techniques that optimize this model for low cost and high investment return. Usability studies and controlled experiments on benchmark tasks help us understand the cost of a particular design, and insight-based evaluations attempt to assess the other side of this equation. However, measuring the number of insights without any quality weighting paints an incomplete picture and has been mentioned specifically as a limitation of such study designs [175].

Several proxy metrics have been proposed, for example, insight "value" as assessed by domain experts [138] or "originalty" scores (how often the same insight was reported). We augment this work with an experimental method based on synthetic datasets that assigns each insight a binary quality score: true or false. Our definition of insights is comparatively narrow [125, 33] and only encompasses observations, hypotheses and generalizations directly related to the data and not on any other sources such as prior knowledge or domain expertise.

7.3.2 Visual Inference and Randomness

Buja et.al. [29] outline the parallelism between quantitative testing and visual analysis and argue that the term "discovery" (or insight) in visual analysis can often be equated to "rejection of a null hypothesis". Seeing an interesting upward trend in a visualization, for example, can be taken as a rejection of uniformity. We base our study on the same notion and manually extract null hypotheses from user reported insights. Follow-up work [119] shows that visual inference can perform comparably to quantitative testing under certain protocols. Similarly, there is a large body of work in visualization and graphical perception covering individual value estimation [36], correlation [134, 112], searching [73] or regression [40]. The consensus is that given the right visualization for the task, users are fairly accurate in their estimations. However, none of this work analyzes if or how the problem of multiple comparisons affects visual inference when comparisons are done in series.

People are known to judge randomness inaccurately [103, 75] and see patterns where there are none [88]. Many well-known examples and studies illustrate this, such as the *Gambler's fallacy* [152], the *Hot-hand fallacy* [13] or the *birthday paradox*. The "Rorschach protocol" [29, 167] can be used to test people's tendency to see patterns in random data.

In this work, we consider the interplay between visual inference and judgment of randomness. *Gambler's fallacy* is famously known in the game of roulette where people might miss-interpret a streak where the ball falls in red 20 times in a row as a pattern. Humans are good at spotting such patterns but are bad at judging that this outcome is not more or less uncommon than any other red and black sequence of the same length. Data exploration allows users to browse a large number of visualizations quickly especially when using automatic visualization recommendation systems [163, 145]. While scanning through lots of visualizations we might find one with an interesting pattern but are unaware and do not consider that this could be the artifact of random noise.

7.3.3 Multiple Comparisons Problem in Statistics

When more than one hypothesis is considered at once, the risk of observing a falsely significant result increases. Formally, this phenomenon is known as the *multiple comparisons problem* or *multiple hypothesis error*, or data dredging and p-hacking. Suppose we are looking for indicators in a census dataset that affects salary distribution in the United States. To examine an effectiveness of a factor such as "age" or "state", we set up the corresponding *null hypothesis* that states the proposed

attribute has no correlation with the salary distribution. We then use a statistical test to infer the likelihood of observing a spurious correlation under the null hypothesis. If this likelihood, commonly referred to as the p-value, is lower than the chosen significance level such as 0.05, then the null hypothesis is rejected, and the alternative hypothesis that the proposed attribute is correlated with salary is deemed statistically significant.

However, if we keep searching through different indicators in the dataset with the same methodology, we are guaranteed to find a statistically significant correlation within a finite number of tests. For example, choosing a significance level for each test of 0.05 means that statistically we have a 5% chance of falsely rejecting a given null hypothesis; even if the dataset contains completely random data, we would, on average, falsely discover a spurious correlation that passes our significance level after only 20 hypothesis tests.

Several techniques exist to control for multiple hypothesis error. Procedures such as Bonferroni [50] control for *family-wise error rate* (FWER), which is the probability of incurring *any* false discovery given the hypotheses. For the previous example, with Bonferroni it is possible to lower the FWER from 100% to just 5%. FWER procedures in general provide the safest control.

The downside of FWER procedures is their statistical power decreases as the number of hypotheses increase in order to compensate for the risk of making any error. A strict control target as FWER may well be applicable to situations where the false discovery is costly, such as medical trials. However, other common data science applications are more concerned with the accuracy of observed significant results than the likelihood of making an error. Thus the *false discovery rate* (FDR) is proposed as an alternative control target which specifies the expected proportion of false discoveries among only the discoveries made (i.e. the rejected null hypotheses), instead of all the hypotheses examined. An FDR procedure such as Benjamini-Hochberg [20] bounds the ratio of false rejections among only the rejected tests to be say 5%. In general FDR is a weaker guarantee than FWER but has shown tremendous benefit for discovering truly significant insights. Recent work on FDR and its variants such as the marginal FDR (mFDR) improves over Benjamini-Hochberg for dynamic settings [66] and specifically interactive data exploration [179].

The multiple comparisons problem also applies to data mining and machine learning where it manifests itself as overfitting. A common practice is for models to learn from training data and then evaluate them on an independent validation dataset. However the validation dataset is not reusable in this simple form, because otherwise the validation metric inflates with multiple comparisons

error. Recent work presents a remedy to reuse the hold-out dataset in adaptive data analysis with techniques from Differential Privacy [52].

In summary, the multiple comparisons problem is well known and studied in statistics but is is very much unregarded in the visualization community. Oftentimes we have a mental model of what is of interest to us when interpreting a visualization. We compare our mental model against the visualization we are observing. If the visualization matches our mental model for interestingness - it shows a particular trend or a distribution - we will notice it. Perhaps unconsciously, but we are doing comparisons. While these comparisons are not a well-expressed mathematically, they still are tests and subject to the same multiple comparisons problem as statistical tests are.

7.4 Experimental Method

The aim of this work is to investigate the effect of the multiple comparison problem in visual analysis. To achieve this, we need to evaluate the accuracy of user reported insights. We designed an experiment where an insight is an observation, hypothesis or generalization that could be directly extracted from the data and that did not require prior knowledge or domain expertise. Our study followed an open-ended protocol where we let participants explore a synthetic dataset and instructed them to report their reliable insights by writing them down. We believe this workflow models common real-world data exploration scenarios where users report interesting observations to discuss later with colleagues, to draw a conclusion or take an action, to analyze further or to convert into a publication. By using synthetic datasets generated from mathematical models we control, we can match ground truth labels to user observations and label each reported insight as being true or false. We use a full-factorial 2 dataset-types (*shopping* and *sleep*) $\times$ 2 dataset-sizes (300 and 1000 records) experiment. The remainder of this section provides detailed description about the individual pieces of our experimental design.

7.4.1 System

The interactive data exploration tool used in our study is based on systems like Vizdom [41] and PanoramicData [176]. Similar to imMens [117], our tool supports two types of binned visualizations, bar-charts and heat-maps whereas actual raw values of any visual element can be revealed by selecting bars or bins in visualizations. It offers common tools such as brushing and filtering and supports

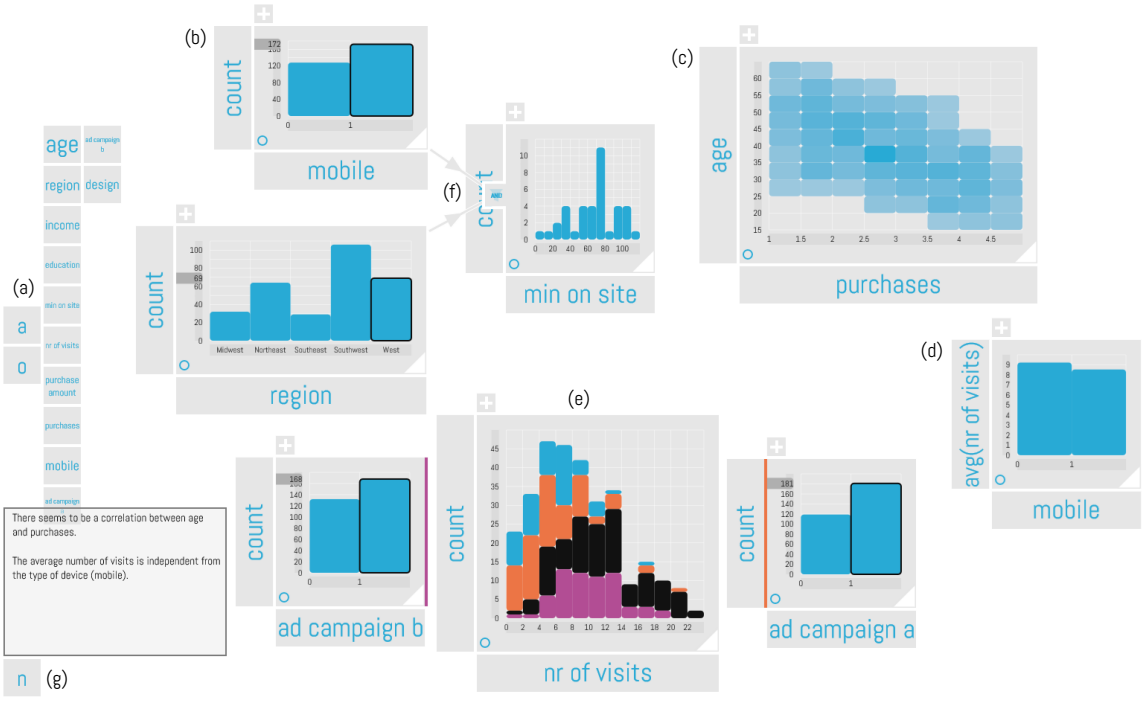

Figure 7.2: Screenshot of the visual analysis tool used in our study. The tool features an unbounded, pannable 2D canvas where visualizations can be laid out freely. The lefthand side of the screen gives access to all the attributes of the dataset (a). These attributes can be dragged and dropped on the canvas to create visualizations such as (b). Users can modify visualizations by dropping additional attributes onto an axis (c) or by clicking on axis labels to change aggregation functions (d). The tool supports brushing (e) and filtering (f) operations. Where filtering operations can be arbitrarily long chains of Boolean operators (AND, OR, NOT). The system offers a simple textual note-taking tool (g).

creating and modifying visualizations through drag and drop gestures similar to Tableau [153].

7.4.2 Datasets

Our experiment uses three dataset-types from different domains. A dataset-type specifies the attributes and value ranges the data consist of. The first dataset-type (*shopping*) contained customer information from a fictional shopping website. This dataset-type contained 12 attributes (4 quantitative, 3 nominal, 5 ordinal), and included information like ages of customers, incomes, region customers are from, average purchase amount per month and average minutes they spend on the site. The second one (*sleep*) consisted of data from a fictional study about people's sleep patterns and contains 10 attributes (5 quantitative, 1 nominal, 4 ordinal). Some example attributes include average hours of sleep, time to fall asleep, sleep tracker usage, fitness and stress levels. The third dataset-type (*restaurants*), used only to introduce participants to the system, contained ratings and attributes from restaurants of four different cities.

From these dataset-types we generated actual datasets of two different sizes: 1000 and 300 entries. The first size is derived from the fact that roughly half the datasets found on a popular website collecting machine learning-related datasets [90] have less than a 1000 records. The second size originates from Anthoine et al. [12] which analyzed 114 recent medical publications and found that the median sample size of studies in those publications was 207. While there are many others, these models, we believe, represent two common scenarios for visual data exploration and analysis: a user wants to draw conclusions from a study or a user wants to explore a dataset to inform feature selection for a machine learning task.

Our scheme for synthetic data generation is similar to [7] and addresses several challenges. First, in order to convey a realistic context to the user, the generated data should retain the domain-specific properties of the empirical distributions. For example, the synthetic sample of "age" should not be negative, and the sample mean should be within reasonable range. To this end, we extract the domain-specific parameters from empirical sample datasets to generate synthetic data.

