Understanding Data Analysis Activity via Log Analysis

Sara Alspaugh

August 3, 2017

Acknowledgement

This research was supported in part by DHS Award HSHQDC-16-3-00083, NSF CISE Expeditions Award CCF-1139158, DOE Award SN10040 DE- SC0012463, DARPA XData Award FA8750-12-2-0331, and gifts from Amazon Web Services, Google, IBM, SAP, The Thomas and Stacey Siebel Foundation, Apple Inc., Arimo, Blue Goji, Bosch, Cisco, Cray, Cloudera, Ericsson, Facebook, Fujitsu, HP, Huawei, Intel, Microsoft, Mitre, Pivotal, Samsung, Schlumberger, Splunk, State Farm and VMware. This research was also supported by a Tableau Fellowship.

Understanding Data Analysis Activity via Log Analysis

by

Sara M. Alspaugh

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Summer 2017

Understanding Data Analysis Activity via Log Analysis

Abstract

Understanding Data Analysis Activity via Log Analysis

by

Sara M. Alspaugh

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Randy Katz, Co-chair

Professor Marti Hearst, Co-chair

The study of user analysis behavior is of interest to the designers of analysis tools. Specific questions studied include: What types of tasks do users perform using this analysis tool? What approaches do users take to gain insights? What interface features help or hinder users in their work? What are the the distinguishing characteristics of different types of users? These questions are often investigated through controlled experiments, observational studies, user interviews, or surveys. An alternative avenue of investigation is to analyze the logs – the records of user activity – generated by analysis tools themselves. In this dissertation we present two case studies using log analysis to understand user behavior. In the first, we analyze records of user queries from Splunk, a system for log analysis, as well as a survey of Splunk users. In the second, we analyze detailed event logs and application state from Tableau, a system for visualizing relational data. We focus in particular on methods of identifying higher-level units of activity, which we refer to as tasks. We include a discussion of the particular challenges associated with collecting and analyzing log data from analysis systems. In addition to this discussion, our contributions include the description of two different approaches for identifying higher-level analysis activity from logs and a summary of the tasks represented in our datasets.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

I would first like to thank my advisors, Professors Randy Katz and Marti Hearst, for their invaluable intellectual guidance and interpersonal advice during my graduate studies. Randy – thank you for guiding me all the way from my first research project as a new graduate student in your computer networking class, through several significant changes in research area, to the very end. Marti – thank you for providing essential technical expertise as well as moral support, as we navigated from quals through to the final thesis. I have learned so much from both of you. I would also like to thank my committee members – Professors Maneesh Agrawala, Mike Franklin, and Tom Griffiths supplied many useful insights for my research.

Special thanks goes to my co-authors, colleagues, and friends. First and foremost, Yanpei Chen stepped in as a mentor and academic older brother to me at a crucial point in my studies, and I was fortunate to learn from his abilities to discern important insights from data and astutely communicate research results, through the course of our friendship and several fruitful research collaborations. Thanks also to Archana Ganapathi for being the best manager a research intern could ever ask for. Thanks to Andrew Krioukov for being my first graduate student research collaborator and continual friend. Thanks to Shivaram Venkataraman and Andrew Wang for their brilliant collaboration across several research publications. Thanks to Ari Rabkin for many years of kind commiseration and engaging conversations, particularly via IM. Thanks also to the many other graduate student friends I made throughout the years for joining me for many laughter-filled hijinks and escapades. There are too many of you to name and more fond memories than I can count.

I would also like to thank Splunk, both for hiring me as an intern to find my dissertation topic, and for allowing me to study their data. Special thanks to my colleagues at Splunk who helped me through many technical challenges during my internship, especially Mitch Blank. I would also like to thank Tableau, particularly Jock Mackinlay, Melanie Tory, and Heidi Lam, for listening to my ideas and enabling me to collect the data I needed for my second case study. Thank you also to all of the participants in my research, whose vital contributions to creating the datasets I studied made this entire dissertation possible.

This research was supported in part by DHS Award HSHQDC-16-3-00083, NSF CISE Expeditions Award CCF-1139158, DOE Award SN10040 DE- SC0012463, DARPA XData Award FA8750-12-2-0331, and gifts from Amazon Web Services, Google, IBM, SAP, The Thomas and Stacey Siebel Foundation, Apple Inc., Arimo, Blue Goji, Bosch, Cisco, Cray, Cloudera, Ericsson, Facebook, Fujitsu, HP, Huawei, Intel, Microsoft, Mitre, Pivotal, Samsung, Schlumberger, Splunk, State Farm and VMware. This research was also supported by a Tableau Fellowship.

Last but not least, my deepest thanks goes to my family for their unwavering love and support. Thanks to my parents, Jean and Brad, for their firm if unsupported confidence in my genius. Thanks to my sister, Samantha, for taking my numerous phone calls to vent. Thanks to my husband, Steve, for believing in me even when I did not. And thanks most of all to my son and his future siblings, for filling my life with pure love and joy.

Chapter 1

Introduction

In this dissertation, we address the problem of identifying higher-level user activity patterns, or tasks, from lower-level user interaction logs, acquired from analysis and visualization systems. Via two distinct case studies, we develop a new analytical method to describe the analysis and visualization tasks that users perform. In the next section, we describe this problem and why it is important. We then outline our basic approach and how it compares to related work in this area. Lastly, we briefly relate our findings from our case studies, and describe the organization of the rest of this document.

1.1 Trends and Motivation

There are several broad trends that motivated the research of this dissertation. We first describe several important factors that make this problem broadly relevant, then describe what makes it challenging to solve.

1.1.1 Data Analysis and Visualization, Logging, and User Behavior

Three trends contribute to the relevance of the problem of identifying higher-level activity patterns from lower-level user interaction logs: the importance of analysis and visualization tools to modern business and science, the prevalence of logs as a source of data about tool use, and the rising interest in understanding and modeling the behavior of software users. We discuss each in turn.

Analysis and visualization of data was one of the first applications of computing and remains one of the most important. As the ability for computing systems to store and process more data has increased, so has interest in developing new analysis techniques and applications to apply data analysis in a wider variety of domains. The increase in access to and interest in data and its analysis has been reported in various outlets under the terms “big data,” “the data deluge,” and the rise of “data science.” The increasing importance of data and its analysis, especially in the job market, has been underscored by many business and technology leaders [164].

Logging emerged as an important early feature of computing systems due to the aid it provides to application developers and users in understanding what happened during program execution, including how the user interacted with the program. Logs are now a major source of business-critical data, particularly in the software technology industry. In software technology businesses, data about how users interact with the product, recorded via logging, is an important source of insights. Software companies collect vast amounts of data about their customers, which they can use to understand who their customers are and how they interact with the product.

Combining these two ideas – the importance of data analysis and visualization, along with the importance of analyzing logged data to understand user behavior – we find that logs from data analysis and visualization tools in particular can be analyzed to generate insights about how people analyze data. The user interaction data can even be used to create statistical models of user behavior that can be incorporated into the "intelligent interface features of the application, such as recommender systems that suggest possible next analysis or visualization actions to the user, as discussed in Section 2.3.3.3. Such applications have the potential to greatly increase user analyst productivity, which is crucial considering the importance of data analysis in the future economy.

1.1.2 Summarizing User Activity and Identifying Tasks

One important component for gaining such insights and enabling these applications is identifying and understanding what types of activity users are performing with analysis and visualization tools. While these tools are designed to enable specific tasks, its important to track actual usage to evaluate how well they support the targeted tasks and create intelligent interface features. Particularly in the case of recommender systems, it has long been recognized that it is useful for behavioral models to track what the user is trying to do at various levels of abstraction in order to make relevant recommendations about what other visualizations and analyses to try [27].

Useful insights may include: What features of analysis and visualization tools do users use most? What aspects of the tools need to be improved? Are there different types of users and if so how does their tool usage differ? In terms of insights, tool designers can use evidence regarding which activities were most common in these participants' analyses to support decisions regarding which features to protect and prioritize. On the other hand, researchers can use evidence regarding which tasks are least common or not pursued at all to guide future investigations into whether these tasks are not common because they are poorly supported, or because users don't need to do them. Examining how the frequency of different tasks changes based on data domain indicates which tasks are useful for different types of customers. We also learn if the approaches users take to solve their problems line up with how the tool was designed to solve these problems. We see the different ways users have discovered to accomplish the same task, and if these different approaches are reflective of different user personas [177].

Most importantly, we learn which tasks users most struggle to perform, because this suggests weaknesses in the tool design. We learn if the mistakes the users make are due to usability issues with the tool, or errors in planning or reasoning on their part. Tool designers are interested in making tools that help users be productive, which usually means reducing mistakes, though some mistakes may be serendipitous. Researchers will be interested in real-world examples of usability issues, but may be even more interested in mistakes that generalize.

Lastly, information about users' motivations (i.e., what higher-level actions they are interested in) could be used to drive a variety of future features.

1.1.3 Challenges in Using Logs to Understand User Behavior

It can be difficult to identify higher-level actions from lower-level interaction logs unless the interface is designed to explicitly encode these higher-level actions. This is because lower-level activity is difficult to group together and then describe at a higher-level. Logging higher-level, or semantic-level interface commands better reveals user behavior and intent than low-level commands do. For example, HARVEST provides insight provenance by providing visualization widgets that can be re-used across applications and that support "semantic interaction" to help identify and capture user intent [50, 51]. The creators of the visual analytics system HARVEST, described in Section 2.3.3.1, developed an action and task taxonomy as the basis of the operators provided in their tool, so that user tasks could be captured at the application-level rather than having to code user interactions post facto [50]. Unfortunately, most visual analytics interfaces are not designed, as HARVEST was, to explicitly encode higher-level actions. Thus, it can be difficult to make sense of what a user was trying to do by looking at the logs. There are several more specific reasons for this.

- **Missing context:** The lower-level events do not contain all the information and context needed to understand their significance within the whole, such as what the current application state is, or what actions they are logically related to. Summarizing activity at a higher level implicitly provides some of this context. Challenges in this vein are described in detail in Chapter 3.

- **Volume and frequency:** The lower the level of abstraction of the event, the higher the frequency and volume of occurrence. This sheer volume can be hard to understand without techniques that summarize the information contained in the larger set.

Thus, understanding user behavior from typical data analysis system logs is a difficult problem, and our approaches attempt to mitigate these challenges.

1.2 Problem and Approach

Given this context, the problem we solve can be stated as: given a log of user activity from a visual analysis tool, summarize the activity to reveal the types of higher-level activity or tasks they represent. The tools we use in our case studies are Splunk (Chapter 4), a software tool for the analysis of machine-generated data, such as logs, and Tableau (Chapter 5), a software application for visualizing relational data for analysis. The user interaction data is information that was logged during the use of these visual analysis systems about the actions users took to analyze their data.

1.2.1 Background

We provide more background information about logs and logging in Section 2.1. In our Splunk case study, we collected user queries, which are pipelines of data transformation operators. In our Tableau case study, we collected event logs and application state from user analysis sessions, augmented with user annotations about their thought process, and reports summarizing their work. We summarize this lower-level interaction data from the logs into more abstract, higher-level descriptions of the tasks or types of activity users were performing by taking those actions. What approach should be taken specifically depends on what kind of information is contained in these activity records. Some logs will be better for understanding user behavior than others by having more and better quality information. This is elaborated upon in more detail in Chapter 3.

As mentioned, in our Splunk case study, we only have user queries, whereas in our Tableau case study, we have user actions, application state, and annotations and reports from the user indicating their thought process. The relevant information that might be contained in such a log is shown in Figure 1.1. In this figure, the user interacts with the visualization system by transforming and visualizing their data. They are influenced in their actions by such things as their mental understanding of the data, their role and its corresponding requirements (e.g., business analyst, student, salesperson), and the tasks or goals they are trying to achieve. The cylinders in the figure correspond to the analyzed data and application state. The user applies transformations to modify and visualize this data, which is then displayed. The information that is available depends on the log. In our Splunk case study (Figure 4.1), we collected query logs, so we only had access to the transformations that were applied, not the data and application state, or how the data was visualized. In our Tableau case study (Figure 5.1), we were able to collect significantly more data, including the data analyzed and screenshots of the application after each action. We also collected information about users mental processes, via log annotations and analysis reports. We highlight the roles, tasks, and mental model in the figure using a striped pattern to indicate that this data gives us only partial access to this information, because the annotations and report are an incomplete representation of the users thought process.

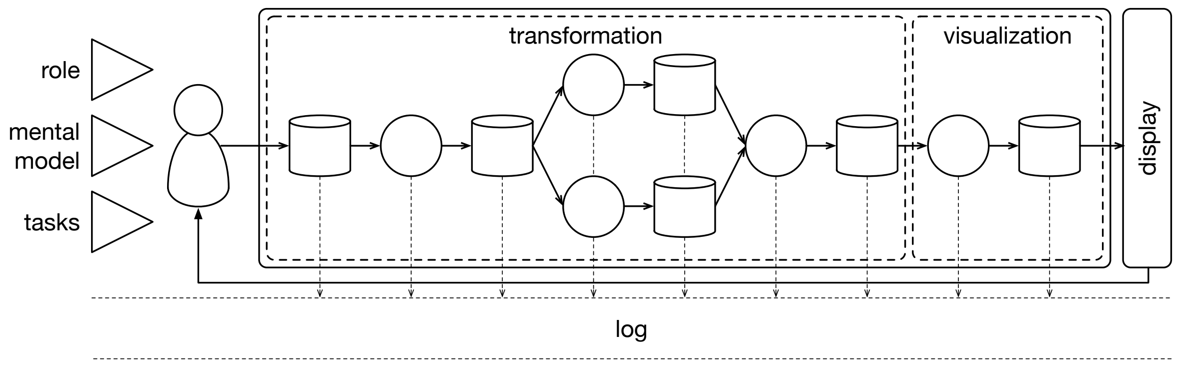


Figure 1.1: In the Splunk case study, we collected only the *transformation operations* users applied, not the *visualization operations* that users applied, the *data* they applied them to, nor any information about their *goals*.

1.2.2 Approach

Given this problem setting, our core approach is as follows. First, we categorize every transformation (in Splunk) or action (in Tableau) that the user can take in the application into a smaller number of categories that represent type of activity involved, for example, filtering. We use this category information to group the transformations or actions respectively into groups and then use these groups to summarize and describe tasks, or user activity at a higher level of abstraction.

In our Splunk case study, this involved categorizing hundreds of command invocations from the query pipelines and then clustering them to identify common usage patterns representing common tasks, as well as examining common pipeline patterns. In our Tableau case study, this involved categorizing hundreds of different action types, and then segmenting each user's event stream into consecutive subsequences that represent some higher level activity or task. One difference in these two cases, besides the type of data we had access to, is that in the Splunk case, we did not group actions by time because we did not have access to information that would allow us to sessionize user queries or take user information into account. Thus, we focused on grouping common actions across users, whereas in the Tableau case, we focused on grouping actions within each user's session. The key idea in both cases is to cluster similar types of low-level activity to identify common tasks.

1.2.3 Comparison to Related Work

In comparing our work to the related work, there are several key differences between this dissertation and the most relevant pieces of research, which are described in detail in Section 2.5. We summarize them here.

Small, controlled studies versus real-world analyses: The first difference is that in much of the related work, the studies involved assigning a small set of participants a short, specific, researcher-designed data analysis task over a researcher-chosen dataset. In contrast, in our case studies, participants chose their own datasets to analyze and formulated their own investigations, many of which took many hours or days to complete. Thus, we have a much larger, diverse, and in many ways more realistic set of usage data in both our case studies. **Encoding insights versus summarizing activity:** One drawback of our larger, more realistic datasets, however, is that due to their scale, they are harder to analyze in-depth using manual approaches, such as coding all user analysis "insights" by hand, as several of the related studies do – constituting the second main difference between our approaches. The related studies tend to take this other approach over their smaller, carefully designed data analysis tasks because they have different goals than ours, such as identifying specific interface features that lead to particular types of insights. **Prototype analysis tools versus real-world products:** The third difference is that these studies also often tend to use data analysis tools that are research prototypes, whereas the tools in our case studies are widely-used commercial platforms. (As an exception, the study in question evaluated a workflow history prototype, so again it differs in its main goal [61].) **Differing analysis and visualization methods:** Lastly, due to their differing goals and chosen data analysis systems, much of the related work uses different types of analysis methods and visualizations than we do, though there are several similarities, such as between our Chapter 5 visualization system, and our use of transition diagrams to characterize user activity. We describe these similarities in more detail with respect to specific work in Chapter 2.

In Chapter 2, we also describe other areas of work that are related in that they are other applications of log analysis or other attempts at understanding analysis tasks, but they have either completely different goals or different approaches than we have here.

1.3 Contributions and Roadmap

With the approach described above, we learned several things about how our specific case study systems are used.

In the Splunk case, for example, we found that a large portion of log analysis activity in Splunk consists of filtering, followed by aggregation. Filtering is also used in conjunction with aggregation to count occurrences of different things, which is an important task in log analysis. The third most common activity was augmenting the data, such as reformatting it. The prevalence of reformatting as a portion of log analysis activity is reflective of the fact that much log data is structured in an inconsistent, ad hoc manner. Taken together, the prevalence of filtering and reformatting activity in Splunk suggest that it may be useful for system developers to collaborate with their end users to ensure that data useful for the day-to-day management of such systems is collected. In log analysis with Splunk, summarizing with respect to the time dimension is an important use case – of all the fields accessed in all the datasets analyzed by users, time was one of the more common ones. We also found that task complexity tended to be low – filtering, aggregating, and reformatting, versus complex machine-learning pipelines. Lastly, our survey results show that log analysis is not only for IT departments; non-technical users need to extract business insights from logs to drive their decision making as well.

In the Tableau case, we found that the submissions tend to follow the workflows proposed in the literature: first data munging, then analysis, then presentation, though there is some iterating back and forth. This likely reflects the fact that the data we collected was from students following particular instructions in an assignment that directed them to follow this workflow, and so these answers are not necessarily representative of other use cases. We also saw that activities like pan, zoom, and highlight, and filter, tend to happen more in the presentation rather than the analysis phase. Many segment types are followed by presentation, but this might be because presentation is the most common segment type. It could also be that what are often thought of as exploratory actions like pan, zoom, or highlight, and filter, are actually more often used to tweak a presentation than to explore during analysis. Lastly, we compared our dataset to findings in related work over similar datasets and found that of only the high undo-to-redo ratio also held for our dataset.

While we use the same underlying approach in both the Tableau case and the Splunk case, the results are not very comparable, because of the different purposes and features of the two tools, as well as the different quality of our datasets (i.e., we had much less information in the Splunk case). Though there are some high-level similarities, such as in the use of aggregation and visualization, Splunk is used primarily for working with logs, while Tableau is used for working with tabular data. Nonetheless, we take the fact that our high-level approach works with two different systems to be an encouraging sign that the approach would work well for interaction logs from other systems as well, though as in our case studies, it will vary in the specifics.

In developing these descriptions via our approach described in the last section, we had to overcome several challenges. We outline these challenges in Chapter 3 as a reference to future researchers who may be interested in undertaking projects to analyze log data to understand user behavior. Thus, this represents a contribution in addition to our approach and the above findings.

To summarize, we present a brief synopsis of our contributions along with a roadmap identifying the relevant chapters where these are discussed. First, the next chapter reviews work on logging, using logs to understand, model, or present user behavior, applications of models of user behavior, tools and techniques for log analysis, visual analysis task taxonomies. The next three chapters represent our contributions:

- A discussion of the key challenges in using log data to understand user behavior (Chapter 3).

- An approach to summarizing interaction data – categorize low-level activity and then cluster it into similar groups to identify common tasks – that can be used with logs from other visual analytics systems.
- Two case studies in applying this high-level approach, resulting in summaries of user activity specific to each system and set of users in question, the details of which are summarized in Chapter 4 (Splunk) and Chapter 5 respectively (Tableau).

Chapter 6 summarizes and concludes. As the rest of the dissertation illustrates, our high-level approach, which involves re-categorizing, clustering, and then labeling the low-level interaction log data, provides a way to summarize the analysis activity in interaction logs at a more general and understandable level.

Chapter 2

Related Work

In this chapter, we provide background and related work on the topics of logging, log analysis, and analysis and visualization task taxonomies.

2.1 Logging

In computing, a *log* is a record of occurrences that occurred during a running computer program. To log is to write this information to the log. Logging is done by adding lines of code to a program that write out the targeted information when a relevant occurrence takes place. The types of occurrences that are logged can include interactions from a user, internal conditions related to program execution, communication received from other systems, and more.

Units of information in a log are often called *events*, corresponding to the fact that this information is often issued when an *event* – an occurrence that is handled by the software – is triggered. Graphical User Interfaces (GUIs) and other interactive programs are usually event-driven. In general, events can be triggered by a user or some other external input, as with GUIs, or internally by conditions within the program. Exactly what information is logged for each event and its format varies widely – it may include information such as the parameters, execution duration, caller, and source code location of the function or functions that executed. At a minimum, the information logged usually includes the timestamp and the type of occurrence. Often, other important details characterizing the occurrence are also logged. Instead of events corresponding to actions, another alternative is to log the current application state. This is less common in logs in general especially because it can take up a great deal of storage space, but is sometimes done when recording workflow histories, as discussed in Section 2.3.3.1. Sometimes, both the states and the actions that led from one to another are recorded, for example, linearly, or in a graph-like structure. Action logging is sometimes referred to as the command object model [45].

An example event logged by Tableau is shown in Table 2.1. Such events are typically logged for debugging and performance monitoring purposes. In Chapter 3 we discuss specif-

```
2013-09-24 22:06:34.197 (-,-,-,-) 0d08:  UpdateThumbnail called,
WorksheetWnd.cpp, line 1685
```

Table 2.1: Example of an event logged by Tableau. This event records that a function was called, giving its location in the source.

ically what types of events and associated information should be logged for user modeling. Below, we discuss which basic types of information should be included in events:

- **Timestamps:** Events should always be accompanied by a timestamp that describes the date, time, and timezone information. Timestamps are important for understanding the order and rate of events but are not always accurate reflections of when an event truly occurred. There is work not only on how to synchronize UI events but also on distributed system events more broadly e.g., for debugging. A discussion of how to deal with this problem for UI events is beyond the scope of this dissertation since it was not a challenge we faced; we refer the reader to other work [64].
- **User ID:** Ideally each event should be connectable to information about the user "responsible" for initiating the event, in the sense that their interaction with the program "caused" the event. For some events, the "user" responsible may be the system itself, for example, in the case of garbage collection. In general, determining causality is not trivial, but for the events of interest for user modeling, it should be straightforward.
- **Version and configuration:** It is critical to provide some information that ties each event recorded to metadata about the version and configuration of the interface that generated that event. This is important because exactly what information is logged and the format it is logged in tends to change across versions and configurations. Without this information, it can become unnecessarily difficult to parse the logged data, and ambiguities may be introduced. Ideally, even changes to minor features of the interface would be versioned, to facilitate A/B testing.

In the context of the Internet, there are additional challenges associated with logging. Activity that occurs on remote hosts must be sent back to a collection point or collected post-facto. This introduces the additional challenge of ordering the combined set of events into one timeline, as noted by Hilbert and Redmiles, which is difficult without globally synchronized timestamps, itself an engineering challenge [64]. Many cloud-based analytics applications involve a client-side user interface where much of the user's analysis actions, especially those pertaining to visualization, are executed. To collect information about this activity, the client must record and store or send it. If not done carefully, this logging has the potential to hurt application or website performance, due to the storage and communication requirements. Hilbert and Redmiles propose an approach to monitoring web-based applications that involves being explicit about what usage data is needed during application development, implementing a flexible and modular monitoring architecture separate from other program logic, and abstracting events to provide the ability to filter and define usage at multiple levels of abstraction [63]. We discuss these points more in Chapter 3, as these are important considerations for logging user behavior that if followed would aid in case studies like our own. Some researchers have proposed frameworks for tracking events like mouse movements, scrolling, and clicks in webpages [13, 29]. This general interaction information is often important; however, for many applications, it may be more useful or necessary to implement application-specific tracking code.

Log analysis is often used in research as a means to an end, rather than as the subject itself. In the next section, we review log analysis techniques. In the following sections, we review some examples of applications to which log analysis has been applied. We provide extra detail for those applications that are most similar to the work presented in this dissertation.

2.2 Tools and Techniques

In this section we review some common tools and techniques for log analysis. This is relevant not only in comparison to our approach, but also in comparison to the uses revealed in our case study of Splunk, a log analysis system (Chapter 4).

2.2.1 Tools

As described in Chapter 4, we studied logs from Splunk, which is itself a system for log analysis. Here we describe some others, both academic and commercial. Dapper, Google's system tracing infrastructure, is used by engineers to track request latency, guarantee data correctness, assess data access costs, and find bugs [145]. These engineers use transformations to analyze the logs similar to those used by Splunk users, such as filtering events, aggregating the data, projecting fields, and visualizing the result. Other systems, such as Sawzall and PigLatin, include query languages that extract data from logs with heavy use of these same types of transformations [122, 115]. This suggests these basic types of activity are typical of log analysis usage. However, more involved techniques for log analysis have also been developed, which we describe below.

2.2.2 Log Analysis Techniques

Many published techniques for log analysis center around the main challenges in working with logs, such as dealing with messy formats, and solving event-based problems, like causal analysis [112]. This includes event and host clustering [99, 102], root failure diagnosis [33, 91], anomaly detection and dependency inference [113, 95], and data extraction [90, 170]. Although these techniques' motivating use cases overlap with Splunk use cases described in Chapter 4, we did not observe much use of such techniques in our Splunk dataset (though Splunk does provide, e.g., clustering and anomaly detection functionality). Techniques for analyzing search query logs involve examining query terms and extracting user sessions [72, 130]. Such techniques would be most applicable to our Splunk dataset, but due to event provenance issues described in Chapter 4, we were not able to extract sessions from our dataset [99, 102]. In terms of how these compare with our work, the most relevant approaches are those that focus on clustering events, because that is our main strategy in both of our studies. Clustering is most often applied to query logs to find similar queries, though the similarity metrics used differ [177, 73, 40, 14]. Similarity metrics include query result set similarity [14], syntactical similarity [14], datatype and topic similarity [73, 40], time [73, 40], action set similarity, specifically the Jaccard index [177], and human judgment via manual coding and grouping [55]. In in our Chapter 4 use case, we use the second and third type of similarity metric listed, and the third, fourth, fifth, and sixth type in our Chapter 5 use case.

Hilbert and Redmiles' survey on the use of UI events to evaluate usability describes techniques that are more relevant to this dissertation than those described above [64]. Such techniques include:

- Synchronizing and searching
- Selecting, abstracting, and recoding
- Counting and summarizing
- Sequence detection
- Sequence comparison
- Sequence characterization
- Visualizing events
- Integrating event information into other evaluation frameworks

. They cite several examples of each technique in their work; we refer the reader to their survey for these references and instead describe the general techniques and how they compare to our approaches below.

Synchronizing and searching is a technique whereby events are synchronized with other types of session data, such as recorded video or hand-written observations. This technique appears in a sense in our work in that in our Chapter 5 case study, we asked participants to annotate their event logs with their thought processes and indicate the section of their analysis findings report the annotation pertained to; this technique was not applicable to our other case study.

Selecting refers to filtering events, *abstracting* refers to grouping or splitting events, and *recoding* refers to re-categorizing these filtered, split, or grouped events. In both of our case studies, we apply some filtering to remove machine-generated events, but primarily we recategorize events into coarser-grained, higher-level analysis categories to make better sense of them.

Counting and summarizing are the straightforward techniques of aggregating the different types or higher-level categories of events and we also make extensive use of these approaches in both of our case studies.

Sequence detection refers to the technique of identifying target sequences – abstract or specific to an application – within the event stream, for example to abstract and recode the event stream or to identify subsequences for investigators to inspect further. This technique is not applicable to Chapter 4 because we were not able to sessionize the queries, as we describe in that chapter. However, it is applicable to our case study in Chapter 5, where we seek to identify high-level tasks represented by our event stream. However, one difference is that in our work, we seek to partition the entire event stream instead of identifying small disjoint subsequences throughout. Also, the approaches in the literature often assume that the subsequences of interest are identified a priori, either completely or partially, whereas we seek to identify reasonable boundaries within the entire event stream that divide it into subsequences or segments without specifying the segments themselves in advance. One exception is the work of Santos and Badre, which detects "chunks" (which we call segments) based on pauses in user activity, which serve as the boundaries between chunks [134]. In Chapter 5, we discuss how such pauses did not seem to be good indicators of boundaries as judged manually by investigators.

Sequence comparison refers to the technique of detecting divergence between a specified subsequence and the event stream, to compare this actual usage against a specified trace of expected usage, to identify potential usability problems. Like with the techniques described in the previous paragraph, these techniques assume that the subsequence to compare is provided by investigators a priori, since they target a different problem than the one we are investigate here.

Sequence characterization refers to the technique of generating models to describe the sequential structure of these sequences. Hilbert and Redmiles describe two different approaches in this case: Markov-based and grammar-based. We use the first approach in both of our case studies, in which the sequences are modeled as nodes in a graph, with edge weights corresponding to the frequency of transitions between each sequence. Grammar-based approaches define equivalent sequences by describing lower-level sequences that combine into a higher-level action; for example, "print" may be achieved in a user interface by clicking on the print button then clicking "OK" in the print dialogue, or by using a short-cut key. Both of these subsequences could be mapped in a grammar-based approach to the same action: print. This is an interesting approach, but we did not use it because we did not find frequent enough occurrences of any sub-sequence in an initial investigation to justify pursuing it. In other words, there were too many unique subsequences to create a grammar for without a prohibitively long manual investigation. One possible reason is that participants in both of our case studies all analyzed different datasets, so that the data being analyzed would have to be discounted to make a literal match. The drawback of this is that it makes the more abstraction that is applied, the less informative or plausible it is to make claims about what sequences of high-level abstract categories of activity are trying to accomplish. However, with the right kind of selection, abstraction, and recoding, as described above (particularly of the data analyzed), a grammar-based approach may have yielded interesting results. Incorporating information about the data analyzed would have only been possible in our Chapter 5 case study because we did not have such information for our other case study.

Visualizing events is self-explanatory and for both case studies we created both interactive visualization tools as well as static graphs to better understand and present user activity. *Integrating event information into other evaluation frameworks* is not a technique we used at all, but some recent work related to this technique is discussed below [117, 16].

2.2.2.1 Other Means of Studying Behavior

In Sections 2.3.2 and 2.3.3, we reviewed the use of logged interaction data to understand user behavior, enhance interfaces, and build user models. For contrast, here we briefly review some examples of ways researchers study user behavior without the logged interaction data. Approaches include conducting controlled experiments, short-term observational studies, long-term field studies, user interviews or surveys, or literature and product functionality reviews.

For example, Gotz, et al., conducted an empirical study to examine user behavior, in which they asked 30 participants to perform one of two researcher-designed analysis tasks using commercial off-the-shelf tools [49]. The researchers provided material for participants to take notes and recorded video and audio of each session using a "think-aloud" protocol. The researchers also manually recorded all of the activity the participants performed, and later coded these records into a standard format. The two main analysis patterns researchers identified are the scan pattern, in which users inspect a series of objects to compare them, and the flip pattern, in which users toggle between data subsets to compare them. The researchers argue that these patterns reflected limitations in the analysis tools provided. The tools we examine in our studies are not as limited, and also participants in our studies performed a variety of different tasks rather than tasks we assigned them; thus, we did not observe these as prevalent patterns. The researchers developed several recommendations, centering on the importance of tracking analysis actions and exposing that to the user along with note-taking functionality to enable re-usability and communication. This work could have in theory been conducted using log analysis, but the researchers chose to manually record activity instead.

Some researchers conducted a longitudinal, observational field study of intelligence analysts to describe their methods and processes at a high-level, and produced design recommendations based on this [79]. Similar studies have been conducted in the field of bioinformatics [136]. Shorter, more controlled studies are often used when the researchers want to code and analyze the insights gained from such studies [87, 135, 78]. For example, Gomez, et al., carefully designed a novel evaluation approach called layered insight- and task-evaluation (LITE) that combines features of other controlled studies to attempt to make-up for their shortcomings [48]. However, some researchers advocate for the importance of multidimensional long-term case studies (MILCs) for evaluating tools [144].

VEEVVIE is a software tool for collecting and analyzing data from carefully designed visualization and human-computer interaction experiments [117]. Similarly, VA^2 provides an integrated environment for visually displaying and analyzing different types of UI data together, such as think-aloud, interaction, and eye movement data [16]. A literature survey of methodological approaches in general user experience research supports that idea log analysis is not a common approach [16].

Other work has focused on specifically understanding user visualization behavior and preferences, as opposed to analysis behavior. Specifically, researchers seek to understand the characteristics of both the user and the visualization that impact the user's effectiveness, efficiency, and satisfaction with their visualization task. This work is important, but has been limited in that studies understandably tend to focus on only a few visualization types, user characteristics, and task characteristics at a time. For example, one team of researchers examined the impact of four user characteristics (perceptual speed, verbal working memory, visual working memory, and user expertise) on the effectiveness of two common data visualization techniques: bar graphs and radar graphs [153]. These researchers did not use log data, but instead relied upon a carefully designed experiment where participants were asked questions relating to the visualizations presented to them. Their findings include conclusions like: users with a higher working memory prefer radar graphs to bar graphs. In a similar study, researchers collected textual descriptions of the insights users gained from simple bar graphs, simple line graphs, and combinations thereof [172]. The researchers used these results to develop a taxonomy of visual insights according to the analysis task at hand, as well as identify what type of graphs users prefer for which tasks. The information in these studies could be used to create systems that suggest the best visualization for a user based on the user's characteristics and task at hand.

More examples of the means of studying the behavior of users of visual analysis tools can be found in the BELIV workshop (Beyond Time and Errors: Novel Evaluation Methods for Visualization), dedicated to the discussion of challenges in and new methods for visualization evaluation.

2.3 Applications

Log analysis is often used in research as a means to an end, rather than as the subject itself. In the next sections, we review some examples of applications to which log analysis has been applied. We provide extra detail for those applications that are most similar to the work presented in this dissertation.

2.3.1 Computer Systems

Logs have been used in the context of computer systems to explain system behavior [31, 171], understand failures [123, 114], identify system design flaws [52], spot security vulnerabilities [85], highlight new phenomena [116], and drive system simulations [58]. The above examples are often of a more general or academic nature, whereas users of the analysis tools we study (Splunk and Tableau) are often looking for timely business insights specific to their situation. Nonetheless it is useful to keep these use cases in mind, particularly when reviewing the Splunk user log analysis activity in Chapter 4, because these are typical applications of Splunk. By contrast, Tableau is originally designed for and best suited to the examination of relational data.

2.3.2 User Behavior

Log analysis is used not just to understand computer system behavior, but also to understand human behavior, as we do in both Chapters 4 and 5. The types of human behavior studied via log analysis can include how users interact with interfaces, how users seek information, and more. Hilbert and Redmiles surveyed the use of UI event logs for providing usability information [64]. The type of usability data captured in the work they survey includes task performance, behavior, understanding, opinion, and stress, of which, only the behavioral portion is our focus. Our focus is on identifying the types of tasks users perform while analyzing data, as part of a broader effort to understand analysis and visualization behavior. We review work that uses log data for this purpose in Section 2.3.2.1. Most of this work differs from ours in that the data gathered is usually from small experiments (i.e., 10-16 participants and 20-45 minute analyses sessions) in which all participants were given the same dataset and asked to perform the same tasks. In contrast, in both of our case studies, participants engaged in in-depth analyses over a wide range of datasets of their choosing.

2.3.2.1 Analysis and Visualization

Logs from analysis tools can be used to investigate the reasoning process of analysts. Topics investigated by researchers include how to improve analysis tools, how to discover analysts' reasoning process for solving complex problems with these tools, how to characterize different types of analysts based on their approaches, and more.

One comprehensive system for studying user analysis behavior is GlassBox, a system for capturing and leveraging user activity across applications, created by Cowley and collaborators [36, 37]. In addition to data capture, GlassBox aims to serve as a platform for interoperable analysis tools that leverage this data to provide intelligent assistance to analysts, as well as a platform for evaluating the effectiveness of analysis tools. We are unaware of any reported results from specific studies done using this platform. Most studies of using log data to understand user analysis, like ours, have focused on datasets collected from specific analysis applications, sometimes from user experiments, and other times from everyday use.

In a set of two studies, Jeong and Dou, et al., asked ten financial analysts to analyze a synthetic bank transaction dataset to uncover suspicious activity using an academic transaction analysis tool [73, 40]. They asked participants to think aloud to reveal their thought processes by describing what steps they were taking and why. They collected log data from the analysis tool, as well as information about participants' strategies and findings. This is very similar to the data we collected for the case study described in Chapter 5, except the Jeong and Dou, et al., study involves the use of a different tool, and participants were all asked to perform the same, shorter analysis task, whereas our participants performed

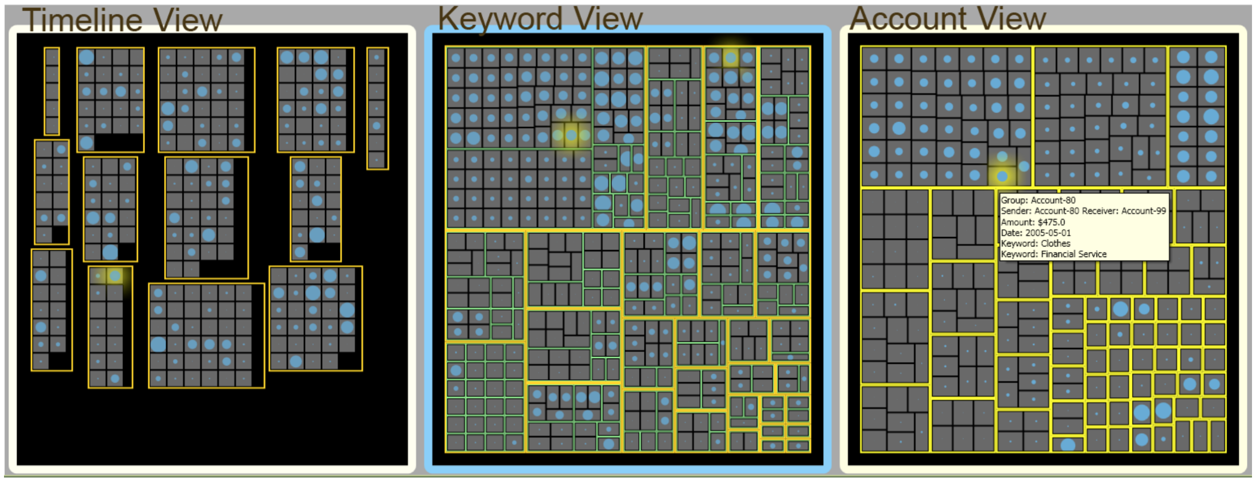


Figure 2.1: Treemaps used by Jeong and Dou, et al., to visualize user financial analysis activity [73, 40].

in-depth explorations of different datasets of their choosing. The researchers created two visualization tools that present the log data to help them understand the analysts' reasoning process, also similar to our approach. These visualization tools were specific to the analysis tool the participants used, much like our visualizations are. One of the visualization tools in particular resembles the tool we describe in Section 5.3.1. The other visualization uses a set of three tree maps to cluster user activity by the type and topic of data examined, as well as time, as shown in Figure 2.1. The researchers asked four coders to reconstruct the analysts' reasoning process using only these visualization tools and the log data. The coders were able to reconstruct the analysts' reasoning process with 60-82% accuracy, in about as much time as the original analysts had taken. This suggests that given the right visualization tools, log data can be used by trained coders to reconstruct analysts' reasoning process with enough accuracy to justify larger studies that could be used to learn more general principles about how analysts reason. One difference between our work and this work is that rather than coding all of our data and investigating whether human analysts could reconstruct what users did, we examined ways of summarizing user activity both manually and automatically – a related, but not identical formulation.

Heer, et al., implement graphical analysis histories in Tableau and use this history data to identify usage patterns [60]. They capture and present the application states and the user's path through them, as well as user actions that they categorize into coarser-grained groups. This is essentially the same type of data as we collect and analyze in Chapter 4. Their findings are that undo actions are used to delete, formatting is the most prevalent action, the ShowMe visualization recommendation tool is useful, and that their rules for summarizing the history presented are effective. These rules are particularly relevant to our work, as we employ a different approach to summarizing user activity, the details of which are discussed in Section 5.3.4.

Boyadin, et al., use interaction logs in a study with 16 participants, to compare two different interaction techniques as to whether they result in users arriving at different findings [21]. Participants were asked to perform a short but fairly open-ended exploration of a dataset using these different interaction techniques. The authors also collected descriptions of the participants' findings, the participants' assessment of the importance of each finding, screenshots associated with each finding, a user questionnaire, and audio and video recordings of each session. They use a Gantt chart, which highlights the different segments of a timeline of events, to visualize participants interactions as they develop each finding. This is similar to our visualization of segments in Chapter 5, except that ours are oriented vertically, and we don't compare multiple subjects in the same view since they pursued entirely different analyses.

Reda, et al., present another evaluation technique for characterizing visual exploration that incorporates interaction data from log files as well as video recordings. It involves identifying and coding mental states from the video and then analyzing the flow between mental states and logged interactions using transition frequency diagrams [127]. This approach builds upon the work of Saraiya, North, and others, described in Section 2.4 [135, 110]. In a study with ten participants performing open-ended exploration of a single dataset on two different displays, they found that the larger display resulted in less display modification and more insight generation. Similar to this work, we also use transition frequency diagrams in both of our case studies, though we do not code and incorporate insight information, as already mentioned. Doing so would be an interesting extension of our work. Their results are not very comparable to ours as they were obtained from users analyzing a different dataset on a different tool from those included in our case studies.

In a similar study, Guo, et al., asked ten students to analyze a synthetic corpus of news reports, resumes, and email headers, using a text analysis tool [55]. They collected the insights reported by the user, as well as interaction data from logs. Like the previous study, they then coded this data into actions and insights. Specifically, they coded sequences of actions into patterns representing higher-level activity. This is similar to our notion of tasks; however, we follow a different approach. Guo, et al., also created a visualization of these higher-level actions that resemble our visualization of the Tableau log data in Chapter 5, which suggests this is a useful format for viewing log data. They looked at frequently recurring patterns, as well as the transition frequency between high-level actions, which we also examined for our datasets. We also examined frequently recurring patterns in our Splunk case study in Chapter 4, using sequences of high-level transformation categories. However, when we tried to do so in our Tableau case study in Chapter 5, we did not find any interesting pattern that occurred frequently enough from which to draw conclusions – we discuss possible reasons for this in Chapter 5. Guo, et al., then examined the correlation coefficients of each action and each insight, as well as which actions participants preferred when many equivalents were available. Using this analysis, the researchers were able to identify specific useful and not useful interface features, but it is unclear to what extent these generalize even to other text analysis tools, or visual analysis tools in general.

One feature that distinguishes these studies from our work is that in these, researchers undertook extensive coding work to identify and classify findings or "insights", whereas we only examine actions. To make the classification and analysis of findings or insights feasible, they gave all participants the same small analysis assignment over the same dataset. By contrast, in our case studies, participants chose whatever dataset they wanted, investigated whatever questions they wanted, and spent as much time as they wanted analyzing it. This resulted in a higher volume of interaction records and a large number of different situations in which to classify findings and insights. This volume makes insight classification less feasible, and the different dataset and analysis goal contexts would make it difficult to compare different analyses to one another, as the other studies did. Thus, there is a trade-off between having an analysis interaction dataset that is amenable to classifying insights versus having more "real-world" analyses in the dataset, in which the datasets vary and much more work is done. The goal of the studies that classified insights were to understand what kinds of interactions and interface features lead to certain insights, whereas our goal is to understand how to summarize lower-level activity into higher-level task descriptions. We discuss insight studies further in Section 2.4.

In a different study using logged interaction data, Zhang, et al., sought to identify different types of user personas [177]. A persona is an abstract archetypal user, used as a UX design tool, that describes a set of behaviors, goals, wants, needs, and frustrations. They gathered millions of clickstreams from users of a real-life data analysis product and clustered these into ten common workflow types. They then classified users into five different personas based on their mix of workflow types. This is an approach that we may have been able to take in our Splunk study; in our Tableau study, we did not have enough data for this approach. In another study, researchers demonstrated they could predict users' task performance and certain personality traits, such as locus of control, extroversion, and neuroticism, by applying machine learning techniques to interaction data from a visual search task [23].

2.3.2.2 Other Domains

While in Chapter 4 we use query logs to study *analysis* behavior, query logs are often used to study *search engine user* behavior. People have used search engine query logs to model semantic relationships [108], track user preferences [146], and identify information needs [125]. In addition to query logs, Guo and Agichtein proposed using fine-grained user interactions with the search results, such as mouse movements and scrolling, to detect users' search goals [57]. Waterson, et al., developed systems for gathering data to evaluate the usability of web pages [166]. This system also stores the URLs visited in a network diagram, in addition to metadata about those states. In Chapter 4, we only have access to actions, as shown in Figure 4.1, but in Chapter 5, we gather actions, states, and metadata about the data analyzed, as shown in Figure 5.1. Chi, et al., use user web surfing patterns to attempt to predict user needs or tasks and simulate usage patterns based on these [32]. Agichtein, et al., use implicit user feedback as revealed through their search behavior to improve search results [3]. Business process mining,uses data about business activities, often from logs, to learn models of the operational processes participants used in that activity [1]. Van der Aalst, et al., use process mining to analyze the invoices sent by the various subcontractors and suppliers of the Dutch National Public Works Department, responsible for the construction and maintenance of the road and water infrastructure [1].

Due to data quality issues discussed in Section 4.2, we could not analyze Splunk user sessions, but other aspects of our work parallel these techniques [8]. Collecting client-side user interactions is also a challenge in the many analysis applications which rely on a browser-based interface for visualization rather than a desktop application. Thus, employing some of the aforementioned techniques to collect examine data analysis activity logs is a promising avenue of future research. Going forward we expect that the study of human information seeking behavior will be enriched through the study of analysis query logs and fine-grained client-side interface interaction, much as the study of web search has been.

2.3.3 Interfaces

Logging and log analysis have been used not only to understand systems and users, but also to directly develop new user interfaces and improve the user experience. We discuss three main categories of interface features that are enabled through the logging of user activity:

- Exposing analysis provenance or workflow histories
- Automatically recommending possible analyses or visualizations
- Facilitating collaboration through the recording and sharing of analysis activity

Many of these features are found in what are known as intelligent interfaces. They are often proposed for use in interfaces for exploratory analysis and search, because tracking and sharing the work done so far, as well as providing further suggestions is particularly helpful in these contexts. Several of the systems we discuss below provide all three types of functionality, though many focus on just one aspect.

2.3.3.1 Workflow Histories and Analysis Provenance

In computer science, the term *provenance*, originating from archival and library science, commonly refers to the origins of a piece of data. In analysis and visualization applications, users transform data to understand it, and to obtain insights about the question that they gathered the data to answer. Thus, in analysis and visualization applications, provenance is also in some sense a record of the user's activity, which can be recorded in and obtained from logs. To help users better track, understand, and share their findings, researchers have proposed exposing provenance – such as what transformations the user applied to the data – within the interface of the analysis application. Such data is sometimes referred to as workflow histories, history models, insight provenance, or trails, to reflect that this data is a record of the analysis work a user did to reach insights. This work bears a particular relevance to ours when workflow histories are used to help understand users' tasks or when it involves segmenting the activity or state sequences into subsequences and summarizing them.

Kurlander presents a linear history for graphical editing tools, in which actions are displayed grouped together into segments, or panels, which are depicted graphically by a thumbnail of a representative state in that segment [89]. The segments are created via heuristics, like we do when segmenting Tableau event logs in Section 5.3.4, and can be used to create macros. We discuss the differences between our approaches and others in that section.

Amulet is a development environment for user interfaces that provide commands objects organized into a reusable command hierarchy [107]. In this system, lower-level user interactions invoke lower-level commands, which invoke higher-level commands, and at the highest, application-specific level, commands correspond to the semantic actions of the program. This is intended to provide better re-usability, but other projects have also found that logging semantic-level interface commands better reveals user behavior and intent than lower-level commands do. For example, HARVEST provides insight provenance by providing visualization widgets that can be re-used across applications and that support "semantic interaction" to help identify and capture user intent [50, 51]. Like in the previously discussed system, presenting widgets that are at this level of abstraction has the benefit of making it possible to capture tasks at the application-level rather than needing to code interactions post facto. Users analyze data in HARVEST by interacting with these widgets and HARVEST recommends visualizations to the user based on this analysis history. We discuss analysis and visualization recommendation later in Section 2.3.3.3. Researchers also used interaction data collected from HARVEST to create a taxonomy of analysis activity [50]; analysis task taxonomies are discussed in Section 2.4.

TOME is a framework by Gomez and Laidlaw for building models of user performance at quick, interactive UI tasks, such as searching for a place in a mapping application [47]. With this framework, interfaces can be implemented to allow for the collection of interface event and keystroke data. The authors aggregate interaction histories for a given task and choose the most common as the canonical approach which is then displayed as a storyboard. This storyboard can be manipulated by adding new event types to reflect new interface features; TOME can then predict the effect of these.

One recent novel approach to exposing provenance is, rather than exposing analysis history as a collection of past actions and/or states, revealing it in terms of the dimensions of the data that have been analyzed [138]. In a user study, these researchers found that participants with access to their dimension coverage information during their analysis formulated more questions, investigated more dimensions, and discovered more findings than participants without.

Heer, et al., deployed workflow history prototypes in Tableau, which is the commercial analysis product we use as the source of data for our case study in Chapter 5 [61]. They first performed a design study of various graphical history tools. They then created a history model for Tableau and used the data they collected for this to study user behavior. They use heuristics to group history into coarser-grained chunks, similar to how we do for our Tableau event data, described in Section 5.3.4. Their findings are discussed in the context of our findings in Section 5.3.5. They also created visualizations of this user activity. They found that using behavior graphs were the most useful for understanding user behavior, which contain some characteristics similar to our visualization, such as depicting application state and color-coding by action type. We did not try this approach, because the sessions we examined were generally several orders of magnitude longer, so we sought visualizations that could compactly display more events. However, it would be interesting to incorporate the graph aspect into our visualization as future work.

Lastly, some analysis systems can be thought of as partially incorporating analysis provenance information explicitly in their analyses specifications, particularly when these take the form of pipelines [17, 88, 51]. One such system was used to generate scientific visualization pipelines that the researchers later mined to automatically recommend visualizations [17, 88]. In our Splunk dataset (Chapter 4), analyses are also expressed via pipelines, called *queries*.

Other projects have applied history mechanisms to graphical editing [89, 103], network analysis [42], document editing [54], computational science [22], web design [86], and web browsing [15, 169, 74]. Reviewing all of the workflow history mechanisms in the literature is beyond the scope of this chapter, but Heer, et al., [61] and Sarvghad, et al., [138] provide good overviews of this space.

2.3.3.2 Collaborative Analysis

Another application of logging interaction data is found in tools for analyzing processes that are themselves collaborative. Collaboration efforts are aided by sharing the analyses that individual analysts have performed, so that others can understand and build upon that analysis. While there are many collaboration tools, here we provide just a few examples for collaborative data analysis tools.

ManyEyes is a public website for collaborative visual analytics, where users can create, share, and discuss visualizations [159]. Willet, et al., created an analysis system that lets users outsource parts of their exploratory analysis to paid crowd workers and then interactively examine and build upon these workers results [168]. DataHub is another platform for collaborative data analysis [19]. Rcloud is a collaborative analysis environment for R programs that provides an overview where users can browse popular and recent analyses and supports search, annotation, and recommendation over all analyses [111]. Lastly, Sarvghad, et al., create an analysis tool in which the dimensions analysed by the user are recorded and shared with collaborators, so that analysts can see which data dimensions have been explored and how. They find that presenting this information reduces the amount of time it takes collaborators to find and investigate unexplored subsets of the data [137].

2.3.3.3 Automatic Analysis or Visualization Generation

Logged activity records can be used as the basis for automatically generating analyses and visualization recommendations, though these recommendations can also be generated via other means. Some projects, like the Gene Ontology [46], the Sloan Digital Sky Survey [147], or the Aqua Data Studio IDE and workspace [12] maintain query repositories with example queries, or query logs, where users can search for related queries. Aside from basing recommendations on records past user activity, recommendations can be based on predetermined principles about which visualizations are appropriate for various types of data and tasks, on information about the data and visualization types stored in ontologies, on measures of how interesting the visualization is, or other statistics, such as similarity metrics, or on user preferences or feedback. Here we review the different types of systems organized according to the basis upon which they make their recommendations.

Based on the user task: Casner automatically designed graphic presentations based on an analysis of a logical description of a task the user was attempting to undertake [27]. ADE, or Active Data Environment, is a system that recommends data for the user to analyze based on inferring the user's current analytic tasks from their interactions with the tool [35]. The set of possible tasks was predetermined by the prototype developers and the mapping between interactions and possible corresponding tasks maintained by hand. Our work could be used in systems like ADE to replace the manual task modeling with the automatic identification of tasks and the interactions that imply them.

Based on analysis and visualization principles: SAGE is part of a collection of tools for constructing graphical designs by selecting graphical elements, mapping them to data, combining them, and for finding and customizing prior designs [132]. Users of SAGE can partially sketch out designs and SAGE infers the remaining properties based on principles of which visualizations are appropriate for the given data and task specified by the user [131]. In addition, SAGE provides users the ability to browse and search related designs based on criteria such as the graphical elements used and the data types of the data mapped to these graphical elements. Schiff developed a method to automatically create cognitively motivated visualizations from first principles using very small data sets [139]. St. Amant and Cohen developed a knowledge-based planning system called AIDE to help users strike a balance between conventional statistical packages and automated systems which do not ask the user for any kind of input [9]. Tableau includes a "Show Me" feature, which is meant to reduce the burden on users by automatically creating a good visualization of a view of their data [98]. The user selects a tuple of columns, and Tableau returns a list of possible visualizations templates that could be used with these columns, by suggesting all visualizations that are compatible with the selected columns' types. Profiler is a tool for data quality assessment, a common precursor to data analysis, which involves finding missing values, outliers, and misspellings [76]. It recommends views of the data based on statistics regarding the distribution of and the presence of anomalies in the dataset.

Based on user feedback: VizDeck also aims to automatically suggests visualizations for exploratory data analysis [68, 81]. It generates all possible visualizations for a given data set from a set of seven possible chart types, then ranks them by predicting how interesting the user will find them, based on the user's other feedback. DataPlay auto-corrects SQL queries based on user feedback about which tuples to keep or drop from the answers and non-answers [2]. VizRec uses a combination of rules like the above systems, as well as user preferences elicited crowd-sourcing experiment on Amazon Mechanical Turk [106].

Based on how interesting the visualization is: Some researchers developed a tool for refor-

matting messy input data for use in downstream analysis tools that proactively suggests data transforms according to a metric that scores tables according to type homogeneity, sparsity and the presence of delimiters [56]. They found their top-ranked suggestions is preferred by the user 77% of the time. YmalDB is a system for recommending SQL queries that have results similar to those of a user's original query [41]. It does not assume the existence of query logs upon which to base its recommendations, but rather, computes recommendations based on the most interesting sets of (attribute, value) pairs from the original query, using a metric that depends on the frequency of the pair in the query and in the database. SeeDB generates recommended visualizations for the results of SQL queries also based on interestingness, but uses a slightly different metric that measures how much the visualization technique applied to the current result set differs from the visualization applied to the entire database [118]. Atlas is a database interface, designed for data exploration, that responds to each user database query with a set of automatically suggested "data maps" that each describe the results of that query clustered by various attributes [141].
Based on annotated ontologies: Other researchers have used a formalized, annotated ontology of data and visualizations to generate context-aware visualization recommendations [163]. In a prototype called VizBoard, they then expanded upon this work by incorporating explicit user feedback about these visualizations in the recommendation system [161, 162]. Other authors have also used ontologies to enumerate valid data mining processes [18].
Based on logged analysis activity: In the Lumière project at Microsoft, researchers developed a Bayesian model of user behavior, trained on logged event data from the spreadsheet program Excel [67]. This model was developed to drive an intelligent user assistant within Excel that would reason about the user's goals, identify when the user needed help achieving those goals with the software, and then provide hints to guide the user to possible solutions. We discuss some of the challenges that the Lumière team faced in using logged event data in Chapter 3.

The Systematic Yet Flexible (SYF) architecture is a framework for supporting exploratory data analysis in a variety of analysis applications by tracking the user's analysis process, suggesting unexplored states, allowing the user to annotate useful states, and enabling the reuse of successful analysis strategies [120]. This system is an example of one that both exposes analysis provenance to the user, as well as provides suggestions; it also enables collaboration, which is an application discussed in the next section (2.3.3.2).

VisComplete suggests analysis visualization pipelines based on correspondences between existing pipeline subgraphs in a database of recorded user visualization pipelines [17, 88]. Yang, et al., recommend new query joins by looking for new ways to combine or extend previously-issued query joins from the database query log [173]. SnipSuggest also suggests possible additions to queries using relevant snippets collected from a log of past queries, except it applies to SQL queries rather than visualization pipelines [84].

The HARVEST system described in the previous section (2.3.3.1) generates recommendations by returning the most similar visualizations to the current visualization from a database of example visualizations [51]. Wrangler is a tool for facilitating data reformatting, which is a common precursor to data analysis [77]. It defines a set of data cleaning transformations, and in response to user interaction, it provides some guesses as to which of these transformations the user might want, based on the relative frequency of transformations selected before, as recorded in a corpus of usage statistics. Smart Query Browser is a tool for browsing SQL query logs of large databases that returns query sessions instead of single queries in response to user query terms, though it is unclear how they match queries [83]. QueRIE is another system that recommends SQL queries from a log of the user's past queries using one of two possible metrics – one based on the similarity of tuples returned and the other based on the similarity of syntactical features [30]. The latter is most relevant to our work in Chapter 4 because we use some syntactical features to group similar queries, whereas in Chapter 5, we group consecutive parts of the event log based on the visualization operated upon, which is usually related to the same data items, so this is more relevant to the former metric.

Marcel and Negre provide a 2011 survey of query recommendation technique for databases, and point out that many of the recommendation algorithms they survey, including those discussed above, have not been assessed with real users and use cases [101]. Frequent Itemsets Mining for Interactive OLAP Query Recommandation (FIMIOQR) is a query autocompletion framework that extracts query fragments, or elements, from the database query log, and constructs frequent itemsets from these as the basis of recommendation [82]. In other words, it recommends query fragments that most frequently accompany the current partial query. In Remix, a business intelligence reporting prototype, researchers implemented a system for recommending similar SQL queries to users from a database of all recorded report queries, based on an index-based query matching approach [152]. This system recommends queries that involve similar data content to the queries issued by the user, rather than focusing on syntactical similarity. Aufare, et al., recommend similar queries to database users by first clustering their past queries, representing the transitions between clusters as a Markov model, determining which cluster the current user query belongs to, and then suggesting a query from the cluster following it in the model [14]. This work bears strong resemblance to ours in that we also cluster Splunk queries based on similarity in Chapter 4, and model transitions between queries as a Markov model in both Chapter 4 and 5. Two differences are that they use a similarity metric based on the overlap of the parameters passed to the SQL queries [4], whereas our metric is per-command type, and based on similarity in parameters features specific to the Splunk query in question.