Second, to assess the accuracy of insights, we need to embed ground truth labels in the dataset. To simulate real-world scenarios, we want to make sure our generated datasets are a mix of signal and noise. There should be some real insight in the data. However, we need to be able to discern if a given insight is true or false. To inform how to construct such datasets, we ran a pilot study with six participants using the same tool and real-world datasets. We analyzed the user-recorded insights in

this pilot study and found that most concerned distribution characteristics and relationships among attributes. Distribution characteristics include "the mean of age distribution is between 20 and 30" whereas the attribute relationships range from "the income and the age are correlated" to "the people of age between 40 to 50 work the most".

To create a synthetic dataset we construct a model based on multivariate normal random variables. For a generated dataset the ground truth of both of these types of insights can be determined from such multivariate normal random variables. Concretely, we randomly group attributes in a dataset as correlated bivariate normal random variables with random correlation coefficient, with means and variances extracted from empirical datasets. For a dataset with n attributes, we embed $n/2$ true relationships where the attributes in the relationship are randomized. The data itself is randomized at two levels: 1) correlation coefficient in any true relationships is randomized between -1 and 1 (excluding 0), 2) we sample from multivariate normal distributions that are parameterized by these correlation coefficients. This process is repeated for each participant in our study.

If two attributes are sampled from independent normal random variables, then any insight involving the relationship between these two attributes is false. For two correlated variables, any simple insight is true. For more complex insights (e.g., statements about a sub-population mean like "the average age of married people is 35 to 40"), the ground truth can be calculated either analytically or computationally. Due to the truncation of the random variables for empirical domains, the analytical computation of ground truths for correlated attributes is complicated [168]. Instead, we generate datasets with 100M records from the same model and extract ground truth labels using hypothesis testing with Bonferroni correction [50]; labels are generated with 95% confidence [1].

7.4.3 Procedure

We recruited 28 participants from a research university in the US. All participants were students (25 undergraduate, 3 graduate), all of whom had some experience with data exploration or analysis tools (e.g., Tableau, Pandas, R) and have taken at least introductory college-level statistics and probability classes. Our experiment included dataset-type (*shopping* and *sleep*) and dataset-size (300 and 1000 records) as between-subject factors. Each participant got to see one pairing of dataset-type and dataset- size. The study design was fully balanced - each unique combination of dataset-type and

[1] The code used in this study to generate synthetic datasets and run empirical tests can be found at `https://github.com/zheguang/macau`.

dataset-size was given to 7 participants. The actual dataset records and correlations were generated uniquely for each participant using different random seeds and according to the method outlined in section 7.4.2. Even if two users saw the same combination of dataset-type and dataset-size they still saw a unique dataset in terms of the individual values of attributes and ground-truths.

Each participant session was split into three parts. The first part consisted of a 15 minute tutorial on how to interact with the system using a separate third dataset-type (*restaurant*). In this tutorial we showed the participants all the relevant features of the tool and answered questions about them.

In the second part, participants read a handout describing the dataset and instructions about the scenario. These instructions mentioned that the given datasets were "a small but representative sample" and that they should find and report "any reliable observations" that could be used to understand the "site's customer population" or "patient's sleeping patterns" or could be used to improve "customer growth" or provide "sleep recommendations". The handout stated that participants should write down textual descriptions (using the note-taking tool of the system) about observations they want to report. After clearing up potential questions about the instructions, participants were given up to 15 minutes to analyze the dataset at their own pace. At any time participants were free to stop if the felt they exhausted the use case. During this second part, we instructed users to think-aloud [56] and we captured screen and audio-recordings as well as eye-tracking data. An experimenter was present throughout all of the session, and users were encouraged to ask technical questions or questions about the definition of dataset attributes.

In the third part, the experimenter and participant re-watched video recordings of the session together (with overlaid eye-tracking data). This was an involved process where the examiner paused playback at every interaction, instructed users to explain their thought process, re-wound if necessary and let participants recount which parts of visualizations they were observing (reinforced by the eye-tracking data) and what visual features they had been looking for. The examiner asked detailed questions about the above points if needed. In a post-session questionnaire, participants ranked their background in statistics, familiarity with statistical hypothesis testing, and experience interpreting visualizations on a 5-point Likert scale.

7.5 Accuracy of User Insights

For our analysis, we considered all insights reported by users through the tool's note-taking feature with a few noted exceptions. We excluded insights that were based on prior knowledge or personal experience, that were not directly observable from the dataset, that were based on reading numbers and in no way made a broader statement applicable to a larger population or that misintepreted the visual display. Examples of excluded insights include: "Users wealthy on average compared to median income in the US", "design: 399 red, 329 green, 195 yellow, the rest blue" or "Between stress level and average age, the people with stress level 5 tend to be the oldest at 40 among females" (the participant thought they were filtering to only females but in fact did not). In total, we excluded six insights from five participants. We were left with an average of 5.536 ± 2.742[2] insights per participant ($n = 28$).

Since the datasets used in this study were generated synthetically, each user-reported insight had a known ground truth, and thus was either true or false. For example, we know whether the insight "age is correlated with purchase amount" is true since the model generating the dataset contains this information. If age and purchase amount are sampled from independent normal random variables, this insight is false. If the two variables are correlated, it is true. For roughly 70% of the insights, ground truth labels were extracted by inspecting variable relationship in the dataset-models. For the remainder (e.g., statements about means like "the average age of people in the Southwest is between 35 and 40"), we used empirical methods as described in section 7.4.2 . For example, if a user made 10 observations during a session, on average, we extracted ground truth labels analytically for 7 of them. For the rest we generated ground truth labels empirically under Bonferroni correction (over only those 3 insights) on larger datasets.

We modeled this experiment as a binary classification problem and borrow standard techniques and metrics from the machine learning community to evaluate the results. For each insight, a user, depending on how it was reported, either made a positive observation ("Age and purchase amount look correlated") or a negative one ("There is no correlation between age and purchase amount"). We summarize the accuracy of a user's insights in a confusion matrix where an insight falls into one of four categories: *True positive* (TP): the user insight is a positive observation and the ground truth agrees, *false positive*, or Type I error, (FP): user insight is positive but the ground truth

[2] Averages appear with the standard deviation as the second number.

says otherwise, *true negative* (TN): user insight is negative and the ground truth agrees and finally *false negative*, or Type II error, (FN): user insight is negative and the ground truth disagrees. The insight "There is no correlation between age and purchase amount," for example, would fall into the FN category if in our dataset-model the age and purchase amount values were sampled from two independent normal random variables.

We report the following averages: $TP = 1.000 \pm 1.217$, $FP = 3.250 \pm 2.287$, $TN = 1.250 \pm 1.404$ and $FN = 0.036 \pm 0.189$. Additionally, we computed the following per-user metrics:

$$\text{Accuracy (ACC)} = (TP + TN)/(TP + TN + FP + FN)$$

$$\text{False discovery rate (FDR)} = FP/(TP + FP)$$

$$\text{False omission rate (FOR)} = FN/(TN + FN)$$

Where ACC measures the overall accuracy (the percentage of times users' insights were correct), FDR the percentage of times users reported an insight as positive ("age and purchase amount is correlated") but it turned out not to be true and FOR the percentage of times users reported an insight as negative ("there is no relation between age and purchase amount") but it turned out not to be true. ACC summarizes overall performance, whereas FDR and FOR give a more detailed view about where mistakes where made. We found that the average ACC across all users is 0.375 ± 0.297, the average FDR is 0.738 ± 0.296 and the average FOR is 0.018 ± 0.094.

Our study featured a full-factorial 2 dataset-types (*shopping* and *sleep*) $\times$ 2 dataset-sizes (300 and 1000 records) study design. We applied an analysis of variance test (ANOVA) with dataset-type and dataset-size as the between-subject factors. We found that dataset-type as well as dataset-size had no significant effect on accuracy ($p = 0.792, \eta^2 = 0.003$ and $p = 0.091, \eta^2 = 0.109$ respectively, $\alpha = 0.05$).

7.6 Confirmatory Statistical Hypothesis Testing

The results of our study show that for our synthetic datasets and the particular tool we used, over 60% of user reported insights were wrong. Results from visual analysis are often considered exploratory. They need to be confirmed in a second phase. In fact, roughly a fourth of our study participants mentioned at one point or another that they would want to verify a reported insight

through statistical testing. In this section, we report on an experiment where we validated user insights through different confirmatory analysis approaches.

The approaches we used included *confirmation; same dataset* and *confirmation; validation dataset*. *Confirmation; same dataset* models an approach where statisical analysis is done on the same dataset used in the exploration phase. *Confirmation; validation dataset* follows Tukey's model [158] of conducting exploratory and confirmatory data analysis on two separate datasets; for this we generated a new dataset of the same size and with the same parameters (i.e., random variables have the same mean and variance and the correlation coefficients between variables are the same) as used during exploration. For both approaches, we use the Benjamini and Hochberg procedure [20] to correct for multiple hypotheses.

7.6.1 From Insights to Statistical Tests

To perform this experiment, we converted user insights into testable statistical hypotheses via multiple steps. We first created an encoding scheme based on insight classes. Insights were coded as instances of these classes. An insight became an object where its type and necessary properties were defined by its class. We then defined null hypotheses and testing procedures for each insight class. To transform insights into testable hypotheses: the insight class indicates the statistical test to use and the properties of the encoding inform that test's input parameters. The following sections explain these steps.

7.6.2 Insight Classes

Our goal was to find a classification model with the fewest classes that could still accurately describe all insights gathered in our study. We employed a process where we first generated candidate classes which we then iteratively refined. In this process we considered all available data from our user study. This includes the video, audio and eye-tracking recordings, the textual description of insights provided users, as well as participant commentary that we gathered when re-watching session videos with the participants.

We arrived at a system that encompasses five insight classes: *shape*, *mean*, *variance*, *correlation* and *ranking*. The *mean* and the *variance* classes described insights containing direct statements about the means or variances of distributions. *Shape* were observations about the shape of one

Insight Class	Null Hypothesis	Permutation π	Test Statistic
Mean	$E[X] = E[Y]$	$X \cup Y$	$\lvert\mu_X - \mu_Y\rvert$
Variance	$var(X) = var(Y)$	$X \cup Y$	$\lvert\sigma_X^2 - \sigma_Y^2\rvert$
Shape	$P(X\lvert Y = y_1) = P(Z\lvert Y = y_2)$	Y	$\lVert P(X\lvert Y = y_1) - P(Z\lvert Y = y_2)\rVert$
Correlation	$X \perp Y$	X	$\lvert\rho(X, Y)\rvert$
Ranking	$X \sim Unif(a, b)$	$\pi \sim Unif(a, b)$	$\begin{cases} 1 & rank(X_\pi) = rank(X_{obs}) \\ 0 & \text{else.} \end{cases}$

Table 7.1: Summary of randomization hypothesis tests to which insights are mapped to for confirmatory analysis. The random variables represent attributes with arbitrary conditions from the dataset.

or more distributions. The *correlation* class covered all insights where a relationship between two variables was established and, finally, the *ranking* class included observations about rankings or orderings among sub-populations. Each class of insight defined several properties that fully described its class instances, such as which attributes were involved, which sub-populations were getting compared, whether parts of the data were filtered out and what comparison were being made (e.g., is something smaller or bigger than something else).

7.6.3 Coding

We describe our 155 insights as instances of their corresponding classes. On average per participant we encoded 1.250 ± 1.404 *correlation*, 2.786 ± 2.347 *mean*, 1.143 ± 1.860 *ranking*, 0.821 ± 1.517 *shape* and 0.107 ± 0.315 *variance* insights. Following Liu et al. [115], the first author did the majority of the coding, revising it with co-authors to reduce bias.

Figure 7.3 illustrates examples of user reported insights from our study. It shows the visual display that triggered an insight alongside the textual description provided by participants and the corresponding insight class, with its properties, that encoded the insight. Note that we again relied heavily on the commentaries made in our post-session video review with the participants, as well as our recorded eye-tracking data. Figure 7.3 (c) depicts an instance where the user's statement alone did not make the mapping to an insight class obvious; however, post-session commentary and eye-tracking data resolved this.

7.6.4 Mapping Insight Classes to Null Hypotheses

We encoded insights as instances of one of our five insight classes. We now want to convert these encodings into testable hypotheses. For this, we define a general null hypothesis pattern for each insight class that we then fill out with specifics from the actual insight. For example, a user insight

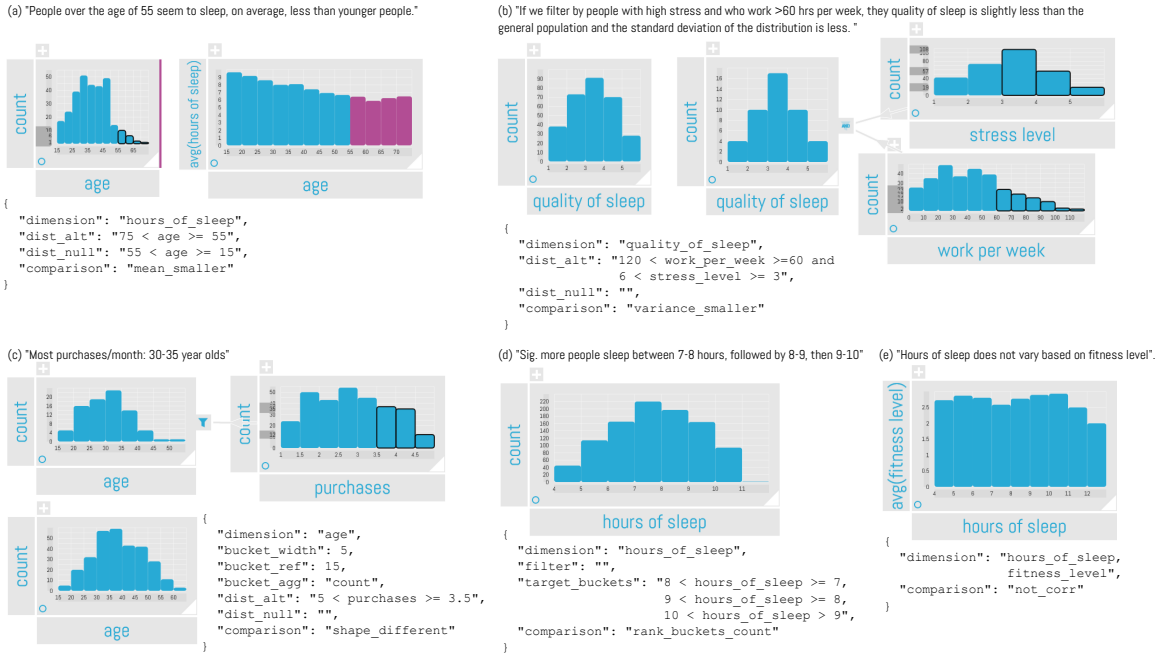

Figure 7.3: Examples of user reported insights from our study. The figure shows the visual display that triggered an insight alongside the textual description participants reported and the corresponding insight class with its properties we encoded it to. (a) An example of a *mean* insight. The user directly mentions that he is making a statement about averages. We encode the dimension that we are comparing across ("hours of sleep"), the two sub-populations that are being compared ("75 < age >= 55" and "55 < age >= 15") as well as the type of comparison ("mean_smaller"). (b) Our user compares the standard deviation of the two "quality of sleep" charts. We encode this as a *variance* insight. We again describe this instance fully by recording the dimension involved, the sub-populations compared and the type of comparison being made. (c) Example of a *shape* class insight. From the user statement alone, it was not obvious to which class this insight corresponded. However, in the post-session video review, the participant mentioned that she was "looking for changes in age distribution for different purchases" and that she observed a change in the shape of the age distribution when filtering down to high purchase numbers. This was reinforced by analyzing the eye-tracking data of the session. The participant selected bars in the "purchases" histogram and then scanned back and forth along the distribution of the filtered "age" visualization and the unfiltered one. (d) An example of an insight where a *ranking* among parts of the visualization was established. (e) The user created a visualization with two attributes. The y-axis was mapped to display the average "fitness level". Our user notes report insights that discuss the relationship of the two attributes. We classified this as a *correlation* insight.

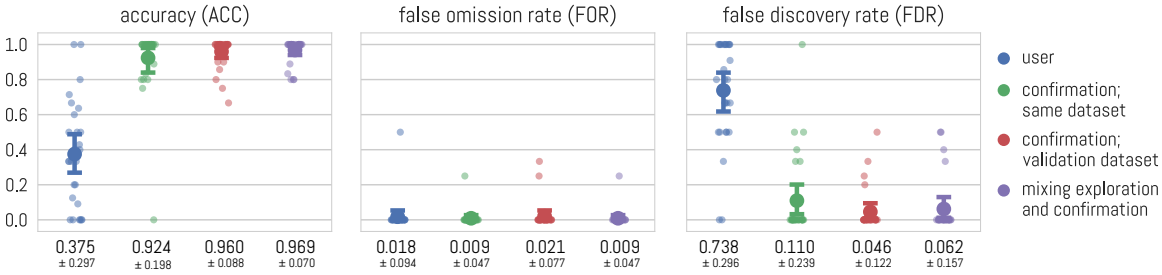

Figure 7.4: Plot with average scores, where the second number is the standard deviation, and rendered 95% confidence intervals for accuracy (ACC), false omission rate (FOR) and false discovery rate (FDR) for users and different confirmatory approaches. Overlaid are all individual datapoints ($n = 28$).

that "the average age is 50" would be categorized as an object of class *mean*. The corresponding null hypothesis pattern for this class is defined as $E[X] = E[Y]$ which we then replace with the properties of the insight. The resulting null hypothesis for this example will be $H_0 : E[age] = 50$. All null hypotheses patterns are defined in Table 7.1.

There are certain ambiguities in these translations from insights to hypothesis. For instance in the above example, the histogram bin of age 50 was perhaps the highest, but also the bin width was 5. So the user was more likely to imply that the population mean was around 50 and not exactly 50. We modified null hypotheses in such cases to account for this level of ambiguity by adding an interval of 10% around the hypothesized mean. Another drawback of this null hypothesis mapping is that we need the null hypothesis to be testable. For example in the insight class of *ranking*, if the user insight is of a certain order of the age groups, then conceptually we should test against all other possible orders. However, this null hypothesis would be very hard to test statistically. Thus, we chose uniformity as the null hypothesis, meaning no particular order in all the age groups. In general, we resolve ambiguities by erring on the side of users by choosing more relaxed null hypotheses where statistical results are more likely to "agree" with user judgment.

For each null hypothesis pattern we define a corresponding Monte Carlo permutation or bootstrap test. We chose resampling for hypothesis testing because it offers several advantages over the parametric testing such as the t-test and the χ^2-test. First, randomization tests do not assume the distributions of the test statistics [48]. Moreover, some parametric tests such as the χ^2-test require samples to be large enough to make accurate inferences [21]. However, many user insights were based on skewed data or highly selective filters, and hence might not always meet the sample size requirement.

In general, the Monte Carlo permutation or bootstrap tests share a common computational form [48]. First a test statistic is determined based on the hypothesis. Then the data under the hypothesis is permuted or bootstrapped to obtain the distribution of the test statistic under the null hypothesis. Finally the proportion of the test statistics in permutations that are more extreme than in the user observation forms the estimated p-value, $\hat{p}$. To determine how many permutations we needed for a sufficiently accurate estimate, we used the Central Limit Theorem to derive the 95% confidence interval,

$$\hat{p} \pm 1.96\sqrt{\hat{p}(1-\hat{p})/n}$$

where n is the number of permutations [21]. With enough permutations, we obtained a non-overlapping confidence interval of $\hat{p}$ against the significance level α, which follows the decision rule of rejecting the null hypothesis if it is no greater than α. We summarize the details of the randomization tests for each insight class in Table 7.1.

7.6.5 Analysis

Using the procedure outlined above, we mapped all insights to hypotheses tests. Depending on the confirmatory approach, we computed hypothesis test results either on the same dataset shown to users (*confirmation; same dataset*) or an a newly generated validation dataset (*confirmation; validation dataset*). We again modeled this experiment as a binary classification problem where statistical significance (using a cutoff p-value of 0.05) provided positive or negative predictions for insights. Those predictions were then evaluated against ground truth labels. The detailed numbers of this experiment are reported in Figure 7.4 including individual datapoints, means and 95% confidence intervals.

7.7 Mixing Exploration and Confirmation

The two confirmatory analysis approaches outlined and compared in the previous section have their drawbacks. For example, while *confirmation; validation dataset* is statistically sound, it requires users to collect additional data which is often unfeasible in practice. Data collection might be expensive, as is the case for user studies, medical trials or crowd-sourcing, or might be done by an outside provider over which the user has no control. Splitting a dataset into exploratory and

confirmatory parts significantly lowers the power of comparison done on either part due to smaller sample sizes. Perhaps more problematic, *confirmation; same dataset* can lead to inherent systematic bias because we are statistically testing insights on the same data that initially informed them.

We want to compare these approaches to one that we call *mixing exploration and confirmation*. So far we have only analyzed insights that users reported. We call these *explicit* insights. However, during an exploration session, users might have made a significant number of along-the-way comparisons that did not trigger insights. The reasons for these could be manifold. These comparisons may have involved uninteresting visualizations, confirmed some assumption that the user already had or just been inconclusive. Regardless, the result of such a comparison is still a type of insight. The insight being that the result of the comparison was not interesting enough to be reported. We call these *implicit* insights. The previous confirmatory approaches ignore *implicit* insights completely. With *mixing exploration and confirmation* we simulate an approach where comparisons that resulted in either type of insight, *explicit* or *implicit*, were incorporated.

7.7.1 Coding

We again considered all collected data from our user study: video, audio and eye-tracking recording and commentary from participants. However this time we focused on *implicit* insights where users made a comparison but did not report it, likely because it was uninteresting. We encoded such instances with the same insight classes as before. For example, a user might view "age" for two different sub-populations; eye-tracking data indicates visual scanning between the two; the post-session commentary reveals the user was comparing the two trends but did not see any difference. Our coding process marks this as an *implicit* insight of type *shape*.

7.7.2 Analysis

Overall we found a total of 620 *implicit* insights, with 22.143 ± 12.183 average *implicit* insights per user. We followed the same procedure as previously to convert these *implicit* insights into statistical tests. We conducted statistical analysis as with the other confirmatory approaches, but added tests based on *implicit* insights to the multiple hypotheses correction procedure. We used the same p-value cutoff as before and report the same metrics.

Note that we did not report the accuracy of *implicit* tests. All metrics were solely based on

explicit insights since we only cared about their correctness. Consider again the example from the introduction. Jean made nine *implicit* and one *explicit* insights but only shared the *explicit* one with his co-workers. Only the accuracy of the explicit one matters since only it will be acted upon; yet its accuracy depends on the set of *implicit* insights. Using the Benjamini and Hochberg procedure [20], we get the results summarized in Figure 7.4.

7.8 Mitigate the Effect of the Multiple Comparison Problem

The results of our experiments show that the performance of *mixing exploration and confirmation* is almost on par with *confirmation; validation dataset* while providing the benefit of not requiring additional data. *Mixing exploration and confirmation* also clearly outperforms *confirmation; same dataset* which produce higher FDR. From a statistical viewpoint this makes sense. By ignoring *implicit* insights, and therefore excluding them from a multiple hypotheses test correction procedures, we automatically introduce systematic bias. Even on completely random data, if we look long and hard enough, we will find something that, by itself, is statistically significant or in our case, looks visually like an interesting or unexpected pattern. But given that we have seen a bunch of uninteresting observations before makes it likely that the insight was due to random noise in the data.