RapidMiner is a software framework for developing analysis workflows that can recommend additional operations to add on to the current workflow, using a prediction model that is trained on a set of thousands of real-world data analysis workflows [71].

In addition to recommendation systems, there are also autocomplete systems that recommend query or pipeline completions The Wolfram Predictive Interface provides both autocomplete suggestions as well as suggestions for the next step to take, though it is unclear what this technology is based upon [129].

2.4 Tasks and Models

In this dissertation, we seek to empirically identify analysis tasks represented in our datasets of activity records. There has been considerable work in the area of identifying, modeling, and taxonomizing user tasks, particularly with respect to analysis activity. The taxonomies of analysis activity are not limited to visual analysis, which is our focus, but also include scientific data analysis and intelligence analysis. There are also taxonomies for the somewhat related task of web search, but we omit those here.

Different taxonomies break down the space of activity along different dimensions, such as by datatype, visual representation, task, interaction, or other. Nonetheless, there is a fair amount of overlap between taxonomies as well as the activity we observed in both of our case studies. The analysis task taxonomies are the most relevant to our work, with visual interaction and representation taxonomies being somewhat less relevant. Though the focus of this dissertation is on methods of identifying tasks represented in log data, one interesting avenue for future work might be to compare the tasks we identified to each of the taxonomies and activity models in the literature, though some are at a higher or lower level of detail than we identify. Here we briefly describe some of the taxonomies and activity models that have been proposed.

GOMS is a model, or family of analysis techniques, proposed by Card for predicting and explaining user task performance in general, not specific to analysis tasks [26]. GOMS stands for goals (what the user wants to do), operators (the means available to the user for accomplishing these goals), methods (sequences of operators and subgoals), and selection (the rules for choosing among different methods). Researchers in Project Ernestine used GOMS in conjunction with user activity records to evaluate different workstations for telephone company toll and assistance operators [53]. Paterno has a book chapter describing different approaches to task modeling, including GOMS, as well as others [119]. Also at a more generic level not specific to analysis applications, Hoppe proposed determining the current user task by creating a parser that maps different tasks to different sequences of user interactions [65]. He introduces a formalism for the representation of tasks, represented in a Prolog program, and implements an example application of this for a UNIX-like file management system. Tasks in his framework must be predefined in order to be detected. Work such as ours in detecting user tasks post-facto could be coupled with such a parser that operates in real-time to detect user task in intelligent interfaces.

With respect to information visualization tasks specifically, Card describes an abstract cycle of data transformation, visualization, and rendering [25]. Likewise, the Application Visual System (AVS) also envisions the scientific visualization process as an iterative cycle [157]. In a study by Russell, et al., on the cost structure of sensemaking, the authors likewise identify a recurring cyclical structure of sensemaking, which they refer to as the "learning loop complex" [133]. Later, Card in collaboration with Pirolli conducted an observational study using a think-aloud protocol to characterize the sensemaking, or intelligence analysis process, at a high level, as another loop [124]. At this level, sensemaking shares many commonalities with visual analytics; however the elements of the loops identified in the work above tend to be at a higher-level of abstraction than is our focus. Also, because we did not have sufficient information to sessionize the queries in our Chapter 4 case study, we could not identify loops there, but the participants in the Chapter 5 often followed an iterative process, at least in part because of analysis instructions given to participants.

Shneiderman proposed seven task and seven datatype taxonomy for visual analysis that shares many parallels with the tasks we discover in Chapter 4 [143]. He also poses a Visual Information Seeking Mantra, "Overview first, zoom and filter, then details-on-demand", that participants who contributed to the dataset we study in Chapter 5 learned about prior to performing their analyses. Springmeyer, et al., propose a taxonomy of scientific data analysis tasks based on an observational study [149]. This taxonomy is most similar to the activities we observe in Chapter 5, though again at a more fine-grained level than we have yet analyzed.

The GRASPAC project discussed in Section 2.3.3.1 decomposed visualization tasks into data sources (generated by a computational process that can be adjusted and rerun), filters, mappers, and renderers [22]. This is a more implementation-oriented view of visualization than is reflected in the records we collected; moreover it includes the data generation portion of the process, which was often not part of the analysis process for participants in our datasets, and wouldn't have been captured in any case.

Buja, et al., describe a rudimentary interactive high-dimensional data analysis taxonomy and discuss how this relates to features of a tool called XGobi [24]. This taxonomy is more relevant to Chapter 5 than to Chapter 4 because we were not able to collect much visualization information in the latter case. Chuah and Roth describe a very-low level hierarchical taxonomy of visual analysis tasks that has much overlap with Tableau functionality, though we describe tasks at a higher level of granularity in our study [34].

Dix and Ellis provide a taxonomy of interaction techniques for visualization, as part of their position paper arguing for adding interaction to visualizations to improve their utility, in order to make it easier to match interaction techniques with the task at hand [39]. Lee and Grinstein present an architecture for uniting database and visualization systems that records user interactions – queries, query results, and the associated visualization – so that the process by which a database is explored can be analyzed [93, 92]. The authors use a graph-like structure to model the process in which vertices are the visualization states and the edges are relationships between them that are based on similarities between the states. They instantiated their approach in a system called Exbase and from that learned of two common exploration patterns, panning and zooming; however, it's unclear how many users they gathered this exploration data from, or in what context.

Zhou and Feiner synthesized a taxonomy from several other research papers that connects high-level tasks to low-level visualization techniques, with the ultimate goal of facilitating automatic visualization design (research on which was discussed in Section 2.3.3.3) [178]. Similar to work discussed in this section and Section 2.3.3.1, this work concerns itself with identifying visual activity at different levels of abstraction or hierarchy [107, 50, 51]. Keim presents a taxonomy for visual exploration along the axes of data, visualization, and interaction, which he derives from first-hand knowledge [80]. This taxonomy is much broader than our use cases, applying to many different types of data. Andrienko, et al., create a taxonomy for visual analysis of spatiotemporal data in terms of what type of data topology the analysis techniques are for and what exploratory tasks they support [11]. Wilkinson at al. proposed a visual taxonomy in their "Grammar of Graphics" for describing and composing visualizations [167]. Amar, et al., construct visual analytic taxonomies organized around the user's task, rather than around visual representations, by collecting almost 200 potential analysis questions posed by students about five datasets from different domains [10]. Yi, et al., provide a taxonomy of visual interactions, as opposed to tasks, but they did consider task in their compilation [174]. They did this via clustering techniques identified via a literature review. Yi, et al., cite several other taxonomies of interactions types and modes, but these are less relevant to our task-oriented work [156, 148, 165]. Jankun-Kelly, et al., have proposed a formal model for representing the visualization and exploration process as a set of parameters describing a visualization, with actions representing a transformation of these parameters [70].

The creators of the visual analytics system HARVEST described in Section 2.3.3.1 developed an action and task taxonomy as the basis of the operators provided in their tool, so that user tasks could be captured at the application-level rather than having to code user interactions post facto [50]. Heer and Shneiderman's taxonomy has 12 tasks, four in each of three categories: data and view specification, view manipulation, and process and provenance. These map well to the tasks we identify in Chapter 5. Lastly, Shulz, et al., reviewed all of this literature on visualization task taxonomies and created a visual design space in which to place them, with dimensions of the design space being task goal, task methods, data characteristics, data target and cardinality, task order, and type of user [140]. This work is an important survey of the research on task taxonomies.

Closely related to the notion of tasks is the notion of insights, which we have mentioned before in Section 2.3.3.1 without defining. Yi, et al., conduct a literature review to attempt to define the term "insight" in the context of analysis, and offer a high-level taxonomy of the means of reaching insights [175]. Chang, et al., similarly attempt to define "insight," arguing for a dual definition as both a moment of enlightenment as well as a piece of information [28]. North attempts to define the term "insight" and argues for expanding beyond simple benchmark tasks for evaluating how well visualization tools help users reach insights in their analysis [110]. In a later study, North and collaborators compare the simple benchmark task evaluation method with their broader, more qualitative insight evaluation, and find that the latter confirms the results of the former while also illuminates additional types of tasks [109]. This follows on earlier work by Saraiya, et al., that had quantified insights to determine which bioinformatics analysis tools were more suitable for different analysis tasks [135]. Other studies involving the coding of insights for evaluating visualizations have been conducted by Vande Moere et al [105] and Liu and Heer [94]. The difficultly in precisely defining insight is similar to that for defining "task." This is complicated by the fact that these are usually colloquial and not exact technical terms and they can reference differing levels of abstraction. Furthermore coding both tasks and insights is a subjective process. For this reason, in this dissertation, we don't claim to provide the perfect definition of task; rather we rely on a simple definition supplemented by judges' colloquial understandings in conjunction with concrete, empirical examples derived from our datasets.

2.5 Summary

Here we summarize the main differences between the work discussed in this chapter and our work presented Chapters 4 and 5 of this dissertation. The work perhaps most similar to ours is described in Section 2.3.2.1, on using logged interaction data to understand the behavior of users of analysis tools. Some specific differences and similarities between the most related pieces of work in that section and ours are:

- In several of the studies, the analyses that participants completed were designed by the authors of the studies, meaning they were more limited in both time, scope, and data analyzed [73, 40, 21, 127, 55], whereas in both of our case studies, users analyzed whatever dataset they wanted, to investigate whatever questions they wanted, for however long they wanted.

- In part due to the larger scope and real-world setting of the analyses reflected in the data we collected, we did less manual coding of user activity and no coding at all of user insights or findings, as several other groups of researchers did [73, 40, 21, 127, 55]. We did not have access to insights at all for our Splunk case study in Chapter 4, and we did not attempt to relate these to user tasks in our Tableau case study in Chapter 5, instead, focusing on user actions.

- We focus on summarizing lower-level log data to identify higher-level user tasks from the data, rather than summarize it for another purpose, such as testing whether analysts can reconstruct user activity from the logs [73, 40], comparing different analysis techniques or interface features [21, 127, 55], or presenting workflow history summaries to users [60].

- Guo, et al., examined frequently recurring patterns [55]. We also did this in our Splunk case study in Chapter 4, but not in our Tableau case study in Chapter 5, for reasons we describe later in Chapter 5.

- We are similar to two other studies in the presention of transition frequencies between different types of user activity [55], though some incorporated additional information about mental states [127].

- Much of the related work presented visualizations of the log data they examined to help them make sense of that data, as we did. Some of the visualizations bore some commonalities to ours, for example in the presentation of events [21, 55], or the use of transition diagrams [127, 55], whereas others were more specific to the analysis applications in their study [73, 40], or used techniques we did not adopt, like treemaps [73, 40].

- Almost all of the other studies were conducted on different analysis systems from those in our case studies, and many of their findings aren't comparable to ours for that reason, with the exception of work done in the context of Tableau [60]. Like in these studies, limited applicability to analysis systems that are reasonably different from those we include in our case studies is also one possible limitation of our own task descriptions.

Lastly, here we summarize the relationship between our work and the other related work sections. Section 2.2 describes tools and techniques for log analysis for various purposes, as well as presents some other means of understanding user behavior, for contrast. Most of these techniques differ from our approaches, which are detailed in Chapters 4 and Chapter 5, and are thus part of the novelty of our contribution. Section 2.3 describes applications of log analysis. Section 2.3.1 briefly describes its application to computer systems, Section 2.3.2 to understanding user behavior, and Section 2.3.3 to interfaces. Section 2.3.3.1 describes interfaces that present user interaction data from logs for a variety of types of systems, not limited to analysis and visualization systems, to present users with a record of their activity, rather than to identify higher-level user tasks, as we do. Similarly, the work in Section 2.3.3.2 describes the application of user interaction data in collaboration tools where activity records are used to share progress with collaborators. We describe this work because presenting records of user activity to users is related to our work of examining records of user activity to identify their higher-level tasks. Section 2.3.3.3 describes research on generating analysis and visualization recommendations to users, which can be done using logged interaction data, among other means – our work could feed into such systems in that the tasks we identify could be used to inform the recommendations made. Lastly, Section 2.4 describes analysis task taxonomies – we also identify sets of analysis tasks, but we identify tasks in a bottom-up fashion from the logged interaction data, whereas the taxonomies in the related work are generally created top-down via the authors' personal understanding, sometimes synthesized with information from other published task taxonomies and other literature.

Chapter 3

Challenges Studying Use Via Logs

Despite longstanding research interest in data exploration and automation, there is a scarcity of automatically logged, high-quality activity records of data exploration activities at an appropriate level of granularity, available for study by researchers and developers. In our experience, one reason is that much of the data logged by data analysis systems is intended for the purpose of debugging and system performance monitoring, not for understanding user behavior [50, 67, 128]. As Horvitz, et al., note in their paper on Bayesian user modeling for the Lumière project, whose goal was to build an automated assistant in Excel, "...it is critical to gain access to a stream of user actions. Unfortunately, systems and applications have not been written with an eye to user modeling" [67]. Many of the most popular data analysis systems, like Tableau, are closed-source, with the implication being that outside researchers can't instrument them. Without access to these closed-source code bases, one alternative is for researchers to build their own tools. This is a long-term endeavor and comes with its own challenges. As a result, much effort has been devoted to devising ways to extract useful usability information from UI events, as reviewed by Hilbert and Redmiles in their extensive survey on the topic [64], and as we discuss in Section 2.3.2.1.

As an alternative to analyzing automatically logged user actions, much of the research on understanding and improving how users analyze data relies either on author intuition born of first-hand experience [18, 27, 77, 9] or on observational studies using manually-recorded and synthesized information that is hard to share and compare [78, 79]. This lack of automatically logged activity records hinders research into improving tools for data exploration and analysis.

Currently, researchers have few good options for using logs to understand what users are doing, build user models to improve interfaces, yield predictions about user actions, and make recommendations to users. One option is to take great pains to extract high-level information out of low-level event logs, as discussed in Section 3.3 [50, 64, 67, 128]. Another option is to manually observe user behavior, interview experts, read through the literature, and synthesize all of their observations "by hand," then encode this synthesized information into tools [160, 77, 97, 120].

If user actions were instead encoded in a machine-digestible format at an appropriate level of granularity, researchers could create software to automatically detect these patterns, much like clickstream analysis and webmining [13, 44, 57, 150]. For example, HARVEST is a visual analytics system that exposes interface elements for performing tasks that directly correspond to "semantic actions," like bookmark, brush, filter, and sort. In other words, the interface is designed so that the actions the researchers are interested in tracking must be explicitly indicated by the user. This, however, is not the norm in most interfaces. As Hilbert and Redmiles conclude, "our understanding of the nature of UI events . . . leads us to conclude that more work will likely be needed in the area of transforming the 'raw' data generated by such event-based systems in preparation for other types of analysis in order to increase the likelihood of useful results. . . . Thus, we conclude that more work is needed in the area of transformation and data collection to ensure that useful information can be captured in the first place, before automated analysis techniques, such as those surveyed above, can be expected to yield meaningful results (where 'meaningful' means the results can be related, without undue hardship, to aspects of the user interface and application being studied as well as users' actions at higher levels of abstraction than simple key presses and mouse clicks)" [64].

This chapter identifies specific challenges in logging data to enable the study of user behavior in visual analysis systems, and includes some recommendations for better practices to ameliorate these challenges. These challenges and recommendations are based primarily on our experiences conducting the case studies described in Chapters 4 and 5, but also on a review of the literature and conversations with industry personnel [1]. The challenges we identify are as follows:

- Dealing with missing and heterogeneous information
- Tracking interaction provenance
- Obtaining high-level user actions
- Observing intermediate user actions
- Tracking analysis state
- Obtaining information about analyzed data
- Determining user intent

We review these challenges in the following sections. We illustrate each with examples from three systems: Splunk, an enterprise log management and analysis platform, and the system in our Chapter 4 case study, Tableau, an interactive data visualization product, and the system in our Chapter 5 case study, and Excel, a widely popular spreadsheet application.

3.1 Coping with Missing and Heterogeneous Information

Logging is important but challenging, because while it can provide crucial information about a program's execution, it usually depends on the intuition of developers to decide which in-

[1]Our interviews with industry personnel will be written up at a later date.

formation is important to log. Developers do not always foresee all possible desired uses of the logs. Most often, they log information primarily for debugging and performance monitoring purposes. It is less common to log information with an eye towards reconstructing and understanding user behavior. This makes it difficult to analyze for this purpose, because often needed information is missing. For example, in our Splunk case study in Chapter 4, inconsistency in the description of how a query was issued made it difficult to sessionize queries, so we were not able to do so for our study; we discuss this in more depth in Section 3.2. Similarly, in pilots for our Tableau case study in Chapter 5, we found that certain crucial events, or event parameters, were not being logged, which made it impossible to reconstruct what the user had done; for example, not logging when the user switches sheets, or logging that a sheet was deleted but not which sheet it was. To address this, we made several recommendations to Tableau to help them improve their logs and allow us to collect better data for our main study.

In addition to the above challenge described, a 2012 study of logging practices in four large open-source software projects found that: "Despite the importance of log messages, logging is often a subjective and arbitrary practice in reality. In general, there is no rigorous specification or systematic process on software logging, partially because logging is seldom a core feature provided by the vendors. Therefore, in many cases, log messages are written as 'after-thoughts' after a failure happens and logs are needed. There are a number of logging libraries . . . But even with them, it is still the developers' arbitrary decisions on when, what and where to log" [176]. This arbitrariness could also have been the source of problems like those we encountered, described in the previous paragraph. Also, due to this arbitrariness, different systems often take different approaches to logging, following different formats and using different abstractions, making it difficult to reuse analyses code from system to system.

Contributing to the above challenges is the fact that designing good logging infrastructure can be a challenging software engineering problem. To provide good logs, it helps to identify downstream use cases and design for these accordingly. It is also necessary to ensure that the logging code not be too brittle or messily intertwined with the application code, so that it can evolve along with the application without breaking. Lastly, a difficult design decision is to determine what level of abstraction activity should be logged at, which also involves the design of the application itself. The creators of the visual analytics system HARVEST described in Section 2.3.3.1 developed an action and task taxonomy as the basis of the operators provided in their tool, so that user tasks could be captured via application-level logging, rather than having to code user interactions from low-level log events post facto [50]. In their work on application logging, Hilbert and Redmiles, propose "(a) treating 'usage expectations' explicitly in the development process to improve design and focus data collection, (b) a flexible and incrementally evolvable monitoring architecture that separates evolution of monitoring code from evolution of application code, and (c) event abstraction mechanisms embedded within probes to provide distributed filtering and multiple levels of abstraction in collected data" [63].

Lastly, another challenge of attempting to understand user behavior is that often users rely on a number of different applications to do their work, and it can be difficult to combine log data from these different systems. In our work, we did not attempt to address this problem, and instead ignore any activity that occurs outside the target application. However, for future research, efforts should be made where possible to log information that facilitates combination with and comparison to other systems. This was the focus of the GlassBox project, a system for capturing and leveraging user activity across applications [36, 37]. This would allow researchers and developers to understand what functionality is missing from existing tools, better characterize portions of the analysis pipeline that are currently less well understood, such as exploration, and identify how best to integrate new data mining techniques into existing workflows.

Ideally, there will one day be cross-system instrumentation that would allow researchers to understand the entire ecosystem of data analysis tools and visualization and the roles they play in users' daily workflows. To accomplish this, it may benefit the community to develop an open standard for logging, similar to the Common Information Model, which defines a way for objects in an IT environment and relationships between them to be described so as to facilitate management of systems, networks, applications and services independent of manufacturer or provider [38].

Lee and his collaborators proposed a model for storing user interaction data as a graph-like structure to model the process in which vertices are the visualization states and the edges are relationships between them that are based on similarities between the states [93, 92]. Similarly, Jankun-Kelly, et al., have proposed a formal model for representing the visualization and exploration process as a set of parameters describing a visualization, with actions representing a transformation of these parameters [70]. Examples of standard schemas in other domains that may serve as inspiration includes IEC61850 for electric grids and SensorML for sensor data [69, 142].

In summary, dealing with missing and heterogeneous information in logs makes it challenging to understand user behavior in analysis systems using these logs for the following reasons:

- Data is usually logged for purposes other than understanding user behavior and it is difficult to re-purpose because often needed information is omitted, or presented at a difficult-to-use level of abstraction,

- The information that is logged is sometimes subjective and arbitrary decision, not based on a coherent design plan, resulting in missing or erroneous information,

- Logging practices, formats, and abstractions vary from product to product, so that analyses have to be adapted or recreated for each different system,

- Logging is an inherently challenging software engineering problem especially in large software systems, in that it can be difficult to design a logging system that is modular and can flexibly evolve with the rest of the application code, leading again to quality issues, and

```
09-28-2012 18:28:01.134 -0700 INFO AuditLogger - Audit:[timestamp=09-28-2012
18:28:01.134, user=splunk-system-user, action=search, info=granted ,
search_id=‘scheduler__nobody__testing__RMD56569fcf2f137b840_at_1348882080_101256’,
search=‘search index=_internal metrics per_sourcetype_thruput | head 100’,
autojoin=‘1’, buckets=0, ttl=120, max_count=500000, maxtime=8640000,
enable_lookups=‘1’, extra_fields=‘’, apiStartTime=‘ZERO_TIME’, apiEndTime=‘Fri Sep
28 18:28:00 2012’, savedsearch_name=‘‘sample scheduled search for dashboards (existing
job case)’’]
```

Table 3.1: Example of an event logged by Splunk. An example query appears as the value to the `search` key in the query. Queries represent useful records of user activity. These queries are written in the Splunk query language, modeled after UNIX pipes and utilities.

- Logging rarely spans multiple applications, even though users' activity does, and it can be challenging to combine interaction data from multiple applications into a coherent whole.

3.2 Tracking Interaction Provenance

Information about what interaction triggered an event, and possibly the path through the application logic that was executed to reach the point at which the event was ultimately issued, should be logged along with the event itself. This information reflects the event's provenance or point of origination. The following real-life anecdote demonstrates the importance of this.

Splunk records an event every time it executes a query, as shown in Table 3.1. The query, highlighted in bold, is an important source of information about the analysis actions users perform on their data.

Some of these queries may be written interactively by a human user. The user may issue these queries by typing it into the Splunk shell at their command line interface, or by typing it into the search bar in Splunk's in-browser GUI, or by hitting a button in an application that then issues the query to Splunk's back-end, or by loading a web page in a browser-based dashboard that triggered the query to execute. Other queries may be written by a user once, but then set up by that user to execute programmatically, via a script, for instance. Still other queries may have been issued from system code to fulfill some function of the system, and were written by a programmer of the system, not by a human user of that system. (Usually in this case, the user will be indicated to be the system, as in the example given above.)

To correctly model a user's cognitive state and extract artifacts like user sessions, it is critical that each time a query is logged, it is possible to easily and unambiguously recover this origination or application context information. Otherwise, the record of what actions a user took may be tainted with actions that they did not actually take, or actions that a user actually took may be inadvertently discarded. Our first-hand experience attests that it is extremely difficult to recover this information if it is not captured when the data is originally logged. Statistical learning techniques like clustering and classification can aid in probabilistically identifying origin information post facto, but rather than relying on such techniques, it is better to plan ahead and capture this information when the data is initially logged.

As another example, in Tableau, if a user creates a scatter plot by dragging a variable onto a shelf and receiving the default view (as opposed to selecting a scatter plot from the *Show Me Alternatives* menu), the log should record not only that the user created a scatter plot but also how the user created the scatter plot i.e., the provenance. Moreover, it should be noted, as Heer, et al., do, that different combinations of fields and visualization parameters can of can often result in the same visualization views [60]. Different sequences can also result in identical visualizations, but we discuss this issue further in 3.3. However, even when efforts to log provenance are made, we have observed in our work with Splunk logs that sometimes users perform actions that (unintentionally) circumvent efforts to accurately model their behavior, for example, by writing external scripts to interact with a browser-based GUI (i.e., a bot).

3.3 Obtaining High-Level User Actions

As Hilbert and Redmiles explain, one challenge of working with log data to understand user behavior is that low-level, high-frequency events do not, in themselves, reveal how they combine to form higher-level, lower frequency events [64]. Conversely, high-level events do not generally reveal what low-level events they are composed of. "As a result, either low frequency band events must be recorded in conjunction with high frequency band events or there must be some external model (e.g., a grammar) to describe how high frequency events combine to form lower frequency events." Similarly, Horvitz, et al., state, "a critical problem in developing probabilistic and decision-theoretic enhancements for user interface applications is establishing a link between user actions and system events."

Thus, it is important not only to log the actions that happen in the system but the actions that the user takes. In other words, log the operations that are applied to the data at the level of the user's perspective, not just the executed code that the user's command calls. To model user behavior and cognition, we are primarily interested in the former, but the systems engineers who typically write the logging code are primarily interested in the latter.

In some cases, this may be a conceptually simple fix. For example, Tableau had functionality "called *Show Me Alternatives* and *Show Me*, which are respectively a dialog of commands that automatically build views from scratch and a button that is a shortcut to the default choice for the dialog" [98].[2] In a paper describing this functionality, the authors of Tableau have a discussion about their efforts to evaluate their interface using the Tableau logs. They note that, "the log files do not differentiate between *Show Me* and *Show*

[2] *Show Me Alternatives* is no longer part of Tableau.

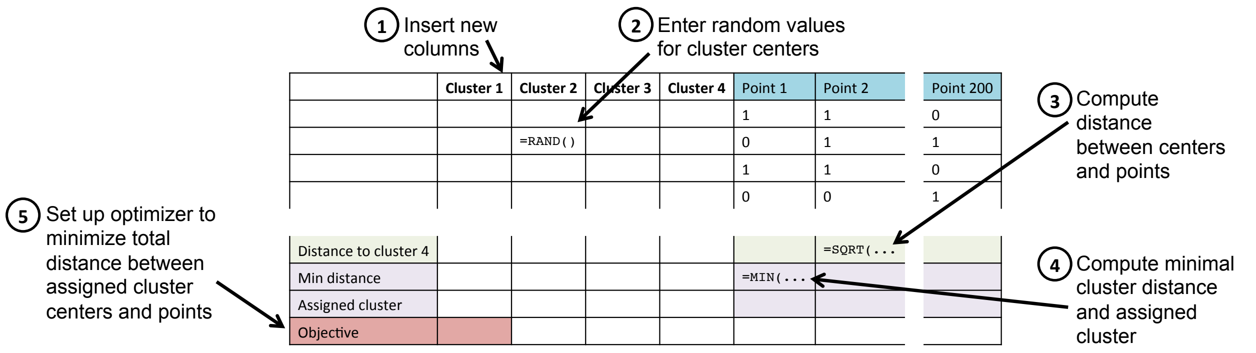

1 Insert new columns
2 Enter random values for cluster centers
3 Compute distance between centers and points
4 Compute minimal cluster distance and assigned cluster
5 Set up optimizer to minimize total distance between assigned cluster centers and points
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Point 1
Point 2
Point 200
=RAND()
Distance to cluster 4
Min distance
Assigned cluster
Objective
=SQRT(...
=MIN(...

Figure 3.1: Shown above are the steps a user takes to perform k-means clustering on their data in Excel. Ideally, the interface would be designed to easily capture the user's task, which is clustering. Barring that, the task may have to be inferred from the sequence of user actions, shown here.

Me Alternatives. These commands are implemented with the same code and the log entry is generated when the command is successfully executed." In other words, two different functionalities are described with the same logging code. This exactly describes the problem with logging events from the perspective of what the system executes versus from the perspective of the user taking the action. It complicates the work of trying to understand how users are interacting with a tool and especially complicates trying to build statistical models of user behavior. For example, the authors of the Lumière project, whose goal was to build an automated assistant in Excel, found transforming system events into modeled events to be quite challenging [67]. To establish the link between lower-level atomic events and the higher-level semantics of user actions, they built a special events system to analyze the atomic event streams and map them into higher-level observations. If higher-level actions had been logged, they may have been able to avoid this trouble.

Figure 3.1 and Figure 3.2 illustrate this problem graphically, using Excel as an example. Figure 3.1 shows the steps required to perform k-means clustering on data in Excel from a user's perspective [43]. The user may be doing this to segment their customer base into different markets for the purpose of releasing targeted advertising. Market segmentation is the user's intention, as shown in the uppermost level of Figure 3.2. Ideally, we could record that the user is performing clustering – this is the user's task, shown in the second level. However, in reality, the best information we can capture is the stream of the user's activities in the GUI. This is the third level of information shown in Figure 3.2. In practice, the information captured tends to be low-level system events – the lowest level of Figure 3.2. Using events from a lower-level to infer what action is being taken at the higher-level can be painful if insufficient information is recorded. It may rely on human-defined rules or on statistical inference, as in the events system of the Lumière project.

As mentioned in the introduction, HARVEST is a visual analytics system specifically designed to deal with this problem and capture the provenance of "insights" by exposing interface elements for performing tasks that directly correspond to "semantic actions," like bookmark, brush, filter, and sort [50]. In this system, the actions the researchers are interested in tracking are made explicit in the interface, so that the logs contain a higher-level record of user activity that require less processing and summarization to understand. Contrast this to a system that allows the user to select a view with many potential elements of interest at once, so that it is difficult to discern what the user most wanted, or to a system that requires the user to achieve their task through several direct but low-level actions like clicking and dragging, as in the example above. Designing the interface to facilitate easier capturing of high-level user behavior is a useful approach that should be considered wherever possible.

If this is not possible or desirable for a particular application, researchers can rely on techniques for combining sequences of events into high-level tasks [128]. For example, if the system was designed such that the user interacts with it at a relatively low-level (such as clicking and dragging, or entering functions into an interactive shell), it will not be possible to log the user's high-level tasks because those will not be reflected in their activity stream. In this case, it is important to keep the end use case of the interaction data in mind while developing the application and the logging code. Ideally, tool builders would design the system that identifies from low-level events the high-level tasks employed in user models in tandem with the application, rather than as an afterthought.

Alternatively, to deal with this problem, researchers have investigated ways to detect sequences, chunks, or segments of low-level events that correspond to some higher-level activity. Hilbert and Redmiles describe several techniques from the literature, but all of the techniques they describe depend on defining the high-level sequences in advance, so we discuss these only briefly in Section 2.2.2 [64]. Defining sequences in advance is hard to do because it can be difficult to define what low-level steps constitute a task. As Perer and Shneiderman state, "Although some professions such as physicians, field biologists, and forensic scientists have specific methodologies defined for accomplishing tasks, this is rarer in data analysis. Interviewing analysts, reviewing current software approaches, and tabulating techniques common in research publications are important ways to deduce these steps" [120]. As an alternative approach, Heer, et al. [61], and Kurlander and Feiner [89] both hand-crafted a set of application-specific "chunking rules," based on the data, to combine low-level actions into a group for the purpose of condensing or summarizing workflow histories. The example given by Heer, et al., is that, for example, "in a word processor the keystrokes [c][h][u][n][k] might be represented as the word [chunk]." Likewise, the focus of our Tableau case study in Chapter 5 is to segment low-level events from the logs into consecutive subsequences that reflect the user's higher-level activity, which we refer to as the user's task, in this case. As such, we discuss the chunking rules proposed in other work in comparison to ours in that chapter.

3.4 Observing Intermediate User Actions

Modelers and researchers of user behavior may also be interested in user activities that are not "submitted" to the system. This includes information such as:

- Text that a user types in a search box and then deletes and then types again (not just the text that is finally submitted when they hit enter)
- Data on where the user's mouse hovers
- Interface selections that the user makes that are done client-side and not sent to the back-end

For example, Splunk provides a browser-based interface for visualization. The bulk of data transformation operations occur on the Splunk servers, which is where the logs are written. In order to capture the full extent of user behavior, such as extracting data, aggregating it, then toggling between a pie chart and a bar chart, it is necessary for the client to send this client-side activity information back to the system to be logged. Otherwise the system will not "see" this activity, since it does not pass to the back-end in the course of normal operation [6]. This is often not done, however, because sending this information can be expensive in terms of storage and network utilization and can slow the performance of the client. Because this information was not logged in Splunk when we collected our data, we were unable to collect explicit visualization and interaction activity in our Splunk case study described in Chapter 4, as shown in Figure 4.1. We have more information in our Tableau case study, but still not information about where the user's mouse hovers and interactions of that nature. There have been proposals to capture this kind of information, particularly in the context of the web [13, 29]. Developers of automated help systems may be particularly interested in such intermediate behavior because it may indicate confusion on the part of the user. For example, the Lumière project to create an automated assistant in Excel, modeled events such as "menu surfing," "mouse meandering," and "menu jitter."

3.5 Tracking Analysis State

One of the challenges is analyzing interaction data from analysis tools to understand user behavior is that much of what is logged are events triggered by user actions. As Hilbert and Redmiles explain, it can be difficult "to interpret the significance of UI events based only on the information carried within events themselves" [64]. Often, these events and the actions they represent can only really be understood in the context of the current application state that the user has changed via their previous actions. Thus, if this state is not also logged, it must be reconstructed in the log analysis code, which can be quite complicated, amounting to emulating the original program. An alternative is to log all relevant application state. This is more commonly done in workflow history tools, like those discussed in Section 2.3.3.1, for presenting the user and possibly their collaborators a record of their activity. For example, Heer, et al., model interaction histories as movement through a graph of application states [61]. Whether to log actions, states, or both, and how to organize the state representations (e.g., in a graph, or linearly) is a software engineering decision that has to be considered in light of trade-offs between log storage requirements and the amount of information retained. In our Splunk case study in Chapter 4, we unfortunately only had access to the queries, not the data that the queries were run over, nor the visualization that they resulted in, which is the relevant application state in that case. This is depicted visually in Figure 4.1. In our Tableau case study in Chapter 5, we were able to get access to both the actions, the data analyzed, and the application state, though this did cause some issues in that longer sessions with lots of state would not work with the annotation tool we used, due to space limitations.

3.6 Obtaining Information about Analyzed Data

It is challenging to understand users' analysis actions without visibility into the data and application context to which these actions were applied. We discuss the challenge of collecting application state in Section 3.5. If possible, with the user's permission, metadata and statistics about the data over which the user is operating should be also logged. Metadata includes information like schema (column names and data types) and provenance. Statistics includes things like descriptive statistics (describing the empirical distribution of the data), correlations, and measures such as entropy and cardinality. This would allow an inference model that supports an intelligent interface's predictions and suggestions to incorporate variables that reflect information about the user's data. It would also allow product managers to identify important user personas and their needs. For example, a company may be able to recognize by tracking this information that 35% of the users of their system use their browser-based GUI to analyze email marketing data specifically, and further observe that these users often follow very similar analysis workflows. This may spur the company to create a specialized product targeted towards these users needs that conveniently encapsulates these workflows. Less hypothetically, Splunk provides on top of its framework targeted "apps" – pre-built dashboards and tools designed for certain types of users with certain data sets. However, these apps are currently designed based on information manually gleaned from extensive interaction with customers, not based on data gathered through Splunk.

We acknowledge that collecting this data is a challenging proposition in many scenarios because users may be unwilling to provide information about their data, which may include operational, propriety, or personally identifying information. The "Show Me" paper by researchers at Tableau discusses the challenge this poses to developers trying to use logs to evaluate interface innovations [98] In our case studies, we were not able to collect this data for Splunk (Chapter 4), but we were for the Tableau case study (Chapter 5) because it was a more controlled environment administered in the context of a course assignment.

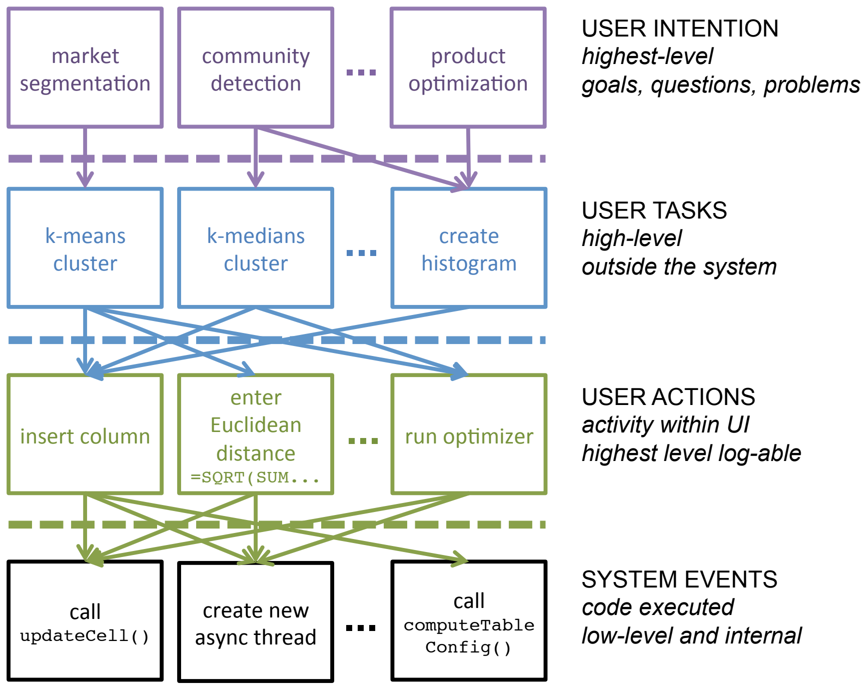


Figure 3.2: Users intentions motivate their actions, but may be hard to know (top row). When trying to understand user behavior especially with respect to data exploration and visualization, we are often interested in the high-level task the user is performing (second row). In the best case, we can log the actions the user takes via the UI (third row). Sometimes what is logged are low-level system events, which can make it very hard to reconstruct the user's behavior (bottom row).

3.7 Determining User Intent

The final significant challenge in trying to understand a user's analysis activity from their interaction records is identifying their high-level analysis goals or intentions. Even precisely defining the terminology related to "goals", "tasks", and so on, and identifying what level of abstraction to target for research and for use in interface representations, can be a challenge. Different researchers have proposed different taxonomies and terminology for activity at different levels of abstraction and granularity – work on this problem was discussed in Chapter 2.4. For example, in Hilbert and Redmiles terminology, "abstract interaction events are not directly generated by the user interface system, but may be computed based on UI events and other contextual information such as UI state. Abstract interaction events are indicated by recurring, idiomatic patterns of UI events and indicate higher level concepts such as shifts in users' editing attention or the act of providing values to an application by manipulating application components." [64] Moreover, "domain/task-related and goal/problem-related events are at the highest levels. Unlike other levels, these events indicate progress in the users tasks and goals. Inferring these events based on lower level events can be straightforward when the user interface provides explicit support for structuring tasks or indicating goals.” [64] As discussed in Chapter 1, in our usage, we roughly refer to transformations and actions (as represented by Splunk queries and Tableau events, respectively) at the lowest level, which together are aimed at a higher-level and more abstract task. The set of tasks together are aimed at accomplishing a higher-level goal, which is reflective of the users’ intent.

One reason to try to understand the user’s analysis behavior in the first place is to enable the creation of intelligent interfaces that can assist the user via recommendations and interface adaptions. The user’s goal, or the task they are trying to perform, as well as their position and their level of expertise, are all likely important factors that will likely greatly impact what interface elements an adaptive interface should show or what recommendations should be given. This has long been recognized as important for determining what visualizations to automatically generate for a user [27].

Where possible, information about goals, expertise, and other relevant context could be solicited from the user. However, if this solicitation requires the user to do additional work that does not benefit them, it is highly unlikely to be successful (see Chapter 6 of *Search User Interfaces*) [59]. One possible solution could be, for instance, asking the user to select an answer from a list at a natural inflection point in their workflow, to reduce the inconvenience to the user (for example, as is often done when one unsubscribes from a mailing list). More research will be required to determine how to do this in a way that is not annoying and that still yields useful information. If an adaptive, predictive, customized, or mixed-initiative interface is implemented that provides suggested actions or tasks to the user, the interface should also provide the user with the opportunity to comment on, rate, rank, and mark as interesting or uninteresting each suggested action, as suggested by Perer and Shneiderman [120]. This data should be recorded to improve the underlying model used to generate the suggestions and can also be provided to the user. Such data also becomes very useful to the user as an artifact of their analysis and exploration process.

In our Chapter 4 case study with Splunk, we gathered queries post-facto such that we were not able to arrange to solicit information about the query writers’ intent. However, in our Tableau case study in Chapter 5, we designed the study so that participants would submit both annotations of their event logs, to reveal their higher-level thought processes, as well as reports, which summarize even more of their high-level goals. In both cases, our goal was to determine not the highest levels of user intent, but rather higher-level user tasks from lower-level activity.

3.8 Conclusion

We conclude our discussion of the challenges associated with using log analysis to understand the behavior of users of analysis systems with a brief discussion of implementation considerations. As suggested previously, one reason high-quality user activity records may not be collected from data analysis systems is that often, understanding and modeling user behavior is not a first priority for developers of such applications, who instead focus on logging for system debugging and performance monitoring. Horvitz, et al., note that "establishing a rapport with the Excel development team was crucial for designing special instrumented versions of Excel with a usable set of events" [67]. Similarly, in our experience with Splunk, we found that some of the interaction data that we were interested in, particularly the visualization activity that occurs on the client side, was not data that the development team had previously needed to log [6].

Hilbert and Redmiles have raised the concern that requiring more data about user behavior "places an increased burden on application developers to capture and transform events of interest ... [and] complicates software evolution since data collection code is typically intermingled with application code" [64]. We acknowledge that there will be costs associated with more thorough and high-quality logging of user activity, but we argue that as the examples in Chapter 1 demonstrate, this effort will be well worth it for the wide variety of applications and research it enables. Such work will ultimately benefit the end-user of data analysis and visualization systems, particularly during data exploration, by enhancing human problem-solving abilities and speeding the pace of discovery.

There are a number of concerns related to collecting data from or about the user, particularly if it is personally identifying information or sensitive operational data from an organization. A full discussion of these concerns and their possible solutions are outside the scope of this chapter. We argue though that through careful planning and working to develop trusted collaborations with the users who will be the ultimate benefactors of such efforts, researchers and tool builders should be able to improve their practices for logging interaction data from data analysis and visualization tools. Obtaining permission from users before logging their data, and allowing them to see what data is logged and edit and remove portions that they do not wish to have remain in the logs provides important protections. Policies to permanently remove data after a fixed amount of time, after useful information has been derived from the specifics and placed into general models can further help to protect individual privacy. Some users have already shown themselves willing to share data about their usage and behavior with companies in the interest of improving their user experience and the company's product. As already noted, it is important that such data be collected with users' full knowledge and consent.

Chapter 4

Analyzing Transformation Sequences

In this chapter, we present a case study of data logged from Splunk, as our first case study in how to summarize user activity and identify tasks in interaction data logged from visual analytics tools. Splunk is a technology company that makes software for the analysis of machine-generated data, such as logs. Log analysis is one of the earliest applications of data analysis in computer and information systems, with papers on the topic dating back to at least 1973 [104]. System developers began using log analysis to understand and improve system performance and usability. An entire industry for log management and analysis has since arisen. Splunk has been a market-leader in this industry for several years, with 950M USD in revenue in 2016 [62].

Founded in 2003, Splunk originally targeted users in IT operations. It quickly found uses in other domains, such as security, healthcare, sales, finance, manufacturing, and more, as business processes in many domains produce machine-generated logs as a byproduct of their operations. Consequently, the types of data sets users analyze with Splunk span a wide range. Examples include system event logs, web access logs, customer records, call detail records, and product usage logs. Similarly, Splunk users have a variety of data analysis needs. For example, a software engineer might use Splunk to identify the root cause of a system failure. A user experience designer might use it for A/B testing, to determine which version of a webpage, A or B, users prefer. An IT administrator might use Splunk to detect unusual and possibly malicious traffic to their servers. An executive might use Splunk to calculate product usage statistics to share with potential investors.

As these examples make clear, log analysis is a type of data analysis where the source of the data is machine-generated logs. Moreover, it is an important type of data analysis, because as more human activity becomes mediated by computers, which produce logs as a by-product of their operation, the amount and variety of log data will continue to increase. Understanding how people analyze logs will help us understand how people analyze data. The Splunk data we collected provides extensive records of how people analyzed logs in a diverse set of situations.

To provide context about the data and our analysis, we begin by describing the Splunk interface and how users interact with it (Section 4.1). We then describe the data we collected,

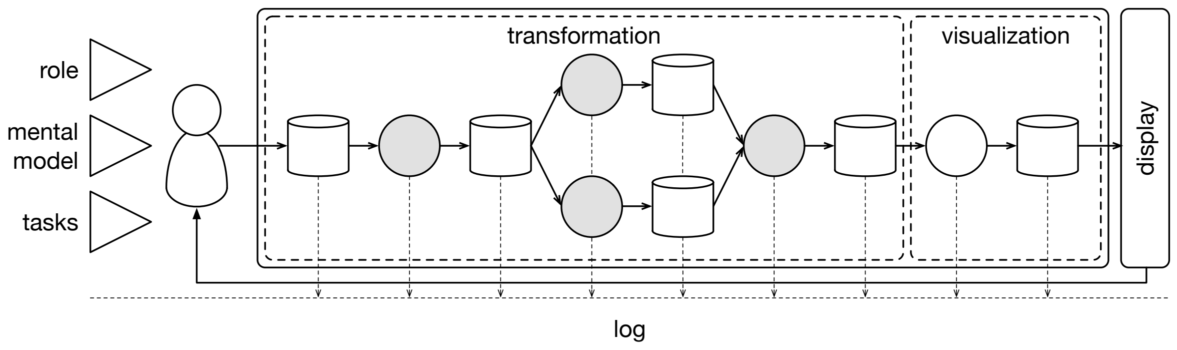


Figure 4.1: In the Splunk case study, we collected only the *transformation operations* users applied, not the *visualization operations* that users applied, the *data* they applied them to, nor any information about their *goals*.

to highlight the specific challenges we faced and to set the stage for our analysis (Section 4.2). The data we collected represents only a partial subset of the data we would have in an ideal case: as Figure 4.1 highlights, we collected only the *transformation operations* users applied, not the *visualization operations* that users applied, the *data* they applied them to, nor any information about their *tasks*. The bulk of this chapter discusses what we can learn from this type of data and how, by describing the approach and results of our analysis (Section 4.3). We supplement this analysis with a survey of Splunk sales engineers about how customers use Splunk (Section 4.4). We conclude by summarizing what can be learned from data of this type given its limitations, and suggest possible extensions of this work (Section 4.5).

The main result of this chapter is a description of log analysis activity based on the queries we collected from Splunk and our survey results. The data reveal that log analysis is an activity dominated by filtering, data munging, and summarization, which makes sense in light of known log analysis use cases like monitoring and troubleshooting. The data provides a detailed picture of the frequency of different tasks in these categories for a large sample of Splunk logs. The data also shows that log analysis is being adopted by non-technical users for a variety of business applications. The following chapters will use the Splunk data to paint an evidence-based picture of log analysis activity, and as such will be important for developers of data analysis tools and other students of data analysis behavior.

Term	*Definition*
query	a small program written in the Splunk query language, consisting of pipelined stages
stage	a portion of a query syntactically between pipes; conceptually a single transformation
task	roughly, a unit of work that needs to be done to accomplish a goal
event	a raw, timestamped item of data indexed by Splunk, similar to a tuple in a database
field	a key used to look up a value in an event, similar to the concept of a column name
value	event data corresponding to a field, similar to a column entry in a particular row
transformation	a group of similar commands e.g., filter or aggregate; each stage is a transformation
pipeline	the sequence of transformations in a query
command	the part of a stage that indicates which operation to apply to the data
argument	the parts of a stage that indicate which fields, values, and options to use
interactive	a query run when it is entered by the user into the search bar
scheduled	a query saved by a user and scheduled to run periodically like a `cron` job

Table 4.1: Definitions of terms used.

4.1 Case Study: Splunk

The data we collected from Splunk are query logs. These contain queries written in Splunk Processing Language (SPL). This section describes this query language and Splunk's interface in more detail. Table 4.1 lists the terminology introduced in this section.

4.1.1 Overview of the Splunk Interface

Uploading data To use Splunk, the user first indicates which data they want Splunk to index, such as a log directory on a file system. Because Splunk is most often used to analyze event logs, Splunk segments this data into temporal *events* by using timestamps as delineators, and processes these events using a MapReduce-like architecture [20]. Splunk does not require the user to specify a schema for the data, because much log data is semi-structured or unstructured, and there is often no notion of a schema that can be imposed on the data a priori. Rather, *fields* and *values* are extracted from events at run time based on the *source type* of the data (example source types include Apache web server logs or Palo Alto Networks firewall logs). When a user defines a new source type, Splunk guides the user in constructing regular expressions to extract fields and values from each incoming raw event.

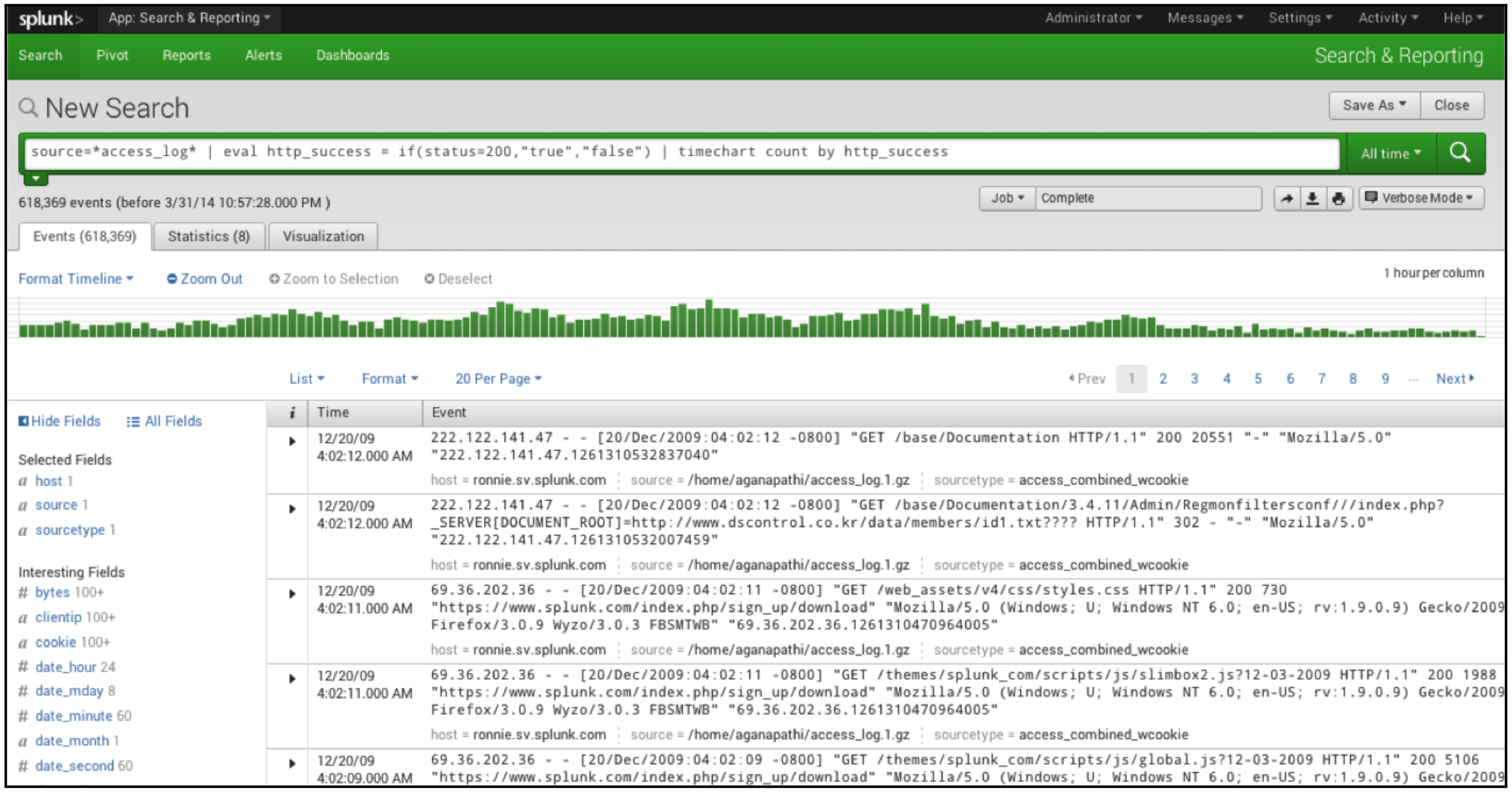
Analyzing data Once their data is uploaded into Splunk, users can examine it in a variety of ways. Splunk includes a *query* language for transforming data and a graphical user interface (GUI) with tools for visualizing query results. The default Splunk GUI displays the first several events indexed, with extracted fields highlighted on the left hand side, and a histogram of the number of events over time displayed along the top (Figure 4.2). The user types queries written in the Splunk query language into the search bar at the top of this view. When the user composes their query in the GUI, we call it an *interactive* query. When the user enters a query that performs a filter, the GUI updates to display events which pass through the filter. When the user uses a query to add or transform a field, the GUI displays events with this field updated. Users can input the results of their queries into interactive visualizations like tables, time series, and bar charts, and even assemble these visualizations into dashboards that update in real-time (Figure 4.3).

Query language The Splunk query language is modeled after Unix command line utilities like the `grep` tool and pipe operator. The Splunk query language is complex and supports a wide range of functionality, including but not limited to: reformatting, grouping and aggregating, filtering, reordering, converting numerical values, and applying data mining techniques like clustering, anomaly detection, and prediction. These activities are examples of different *tasks*. Individual commands can be thought of as accomplishing a task individually, as well as combining together in a pipeline accomplish a higher-level task. The Splunk query language has 134 distinct core commands at the time of this writing, and commands are often added with each new release. In addition, users and Splunk app developers can define their own commands. A Splunk query consists of a set of *stages* separated by the pipe character, and each stage in turn consists of a *command* and *arguments*. A set of stages together is called a *pipeline*. Splunk passes events through each stage of a query. Each stage filters, transforms, or enriches data it receives from the previous stage, and pipes it to the subsequent stage, updating the displayed results as they are processed. A simple example of a query is a plain text search for specific strings – this query would filter out events not containing this string. More complex queries can perform advanced transformations, such as clustering the data using k-means. Users can save certain queries and schedule them to be run on a given schedule, much like a `cron` job. We call these queries *scheduled* queries.

4.1.2 An Example Splunk Query

In this section, we walk through each stage of a simple yet typical Splunk query, as an example. In this example, we will consider a query meant to run over web server access log data. A web server access log contains an event for every HTTP request that was made to that server. HTTP is one of the protocols used to send webpages over the Internet. For example, when you navigate to a webpage in your browser, your browser makes an HTTP request to the server that stores that webpage. The server writes information about this request, such as what page was requested and the status of the server's response, to its access logs. The request can result in multiple outcomes, which are indicated by the response status code. A 200 status means that the web server successfully returned the page requested. A 404 status means that the web server was unable to find the page requested, so the request failed. A 500 status means that the web server experienced some internal error. This example query provides a count of errors from a web server access log. It aggregates the error counts by status code, and augments these groups with status descriptors:

```
search error | stats count by status | lookup statuscodes status OUTPUT statusdesc
```

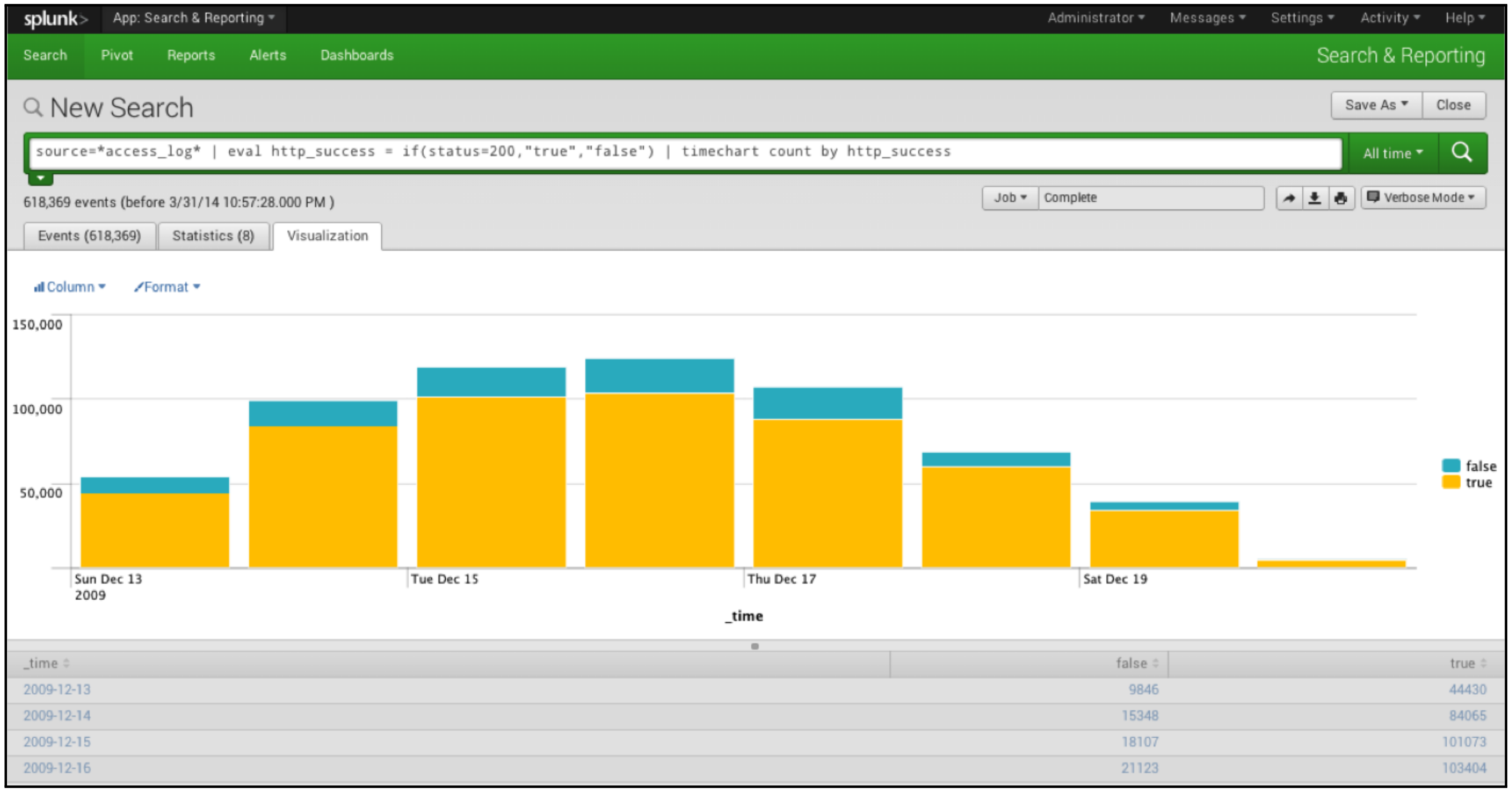


```
source=*access_log* | eval http_success = if(status=200,"true","false") | timechart count by http_success
```

618,369 events (before 3/31/14 10:57:28.000 PM)

```
Time                   Event
12/20/09               222.122.141.47 - - [20/Dec/2009:04:02:12 -0800] "GET /base/Documentation HTTP/1.1" 200 20551 "-" "Mozilla/5.0"
4:02:12.000 AM         "222.122.141.47.1261310532837040"
                       host = ronnie.sv.splunk.com | source = /home/aganapathi/access_log.1.gz | sourcetype = access_combined_wcookie
12/20/09               222.122.141.47 - - [20/Dec/2009:04:02:12 -0800] "GET /base/Documentation/3.4.11/Admin/Regmonfiltersconf///index.php?
4:02:12.000 AM         _SERVER[DOCUMENT_ROOT]=http://www.dscontrol.co.kr/data/members/id1.txt???? HTTP/1.1" 302 - "-" "Mozilla/5.0"
                       "222.122.141.47.1261310532007459"
                       host = ronnie.sv.splunk.com | source = /home/aganapathi/access_log.1.gz | sourcetype = access_combined_wcookie
12/20/09               69.36.202.36 - - [20/Dec/2009:04:02:11 -0800] "GET /web_assets/v4/css/styles.css HTTP/1.1" 200 730
4:02:11.000 AM         "https://www.splunk.com/index.php/sign_up/download" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.9) Gecko/2009
                       Firefox/3.0.9 Wyzo/3.0.3 FBSMTWB" "69.36.202.36.1261310470964005"
                       host = ronnie.sv.splunk.com | source = /home/aganapathi/access_log.1.gz | sourcetype = access_combined_wcookie
12/20/09               69.36.202.36 - - [20/Dec/2009:04:02:11 -0800] "GET /themes/splunk_com/scripts/js/slimbox2.js?12-03-2009 HTTP/1.1" 200 1988
4:02:11.000 AM         "https://www.splunk.com/index.php/sign_up/download" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.9) Gecko/2009
                       Firefox/3.0.9 Wyzo/3.0.3 FBSMTWB" "69.36.202.36.1261310470964005"
                       host = ronnie.sv.splunk.com | source = /home/aganapathi/access_log.1.gz | sourcetype = access_combined_wcookie
12/20/09               69.36.202.36 - - [20/Dec/2009:04:02:09 -0800] "GET /themes/splunk_com/scripts/js/global.js?12-03-2009 HTTP/1.1" 200 5106
4:02:09.000 AM         "https://www.splunk.com/index.php/sign_up/download" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.9) Gecko/2009
```

Figure 4.2: The default Splunk GUI view displays the first several events indexed, with extracted fields highlighted on the side, and a histogram of the number of events over time displayed along the top.The user types their query into the search bar at the top of this view.



```
source=*access_log* | eval http_success = if(status=200,"true","false") | timechart count by http_success
```

618,369 events (before 3/31/14 10:57:28.000 PM)

_time	false	true
2009-12-13	9846	44430
2009-12-14	15348	84065
2009-12-15	18107	101073
2009-12-16	21123	103404

Figure 4.3: Some Splunk commands generate visualizations that can be viewed by switching over to the "Visualization" tab. Here, the user has the option to alter the parameters of the visualization, like color and chart type. This visualization shown here corresponds to the query shown in Figure 4.2.