However, requiring users to manually encode all their *explicit* and *implicit* insights into testable hypotheses places a huge burden on them.and is unfeasible in practice. Can we create tools that automatically do this encoding as the users is exploring a dataset? We present and discuss two approaches as first steps to potentially address this problem.

7.8.1 Simple Heuristics

We extended Vizdom (our data analysis system presented in Chapter 6) by three new features:

Explicit and implicit hypothesis formulation: For cases where users know which effect they want to statistically verify, we included support to explicitly create hypotheses through a gestural user interface that poses minimal overhead to users. For *implicit* hypotheses we augmented our system to formulate tests and display test results automatically. The class of tests we formulate automatically includes comparisons of subsets against the global population of a dataset such as the one illustrated in Figure 7.5. The example shows a visualization chain where without an implicit

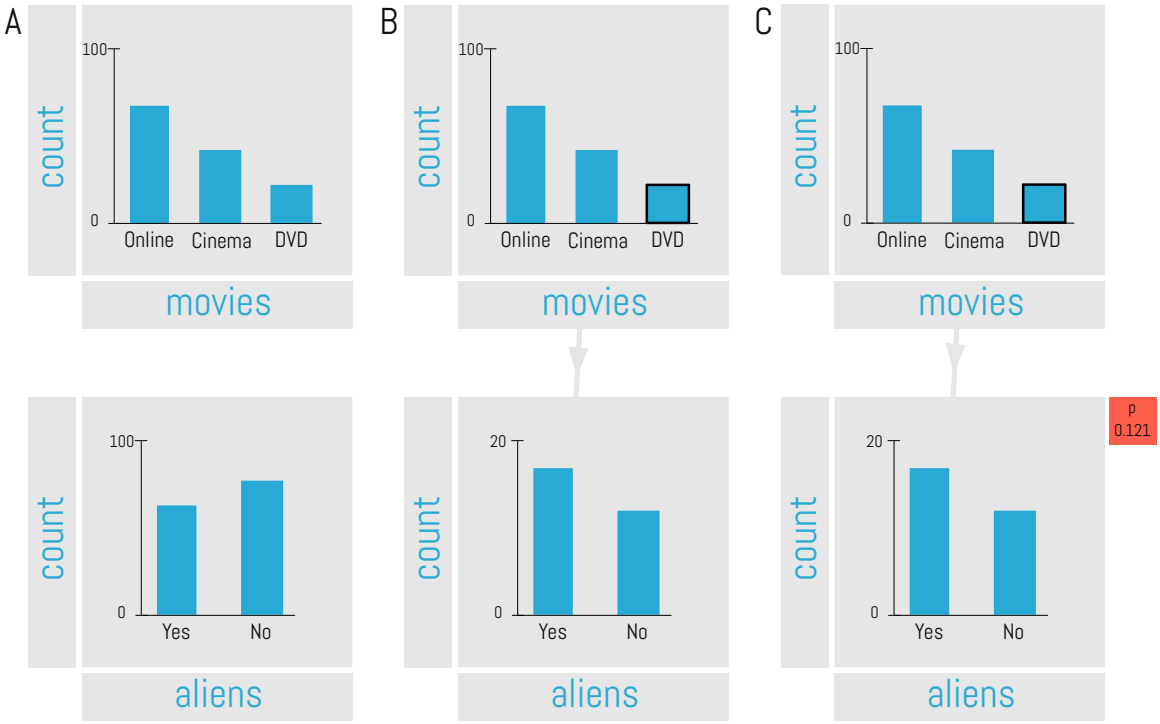

Figure 7.5: Example of a visualization network where users might be led to false discoveries without automatic hypothesis formulation. (A) two separate visualizations showing preferences for watching movies and how many people believe in alien existence; (B) the two visualizations combined where the bottom one shows proportions of belief in alien existence for only people who like to watch movies on DVD, displaying a noticeable difference compared to the overall population. (C) same visualizations as before but now with automatic hypothesis formulation turned on, highlighting that the observed effect is not statistically significant.

hypothesis test (C), users might wrongly observe and perceive a significant effect (difference in bottom visualization between A and B). As shown in the the examples in Figure 7.3 as well as from our own experience of manually extracting and encoding insights this is a challenging task. There are many nuances: Is the user comparing shapes of distributions or means? Or maybe even variances? Does a user compare the means of two sup-populations or the mean of each sub-population against the general population? These automatically generated hypotheses based on simple heuristics could oftentimes be wrong, but we envision that they could be improved through additional user feedback.

Visualization recommendations: To speed up the potentially laborious process of manually exploring a dataset we added a visualization recommendation engine with false discovery control. Similar to SeeDB [163] our *recommender* allows users to search for filter conditions that have a significant effect (positive or negative) on a given reference visualization. We expose this functionality

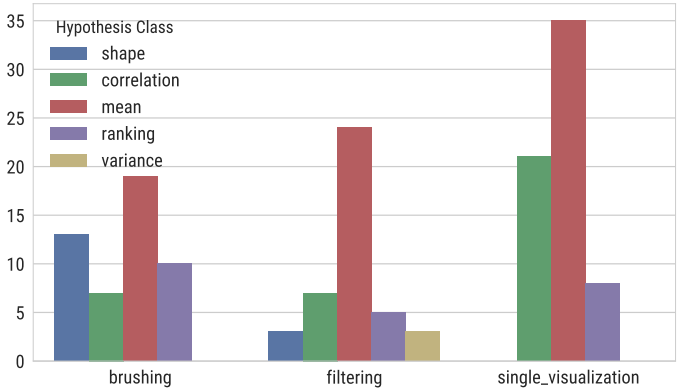

Figure 7.6: Count-chart that illustrates which types of visual displays led to what kind of hypothesis class.

though a gestural touch UI that can be accessed from any visualization.

Hypotheses tracking: A "risk-gauge" on the right-hand side of the display (Figure 7.7 A) serves two purposes, namely, to give user a summary of the multiple hypothesis correction procedure (e.g., in this case α-investing [179] is used with a false discovery rate of 5% and with current remaining budget of 70%), and to provide access to a scrollable list of all the hypotheses that have been explored. Each list entry can be expanded (in the example all are expanded) to display details about an observation and its statistical significance. The text labels describe the null and alternative hypotheses for each observation and the corresponding hypothesis test and p-value . Each color coded tile indicates whether the observation is statistically significant or insignificant, which corresponds to green or red respectively. The distributions of null and alternative hypotheses and the color coded effect size are also visualized (C). To help the user understand the effect of data collection, the sample size estimate for the current significance level is displayed for each hypothesis test assuming the effect size is fixed (B). For example, the five green squares in (B) indicates approximately five times the current data size with the same effect size would make this observation significant. Finally, important insights can be marked by tapping the "star" icons (D).

7.8.2 Automatic Hypothesis Generation

Figure 7.6 shows which kind of visual display participants looked at when reporting an insights and to which hypothesis the insight was mapped. Even within one class of visual display there is large variety in the types of hypotheses users might formulate. However, we hypothesize that an

inference engine that takes all available information into account, such as the visual display, which data attributes are involved, the interaction patterns as well as mouse-events (as a proxy for eye-tracking data [87]) could achieve reasonable accuracy for this task especially when coupled with additional user input for cases where the engine is unsure. New types of multi-modal user interface that encourage users to talk and describe what they are looking at could furthermore help with this inference task. John et. al. [94] for example successfully show how natural language processing can be integrated into a conversational user interface that guides and helps users create machine learning pipelines. In a preliminary experiment we transcribe what participants said when they reported an insight. From our coding we know for each insight to which hypothesis class it got mapped. We use the transcribed text as the feature and the hypothesis class as the label and train a classifier that tries to predict hypothesis classes from text samples. We split our data (n=155) into random training and testing sets and find that the accuracy of a simple Naive Bayes classifier is 0.679 ± 0.103.

A separate question becomes how such an automatic hypothesis inference engine is integrated into the user interface of a tool. We can envision two types of systems, one that surfaces results of automatic hypothesis tests directly as soon as they are inferred and one that accumulates them in the background and only warns the users once a certain "risk" threshold is met or lets them analyze test results at the end of a session. While the latter introduces little distraction from current workflows, the former offers several advantages. Users can correct wrong inferences immediately and can use results of hypotheses tests to steer their exploration. An anecdotal case from our user study demonstrates this advantage. Our participant analyzed a visualization of time it takes someone to fall asleep and quality of sleep and wrongly concluded that less time to sleep has a positive effect on the sleep quality. She then went on and spent the rest of her session searching for factors that might influence the time to sleep such as stress levels and fitness levels which she might otherwise would not have looked at. We are planning to explore the feasibility of building an hypotheses inference engine as well as comparing these design trade-offs in future work.

7.9 Discussion

Real-world datasets are weighted compositions of noise and signal. One goal of visual data analysis is to assist users at efficiently separating the two. We want visualization systems where users can maximize their total number of insights while minimizing false insights. Insight-based methods allow

Figure 7.7: User interface design showing a "risk-gauge" on the right which keeps track of all hypotheses and provides details for each of them.

researchers to compare systems based only on the former. Analyzing errors requires quantification of the correctness of insights. For real-world datasets this is frequently not possible because there is no way to know which effects are true or not.

In this paper we use a method based on synthetic datasets where we can classify the correctness of an insight as either true or false. Our notion of insight is limited to observations, hypotheses and generalizations directly extracted from the data. If a user tells us "age" and "hours of sleep" are correlated we know if that statement is true or not.

For the visualization tool used in our study, we found over 60% of user reported insights were incorrect. However, this error rate needs to be interpreted anecdotally. Results may vary greatly between users, visualizations, tools and datasets. The high error rate is perhaps unsurprising. Mathematical procedures might be better suited to make such data-driven inferences than humans. More surprisingly, when following up user generated insights with statistical tests on the same dataset, we are still left with 11% false discoveries (*confirmation; same dataset*, Figure 7.4). Double of what statistics promise when using a significance level of 5%. We introduced systemic bias by testing hypotheses on the same dataset that informed them and we inflated our FDR due to the multiple comparisons problem.

The key takeaway here is that without either confirming user insights on a validation dataset (*confirmation; validation dataset*) or accounting for all comparisons made by users during exploration (*mixing exploration and confirmation*) we have no guarantees on the bounds of the expected number of false discoveries. This is true regardless which specific visual analysis tool or visualization is used. Taking action, making decisions or publishing findings this way becomes risky.

Validating user generated insights with the *confirmation; same dataset* approach is not statistically sound and *confirmation; validation dataset* requires additional data, which in practice is often hard to acquire. In our experiments we manually coded *explicit* and *implicit* insights to show the benefits of *mixing exploration and confirmation*: it guarantees the same FDR bounds as confirmation on a validation dataset. However, burdening users to remember and code all of their insights during an exploration session is unfeasible. Could we create tools that automatically do this encoding while users explore a dataset? We believe that augmenting visual analysis systems to do this is a promising direction for future research.

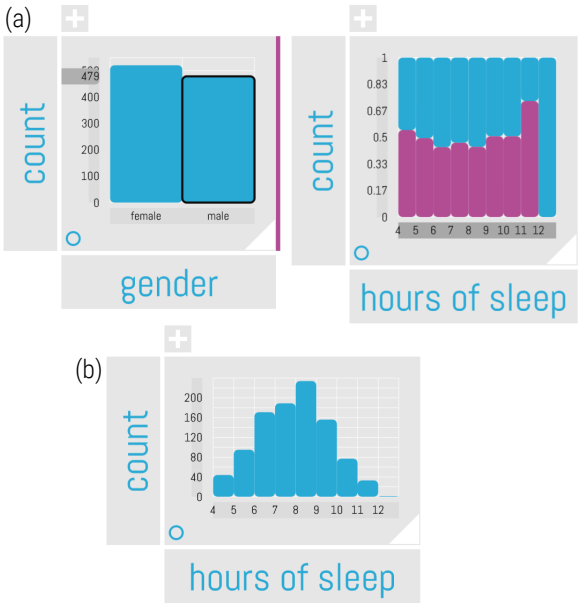

Figure 7.8: Example visual display that lead to a reported insight by a participant. By comparing the normalized shape of the hours of sleep distribution for males and females, the user correctly observed that "A thin majority of females get more sleep than males" (a). The participant double-check her finding by making sure she accounted for small sample sizes (b). All statistical confirmatory procedures (falsely) failed to confirm this insight.