This query has three stages: `search`, `stats`, and `lookup` are the commands in each stage, `count by` and `OUTPUT` are functions and option flags passed to these commands, and "error", "status", "statuscodes", and "statusdesc" are arguments. In particular, "status" and "statusdesc" are fields.

To see how this query works, consider the following toy data set:

0.0	-	**error**	404
0.5	-	OK	200
0.7	-	**error**	500
1.5	-	OK	200

The first stage of the query (`search` error) filters out all events not containing the word "error". After this stage, the data looks like:

0.0	-	**error**	404
0.7	-	**error**	500

The second stage (`stats count by` status) aggregates events by applying the `count` function over events grouped according to the "status" field, to produce the number of events in each "status" group.

count	status
1	404
1	500

The final stage (`lookup` status codes status `OUTPUT` statusdesc) performs a join on the "status" field between the data and an outside table that contains descriptions of each of the codes in the "status" field, and puts the corresponding descriptions into the "statusdesc" field.

count	status	statusdesc
1	404	Not Found
1	500	Internal Server Error

4.2 Data Collected: Splunk Query Logs

We collected over 200K Splunk queries. The data set consists of a list of timestamped query strings. Table 4.2 summarizes some basic information about this query set.

We wrote a parser for this query language; the parser is freely available [3]. This parser is capable of parsing over 90% of all queries in the data set, some of which may be valid failures, as the queries may be malformed. (This limitation only affects the cluster analysis in Section 4.3.)

4.2.1 Challenges and Limitations

One challenge we faced is that the queries we collect only covered the data transformation portion of analyses in Splunk and not the visualization portion, because the visualization portion is done in Splunk on the client and thus is not logged. We could infer some information about the visualization in some cases based on the use of certain visualization-related transformation commands at the ends of some pipelines, but the amount we could learn this way is limited. This challenge of missing data from logs was discussed in Section 3.1. As visualization is an important part of data analysis in addition to data transformation, missing this information is a limitation of our findings. Likewise, having only access to the queries the users finally entered, we did not have access to any of users' other interactions with the client interface. This lack of access to intermediate actions such as scrolling, adjusting client-side parameters with buttons, and so on, was discussed in the chapter on challenges in log analysis in Section 3.4.

Another challenge was that some of the queries labeled as interactive in our data set turned out to be programmatically issued from sources external to Splunk, such as a user-written script. In other words, the correct provenance of the queries was not available – another common challenge, which we discussed in Section 3.2. It is difficult to separate these mislabeled queries from the true interactive queries, so we leave their analysis to future work, and instead focus our analysis in this paper on scheduled queries.

It is important to note that we do not have access to any information about the data over which the queries were issued because these data sets are proprietary and thus unavailable. This common challenge was discussed in Section 3.6. This is related to the challenge of not having access to state information, because application state encompasses the current state of the data after transformations and visualizations have been applied. In a pipelined query language like Splunk, the transformations are expressed all together rather than the user incrementally changing the data via successive transformations, so in this case the relevant state would be the state of the data after the query is applied, which we also don't have access to. This challenge is described in Section 3.5. Having access only to query logs is a common occurrence for data analysis, and methodologies that can work under these circumstances are therefore important to develop. Further, by manually inspecting the queries and using them to partially reconstruct some data sets using the fields and values mentioned in the queries, we are fairly certain that these queries were issued over many different sources of data (e.g., web server logs, security logs, retail transaction logs, etc.), suggesting the results presented here will generalize across different datasets.

[3] `https://github.com/salspaugh/splparser`

Total queries	203691
Interactive queries	18872
Scheduled queries	184819
Distinct scheduled queries	17085

Table 4.2: Characteristics of the set of queries analyzed from the Splunk logs.

This data cleanliness issue underscores the importance of better system instrumentation that tracks context, such as whether a given action was issued by a human or a machine [7]. However, most logging statements, including the data we have collected, are logged for the purpose of debugging systems like Splunk, rather than understanding usage, as discussed in Chapter 3. This is a common issue for logs from all types of systems, requiring those who hope to gain insight about user behavior to attempt to reconstruct usage patterns from low-level debugging statements [50], as discussed in Section 3.3. A related challenge that we faced – not a problem with the logging but with the structure of the query language itself – is that many transformation commands in Splunk are overloaded, so that the invocation of the commands do not correspond one-to-one with the high-level task the user is performing. Much of our analysis is directed at handling this challenge – we first re-categorize all commands into a smaller set of categories, then examine uses of the most common categories to identify common high-level actions. For tools like HARVEST discussed in Chapters 2 and 3, which are designed from the outset to reflect the users' high-level "semantic actions" directly in their interfaces, this problem may be avoided, but it does require planning for this from the interface design phase [50].

Lastly, other important factors that we don't have access to include who the user is and what problems they are trying to solve. These factors all underly the user's intent, which is difficult to gain information about from logs – another challenged discussed in Section 3.7. For example, in the case of web access log data, an operations user will want to know, "Where are the 404s?[4] Are there any hosts that are down? Is there a spike in traffic that I should add capacity for?" On the same data, a marketer will want to know, "What keywords are people searching today after my recent press release? What are the popular webinars viewed on my website?" On the other hand, a salesperson may ask, "Of the visitors today, how many are new versus returning, and how can I figure out whom to engage in a sales deal next based on what they're looking for on the web site?" Capturing this supplemental information from data analysis tools to include in the analysis would be useful for later tailoring tools to particular use cases. We have gathered some information about this (Section 4.4) but unfortunately we could not cross-reference this data with query data. Due to this missing context information, the utility of this dataset for generating recommendations to users is limited, but nonetheless, we can learn much about what high-level activity users engage in, as we show in the following sections.

[4]404 is an HTTP standard response code indicating the requested resource was not found.

4.3 Analysis of Splunk Query Logs

The preceding sections provided context about the log data we collected and the system it came from. Here, we will further set the stage for our analysis by discussing what we might hope to learn from it. As discussed in depth in Chapter 1, we can analyze logs that contain data about how a tool is being used to investigate questions regarding performance, usability, business processes, and user behavior. We can also use them to inform our efforts in documentation, configuration, reproducibility, user simulation, and new feature development. Exactly what we can learn depends on the quantity and quality of data. We discussed the limitations of our data in Section 4.2. With these in mind, we consider what more specific questions we can ask of this data.

For researchers and developers interested in improving data analysis tools for log data, an important question is: what tasks constitute log analysis, and how does it differ from other types of analysis activity, such as machine learning or on-line analytic processing (OLAP) in databases? For example, an important task in OLAP is computing aggregates of various columns grouped over various other columns,whereas in machine learning, one of the core tasks is predicting an unknown value given a set of input values. While the analysis of logs from "information systems" to understand user behavior dates back decades [154], to our knowledge, there are no studies of logs from *log analysis systems* to understand *log analysis* behavior.

In addition to understanding what log analysis is in general, developers of systems like Splunk can use logs to understand the usage of their system in particular. They can compare system usage to underlying assumptions for system design, such as the need for various display options, and use this comparison to evaluate and improve the system. Our analyses have in fact been used for these purposes at Splunk. However, here we focus on the more general question: What can we learn about what tasks are involved in log analysis from a record of the transformations users applied while analyzing logs?

4.3.1 Transformation Frequencies

As a first pass at understanding the tasks involved in log analysis, we start with the most basic question: which transformations did users apply to their data, and how many times did they do it? The simplest approach to answer this question would be to count the number of times each Splunk command was issued. However, Splunk command frequencies are difficult to generalize outside of Splunk [5]. So, to allow for comparisons to other workflows and abstract our observations beyond the Splunk search language, we manually classified the 134 Splunk commands used into 17 course-grained categories representing the types of transformations encoded, such as filtering, aggregating, and reordering (Table 4.3). We refer to these categories as transformations. The categorization of each command is listed in Appendix A.

Note that because some Splunk commands are overloaded with functionality, several commands actually perform multiple types of transformations, such as aggregation followed

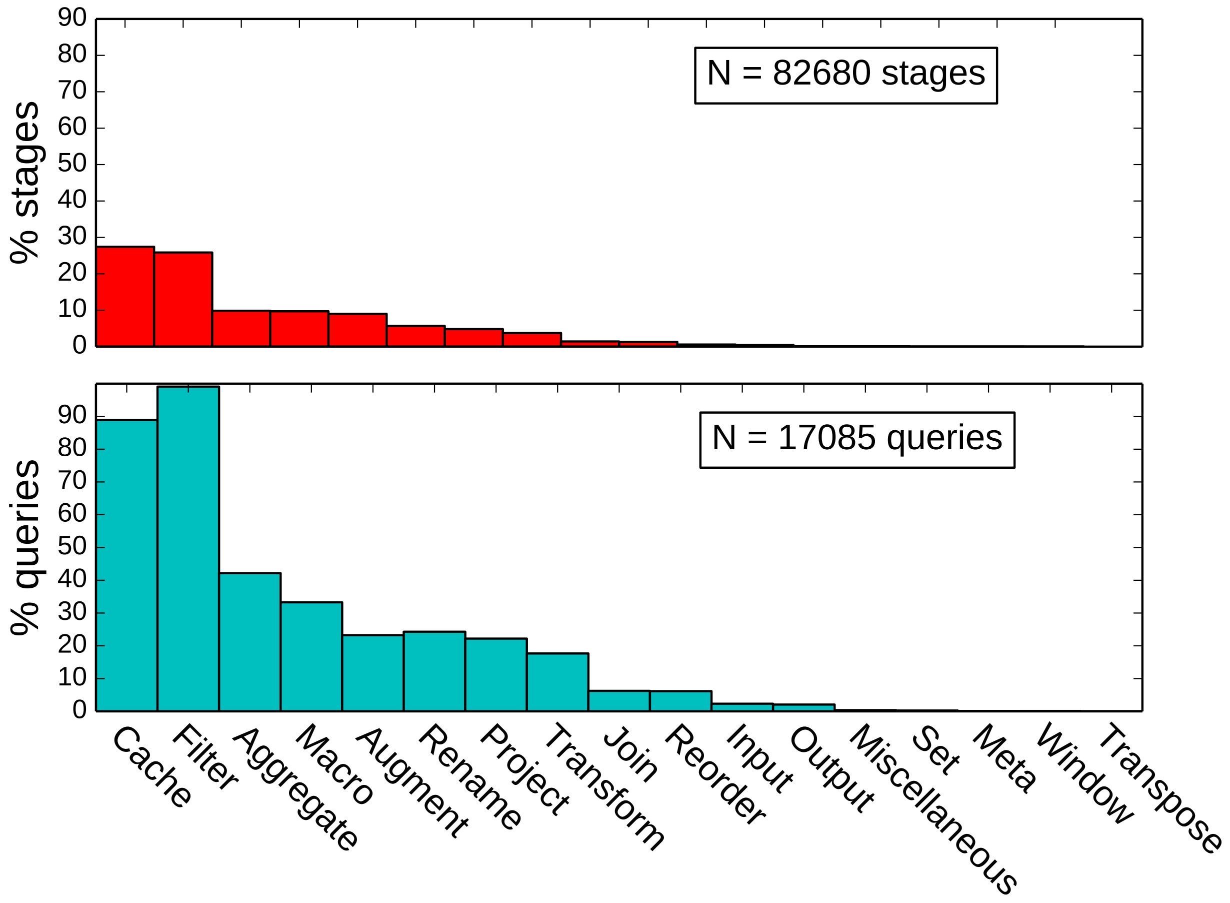


Figure 4.4: The distribution of data transformations that are used in log analysis. The top graph shows, for each transformation, the percent of stages that apply that transformation. The bottom graph shows, for each transformation, the percent of queries that contain that transformation at least once (so the percents do not add to 100).

by renaming. In these cases, we categorized the command according to its dominant use case.

We first counted the number of times that each transformation was invoked (Figure 4.4). The most common transformation, or command category, used is **Cache** (27% of stages). Scheduled queries are crafted and set up to run periodically, so the heavy use of caching and macros is unsurprising: Splunk adds caching to scheduled queries to speed their execution, and macros capture common workflows, which are likely to be discovered by users after the iterative, ad hoc querying that results in a "production-ready" scheduled query. Although we do not report directly on them here due to data quality issues (Section 4.2), our preliminary investigation suggests that interactive queries have a similar distribution except that the use

Transformation	*Description*	*Top Commands*	*% Queries*	*Examples*
Aggregate	coalesce values of a given field or fields (columns) into one summary value	stats timechart top	86 9 3	stats sum(size_kb) timechart count by region top hostname
Augment	add a field (column) to each event, usually a function of other fields	eval appendcols rex	57 19 15	eval pct=count/total*100 spath input=json rex "To: (?<to>.*)"
Cache	write to or read from cache for faster processing	summaryindex sitimechart	98 30	summaryindex namespace=foo sitimechart count by city
Filter	remove events (rows) not meeting the given criteria	search where dedup	100 7 4	search name="alspaugh" where count > 10 dedup session_id
Input	input external data into the system	inputlookup	88	inputlookup data.csv
Join	join two sets of events based on matching criteria	join lookup	82 16	join type=outer ID lookup
Macro	apply user-defined sequence of Splunk commands	`sourcetype_metrics` `forwarder_metrics`	50 13	`sourcetype_metrics` `forwarder_metrics`
Meta	configure execution environment	localop	83	localop
Miscellaneous	commands that do not fit elsewhere	noop	39	noop
Output	write or send results externally	outputlookup	89	outputlookup results.csv
Project	remove all columns except those selected	table fields	80 22	table region total fields count
Rename	rename fields	rename	100	rename cnt AS Count
Reorder	reorder events based on some criteria	sort	100	sort - count
Set	perform set operations on data	append set	66 40	append [...] set intersect [...] [...]
Transform	mutate the value of a given field for each event	fillnull convert	96 2	fillnull status convert num(run_time)
Transpose	swap events (rows) with fields (columns)	transpose	100	transpose
Window	add windowed data as new field	streamstats	90	streamstats first(edge)

Table 4.3: Manual classification of commands in the Splunk Processing Language into transformations categories. For each category, *Top Commands* shows the most-used commands. *% Queries* shows, for all queries containing a given transformation, what percent of queries contained that command.

of **Cache** and **Macro** is less frequent, and the use of **Input** is more frequent.

The next most common transformations include **Filter** (26% of stages) and **Aggregate** (10% of stages). Filtering makes sense as a primary task of log analysis – logging collects a great deal of information over the course of operation of a system, only a fraction of which is likely to be relevant to a given situation. Thus it is almost always necessary to get rid of this extraneous information.

These activities support the task of counting how many things with certain attributes there are. This task is similar to computing aggregates grouped by various attributes in OLAP, optionally with `WHERE` clauses. However, one important potential difference is that filters in Splunk can be used to very flexibly and easily select complex-to-specify subsets of data, which could be difficult to replicate in a environment with a predefined schema. (The different types of Splunk filters will be discussed in a later section.) Though counting may sound like an oversimple data analysis task compared to tasks like prediction, this flexibility gives great variety to the types of things that can be defined and counted, and provides important business and operational information.

Another common transformation includes **Macro** (10% of stages). Macros are commands that get replaced with a longer sequence of pre-defined commands at run time; thus they represent a means to more efficiently express common multi-step computations. The prevalence of **Macro** use means that log analysis involves many common, repetitive sub-tasks, as opposed to one-off work; however, because we lack access to the macro definitions, we can't say more about what types of tasks these macros represent.

The next most common transformation includes **Augment** (9% of stages). This type of transformation involves adding new data, often derived as transformations of other data in the log or of external information. Among other things, such activity supports the task of data munging – reformatting or otherwise enhancing the data for later analysis. Data munging is treated in some sources as a separate stage of the overall data analysis process. It is not well supported within traditional data analysis systems like databases. To the extent that it is considered to be a task in machine learning, it is considered to be part of feature engineering. Good feature engineering, though recognized as critically important, is considered to be a highly manual, domain-specific, informal, and applied (as opposed to theoretically supported) process. Its prevalence here shows that it is an integral part of log analysis.

For each transformation type, we also computed the number of queries that used that transformation (Figure 4.4). This gives us some idea of how many of the queries would be expressible in a restricted subset of the language, which is interesting because it tells us the relative importance of various transformations.

From this we see that **Filter** transformations are extremely important – 99% of scheduled queries use such transformations. Without **Aggregate** transformations, 42% of scheduled queries would not be possible. This supports our earlier point that an important task in log analysis is counting things.

Around a quarter of queries use **Augment**, **Rename**, and **Project** transformations, and 17% use commands that **Transform** columns. These are all important types of data munging tasks that are being done before the rest of the analysis begins, or they could be analysis tasks followed by some work to clean up the data for presentation.

On the other end, **Join**s are only used in 6% of scheduled queries. This could be because log data is not usually relational and generally has no schema, so it may often not have information that would satisfy key constraints needed for join, or it may already be sufficiently denormalized for most uses. It could also be because these are scheduled queries, and expensive **Join** operations have been optimized away, although again anecdotally the interactive queries do not suggest this. **Reorder** transformations are also used only 6% of the time – log events are already ordered by time by Splunk, and this is probably often the desired order. **Input** and **Output** transformations are used in only 2% of scheduled queries – these again could have been optimized away, or possibly captured in **Macro**s.

Lastly, the other transformations are used in nearly zero queries. In the case of **Windowing** transformations, this could be because windowed operations are accomplished "manually" through sequences of **Augment** transformations or via overloaded commands that were classified as other transformation types. We were surprised such operations were not more common. In the case of the others, such as **Transpose**, it is more likely because log data is rarely of the type for which such operations are needed.

4.3.2 Common Pipelines

Now that we have assessed the frequency at which individual transformations occur in the logs, we'll next consider sequences of such transformations, and how frequently these occur. We refer to these sequences of transformations as pipelines, and we examine them in this section. (See Table 4.1 for a definition of these terms.) The reason to consider pipelines in addition to individual transformations is that tasks do not generally require a single transformation. Thus, to understand the tasks involved in log analysis, we must examine sequences of transformations as well. This way, we can discover typical log analysis pipelines employed by Splunk users. To examine these sequences of transformations, we compute a transition matrix, as described below. This matrix describes how often, in a log analysis pipeline, each transformation is followed by each of the other transformations. We then create a visual representation of this matrix, called a transition frequency diagram, that highlights the most common transitions. This is useful because it provides a concise way to identify many transformation pipelines.

This approach is related to process mining, which uses data about business activities, often from logs, to learn models of the operational processes participants used in that activity [1]. One difference between that and what we have done is that we make no attempt to fit a model to the data. Rather, here we report on the empirical frequencies.

To create the aforementioned matrix and graph, we did the following. For each pair of transformation types, we counted the number of times within a query that the first transformation of the pair was followed by the second transformation of the pair. We used these counts to compute, for each transformation, how frequently each of the other transformation

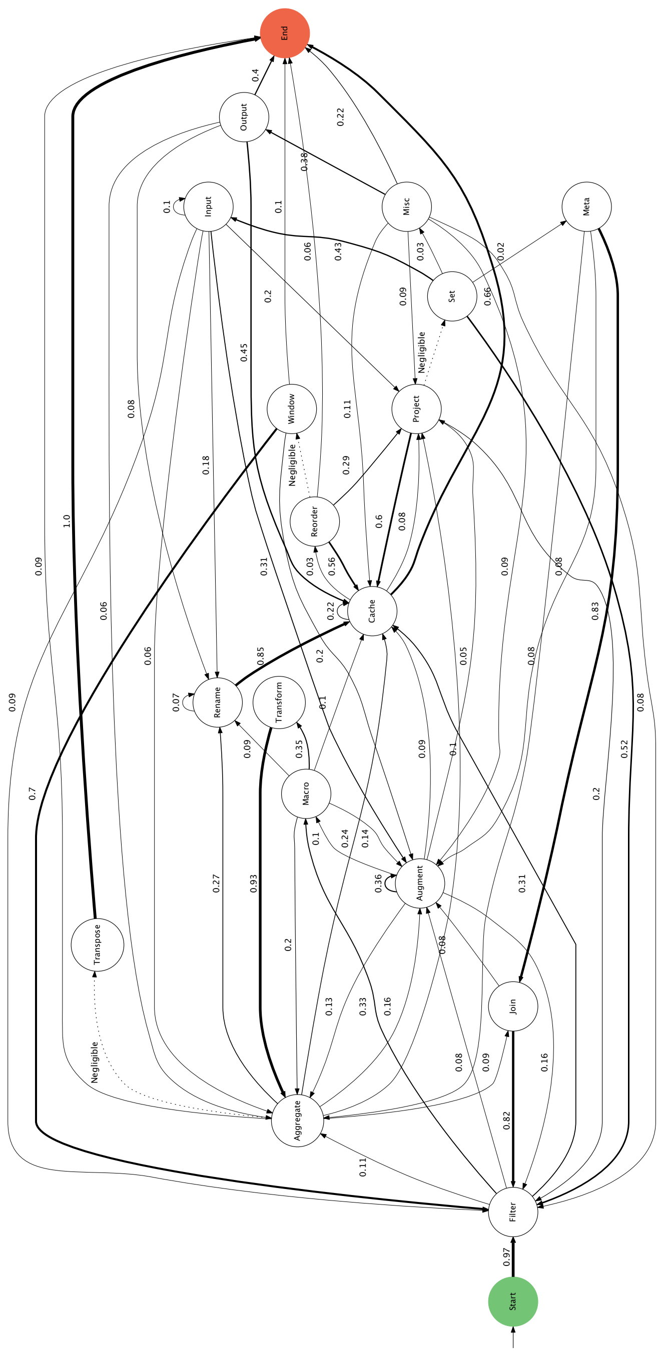


Figure 4.5: Transition frequency diagram describing, for all distinct scheduled queries, the pairwise transition frequency between transformations. Only edges with weight greater or equal to .05 are shown, for clarity. This visual representation of transition frequencies shows likely log analysis pipelines employed by Splunk users.

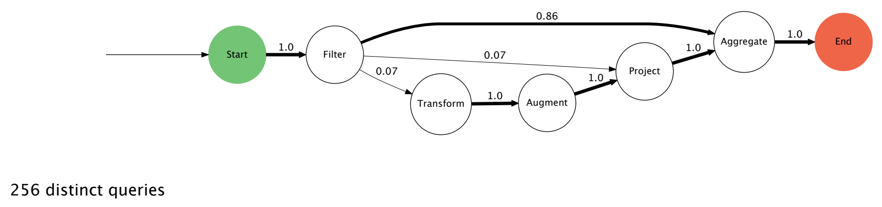


Figure 4.6: The pairwise transition frequency between transformations for web access log queries.

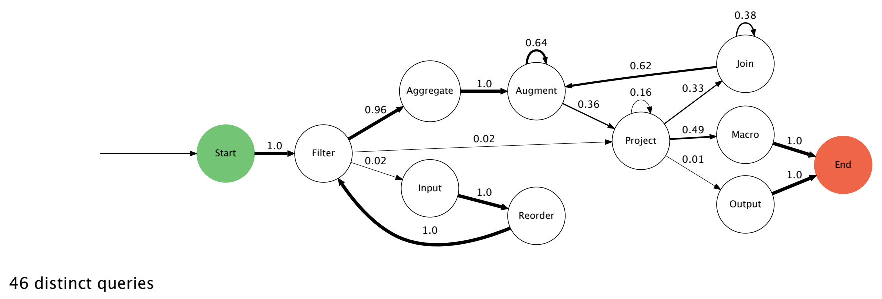


Figure 4.7: The pairwise transition frequency between transformations for OS event log queries.

types followed it in a query. We averaged these frequencies across users, since we have far more queries from some users than others [5].

We used these frequencies to create the transition frequency diagram as shown in Figure 4.5. Each node is a type of transformation, and each edge from transformation A to a transformation B indicates the number of times B was used after A as a fraction of the number of times A was used. Also included as nodes are states representing the start of a query, before any command has been issued, and the end of a query, when no further commands are issued. The edges between these nodes can be thought of as transition probabilities that describe how likely a user is to issue transformation B after having issued transformation A.

Using these graphs, we can discover typical log analysis pipelines employed by Splunk users. We exclude from presentation sequences with **Cache** transformations, as such transformations have in most cases been automatically added to scheduled queries by Splunk to optimize them, as well as **Macro**s, because these can represent any transformation, so we do not learn much by including them. We can read the remaining top transformation pipelines by weight (where the weight of a path is the product of its edges). These are:

[5]Because average is sensitive to outliers, we attempted to use the median over users, but then we found that too many of the transitions were close to zero, with the exception of the transition **Start -¿ Filter -¿ End** – representing simple searches. Readers should keep in mind that the presented observations may be overly influenced by some outlier user accounts, although this is somewhat mitigated by the fact that the data from all users are equally weighted.

- **Filter**
- **Filter** | **Aggregate**
- **Filter** | **Filter** [6]
- **Filter** | **Augment** | **Aggregate**
- **Filter** | **Reorder**
- **Filter** | **Augment**

The preponderance of **Filter** transformations in typical pipelines is not surprising given that it is the most frequently applied transformation. These pipelines support our earlier observations about tasks involved in log analysis. We investigate **Filter**, **Aggregate**, and **Augment** transformations in more detail later in this section to explain why these commonly appear in pipelines. These pipelines could be used to answer questions such as: What are the common characteristics of all events with a certain property? (**Filter**) What are the counts and other aggregate properties of such events? (**Filter** | **Aggregate**) What events with this property have the most or least of another property? (**Filter** | **Reorder**) What is the value of other properties of interest for this subset of events? (**Filter** | **Augment**)

One research question here is: How do the tasks implied by these pipelines compare to the tasks supported by the literature on log analysis techniques? In comparison to many of the more sophisticated techniques, such as event clustering or causal analysis, these pipelines can seem simple [99, 113]. These pipelines more closely correspond to the tasks described for the Dapper or Sawzall, where users filter application traces and events, and compute aggregate properties like latency [145, 122].

Here we speculate on some possible explanations for this state of affairs: One possibility is that these transformations represent course-grained descriptions of the underlying operations being performed. This is necessary to be able to summarize the data and to describe the operations applied in a way that is meaningful to those without expert-level knowledge of the Splunk query language. However, one unfortunate consequence may be that it masks the underlying complexity and variety of activity. This might contribute to the appearance that log analysis tasks are simpler than they are in reality.

Another possibility is that many log analysis tasks may not require sophisticated machine learning or data mining approaches – the example questions we listed above are examples of this, and our survey results in Section 4.4 support this. It may also be difficult to map to published data mining and machine learning algorithms to the log analysis setting. Human intuition and domain expertise may be extremely competitive with state of the art machine learning and other techniques for a wide variety of problems – simple filters, aggregations and transformations coupled with visualizations are powerful tools in the hands of domain experts. Other reasons are suggested by user studies and first-hand industry experience [112, 158]. These suggest log analysis users may prefer interpretable, easily adaptable approaches over black-boxes that require lots of mathematical expertise. It is worth further investigating the types of log analysis techniques currently in widespread use and assess how research can better address practitioner needs.

[6]These can be thought of as one **Filter** that happened to be applied in separate consecutive stages.

An important factor determining what tasks are performed is the data being operated upon. Unfortunately, as described in Section 4.2, we lack direct access to this information. However, we can infer what type of data queries were applied to in some cases by considering queries that explicitly specify the source type.[7] In this way, we created more focused transition frequency diagrams for two commonly analyzed source types: Figure 4.6 shows the analysis applied to server access logs, used for web analytics (where tasks include measuring traffic, referrals, and clicks). Figure 4.7 shows the results on operating system event logs (where tasks include analyzing processes, memory and CPU usage). In these figures, query patterns differ from the patterns revealed by the aggregate queries; there is also less diversity. This is unsurprising and could be due to firstly to the quantity of the data. More interesting is how the query patterns in these figures differ from one another. It could also be due to the domain of the data, which could cause the types of questions asked to vary, or it could be due to the format of the data. For example web logs may have a more regular format, allowing users to avoid the convoluted processing required to normalize less structured data sources.

4.3.3 Longest Subsequences

To investigate what longer, possibly more complex, queries look like, we looked at the longest common subsequences of transformations (Table 4.4). This could help address the question raised in the previous section, of whether log analysis generally involves simpler tasks compared to the sophisticated techniques published in the literature or not. Again, we excluded **Cache** and **Macro** transformations from presentation. We again see the preponderance of **Filter**, **Aggregate**, and **Augment** transformations. Beyond that, the most striking feature is the preponderance of **Augment** transformations, particularly in the longer subsequences. These could be indicative of more complex processing tasks. To gain more insight into exactly what such sequences of **Augment** transformations are doing, we look more closely at such transformations in the following section.

4.3.4 Types of Tasks

Recall from an earlier section that three of the most common transformation types in log analysis are **Filter**, **Aggregate** and **Augment**. To find out more details about why and how such transformations are used, we clustered query stages containing these types of transformations, and then examined the distribution of transformations across these clusters. Analysis of less frequent command types is left to future work.) We performed clustering in order to discern patterns of use of query stages. We are interested in finding patterns that correspond to user goals and use cases, ultimately to use as variables in statistical models of user behavior, which could be used to create predictive interfaces and make recommendation systems. Clustering provides an alternative to manually looking through thousands

[7]Source type can optionally be specified as an argument to **Filter** transformations.

Length	*Count*	*% Queries*	*Subsequence*
2	2866	16.77	**Transform** | **Aggregate**
2	2675	6.13	**Augment** | **Augment**
2	2446	14.06	**Filter** | **Aggregate**
2	2170	12.70	**Aggregate** | **Rename**
2	1724	8.42	**Filter** | **Augment**
3	2134	12.49	**Transform** | **Aggregate** | **Rename**
3	1430	4.00	**Augment** | **Augment** | **Augment**
3	746	4.24	**Aggregate** | **Augment** | **Filter**
3	718	4.20	**Aggregate** | **Join** | **Filter**
3	717	4.20	**Aggregate** | **Project** | **Filter**
4	710	4.16	**Aggregate** | **Project** | **Filter** | **Rename**
4	710	4.16	**Transform** | **Aggregate** | **Augment** | **Filter**
4	694	2.71	**Augment** | **Augment** | **Augment** | **Augment**
4	472	2.73	**Filter** | **Augment** | **Augment** | **Augment**
4	234	1.37	**Augment** | **Augment** | **Augment** | **Project**
5	280	1.62	**Filter** | **Augment** | **Augment** | **Augment** | **Augment**
5	222	1.30	**Augment** | **Augment** | **Augment** | **Augment** | **Project**
5	200	0.61	**Augment** | **Augment** | **Augment** | **Augment** | **Augment**
5	171	1.00	**Augment** | **Augment** | **Augment** | **Augment** | **Filter**
5	167	0.98	**Filter** | **Augment** | **Augment** | **Augment** | **Aggregate**
6	161	0.94	**Augment** | **Augment** | **Augment** | **Augment** | **Filter** | **Filter**
6	160	0.94	**Augment** | **Augment** | **Filter** | **Filter** | **Filter** | **Augment**
6	160	0.94	**Augment** | **Augment** | **Augment** | **Filter** | **Filter** | **Filter**
6	148	0.87	**Filter** | **Augment** | **Augment** | **Augment** | **Augment** | **Filter**
6	102	0.60	**Augment** | **Aggregate** | **Augment** | **Augment** | **Augment** | **Augment**

Table 4.4: Longest common subsequences of transformations along with count of how many times such sequences appeared, and the percent of queries they appeared in.

of examples to find patterns. Manual coding techniques (i.e., content analysis) would likely have yielded similar conclusions, but would have been more time-consuming.

Clustering these transformations reveals subtypes of each transformation, and thus more details on what tasks they were used for. This combats the problem of oversimplification we introduced by using course-grained categories to understand transformation patterns, which we noted earlier.

We investigate the following sets of questions: What are the different ways **Filter**, **Aggregate**, and **Augment** transformations are used? Can we identify higher-level tasks and activities by identifying related clusters of transformations? Do these clusters allow us to identify common workflow patterns? What can we infer about the user's information needs from these groups? How well do the *commands* in the Splunk query language map to the *tasks* users are trying to perform? What implications do the clusters we find have on data transformation language design?

To cluster each set of transformations, we:

(1) Parsed each query (see: Section 4.2)

(2) Extracted the stages consisting of the given transformation type
(3) Converted the stages into feature vectors (see: Appendix B)
(4) Projected these feature vectors down to a lower dimensional space using PCA
(5) Projected these features further down into two dimensions, to allow visualization of the clusters using t-SNE [96]
(6) Manually identified and labeled clusters in the data

Then, to count the number of transformations in each cluster, we use a random sample of 300 labeled examples from the clustering step to estimate the true proportion of stages in each cluster within 95% confidence intervals.[8]

4.3.4.1 Filtering

Filter stages primarily consist of the use of the `search` command, which almost all Splunk queries begin with, and which allows users to both select events from a source and filter them in a variety of ways. We clustered all distinct **Filter** stages and discovered 11 cluster types using 26 features (Figure 4.8). Some of the clusters overlap, in that some examples could belong to more than one group. We discuss how we resolve this below.

The most common application of **Filter** is to use multi-predicate logical conditions to refine an event set, where these predicates are themselves filters of the other types, such as those that look for matches of a given field (e.g., `search status=404`), or those that look for any event containing a specified string (e.g., `search "Authentication failure for user: alspaugh"`). This use of **Filter** supports our earlier supposition that these are used to very flexibly and easily select potentially complex-to-specify subsets of data. When a **Filter** could go into multiple categories, it was placed into this one, which also contains **Filter**s with many predicates of the same type in a statement with many disjunctions and negations. Thus, it is the largest category. Considering each filter predicate individually might be more informative; we leave that to future work.

Another common **Filter** pulls data from a given source, index, or host (like a `SELECT` clause in SQL). These resemble **Filter**s that look for a match on a given field, but return all events from a given source rather than all events with a specific value in a specific field.

Other types of filters include those that deduplicate events, and those that filter based on time range, index, regular expression match, or the result of a function evaluation on the fields of each event. Lastly, some **Filter** transformations include the use of macros, others, the use of subsearches, the results of which are used as arguments to further constrain the current filter.

These use cases reveal several things:

- It is helpful to be able to simultaneously treat log data both as structured (field-value filters, similar to SQL `WHERE` clauses) and as unstructured (string-contains searches,

[8] Assuming cluster distribution is multinomial with k parameters $p-i$ we use the formula $n = \frac{k^{-1}(1-k^{-1})}{(.05/1.96)^2}$ (which assumes each cluster is equally likely) to estimate the sample size required to estimate the true parameters with a 95% confidence interval. The maximum required size was 246.

similar to `grep`).

- Some commands in Splunk, like `search`, are heavily overloaded. A redesign of the language could make it easier to identify what users are doing, by bringing the task performed and the command invoked to perform it more in line with one another. For example, there could be a distinct command for each task identified above. This might also form a more intuitive basis on which to organize a data transformation language or interface, but would need to be evaluated for usability.
- Though it may appear that time range searches are not as prevalent as might have be suspected given the importance of the time dimension in log data, this is because the time range is most often encoded in other parameters that are passed along with the query. So time is still one of the most important filter dimensions for log analysis, but this is not reflected in these results.

4.3.4.2 Aggregating

We discovered five **Aggregate** cluster types using 46 features (Figure 4.9). The most common **Aggregate** command is `stats`, which applies a specified aggregation function to any number of fields grouped by any number of other fields and returns the result. Most often, commonplace aggregation functions like `count`, `avg`, and `max` are used. Almost 75% of **Aggregate**s are of this type. Another 10% of **Aggregate**s do this, but then also prepare the output for visualization in a a chart rather than simply return the results (see the "Visualization" tab discussed in Section 4.1). Another common type of **Aggregate** is similar to these, but first buckets events temporally, aggregates each bucket, and displays the aggregated value over time in a histogram. Another type first aggregates, then sorts, then returns the top N results (e.g., `top user`). The last type groups by time, but not necessarily into uniformly sized buckets (e.g., when forming user sessions).

The takeaways from this are:

- Visualizing the results of aggregations is reasonably popular, though much of the time, simply viewing a table of the results suffices. Aggregations lead to the types of relational graphics that many people are familiar with, such as bar and line graphs [155]. Users might also appreciate having the ability to more easily visualize the result of **Filter** transformations as well; for example, using brushing and linking.[9]
- For log analysis, when visualization is used, it is more likely to visualize an aggregate value over buckets of time than aggregated over all time.

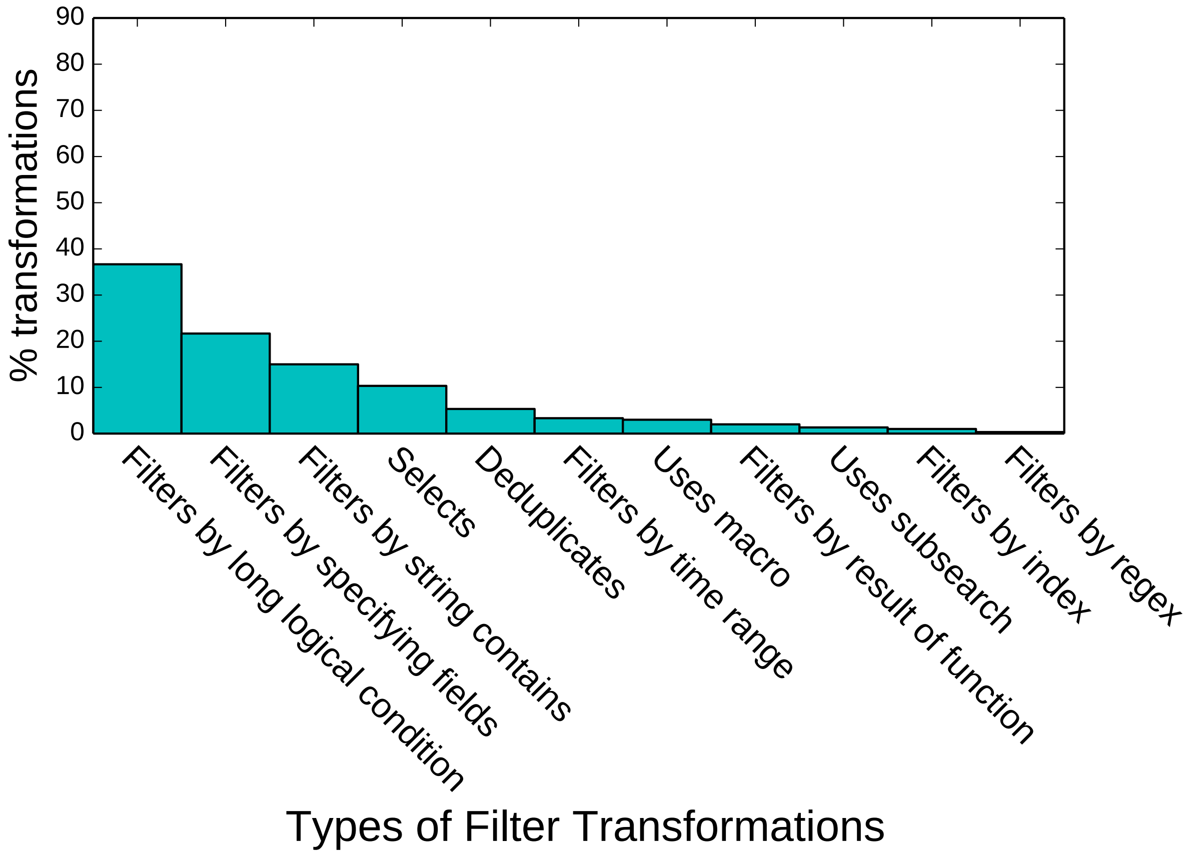


Figure 4.8: Distribution of different types of **Filter** transformations.

4.3.4.3 Augmenting

Augments add or transform a field for each event. The most commonly used such command is `eval`, which is another example of a heavily overloaded command. We discovered eight classes of **Augment** use by clustering over 127 features (Figure 4.10). These classes shed light onto the results of Section 4.3.2 and reveal what some of the long pipelines full of **Augment** transformations were likely doing.

The most common ways users transform their data are, for example:

- By manipulating strings: `eval name=concat(first, " ", last)`
- Conditionally updating fields: `eval type=if(status>=400, "failure", "success")`
- Performing arithmetic: `eval pct=cnt/total*100`

[9]Brushing and linking is an interactive visualization technique wherein multiple views of data are linked and data highlighted in one view (i.e., a filter) appears also highlighted in the other view (i.e., a bar graph or heat map).

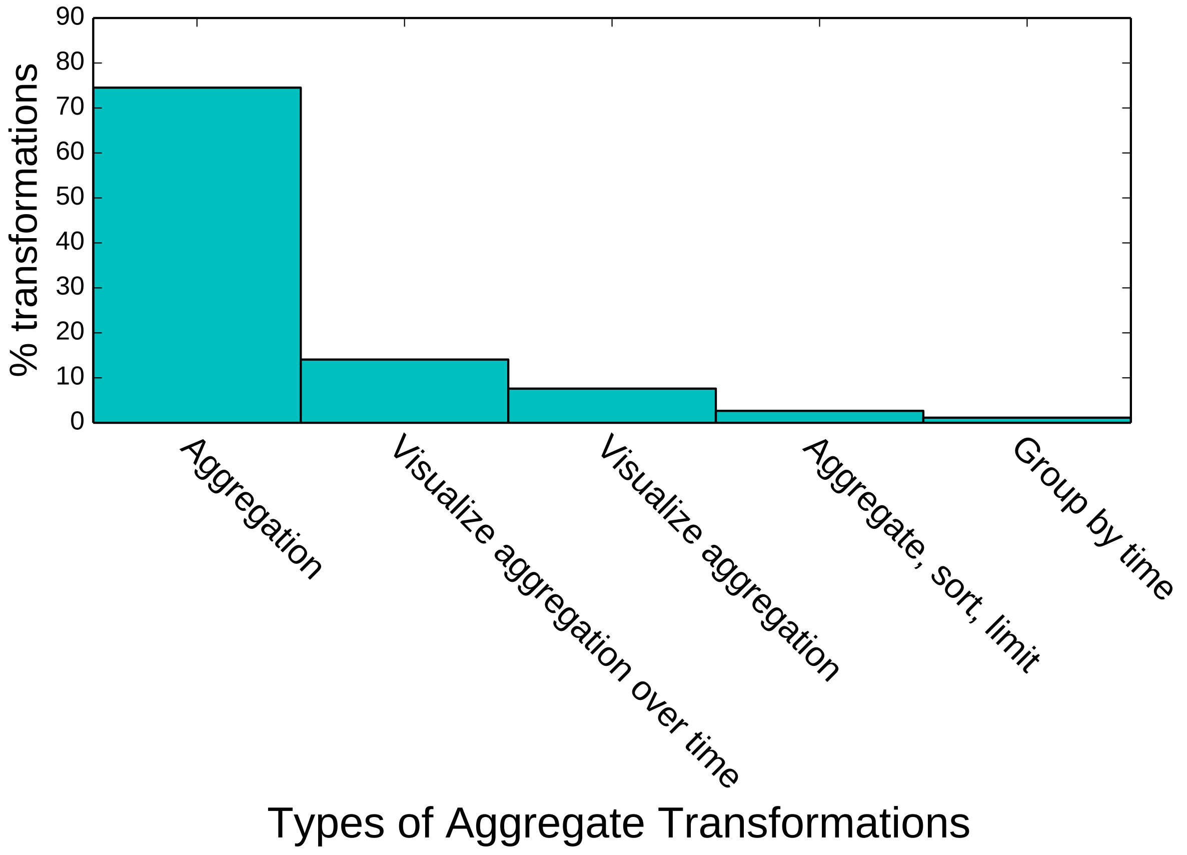


Figure 4.9: Distribution of different types of **Aggregate** transformations.

- Calculating datetime information: `eval ago=now()-_time`
- Applying multi-valued operations: `eval nitems=mvcount(items)`
- Or simple value assignments: `eval thresh=5`

Other `Augment` operations add a field that indicates which group an event belongs to and still others use the results of a subsearch to update a field.

These tasks reveal that:

- Aside from filtering and aggregation, much of log analysis consists of data munging (i.e., translating data from one format into another, such as converting units, and reformatting strings), as we discussed earlier. This is supported by other studies of data analysis in general [75]. Such data munging transformations could be mined to create more intelligent logging infrastructure that outputs data in form already more palatable to end-users, or could be incorporated into an automated system that converts raw logs

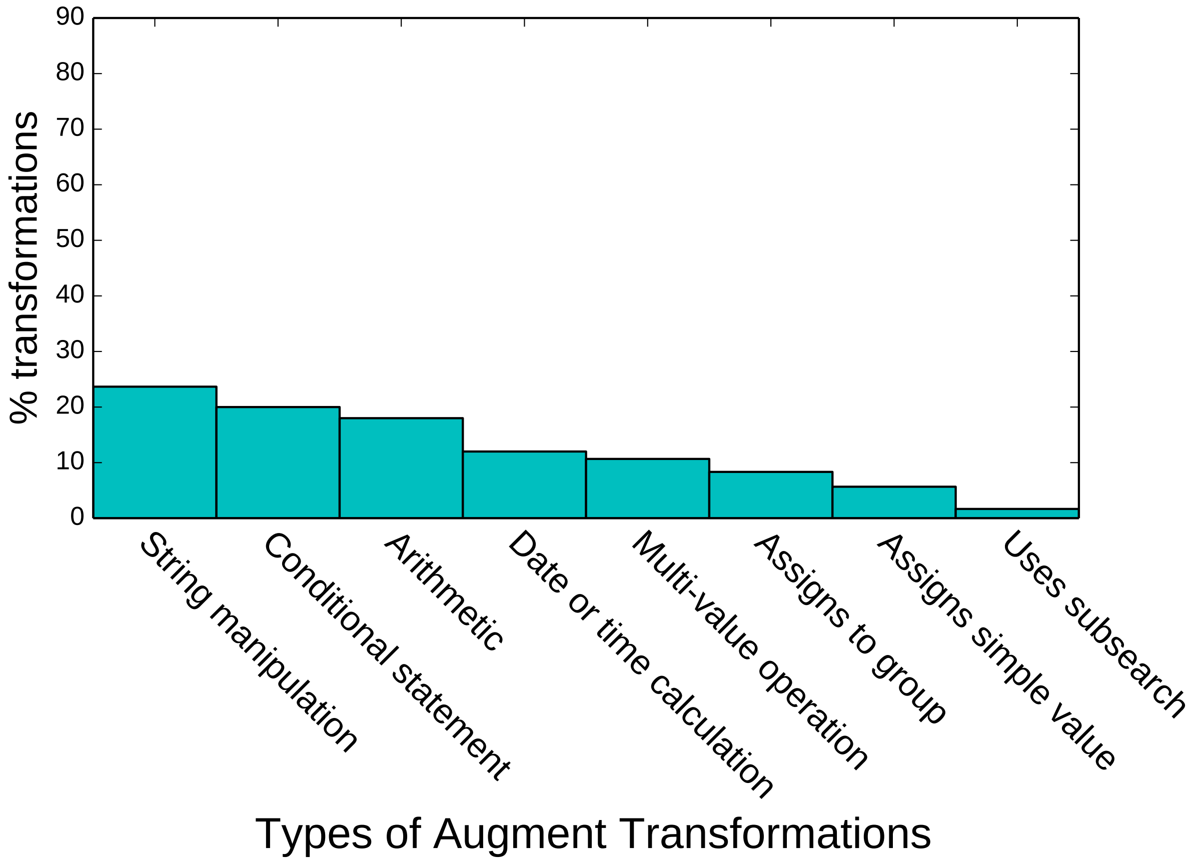


Figure 4.10: Distribution of different types of **Augment** transformations.

into nicely structured information. The more complicated transformations should be evaluated to identify whether the tool could be made more expressive.

- Just as with **Filter** transformations, here we observe heavily overloaded commands (i.e., `eval`). Refactoring functionality to clean up the mapping between tasks and commands would help here for the same reasons.
- At least 15% of cases involve the time dimension, which again suggests time is a more important consideration in log analysis than was revealed in previous sections, which makes sense given the temporal nature of event logs.

4.3.4.4 Summary

In this section, we identified different ways in which users apply each of the most commonly used transformations: **Filter**, **Aggregate**, and **Augment**. Breaking down transformations into different uses provides richer detail about log analysis tasks than could be gleaned by looking at transformation frequencies alone. For example, these details reveal the importance of the time dimension in log analysis, both for filtering and for identifying trends. This makes sense as a common use of logs is monitoring, but this was not discernible from the transformations alone. These details also provide support for the idea that an important category of tasks in log analysis is data munging, primarily via varieties of the **Augment** transformation.

In addition, these details have implications for interface design. One lesson is that overloading analysis commands to perform many different tasks makes understanding the analysis process more difficult. Refactoring these commands to map better to the separate tasks users perform would make this easier. It may also benefit the users of the analysis commands by improving usability. In this process, the syntax of queries to perform common tasks could be streamlined to make these tasks easier to accomplish. Finally, the absence of certain tasks, particularly those which have been identified elsewhere in published taxonomies, point to areas where the tool might need to be extended to provide support for a wider range of activity. For example, commands that allow users to examine the relationship between two quantitative variables (except when one of the variables is time) were so rarely used that they did not make it into our transformation categorization. The details in this section indicated no such activity either. This could be because this type of task is not common in log analysis. Alternatively, this task might need to be better supported by the tool.

Thus, not only do the additional details provide a more nuanced picture of log analysis tasks, they also provide additional lessons for log analysis tool designers. The next section continues with this theme of providing additional details of log analysis activity. It also provides a different perspective, by presenting data from a different source, specifically, a survey of Splunk sales engineers.

4.4 Survey on Splunk Usage

The analytic results suggest questions about tasks and goals that can best be answered by talking to the people who use the system. Specifically, what professional roles and technical background do Splunk users have? What are the main use cases Splunk users encounter, and what data and tools do they use in these situations? To answer these questions, we administered a survey to Splunk sales engineers. Their responses describe the use cases, data sources, roles, and skill sets of 39 customer organizations. Note: these are not responses directly from customers, rather each sales engineer answered each question once for each of three customers, based on their firsthand knowledge and experience working with those customers. Figure 4.11 summarizes the results visually.

4.4.1 Survey Results

The main results are:

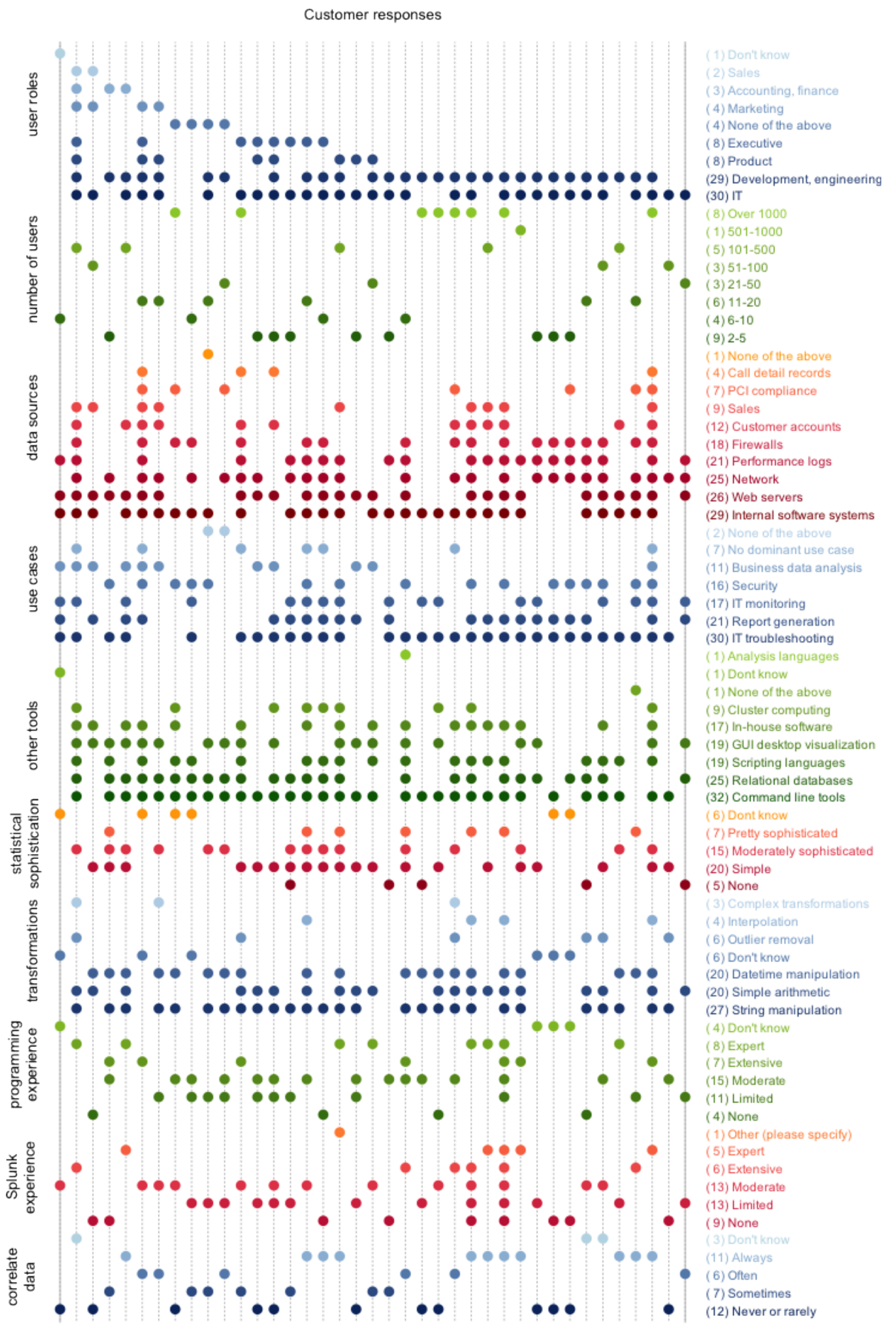


Figure 4.11: Summary of survey answers. Each vertical line represents a customer. Each colored grouping represents a different question and each row in the group represents one possible response to that question. A dot is present along a given column and row if the option corresponding to that row was selected for the question in that group, for the customer in that column.

- **User roles:** The bulk of Splunk users are in IT and engineering departments, but there is an important emerging class of users in management, marketing, sales, and finance. This may be because more business divisions are interleaving one or more machine generated log data sources for business insights. The technical aptitude of this class of users might explain the prevalence of scheduled queries and **Cache** and **Macro** operations.

- **Programming experience:** Although most Splunk users are technically savvy, most only have limited to moderate amounts of programming experience.

- **Splunk experience:** Surprisingly, many of the customers reported on did not consistently have expertise with Splunk, in fact, some users had no Splunk experience. This may be an artifact of the fact that the survey respondents were sales engineers, who may have opted to reply about more recent or growing customer deployments.

- **Use cases:** Along with the main user roles, the main use cases are also IT-oriented, but, consistent with the other responses, Splunk is sometimes used to analyze business data.

- **Data sources:** Correspondingly, the main type of data explored with Splunk is typical IT data: logs from web servers, firewalls, network devices, and so on. However, customers also used Splunk to explore sales, customer, and manufacturing data.

- **Transformations applied:** Customers primarily use Splunk to extract strings from data, perform simple arithmetic, and manipulate date and time information. In some cases, customers perform more complex operations such as outlier removal and interpolation. This provides additional context to supplement our earlier speculation about the types of tasks involved in log analysis and how often sophisticated techniques are used. This suggest that the sophistication of techniques required varies over customers and the tasks they need to perform, though the bulk of tasks are relatively simple.

- **Statistical sophistication:** Customers generally do not use Splunk to perform very complicated statistical analysis, limiting themselves to operations like computing descriptive statistics and looking for correlations. In one instance, a customer reported having a team of "math junkies" that exported data out of Splunk, ran "very sophisticated batch analytics," and then imported those results back into Splunk for reporting. The same observations made for the previous point apply here as well.

- **Data mash ups:** The degree to which customers combine data sources in their analysis varies across individual users and organizations. Some organizations almost always combine data sources for their analysis while a nearly equal number almost never do. This could be in part due to diversity in Splunk expertise and use cases.

- **Other tools:** To better understand the ecosystem in which Splunk exists, we asked what other data analysis tools customers used. In keeping with their IT-oriented roles and use cases, command line tools are frequently used by most Splunk users, in addition to databases, scripting languages, and desktop visualization tools like Tableau. A significant number of customers used custom in-house applications for analyzing their data. A relatively small number used cluster computing frameworks or analysis languages like MATLAB.

Based on these results, we make the following observations.

- IT and engineering professionals will be increasingly called upon to use their expertise working with machine data to aid other business divisions in their information-seeking needs, and will gain some expertise in these other domains as a result (deduced from user role and use case data).
- Classic tools of the trade for system administrators and engineers will be increasingly picked up by less technical users with other types of training, causing an evolution in both the features offered by the tools of the trade as well as the skills typically held by these other users (deduced from user role data). Although it is likely that more people in a growing variety of professions will learn how to program over the coming years, the market for log and data analysis tools that do not require programming experience will likely grow even faster (deduced from programming experience data).
- There is still no "one stop shop" for data analysis and exploration needs – customers rely on a variety of tools depending on their needs and individual expertise (based on the other tools data). This may be due to the existence of a well-established toolchain where different components are integrated into a holistic approach, not used disparately. Better understanding of which parts of different tools draw users would help both researchers and businesses that make data analysis products understand where to focus their energies.

4.5 Discussion

Here we summarize our main observations.

- **Filtering:** In our observations, a large portion of log analysis activity in Splunk consists of filtering. One possible explanation is that log analysis is often used to solve problems that involve hunting down a few particular pieces of data – a handful of abnormal events or a particular record. This could include account troubleshooting, performance debugging, intrusion detection, and other security-related problems. The prevalence of filtering could also reflect the importance of monitoring as an application of log analysis, which would involve keeping a lookout for particular events. Another possible explanation is that much of the information collected in logs, such as that used for debugging during development, is not useful for all users of the system. In other words, logs include many different types of data logged for many different reasons, and the difference between signal and noise may depend on perspective. Filtering is also used in conjunction with aggregation to count occurrences of different things, which is an important task in log analysis.

- **Reformatting:** Our analysis of **Augment** transformations suggested that most of these transformations were for the purpose of data munging, or reformatting and cleaning data. The prevalence of reformatting as a portion of log analysis activity is likely reflective of the fact that much log data is structured in an inconsistent, ad hoc manner. Taken together, the prevalence of filtering and reformatting activity in Splunk suggest that it may be useful for system developers to collaborate with the end users of such systems to ensure that data useful for the day-to-day management of such systems is collected. Alternatively, another possible explanation is that the Splunk interface is not as useful for other types of analysis. However, other reports indicate that indeed, much of data analysis in general does involve a lot of data munging [75].

- **Summarizing:** We observed that it is common in Splunk to **Aggregate** log data, which is a way of counting and summarizing it. Summarizing is a frequently-used technique in data analysis in general, and is used to create some of the more common graph types, such as bar charts and line graphs [155]. This suggests it may be useful to automatically create certain types of summarizing to present to the user to save time. In log analysis with Splunk, summarizing with respect to the time dimension is an important use case.

- **Time:** Since time is a default parameter to many commands that is set through buttons in the interface rather than explicitly via the command invocation, it was difficult to get a true picture of how important this dimension was in log analysis tasks. However, our detailed cluster analysis in Section 4.3.4 indicated that time was involved in several tasks, such as filtering, manipulating the time dimension, and viewing trends in aggregate data over time. At least 15% of **Augment** cases involve the time dimension, which again suggests time is an important consideration in log analysis, which makes sense given the temporal nature of event logs. In total, time is explicitly used in about 4% of stages, which is probably high relative to other non-temporal fields, given how many others there are. However, as mentioned, that is likely a vast underestimate, since filters on time can be specified through the GUI instead of through the queries.

- **Task complexity:** We were not able to determine whether Splunk users make use of some of the more advanced data mining techniques proposed in the literature, such as techniques for event clustering and failure diagnosis [99, 113]. One possible explanation for this is that due to the complexity and variability of real world problems, as well as of logged information, designing one-size-fits-all tools for these situations is not feasible. Alternatively, such analyses may occur using custom code outside of Splunk or other analytics products as part of a large toolchain, into which we have little visibility.

This idea is supported by some of the Splunk survey results (Section 4.4). Other possible explanations include lack of problems that require complicated solutions, lack of expertise, or requirements that solutions be transparent, which may not be the case for statistical techniques. It could also be the case that such techniques are used, but are drowned out by the volume of data munging activity. Finally, it may be that we simply were not able to find more complex analytics pipelines because programmatically identifying such higher-level activities from sequences of smaller, lower-level steps is a difficult problem.

- **Users and use cases:** Our survey results show that log analysis is not only for IT departments; non-technical users need to extract business insights from logs to drive their decision making as well.

4.6 Conclusion

In this chapter we presented detailed, quantitative data describing the process of log analysis. This assessment follows in a long tradition of analyzing logs of user commands to determine usage patterns and compare these with the underlying assumptions for system and interface design [154]. In addition, we provide qualitative survey data for high-level context. Together these are important sources of information that can be used to to inform product design, guide user testing, construct statistical user models, and even create smart interfaces that make recommendations to users to enhance their analysis capabilities.

In the next chapter, we analyze logs from another data analysis system, Tableau. In our Tableau case study, we used augmented logging capabilities implemented by our collaborators at Tableau to obtain additional context about the analysis environment at every step the user takes – namely, the analyzed data and the application state. We also prompted participants to annotate their log activity to indicate the tasks they were performing at the time, as well as collected their reports on their analyses, so we have some insight into their thought processes. Like in this chapter, we group similar activity to identify tasks; however, due to the different nature of the datasets (queries versus events) and the additional data we collected, in the next chapter we take a different approach and group the events of each user into consecutive subsequences. This allows us to get a higher-level summary of each user's entire analysis session.

Chapter 5

Analyzing Full Annotated Activity

In this chapter, we present a usage case study of data logged from Tableau.[10] Tableau is a software application for visualizing relational data for analysis. It was initially developed in 2000 as a research project called Polaris, [151]. It provides a graphical interface that lets users create, via simple dragging and dropping, visualizations of information stored in their databases. By using Tableau to transform and plot, users can answer questions about their data. For example, a bookstore owner may have a database of sales information that describes the products they have sold. Using Tableau, the user can create a graph that shows the number of children's books customers purchased each month of the past five years.

Because Tableau lets users create attractive and useful graphs of data quickly, without writing code or having deep knowledge of database technologies, it is valuable to many people of different expertise in different contexts. This makes it a popular tool with a large user base, and as such, many other data analysis products on the market have mimicked aspects of Tableau's interface for visualizing data. Thus, among the large set of data analysis and visualization tools, Tableau is a good case to study, being arguably representative of its class.

In studying Tableau, we want to identify quantitative descriptions of how it is used, along with methods for deriving these descriptions, which can one day be incorporated into qualitative and statistical models. The applications of these models were described in detail in Chapter 1, and include providing an empirical basis for UI/UX studies, as well as providing input into increasingly automated and intelligent analysis interfaces, including AI assistants and recommender systems. Though this information is a starting point for these other applications, it is critical to have, and not trivial to come by, for reasons detailed in Chapter 3.

We obtained this information by first collecting several dozen highly-detailed, user-

[10] Special thanks for the contributions to this work presented in this chapter by Marti Hearst, faculty at the University of California, Berkeley, and Melanie Tory and Heidi Lam, researchers at Tableau. In this chapter, the terms "we" and "our collaborators at Tableau" refer to them. Heidi Lam aided by augmenting Tableau's logging capabilities and developing the annotation tool. Marti Hearst helped design and facilitate our data collection studies. Marti Hearst and Melanie Tory performed the submission segmentations we analyze in depth in this chapter. Their assistance was invaluable.

annotated Tableau event logs. These logs were part of student submissions for an exploratory data analysis assignment in a graduate-level information visualization course at the University of California, Berkeley. For this assignment, participants chose their analysis topics and datasets individually, according to their interests. Participants were asked to first pose several hypotheses about their data, then use Tableau to explore their theories, using the techniques and strategies presented in the course. The datasets and hypotheses that participants analyzed varied widely. Participants each wrote a report detailing their findings, then annotated their logs to reflect upon and describe their analysis process. The participants submitted their report, workbooks, logs, and annotations, all of which we collected.

We investigated ways to segment the participants' low-level event streams into consecutive subsequences of similar events. The purpose of this segmentation is to facilitate the summary of the activities the logs record. These summaries might be then consumed more effectively by end-users and other applications. We developed an event log visualization tool to help human segmenters develop segmentations, which we in turn used to derive segmentation heuristics. We then analyzed and summarized these segments. The participants' reports and annotations serve as a source of ground truth to compare with these automatically generated summaries.

In developing our segmentation and summary approach, we first deep-dive into one submission. Our collaborators segmented this submission manually, which yielded several examples on which to base segmentation heuristics. We analyzed these manual segmentations to determine the degree of agreement between them and reconciled differences where possible. When analyzing the manual segmentations, we found that they did not agree with one another to a high degree – the best case amount of overlap between two segmentations was 83% and the worst case was only 19%, even though we provided instructions to attempt to guide segmenters to using the same approach. We speculate that one reason for this is that segmentation involves a lot of subjectivity, and that likely, different segmentations would be better for different uses. However, it also could be because we only had a few segmentation examples, and did not iterate on the segmentation instructions. Nonetheless, the segmentations do share some properties that distinguish them from randomly-generated segmentations, which suggests that these segmentations are not completely meaningless groupings of the event stream. Namely, the events within a segment tend to be more similar to one another in certain respects than they are to events in adjacent segments.

Thus, based on these findings, we created heuristics for segmenting the event stream. We tested the heuristics on the submission that was manually segmented, and found that the segmentations created with these heuristics exhibited characteristics similar to the manual segmentations. We then used these heuristics to segment all the submissions. We generated simple summaries of these segments and investigated some simple general questions, like what is the most common type of activity, and what are some common analysis workflows? In asking these questions, we seek to demonstrate the types of questions that can be investigated rather than definitively address certain usage hypotheses. Moreover, the answers to these questions were constrained in our case by the fact that the data we collected was from students following particular instructions in an assignment that directed them to follow this

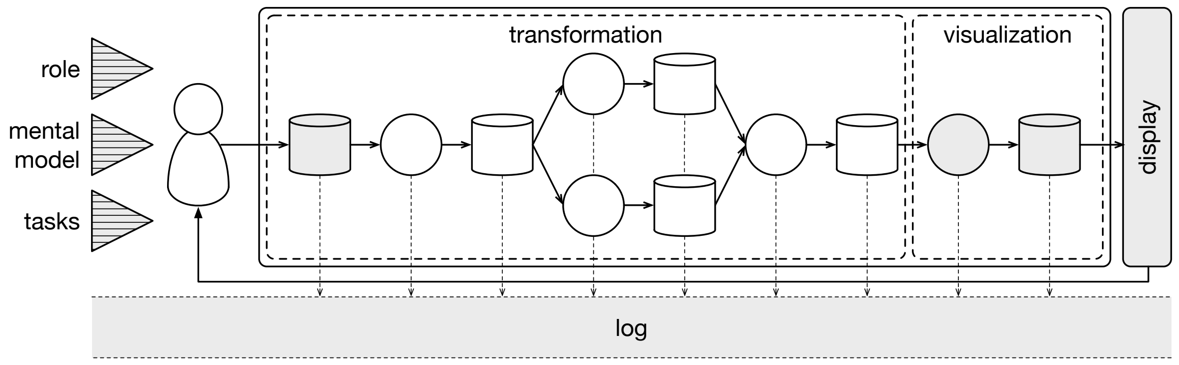


Figure 5.1: This figure shows a diagram of the instrumented analysis process. The cylinders represent data and the circles represent transformations applied to that data. In the Tableau case study, we collected information about the *data* the users analyzed, the *visualization operations* that users applied and the *visualizations* they created, as well as information about their *goals*. These are highlighted in grey in the figure. We don't have any information about the *transformation operations* they applied to their data outside of visualization beyond a high-level description.

workflow, and so these answers are not representative of other use cases. Nonetheless, in our dataset, we found that the most common activity is presentation (edit dashboard) both by time and by segment count, and this is true across all submissions. This makes some sense, given the assignment that instructed participants to create dashboards of each of their findings. After that, segments that are a mix of categories and analysis segments (edit vis) are the most common. We also found that the submissions tend to follow the workflows proposed in the literature: first data munging, then analysis, then presentation, though there is some iterating back and forth. Also, activities like pan, zoom, and highlight, and filter, tend to happen more in the presentation rather than the analysis phase. Though these findings do not generalize to Tableau or visual analytics tool usage more broadly, our approach could easily be applied to other datasets. This would enable the discovery of characteristics of various classes of use cases and ultimately of more general descriptions of human analysis behavior.

To set the stage for this analysis, we begin by describing the Tableau interface and how users interact with it in Section 5.1. We then describe our method for collecting detailed and annotated event logs from Tableau in Section 5.2. Importantly, as Section 5.1 shows, this approach allowed us to collect data about the entire analysis cycle, from the user's goals and data at each step, to the transformation and visualization of that data, and finally to

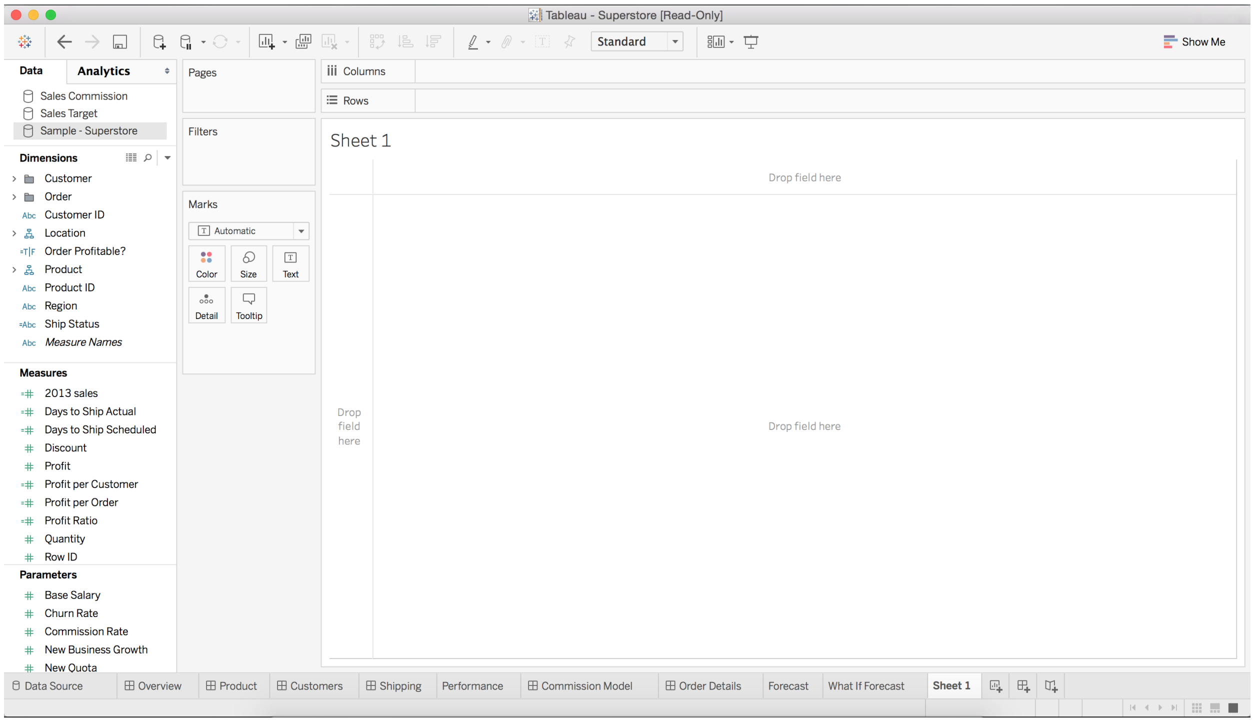


Figure 5.2: When a new sheet is created in Tableau, it is blank and there are no fields on the shelves.

the user's interpretation of the resulting visualization. This is an innovation over what is typically collected in log analysis studies and provides a substantially more complete picture of user analysis activity. We describe how we analyzed this data, and our resultant findings in Section 5.3. We discuss the implications of our results and opportunities for future work, before summarizing all of the work and findings presented in Section 5.5.

5.1 Case Study: Tableau

Tableau provides an interface that allows users to create visualizations of relational data. To do this, users drag and drop fields from their dataset (i.e., columns) from a list in a sidebar, onto a set of shelves in the main window of the interface, as shown in Figure 5.2 [11]. The placement of fields on these shelves results in an automatically generated plot, such as a bar chart or timeseries plot. The type of plot created depends on the datatype of the fields and on where the fields are placed on the shelves. The user dictates which fields are mapped to which visual dimension of the graph, such as vertical position or color, by controlling which fields are placed onto which shelves.

[11] All screenshots shown here are of Tableau 10.

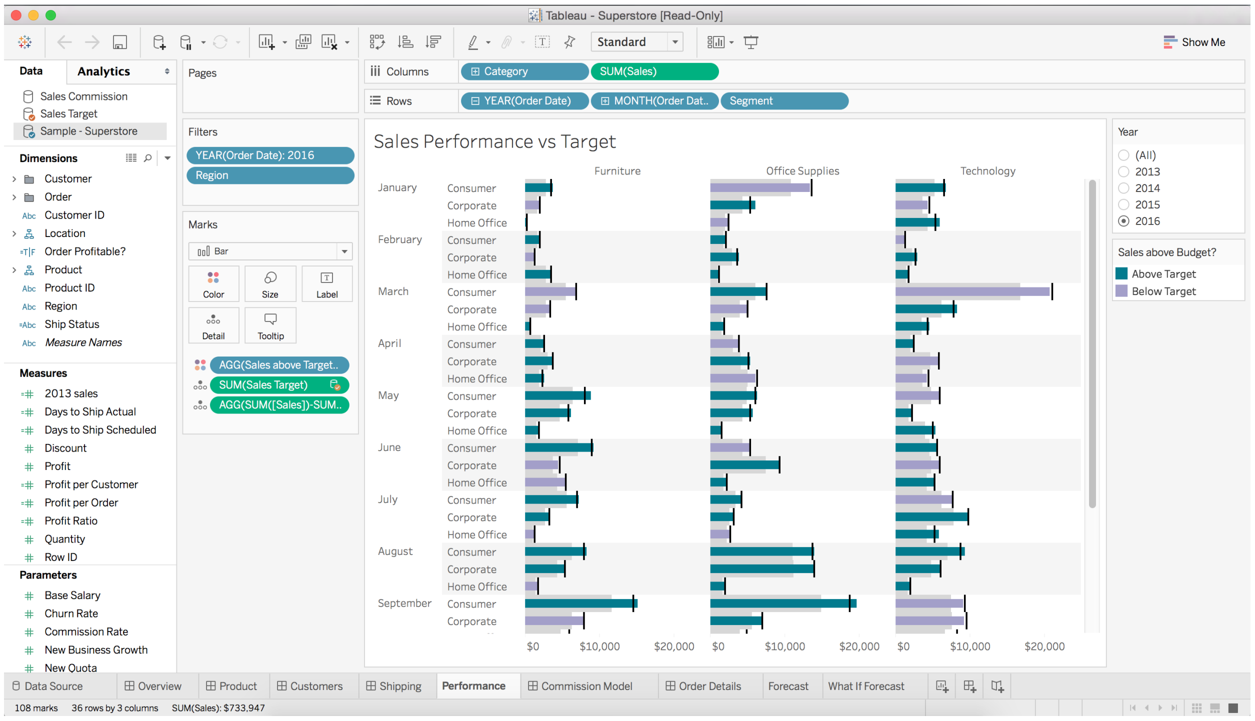


Figure 5.3: After the user drags and drops several fields onto various shelves, the Tableau sheet contains a visualization. In this visualization, sales by category (column shelf) are shown for three different industry segments over each month of the year (row shelf) for the year 2016 in a certain region (filter shelf). The bars are color-coded according to a certain criteria (details shelf).

In addition to dragging and dropping fields onto shelves to create visualizations, users can interact with their graphs to sort, filter, annotate, and more. Users can create derived fields based on calculations using other fields. Tableau supports common relational datatypes such as categorical and quantitative data, as well as geographical data and more. Users save their work in Tableau workbooks. A single workbook consists of one or more sheets, similar to sheets in spreadsheet applications, in which the users create visualizations of the data they input into the workbook, in the manner described in the previous paragraph.

A central feature of Tableau is ShowMe – a menu of possible graph types that users can select from to visualize the data on the current sheet. After selecting a graph type, Tableau automatically generates the graph for the user with the fields they selected. Users can also undo, redo, export, and perform other operations standardly supported in modern GUIs. Figure 5.3 shows a screenshot of what the interface looks like after creating several visualizations. It is beyond the scope of this chapter to describe in depth how Tableau and ShowMe operate, but there are several published papers that do so [97, 98].

5.2 Data Collected: Annotated Tableau Event Logs

In this section, we first describe what data was logged by Tableau by default. In a pilot study, we collected data from Tableau 8.0. Through a preliminary analysis of this data, we determined that these default logs lacked some important information required to accurately capture users' analysis activities, as we describe below. (This is typical, as explained in Chapter 3.) To address this, we created a set of recommendations enumerating what additional instrumentation should be added to Tableau to produce the needed information. Our next study, described here, used Tableau version 9, which improved upon the logging of the previous version; however, some important information was still missing. Our collaborators at Tableau implemented additional logging based on our suggestions, providing Tableau with the capability to produce augmented logs. We describe these augmented logs later in this section.

To collect a set of these augmented logs of real analysis activity, we ran a study. In this study, participants used Tableau with augmented logging enabled to explore a dataset of their choosing. They then annotated the logs that were recorded during their exploratory analysis with their thought processes, indicating their goals and expectations in each step of their analysis. This resulted in a final dataset of 24 augmented and annotated logs. We describe this study and the resultant data in the final portion of this section.

5.2.1 Default Tableau Logs

Tableau Desktop logs are text files on the local filesystem containing sequences of Tableau event data. Event data is appended to these files in response to most user actions and to certain internal triggers, though here we will concern ourselves with only the former. For example, when the user drags a field to a shelf, Tableau writes a new event at the end of the log. Each event includes a timestamp representing when it was written, along with what user action triggered the event, and depending on the action, certain parameter information, such as the name of the field that was dragged to a shelf and the name of the shelf it was dropped on. Listing 5.1 gives some examples of such events. A sequence of such events is called an event stream.

Listing 5.1: Events in default Tableau logs.

```
{"ts":"2016-03-13T20:36:04.832","pid":673,"tid":"6580","sev":"info","req":"-","sess":"-","
    site":"{B6A85965-6F2C-4F41-912C-7842D6329477}","user":"-","k":"command-pre","v":{"args
    ":"tabui:edit-reference-line reference-line-id=\"refline0\"","name":"tabui:edit-
    reference-line"}}
{"ts":"2016-03-13T20:36:20.071","pid":673,"tid":"6580","sev":"info","req":"-","sess":"-","
    site":"{B6A85965-6F2C-4F41-912C-7842D6329477}","user":"-","k":"command-pre","v":{"args
    ":"tabui:delete-sheet-ui worksheet-orphans-handling-mode=\"ask-user\"","name":"tabui:
    delete-sheet-ui"}}
```

If Tableau issued an event containing all the relevant parameter information every time a user performed an action, it would represents a complete record of the user's analysis activity. However, by default, some actions were not logged at the time of our study, such as when the user switches from one sheet to another. Moreover, some events of some actions that are logged did not contain important parameter information, such as sheet deletion events not listing the name of the sheet that was deleted. As a result, it is not possible to fully reconstruct what steps the user took using the default logs.

Moreover, even if the logs did record all user actions and their relevant parameter information, building a complete picture of the user's analysis activity would require keeping track of the current state of the application, because the effect of most actions depends on the current application state. For example, what visualization is shown after a user drops a field on a shelf in a sheet depends on what other fields were already on shelves in that sheet. Reconstructing this from a complete event stream would be possible; however, it would require effectively building a Tableau Desktop emulator, which would be extremely labor-intensive and error-prone. This is undesirable especially considering that a cleaner and easier solution would be to have Tableau augment the logs with the desired information about its own state. This is in fact the solution we chose; these augmentations are described in the next section.

5.2.2 Augmented Tableau Logs

To assess the quality of data available in the default Tableau logs, we conducted a pilot study in which we asked participants to use Tableau to explore a dataset and provide us with the resultant logs. By attempting to use these logs to understand what analysis steps the participants had taken, we were able to identify what additional information the logs needed to have for us to do this effectively. This information included approximately two dozen actions that were missing event or parameter information. In addition, we recommended that Tableau add the ability to log the current application state and information about the data users load into Tableau. Some of these recommendations were implemented by our collaborators at Tableau.

Because the additional information added to the logs causes them to grow quite large, the augmented logging feature is not turned on by default; we enabled it for our study, which we describe in the next section.

5.2.3 User Study and Annotated Tableau Logs

Listing 5.2: Summary of instructions to study participants.

```
1. Identify a topic area and a set of hypotheses.
2. Explore these hypotheses using Tableau, and write up your results in
    a written report and using one or more Tableau Dashboards.
3. Reflect on your process: use the annotation tool to comment on the
   process you went through. Annotate major portions of the process.
   In particular:
         a. New task or subtask: Mark events  that indicates when you
            began working on a new hypothesis, goal, task, or question.
```

```
       Examples include: creating a visualization, trying to
       understand some aspect of Tableau?s interface, reformatting
       a previously created visualization, or trying a different
       approach accomplish a previously stated goal.
    b. Mistakes: Make a note on an event if you generated it
       accidentally.
    c. Commentary: Feel free to interject commentary in the middle
       of a task or subtask
    d. Unknown: Feel free to mark a portion of the log as unknown
```

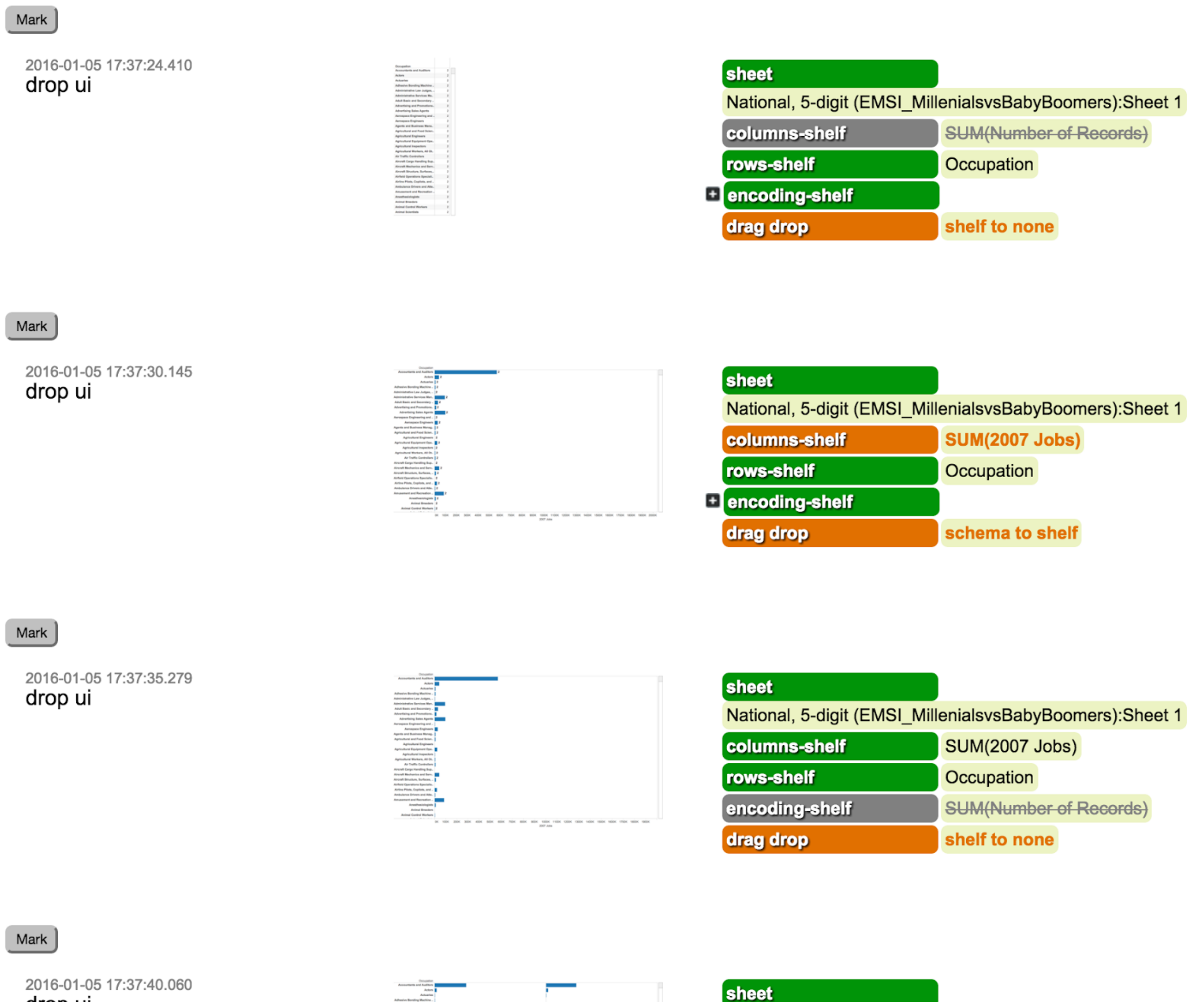
In addition to augmenting the default Tableau logs with more information, we arranged to collect annotations of these logs by the users whose actions generated them. By annotating their logs, users can provide information about the goals and plans they had as they analyzed their data in Tableau.

The augmented logs make this possible because the additional information in them – namely, the state of the application after each action – provides the ability to represent the sequence of steps users took in Tableau in a more recognizable form. This makes it feasible to ask users to review their logs and describe their motivations for and reactions to their analysis steps.

To collect a set of augmented and annotated logs to analyze, we arranged a user study with students in an information visualization course. As part of this course, the students were required to perform a detailed exploratory analysis of a dataset and hypotheses of their choosing using Tableau. For this assignment, we asked these students to use the version of Tableau with augmented logging enabled. Afterwards, they were required to reflect on their analysis process and describe their findings in a detailed report. Lastly, they were then asked to annotate their augmented logs using an annotator tool, called LogViewer, built by one of our collaborators at Tableau. LogViewer displays a subset of the events in the log, and for each of these provides three text boxes, GOAL, EXPECTED, and COMMENT, that users can enter annotations into.

Upon completion of this assignment, students submitted their Tableau workbooks, their reports, their augmented logs, and their annotations. The instructions for the assignment are summarized at a high-level in Listing 5.2. Screenshots of the annotator tool are shown in Figure 5.4, Figure 5.5, and Figure 5.6.

Students were not required to participate in our study; all students were required to follow the same instructions regardless of whether or not they opted in to the study and the list of students who opted in was kept hidden from the course staff, so that students did not feel pressured to participate. After excluding submissions from those who opted-out and those who were not able to get the augmented logging or annotation tool working for one reason or another, we collected 24 complete submissions with augmented and annotated logs. Nearly every submission involved a different analysis dataset. The number of events



Mark

2016-01-05 17:37:24.410
drop ui

sheet	National, 5-digit (EMSI_MillenialsvsBabyBoomers):Sheet 1
columns-shelf	~~SUM(Number of Records)~~
rows-shelf	Occupation
encoding-shelf	
drag drop	shelf to none

Mark

2016-01-05 17:37:30.145
drop ui

sheet	National, 5-digit (EMSI_MillenialsvsBabyBoomers):Sheet 1
columns-shelf	SUM(2007 Jobs)
rows-shelf	Occupation
encoding-shelf	
drag drop	schema to shelf

Mark

2016-01-05 17:37:35.279
drop ui

sheet	National, 5-digit (EMSI_MillenialsvsBabyBoomers):Sheet 1
columns-shelf	SUM(2007 Jobs)
rows-shelf	Occupation
encoding-shelf	~~SUM(Number of Records)~~
drag drop	shelf to none

Mark

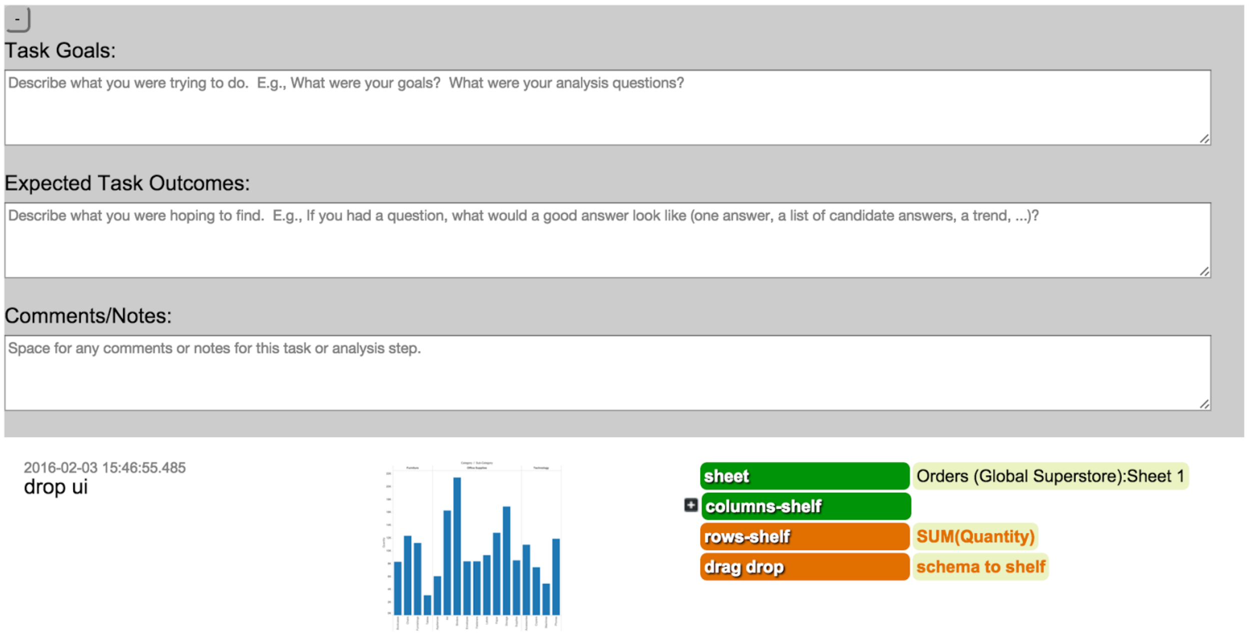
2016-01-05 17:37:40.060

sheet

Figure 5.4: Events as they are shown by the annotator tool.

Number of submissions	24
Median active duration (mins)	515.2
Median number of events (user-generated)	1,864
Median number of annotations	40

Table 5.1: Characteristics of the set of events analyzed from the Tableau submissions.



Task Goals:

Describe what you were trying to do. E.g., What were your goals? What were your analysis questions?

Expected Task Outcomes:

Describe what you were hoping to find. E.g., If you had a question, what would a good answer look like (one answer, a list of candidate answers, a trend, ...)?

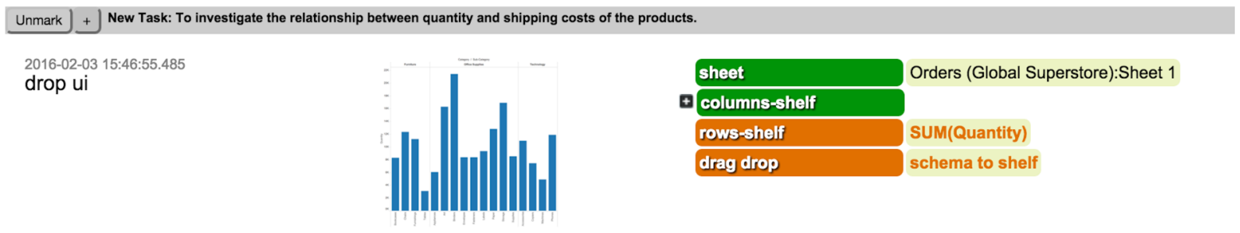
Comments/Notes:

Space for any comments or notes for this task or analysis step.

2016-02-03 15:46:55.485
drop ui

sheet	Orders (Global Superstore):Sheet 1
columns-shelf	
rows-shelf	SUM(Quantity)
drag drop	schema to shelf

Figure 5.5: Creating an annotation in the annotator tool.



New Task: To investigate the relationship between quantity and shipping costs of the products.

2016-02-03 15:46:55.485
drop ui

sheet	Orders (Global Superstore):Sheet 1
columns-shelf	
rows-shelf	SUM(Quantity)
drag drop	schema to shelf

Figure 5.6: Completed annotation in the annotator tool.

and annotations is listed in Table 5.1.

5.2.4 Submission Deep Dive

In the analysis that we present in Section 5.3.3, we first focused on a single submission that we segmented by hand for use as a basis for developing heuristics. Thus, in this section, we describe that submission, to provide context for our later analysis. In this submission, the student performed an analysis of student loan default rates and graduate debt levels. The four hypotheses they explored were:

- Hypothesis 1: Default rates did not substantially vary over the three-year period.
- Hypothesis 2: Non-degree programs have higher default rates than degree programs.
- Hypothesis 3: Private, proprietary, and for-profit institutions have higher default rates

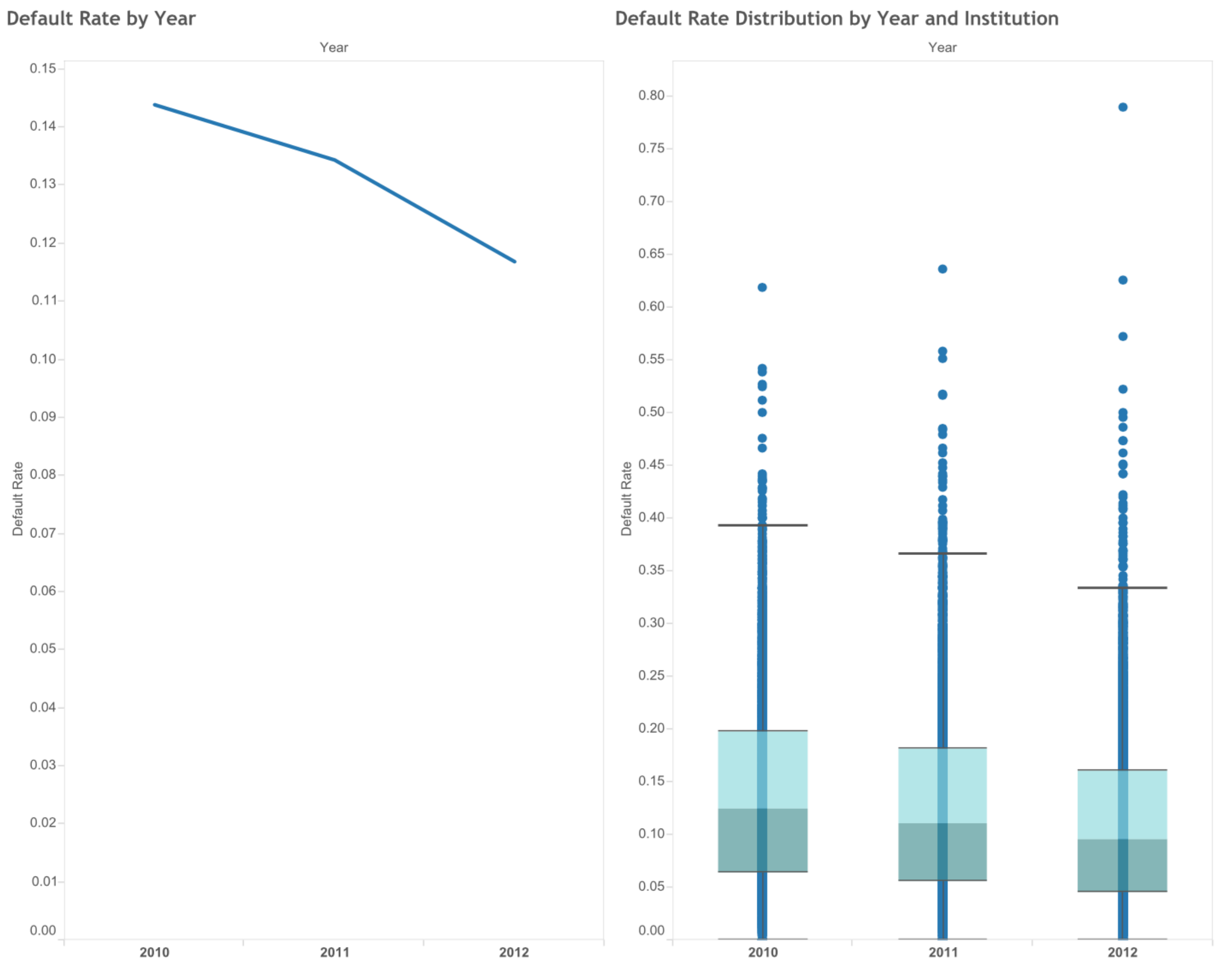


Figure 5.7: Dashboard created for "Hypothesis 1: Default rates did not substantially vary over the three-year period." The participant decided this hypothesis should likely be rejected.

than public.

- Hypothesis 4: States with higher undergraduate debt have higher default rates.

The participant investigated each hypothesis in order using the strategy identified in Shneiderman's "overview, zoom and filter, details on demand" mantra [143]. The final graphs they made present their findings on these hypotheses are shown in Figures 5.7 - 5.10.

This participant did a very thorough job annotating this submission. Figure 5.11 shows the annotations the participant made on the workbook state that was recorded when they were working on a task for Hypothesis 1. In fact, this work is the first task, and the first segment, of their submission.

This segment's annotations are shown in Figure 5.11 with the corresponding event ID, timestamp, screenshot (if available), and annotation. In the annotations, the text "3.1"

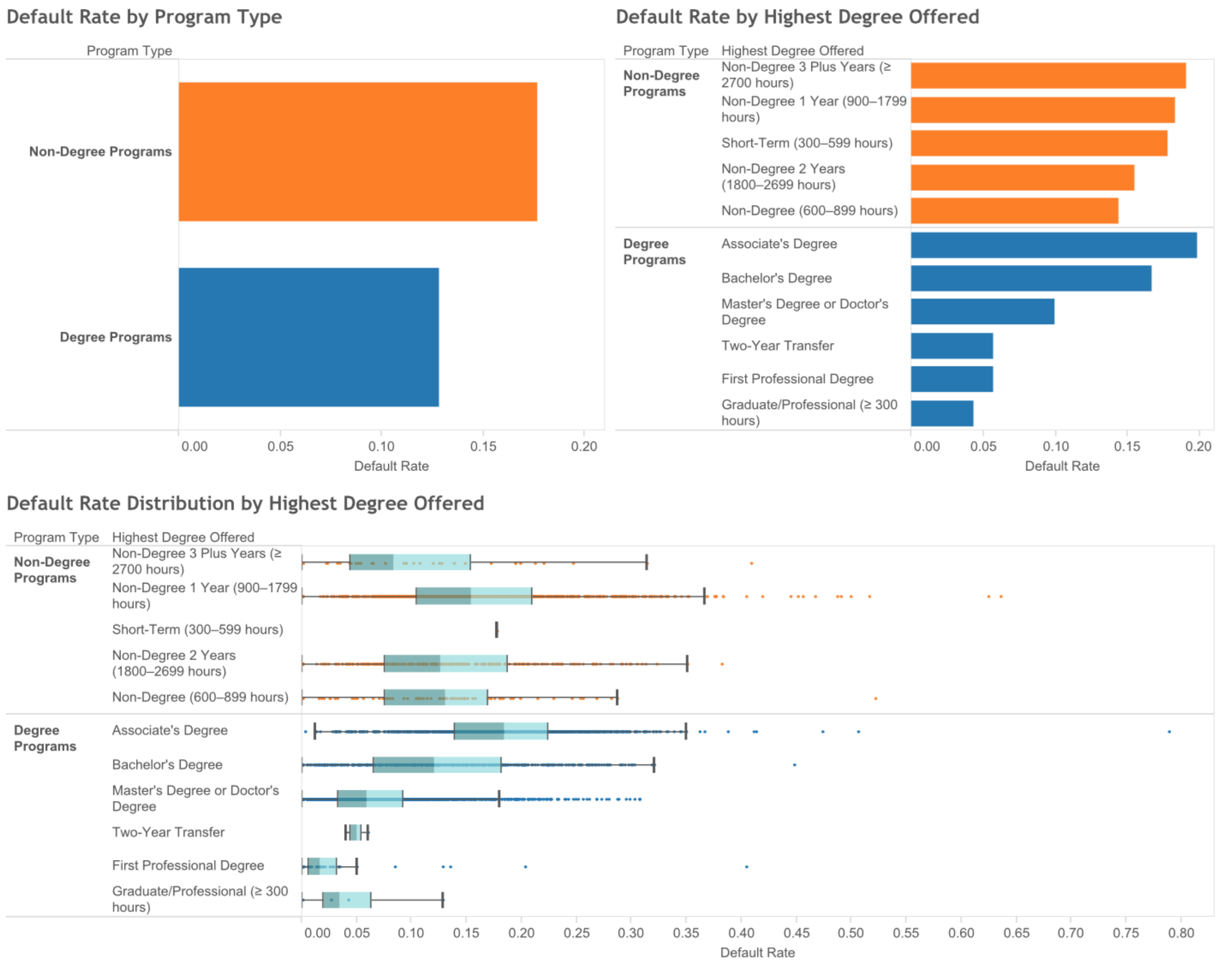


Figure 5.8: Dashboard created for "Hypothesis 2: Non-degree programs have higher default rates than degree programs." The participant decided that this hypothesis was "confirmed" by this but likely not meaningful due to not addressing the substantial variation caused by the highest degree offered.

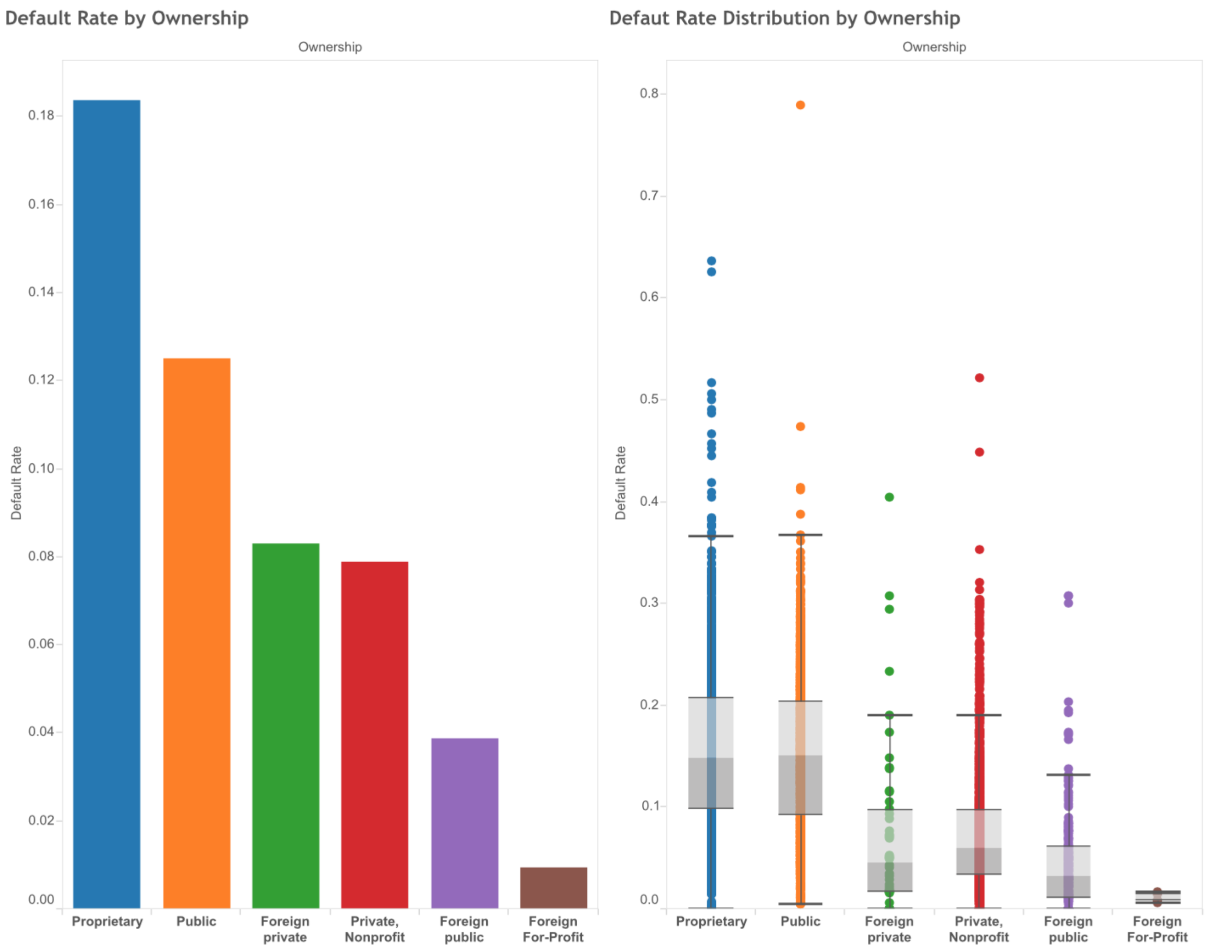


Figure 5.9: Dashboard created for "Hypothesis 3: Private, proprietary, and for-profit institutions have higher default rates than public." Based on this, the participant decided this hypothesis should likely be rejected, and thought of several other hypothesis they might investigate, such as that foreign institutions have lower default rates.

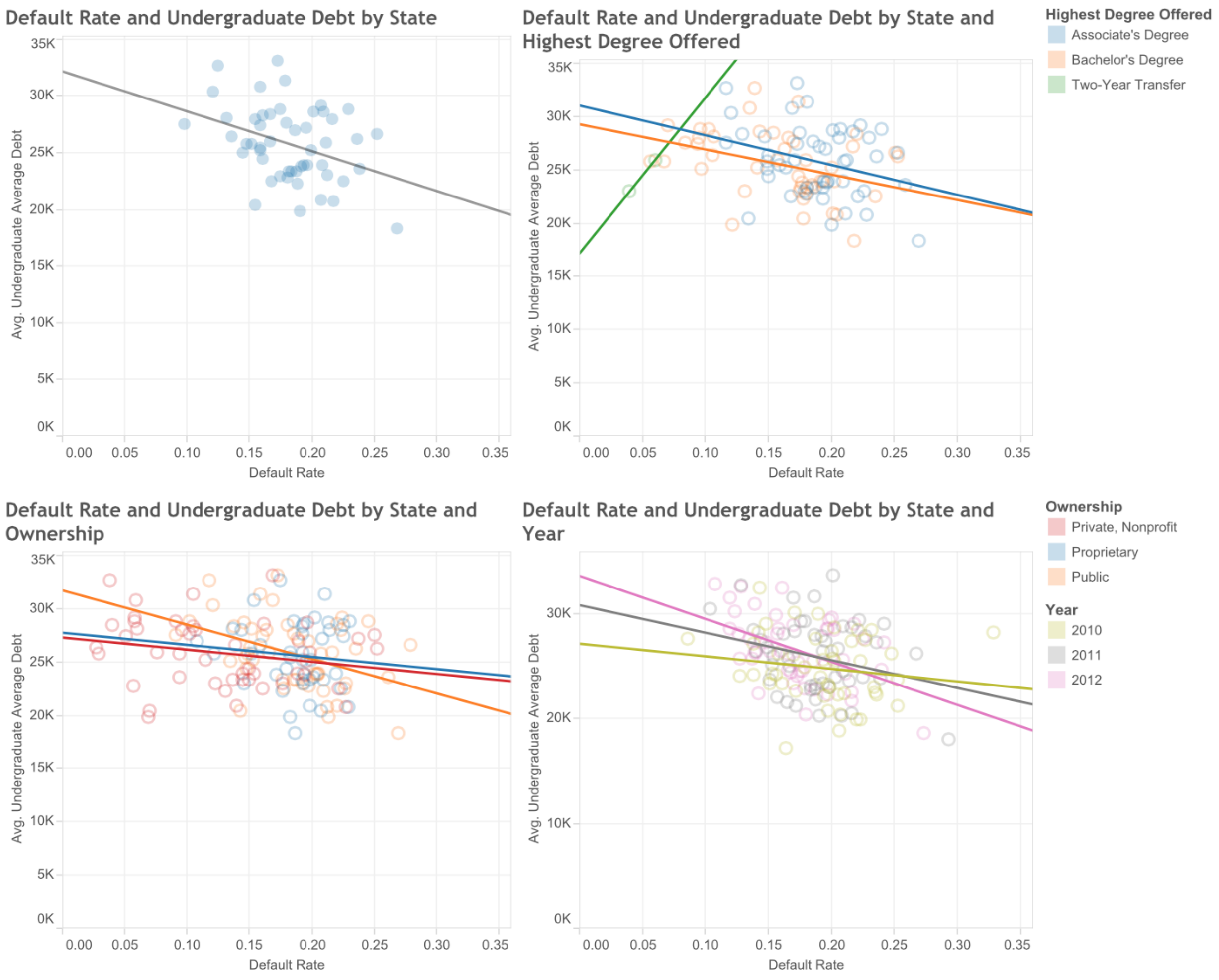


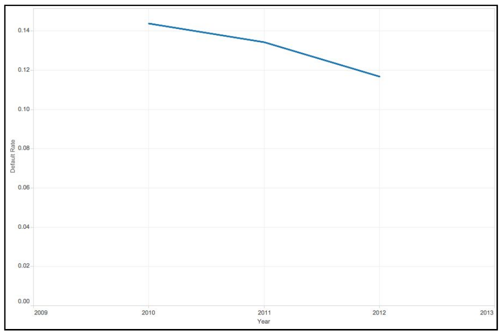
Figure 5.10: Dashboard created for "Hypothesis 4: States with higher undergraduate debt have higher default rates." Based on this, the participant concluded their hypothesis should probably be rejected, but they were unable to discover an explanation in their further explorations on this topic.

[event 0] 7:51:03 PM: tabui:get-open-pane-sample-workbooks
GOAL: [3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. Subtask: Aggregate view

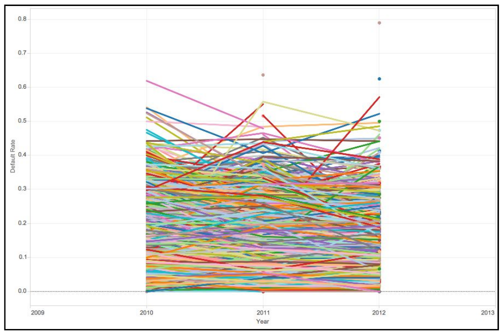
2009
2010
2011
2012
2013
Year

[event 13] 7:56:42 PM: tabui:drop-ui
GOAL: [3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. Subtask: Aggregate view;
EXPECTED: Add year to the sheld

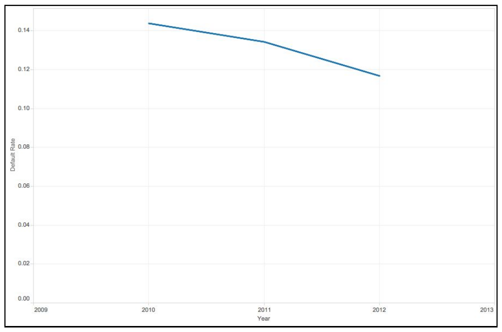
[event 15] 7:58:54 PM: tabdoc:apply-calculation
GOAL: [3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. Subtask: Aggregate view;
EXPECTED: Define the calculated field for Default Rate



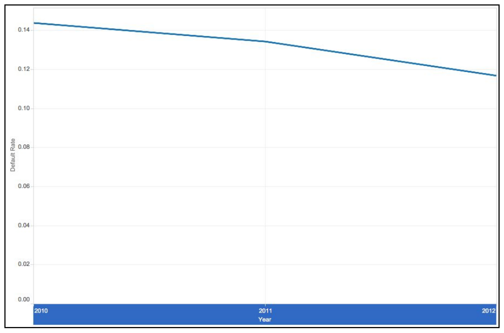
[event 16] 7:59:22 PM: tabui:drop-ui
GOAL: [3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. Subtask: Aggregate view;
EXPECTED: Plot default rate over the 3-year period



[event 17] 8:00:33 PM: tabui:drop-ui
GOAL: [3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. Subtask: Aggregate view;
EXPECTED: Get a feel for where the bulk of institutions fall in terms of default rate over the 3-year period



[event 19] 8:02:45 PM: tabui:drop-ui
GOAL: [3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. Subtask: Aggregate view;
EXPECTED: Remove institution name as the plot was no valuable for analysis due to density.



[event 20] 8:02:47 PM: tabui:edit-axis-ui
GOAL: [3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. Subtask: Aggregate view;
EXPECTED: Change the axis so the line covers the length of the chart.

[event 27] 8:05:04 PM: tabdoc:rename-sheet
GOAL: [3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. Subtask: Aggregate view;
EXPECTED: Update the sheet name to reflect the sheet contents.

Figure 5.11: The participant's annotations for Submission 16's first segment.

refers to the section of the participant's report to which the annotated work pertains. As instructed, the participant prepended all of the annotations pertaining to one task with the same description, using the "GOAL" box of the LogViewer tool. The highlighted green text is what the participant entered into the "EXPECTED" box, and is what changes from annotation to annotation.

5.2.5 Challenges and Limitations

For this case study, we were able to collect much more data than in the previous one, which helps to address some of the challenges identified in Chapter 3. For example, via our pilot study, we were able to identify a lot of important missing information, so this challenge, discussed in Section 3.1, was not as much of an issue here, though there were still a few parameters missing from some events. The issue of interaction provenance, discussed in Section 3.2, was not an issue, both because we are more interested in the higher-level activity and not the exact interface mechanisms used to perform a given action, and also because there were no issues where automation could be confused with manual interaction, as in the Splunk case study (described in Section 4.2). As in the previous case study, we did not have access to intermediate actions, a challenge discussed in Section 3.4. While information like where the user's mouse hovered could have revealed more about the user's thought process, we had a great deal of other information to process, and making sense of high volumes of data comes with its own challenges. In this case study, we did have access to a great deal of analysis state, a challenge discussed in Section 3.5. However, there is some additional information that would have been helpful to have, such as information about what visualization was displayed, in a machine-readable text format, rather than just a screenshot, and more information about the dashboarding activity. This information was not included in the augmented logging implemented by our Tableau collaborators. Likewise, we also had access to metadata about the data analyzed, such as the datatype, as discussed in Section 3.6, though information about other statistical properties of the data could allow for a richer analysis, and collecting this information from the workbooks we gathered would have been a time- and manual labor-intensive process. Lastly, we did have some information about user intent, a challenge discussed in Section 3.7. Namely, we have the annotations and reports. These are not a perfect reflection of user intent because users' necessarily did not capture their every thought, and in fact many users' annotations omitted a great deal of detail. However, it does provide some ground truth. Lastly, the challenge of identifying higher-level actions, discussed in Section 3.3, is the core focus of our analysis. Capturing these higher-level actions in the logs would require exposing functionality to the user at this level in the application, as the visual analytics system HARVEST does, as discussed previously [50]. Since this would require an interface redesign, this challenge remains when dealing with most systems, Tableau included. In the following sections we discuss our approach to identifying these higher-level activities, or tasks.