7.9.1 User Performance

In addition to our main analysis, we examine several questions regarding user insights and performance. Is user accuracy correlated to self-reported expertise levels in statistical hypothesis testing, interpreting visualization and general statistical background? Are there cases where users intuition wins over statistical measures? Does the support size of individual insights influence accuracy? Is there inherent study design bias that makes users feel pressured towards the end of a session?

We correlated participant accuracy scores with their self-reported expertise levels. We found that neither background in statistics ($r = -0.258, p = 0.223$), their familiarity with statistical hypothesis testing ($r = -0.328, p = 0.117$) or their experience with interpreting visualizations ($r = 0.005, p = 0.982$) had a significant correlation with accuracy percentages. In general we focused on novice participants since we believe that they are a large part of the target audience for visualization tools. More experiments are needed to assess if our empirical results generalize to different user populations (e.g., users with statistical expertise).

Over all of the 155 user reported insights we only find one instance where a user outperforms all

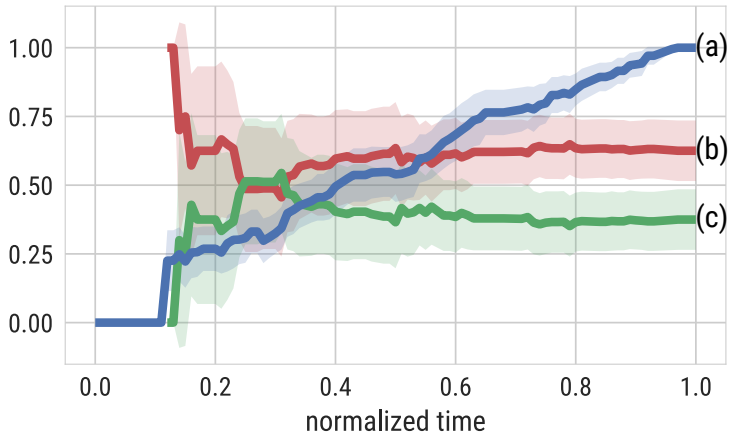

Figure 7.9: Graph showing cumulative, normalized number of reported insights (a), 1 - ACC (b) and ACC (c) averaged across all users over normalized time axis. The data is sliced and aggregated over 100 bins (x-axis) and interpolated between each bin. Bands show 95% confidence intervals. Note the missing start lines for (a) and (b) are because users did not report any insights for the first few minutes of a session.

statistical methods. The example is illustrated in Figure 7.8. This perhaps anecdotally demonstrates that for these targeted and narrow insights, that are directly observable on the data, humans are outperformed by statistical methods and this further supports a tight integration of statistics and visualizations.

The average normalized support (i.e., the percentage of data-records involved in an insight) per user was 0.590 ± 0.270 for correct insights and 0.463 ± 0.282 for incorrect ones. While the difference is not statistically significant ($p = 0.118, d = 0.471$) examining this trend in more detail is warranted.

We extracted timestamps of insights and normalized them by session length. We found that the average normalized time for incorrect and correct insights is 0.585 ± 0.271 and 0.587 ± 0.248 which is not a statistically significant difference ($p = 0.957, d = 0.009$). Figure 7.9 visualizes the progression of user accuracies and its inverse over a normalized time-scale. The accuracy is stable after the halftime mark, hinting that users did not report an increased number of false insights towards the end of a session.

7.9.2 Relating to Statistical Theory

Without any hypothesis testing, the false discovery rate averages over 73% (*user*, Figure 7.4). With hypothesis testing on the same dataset (*confirmatory; same dataset*) the false discovery rate reduced but still scored around 11% (Figure 7.4). With multiple hypothesis control only on the *explicit* insights, the average FDR inflated above the theoretical bound of 5%. This is because by not controlling for *implicit* tests, we are exposed to the multiple hypothesis error as described in section 7.3.3. Essentially this is a misuse of the control procedure.

With proper multiple hypothesis correction on both *implicit* and *explicit* hypotheses *mixing exploration and confirmation* achieved average false discovery rates around 5% (Figure 7.4). This can be seen from the theoretical perspective where the Benjamini and Hochberg procedure [20] guarantees the expected proportion of false discoveries V among all discoveries R is upper bounded by a given significance level $\alpha = 0.05$:

$$E[\|V\| \,/\, \|R\|] \leq 0.05$$

We achieved a similar false discovery rate with *confirmation; validation dataset* which tested hypotheses on a separate dataset. This is akin to replication studies in science.

Statistical procedures only provide bounds on the expected number of false discoveries. Tightening these bounds will automatically result in higher false omission rates. Balancing this trade-off is highly domain specific. In drug trials, false discoveries must be avoided, whereas, in security related scenarios, false omissions can have disastrous effects. Sound statistical methods, like *confirmatory; validation dataset* and *mixing exploration and confirmation*, facilitate these trade-offs.

7.9.3 Larger Datasets

Theoretically, having complete knowledge of the population distribution, or infinite resource to sample from it to appeal to the Law of Large Numbers, could eliminate the risk of false discovery. However, we believe that in many practical cases it is difficult to approximate this theory. Fundamentally, the uncertainty of the statistical inference on the data is affected by many factors, including sample sizes, variations, effect sizes, and measurement errors [70]. Thus it requires significant upfront effort to determine the sample size with enough statistical power by controlling the other factors. Furthermore, visual analysis may select and compare many different subsets of the

data; the space of possible questions also varies. These aspects also complicate the notion of having a single sufficiently large data size for all possible analysis. Nonetheless, we plan to run similar studies that will explore the relationship between user accuracy and dataset-size in more detail.

7.9.4 Base Rate Fallacy and Other Errors

The psychological tendency to neglect the global base rate while overestimating by the local, more specific likelihood has been well studied in Pscyhology and Behavioral and Brain Sciences [15, 102]. In the context of visual analysis, it would be interesting to see how user performance changes if the users were given some information about the noisiness of the underlying dataset. Interestingly, some statistical testing procedures can automatically approximate the data randomness and improve performance based on it [179]. Our study however excludes this variable by fixing the base rate of ground truths and not disclosing it to the participants.

Beyond Type I and Type II errors, other error types have been proposed to quantify the likelihood of mistaking the sign (Type S) or overestimating the magnitude (Type M) [71, 69]. However in our study user observations were often vague regarding these effects. For example, instead of saying "mean A is 45 and less than mean B" participants would typically say "mean A is different than mean B". Furthermore, sign and magnitude do not apply to several types of user insights (e.g., when comparing shapes or rankings). For the subset of insights where users stated directions, we setup one-sided null-hypotheses which captured sign errors in our FDR calculation. Based on our experience, a more detailed study of Type S and Type M errors would likely involve a new study design that invites the users to be more conscious about making observations on signs and magnitudes.

7.10 Conclusion

Comparing a visualization to a mental image is akin to performing a statistical test, thus repeated interpretation of visualizations is susceptible to the multiple comparisons problem. In this work we attempted to empirically characterize this. We presented an experiment based on synthetically generated datasets that enabled us to assess the correctness of user reported insigths. We showed that by not accounting for all visual comparisons made during visual data exploration, false discovery rates will be inflated even after validating user insights with further statistical testing. We demonstrated that a confirmatory approach that addresses this can provide similar statistical guarentees to one

that uses a validation dataset.

Chapter 8

Discussion & Conclusion

In the previous chapters we discussed work towards making data analysis accessible. It is increasingly common practise across many different domains to ground decisions with quantitative insights from data. While skilled programmers have access to tools and APIs such as SQL, R, Python, Spark etc., users without a computational background are at a disadvantage. They can not capitalize on the power of data. In this dissertation we present visual interfaces that expose a range of common data analysis tasks to non-programmers, we show how these visual tools can be made scalable to large datasets and we discuss pitfalls that arise when empowering novice users. In this last chapter we reiterate the main contributions of this dissertation and discuss open problems and future research.

8.1 Visual Languages for Data Analysis

Contributions In chapters 2, 3, 4 and 6 we presented a set of user experiences that expose various data analysis task on different types of data in visual ways suited for non-programmers. These interfaces allow for incremental and piecewise specification of complex workflows where intermediate results serve as feedback as well as interactive handles to adjust parameters. The user interface designs follow the *fluid* [54] paradigm which promotes immersive interaction where users can focus on the current activity while not being distracted by interface.

Open Problems and Research Directions There are a lot of potential ways of improving this line of work. The most immediate next steps are to extend this style of visual interfaces to other

types of data, such as time-series, graph, image, and text data, and to include more data analysis tasks that the current work ignores including data-cleaning, exporting or sharing of worfklows, collaboration and storytelling. Especially for machine learning workflows, there is a huge opportunity to use visualizations, user interface techniques and visual languages to make these methods more understandable to users. We also believe that more quantitative user studies are needed to better understand the benefits, learnability and limitations of our current work.

8.2 Progressive Visualizations

Contributions In chapter 5 we studied how progressive visualizations affect users in exploratory settings in an experiment where we capture user behavior and knowledge discovery through interaction logs and think-aloud protocols. We reported that progressive visualizations outperform blocking visualizations in almost all metrics and that progressive visualizations do not significantly differ from instantaneous visualizations. This finding suggest that progressive visualizations, and potentially progressive computation, might be a viable solution to achieve scalability in data analysis systems.

Open Problems and Research Directions The study we presented in chapter 5 is just a first step towards understanding progressive visualizations. We studied only simple forms of uncertainty visualizations and we also excluded other factors from the study, such as if users are able to fully grasp the meaning of approximate answers and how distracting the visualization updates are. We plan to carry out several follow-up studies where we compare different update strategies and different types of uncertainty representations. Furthermore, we intend to get a better understanding of how a user's behavior changes between instantaneous and progressive visualizations through in-depth sequential interaction log analysis, potentially coupled with eye-tracking data. Gaining such understanding would help optimize visual representations and interactions with progressive visualizations. Furthermore, progressive computation has some limitations. Incremental and approximate answers can be hard to interpret. As shown in prior work [45, 43, 44], even experts make mistakes when interpreting standard statistical accuracy metrics. Advances in uncertainty visualizations are needed to fully mitigate some of these issues. Additionally, not all algorithms and analysis tasks lend themselves well to progressive computations. Some methods are highly susceptible to outliers, or are not guaranteed to converge to the true result when implemented progressively. Techniques

to mitigate these problems (some our outlined in our own related work [68]) or guard against false insights that arise because of these limitations would greatly benefit the presented work.

8.3 Accuracy in Visual Data Analysis

Contributions Having novice users directly analyze data also comes with drawbacks. Once such drawback, the multiple comparison problem, is well-known in statistics but has for the most part been ignored in the visualization community. In chapter 7 we argue that comparing visualizations to a mental image is similar to statistical testing. Repeated interpretations of visualizations are susceptible to the same multiple comparisons problem as arise in statistics. The contribution of our work is twofold. First, we contribute a method to quantify the rate of false discoveries and false that occurred in common visual analysis tasks. Our method comprises of two parts: (1) a procedure to generate synthetic data with associated ground-truth labels, and (2) a procedure to transform participants' insights into testable hypotheses. And second, we empirically show that visual analysis can lead to a high false discovery rate.