Term	**Definition**
participant	a student who consented to having their submission be included in our study
submission	the event logs, annotations, report, and Tableau workbooks from the exploratory data analysis assignment, which we collected for this study
task	an imprecise term meant to roughly discretize activity directed towards a common goal; e.g., examine the relationship between X and Y. The level of granularity that is best for this discretization is subjective, depending both on intended use and personal preference.
event	a unit of data issued by software and usually written to a log, in response to some trigger such as user activity, like a mouse click. An event always has a timestamp. Exactly what other data the event contains depends on the software and the type of event.
action	associated with each event, the trigger that caused the event to be issued. We consider only events triggered by user activity. Each event provides the action that triggers it as part of its information.
category	the type of action
field	a column in the dataset examined by the user in Tableau
segmentation	a division of the event stream into segments
segment	a discrete subsequence of consecutive events representing actions directed towards the same task
segment boundary	the point between events at which one segment ends and the next begins; denoted by the index of the first event of the segment after the boundary
segmenter	the creator of a segmentation
manual segmentation	a segmentation created by a human person
random segmentation	or randomly-generated segmentation, a segmentation generated by randomly selecting event indexes from the event stream
heuristic	a set of explicit rules for creating a segmentation
heuristic segmentation	a segmentation generated from a heuristic
metric	a function defined over segments or a segmentation to measure some property, such as similarity between the set of fields in two different segments
overlap coefficient	a measure of set similarity defined as the size of the intersection divided by the smaller of the size of the two sets
Jaccard index	a measure of set similarity defined as the size of the intersection divided by the size of the union of the sample sets
Herfindahl index	a measure of the degree of concentration of types in a set equal to the sum of the square of the percent of each type in the set
Jaccard distance	one minus the Jaccard index

Table 5.2: Definitions of terms and metrics.

5.3 Analysis of Annotated Tableau Logs

The goal of our analysis is to assess if we can summarize the event logs, by first segmenting the event stream into subsequences of similar consecutive events, then describing these segments. First, we describe a visualization tool we developed to aid in the discovery of subsequences of like events that could be grouped into a segment. We then describe our analysis of several segmentations that we created using this tool. In particular we assess the internal similarity of segments with respect to properties like field sets, action sets, sheets, and time, as well as the degree of similarity between pairs of consecutive segments with respect to these same properties. We find that adjacent segments in manual segmentations do tend to be more different from one another in terms of these properties than would be the case if the segmentations were meaningless (i.e., random). However, paradoxically, the events within the segments are not more similar than those within randomly-generated segmentations, according to our metrics.

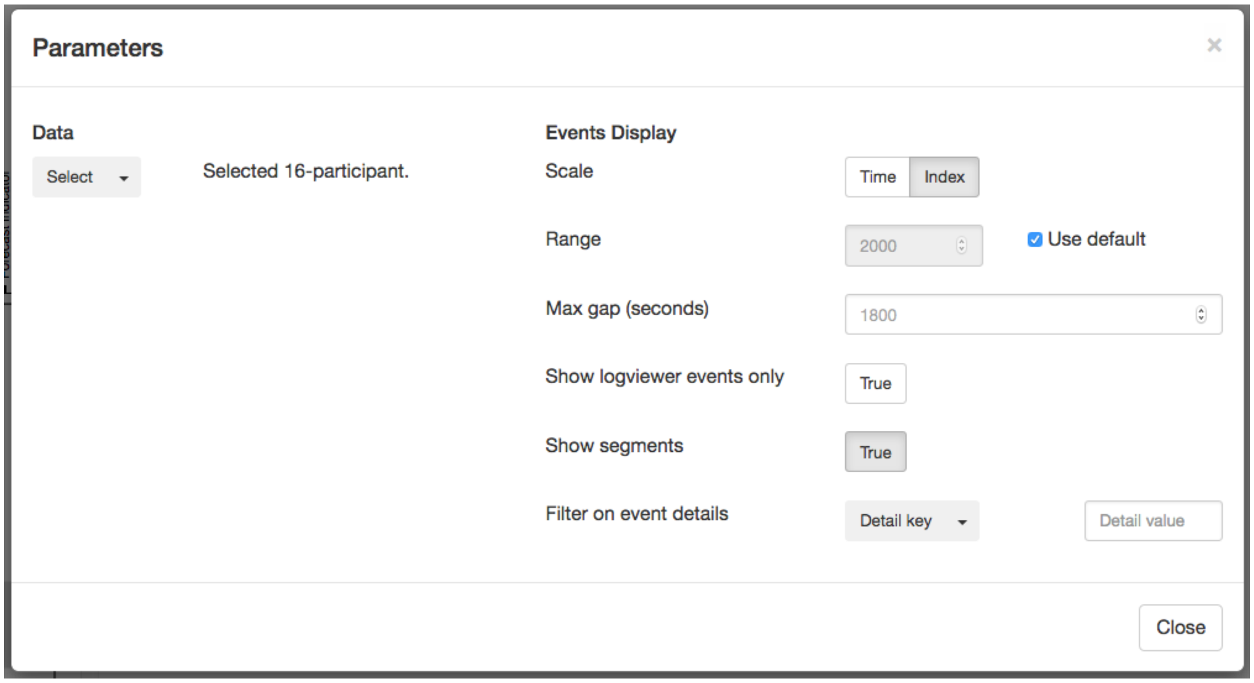
We assess the degree of similarity between the manual segmentations and propose several adjustments to rectify inconsistencies and bring the manual segmentations more in line with one another. We find that even with the adjustments, there is not a very high degree of overlap between manual segmentations – overlap ranges from 12% to 88% across the set of original and adjusted segmentations. We then develop a heuristic segmentation scheme based on this analysis of these manual segmentations, and analyze the results of the heuristic segmentations in the same way. We find that the heuristic segmentations behave similarly to the manual segmentations in terms of their properties and the degree of agreement with the manual segmentations. Lastly, we use the best heuristic segmentation scheme to summarize all of the submissions and provide some examples of what can be learned from these segmentations.

5.3.1 Event Log Segment Visualization Tool

We developed a visualization tool to depict properties of the events in the event stream, to make it easier to identify consecutive subsequences of similar events for segmentation.

5.3.1.1 Motivation

It is difficult to understand what a Tableau user did during a session by looking at their raw event log data for several reasons. The first is that there are many events – too many to inspect individually. The events contain many low-level details presented in a nested structured form that is not very amenable to being read by humans. The information in a single conceptual step is spread out over several linked events (i.e., an event describing a user action is usually accompanied by an event to describe various aspects of the application state). It's hard to take in information about more than one event at a time, and there is little context that is immediately apparent when examining each event in isolation. Any context gathered has to be built up in one's head or in some external format (via note-taking) in order to track what is happening. Thus, to be able to view all of the details associated with each event with greater context, we developed a visualization tool. Heer, et al., also created visualizations of this user activity in their Tableau work [61]. They found that using behavior graphs were the most useful for understanding user behavior, which contain some characteristics similar to our visualization, such as depicting application state and color-coding by action type. We did not try this approach, because the sessions we examined were generally several orders of magnitude longer, so we sought visualizations that could


Parameters

Data

Select

Selected 16-participant.

Events Display

Scale: Time [✓ Index]

Range: 2000 [✓ Use default]

Max gap (seconds): 1800

Show logviewer events only: True

Show segments: [✓ True]

Filter on event details: Detail key, Detail value

Close

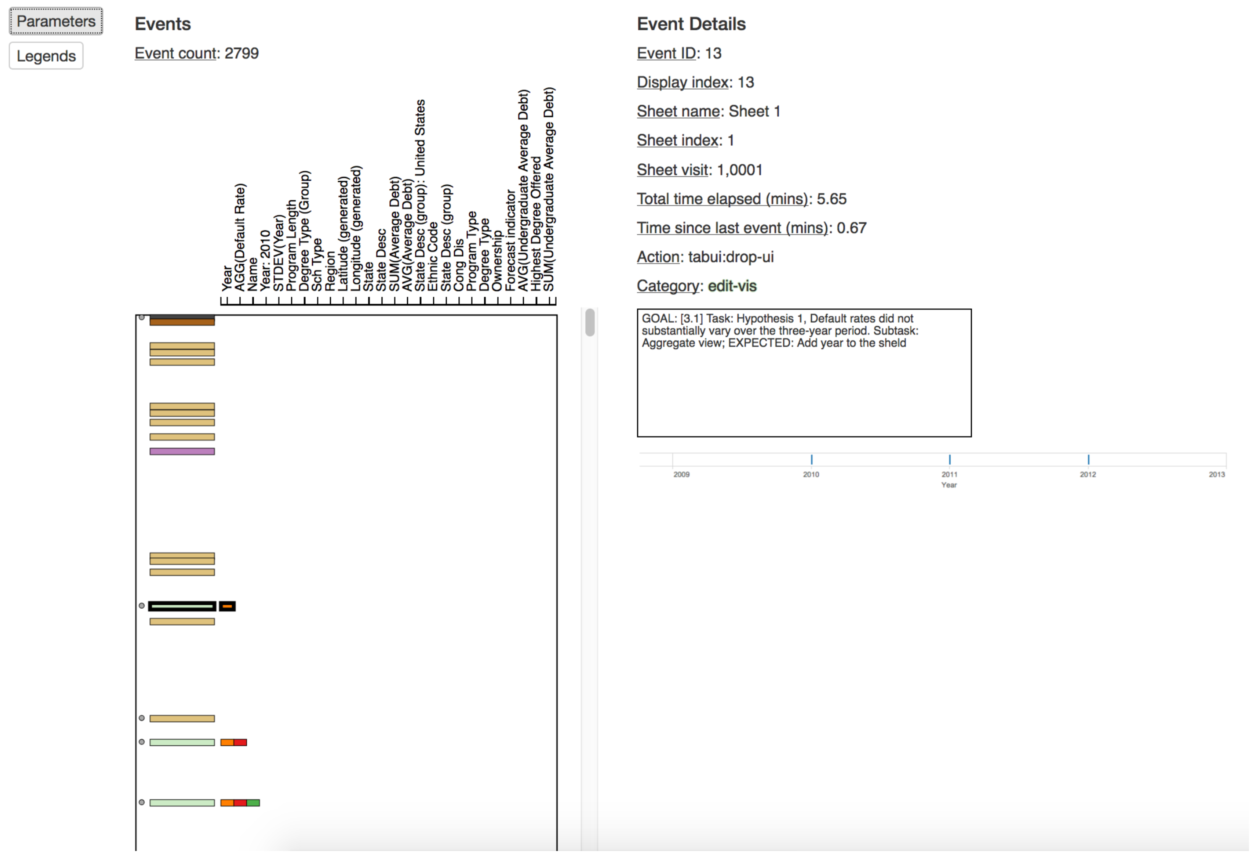
Figure 5.12: The modal window where the user selects the dataset to examine and sets other display parameters.

compactly display more events. However, it would be interesting to incorporate the graph aspect into our visualization as future work.

5.3.1.2 Description

This tool presents as many events as possible on one screen by representing the event stream as a vertical sequence of small color-coded rectangles (Figure 5.14). They can be displayed by event index or by time, so that the space between events corresponds to the amount of time elapsed between them (Figure 5.13). These rectangles reveal additional information on mouseover, as described below. The event that is moused over is highlighted in bold. The rectangles can be filtered to display only a subset, as described below. The rectangles can be plotted according to their index (default) or their timestamp (Figure 5.12). The color of the event rectangle corresponds to the category of the corresponding action (Figure 5.15). These events are shown in the Events window of the tool (Figure 5.14).

Using the default settings, approximately 60 events fit into the Events window at one time when using the index scale and a window size that fits on standard 15-inch laptop full screen windows. Fitting more events into the window at one time requires shrinking the event rectangles, which makes them more difficult to moused over, but this can be done by adjusting the range parameter to see more events at one time. For reference, the submission



Event count: 2799

Event ID: 13
Display index: 13
Sheet name: Sheet 1
Sheet index: 1
Sheet visit: 1,0001
Total time elapsed (mins): 5.65
Time since last event (mins): 0.67
Action: tabui:drop-ui
Category: edit-vis

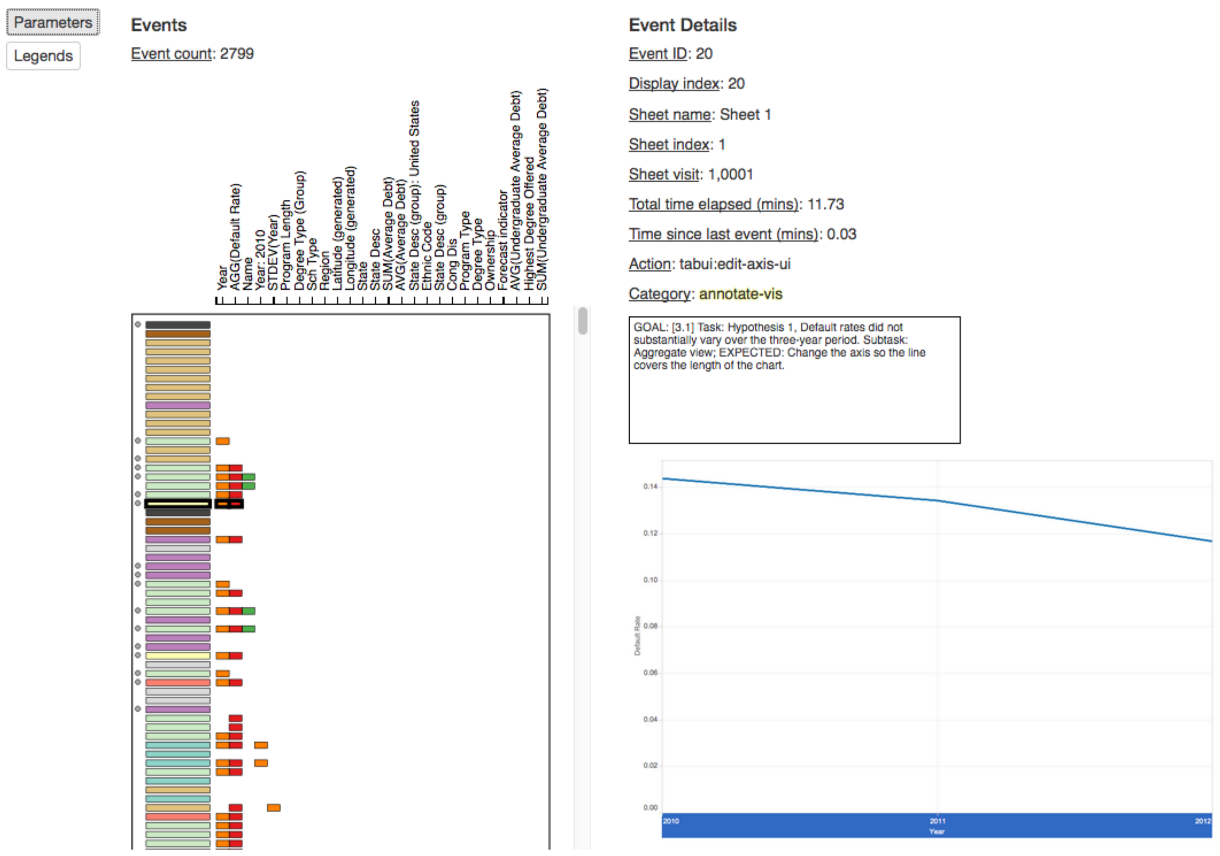
GOAL: [3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. Subtask: Aggregate view; EXPECTED: Add year to the sheld

Figure 5.13: Examining event details with the events plotted by time instead of by index so that the space between events corresponds to the time elapsed.

we spend the most time examining in this chapter has just under 3,000 events. The presence of a small grey circle next to an event indicates that that event is annotated. The annotation can be read by mousing over the event and looking at the data in the Event Details window. Also in the Events window of the tool, adjacent to the vertical event stream column, is a table, where each row corresponds to the adjacent event and each column to one of the fields in the dataset. When an event involves one of the fields, there is another color-coded rectangle in the entry corresponding to that event row and field column. The color of the field rectangles correspond to the datatype of the field (Figure 5.15). The font of the field labels shrinks when necessary to accommodate more fields in the table.

Upon mousing over an event, a set of details are displayed in an Event Details window. The details presented are:

- Event ID: the index of the event in the entire event stream
- Display index: the index of the event in the currently displayed subset (which differs from the above when a filter is applied)
- Sheet name: the current name of the sheet the event happened on (this changes when



Event count: 2799

Event ID: 20
Display index: 20
Sheet name: Sheet 1
Sheet index: 1
Sheet visit: 1,0001
Total time elapsed (mins): 11.73
Time since last event (mins): 0.03
Action: tabui:edit-axis-ui
Category: annotate-vis

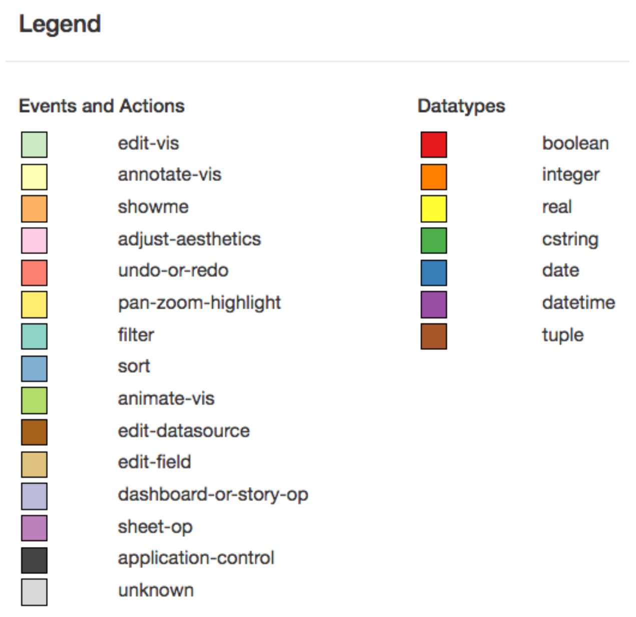
GOAL: [3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. Subtask: Aggregate view; EXPECTED: Change the axis so the line covers the length of the chart.

Figure 5.14: Examining event details in an unsegmented view of the event stream.

the user renames the sheet)

- Sheet index: the index of the sheet the event happened on (this is assigned upon sheet creation and does not change throughout the life of the session)
- Sheet visit: the number of times the current sheet (identified by index) has been visited
- Total time elapsed (in minutes): minutes since the first event
- Time since last event (in minutes): minutes since the previous event
- Action: the action associated with the event
- Category: the category the action falls under (see Table 5.2)
- Annotations: from the annotations generated with the LogViewer tool, if there are any annotations on the current event
- Screenshot: of the current state of the tool after the event, if available

As in the previous case study, we manually classified each of 424 low-level actions into one of 15 categories. The category of each event is listed in Appendix C. This categorization makes it easier to understand what an action accomplishes at a quick glance. Other work on making sense of Tableau event logs by Heer, et al., took a similar approach, except they only used five categories: "shelf (add, remove, replace), data (bin, derive field), analysis



Legend

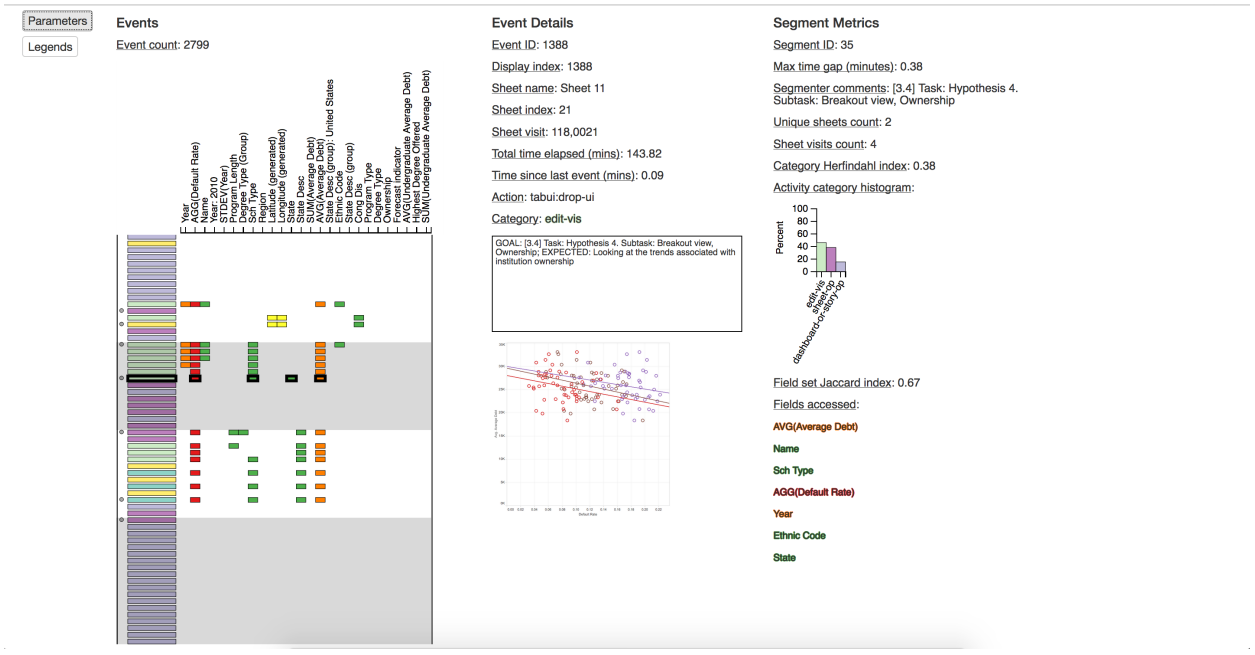
Events and Actions

🟩	edit-vis
🟨	annotate-vis
🟧	showme
🟪	adjust-aesthetics
🟥	undo-or-redo
🟨	pan-zoom-highlight
🟩	filter
🟦	sort
🟩	animate-vis
🟫	edit-datasource
🟫	edit-field
🟪	dashboard-or-story-op
🟪	sheet-op
⬛	application-control
⬜	unknown

Datatypes

🟥	boolean
🟧	integer
🟨	real
🟩	cstring
🟦	date
🟪	datetime
🟫	tuple

Figure 5.15: The color legends for the event action categories and datatypes.



Events

Event count: 2799

Event Details

Event ID: 1388

Display index: 1388

Sheet name: Sheet 11

Sheet index: 21

Sheet visit: 118,0021

Total time elapsed (mins): 143.82

Time since last event (mins): 0.09

Action: tabui:drop-ui

Category: edit-vis

GOAL: [3.4] Task: Hypothesis 4. Subtask: Breakout view, Ownership; EXPECTED: Looking at the trends associated with institution ownership

Segment Metrics

Segment ID: 35

Max time gap (minutes): 0.38

Segmenter comments: [3.4] Task: Hypothesis 4. Subtask: Breakout view, Ownership

Unique sheets count: 2

Sheet visits count: 4

Category Herfindahl index: 0.38

Activity category histogram:

Field set Jaccard index: 0.67

Fields accessed:

AVG(Average Debt)

Name

Sch Type

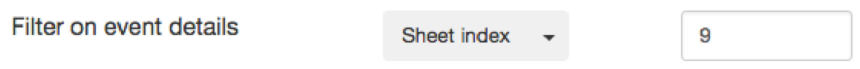
AGG(Default Rate)

Year

Ethnic Code

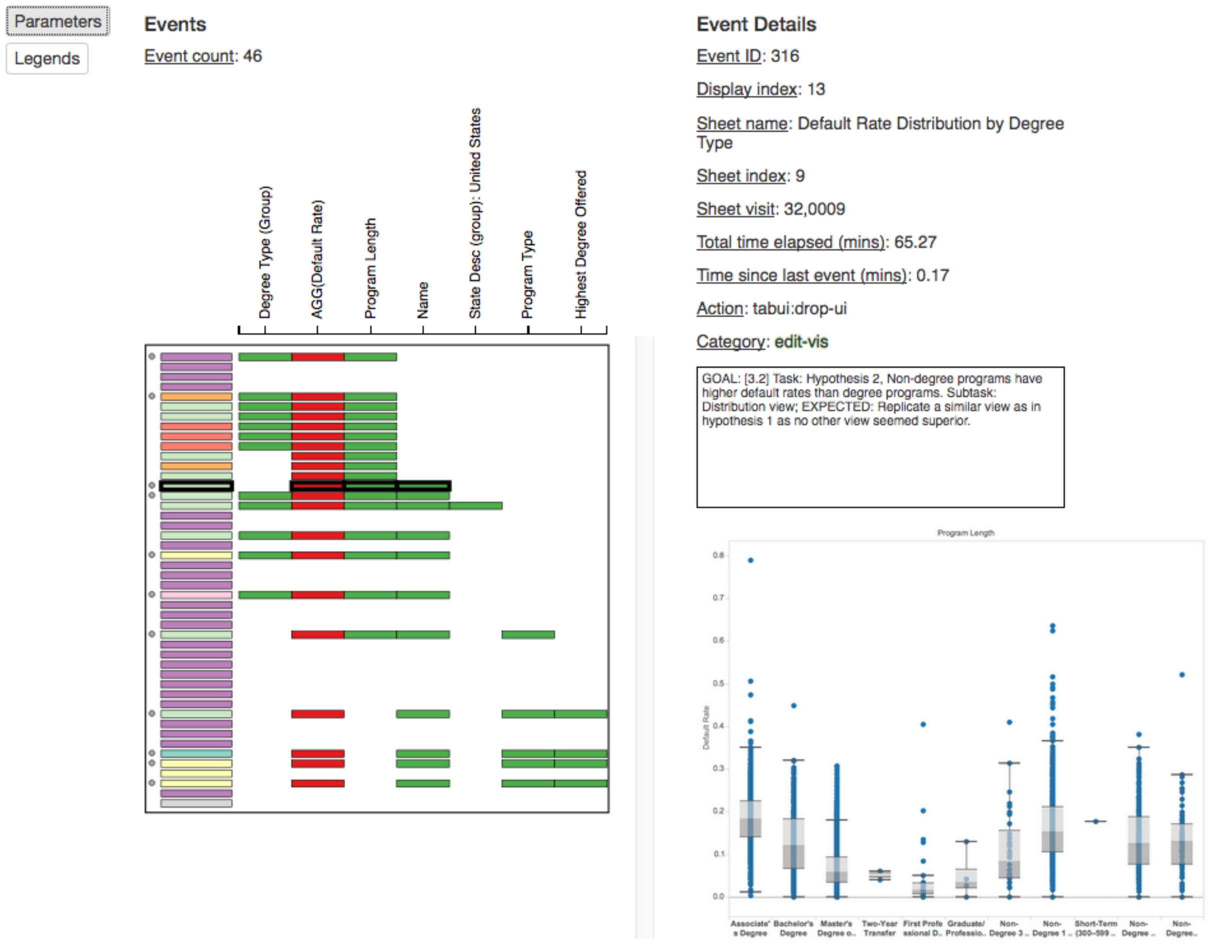
State

Figure 5.16: Examining a segmented view of the event stream along with segment properties.


Filter on event details

Sheet index: 9

Figure 5.17: Applying a filter over the events to show only events on sheet indexed 9.



Event count: 46

Event ID: 316

Display index: 13

Sheet name: Default Rate Distribution by Degree Type

Sheet index: 9

Sheet visit: 32,0009

Total time elapsed (mins): 65.27

Time since last event (mins): 0.17

Action: tabui:drop-ui

Category: edit-vis

GOAL: [3.2] Task: Hypothesis 2, Non-degree programs have higher default rates than degree programs. Subtask: Distribution view; EXPECTED: Replicate a similar view as in hypothesis 1 as no other view seemed superior.

Figure 5.18: The result of the filter to show only events on the sheet with index 9.

Category	*Description*
edit-vis	drag and drop fields onto shelves to create or modify a graph
annotate-vis	add text, pointers, and other annotations to a graph
showme	use the ShowMe tool to generate a desired graph type
adjust-aesthetics	change colors, marks, or other stylistic elements of the graph
undo-or-redo	undo or redo an
pan-zoom-highlight	pan, zoom, or highlight a portion of a graph
filter	filter the values of a field shown on a graph
sort	sort the marks of a field shown on a graph or an axis
animate-vis	use slideshow feature to show successive graphs over time
edit-datasource	edit the datasource being analyzed in the intro menu
edit-field	create a field or make changes to a field's datatype
dashboard-or-story-op	create or edit a dashboard or story
sheet-op	create, delete, or copy a sheet
application-control	open, save, or close workbook
unknown	unknown action – could not reproduce

Table 5.3: Name and description of action categories used to classify each Tableau event according to the action that triggered it.

(filter, sort), worksheet (add, delete, duplicate), and formatting (resize, style)" [61]. Their parenthetical description of the categories more closely correspond to the categories we used, which are listed in Table 5.3.

The tool provides a Parameters modal (Figure 5.12), which provides the following options:

- Select the data: specifies a submission as well as, optionally, a segmentation of that submission
- Scale: whether the events in the events column should be spaced according to their index or by timestamp
- range: the height in pixels of the area to map the rectangles are mapped to
- Max gap: when events are plotted by timestamp, the quantity in seconds to cap the max gap in time, so that sessions with a day or more between event don't result in an extremely large gap in displayed events, which would require too much scrolling and/or the shrinkage of event rectangles to fit
- Show LogViewer events only: display only events that were also shown by the LogViewer tool (useful for understanding what participants saw when they made their annotations)
- Show segments: toggle the display of the segmentation of the data if one was provided (described more below)
- Filter on event details: filter the displayed events according to the specified sheet name, sheet index, action, or category

If the “show segments” option is selected, the segments are shown in the event stream by highlighting the background of every other segment’s event sets (Figure 5.16). An additional window, Segment Metrics, is shown, that provides the following information about each segment upon mouseover:

- Segment ID: the index of the current segment
- Max time gap (minutes): the largest gap between events in the segment in minutes
- Segmenter comments: any comments made by the segmenter about this segment
- Unique sheets count: the number of unique sheets visited in this segment
- Sheet visits count: the number of distinct sheet visits in this segment
- Action category plot: a bar chart showing the percent of events in each action category present in the segment
- Category Herfindahl index: the Herfindahl index of the action category set (described in Table 5.2)
- Fields overlap coefficient: the average overlap coefficient between the segment’s adjacent event pairs’ field sets
- Field set Jaccard index: the median Jaccard index of the segment’s adjacent event pairs’ field sets (see Table 5.2)
- Fields accessed: the set of fields accessed in the segment

Two of the above metrics are worthy of further explanation. The first is the Herfindahl index – this measure is used to describe how how “diverse” or conversely, how “concentrated” the segment is with respect to different types of activities. The more different categories of activities there are in a segment, the more diverse it is. Ideally segments would be formed in such a way as to be dominated by one category of activity, because we are trying to group similar activity to create the segments, which we can then describe in terms of the dominant type of activity. The next metric to describe in more detail is the Jaccard index – we use this metric to compare the similarity of two separate sets, rather than the similarity of types within a set, as the previous metric does. The segmentation tool shows the median of the set of Jaccard indices computed over all consecutive pairs of events in a segment – specifically over the set of fields associated with each event.

We asked our collaborators to use this tool, along with instructions that we describe below, to segment a single submission. We describe this in Section 5.3.3.

5.3.2 Participant Annotations

Participants’ annotations vary widely in style and quality. We initially hoped to use participants’ annotations to segment their event logs, and wrote the annotation instructions with this in mind. However, we were ultimately not able to use all of them to segment, for several reasons: We found, via manual inspection, that the quality and quantity of annotations across and even within submissions is too inconsistent, We also later found, via our own attempts to annotate and manually segment, that our annotation instructions were difficult to objectively and consistently apply. Properties of the participant annotations are presented in Table 5.4.

ID	**Annotations**	**Events**	**Avg Annotation Length**	**Total Duration (mins)**	**Active Duration (mins)**	**Used Instructed Format**
32	22	1949	79.27	1439.77	602.44	0
21	20	2083	71.55	693.03	502.8	2
19	27	3607	126	1493.64	657.95	0
37	38	1780	70.23	7536	527.59	0
12	25	2923	175.16	1313.37	620.01	0
33	33	10540	86.36	1745.22	705.01	0
10	61	1501	82.51	1413.91	988.69	0
18	23	1415	129.6	349.02	331.13	0
23	32	7199	97.88	1867.9	609.86	2
38	60	4145	66.65	260.03	224.92	0
29	108	4437	70.83	7833.73	566.4	0
15	70	9227	218.19	1475.6	745.67	0
28	64	3415	160.67	1738.34	792.42	2
44	51	1013	100.9	409.21	384.94	1
31	146	5270	68.34	3827.76	455.48	0
40	53	1620	135.2	4890.28	1132.15	2
24	42	475	95.35	29.56	29.56	0
26	43	748	162.44	378.05	295.38	0
41	33	1007	134.18	8761.64	337.44	0
39	17	601	361.94	592.98	432.22	0
20	75	1168	111.93	3051.81	900.69	0
14	18	150	193.83	129.07	113.7	2
27	23	1116	100.57	194.88	180.84	0
16	239	2799	104.26	1403.72	494.74	3
min	17.00	150.00	66.65	29.56	29.56	0.00
mean	55.13	2,924.50	125.16	2,201.19	526.33	0.58
median	40.00	1,864.50	102.58	1,426.84	515.20	0.00
max	239.00	10,540.00	361.94	8,761.64	1,132.15	3.00
stddev	49.65	2,738.50	65.66	2,550.70	272.03	0.97

Table 5.4: Properties of Tableau annotations. For the *Used Instruction Format*, scores range from 0 – not at all, to 3 – very well.

Label	Iteration	Segmenter	Number of Segments
A	original	collaborator	53
B	original	collaborator	73
C	original	participant	77
A’	first adjustment	collaborator	49
B’	first adjustment	collaborator	72
C’	first adjustment	participant	82
A”	second adjustment	collaborator	58
B”	second adjustment	collaborator	70
C”	second adjustment	participant	76

Table 5.5: Labels for manual segmentations and their adjustments.

5.3.3 Manual Segmentations

We manually segmented several submissions, to obtain:

- First-hand experience with the segmentation process, to better understand what event conditions should be considered to constitute a distinct segment of events
- Baseline segmentations to compare other segmentations against, and potential training data to use to train a classifier to segment event data, if that turned out to make sense to do

Our approach is as follows. Segmenters were instructed to manually inspect certain submissions, which were chosen by an informal process for having reasonable amounts of annotation and a middling number of events. They were instructed to manually inspect the submission events and annotation using the segment visualization tool. They were instructed to then segment the submission, and label the segments, according to the following rules:

- Mark an event as the beginning of a new segment if one or more of the following is true:
  - It is the very first event in the sequence
  - It marks the beginning of a sequence of events in which the dominant type of activity is different from that of the sequence of events immediately prior (change in activity)
  - It marks the beginning of a sequence of events in which the subset of data involved is different from the sequence of events immediately prior (change in data)
  - It indicates creation of a new sheet, e.g., from scratch or via duplication of an existing sheet (change of sheet) and/or it occurs after a temporal delay after the previous event (gap in time)
- To label a segment:
  - Describe the dominant type of activity
  - Describe the data involved, if relevant

- Describe the likely goal of the activity

For Submission 16, the participant did a particularly excellent job of following the annotation instructions from the original assignment instructions. Namely, they prepended every annotation with a description of their current task. Thus, we also generated a segmentation from their annotation, by creating a new segment every time the task described in the annotation differed from the previous annotation. For example, for their first task, the participant prepended their annotations with "[3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. *Subtask: Aggregate view*" (emphasis ours). Then for the second task, the participant changed the description prefix to "[3.1] Task: Hypothesis 1, Default rates did not substantially vary over the three-year period. *Subtask: Distribution view.*" We mark the first event at which this change occurs as the start of a new segment.

5.3.3.1 Assessment

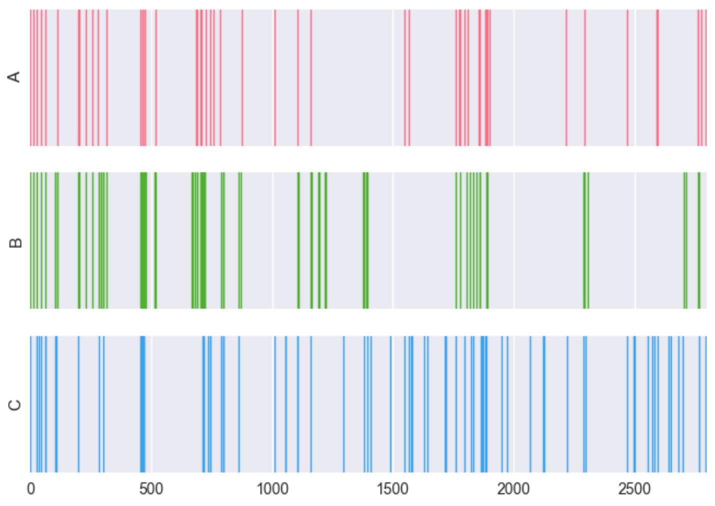
We assessed the manual segmentations to first understand the properties of their constituent segments, and of the events that segmenters chose as segment boundaries, and second to find out how much agreement there was between segmenters' choices. For our assessment, we chose to focus on a single submission that had both high-quality segmentations and annotations. The annotations were high-quality because they were very thorough and detailed – it had the largest quantity of annotations of any submission – and because the participant followed our instructions closely in creating the annotations. The segmentations were high-quality in that segmenters followed the same set of rules and provided commentary on segment boundary explaining their choice and segment synopsis. Two diagrams of these manual segmentations over the event stream, plotted by event index and by time respectively, is shown in Figure 5.19. Each plot represents a different Segmentation And each vertical line in the plots represents an event that was marked as the beginning of a segment.

Where applicable, we compared the manual segmentations to randomly-generated segmentations. The randomly-generated segmentations have the same number of segments as the corresponding manual segmentation that they are being compared to; they are generated by randomly selecting event IDs without replacement, to be used as segment boundaries, until the desired number of segments is reached. We later attempted to reconcile differences by proposing adjustments, described at the end of this section. This manual assessment informed the design of the heuristics we later created to automatically segment the submissions (Section 5.3.4).

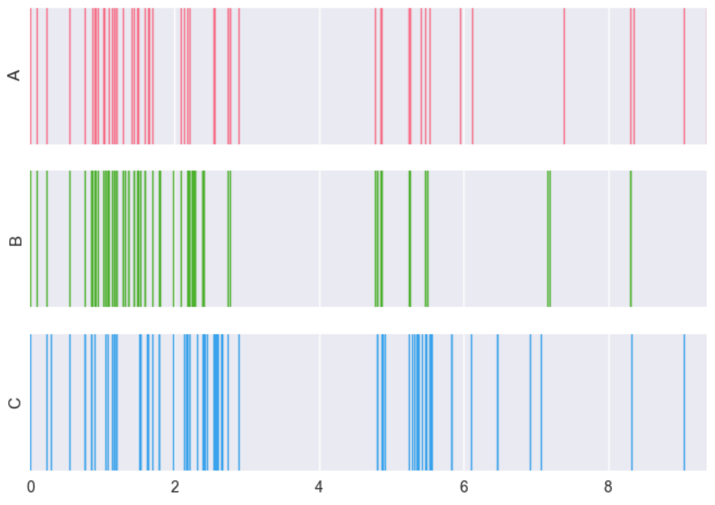
Questions investigated:

- What are the properties of the manual segmentations? E.g., number of sheet switches, number of segments, length of segments, etc.
- How much agreement is there between the manual segmentations?
- What the types of and reasons for disagreements among human segmentations?

Properties of Manual Segmentations Here we present the properties of segments and their boundaries for the manual segmentations of Submission 16. For comparison, we com-



(a) Plotted by event index.



(b) Plotted by time.

Figure 5.19: Segment boundaries in each original manual segmentation, plotted by event index and by time respectively, of Submission 16, which we focus on in detail in this section. Each plot represents a different Segmentation And each vertical line in the plots represents an event that was marked as the beginning of a segment. For the plots by time, pauses in activity greater than one hour were clipped to one hour, for display purposes.

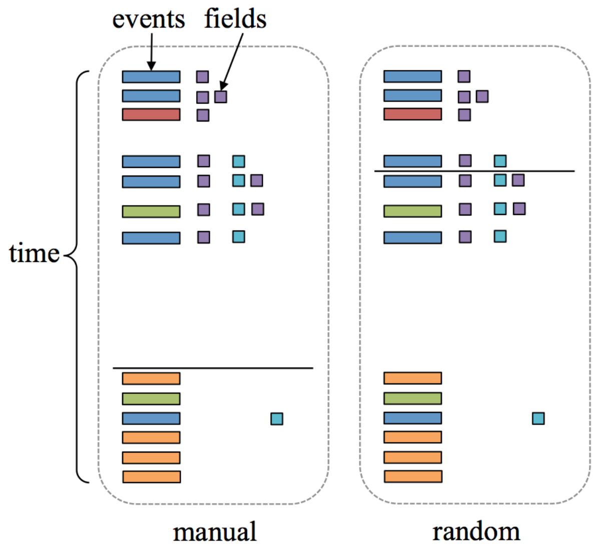


Figure 5.20: An example diagram of our expectations regarding the manual versus random segmentations. The events are color-coded by activity type, and the fields by datatype, and events are shown spaced by the time that elapsed between them. The segment boundaries are shown as a horizontal black line. We expect that compared to the randomly-generated segmentations, manual segmenters will choose segment boundaries so as to group together similar activities or activity categories, similar sets of fields, and events that are close in time, more often.

pute the same properties over randomly-generated segmentations as well. The labels for the segmentations are given in Table 5.5. The labels "A" and "B" refer to the manual segmentations of Submission 16 created by our collaborators, as well as ten randomly-generated segmentations for each group that are the same length as their corresponding manual segmentation. The label "C" refers to the segmentation derived from the annotations of the participant who created Submission 16, as well as ten same-length randomly-generated segmentations. The randomly-generated segmentations for each Group A, B, and C have the same number of segments as the corresponding manual segmentations that we are comparing to, because many of the metrics we consider vary according to the number of segments. Restricting the randomly-generated segmentations to have the same number of segments as their corresponding manual segmentations provides a more fair comparison.

Figure 5.20 provides an example diagram of our expectations regarding the manual versus random segmentations. This diagram is modeled after the display in the visualization tool discussed in Section 5.3.1. As in the visualization tool, the events are color-coded by activity type, and the fields by datatype, and events are shown spaced by the time that elapsed between them. The segment boundaries are shown as a horizontal black line. We expect that compared to the randomly-generated segmentations, manual segmenters will choose

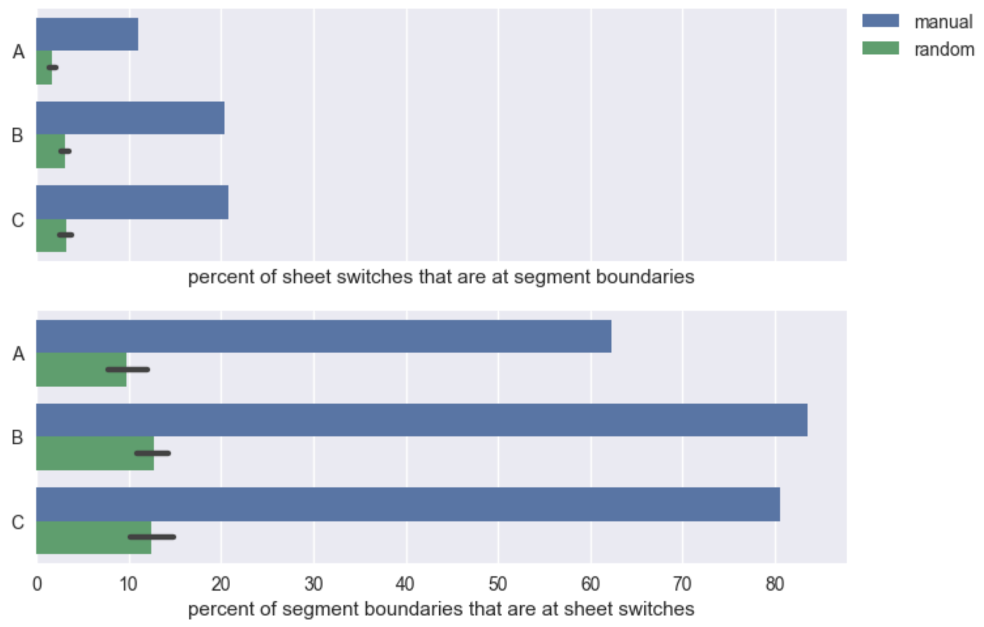


Figure 5.21: Sheet switches and segment boundaries: These figures show the percent of sheet switches in the submission that occur at segment boundaries, as well as the percent of segment boundaries that occur at sheet switches. We would expect that manual segmentations to have a higher percent of both than random segmentations and that is the case for all comparison groups.

segment boundaries so as to group together similar activities or activity categories, similar sets of fields, and events that are close in time, more often. The sheet the events occurred on are not shown in the diagram, but we also expect that events that occur on the same sheet will be more often grouped together than events on different sheets. The diagram shows an example manual segment boundary occurring so as to achieve this similarity within segments, whereas the randomly-placed segment boundary would be less likely to do so since it is randomly chosen and not selected so as to group together similar activity.

The properties we consider are:

- Boundary properties: new sheet creations and sheet switches at segment boundaries (Figures 5.21 and 5.22)
- Intra-segment properties: within each segment, the similarity of its events' actions, categories, fields, and sheets, and also the temporal proximity of pairs of adjacent events (Figures 5.23, 5.24, 5.25, 5.26, and 5.27)
- Inter-segment properties: between pairs of adjacent segments, the similarity of their sets of actions, categories, fields, and sheets, and also their temporal proximity (Figures 5.28 and 5.29)
- Segment length (Figure 5.30)

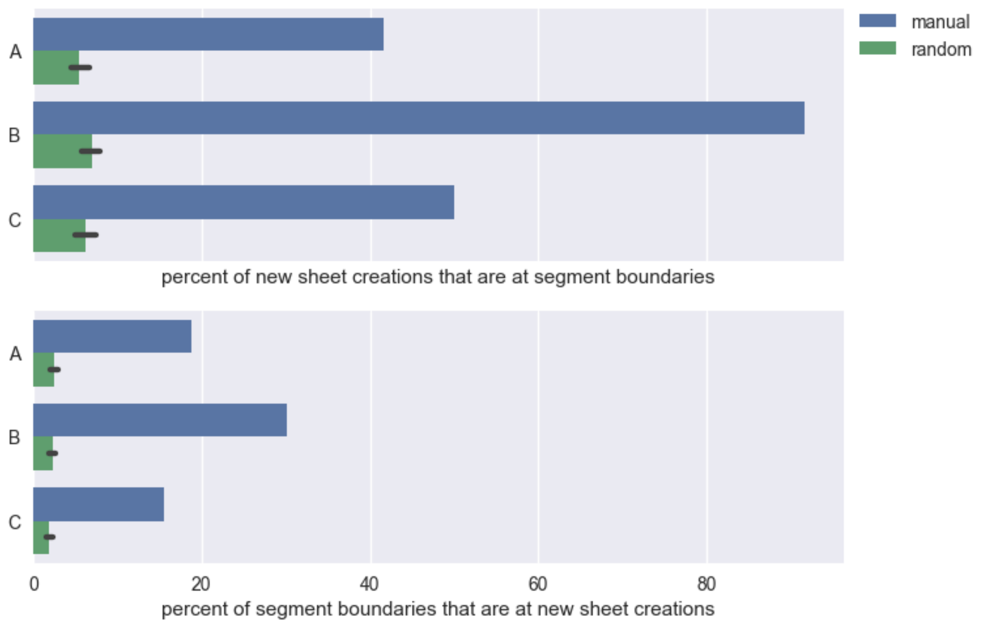


Figure 5.22: New sheet creations and segment boundaries: These figures show the percent of new sheet creations in the submission that occur at segment boundaries, as well as the percent of segment boundaries that occur at new sheet creations. We would expect that manual segmentations to have a higher percent of both than random segmentations and that is the case for all comparison groups.

Expectations:

- Sheet or sheets: We expect that segment boundaries will often occur when the participant switches sheets or creates a new sheet (Figures 5.21 and 5.22). We also expect that the set of sheets will exhibit consistency within a segment and less overlap across segments (Figure 5.28). We expect that there will be more sheet switches and new sheet creations within the segments of random segmentations than for manual segmentations (Figures 5.23 and 5.24).
- Sets of actions: We expect that within a segment, there should be a set of related actions which predominate (Figure 5.25), and from segment to segment, the set of actions should change, so that there is not much overlap between action sets from one segment to the next (Figure 5.28).
- Sets of categories: Since actions are grouped into categories, we similarly expect a few categories to dominate each segment (Figure 5.25) and for the set of categories from segment to segment to not exhibit much overlap (Figure 5.28).

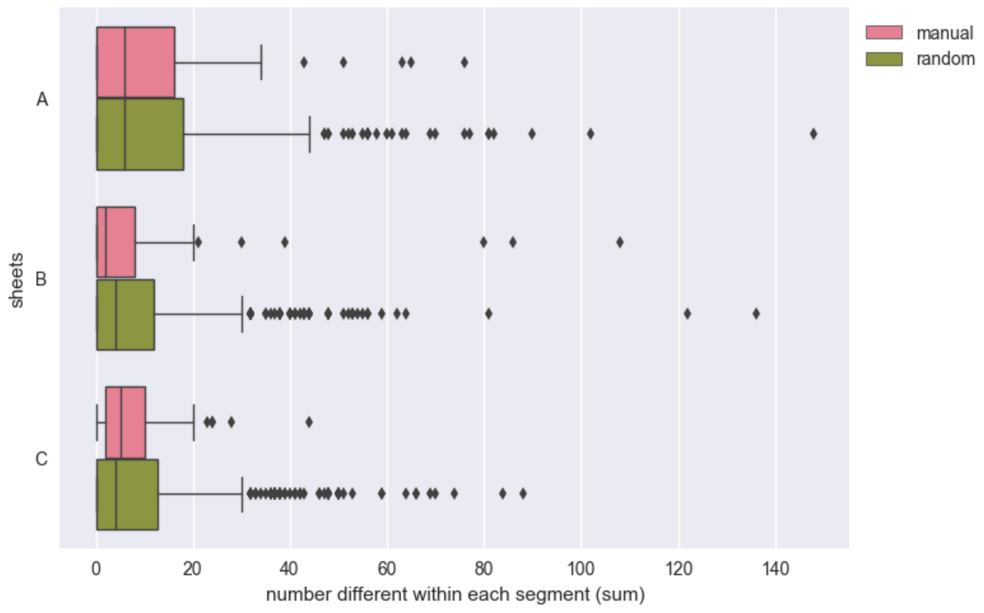


Figure 5.23: Number of sheet switches within each segment: This figure shows the distribution of the number of sheet switches within each segment. We would expect manual segmentations to have fewer sheet switches within each segment, with sheet switches occurring more often on segment boundaries (and these were not included as part of the within-segment count). This is true for comparison Groups A and B but not C.

- Sets of fields: We would expect to make similar observations for sets of fields (Figure 5.26 and Figure 5.28) as for sets of actions and categories, except that not every event has field data, whereas every event does have action and category data. However, we don't anticipate that this will have a large effect.
- Temporal proximity of events: Within segments, we expect events to be closer together (Figure 5.27). We expect segment boundaries to occur when there is a large gap in time, although there will not always be a large gap in time between segments (Figure 5.29).
- Segment length: The purpose of segmentation is to chunk the event stream into sequences of events that share something in common, such that they can be effectively summarized. These chunk summaries can then serve as an overall summary of the event stream. Therefore we expect that each segment should not be too small, otherwise it would not summarize very many events (Figure 5.30).

To summarize our results, we discuss where our expectations were completely met, where they were partially met, and where they were not met at all. As we show later, we tested a subset of these findings – only some of them proved to be statistically significant, and none of the property results were significant for all segmentations.

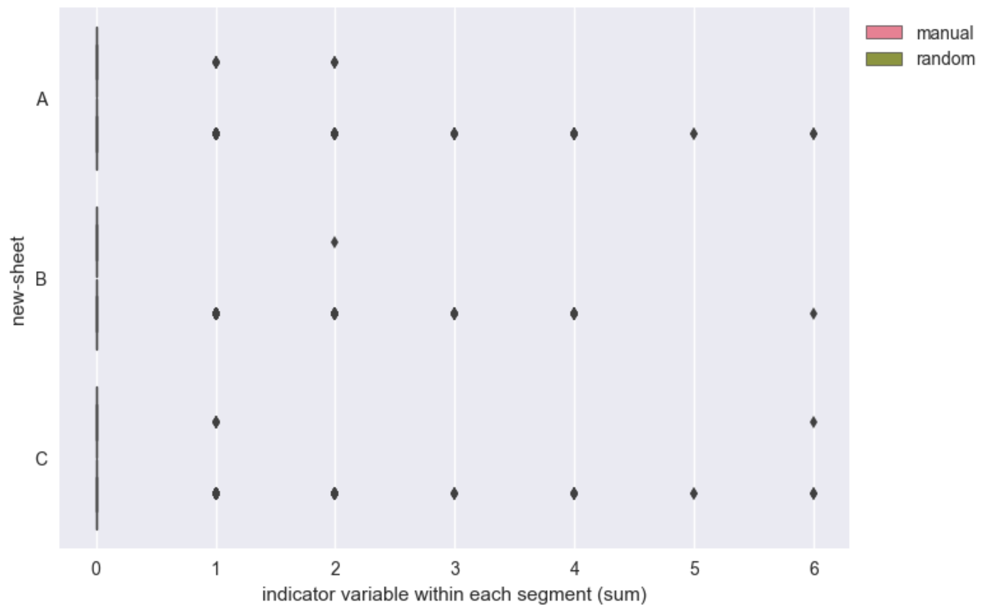


Figure 5.24: Number of new sheets within each segment: This figure shows the distribution of the number of new sheets created in each segment. We would expect manual segmentations to have fewer new sheets created within each segment, because new sheet creations would occur more often at segment boundaries (and these were not included as part of the within-segment count). However, new sheet creations are rare enough that the median number of new sheet creations for all segmentations is zero.

Our expectations were completely met regarding sheet switches and new sheet creations at boundaries – manual segmentations are more likely to choose these as boundaries (Figure 5.21 and Figure 5.22). They were also met regarding the similarity of the field sets of the events within each segment – manual segmentations' segments have events that have more similar field sets than segments from randomly-generated segmentations (Figure 5.26). Lastly, they were also met regarding the similarity of the sets of actions, categories, and sheets between pairs of adjacent segments – manual segmentations' segments are more dissimilar in these respects (Figure 5.28). The similarity of the sets of actions and sheets were the most significant results.

Our expectations were met for comparison Groups A and B, but not for C (the annotations derived from the participant), in the following cases: Between segments, field sets are only more dissimilar across adjacent pairs of segments for manual segmentations as opposed to random segmentations in Groups A and B (Figure 5.28). With respect to the amount of time elapsed between segments, manual segmentations A and B exhibit greater median time between segments than random segmentations, but C does not (Figure 5.29). This suggest that in contrast to the work of Santos and Badre, in which they detect boundaries based on pauses in user activity, pauses may not be a good indicator of segment boundaries for this

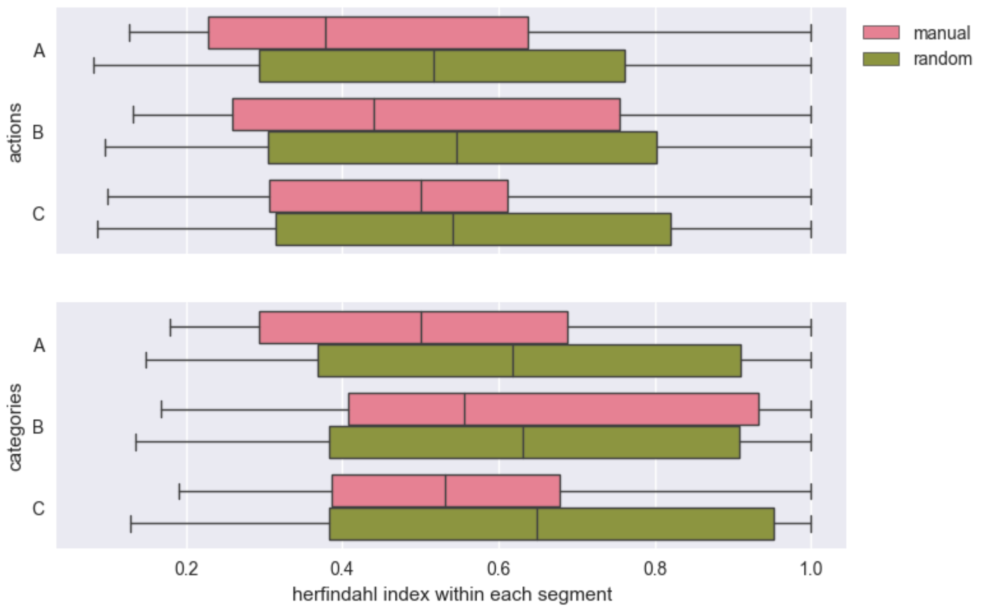


Figure 5.25: Concentration of the distribution of actions and action categories within each segment: This figure shows the distribution of the Herfindahl index of the action and category sets, where a higher number means the set is dominated by fewer actions or categories, respectively. We expected manual segmentations to have a higher Herfindahl index than random; these results would seem to contradict that for all comparison groups, including adjustments.

dataset [134].

There was ambiguous support for our expectation that manual segmentations' segments would contain fewer new sheet creations than those of random segmentations because the median for all groups is zero (Figure 5.24). There was also ambiguous support for our expectation that manual segmentations' segments would contain fewer sheet switches than those of random segmentations, with A having an equal median, B having a lower median, and C having a greater median (Figure 5.23).

Our expectations did not hold for any comparison group with respect to the uniformity of actions and categories within each segment – random segmentations appear to be more uniform; we expected the opposite (Figure 5.25). They also did not hold for the median number of seconds between pairs of adjacent events within each segment. It is greater for manual segmentations' segments than for random segmentations' – we expected the opposite because we thought the submission events would be "clumpy" and that the segments would correspond to these clumps (Figure 5.27). We should note that Segmenter A was the only segmentation to explicitly take time into account, and found that many events didn't fit well into segmentation rules by time, because often it seemed some events with large gaps between them should be grouped together.