Open Problems and Research Directions Both of these contributions can be further extended. For example, our method only allows to examine a limited range of true insights in the generated data. While there is some room to vary the amount of true relationships, extending our synthetic datasets generator to support a wider range of base rates would be helpful for future experiments. Perhaps similarly, we fixed a number of parameters in our experiment which makes it hard to generalize our quantitative findings. Running similar experiments on different tools, varying dataset sizes and with a different user population would further our understanding of the impact of the multiple comparison problem in visual analysis further and could help us designing tools that shield users against it.

Designing such tools that provide guarantees on the accuracy of insights gained through visual analysis is a very interesting and challenging problem. While the approach of capturing all implicit insights during a user session works in theory, it can be a hard problem in practice. Depending on the type of visualization there can be almost an infinite amount of insights a user can extract. Accurately keeping track of all of those possible implicit comparisons without asking the user might be intractable. Especially since some of those comparisons might be done subconsciously. Perhaps

other techniques that either educate users about this potential pitfall or that are based on Bayesian statistics where users would need to state priors before seeing data might be more promising.

Bibliography

[1] Microsoft Power BI. `https://powerbi.microsoft.com/en-us/`. Accessed: 2017-03-23. (Cited on 68)

[2] Tableau. `http://www.tableau.com/`. Accessed: 2015-06-03. (Cited on 68)

[3] Tiboco Spotfire. `http://spotfire.tibco.com/`. Accessed: 2017-03-23. (Cited on 68)

[4] Azza Abouzied, Joseph Hellerstein, and Avi Silberschatz. Dataplay: interactive tweaking and example-driven correction of graphical database queries. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 207–218. ACM, 2012. (Cited on 22, 23)

[5] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Eighth Eurosys Conference 2013, EuroSys '13, Prague, Czech Republic, April 14-17, 2013*, pages 29–42, 2013. (Cited on 69)

[6] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *ICDE '95*, pages 3–14, Washington, DC, USA, 1995. IEEE, IEEE Computer Society. (Cited on 47)

[7] Georgia Albuquerque, Thomas Lowe, and Marcus Magnor. Synthetic generation of high-dimensional datasets. *IEEE transactions on visualization and computer graphics*, 17(12):2317–2324, 2011. (Cited on 111)

[8] James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. (Cited on 46)

[9] Basak Alper, Nathalie Henry Riche, Fanny Chevalier, Jeremy Boy, and Metin Sezgin. Visualization literacy at elementary school. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5485–5497. ACM, 2017. (Cited on 1)

[10] Robert Amar, James Eagan, and John Stasko. Low-level components of analytic activity in information visualization. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 111–117. IEEE, 2005. (Cited on 17, 19, 67, 86)

[11] Saleema Amershi, Max Chickering, Steven M Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 337–346. ACM, 2015. (Cited on 96)

[12] Emmanuelle Anthoine, Leïla Moret, Antoine Regnault, Véronique Sébille, and Jean-Benoit Hardouin. Sample size used to validate a scale: a review of publications on newly-developed patient reported outcomes measures. *Health and quality of life outcomes*, 12(1):2, 2014. (Cited on 111)

[13] Peter Ayton and Ilan Fischer. The hot hand fallacy and the gambler's fallacy: Two faces of subjective randomness? *Memory & cognition*, 32(8):1369–1378, 2004. (Cited on 107)

[14] Ragnar Bade, Stefan Schlechtweg, and Silvia Miksch. Connecting time-oriented data and information to a coherent interactive visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 105–112. ACM, 2004. (Cited on 45)

[15] Maya Bar-Hillel. The base-rate fallacy in probability judgments. *Acta Psychologica*, 44(3):211–233, 1980. (Cited on 131)

[16] Leilani Battle, Remco Chang, and Michael Stonebraker. Dynamic prefetching of data tiles for interactive visualization. 2015. (Cited on 69)

[17] Benjamin B Bederson and James D Hollan. Pad++: a zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 17–26. ACM, 1994. (Cited on 24)

[18] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. The effects of loss and latency on user performance in unreal tournament 2003®.

In *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pages 144–151. ACM, 2004. (Cited on 70)

[19] Yoav Benjamini. Simultaneous and selective inference: current successes and future challenges. *Biometrical Journal*, 52(6):708–721, 2010. (Cited on 102)

[20] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300, 1995. (Cited on 104, 108, 116, 122, 130)

[21] Dimitri P Bertsekas and John N Tsitsiklis. *Introduction to probability*, volume 1. Athena Scientific Belmont, MA, 2002. (Cited on 119, 120)

[22] Peter A. Boncz, Marcin Zukowski, and Niels Nes. Monetdb/x100: Hyper-pipelining query execution. In *CIDR*, pages 225–237, 2005. (Cited on 99)

[23] Danah Boyd and Kate Crawford. Six provocations for big data. In *A decade in internet time: Symposium on the dynamics of the internet and society*, volume 21. Oxford Internet Institute Oxford, 2011. (Cited on 2)

[24] Andrew Bragdon, Robert Zeleznik, Steven P Reiss, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J LaViola Jr. Code bubbles: a working set-based interface for code understanding and maintenance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2503–2512. ACM, 2010. (Cited on 19, 24)

[25] Andrew Bragdon, Robert Zeleznik, Brian Williamson, Timothy Miller, and Joseph J LaViola Jr. Gesturebar: improving the approachability of gesture-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2269–2278. ACM, 2009. (Cited on 11, 38)

[26] Sabrina Bresciani and Martin J Eppler. The risks of visualization. *Identität und Vielfalt der Kommunikations-wissenschaft*, pages 165–178, 2009. (Cited on 106)

[27] Jeffrey Browne, Bongshin Lee, Sheelagh Carpendale, Nathalie Riche, and Timothy Sherwood. Data analysis on interactive whiteboards through sketch-based interaction. In *Proceedings*

of the ACM International Conference on Interactive Tabletops and Surfaces, pages 154–157. ACM, 2011. (Cited on 9, 20, 21, 28)

[28] Jake Brutlag. Speed matters for google web search. *Google. June*, 2009. (Cited on 70)

[29] Andreas Buja, Dianne Cook, Heike Hofmann, Michael Lawrence, Eun-Kyung Lee, Deborah F Swayne, and Hadley Wickham. Statistical inference for exploratory data analysis and model diagnostics. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 367(1906):4361–4383, 2009. (Cited on 102, 107)

[30] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999. (Cited on 106)

[31] Stuart K Card, George G Robertson, and Jock D Mackinlay. The information visualizer, an information workspace. In *Proceedings of the SIGCHI Conference on Human factors in computing systems*, pages 181–186. ACM, 1991. (Cited on 65, 70)

[32] Jacob Carlson, Michael Fosmire, CC Miller, and Megan Sapp Nelson. Determining data information literacy needs: A study of students and research faculty. *portal: Libraries and the Academy*, 11(2):629–657, 2011. (Cited on 1)

[33] Remco Chang, Caroline Ziemkiewicz, Tera Marie Green, and William Ribarsky. Defining insight for visual analytics. *IEEE Computer Graphics and Applications*, 29(2):14–17, 2009. (Cited on 106)

[34] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997. (Cited on 68)

[35] Luca Chittaro and Carlo Combi. Visualizing queries on databases of temporal histories: new metaphors and their evaluation. *Data & Knowledge Engineering*, 44(2):239–264, 2003. (Cited on 46)

[36] William S Cleveland and Robert McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554, 1984. (Cited on 107)

[37] Christopher M Collins and Sheelagh Carpendale. Vislink: Revealing relationships amongst visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1192–1199, 2007. (Cited on 22)

[38] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M Hellerstein, Khaled Elmeleegy, and Russell Sears. Mapreduce online. In *NSDI*, volume 10, page 20, 2010. (Cited on 70)

[39] Michael Correll and Michael Gleicher. Error bars considered harmful: Exploring alternate encodings for mean and error. *IEEE transactions on visualization and computer graphics*, 20(12):2142–2151, 2014. (Cited on 74)

[40] Michael Correll and Jeffrey Heer. Regression by eye: Estimating trends in bivariate visualizations. In *ACM Human Factors in Computing Systems (CHI)*, 2017. (Cited on 107)

[41] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. Vizdom: interactive analytics through pen and touch. *Proceedings of the VLDB Endowment*, 8(12):2024–2027, 2015. (Cited on 68, 70, 86, 87, 109)

[42] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. The case for interactive data exploration accelerators (ideas). In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, page 11. ACM, 2016. (Cited on 99)

[43] Geoff Cumming. Inference by eye: reading the overlap of independent confidence intervals. *Statistics in medicine*, 28(2):205–220, 2009. (Cited on 70, 84, 99, 134)

[44] Geoff Cumming. *Understanding the new statistics: Effect sizes, confidence intervals, and meta-analysis*. Routledge, 2013. (Cited on 70, 84, 99, 134)

[45] Geoff Cumming and Sue Finch. Inference by eye: confidence intervals and how to read pictures of data. *American Psychologist*, 60(2):170, 2005. (Cited on 99, 134)

[46] Alin Dobra, Chris Jermaine, Florin Rusu, and Fei Xu. Turbo-charging estimate convergence in dbo. *Proceedings of the VLDB Endowment*, 2(1):419–430, 2009. (Cited on 74)

[47] Punit R Doshi, Elke Rundensteiner, Matthew O Ward, et al. Prefetching for visual data exploration. In *Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings. Eighth International Conference on*, pages 195–202. IEEE, 2003. (Cited on 69)

[48] Bernd Droge. Phillip good: Permutation, parametric, and bootstrap tests of hypotheses, 2006. (Cited on 119, 120)

[49] Steven M. Drucker, Danyel Fisher, Ramik Sadana, Jessica Herron, and m.c. schraefel. Touchviz: A case study comparing two interfaces for data analytics on tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2301–2310. ACM, 2013. (Cited on 9, 20, 21, 47)

[50] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961. (Cited on 105, 108, 112)

[51] Cody Dunne, Nathalie Henry Riche, Bongshin Lee, Ronald Metoyer, and George Robertson. Graphtrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1663–1672. ACM, 2012. (Cited on 19, 21, 23, 24, 68)

[52] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *STOC*, pages 117–126. ACM, 2015. (Cited on 109)

[53] Philipp Eichmann, Emanuel Zgraggen, Zheguang Zhao, Carsten Binnig, and Tim Kraska. Towards a benchmark for interactive data exploration. *IEEE Data Eng. Bull.*, 39(4):50–61, 2016. (Cited on xvii, 98, 99)

[54] Niklas Elmqvist, Andrew Vande Moere, Hans-Christian Jetter, Daniel Cernea, Harald Reiterer, and TJ Jankun-Kelly. Fluid interaction for information visualization. *Information Visualization*, page 1473871611413180, 2011. (Cited on 2, 9, 20, 21, 27, 47, 133)

[55] Niklas Elmqvist, John Stasko, and Philippas Tsigas. Datameadow: a visual canvas for analysis of large-scale multivariate data. *Information visualization*, 7(1):18–33, 2008. (Cited on 22)

[56] Karl Anders Ericsson and Herbert Alexander Simon. *Protocol analysis*. MIT press Cambridge, MA, 1993. (Cited on 76, 113)

[57] Jerry Alan Fails, Amy Karlson, Layla Shahamat, and Ben Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *Visual*

Analytics Science And Technology, 2006 IEEE Symposium On, pages 167–174. IEEE, 2006. (Cited on 46)

[58] Jean-Daniel Fekete. ProgressiVis: a Toolkit for Steerable Progressive Analytics and Visualization. In *1st Workshop on Data Systems for Interactive Analysis*, page 5, Chicago, United States, October 2015. (Cited on 70, 87)

[59] Jean-Daniel Fekete and Catherine Plaisant. Interactive information visualization of a million items. In *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pages 117–124. IEEE, 2002. (Cited on 86)

[60] Jean-Daniel Fekete and Romain Primet. Progressive analytics: A computation paradigm for exploratory data analysis. *CoRR*, abs/1607.05162, 2016. (Cited on 66)