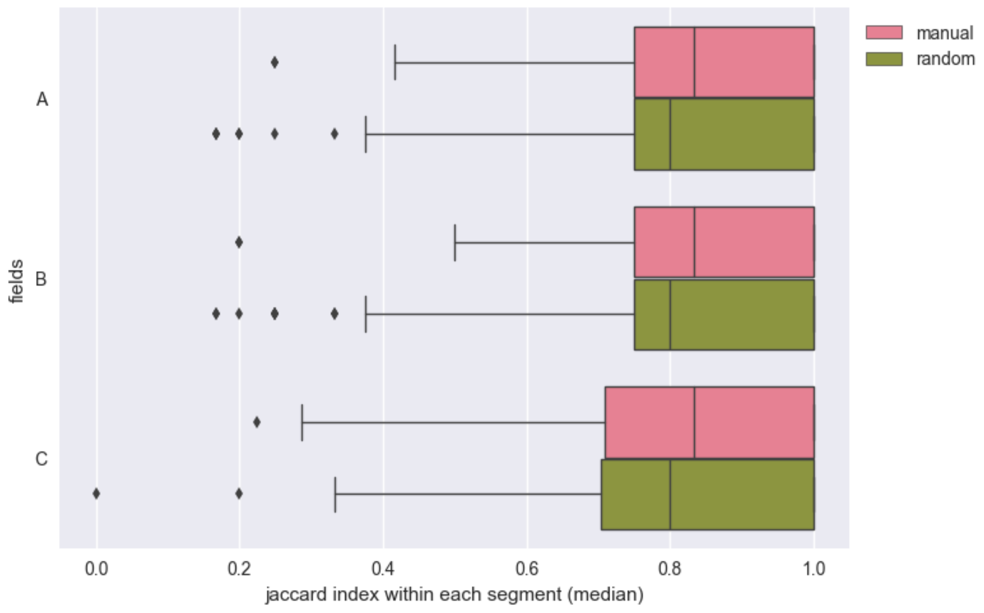


Figure 5.26: Similarity of the fields sets within each segment: This figure shows the distribution of the median Jaccard index (a measure of set similarity) of the field sets of each pair of adjacent events in each segment. A higher Jaccard index means greater similarity. We would expect manual segmentations to have higher similarity between the field sets of their segments' events. This is true for all comparison groups.

To test whether the property metric differences between the manual segmentations and randomly-generated segmentations are significant, we applied Mood's median test to the datasets depicted in the figures listed in Table 5.6. We use Mood's median test because it does not rely on the assumption that the data is normally distributed (such as the two-sample t-test or ANOVA), that there are no outliers (such as the Mann-Whitney test), or that the two comparison datasets have similar variance (such as the Mann-Whitney or Kruskal-Wallis H test) [100]. Moreover we are interested in comparing medians instead of averages, since the average is sensitive to outliers, of which there are many, as shown in the figures. The drawback of using Mood's median test is that it is less powerful than many other common tests, meaning that it is less able to detect actually false null hypotheses (e.g., it may report cases in which there is a significant difference between medians as having no significant difference) [100]. One uncommon feature of the Mood's median test is that it is necessary to specify whether values equal to the median should be counted as less than, equal to, or greater than the grand median. Different statistical packages and authors use different defaults. We opted for the most neutral approach where values equal to the grand median are ignored, though the alternative options, counting them as above or below the grand median, yielded essentially the same results, with only slight differences in p-values.

For the significance threshold, it is common to use a cut-off of 0.05. However, performing

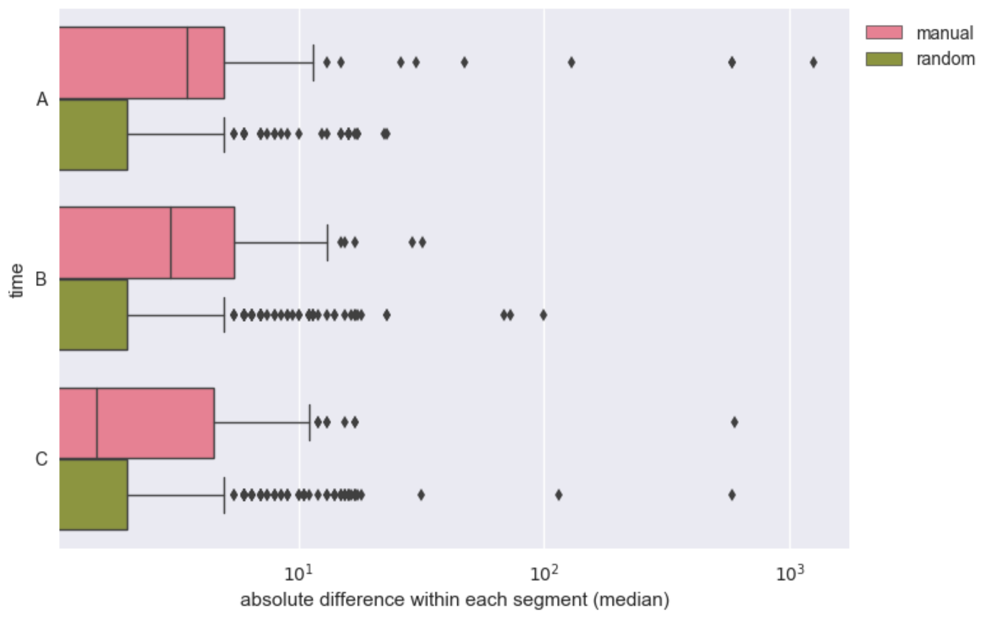


Figure 5.27: Number of seconds elapsed between consecutive events within each segment: This figure shows the distribution of the median number of seconds elapsed between consecutive pairs of events within each segment. We would expect that events within the segments of manual segmentations would be closer together than for random segmentations, but this is not the case for any of the comparison groups.

multiple comparisons, as we are, increases the likelihood of false discoveries, unless corrections are made to this threshold [126]. False discoveries are findings that the null hypothesis should be rejected when it is actually true, or in this case, that the medians of the manual and randomly-generated segmentations are different when they are not. Thus, to adjust for multiple comparisons and lower the false discovery rate, we applied the Bonferroni correction, which results in a threshold of 0.05/81 (the number of comparisons for this data, plus the adjustments in the next section) [126]. The downside of using this correction is that though it reduces the false discovery rate, this comes at the price of failing to find many of the actually false null hypotheses (i.e., cases in which the medians truly are different).

Because the Bonferroni correction depends on the number of comparisons made, and because of this trade-off between using a more stringent threshold to reduce false discoveries at the expense of potentially missing true discoveries, we avoid making unnecessary comparisons. Specifically, we test only those properties in which the difference (or lack thereof) between manual and randomly-generated segmentations is ambiguous. We did not test the data in Figures 5.21 and 5.22, which show the percent of sheet switches and new sheets that occur at segment boundaries and vice-versa, because of the clear differences between the manual and randomly-generated segmentations in this case. We did not test the data

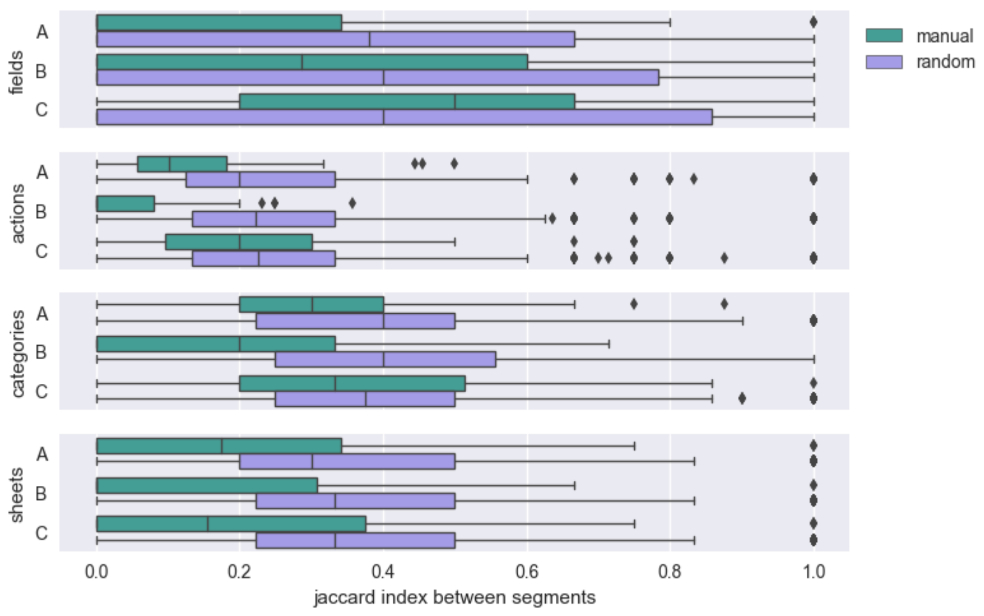


Figure 5.28: Similarity between property sets in pairs of adjacent segments: This shows the distribution of the median Jaccard index between pairs of adjacent segments' sets of actions, categories, fields, or sheets. We would expect the Jaccard index, and thus the similarity, between segments in manual segmentations to be lower than for random segmentations. This is true for all properties in all comparison groups except for Group C's fields.

in Figure 5.24, the number of new sheets created within each segment, because new sheet creations are so infrequent relative to the number of events and segments that the medians in all cases are zero, with an interquartile range of zero, and outliers only above the median. Given these distributions, the medians of the manual and randomly-generated segmentations are not likely to be truly different. Lastly, we did not test the data in Figure 5.27, the seconds between events within each segment, because the relationship between the medians were opposite to what we expected, and it is unclear how to incorporate this fact into a segmentation heuristic even if the difference is significant.

Table 5.6 lists the results of applying Mood's median test for the subset of properties we chose to compare, with respect to both the standard and the adjusted significance thresholds. Both are shown so that the impact of using the lowered threshold can be seen. As we are interested in what characteristics or properties shown in the graphs constitute good segment boundary indications, we focus on the properties for which the difference between the medians of the manual and randomly-generated segmentations are significant across all segmentation groups (A, B, and C). We find that only the results for the similarity of actions between adjacent segments, and the similarity of sheets between adjacent segments, both shown in Figure 5.28, indicate a significant difference between more than one of the manual segmentations and their corresponding random counterpart. The similarity of sheets between

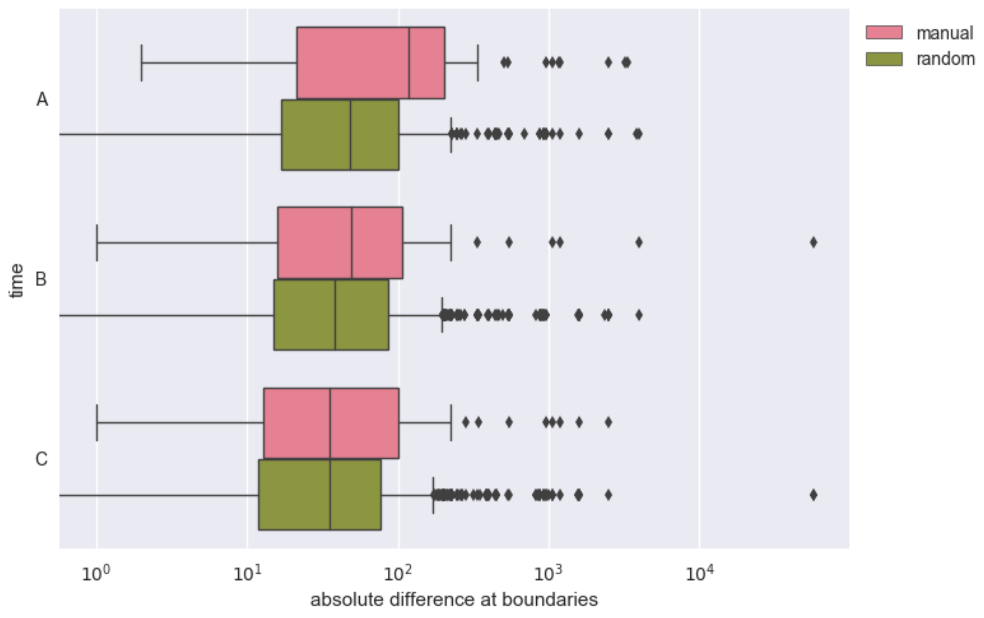


Figure 5.29: Number of seconds elapsed between consecutive segments: This figure shows the distribution of the number of seconds between adjacent pairs of segments in each segmentation. We would expect that the gap between segments in manual segmentations is greater than for random segmentations. This is true for comparison Groups A and B but not C.

adjacent segments is significant for all segmentations at the higher threshold, however. This, combined with the results of Figures 5.21 and 5.22, indicates that sheet switches are a promising indication that a new task has been started, and that a new segment should thus be created.

The distribution of segment lengths is presented in Figure 5.30. There are 2799 events in total in this submission, so if the segmentations each divided these into exactly equal-sized segments, they would each be 36-53 events in length, varying by group, depending on the number of segments created. Manual and random segments both tend to be shorter than this, with outliers on the long side, though the random segmentations are closer to the number expected if they were exactly uniformly divided. This information provides context for the other properties, because more short segments are usually more similar internally than fewer long segments.

Agreement Between Manual Segmentations After considering the above properties of manual segmentations, we next ask how much agreement there is between them, to see if human judges apply similar principles when grouping consecutive sequences of events to summarize. These results are show in Figure 5.31.

Figure	Description	Segmentation	p-value	Cut-off
5.23	number of sheet switches within each segment	A	9.87E-01	
5.23	number of sheet switches within each segment	B	6.98E-04	*
5.23	number of sheet switches within each segment	C	2.30E-01	
5.25	similarity of actions within each segment	A	4.26E-02	*
5.25	similarity of actions within each segment	B	8.57E-02	
5.25	similarity of actions within each segment	C	1.51E-01	
5.25	similarity of categories within each segment	A	8.38E-02	
5.25	similarity of categories within each segment	B	2.57E-01	
5.25	similarity of categories within each segment	C	3.36E-04	**
5.26	similarity of fields within each segment	A	7.32E-01	
5.26	similarity of fields within each segment	B	2.84E-01	
5.26	similarity of fields within each segment	C	8.45E-01	
5.28	similarity of fields between adjacent segments	A	2.14E-04	**
5.28	similarity of fields between adjacent segments	B	9.29E-02	
5.28	similarity of fields between adjacent segments	C	3.19E-01	
5.28	similarity of actions between adjacent segments	A	6.45E-06	**
5.28	similarity of actions between adjacent segments	B	2.10E-15	**
5.28	similarity of actions between adjacent segments	C	2.37E-01	
5.28	similarity of categories between adjacent segments	A	2.28E-02	*
5.28	similarity of categories between adjacent segments	B	6.27E-09	**
5.28	similarity of categories between adjacent segments	C	9.64E-02	
5.28	similarity of sheets between adjacent segments	A	1.23E-02	*
5.28	similarity of sheets between adjacent segments	B	1.02E-08	**
5.28	similarity of sheets between adjacent segments	C	1.72E-04	**
5.29	time between adjacent segments	A	1.95E-02	*
5.29	time between adjacent segments	B	2.71E-01	
5.29	time between adjacent segments	C	9.22E-01	

Table 5.6: Significance of differences between manual and randomly-generated segmentations: P-values resulting from Mood's median test of the difference between medians for select figures in this section. P-values with one star are less than the cut-off $\alpha = 0.05$ and those with two stars are less than the cut-off of $\alpha = 0.05/81$, which is the cut-off used after applying the Bonferroni correction for multiple comparisons for the comparisons shown in this table, plus those of the adjustments discussed in the next section.

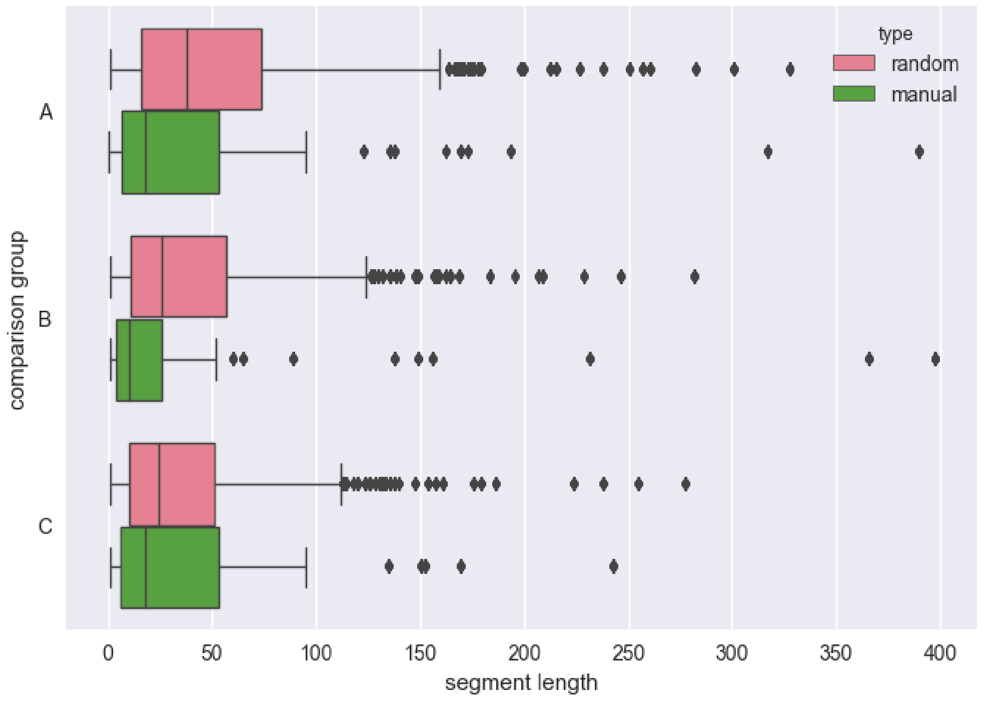
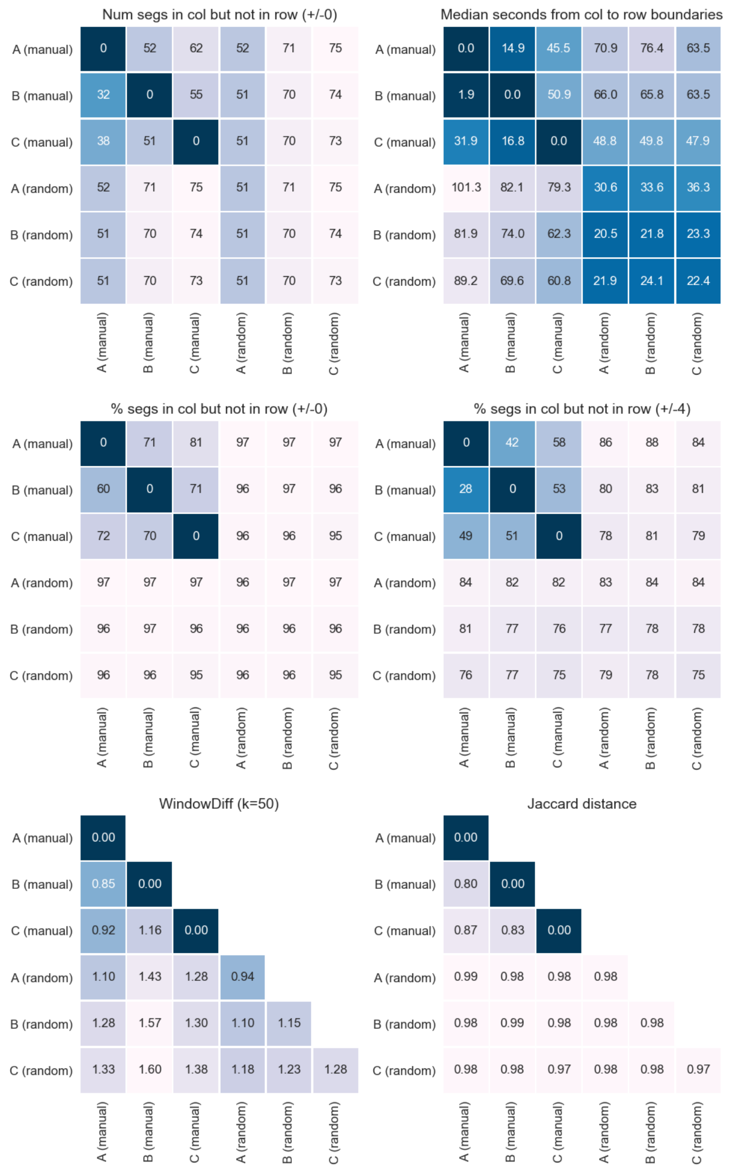


Figure 5.30: Segment lengths: This figure shows the distribution of lengths of segments for the segmentations in each of the comparison groups. Manual and random segmentations are shown for comparison. In this submission, if the segmentations each divided these into exactly equal-sized segments, they would each be 36-53 events in length (varying by group, depending on the number of segments created). Manual and random segments both tend to be shorter than this.

To assess agreement as shown in Figure 5.31, for each pair of segmentations, we computed:

- The number and percent of segment boundaries in the first segmentation that are not present within k events of the second
- The median number of seconds from each segment boundary of the first segmentation to the closest segment boundary of the second
- The WindowDiff between each pair of segmentations – WindowDiff is a metric designed to penalize near-misses less than pure false positives and pure false negatives [121]
- The Jaccard index between each pair of segmentations' segment boundaries sets

The metrics listed in the first two items are not symmetric between pairs, while the last three are. Moreover, the metrics in the first two items improve as the number of segments on one side of the comparison increases, while that is not true for the latter. For the comparison between randomly-generated segmentations and manual segmentations, we take the median value of the comparison between each manual segmentation and all ten randomly-generated



Num segs in col but not in row (+/-0)

	A (manual)	B (manual)	C (manual)	A (random)	B (random)	C (random)
A (manual)	0	52	62	52	71	75
B (manual)	32	0	55	51	70	74
C (manual)	38	51	0	51	70	73
A (random)	52	71	75	51	71	75
B (random)	51	70	74	51	70	74
C (random)	51	70	73	51	70	73

Median seconds from col to row boundaries

	A (manual)	B (manual)	C (manual)	A (random)	B (random)	C (random)
A (manual)	0.0	14.9	45.5	70.9	76.4	63.5
B (manual)	1.9	0.0	50.9	66.0	65.8	63.5
C (manual)	31.9	16.8	0.0	48.8	49.8	47.9
A (random)	101.3	82.1	79.3	30.6	33.6	36.3
B (random)	81.9	74.0	62.3	20.5	21.8	23.3
C (random)	89.2	69.6	60.8	21.9	24.1	22.4

% segs in col but not in row (+/-0)

	A (manual)	B (manual)	C (manual)	A (random)	B (random)	C (random)
A (manual)	0	71	81	97	97	97
B (manual)	60	0	71	96	97	96
C (manual)	72	70	0	96	96	95
A (random)	97	97	97	96	97	97
B (random)	96	97	96	96	96	96
C (random)	96	96	95	96	96	95

% segs in col but not in row (+/-4)

	A (manual)	B (manual)	C (manual)	A (random)	B (random)	C (random)
A (manual)	0	42	58	86	88	84
B (manual)	28	0	53	80	83	81
C (manual)	49	51	0	78	81	79
A (random)	84	82	82	83	84	84
B (random)	81	77	76	77	78	78
C (random)	76	77	75	79	78	75

WindowDiff (k=50)

	A (manual)	B (manual)	C (manual)	A (random)	B (random)	C (random)
A (manual)	0.00					
B (manual)	0.85	0.00				
C (manual)	0.92	1.16	0.00			
A (random)	1.10	1.43	1.28	0.94		
B (random)	1.28	1.57	1.30	1.10	1.15	
C (random)	1.33	1.60	1.38	1.18	1.23	1.28

Jaccard distance

	A (manual)	B (manual)	C (manual)	A (random)	B (random)	C (random)
A (manual)	0.00					
B (manual)	0.80	0.00				
C (manual)	0.87	0.83	0.00			
A (random)	0.99	0.98	0.98	0.98		
B (random)	0.98	0.99	0.98	0.98	0.98	
C (random)	0.98	0.98	0.97	0.98	0.98	0.97

Figure 5.31: Comparison of segment boundaries between pairs of segmentations: In all cases, the manual segmentations are more similar to one another than they are to randomly-generated segmentations. The heatmaps, left to right and top to bottom, are (a) the number of segment boundaries that are in the column segmentation but not the row segmentation, (b) the median number of seconds from each segment boundary of the first segmentation to the closest segment boundary of the second, (c) the percent of segment boundaries in the column segmentation but not the row segmentation, (d) the same as (c) except counting matches as those within plus or minus four events, (e) the WindowDiff between pairs of segmentations, and (f) the Jaccard index between pairs of segmentations' boundary sets.

segmentations of the same length.

When considering the number and percent of exact matches in choice of segment boundaries, there appears to be much disagreement between segmenters. For each pair of segmentations, the percent of boundaries present in one that are not present in the other ranges from 60-81%. However, segmenters' reasoning seems more aligned when considering other metrics, as well as comparison to randomly-generated segmentations. In all cases at least, the manual segmentations are more similar to one another than they are to the randomly-generated segmentations.

5.3.3.2 Adjustments

In this section we describe some corrections we made to the manual segmentations to try to correct where the segmentation rules may have been inconsistently applied.

Sources of Disagreement and Adjustments Made We manually inspected the segmentations to identify possible reasons for disagreement between them, and then reconcile these differences where possible, by making adjustments. We combined the lists of event IDs that denote the beginning of each segment from each manual Segmentation A, B, and C, into a spreadsheet. To this list of event IDs, we later added all the event IDs chosen as segment boundaries by the heuristics, described in Section 5.3.4. We then iteratively examined each of these event IDs as possible segment boundaries. We identified several sources of disagreement between segmentations in their choice of segment boundaries, though we did not attempt to classify all instances of disagreement. We attempted to reconcile some of the disagreement between, or inconsistency within, some of the choices of event IDs, by proposing specific adjustments to add, remove, or change event IDs. Below we describe the sources of disagreement that resulted in adjustments. We created two sets of adjustments:

- Adjustment 1: We applied the adjustments for the differences described in items 1-3a below, yielding segmentations A', B', C'. These correspond to the more subjective adjustments.
- Adjustment 2: We applied all of the adjustments below to obtain segmentations A", B", and C". This subsumes all of the adjustments in 1 and includes more.

The sources of disagreement were as follows:
Mistakes: To denote segment boundaries, segmenters seemed to, on occasion, accidentally mark down a different event than they intended, or to be inconsistent in the convention they used for denoting boundaries. There are three types of mistakes we observed:

- Marked end not beginning of segment: One segmenter alternated between marking the the end of a segment versus marking the beginning of a segment to denote segment boundaries, making comments like "finished that hypothesis" in the former case, versus "starting a new hypothesis" in the latter. For consistency, we adjusted all of the segmentations so that each segment is denoted by its first event, rather than by the previous segment?s last event.

- Off-by-N event index: Segmenters occasionally wrote down an event index that seemed to be slightly off from the index they seem to have intended, based on their accompanying comment. For example, the segmenter might have written down event 876 as the start of a new segment, along with a comment like "new dashboard", when in the actual event stream, is was event 873, three events prior, that marked the beginning of that sequence of dashboarding activity, starting with the creation of a new dashboard. In these cases, we adjusted the event index to match the commentary.

- Typo: One segmenter was prone to making what we think are typos due to copy-paste errors. The segmentation of segmenter C, the study participant, was derived from their assignment annotation post-facto, rather than via the same segmentation instructions given to the other segmenters. In creating their annotations, the participant followed our instructions, which asked that they prefix every comment on the same segment with the same description, so that we could use the description prefix to segment. Because these descriptions tended to be long, and because it had to be re-entered with every comment, it makes sense that participants would copy and paste the description when adding a comment to a segment. It was clear from context, such as the actions performed, the current sheet, and the content of their comments, that in several cases the participant copied and pasted the wrong segment description, inadvertently marking it as a new segment. We corrected this by using the segment description it seemed they must have intended.

LogViewer: As mentioned, the segmentation of segmenter C, the study participant, was derived from their assignment annotation post-facto, rather than via the same segmentation instructions given to the other segmenters. Participants used the Tableau research tool called LogViewer (described in Section 5.2.3) to annotate their logs, and their segmentations are derived from these annotations. We observed two types of issues arising from this:

- Not shown by LogViewer: LogViewer does not show all events, so participants cannot access and therefore cannot have marked these not-shown events as the start of a new segment. This causes discrepancies between the participant's segmentation and our other segmentations, because the other segmenters did consider the complete event sequence. The discrepancies are especially apparent for dashboard editing and data editing sequences, as these were most consistently omitted from LogViewer. For example, segmentations A and B both designate the first several events as being part of a data editing segment, but because most of these events were not shown by LogViewer, the participant did not identify this segment.

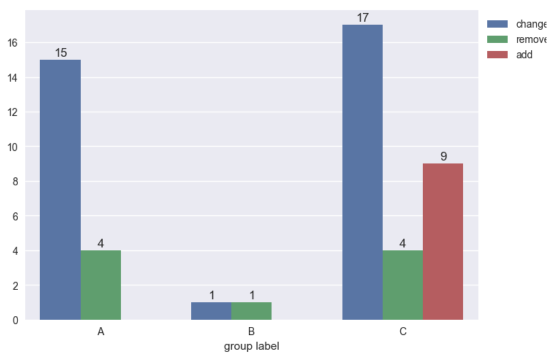
- Faulty annotation leading to faulty segmentation: In addition to affecting the participant's segmentations, LogViewer's not showing all events also could have caused problems if the other segmenters used the participant's comments to guide their own segmentation. In fact we did see this in one instance.

Segmentation rule ambiguity: The segmentation rules we suggested were not complete enough to determine how to decide in every case whether an event should be considered the start of a new segment. For example: sometimes the participant would create a new sheet, but then immediately switch back to what they were doing before on the previous sheet for several consecutive events. The rules say to segment on sheet switch, which implies that a new segment should be marked when the new sheet was created. However, the rules also say to group together like activity on the same sheet, which suggests that a new segment should not be created, since the participant didn't really change what they were doing and only spent one event out of a long sequence on a different sheet. It is often difficult to know how to break up certain sequences of sheet switches and activities, because it can be difficult to understand what the participant's goal was. We suspect that even if we did, there could still be legitimate differences of opinion on how to segment. Thus, we are not able to as objectively resolve all of these differences as we could in the above cases. Of all these more subjective cases, we addressed three:

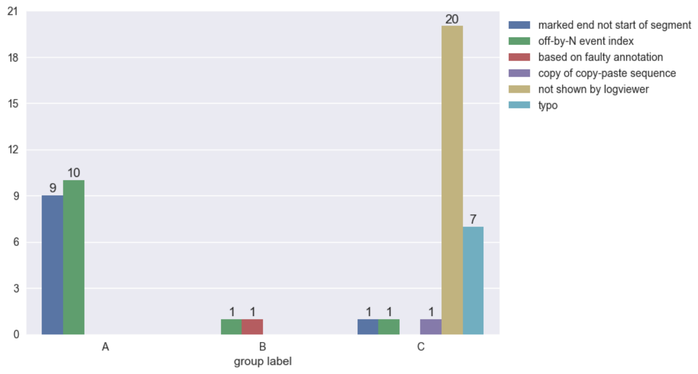
- Copy of copy-paste sequence: This is one exception in terms of objectivity – we dictate that for consistency, when there is a sequence of sheet copy-paste events (e.g., create new sheet that is a copy of an previous sheet) that everyone agrees indicates a segment boundary, the paste event should be marked rather than the copy event, for consistency.

- Missed sheet switch: The segmentation instructions we gave stipulated that segmenters should consider creating a new segment when the sheet being worked on changed; however, if a new segment were created every time, there would be far too many segments (almost 300). So whether a sheet switch warrants a new segment is a judgment call, barring the development of a more specific segmentation strategy. We added segment boundaries upon missed sheet switches according to our judgment.

- Segment too small: An important characteristic of a segmentation is granularity – what size should segments be? There is a trade-off. More, finer-grained segments will summarize fewer events, possibly allowing more internally homogeneous segments and more accurate summary descriptions; however, will result in more segments, possibly becoming too long to be an effective summary. Fewer, coarser-grained segments will summarize more events, providing more condensed overviews of the event stream, but the segments may not be as accurately described. Although our segmentation instructions did not specify anything about segmentation length explicitly, it could be argued that in some cases, having very short segments is not an effective summarizing strategy, because short segments summarize fewer events, which might be more appropriately summarized as part of a larger group of adjacent events. We removed segment boundaries that led to segments that were too short according to our judgment.

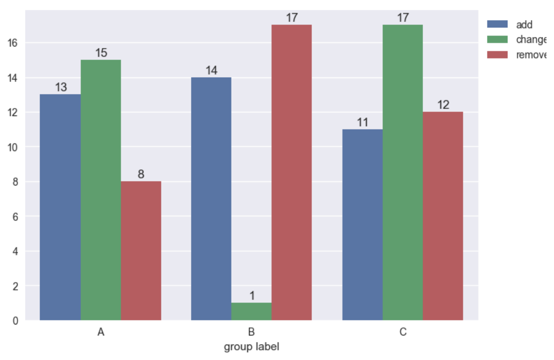


(a) Type and number of adjustments.

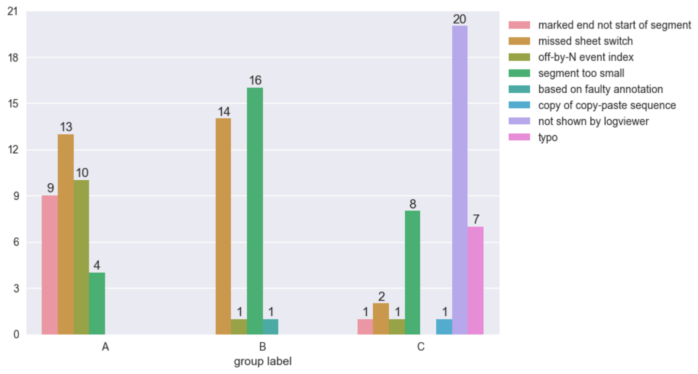


(b) Reason for the adjustments.

Figure 5.32: First adjustments to the original manual segmentations.



(a) Type and number of adjustments.



(b) Reason for the adjustments.

Figure 5.33: Second adjustments to the original manual segmentations.

The number of and reasons for adjustments made for each of A, B, and C are summarized in Figures 5.32 - 5.33.

Properties of Adjustments The properties of the adjusted segmentations, shown in Figures 5.34 - 5.42, are similar to those of the original segmentations. The adjustments are mostly a slight improvement with respect to our expectations, though the second adjustment is sometimes worse than the first (e.g., Figures 5.41 fields). Where our expectations were met or mostly met before, the adjustments are either about the same, or improve in the direction of becoming more in line with our expectations (Figures 5.36, 5.37, 5.39, 5.41 fields, actions, and sheets, and 5.42). This is true for Segmentation C's inter-segment field sets, though the median only improves to being equal (Figure 5.41). The adjustments to Segmentation C mainly correct for idiosyncrasies and errors in that segmentation that were due to the use of LogViewer. Where our expectations were invalidated before, however, they are even more so by the adjustments (Figure 5.38 and Figure 5.40). The adjustments move in the direction

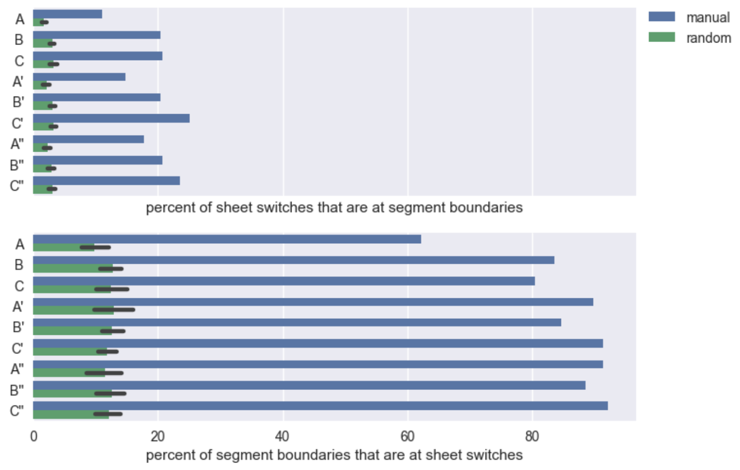


Figure 5.34: Sheet switches and segment boundaries: These figures show the percent of sheet switches in the submission that occur at segment boundaries, as well as the percent of segment boundaries that occur at sheet switches. We would expect that manual segmentations to have a higher percent of both than random segmentations and that is the case for all comparison groups, with the proportion increasing in the adjusted segmentations.

opposite our expectations in one case, and this is for the adjustments for Segmentation C's inter-segment categories (5.41 categories). These results are summarized in Table 5.7.

Agreement Between Adjustments As intended, the adjustments brought the segmentations more in line with one another than before. The Adjustment 1 segmentations are more similar to one another than to the random segmentations or than the random segmentations are to each other, with the exception of Segmentation C' and has tended to violate our expectations in our analysis thus far. The Adjustment 2 segmentations are more similar to one another than to the random segmentations or than the random segmentations are to each other. This combined with the previous graph suggests we should perhaps use Adjustment 2 instead of Adjustment 1, which is contrary to what other figures suggested. The Adjustment 1 segmentations are also preferable because they involve fewer changes to the originals.

As in the last section, we again test whether the property metric differences between the adjusted segmentations and randomly-generated segmentations are significant. To do this, we again applied Mood's median test to the equivalent adjusted datasets, using the same parameters as before (ignoring values equal to the grand median), and again used a threshold of 0.05/81, the cut-off used after applying the Bonferroni correction for multiple comparisons [100, 126]. Table 5.8 lists the results. The only notable difference is that after the second adjustment, all of the adjusted segmentations are significantly different from the randomly generated segmentations in terms of the similarity of the sheets between adjacent segments (Figure 5.41).

Type	**Property**	**Metric**	**Interpretation**	**Expected**	**Verdict**	**True**	**False**	**Equal**
intra	action	Herfindahl index	higher when fewer actions dominate the segment	manual >random	all false		all	
intra	category	Herfindahl index	higher when fewer categories dominate the segment	manual >random	all false		all	
intra	fields	median Jaccard index	higher when field sets of consecutive events are more similar	manual >random	all true	all		
intra	new sheets	counts	higher when there are more new sheet creations in the segment	manual <random	ambiguous			all
intra	sheet switches	counts	higher when there are more sheet switches in the segment	manual <random	mostly true	B, A’, B’, A”, B”	C	A, C’, C”
intra	time	difference	higher when there is more time between events within the segment	manual <random	all false		all	
inter	action	median Jaccard index	higher when action sets of consecutive segments are more similar	manual <random	all true	all		
inter	category	median Jaccard index	higher when category sets of consecutive segments are more similar	manual <random	mostly true	A, B, C, A’, B’, C’, B”	A”, C”	
inter	fields	median Jaccard index	higher when field sets of consecutive segments are more similar	manual <random	mostly true	A, B, A’, B’, C’, A”, B”	C	C”
inter	sheet	median Jaccard index	higher when sheet sets of consecutive segments are more similar	manual <random	all true	all		
inter	time	difference	higher when there is more time between segments	manual >random	mostly true	A, B, A’, B’, C’, A”, B”, C”		C
boundary	new sheets	percent	percent of new sheet creations at a boundary	manual >random	all true	all		
boundary	new sheets	percent	percent of boundaries at a new sheet creation	manual >random	all true	all		
boundary	sheet switches	percent	percent of sheet switches at a boundary	manual >random	all true	all		
boundary	sheet switches	percent	percent of boundaries at a sheet switch	manual >random	all true	all		

Table 5.7: Summary of expectations and results of metrics applied to boundary, intra-, and inter-segment properties.

Figure	Property	Segment-ation	p-value	Significant	Segment-ation	p-value	Significant
5.36	sheet switches	A’	5.97E-01		A”	8.11E-01	
5.36	sheet switches	B’	1.63E-02	*	B”	2.36E-01	
5.36	sheet switches	C’	7.68E-01		C”	8.95E-01	
5.38	actions	A’	6.99E-03	*	A”	5.76E-04	**
5.38	actions	B’	9.44E-03	*	B”	3.94E-03	*
5.38	actions	C’	1.97E-04	**	C”	2.63E-03	*
5.38	categories	A’	7.43E-02		A”	1.92E-02	*
5.38	categories	B’	1.74E-01		B”	5.30E-03	*
5.38	categories	C’	7.83E-04	*	C”	5.66E-03	*
5.39	fields	A’	8.80E-01		A”	7.12E-01	
5.39	fields	B’	3.33E-01		B”	4.49E-01	
5.39	fields	C’	2.09E-01		C”	2.90E-02	*
5.41	fields	A’	1.15E-03	*	A”	1.42E-01	
5.41	fields	B’	3.08E-01		B”	2.96E-01	
5.41	fields	C’	7.54E-01		C”	9.54E-01	
5.41	actions	A’	1.04E-03	*	A”	1.35E-02	*
5.41	actions	B’	1.62E-16	**	B”	3.64E-09	**
5.41	actions	C’	5.88E-01		C”	8.09E-01	
5.41	categories	A’	4.81E-02	*	A”	8.35E-01	
5.41	categories	B’	1.16E-07	**	B”	1.32E-03	*
5.41	categories	C’	6.77E-01		C”	2.99E-01	
5.41	sheets	A’	1.86E-03	*	A”	4.07E-05	**
5.41	sheets	B’	4.59E-08	**	B”	2.43E-06	**
5.41	sheets	C’	1.90E-07	**	C”	4.68E-04	**
5.42	time	A’	5.22E-03	*	A”	2.14E-03	*
5.42	time	B’	5.16E-01		B”	2.03E-01	
5.42	time	C’	3.46E-01		C”	2.72E-02	*

Table 5.8: Significance of differences between adjusted and randomly-generated segmentations: P-values resulting from Mood’s median test of the difference between medians for select figures in this section. P-values with one star are less than the cut-off $\alpha = 0.05$ and those with two stars are less than the cut-off of $\alpha = 0.05/81$, which is the cut-off used after applying the Bonferroni correction for multiple comparisons.

,

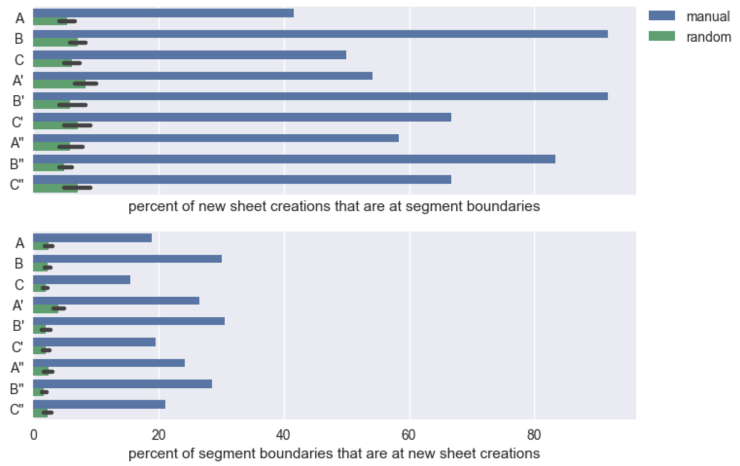


Figure 5.35: New sheet creations and segment boundaries: These figures show the percent of new sheet creations in the submission that occur at segment boundaries, as well as the percent of segment boundaries that occur at new sheet creations. We would expect manual segmentations to have a higher percent of both than random segmentations and that is the case for all comparison groups. The proportion increases in the adjustments for all segmentations, with the exception of Adjustment 2 for Segmentation B , resulting in segmentation H. Likely, removing segment boundaries to eliminate segments that were deemed to be too short is the reason why.

5.3.4 Segmentation Heuristics

The patterns observed in the manual segmentation and adjustments properties suggest some possible segmentation heuristics. The greatest difference between manual segmentations and random segmentations is in the proportion of segment boundaries that occur at new sheet creations and sheet switches, as well as the proportion of these that occur at segment boundaries.

With respect to the set of sheets visited across pairs of adjacent segments, the differences between the manual and adjusted segmentations, and their random counterparts, were significant for more segmentation groups at the more stringent significance threshold than was the case for any other property. Moreover, identifying sheet switches and new sheet creations is easy. Thus, these are good candidate indicators to use in a segmentation heuristic.

For some reason that we have yet to determine, the concentration of actions and categories within segments from manual segmentations were not higher than for segments from random segmentations. However, it is mostly true that the set of actions and categories of consecutive segments in manual segmentations tend to differ more than in random segmentations. This suggests that the action or category concentration of a set of consecutive events is perhaps not a good indicator of when to segment.

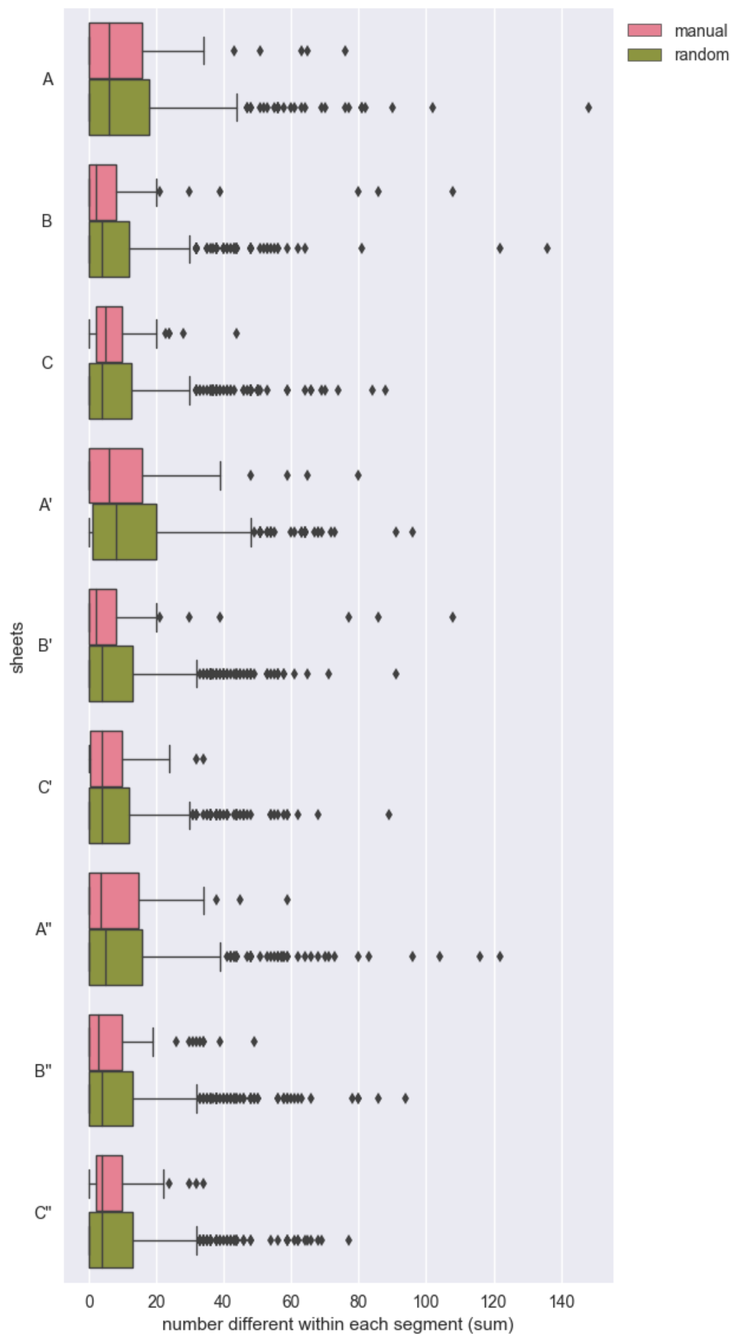


Figure 5.36: Distribution of the number of sheet switches within each segment: We expected manual segmentations to have fewer sheet switches within each segment, with sheet switches occurring more often on segment boundaries (these were not included as part of the within-segment count). This is true for all B (B, B', and B"), for adjusted A (A' and A"), but not for any C (C, C', and C").

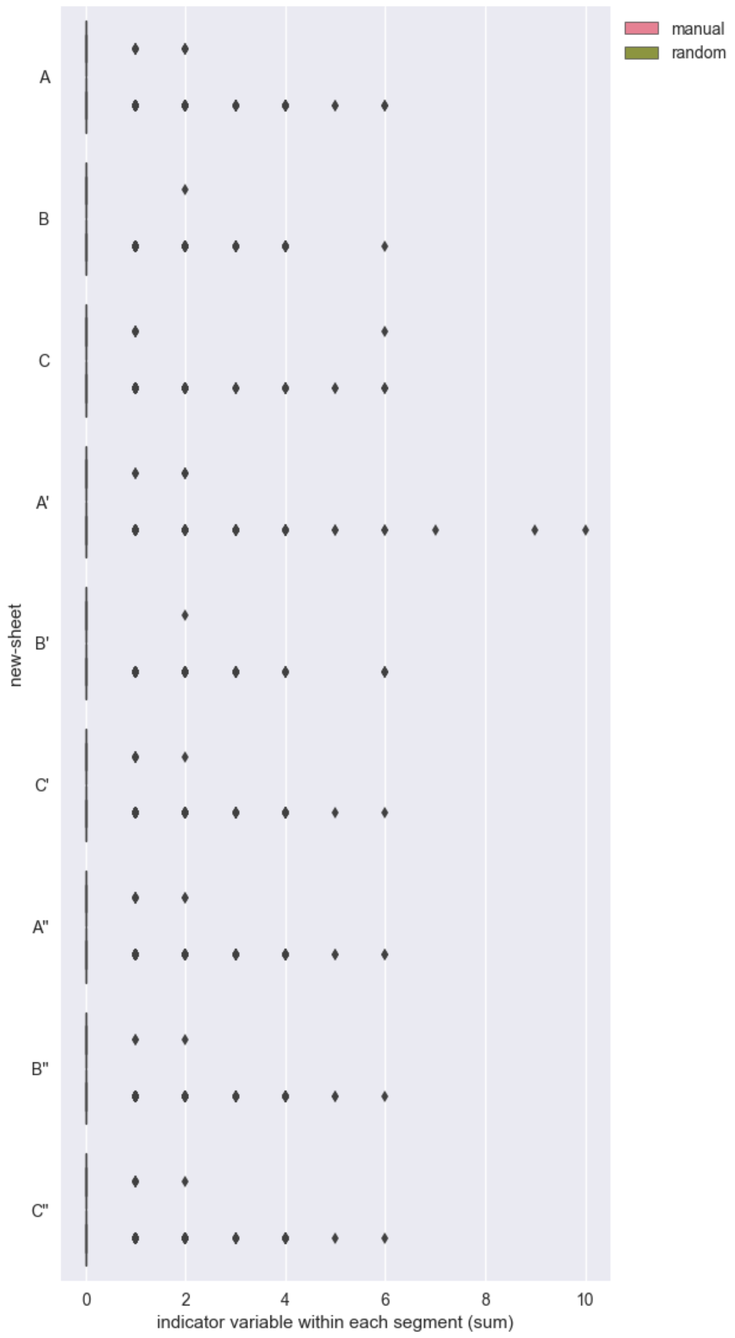


Figure 5.37: Distribution of the number of new sheets within each segment: We expected manual segmentations to have fewer new sheets created within each segment, because new sheet creations would occur more often at segment boundaries (these were not included as part of the within-segment count). However, new sheet creations are infrequent enough that the median number of new sheet creations for all segmentations is zero.

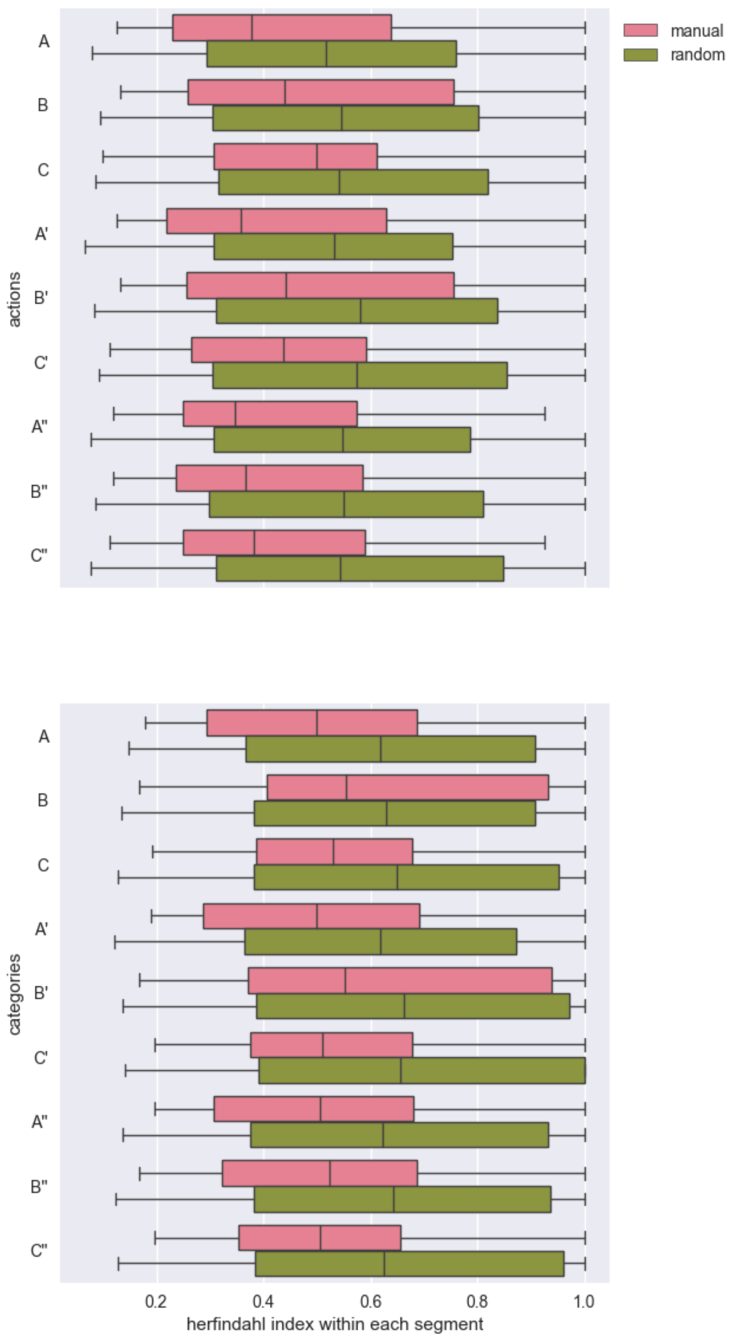


Figure 5.38: Concentration of the distribution of actions and action categories within each segment: This figure shows the distribution of the Herfindahl index of the action and category sets. A higher number means the set is dominated by fewer actions or categories, respectively. We expected manual segmentations to have a higher Herfindahl index than random; these results would seem to contradict that for all comparison groups.

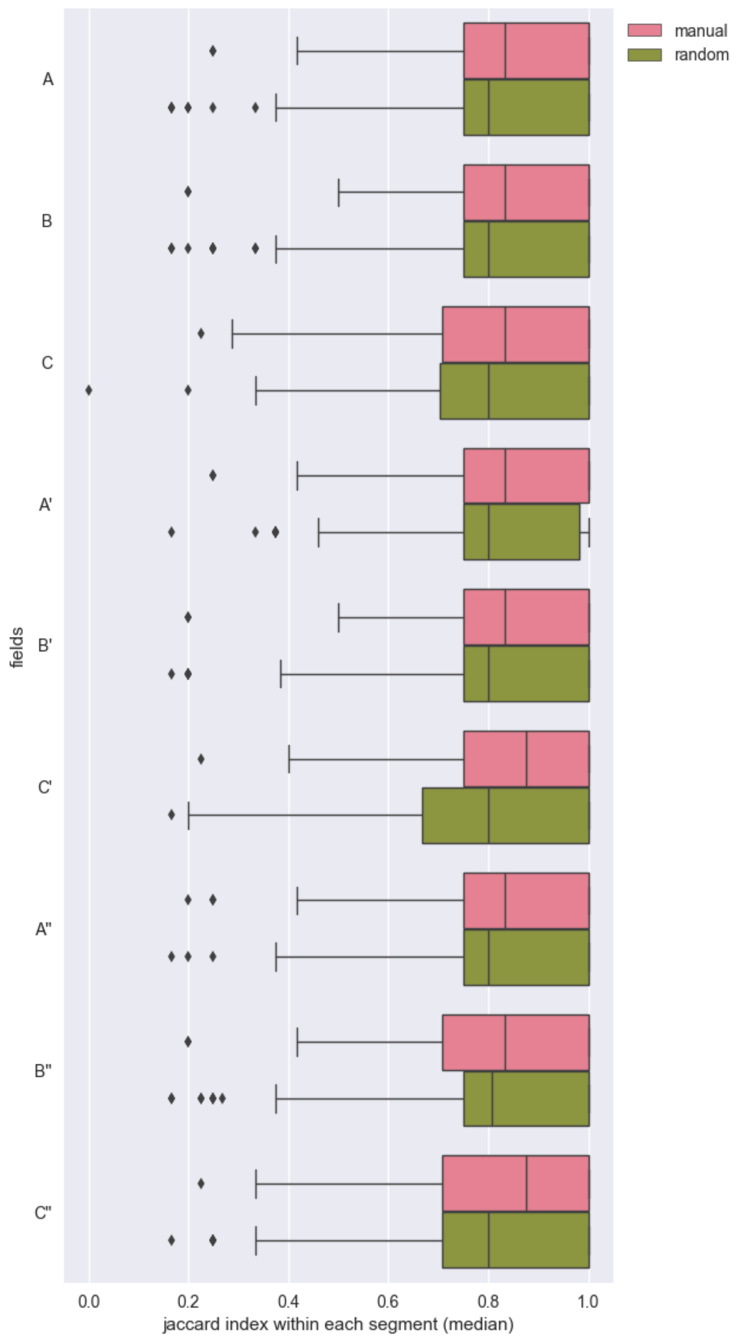


Figure 5.39: Similarity of the fields sets within each segment: This figure shows the distribution of the median Jaccard index (a measure of set similarity) of the field sets of each pair of adjacent events in each segment. A higher Jaccard index means greater similarity. We would expect manual segmentations to have higher similarity between the field sets of their segments' events. This is true for all segmentations.

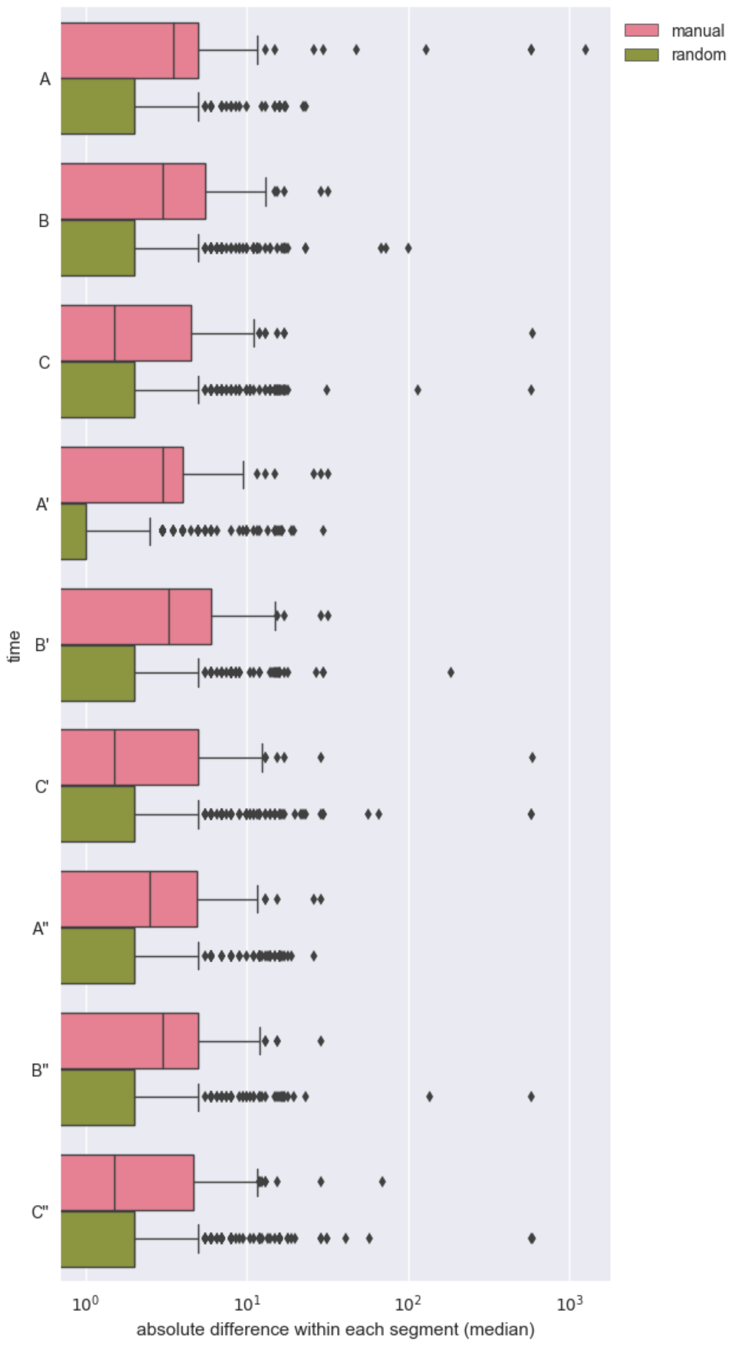


Figure 5.40: Number of seconds elapsed between consecutive events within each segment: This figure shows the distribution of the median number of seconds elapsed between consecutive pairs of events within each segment. We would expect that events within the segments of manual segmentations would be closer together than for random segmentations, but this is not the case for any of the comparison groups, nor the adjustments.

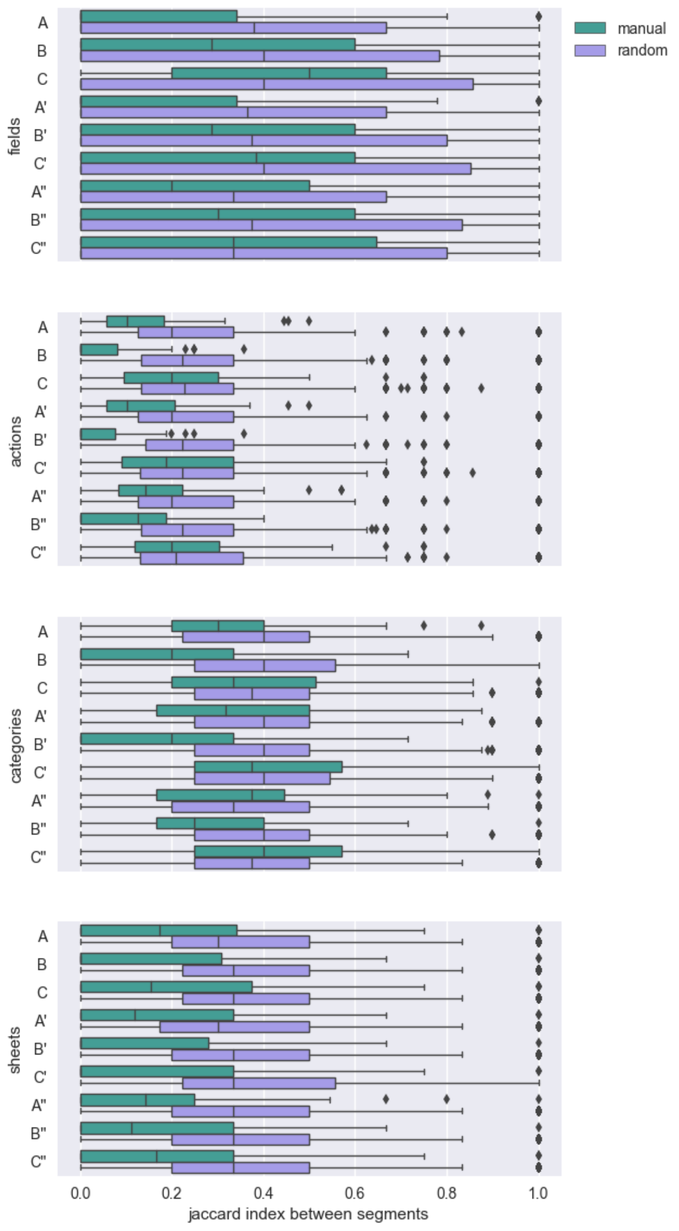


Figure 5.41: Similarity between property sets in pairs of adjacent segments: This shows the distribution of the median Jaccard index between pairs of adjacent segments' sets of actions, categories, fields, or sheets. We would expect it to be lower for manual than for random segmentations. Earlier we saw that is true for all properties in all comparison groups except for Group C's fields. Now, we see that it is true for all properties for Adjustment 1, showing an improvement. However, surprisingly, Adjustment 2 are in some cases are even worse than the original. This suggests we may want to reject these and possibly remove the reasons for making them from our segmentation strategy.

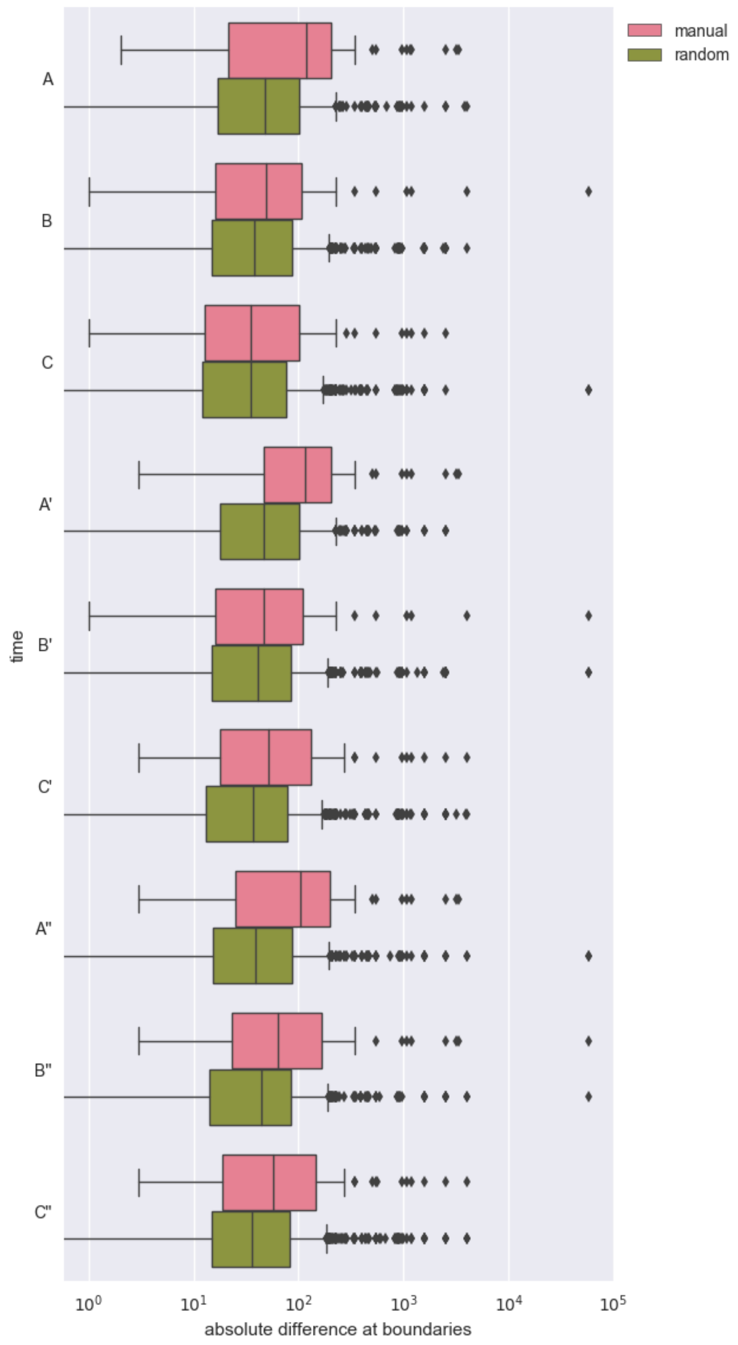


Figure 5.42: Distribution of the number of seconds elapsed between consecutive segments: We would expect that the gap between segments in manual segmentations is greater than for random segmentations. This is true for all segmentations in the figure except the original participant's segmentation, C, which as we've said, was created in a different way, importantly, lacking a view of the complete event sequence.

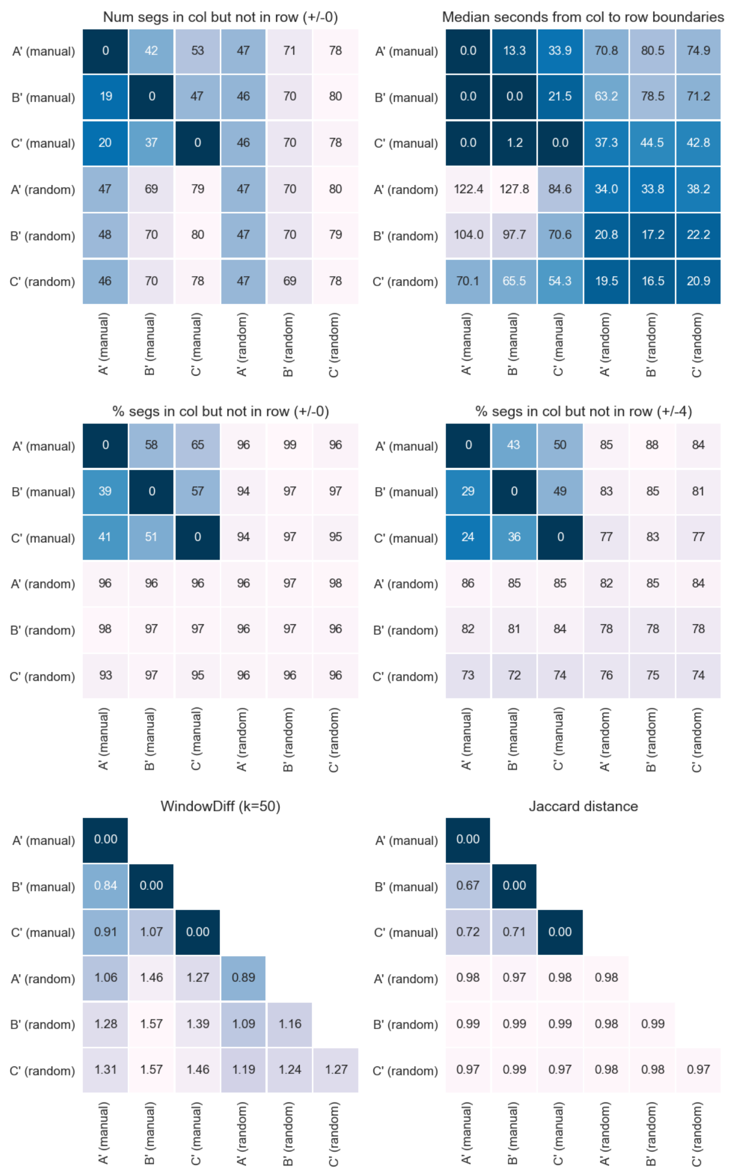


Figure 5.43: Similarity in choice of segment boundaries between each pair of segmentations: The segmentations compared here are the segmentations derived from Adjustment 1, along with randomly-generated segmentations of the same length, for comparison. The Adjustment 1 segmentations are more similar to one another than to the random segmentations or than the random segmentations are to each other, with the exception of Segmentation C'.

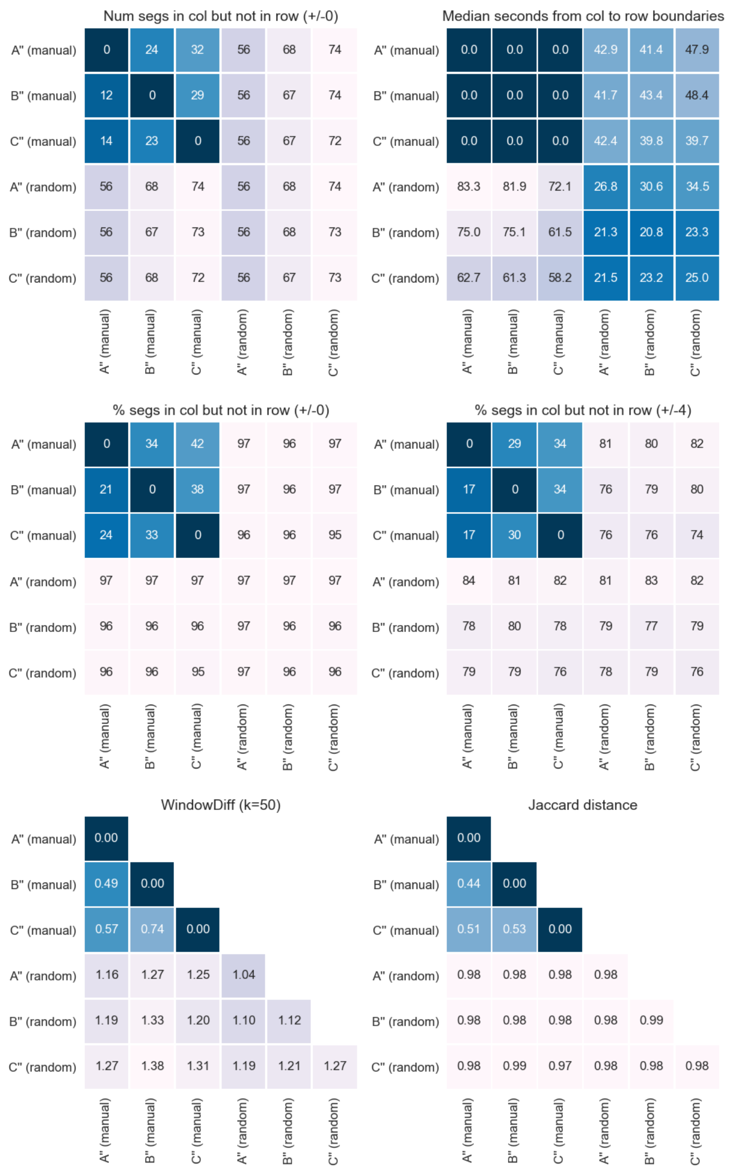


Figure 5.44: Similarity in choice of segment boundaries between each pair of segmentations: The segmentations compared here are the segmentations derived from Adjustment 2, along with randomly-generated segmentations of the same length, for comparison. The Adjustment 2 segmentations are more similar to one another than to the random segmentations or than the random segmentations are to each other. This combined with the previous graph suggests we should perhaps use Adjustment 2 instead of Adjustment 1, which is contrary to what other figures suggested.

Within segments in manual segmentations, the field sets of consecutive events tend to be more similar than within segments in random segmentations. Also, the field sets of consecutive segments in manual segmentations tend to be more different than the field sets of consecutive segments in random segmentations. Thus, field set similarity could be a good indicator to use to segment.

For reasons that we do not yet understand, the amount of time elapsed between events within segments in manual segmentations tends to be higher than for random segmentations, while at the same time, the amount of time elapsed between segments in manual segmentations is higher than for those in random segmentations. The former suggests that time is perhaps not a reliable indicator of when to segment, though the latter suggests that it is.

Given these results, we decided to first focus on sheet switches, which subsume new sheet creations, as the basic indicator to use in our first pass heuristics. The simplest approach would be to designate every sheet switch as a segment boundary. However, there are too many such occurrences. There are about 300 sheet switches, in contrast to at most around 80 segments in the manual segmentations. Which sheet switches should we omit? In our manual inspection of the submission and segmentations, we noticed two types of behavior that suggest which might be safely omitted:

1. Switching to a sheet then switching back to the previous sheet after only one or a few events: This happens, for example, when the user is about to start a new task but then remembers something they forgot to do on a previous sheet.

2. Spending only a short amount of time (one to a few events) on a sheet before continuing to the next, for several sequences of sheet switches: This happens, for example, when the user is making some change in common to a number of sheets, such as adjusting all the colors to fit with a common color scheme when they've finished with their analysis

We created three classes of sheet-switch heuristics based on these observations. The first heuristic is the simplest, segmenting at almost every sheet switch. The second heuristic accounts for the first observation just described and ignores the switch-backs. The third heuristic accounts for both observations just described and ignores switch-backs and sequences of short runs. We anticipate the third heuristic will perform the best, given our observations. The precise heuristic definitions are given in the next section.

5.3.4.1 Definitions

In this section, we provide the heuristic definitions. The definitions include the notion of qualified events. Qualified events are those which are specific to a sheet, rather than being a general application-level event, like "save workbook." Almost all events are qualified events. The list of unqualified events is given in Listing 5.3.

Listing 5.3: Heuristic parameters.

```
unqualified = [
    "tabui:switch-to-document",
    "tabui:save-as-workbook",
    "tabui:rename-sheet-ui"
]
qualified_events = [e for e in events if e.action.name not in
   unqualified]
m in {1,2,3,4}
```

Each heuristic starts by grouping the event sequence into consecutive subsequences according to the sheet the event occurred on. We call these subsequences *runs*, and the heuristics consider the length of these runs when deciding whether to segment at the beginning of them. The heuristics are then as follows.

- **Heuristic-1-m:** Segment at the beginning of every run that contains at least m events.
- **Heuristic-2-m:** Same as Heuristic-1, but also exclude switch backs from segment boundaries. In other words, segment at every change in sheet ID, unless the change does not persist for m or more events i.e., the sheet ID switches back to the previous ID in less than m events.
- **Heuristic-3-m:** Same as Heuristic-2, but also segment at the first run less than m in length if it is the beginning of a sequence of runs less than m in length. That is, segment at every change in sheet ID, unless the ID sequence lasts for less than m events and was preceded by a sequence that lasted for less than m events.
- **Heuristic-kq-m:** Same as heuristic-k, except only count qualified events when tallying the run length.

Table 5.9 gives an example of where these heuristics would segment on a hypothetical sequence of qualified events.

5.3.4.2 Properties

We choose four heuristics to analyze further for the following reasons:

- **Heuristic-2-4:** This segmentation (99 segments) was most closely matched to the most segmentations as judged by the WindowDiff metric.
- **Heuristic-2q-4:** This segmentation (95 segments) was second most closely matched by WindowDiff and fewest segments.
- **Heuristic-3q-4:** This segmentation (123 segments) was generated by the heuristic we thought would do best.
- **Heuristic-1q-4:** This segmentation (115 segments) was the best Heuristic-1 performance.

The heuristics exhibit the similar qualities as the manual segmentations and their adjustments. These are shown in Figures 5.45 - 5.53. These results differ from the manual segmentations and adjustments in only a few cases: With respect to the intra-segment field sets, the two best heuristic segmentations, **Heuristic-2-4** and **Heuristic-2q-4**, have the

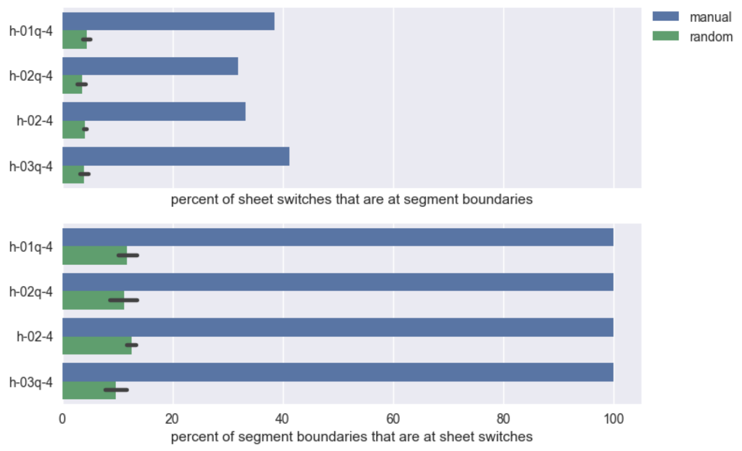


Figure 5.45: Sheet switches and segment boundaries: These figures show the percent of sheet switches in the submission that occur at segment boundaries, as well as the percent of segment boundaries that occur at sheet switches. Because the heuristic segmentations only considered sheet switches as potential segment boundaries, the numbers in both cases are high, as expected.

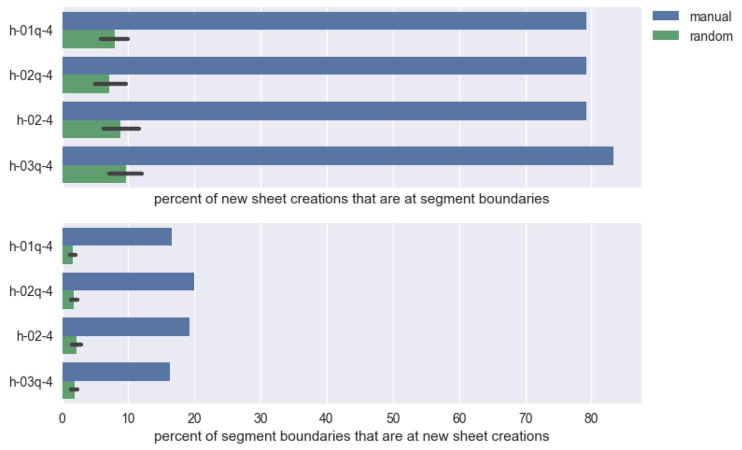


Figure 5.46: New sheet creations and segment boundaries: These figures show the percent of sheet switches in the submission that occur at segment boundaries, as well as the percent of segment boundaries that occur at sheet switches. Because the heuristic segmentations only considered sheet switches as potential segment boundaries, the numbers in both cases are high, as expected.

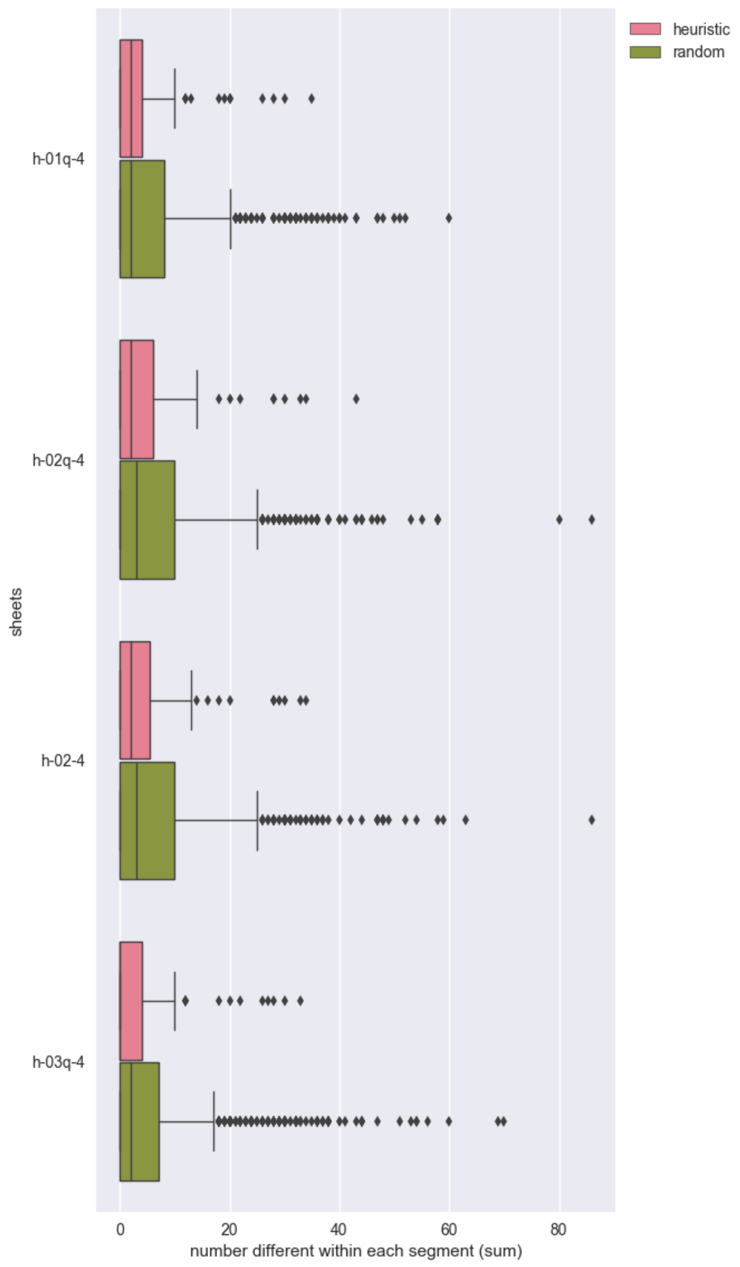


Figure 5.47: Number of sheet switches within each segment: This figure shows the distribution of the number of sheet switches within each segment. Because the heuristic segmentations based segment boundaries on sheet switches, there are much fewer sheet switches within heuristic segments, as expected.

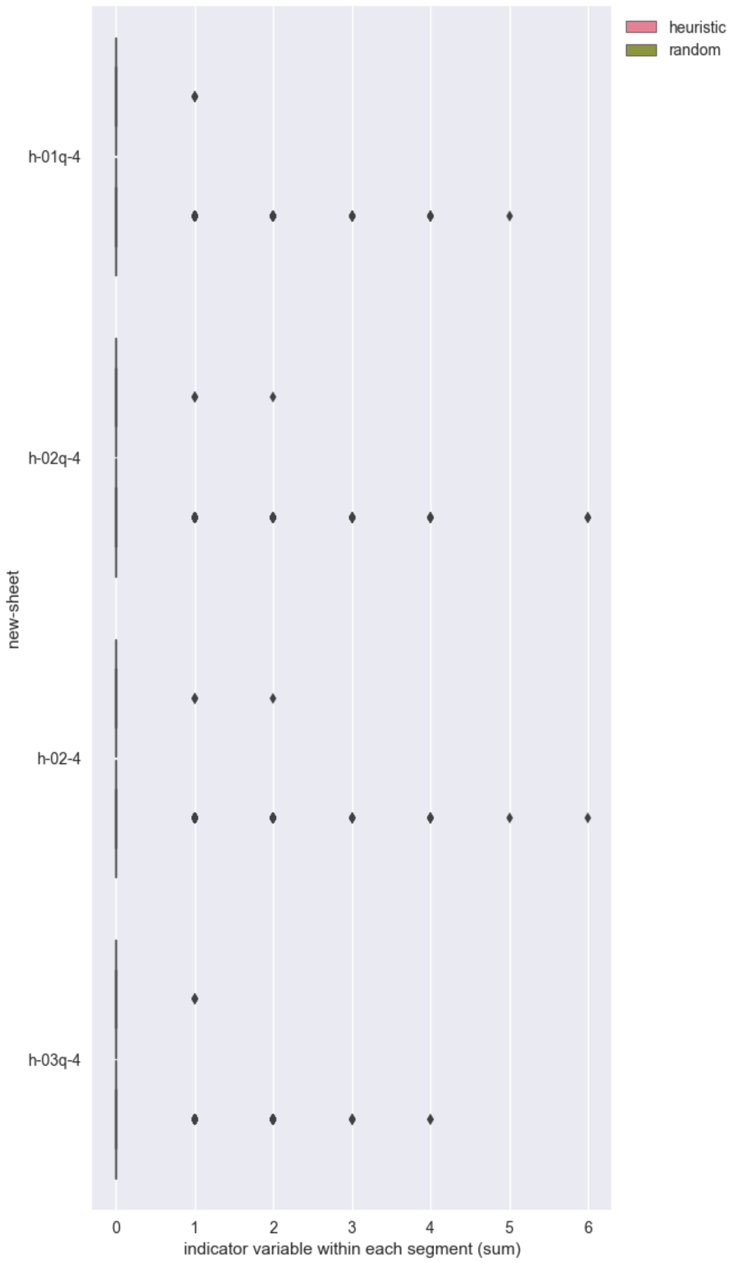


Figure 5.48: Number of new sheets within each segment: This figure shows the distribution of the number of new sheets created in each segment. As we saw for manual segmentations, here again new sheet creations are rare enough that the median number of new sheet creations for all segmentations is zero.

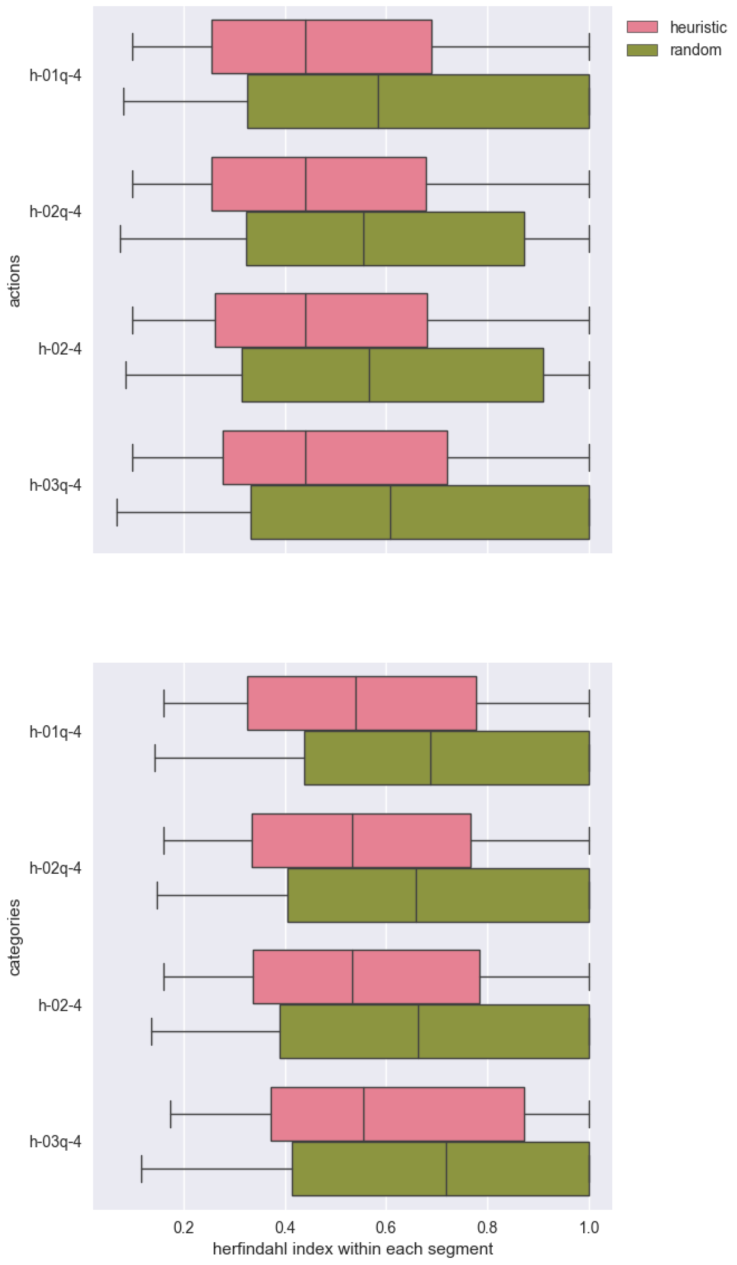


Figure 5.49: Concentration of the distribution of actions and action categories within each segment: This figure shows the distribution of the Herfindahl index of the action and category sets, where a higher number means the set is dominated by fewer actions or categories, respectively. Like the manual segmentations, the heuristics have lower Herfindahl index than the random segmentations. This is not surprising because the heuristics do not take action or category into account.

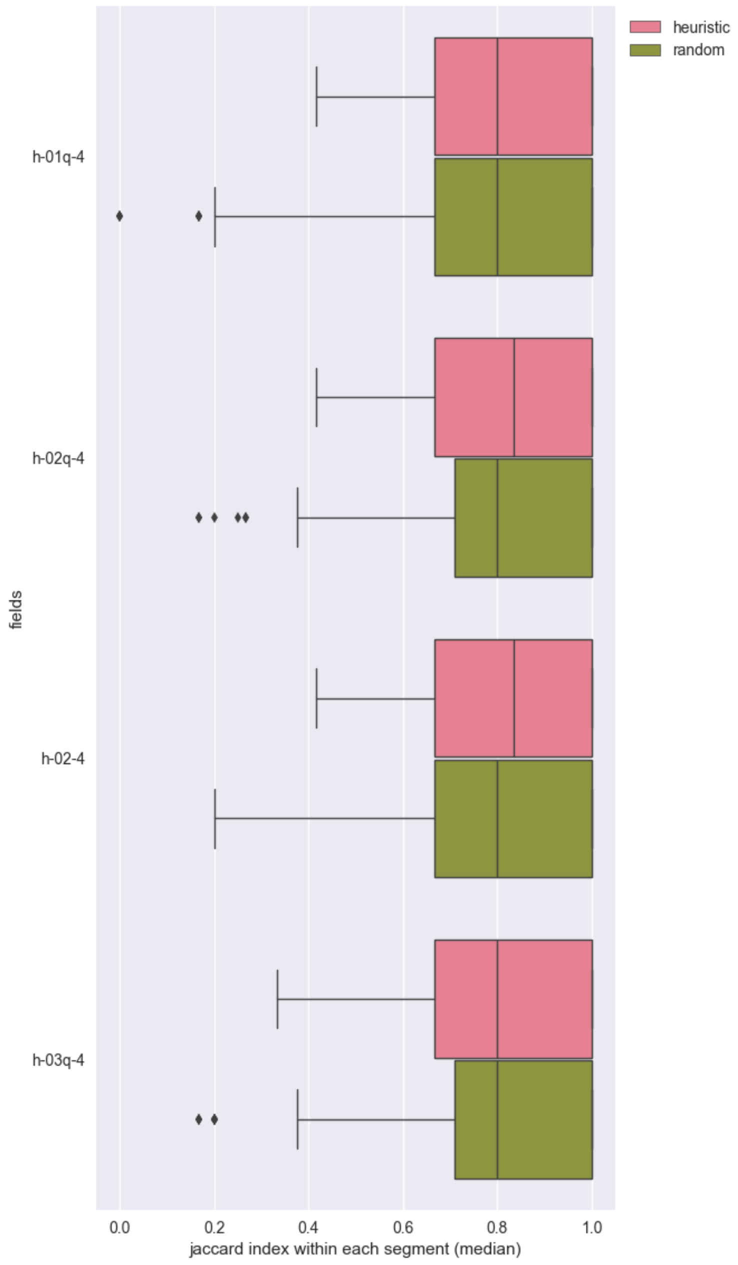


Figure 5.50: Similarity of the fields sets within each segment: This figure shows the distribution of the median Jaccard index (a measure of set similarity) of the field sets of each pair of adjacent events in each segment. A higher Jaccard index means greater similarity. Manual segmentations displayed higher similarity in this respect than random segmentations, but for heuristics that is only true for two, whereas the other two are similar to the random segmentations.

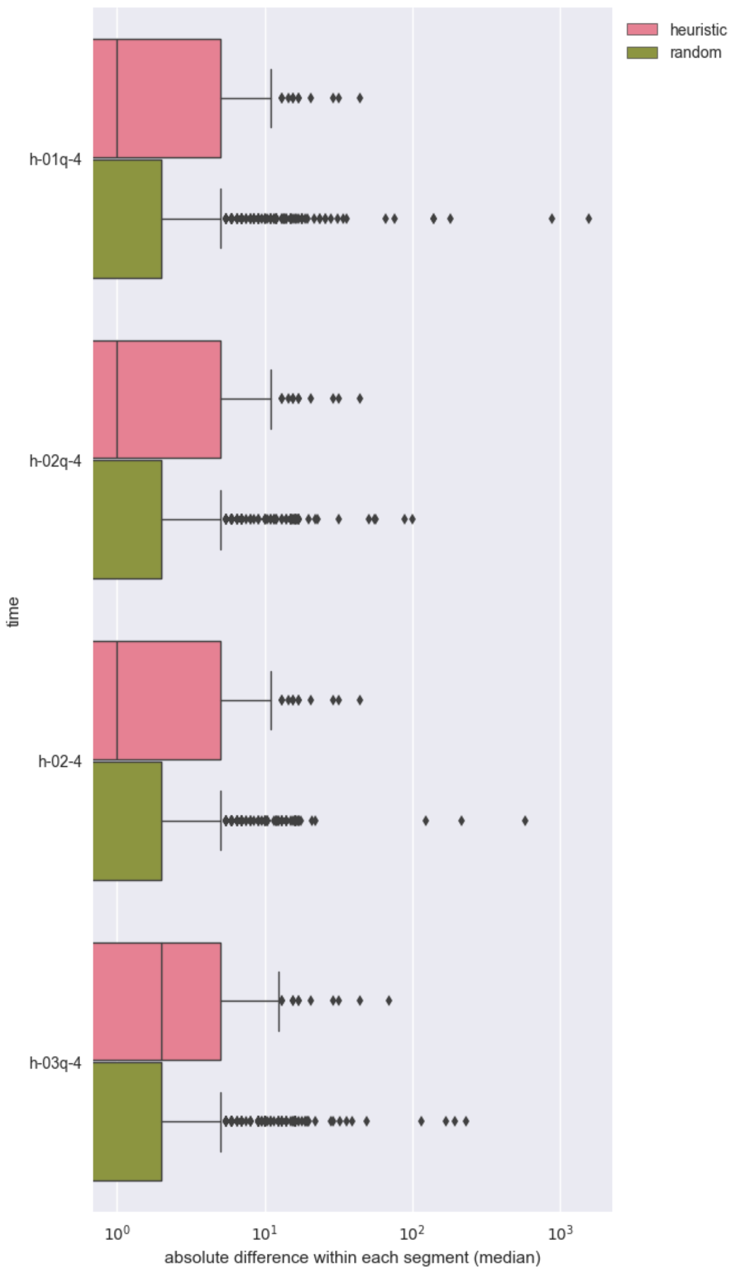


Figure 5.51: Number of seconds elapsed between consecutive events within each segment: This figure shows the distribution of the median number of seconds elapsed between consecutive pairs of events within each segment. Like with the manual segmentations, the time between events within the segments of the heuristic segmentations is greater than for those of the random segmentations, for some reason we don't yet understand.

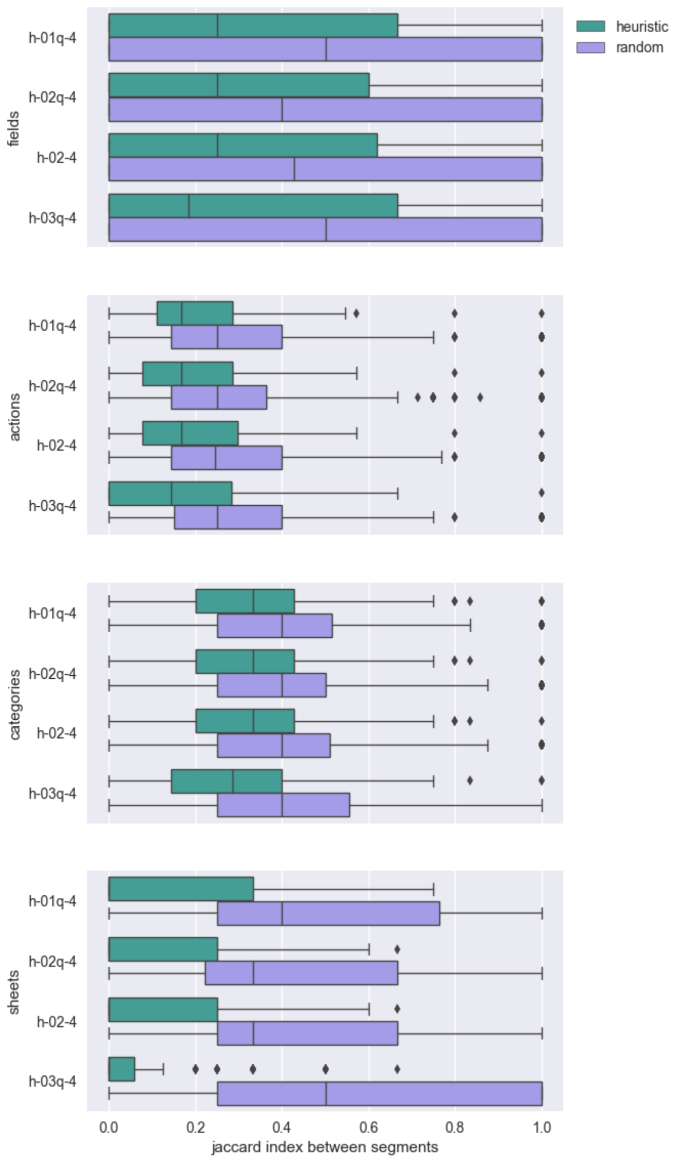


Figure 5.52: Similarity between property sets in pairs of adjacent segments: This shows the distribution of the median Jaccard index between pairs of adjacent segments' sets of actions, categories, fields, or sheets. We expected the Jaccard index, and thus the similarity, between segments in manual segmentations to be lower than for random segmentations and that was true. It is also true for heuristic segmentations, even though the heuristics only explicitly considered sheet switches.

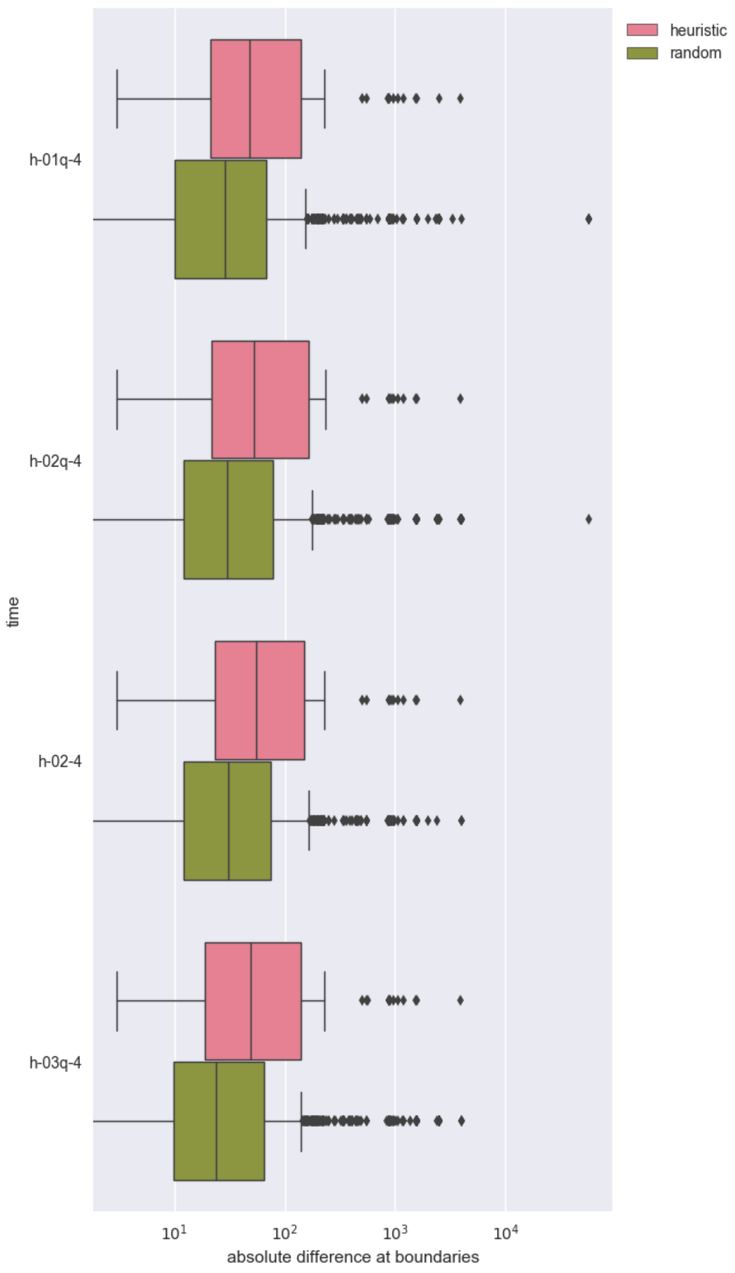


Figure 5.53: Number of seconds elapsed between consecutive segments: This figure shows the distribution of the number of seconds between adjacent pairs of segments in each segmentation. As with the manual segmentations, the gap between segments in the heuristic segmentations is bigger than those for random segmentations.

event ID	**sheet ID**	**run len**	**Heuristic-1-4**	**Heuristic-2-4**	**Heuristic-3-4**
0	1	5	1	1	1
1	1	-	1	1	1
2	1	-	1	1	1
3	1	-	1	1	1
4	1	-	1	1	1
5	2	2	1	1	1
6	2	-	1	1	1
7	1	5	2	1	1
8	1	-	2	1	1
9	1	-	2	1	1
10	1	-	2	1	1
11	1	-	2	1	1
12	3	2	2	1	2
13	3	-	2	1	2
14	4	2	2	1	2
15	4	-	2	1	2
16	5	3	2	1	2
17	5	-	2	1	2
18	5	-	2	1	2
19	6	6	3	2	3
20	6	-	3	2	3
21	6	-	3	2	3
22	6	-	3	2	3
23	6	-	3	2	3
24	6	-	3	2	3

Table 5.9: Example segmentations for each class of heuristic, assuming all events are qualified, using an m cut-off of 4. The heuristic columns list the segment ID assigned by that heuristic.

a higher median than their corresponding random segmentations, whereas the other two had the same median (Figure 5.50). With respect to intra-segment sheet switches, they all meet expectations, which is perhaps to be expected because the heuristic is based on sheet switches (Figure 5.47). The heuristics consistently meet expectations with respect to the boundary (Figures 5.45 - 5.46) and inter-segment properties (Figures 5.52 - 5.53).

5.3.4.3 Comparison with Manual Segmentations

Before comparing the agreement of the heuristic segmentations with the manual segmentations and adjustments, note that maximum agreement we can expect is bounded by the number of segment boundaries in the manual segmentations and adjustments that occur at sheet switches. This ranges from about 62% to 95%. Moreover, the proportion of segment boundaries that are in one manual segmentation but not another ranges to just over 80%.

Given this level of disagreement among some of the manual segmentations, we should not expect the heuristic segmentations to attain good agreement with all of the segmentations.

Figures 5.54 - 5.56 show the agreement between the heuristics and the original manual segmentations, and the first and second adjustments, respectively.

Figure 5.54 shows that when measured by WindowDiff, the agreement between the heuristics and the original segmentations is not as close as the agreement among the original segmentations. The heuristics are still more in agreement with the original segmentations than they are to the randomly-generated segmentations that are of equal length to the heuristics. But the heuristics are only very slightly more in agreement with some of the manual segmentations than the manual segmentations are to their corresponding randomly-generated segmentations; i.e., A (random) - C (random).

These observations are not all true for all metrics. For the Jaccard index, and the number and percent of segment matches, the heuristics are more in agreement with the manual segmentations than either of them are with the randomly-generated segmentations. For the time distance, the heuristics are more in agreement with the manual segmentations than the random segmentations only when considering comparing the distance from the manual segments to the heuristic segments but not vice-versa.

The difference between metrics is in part because the WindowDiff result does not improve when there is a difference in total number of segments between the two comparison segmentations. For the other metrics, however, a heuristic segmentation with more segments should tend to get a better result when compared with the manual segmentation, because having more segments means they are more likely to overlap with the manual segmentations' segments.

In all metrics, the heuristics are more similar to one another than they are to the manual segmentations.

The agreement between the heuristics and the adjustments to the original segmentations follow the same general patterns described above, as shown in Figure 5.55 and Figure 5.56. In terms of WindowDiff, the first adjustment shows only slightly better agreement and the second adjustment shows even more. In terms of the other metrics, however, the adjustments show substantially more agreement than the original segmentations.

Of the instances where all of the heuristics created a segment but none of the manual segmenters do, 75% are dashboarding segments. Transitions in dashboarding activity are easy to overlook, because the augmented Tableau logs did not include much information about dashboarding actions beyond the sheet name. Moreover, segmenters seemed to group dashboarding activity all together, even if it occurred on different sheets. There is only one instance in which all the manual segmenters made a segment and none of the heuristics did, and that was when a new sheet was created in the middle of a dashboarding sequence, but no other events were done on that sheet at that time. The two segmenters on our team were probably mislead by the annotation of the participant, who declared that event the start of new task, which they probably did because they were using the LogViewer tool which did not show any of the dashboarding events around the marked event. If the LogViewer tool

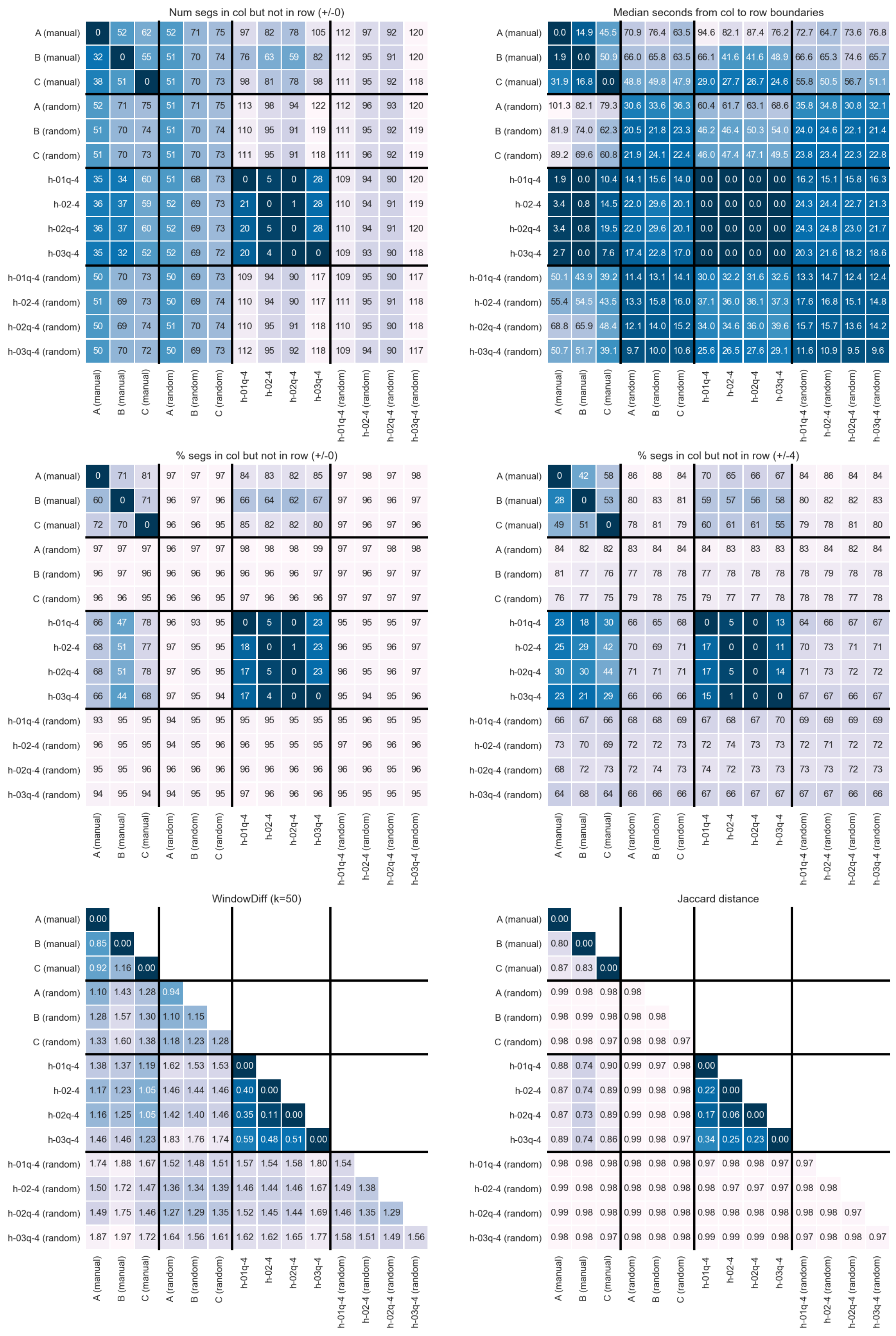


Figure 5.54: Agreement between manual and heuristic segmentations.

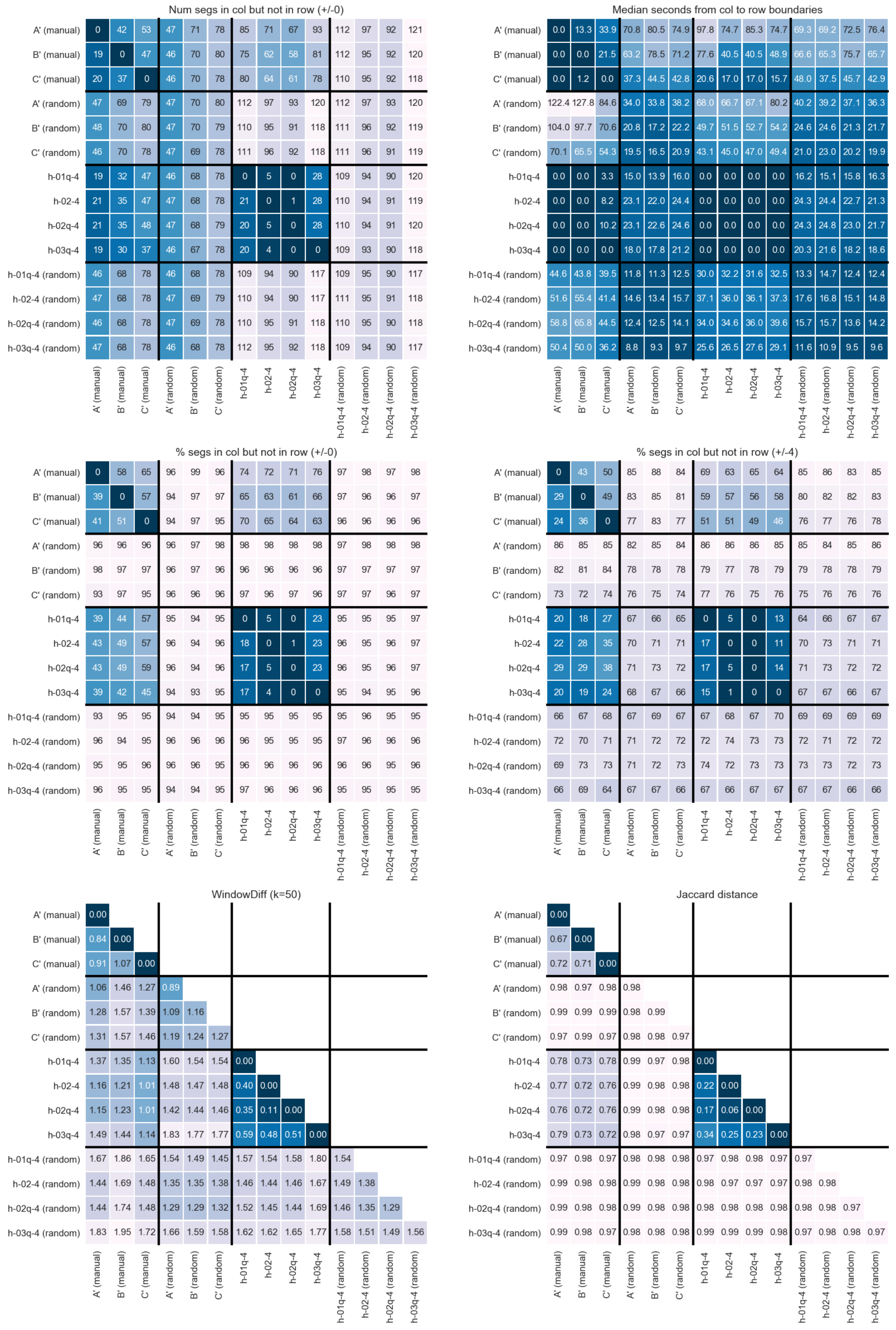


Figure 5.55: Agreement between Adjustment 1 and heuristic segmentations.

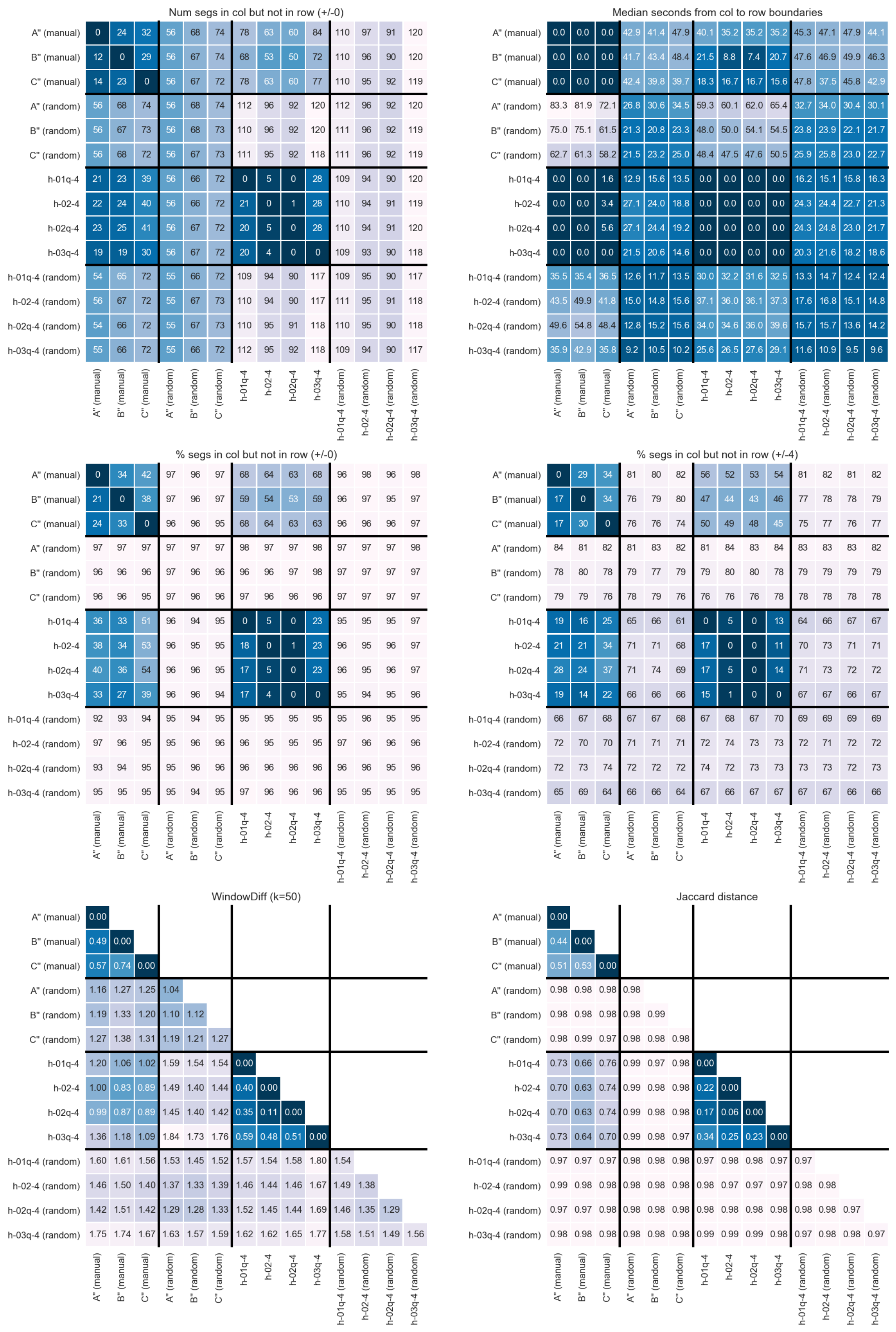


Figure 5.56: Agreement between Adjustment 2 and heuristic segmentations.

had shown more context, it is not likely that that event would have been chosen as the start of a new segment, occurring as it does in the middle of a long bout of dashboarding activity.