[61] Nivan Ferreira, Danyel Fisher, and Arnd Christian Konig. Sample-oriented task-driven visualizations: allowing users to make better, more confident decisions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 571–580. ACM, 2014. (Cited on 70)

[62] Danyel Fisher. Incremental, approximate database queries and uncertainty for exploratory visualization. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 73–80. IEEE, 2011. (Cited on 66, 70, 87)

[63] Danyel Fisher, Rob DeLine, Mary Czerwinski, and Steven Drucker. Interactions with big data analytics. *interactions*, 19(3):50–59, 2012. (Cited on 2, 67, 86)

[64] Danyel Fisher, Steven M Drucker, and A Christian König. Exploratory visualization involving incremental, approximate database queries and uncertainty. *IEEE computer graphics and applications*, (4):55–62, 2012. (Cited on 66, 70, 87)

[65] Danyel Fisher, Igor Popov, Steven Drucker, et al. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1673–1682. ACM, 2012. (Cited on 66, 70, 87)

[66] Dean P Foster and Robert A Stine. α-investing: a procedure for sequential control of expected false discoveries. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(2):429–444, 2008. (Cited on 108)

[67] Adam Fourney, Richard Mann, and Michael Terry. Characterizing the usability of interactive applications through query log analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1817–1826. ACM, 2011. (Cited on 43)

[68] Alex Galakatos, Andrew Crotty, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. Revisiting reuse for approximate query processing. *Proceedings of the VLDB Endowment*, 10(10):1142–1153, 2017. (Cited on 99, 135)

[69] Andrew Gelman and John Carlin. Beyond power calculations: Assessing type s (sign) and type m (magnitude) errors. *Perspectives on Psychological Science*, 9(6):641–651, 2014. (Cited on 131)

[70] Andrew Gelman and Eric Loken. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking"and the research hypothesis was posited ahead of time. *Department of Statistics, Columbia University*, 2013. (Cited on 130)

[71] Andrew Gelman and Francis Tuerlinckx. Type s error rates for classical and bayesian single and multiple comparison procedures. *Computational Statistics*, 15(3):373–390, 2000. (Cited on 131)

[72] Steven R Gomez, Hua Guo, Caroline Ziemkiewicz, and David H Laidlaw. An insight-and task-based methodology for evaluating spatiotemporal visual analytics. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on*, pages 63–72. IEEE, 2014. (Cited on 77, 106)

[73] Connor C Gramazio, Karen B Schloss, and David H Laidlaw. The relation between visualization size, grouping, and user performance. *IEEE transactions on visualization and computer graphics*, 20(12):1953–1962, 2014. (Cited on 107)

[74] Lars Grammel, Melanie Tory, and Margaret Anne Storey. How information visualization novices construct visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):943–952, 2010. (Cited on 23, 33, 40)

[75] Thomas L Griffiths and Joshua B Tenenbaum. Randomness and coincidences: Reconciling intuition and probability theory. In *Proceedings of the 23rd annual conference of the cognitive science society*, pages 370–375. University of Edinburgh Edinburgh, 2001. (Cited on 107)

[76] Hua Guo, Steven Gomez, Caroline Ziemkiewicz, and David Laidlaw. A case study using visualization interaction logs and insight. 2016. (Cited on 71, 77, 78, 106)

[77] Peter J Haas and Joseph M Hellerstein. Ripple joins for online aggregation. In *ACM SIGMOD Record*, volume 28, pages 287–298. ACM, 1999. (Cited on 70)

[78] Pat Hanrahan. Analytic database technologies for a new kind of user: the data enthusiast. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 577–578. ACM, 2012. (Cited on 2, 70, 105)

[79] Beverly L. Harrison, Russell Owen, and Ronald M. Baecker. Timelines: An Interactive System for the Collection and Visualization of Temporal Data. In *Graphics Interface*, 1994. (Cited on 45)

[80] Chris Harrison, Anind K Dey, and Scott E Hudson. Evaluation of progressive image loading schemes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1549–1552. ACM, 2010. (Cited on 70)

[81] Megan L Head, Luke Holman, Rob Lanfear, Andrew T Kahn, and Michael D Jennions. The extent and consequences of p-hacking in science. *PLoS Biol*, 13(3):e1002106, 2015. (Cited on 105)

[82] Jeffrey Heer, Jock Mackinlay, Chris Stolte, and Maneesh Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE transactions on visualization and computer graphics*, 14(6), 2008. (Cited on 32)

[83] Jeffrey Heer and Ben Shneiderman. Interactive dynamics for visual analysis. *Queue*, 10(2):30, 2012. (Cited on xii, 17, 19, 23, 26, 67, 70, 86, 89)

[84] Joseph M Hellerstein, Ron Avnur, Andy Chou, Christian Hidber, Chris Olston, Vijayshankar Raman, Tali Roth, and Peter J Haas. Interactive data analysis: The control project. *Computer*, 32(8):51–59, 1999. (Cited on 70)

[85] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online aggregation. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA.*, pages 171–182, 1997. (Cited on 98)

[86] P. Hind. *Encyclopedia Titanica*. (Cited on 38)

[87] Jeff Huang, Ryen White, and Georg Buscher. User see, user point: gaze and cursor alignment in web search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1341–1350. ACM, 2012. (Cited on 125)

[88] Sandra L Hubscher and August Strindberg. Apophenia: Definition and analysis. *Digital Bits Skeptic*, 2007. (Cited on 107)

[89] Stratos Idreos and Erietta Liarou. dbtouch: Analytics at your fingertips. In *CIDR*, 2013. (Cited on 22)

[90] UC Irvine. Uc irvine machine learning repository, 2017. `http://archive.ics.uci.edu/ml/`. (Cited on 72, 111)

[91] Christian S Jensen and Richard T Snodgrass. Temporal data management. *Knowledge and Data Engineering, IEEE Transactions on*, 11(1):36–44, 1999. (Cited on 46)

[92] Jing Jin and Pedro Szekely. Querymarvel: a visual query language for temporal patterns using comic strips. In *Visual Languages and Human-Centric Computing, 2009. VL/HCC 2009. IEEE Symposium on*, pages 207–214. IEEE, 2009. (Cited on 46)

[93] Jing Jin and Pedro Szekely. Interactive querying of temporal data using a comic strip metaphor. In *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*, pages 163–170. IEEE, 2010. (Cited on 46)

[94] Rogers Jeffrey Leo John, Navneet Potti, and Jignesh M Patel. Ava: From data to insights through conversation. CIDR, 2017. (Cited on 125)

[95] Alexander Kalinin, Ugur Cetintemel, and Stan Zdonik. Interactive data exploration using semantic windows. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 505–516. ACM, 2014. (Cited on 69)

[96] Narendra Kamat, Prasanth Jayachandran, Karthik Tunga, et al. Distributed and interactive cube exploration. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 472–483. IEEE, 2014. (Cited on 69)

[97] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372. ACM, 2011. (Cited on 19)

[98] Sean Kandel, Andreas Paepcke, Joseph M Hellerstein, and Jeffrey Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, 2012. (Cited on 2)

[99] Youn-ah Kang and John Stasko. Examining the use of a visual analytics system for sensemaking tasks: Case studies with domain experts. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2869–2878, 2012. (Cited on 105)

[100] Gerald M Karam. Visualization using timelines. In *Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis*, pages 125–137. ACM, 1994. (Cited on 45)

[101] Stephen Cole Kleene. Representation of events in nerve nets and finite automata. Technical report, DTIC Document, 1951. (Cited on 53)

[102] Jonathan J Koehler. The base rate fallacy reconsidered: Descriptive, normative, and methodological challenges. *Behavioral and brain sciences*, 19(1):1–17, 1996. (Cited on 131)

[103] Robert Ladouceur, Claude Paquet, and Dominique Dubé. Erroneous perceptions in generating sequences of random events. *Journal of Applied Social Psychology*, 26(24):2157–2166, 1996. (Cited on 107)

[104] H. Lam, D. Russell, D. Tang, and T. Munzner. Session viewer: Visual exploratory analysis of web session logs. In *Visual Analytics Science and Technology, 2007. VAST 2007. IEEE Symposium on*, pages 147–154, Oct 2007. (Cited on 46)

[105] Rongjian Lan, Hanseung Lee, A Fong, M Monroe, Catherine Plaisant, and Ben Shneiderman. Temporal search and replace: An interactive tool for the analysis of temporal event sequences. *HCIL, University of Maryland, College Park, Maryland, Tech. Rep. HCIL-2013-TBD*, 2013. (Cited on 46, 64)

[106] Joseph J LaViola Jr and Robert C Zeleznik. Mathpad 2: a system for the creation and exploration of mathematical sketches. In *ACM SIGGRAPH 2007 courses*, page 46. ACM, 2007. (Cited on 10, 13)

[107] Bongshin Lee, Petra Isenberg, Nathalie Henry Riche, and Sheelagh Carpendale. Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2689–2698, 2012. (Cited on 9, 21, 27, 47)

[108] Bongshin Lee, Rubaiat Habib Kazi, and Greg Smith. Sketchstory: Telling more engaging stories with data through freeform sketching. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2416–2425, 2013. (Cited on 9, 19, 20, 21, 32)

[109] Bongshin Lee, Greg Smith, Nathalie Henry Riche, Amy Karlson, and Sheelagh Carpendale. Sketchinsight: Natural data exploration on interactive whiteboards leveraging pen and touch interaction. In *Visualization Symposium (PacificVis), 2015 IEEE Pacific*, pages 199–206. IEEE, 2015. (Cited on 9)

[110] Michael D Lee and Eric-Jan Wagenmakers. *Bayesian cognitive modeling: A practical course*. Cambridge University Press, 2014. (Cited on 77)

[111] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. Wander join: Online aggregation via random walks. In *ACM SIGMOD*, pages 615–629. ACM, 2016. (Cited on 99)

[112] Jing Li, Jean-Bernard Martens, and Jarke J Van Wijk. Judging correlation from scatterplots and parallel coordinate plots. *Information Visualization*, 9(1):13–30, 2010. (Cited on 107)

[113] Lauro Lins, James T Klosowski, and Carlos Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2456–2465, 2013. (Cited on 68, 87, 99)

[114] Zhicheng Liu and Jeffrey Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2014. (Cited on 70, 71, 77, 83)

[115] Zhicheng Liu and Jeffrey Heer. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics*, 20(12):2122–2131, 2014. (Cited on 106, 117)

[116] Zhicheng Liu, Biye Jiang, and Jeffrey Heer. *imMens*: Real-time visual querying of big data. *Comput. Graph. Forum*, 32(3):421–430, 2013. (Cited on 68, 87, 89, 91, 99)

[117] Zhicheng Liu, Biye Jiang, and Jeffrey Heer. immens: Real-time visual querying of big data. In *Computer Graphics Forum*, volume 32, pages 421–430. Wiley Online Library, 2013. (Cited on 109)

[118] Jock D Mackinlay, Pat Hanrahan, and Chris Stolte. Show me: Automatic presentation for visual analysis. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1137–1144, 2007. (Cited on 32)

[119] Mahbubul Majumder, Heike Hofmann, and Dianne Cook. Validation of visual statistical inference, applied to linear models. *Journal of the American Statistical Association*, 108(503):942–956, 2013. (Cited on 107)

[120] Megan Monroe, Rongjian Lan, Juan Morales del Olmo, Ben Shneiderman, Catherine Plaisant, and Jeff Millstein. The challenges of specifying intervals and absences in temporal queries: A graphical language approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2349–2358. ACM, 2013. (Cited on 46)

[121] Megan Monroe, Krist Wongsuphasawat, Catherine Plaisant, Ben Shneiderman, Jeff Millstein, and Sigfried Gold. Exploring point and interval event patterns: Display methods and interactive visual query. Technical report, Citeseer, 2012. (Cited on 46)

[122] Elizabeth D Mynatt. The writing on the wall. In *Proceedings of the 7th IFIP Conference on Human-Computer Interaction*, 1999. (Cited on 21)

[123] Arnab Nandi, Lilong Jiang, and Michael Mandel. Gestural query specification. *PVLDB*, 7(4):289–300, 2013. (Cited on 9)

[124] Jakob Nielsen. Powers of 10: Time scales in user experience. *Retrieved January*, 5:2015, 2009. (Cited on 65, 70, 72)

[125] Chris North. Toward measuring visualization insight. *Computer Graphics and Applications, IEEE*, 26(3):6–9, 2006. (Cited on 106)

[126] Chris North and Ben Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *Proceedings of the working conference on Advanced visual interfaces*, pages 128–135. ACM, 2000. (Cited on 21, 24)

[127] Frank Olken and Doron Rotem. Simple random sampling from relational databases. In *VLDB*, volume 86, pages 25–28, 1986. (Cited on 98)

[128] Frank Olken and Doron Rotem. Random sampling from databases: a survey. *Statistics and Computing*, 5(1):25–42, 1995. (Cited on 98)

[129] Alvitta Ottley, Huahai Yang, and Remco Chang. Personality as a predictor of user strategy: How locus of control affects search strategies on tree visualizations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3251–3254. ACM, 2015. (Cited on 69)

[130] Catherine Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, pages 109–116. ACM, 2004. (Cited on 106)

[131] Catherine Plaisant, Brett Milash, Anne Rose, Seth Widoff, and Ben Shneiderman. Lifelines: visualizing personal histories. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 221–227. ACM, 1996. (Cited on 45)

[132] Catherine Plaisant, Richard Mushlin, Aaron Snyder, Jia Li, Dan Heller, and Ben Shneiderman. Lifelines: using visualization to enhance navigation and analysis of patient records. In *Proceedings of the AMIA Symposium*, page 76. American Medical Informatics Association, 1998. (Cited on 45)

[133] Ramana Rao and Stuart K Card. The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information. In *Proceedings of*

the SIGCHI conference on Human factors in computing systems, pages 318–322. ACM. (Cited on 28)

[134] Ronald A Rensink and Gideon Baldridge. The perception of correlation in scatterplots. In *Computer Graphics Forum*, volume 29, pages 1203–1210. Wiley Online Library, 2010. (Cited on 107)

[135] Nathalie Henry Riche, Bongshin Lee, and Catherine Plaisant. Understanding interactive legends: a comparative evaluation with standard widgets. In *Computer graphics forum*, volume 29, pages 1193–1202. Wiley Online Library, 2010. (Cited on 33)

[136] Patrick Riehmann, Manfred Hanfler, and Bernd Froehlich. Interactive sankey diagrams. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 233–240. IEEE, 2005. (Cited on 45)

[137] René Rosenbaum and Heidrun Schumann. Progressive refinement: more than a means to overcome limited bandwidth. In *IS&T/SPIE Electronic Imaging*, pages 72430I–72430I. International Society for Optics and Photonics, 2009. (Cited on 66, 70, 87)

[138] Purvi Saraiya, Chris North, and Karen Duca. An insight-based methodology for evaluating bioinformatics visualizations. *IEEE transactions on visualization and computer graphics*, 11(4):443–456, 2005. (Cited on 106)

[139] Steven C Seow. *Designing and engineering time: The psychology of time perception in software.* Addison-Wesley Professional, 2008. (Cited on 65, 70)

[140] Ben Shneiderman. Response time and display rate in human performance with computers. *ACM Computing Surveys (CSUR)*, 16(3):265–285, 1984. (Cited on 65, 70)

[141] Ben Shneiderman. 1.1 direct manipulation: a step beyond programming languages. *Sparks of innovation in human-computer interaction*, 17, 1993. (Cited on 47)

[142] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE, 1996. (Cited on 93)

[143] Ben Shneiderman. Visual user interfaces for information exploration. Technical report, 1998. (Cited on 24)

[144] Yedendra Babu Shrinivasan and Jarke J van Wijk. Supporting the analytical reasoning process in information visualization. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1237–1246. ACM, 2008. (Cited on 32)

[145] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *Proceedings of the VLDB Endowment*, 10(4):457–468, 2016. (Cited on 107)

[146] Tarique Siddiqui, John Lee, Albert Kim, Edward Xue, Xiaofo Yu, Sean Zou, Lijin Guo, Changfeng Liu, Chaoran Wang, Karrie Karahalios, et al. Fast-forwarding to desired visualizations with zenvisage. In *CIDR*, 2017. (Cited on 105)

[147] Richard Thomas Snodgrass, Ilsoo Ahn, Gad Ariav, Don S Batory, James Clifford, Curtis E Dyreson, Ramez Elmasri, Fabio Grandi, Christian S Jensen, Wolfgang Käfer, et al. Tsql2 language specification. *Sigmod Record*, 23(1):65–86, 1994. (Cited on 46)

[148] Ramakrishnan Srikant and Rakesh Agrawal. *Mining sequential patterns: Generalizations and performance improvements*. Springer, 1996. (Cited on 47)

[149] Charles D Stolper, Adam Perer, and David Gotz. Progressive visual analytics: User-driven visual exploration of in-progress analytics. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1653–1662, 2014. (Cited on 66, 70, 85, 87)

[150] Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Trans. Vis. Comput. Graph.*, 8(1):52–65, 2002. (Cited on 19, 22, 24, 27, 68)

[151] Marc Streit, Hans-Jörg Schulz, Alexander Lex, Dieter Schmalstieg, and Heidrun Schumann. Model-driven design for the visual analysis of heterogeneous data. *Visualization and Computer Graphics, IEEE Transactions on*, 18(6):998–1010, 2012. (Cited on 22)

[152] James Sundali and Rachel Croson. Biases in casino betting: The hot hand and the gambler's fallacy. *Judgment and Decision Making*, 1(1):1, 2006. (Cited on 107)

[153] Tableau. Tableau product description, 2017. `https://www.tableau.com/products/desktop`. (Cited on 105, 111)

[154] TIBCO. Tibco spotfire product description, 2017. `http://spotfire.tibco.com/data-discovery`. (Cited on 105)

[155] Matthew Tobiasz, Petra Isenberg, and Sheelagh Carpendale. Lark: Coordinating co-located collaboration with information visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1065–1072, 2009. (Cited on 22)

[156] Christian Tominski. *Event based visualization for user centered visual analysis.* PhD thesis, 2006. (Cited on 105)

[157] Edward Tufte. The visual display of quantitative information.; 1983. (Cited on 40)

[158] John W Tukey. Exploratory data analysis. *Addison-Wesley Series in Behavioral Science: Quantitative Methods, Reading, Mass.: Addison-Wesley, 1977*, 1977. (Cited on 105, 116)

[159] Antony Unwin, Martin Theus, and Heike Hofmann. *Graphics of large datasets: visualizing a million*. Springer Science & Business Media, 2006. (Cited on 86)

[160] Craig Upson, Thomas Faulhaber Jr, David Kamins, David Laidlaw, David Schlegel, Jeffrey Vroom, Robert Gurwitz, and Andries Van Dam. The application visualization system: A computational environment for scientific visualization. *Computer Graphics and Applications, IEEE*, 9(4):30–42, 1989. (Cited on 22)

[161] Andries Van Dam. Post-wimp user interfaces. *Communications of the ACM*, 40(2):63–67, 1997. (Cited on 27)

[162] Jarke J Van Wijk. Views on visualization. *IEEE transactions on visualization and computer graphics*, 12(4):421–432, 2006. (Cited on 106)

[163] Manasi Vartak et al. SEEDB: efficient data-driven visualization recommendations to support visual analytics. *PVLDB*, 8(13), 2015. (Cited on 107, 123)

[164] Jagoda Walny, Sheelagh Carpendale, Nathalie Henry Riche, Gina Venolia, and Philip Fawcett. Visual thinking in action: Visualizations as used on whiteboards. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2508–2517, 2011. (Cited on 21)

[165] Jagoda Walny, Bongshin Lee, Paul Johns, Nathalie Henry Riche, and Sheelagh Carpendale. Understanding pen and touch interaction for data exploration on interactive whiteboards.

Visualization and Computer Graphics, IEEE Transactions on, 18(12):2779–2788, 2012. (Cited on 20, 21)

[166] Taowei David Wang, Catherine Plaisant, Alexander J Quinn, Roman Stanchak, Shawn Murphy, and Ben Shneiderman. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 457–466. ACM, 2008. (Cited on 45)

[167] Hadley Wickham, Dianne Cook, Heike Hofmann, and Andreas Buja. Graphical inference for infovis. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):973–979, 2010. (Cited on 107)

[168] Stefan Wilhelm et al. Moments calculation for the doubly truncated multivariate normal density. *arXiv preprint arXiv:1206.5387*, 2012. (Cited on 112)

[169] Krist Wongsuphasawat, John Alexis Guerra Gómez, Catherine Plaisant, Taowei David Wang, Meirav Taieb-Maimon, and Ben Shneiderman. Lifeflow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1747–1756. ACM, 2011. (Cited on 45)

[170] Degi Young and Ben Shneiderman. A graphical filter/flow representation of boolean queries: a prototype implementation and evaluation. 1998. (Cited on 23)

[171] Xiaoru Yuan, Donghao Ren, Zuchao Wang, and Cong Guo. Dimension projection matrix/tree: Interactive subspace visual exploration and analysis of high dimensional data. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2625–2633, 2013. (Cited on 22)

[172] Robert Zeleznik, Andrew Bragdon, Ferdi Adeputra, and Hsu-Sheng Ko. Hands-on math: a page-based multi-touch and pen desktop for technical work and problem solving. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 17–26. ACM, 2010. (Cited on 10, 13)

[173] Robert Zeleznik, Timothy Miller, Chuanjun Li, and Joseph J LaViola. Mathpaper: Mathematical sketching with fluid support for interactive computation. In *International Symposium on Smart Graphics*, pages 20–32. Springer, 2008. (Cited on 10, 13, 31)

[174] Robert C Zeleznik, Andrew Bragdon, Chu-Chi Liu, and Andrew Forsberg. Lineogrammer: creating diagrams by drawing. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 161–170. ACM, 2008. (Cited on 10, 11, 28)

[175] Emanuel Zgraggen, Alex Galakatos, Andrew Crotty, Jean-Daniel Fekete, and Tim Kraska. How progressive visualizations affect exploratory analysis. *IEEE Transactions on Visualization and Computer Graphics*, 2016. (Cited on 65, 106)

[176] Emanuel Zgraggen, Robert Zeleznik, and Steven M Drucker. Panoramicdata: data analysis through pen & touch. *IEEE transactions on visualization and computer graphics*, 20(12):2112–2121, 2014. (Cited on 9, 17, 68, 89, 109)

[177] Emanuel Zgraggen, Robert Zeleznik, and Philipp Eichmann. Tableur: Handwritten spreadsheets. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2362–2368. ACM, 2016. (Cited on 8)

[178] Emanuel Zgraggen, Zheguang Zhao, Robert Zeleznik, and Tim Kraska. Investigating the effect of the multiple comparison problem in visual analysis. In *To appear at CHI 2018*. (Cited on 42, 101)

[179] Zheguang Zhao, Lorenzo De Stefani, Emanuel Zgraggen, Carsten Binnig, Eli Upfal, and Tim Kraska. Controlling false discoveries during interactive data exploration. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 527–540. ACM, 2017. (Cited on 105, 108, 124, 131)

[180] Zheguang Zhao, Emanuel Zgraggen, Lorenzo De Stefani, Carsten Binnig, Eli Upfal, and Tim Kraska. Safe visual data exploration. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1671–1674. ACM, 2017. (Cited on 101)