5.3.5 Submission Descriptions

The purpose of segmenting event logs into subsequences of consecutive events is to facilitate the summary of the activities the logs record. A collection of similar events can be described more succinctly than a collection of dissimilar events, because the same descriptors apply to more events, allowing fewer descriptors to be used. The next step after segmenting the event stream is to generate these summaries of the segments. In this section we summarize the segments in several ways. The first is a barchart showing the frequency of activity types by segment. Recall that we use the Herfindahl index to describe how how "diverse" or conversely, how "concentrated" the segment is with respect to different types of activities. The more different categories of activities there are in a segment, the more diverse it is. Ideally segments would be formed in such a way as to be dominated by one category of activity, because we are trying to group similar activity to create the segments, which we can then describe in terms of the dominant type of activity. We use a Herfindahl index cutoff of .25, which is considered by Antitrust Division of the Department of Justice to represent the level at which markets are highly concentrated – to determine whether or not a segment is dominated by an activity category [66]. If the Herfindahl index is below that, we say the segment is a mix of activity categories.

We counted by:

- The number of segments
- The number of seconds elapsed in segments – for this we omitted idle time periods, which we counted defined as 30 minutes or more of time elapsed with no activity

The second is a transition frequency diagram showing the frequency of transitions from one type of activity to the next. The third is a printout describing each segment according to:

- The categories of activity that dominate the segment
- The sheets that were worked on in the segment
- The data that was worked on in the segment
- The length of time spent in the segment

The first two methods are less specific, but can be generalized across submissions that used different datasets and different sheets In this section we first summarize Submission 16, which has been the focus of our segmentation efforts thus far, and we compare it to the participants' description to see if it is a reasonable summary. We then summarize all of the segmentations to assess whether there are any common patterns of activity.

5.3.5.1 Summarizing Individual Submissions

Based on the results of the previous subsection, we elected to segment Submission 16 using **Heuristic-2-4**. This heuristic ignores switch-backs – it segments at every change in sheet

ID, unless the change does not persist for four or more events i.e., the sheet ID switches back to the previous ID in less than four events. We chose this heuristic because it yielded a number of segments closer to that which the human segmenters created, and because it yielded some of the top results for the metrics displayed in the results in Section 5.3.4.

We provide a rough sketch of the accuracy of submission summaries. Using the **Heuristic-2-4** segmentation, we generated a segmentation and then a summary of Submission 16 that lists the top action categories, the fields examined, and the sheets acted upon within each segment. To this could also be added the last screenshot of each sheet acted upon the segment. This would be useful for sharing the summary with end users, though we did not use them in our evaluation.

Informally, we compared this summary information in the print out to the annotations provided by the participant. This is not a formal method of evaluating the summaries, because the participant's annotations are not in all cases a thorough description of the goals and tasks at hand. Moreover, this was a very cursory evaluation only meant as a rough estimate. Nonetheless, we judged that about 45% of the time, these summaries very closely corresponded to the annotation given by the participant, because the field names and sheet names were very descriptive of the data and analysis done. About 50% of the time, the segment was mainly dashboard editing. Most of these events were not included by the annotation tool, and as such were mostly not annotated. However, the event activities and the name of the sheets being edited are reasonably descriptive of the activity taking place. The remaining 5% of the time, the segment was such a mix of activities that there was not a clear correspondence between the summaries and the participant's annotations.

The activity category counts in Figure 5.57 and Figure 5.58 show that, by segment count, most segments are dashboarding segments, followed by editing segments, followed by sheet operation segments (e.g., renaming), followed by segments that are too much of a mix of activities to describe using any one category.

The following patterns of activity stand out in the transition frequency diagram of Submission 16 activity in Figure 5.59. The user begins by editing their data. The user then alternates between analysis (**edit vis**) and presentation (**edit dashboard**). Undo or redo happens in the context of analysis. Much of what might be thought of as exploratory behavior (**pan, zoom, or highlight**, **sort**, **filter**, **annotate vis**) happens after the user begins to work on the presentation of the data, though this activity would also be useful in organizing the visualizations for presentation. Annotation and adjusting aesthetics happens in the context of examining the visualization via pan, zoom, or highlight. Sheet renaming and organizing sheets (**do sheet op**) happens in the context of presentation (edit dashboard). Adjusting aesthetics happens in the context of presentation. The user finishes on presentation.

These transition frequency diagrams could be made more useful in the context of a log exploration tool by, for example, linking nodes to the log contents, so that when a node is selected, the relevant areas of the event log are highlighted. Also, there is much other summary information that could be computed from the augmented logs. For example, functionality that would be useful to an evaluator who was attempting to understand the

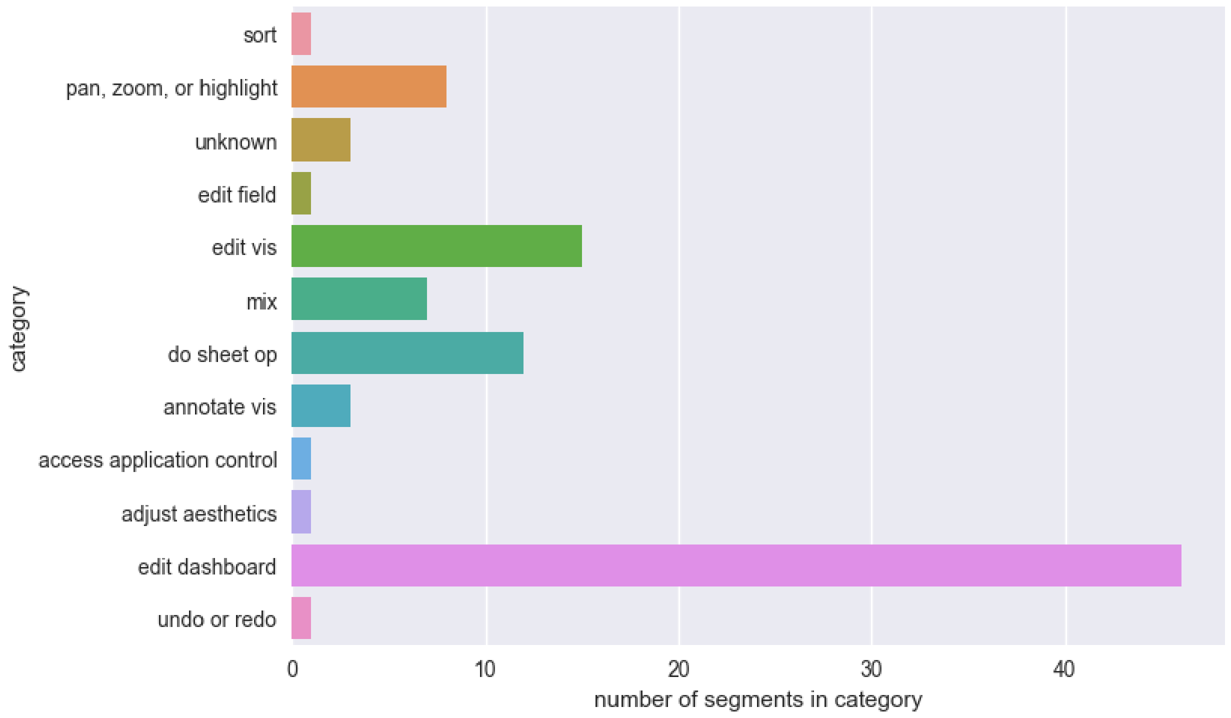


Figure 5.57: Number of segments in each category: Most are **edit dashboard** segments, followed by **edit vis** segments, then **sheet op** segments, then mixed segments, which are segments in which no one activity category dominates.

activity taking place in the session might include:

- Graph of fields used (e.g., number of accesses or times they appear on a sheet)
- Graph of time spent on each sheet
- The ability to interactively segment the event log
- The ability to search for specific patterns

We do not focus on these here because our goal is not to analyze a particular submission.

5.3.5.2 Aggregate Properties of All Submissions

When it comes to frequency of activity types, all of the submissions in general follow the same pattern as Submission 16 in particular. This can be seen in Figure 5.60 and Figure 5.58. These figures show the averages of the number of segments and amount of time per category, respectively, averaged over all 24 submissions.

The following patterns stand out in the transition frequency diagram of all submissions' activities shown in Figure 5.62. Data source and field editing again tend to be at the beginning, like with Submission 16. After that, users tend to move onto analysis (**edit vis**). Presentation (**edit dashboard**) most often follows analysis, but unlike with Submission 16,

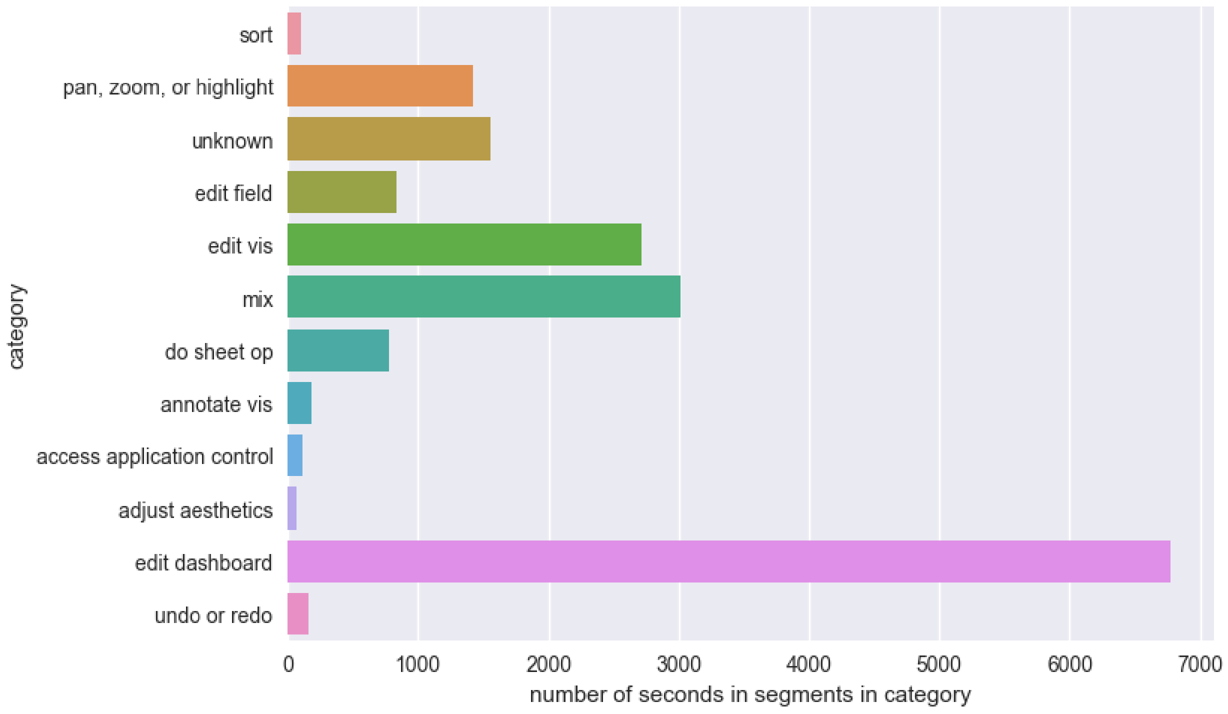


Figure 5.58: Number of seconds elapsed in segments in each category: The same pattern holds as when counting by events, except that segments that are a mix of action categories moves up to the second place in terms of frequency.

analysis and presentation are not alternated between for most submissions, suggesting that for many submissions, presentation comes strictly after analysis. Categories like **undo or redo** and **adjust aesthetics** and others that lack labeled edges happen in enough different contexts (i.e., before and after a wide range of other segment types) that no edge weight exceeds the .25 threshold for display. Many segment types are followed by presentation, but this could be just because presentation is the most common segment type. It could also be, as we suggest above, that what are often thought of as exploratory actions like pan, zoom, or highlight, and filter, are actually being more often used to tweak a presentation than to explore during analysis. This could be because users are creating visualizations as part of their analysis but not actually analyzing the visualizations and drawing conclusions until they begin working on the presentation. This is something that could be investigated in future studies.

5.4 Comparison to Related Work

In this section we compare our approach and findings to related work.

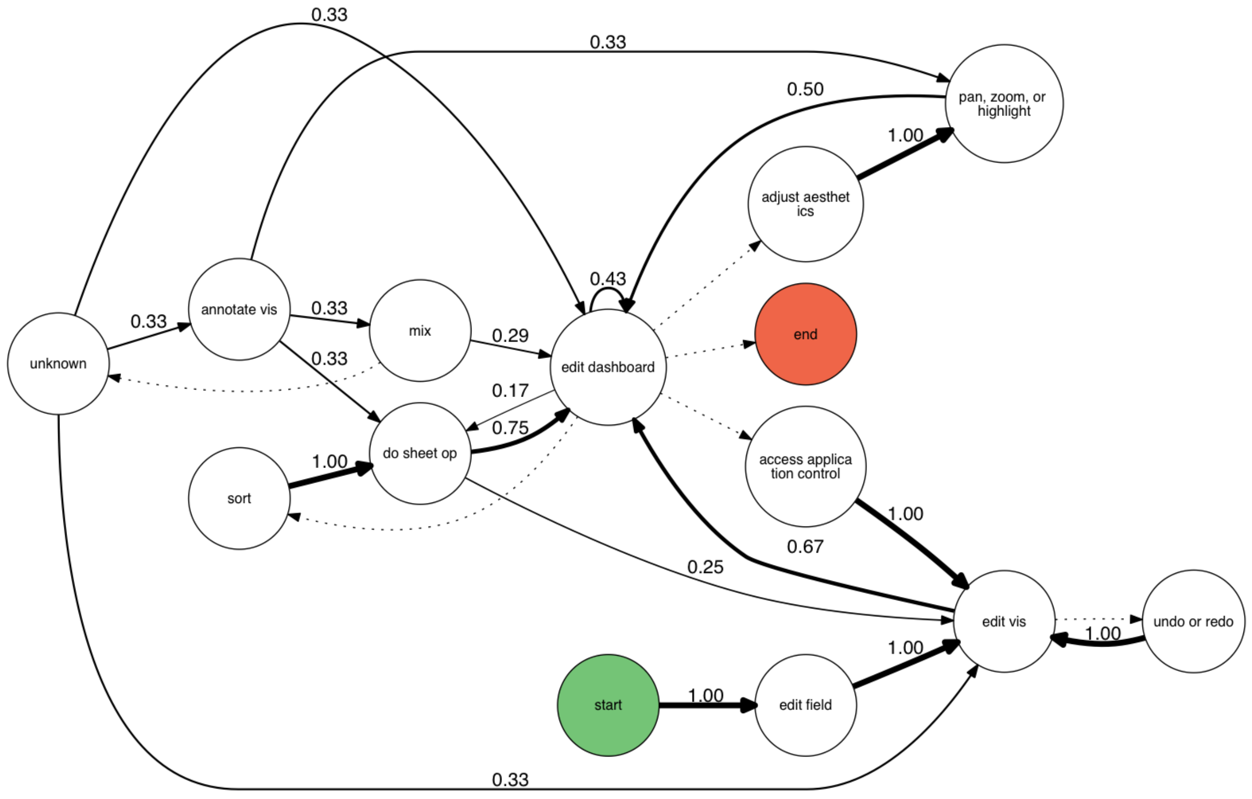


Figure 5.59: Transition frequency diagram showing the frequency of transitions between activity types: In this graph, the nodes represent segments of each given type, and an edge from a node A to a node B represents the percent of time a segment of type B followed a segment of type A. Transitions with weights less than .15 are not shown (unless there is no other incoming or outgoing edge for a node, in which case it is shown as an unlabeled dotted line).

One of our approaches in this chapter was to visualize the event stream to highlight similar activity, for the purpose of segmentation and summarization. Guo, et al., also created a visualization of these higher-level actions that resemble our visualization, which suggests this is a useful format for viewing log data. One thing they did that we did not do, however, was to examine at frequently recurring patterns (as well as the transition frequency between high-level actions, which we also examined for our datasets). We examined frequently recurring patterns in our Splunk case study in Chapter 4, using sequences of high-level transformation categories. However, when we tried to do so for our Tableau data, we did not find any interesting pattern that occurred frequently enough from which to draw conclusions This is possibly in part because that data was at a lower-level of abstraction with higher-frequency, fine-grained action events, and also in part because participants in this case study all examined different datasets, though this data context is generally not present in the action event

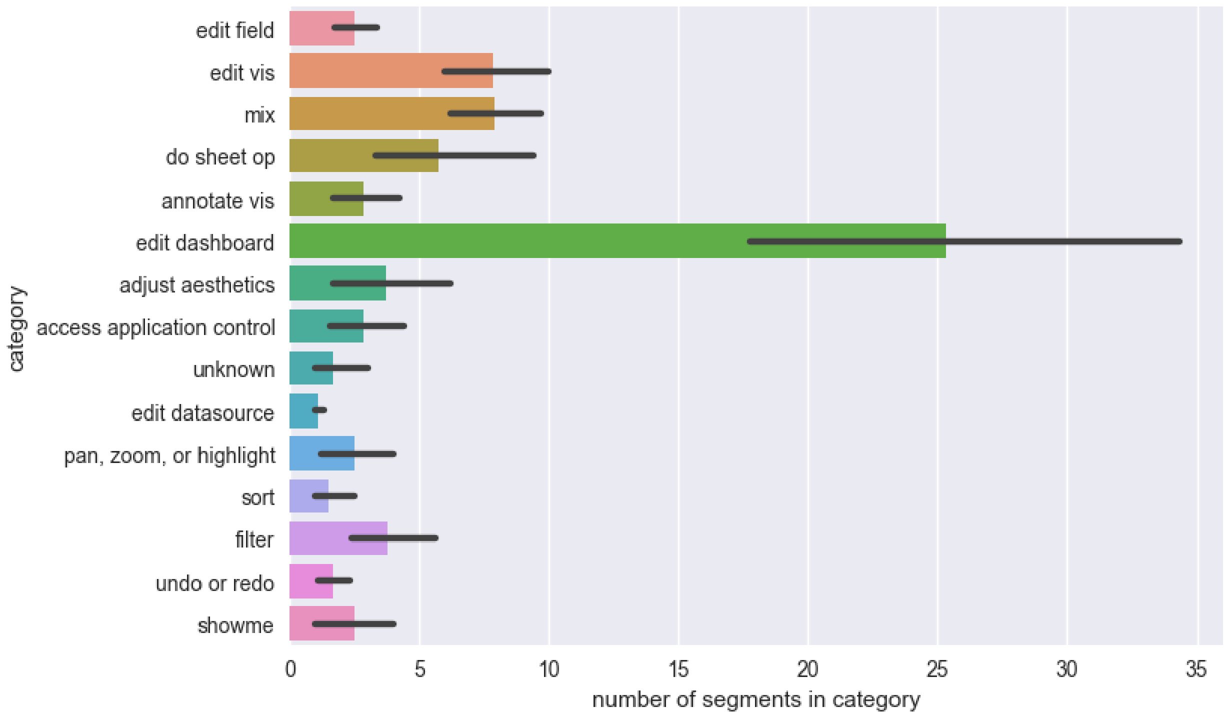


Figure 5.60: Number of segments in each category, averaged over all 24 submissions: Most are **edit dashboard** segments, followed by **edit vis** segments, then **sheet op** segments, then mixed segments, which are segments in which no one activity category dominates. This is true even considering the range across submissions.

information (challenges described in Sections 3.3, 3.5, 3.6). Thus, these datasets would have to be both somehow incorporated into the action events to provide context to them, and also abstracted to generalize across participants. Thus, in our experience, high-level patterns are hard to interpret because of their generality, but using low-level, fine-grained actions instead did not work because no pattern recurred frequently enough at that level from which to draw conclusions.

The purpose of our visualization tool was to facilitate the segmentation of the event stream into consecutive subsequences of similar events. In other work, this segmenting has been referred to as "chunking" [89, 61]. Kurlander used heuristic chunking rules to concisely display workflow histories in a graphical editing tool, which displays steps as a linear sequence of panels that show how the graphical objects were manipulated with each action [89]. In his system, the granularity of the chunks – how many actions are summarized in one panel – can be adjusted by the user. While Kurlander does not explain his heuristics in full detail, he says that sequences of similar actions, like object translations, or successive add operations to draw a line, are coalesced into one panel. This is similar to some of the heuristics used by Heer, et al., who collected data from Tableau similar to ours, but like Kurlander, for the

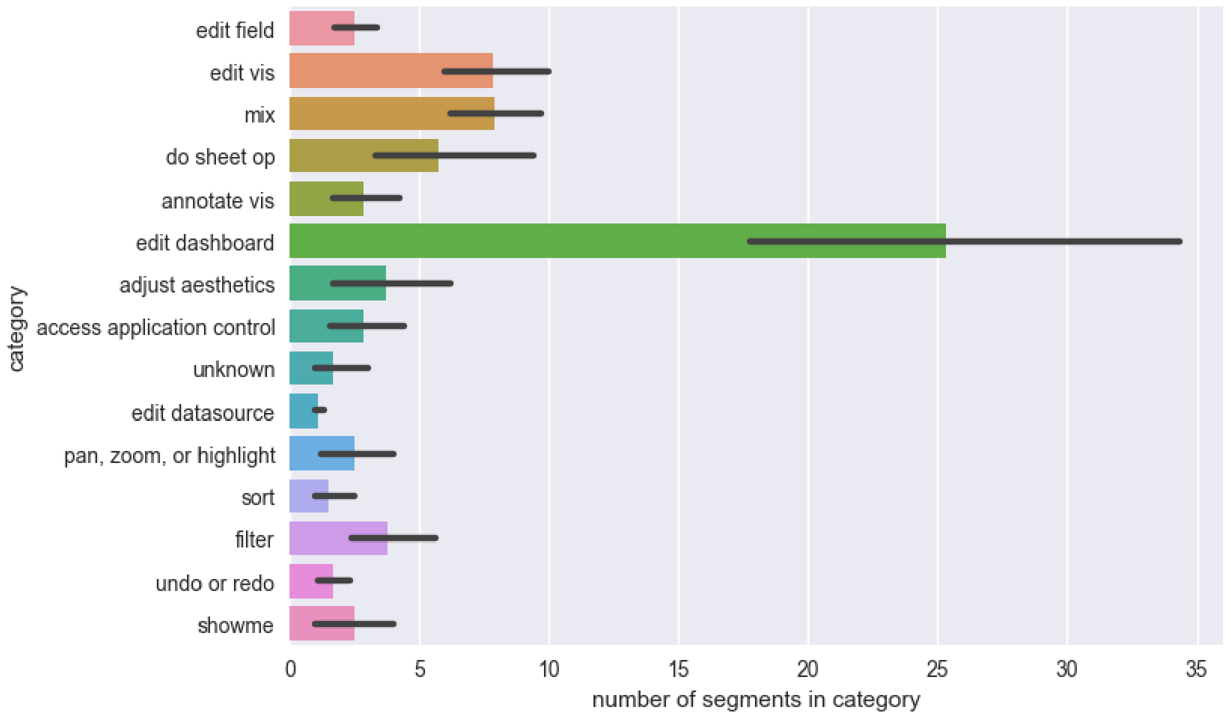


Figure 5.61: Number of seconds elapsed in segments in each category, averaged over all 24 submissions: The same pattern holds as when counting by events, except that segments that are a mix of action categories moves up to the second place in terms of frequency. This is true even considering the range across submissions.

purpose of displaying concise navigable usage histories to users within the application [61]. Their heuristics are applied in an online fashion; that is, when a new action occurs, the rules evaluate whether it should be chunked with the previous state. They have three chunking rules, along with some exceptions that say when an action should not be chunked despite meeting one of the conditions of the other heuristics. Their rules are described as follows: "we include a rule that chunks history items if the most recent state was the result of a formatting operation. Similarly, a quick succession of sort or filter actions (less than 30 sec. apart) likely indicate a multi-step configuration of the view and are chunked together. A rapid series shelf actions to build up (or take down) a view are also common, and so we chunk them when separated by less than five seconds. We also support exception cases: large time durations possibly indicating a break between sessions prevents any chunking, as does bookmarking or annotating a state." Rather than allowing the level of granularity be adjustable, as Kurlander did, Heer, et al., allow the user to expand chunked states to see the full detailed sequence. Santos and Badre took a different approach to chunking in which they detect boundaries based on pauses in user activity [134]. However, as the data in Sections 5.3.3 suggests, such pauses did not seem to be good indicators of boundaries as

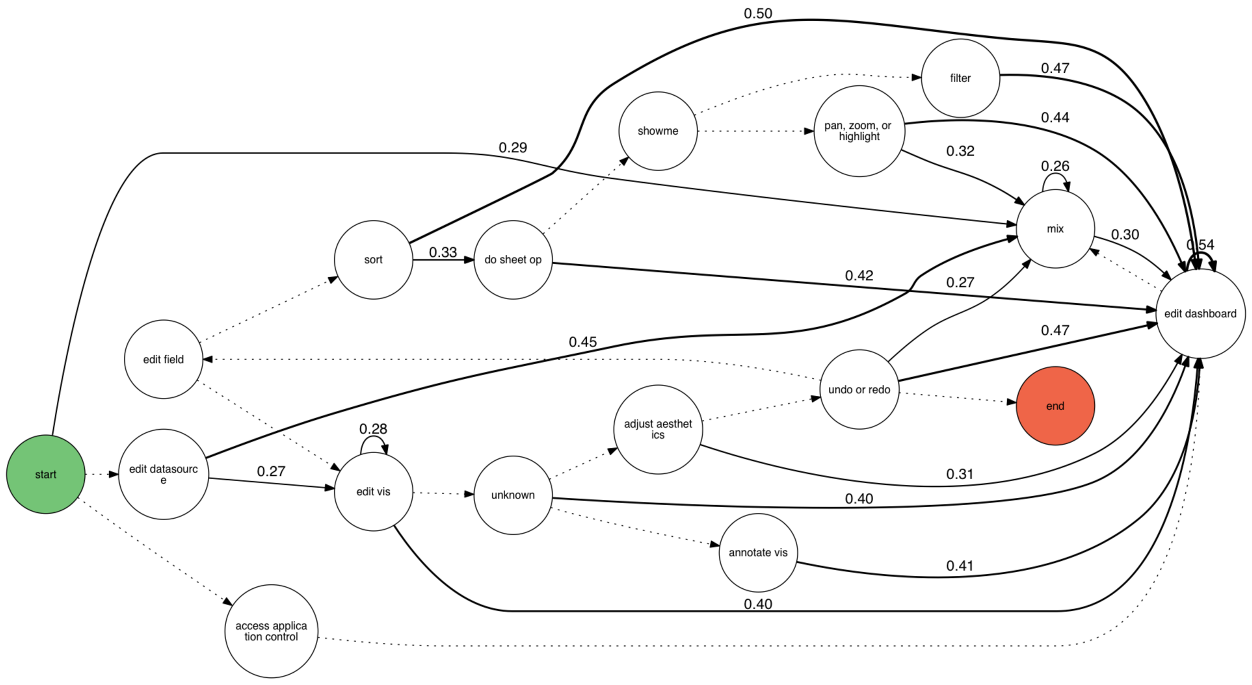


Figure 5.62: Transition frequency diagram showing the frequency of transitions between activity types, computed over all 24 submissions: In this graph, the nodes represent segments of each given type, and an edge from a node A to a node B represents the percent of time a segment of type B followed a segment of type A in all of the submissions. Transitions with weights less than .25 are not shown (unless there is no other incoming or outgoing edge for a node, in which case it is shown as an unlabeled dotted line).

judged manually by our collaborators in our case study.

In terms of findings, the most comparable work is also by Heer, et al., described in the previous paragraph [61]. Though analysis of this data was not the main focus of their work, they did describe three small findings:

- Undo was 12.5 times as common as redo, supporting the idea that undo was used as delete.
- Formatting actions accounted for approximately one quarter of all actions, and that they usually occurred in succession, as did resizing actions, thus their chunking rules were designed to coalesce these.
- The automatic visualization generation feature of Tableau, "Show Me," was commonly used by all users and rarely did they adjust its output, meaning its suggestions were good.
- Their chunking heuristics remove or chunk 60% of the states.

Our dataset compares as follows:

- Undo was even more prominent, being 29.3 times as common as redo.

- Formatting actions represented only 3% of our dataset, and 51% of these were followed by additional formatting commands, so it was much less prominent; however, this could be due to differences in our categorization schemes.
- All participants in our dataset used the "ShowMe" command, but it accounted for only 1% of commands, and we were unable to evaluate how often users kept the results.
- While we did not implement Heer, et al.'s chunking rules, the commands they chunk account for only 12% of our dataset, and so at best only 12% could be removed in this fashion.

The discrepancies between our findings is likely due to the fact that neither of our datasets are representative of population-wide Tableau usage and instead reflect certain subsets of uses. The differences between our approaches to summarizing the data is reflective of our differing goals – theirs, to summarize workflow history for users in an online fashion, and ours to identify high-level tasks from the event stream post-facto.

5.5 Conclusion

In this chapter, we presented a case study of data logged from Tableau. We described our method for collecting detailed and annotated event logs from Tableau. This approach allowed us to collect data about the entire analysis cycle, from the user's goals and data at each step, to the transformation and visualization of that data, and finally to the user's interpretation of the resulting visualization. This is an innovation over what is typically collected in log analysis studies and provides a substantially more complete picture of user analysis activity.

In the analysis section, we described a visualization tool we developed for inspecting the event stream and segmenting it into like subsequences. This tool features a compact display of the event stream – specifically the actions and fields, color-coded by action type and data type, as well as mouseover to reveal additional event and segment details. We then used this tool to create segments, which we used to summarize and describe the submissions. We first asked collaborators to segment the event stream of one submission by hand. We then analyzed these segmentations both in terms of the properties of the events in the segments and how the segments compared to one another, as well as in terms of how well the segmentations agreed. We used randomly-generated segmentations as a baseline to compare the properties and agreement with.

In terms of properties, we expected that people would create segments such that events within the segment were more similar than events outside it. This means we'd expect the similarity within manual segmentations' segments to be higher than randomly-generated segmentations' segments (i.e., high intra-segment similarity). It also means the differences between adjacent segments in manual segmentations should be greater than the differences between adjacent segments in randomly-generated segmentations (i.e., low inter-segment similarity). The latter was true (low inter-segment similarity) for the properties we investigated, but not the former (high intra-segment similarity) in most cases. Segments from manual segmentations tended to occur at sheet switches far more than randomly-generated segmentations' segments.

In terms of agreement, although we provided guidelines for segmentation to attempt to encourage our segmenters to follow a similar approach, we found that there was not a very high degree of agreement between human segmenters, even when attempts at reconciling the segmentations were made. Even after adjustments and allowing for some flexibility in exact location of event boundaries, the best case overlap between segment boundaries is only 83%, and the worst case is a mere 19%. This could be because that segmentation involves a lot of subjectivity, and that likely different segmentations would be better for different uses. However, it also could be because we only had a few segmentation examples, and did not iterate on the segmentation instructions. Thus, we cannot draw strong conclusions from this.

We then developed a simple heuristic based upon the properties which were most consistent about the segmentations' boundaries – the fact that they often occurred at sheet switches. We analyzed the heuristic segmentation in a similar way to the manual segmentations. The heuristics generated about 2-6 times as many segments as the manual segmentations, suggesting that a better heuristic would incorporate some way to iterate over these first pass segmentations and merge segments based on similarity. The heuristic segmentations behaved similar to the manual segmentations in terms of their properties. The heuristic segmentations achieved reasonable agreement with the manual segmentations considering how little agreement there was with the manual segmentations to begin with. The best case agreement that was achieved, over all manual segmentation and heuristic pairs, was 88% of segments in one segmentation being in another, whereas the worst case was 12%.

Lastly, we used this heuristic to segment and summarize the focal submission and all the other submissions additionally. We can't generalize too much about how people analyze data from these, because each submission involved a different dataset. Nonetheless, we found that the most common activity is presentation (**edit dashboard**) both by time and by segment count, and this is true across all submissions. After that, segments that are a mix of action categories and analysis segments (**edit vis**) are the most common. We found that the submissions tend to follow the workflows proposed in the literature: first data munging, then analysis, then presentation, though there is some iterating back and forth. This reflects the fact that the data we collected was from students following particular instructions in an assignment that directed them to follow this workflow, and so these answers are not representative of other use cases. We also saw that activities like pan, zoom, and highlight, and filter, tend to happen more in the presentation rather than the analysis phase. A study that controlled the dataset analyzed might be able to learn more about human analysis patterns. Also, the segmentation facilitates many other more specific analyses of specific submissions, though that is not of interest here.

This case study concludes our analysis of interaction data from visual analytics systems. In the next and final chapter, we summarize the contributions made in this dissertation as well as discuss avenues for future work.

Chapter 6

Conclusion

In this chapter, we summarize the contributions and broader impacts of our work, as well as discuss potential future avenues of research.

6.1 Contributions

The goal of this research was to develop some analysis approaches for identifying higher-level activity patterns, or tasks, from lower-level user interaction logs from visual analytics systems. As part of our work towards this goal, we identified seven key challenges in working with lower-level user interaction logs (Chapter 3), to help researchers and analysts plan their approaches to working with such data. We achieved our goal via two separate case studies, with data logged from two different types of visual analytics systems. These case studies not only demonstrated our approaches, but also provided information about how these particular tools are used in the real world. In the first case study, we collected data from Splunk (Chapter 4), a software tool for the analysis of machine-generated data, such as logs. In the second case study, we collected data from Tableau (Chapter 5), a software application for visualizing relational data for analysis. The data we collected was logged during the use of these visual analysis systems and contained information about what actions users took to analyze their data. In our Splunk case study, the data we collected were Splunk customer queries, which are pipelines of data transformation operators. In our Tableau case study, we collected event logs and application state from student analysis sessions for an exploratory analysis assignment, augmented with their annotations about their thought process, and reports summarizing their work.

Given the different natures of the datasets collected, we had to take different approaches in each case. Nonetheless, there are important commonalities to both approaches, suggesting a potentially general aspects to methods for identifying higher-level activity patterns, or tasks, from lower-level user interaction logs from visual analytics systems. At a high-level, our approach in both cases is as follows:

- Re-map all lower-level commands or operations into a set of broader categories that reflect the higher-level goal of that activity. (The categories used for both case studies are given in Appendix A and C, respectively.)

- Using this category information, along with additional context, group the lower-level operations into clusters of similar activity.
- Label these groups in terms of the higher-level or more abstract action they accomplish.

In the Splunk case, the lower-level operations were query fragments or stages, and we clustered the fragments that used the most common types of commands using the features listed in Appendix B. In the Tableau case, the lower-level operations were events, and we clustered each participant's events into temporally consecutive subsequences, so that the groups represented a a higher-level task accomplished by set of smaller lower-level steps or events.

One key difference between these two cases is that in the Splunk case, we only considered higher-level descriptions of individual stages of single queries, rather than what higher-level task the stages or queries achieved when considered together in sequence. In contrast, in the Tableau case, we considered what higher-level task subsequences of participants' event stream accomplished. There are two reasons for the difference in approach. The first is that in the Splunk case, we lacked important context information, such as what data was being operated upon and what the outcome of applying the given transformations to that data was. This lack of information about the state of the data makes it difficult to consider how different stages of a pipeline interact together. The second reason is that in the Splunk case, due to missing information from the logs, we were unable to sessionize user queries, so we could not properly identify consecutive query sequences for analysis. Thus, the second case study using Tableau represents in some ways an extension of the first case study using Splunk, since, in light of the challenges identified in the first case study, we were able to collect more complete data, though it is a different type of data from a different analysis system. One unique aspect of our approach in the Splunk case is our use of an automatic clustering algorithm to identify the groups of similar commands. Moreover, in the Splunk case, we provided additional context through a survey about Splunk customer characteristics. One unique aspect of our approach in the Tableau case is that we instead of using a completely algorithmic approach, we developed heuristics inspired by the results of human judgment to identify groups of similar events. We also developed a custom visualization for performing this grouping. In the Tableau case, we also had a richer dataset, including user log annotation and analysis reports, which helped us to evaluate the accuracy of our high-level activity summaries.

In addition to the approaches described in the previous paragraph, our other contributions include the specific findings that resulted from our analyses and which provide empirical descriptions of how these tools are used in real-world scenarios (as opposed to carefully designed studies that control what data users analyze and what questions they ask).

In the Splunk case, for example, we found that a large portion of log analysis activity in Splunk consists of filtering. The next most common type of activity was aggregation. Filtering is also used in conjunction with aggregation to count occurrences of different things, which is an important task in log analysis. Summarizing is a frequently-used technique in data analysis in general, and is used to create some of the more common graph types, such as bar charts and line graphs [155]. This suggests it may be useful to automatically create certain types of summarization to present to the user to save time. After this, the third most common activity was augmenting the data – transformations of the data usually for the purpose of data munging, or reformatting and cleaning data. The prevalence of reformatting as a portion of log analysis activity is likely reflective of the fact that much log data is structured in an inconsistent, ad hoc manner. Taken together, the prevalence of filtering and reformatting activity in Splunk suggest that it may be useful for system developers to collaborate with the end users of such systems to ensure that data useful for the day-to-day management of such systems is collected. In log analysis with Splunk, summarizing with respect to the time dimension is an important use case – of all the fields accessed in all the datasets analyzed by users, time was one of the more common ones. We also found that task complexity tended to be low – filtering, aggregating, and reformatting, versus complex machine-learning pipelines. Lastly, our survey results show that log analysis is not only for IT departments; non-technical users need to extract business insights from logs to drive their decision making as well.

In the Tableau case, we found that the submissions tend to follow the workflows proposed in the literature: first data munging, then analysis, then presentation, though there is some iterating back and forth. This likely reflects the fact that the data we collected was from students following particular instructions in an assignment that directed them to follow this workflow, and so these answers are not necessarily representative of other use cases. We also saw that activities like pan, zoom, and highlight, and filter, tend to happen more in the presentation rather than the analysis phase. Many segment types are followed by presentation, but this could be just because presentation is the most common segment type. It could also be that what are often thought of as exploratory actions like pan, zoom, or highlight, and filter, are actually being more often used to tweak a presentation than to explore during analysis. Lastly, we compared our dataset to findings in related work over similar datasets and found that of only the high undo-to-redo ratio also held for our dataset. However, as our dataset in the Tableau case study is from students working on a class assignment, it is not likely to be representative of broader Tableau usage.

To summarize, in our contributions we:

- Identified seven key challenges in working with low-level user interaction logs to help researchers and analysts plan their approaches to working with such data
- Developed approaches for identifying higher-level activity patterns, or tasks, from lower-level user interaction logs from visual analytics systems, via two real-world case studies
- Provided empirical descriptions of the types of high-level activity users of our two case study systems engage in, summarized above and in the relevant chapters

6.2 Broader Impact

In this section we briefly describe the impact our work has had beyond publication. The findings of our Splunk case study in particular have been used by product managers and others within Splunk to learn more about how their tool is used and inform decisions regarding new functionality and interface improvements. It has also inspired internal projects to improve the quality of their logging. Similarly, we made specific recommendations to Tableau on how to improve their logs, and these were incorporated into recent releases, enabling our case study. Our work has also encouraged similar visual analytics companies to undertake studies of their usage data, and we have also advised them on how to improve their logging infrastructure to enable easier analysis and uncover more useful findings. We hope our work incentivizes more organizations to log more context about their users' behavior so that data-driven improvements can be made to their tools and user interfaces.

6.3 Future Work

In this section we describe potential avenues for future research.

In the case of Splunk, one interesting item of future work would be to use the cluster results to characterize whole query pipelines rather than just query stages. In our current work, we characterized query pipelines just in terms of the higher-level category of their individual stages. The more detailed cluster categories would provide even richer definitions of common pipelines. Also in the future, with some slight corrections to the logging code, we could sessionize queries, and then characterize whole sessions much in the way we did with the Tableau events. If we could also collect more information about the datasets the queries are being used to analyze, this would also enable a richer assessment of the tasks they are performing.

Another item of future work would require some changes to the Splunk query language itself. In the process of analyzing the query data, we encountered difficulties relating to the fact that many commands in the Splunk language are heavily overloaded and can do many different things. For example, `stats` can both aggregate and rename data. When this is the case, we are more likely to have to rely on error-prone data mining techniques like clustering and classification to resolve ambiguities involved in automatically labeling user activities. If the mapping between analysis tasks and analysis representation (i.e., the analysis language) were less muddied, it would alleviate some of the difficulties of analyzing this activity data and pave the way for easier modeling of user behavior in the future.

In the case of Tableau, the visualization tool could be enhanced and deployed in many different contexts, such as for researchers inside Tableau trying to understand their customers' use patterns, as well as for researchers outside of Tableau who are trying to study user analysis behavior. The tool could be amended to include whatever additional information these researchers would find useful, such as time spent on each sheet, or the ability to edit segments in the tool. The tool could incorporate Markov diagrams so that researchers could better understand workflows, and these diagrams could link to the log content so that selecting a node highlighted the corresponding events and details. It could also make it possible to search for specific patterns. The usefulness of the tool could then be tested and enhanced by deploying it in long-terms studies in the wild with analysts who work with logs.

Other future work is to work towards a standard segmentation scheme. This could be used to do further work to analyze strategies via submission summaries, especially if data was collected from many analysts analyzing the same dataset. Or if developing a standard segmentation scheme is not realistic given the subjective nature of summary, different schemes to segment and summarize could be developed depending on user's preferences, or made adjustable via parameters. This summary information could be incorporated into models of user behavior that could in turn be incorporated into intelligent interfaces that adapt to and anticipate user needs and make recommendations.

In addition to enabling the above work specific to our case study systems, the data presented here points to the following broader areas of future work:

- **Need for integrated provenance collection:** Understandably, most data that is logged is done so for the purpose of debugging systems, not building detailed models of user behavior [7]. As discussed in Chapter 3, this means that much of the contextual information that is highly relevant to understanding user behavior is not easily available, and even basic information must be inferred through elaborate or unreliable means [50]. We hope to draw attention to this issue to encourage solutions to this problem.
- **Further analyses of data analysis activity:** In this dissertation, we only presented data analysis activity from two systems. It would be informative to compare this to data analysis activity from other systems, and on other types of data besides log data. Thus, we make our analysis code public so others can more easily adapt and apply our analysis to more data sets and compare results.
- **Opportunities for predictive interfaces:** Thinking forward, detailed data on user behavior can be fed into advanced statistical models that return predictions about user actions. Studies such as the one we present are important for designing such models, including identifying what variables to model and the possible values they can take on. Other important variables to model could include who the user is, where their data came from, and what problems they are trying to solve. These could be used to provide suggestions to the user about what they might like to try, similar to how other recently successful tools operate [77, 129].

6.4 Final Remarks

In conclusion, in this dissertation we have presented two case studies in how to summarize user activity and identify tasks in interaction data logged from software tools for visual analysis. The user interaction data is information that was logged during the use of these visual analysis systems about what actions users took to analyze their data. Our contributions include the identification of seven key challenges that must be dealt with when analyzing data like this to understand user behavior, the development of approaches for identifying higher-level user activity or tasks from this data, and empirical descriptions, derived using these approaches, of how our case study systems are used in the real world. We hope that this work leads to future research on how to understand and concisely describe visual analytics tool usage as revealed by logs of user interactions.

Appendix A

Splunk Transformation Categories

The following table shows how we categorized each Splunk command for our analysis in Chapter 4. We used the following 18 high-level categories:

- Aggregate
- Augment
- Cache
- Filter
- Input
- Join
- Macro
- Meta
- Miscellaneous
- Output
- Project
- Read Metadata
- Rename
- Reorder
- Set
- Transform
- Transpose
- Window

Command	Category	Command	Category	Command	Category	Command	Category
`abstract`	Miscellaneous	`addcoltotals`	Aggregate	`addinfo`	Augment	`addtotals`	Augment
`addtotals col`	Aggregate	`addtotals row`	Augment	`anomalies`	Window	`append`	Set
`appendcols`	Augment	`appendpipe`	Set	`audit`	Read Metadata	`bin`	Augment
`bucket`	Augment	`chart`	Aggregate	`collect`	Cache	`contingency`	Aggregate
`convert`	Transform	`counttable`	Aggregate	`datamodel`	Input	`dbinspect`	Read Metadata
`dbquery`	Input	`dedup`	Filter	`delete`	Meta	`delta`	Window
`eval`	Augment	`eventcount`	Aggregate	`eventstats`	Augment	`export`	Output
`extract`	Augment	`fieldformat`	Transform	`fields`	Project	`filldown`	Transform
`fillnull`	Transform	`format`	Miscellaneous	`gauge`	Augment	`geoip`	Join
`geostats`	Aggregate	`head`	Filter	`history`	Read Metadata	`inputcsv`	Input
`inputlookup`	Input	`iplocation`	Augment	`join`	Join	`kv`	Augment
`loadjob`	Input	`localop`	Meta	`lookup`	Join	`macro`	Macro
`makemv`	Transform	`map`	Miscellaneous	`metadata`	Input	`metasearch`	Meta
`multikv`	Transform	`mvcombine`	Aggregate	`mvexpand`	Transform	`nomv`	Transform
`noop`	Miscellaneous	`outlier`	Transform	`outputcsv`	Output	`outputlookup`	Output
`outputtext`	Augment	`overlap`	Miscellaneous	`rangemap`	Augment	`rare`	Aggregate
`regex`	Filter	`relevancy`	Augment	`rename`	Rename	`replace`	Transform
`rest`	Input	`return`	Miscellaneous	`reverse`	Reorder	`rex`	Augment
`savedsearch`	Input	`search`	Filter	`sendemail`	Output	`set`	Set
`sichart`	Cache	`sirare`	Cache	`sistats`	Cache	`sitimechart`	Cache
`sitop`	Cache	`sort`	Reorder	`spath`	Augment	`stats`	Aggregate
`strcat`	Augment	`streamstats`	Window	`summarize`	Cache	`summaryindex`	Cache
`table`	Project	`tags`	Augment	`tail`	Filter	`timechart`	Aggregate
`timewrap`	Aggregate	`top`	Aggregate	`transaction`	Aggregate	`transpose`	Transpose
`tscollect`	Cache	`tstats`	Aggregate	`typeahead`	Read Metadata	`uniq`	Filter
`where`	Filter	`xmlkv`	Augment				

Table A.1: Categories for each type of Splunk event.

Appendix B

Splunk Cluster Features

The following tables describe the features we used to classify three categories of Splunk commands: **Filter**, **Aggregate**, and **Augment** commands. When a feature is described as "parameterized", this means that there were actually multiple features of that kind, one for each possible parameter value passed. Thus, there were more actual features than listed in the table, because the parameterized features actually represent multiple features. The number of features for each category were as follows:

- **Filter:** 26 features
- **Aggregate:** 46 features
- **Augment:** 127 features

Feature Name	**Description**
NumberOfArguments	the number of arguments
NumberOfWildcardMatches	the number of wildcard operators passed as an argument
NumberOfBooleans	the number of the boolean operators `AND`, `OR`, or `NOT` passed as an argument
NumberOfSpecificBooleans	(parameterized) the number of times the specified boolean operator was passed as an argument
NumberOfFieldSearches	the number of times a search on a specific field is specified in the arguments
NumberOfOptionSearches	the number of times options are specified as an argument
NumberOfStringSearches	the number of strings passed as arguments
AvgLengthOfStringSearches	the average length of the strings passed
NumberOfLogicalOperators	the number of comparison operators like `eq` (equal) or `gt` (greater than) passed as arguments
NumberOfTimeSearches	the number of searches over the time dimension
NumberOfSubsearches	the number of subsearches passed as an argument
NumberOfInternalDataSearches	the number internal Splunk fields (not part of the user's dataset) passed as arguments
NumberOfDefaultFieldSearches	the number default Splunk fields (part of all datasets by default) passed as arguments
NumberOfDistinctFields	the number of distinct fields passed as arguments
SpecifiedCommandUsed	(parameterized) whether the specified command was used
PositionInQuery	which stage of the query the command appeared in
RemovesDuplicates	whether the command removes duplicate results
FiltersByPosition	whether the command returns the first or last result

Table B.1: Features used to cluster Splunk **Filter** operations.

Feature Name	**Description**
NumberOfDistinctFields	the number of distinct fields passed as arguments
SpecifiedCommandUsed	(parameterized) whether the specified command was used
PositionInQuery	which stage of the query the command appeared in
NumberTimesAggregateUsed	(parameterized) the number of times a specified aggregate function `AVG` or `COUNT` was passed as an argument
NumberTimesPercentileUsed	the number of times a percentile function was passed as an argument
NumberOfFieldsAggregated	the number of fields aggregated in the arguments
NumberOfFieldsGroupedBy	the number of fields grouped by in the arguments
ReordersAndLimitsResults	whether or not the command reorders the results and limits the number that are returned
ForVisualization	whether or not the command is designed to be piped into a visualization

Table B.2: Features used to cluster Splunk **Aggregate** operations.

Feature Name	**Description**
NumberOfDistinctFields	the number of distinct fields passed as arguments
SpecifiedCommandUsed	(parameterized) whether the specified command was used
PositionInQuery	which stage of the query the command appeared in
NumberOfFunctions	the number total functions passed as an argument
NumberOfDistinctFunctions	the number distinct functions passed as an argument
NumberOfSpecificFunctions	(parameterized) the number times the specified function was passed as an argument
DeepestLevelOfNesting	the depth of the parsed query's tree – a rough measure of query complexity
NumberSpecificTypeFunctions	(parameterized) the number times a function with the specified domain or range datatype was passed as an argument
PercentSpecificTypeFunctions	(parameterized) the percent of times a function with the specified domain or range datatype was passed as an argument

Table B.3: Features used to cluster Splunk **Augment** operations.

Appendix C

Tableau Action Categories

The following tables show how we categorized each Tableau event action type for our analysis in Chapter 5. We used the following 15 high-level categories:

- aesthetics
- animate
- annotation
- dashboard-or-story
- datasource
- edit
- field
- filter
- pan-zoom-highlight
- sheet
- showme
- sort
- tableau
- undo-or-redo
- unknown

Command	Category
`tabdoc:add-object-to-dashboard`	dashboard-or-story
`tabdoc:add-or-replace-trend-lines`	annotation
`tabdoc:add-reference-line`	annotation
`tabdoc:add-sheet-to-dashboard`	dashboard-or-story
`tabdoc:add-to-sheet`	edit
`tabdoc:aggregate-measures`	field
`tabdoc:apply-calculation`	field
`tabdoc:apply-type-in-pill`	field
`tabdoc:calculation-auto-complete`	tableau
`tabdoc:capture-story-point`	dashboard-or-story
`tabdoc:categorical-filter`	filter
`tabdoc:categorical-quick-filter-edit`	filter
`tabdoc:categorical-quick-filter-exclude-values`	filter
`tabdoc:categorical-quick-filter-mode`	filter
`tabdoc:cell-size`	aesthetics
`tabdoc:change-alpha-level`	aesthetics
`tabdoc:change-border`	aesthetics
`tabdoc:change-data-type`	field
`tabdoc:change-label-align`	aesthetics
`tabdoc:change-label-font`	aesthetics
`tabdoc:change-markers`	aesthetics
`tabdoc:change-page`	animate
`tabdoc:change-page-by-direction`	animate
`tabdoc:change-semantic-role`	field
`tabdoc:change-shared-filter`	filter
`tabdoc:change-size`	dashboard-or-story
`tabdoc:clear-alias-cache`	datasource
`tabdoc:clear-calculation-model`	tableau
`tabdoc:clear-filters`	filter
`tabdoc:clear-formatting`	aesthetics
`tabdoc:clear-legend-selection`	filter
`tabdoc:clear-quick-filter`	filter
`tabdoc:clear-sorts`	sort
`tabdoc:commit-alias-changes`	field
`tabdoc:commit-single-alias-changes`	aesthetics
`tabdoc:connection-dataserver-flush-cache`	datasource
`tabdoc:connection-delete-table`	datasource
`tabdoc:connection-edit-join-clauses`	datasource
`tabdoc:connection-edit-join-type`	datasource
`tabdoc:connection-edit-or-rename-union`	datasource
`tabdoc:connection-edit-union`	datasource
`tabdoc:connection-toggle-table-cleaning`	datasource
`tabdoc:connection-toggle-use-extract`	datasource

Table C.1: Categories for each type of Tableau event.

Command	**Category**
`tabdoc:connection-view-table-cleaning-results`	datasource
`tabdoc:convert-to-dimension`	field
`tabdoc:convert-to-discrete`	field
`tabdoc:convert-to-measure`	field
`tabdoc:create-calc`	field
`tabdoc:custom-split-field`	unknown
`tabdoc:default-map-tool-selection`	unknown
`tabdoc:delete-calculation-fields-command`	unknown
`tabdoc:dismiss-null-wart`	pan-zoom-highlight
`tabdoc:domain-quick-filter`	filter
`tabdoc:drop-on-dashboard`	dashboard-or-story
`tabdoc:dual-axis`	edit
`tabdoc:duplicate-fields`	datasource
`tabdoc:edit-calc`	field
`tabdoc:edit-datasource-date-properties`	datasource
`tabdoc:edit-nested-group`	field
`tabdoc:edit-primitive-type`	unknown
`tabdoc:edit-schema-caption`	aesthetics
`tabdoc:edit-story-point-caption`	dashboard-or-story
`tabdoc:enable-quick-filter`	filter
`tabdoc:enable-themed-highlights`	aesthetics
`tabdoc:extract`	unknown
`tabdoc:filter-nulls`	filter
`tabdoc:filter-targets`	filter
`tabdoc:forecast-toggle`	annotation
`tabdoc:get-alias-pres-model`	tableau
`tabdoc:get-aliases-pres-model`	tableau
`tabdoc:get-container-guide`	tableau
`tabdoc:get-dashboard-drag-drop`	dashboard-or-story
`tabdoc:get-data-preview-window-pres-model`	tableau
`tabdoc:get-datasource-date-properties`	datasource
`tabdoc:get-format-pane-layout`	unknown
`tabdoc:get-page-setup-pres-model`	tableau
`tabdoc:get-sort-dialog-pres-model`	tableau
`tabdoc:get-union-info-pres-model`	tableau
`tabdoc:get-zone-border-rect`	tableau
`tabdoc:goto-sheet`	sheet
`tabdoc:group-by-table`	datasource
`tabdoc:hide-column-field-label`	annotation
`tabdoc:hide-field`	datasource
`tabdoc:hide-label`	aesthetics
`tabdoc:hide-legend-data`	tableau
`tabdoc:hide-row-field-label`	aesthetics
`tabdoc:highlight`	pan-zoom-highlight
`tabdoc:highlight-selected-in-legend`	pan-zoom-highlight

Table C.2: Categories for each type of Tableau event.

Command	Category
`tabdoc:highlight-special`	pan-zoom-highlight
`tabdoc:include-in-tooltip`	annotation
`tabdoc:insert-function-in-formula`	unknown
`tabdoc:insert-story-point`	dashboard-or-story
`tabdoc:invoke-hyperlink`	tableau
`tabdoc:keep-only-or-exclude`	filter
`tabdoc:keep-only-or-exclude-selection`	filter
`tabdoc:legend-group-or-ungroup`	unknown
`tabdoc:legend-keep-or-exclude`	filter
`tabdoc:level-drill`	field
`tabdoc:load-bin-size-info`	unknown
`tabdoc:load-domain-info`	unknown
`tabdoc:mark-label`	annotation
`tabdoc:master-detail-filter`	filter
`tabdoc:merge-or-split`	unknown
`tabdoc:modify-legend-layout`	aesthetics
`tabdoc:move-area-annotation-text`	annotation
`tabdoc:move-dashboard-edge`	dashboard-or-story
`tabdoc:move-free-form-zone`	dashboard-or-story
`tabdoc:move-mark`	aesthetics
`tabdoc:move-point-annotation`	annotation
`tabdoc:move-point-annotation-target`	annotation
`tabdoc:move-reference-line`	annotation
`tabdoc:move-zone`	dashboard-or-story
`tabdoc:navigate-to-sheet`	sheet
`tabdoc:new-blank-story-point`	dashboard-or-story
`tabdoc:non-ranged-remove-reference-line`	annotation
`tabdoc:on-toggle-axis-ranges`	annotation
`tabdoc:page-animation-control`	animate
`tabdoc:pane-pan`	pan-zoom-highlight
`tabdoc:pane-zoom`	pan-zoom-highlight
`tabdoc:pane-zoom-factor`	pan-zoom-highlight
`tabdoc:pane-zoom-pan`	pan-zoom-highlight
`tabdoc:percentages`	unknown
`tabdoc:quantitative-mode-quick-filter`	filter
`tabdoc:quantitative-quick-filter-edit`	filter
`tabdoc:quick-sort`	sort
`tabdoc:ranged-by-value-merge`	unknown
`tabdoc:redo`	undo-or-redo
`tabdoc:relative-date-quick-filter-edit`	filter
`tabdoc:remove-all-reference-lines`	annotation
`tabdoc:remove-field-label`	annotation
`tabdoc:remove-fields`	edit
`tabdoc:remove-fields-from-shelf`	edit
`tabdoc:remove-from-schema-field-folder`	datasource

Table C.3: Categories for each type of Tableau event.

Command	Category
`tabdoc:remove-reference-line`	annotation
`tabdoc:rename-field`	field
`tabdoc:rename-sheet`	sheet
`tabdoc:reorder-labels`	aesthetics
`tabdoc:reorder-legend-items`	aesthetics
`tabdoc:resize-area-annotation-text`	annotation
`tabdoc:resize-axis`	aesthetics
`tabdoc:resize-cell`	aesthetics
`tabdoc:resize-header`	aesthetics
`tabdoc:resize-legend-column`	aesthetics
`tabdoc:resize-point-annotation`	annotation
`tabdoc:rotate-field-label`	aesthetics
`tabdoc:rotate-label`	annotation
`tabdoc:select-all`	pan-zoom-highlight
`tabdoc:select-axis-tuples`	unknown
`tabdoc:select-fields-in-shelf`	edit
`tabdoc:select-legend-item`	pan-zoom-highlight
`tabdoc:select-legend-items`	pan-zoom-highlight
`tabdoc:set-active-datasource`	datasource
`tabdoc:set-active-story-point`	dashboard-or-story
`tabdoc:set-dashboard-size`	dashboard-or-story
`tabdoc:set-dashboard-sizing`	dashboard-or-story
`tabdoc:set-data-preview-auto-update`	datasource
`tabdoc:set-data-preview-custom-field-order`	datasource
`tabdoc:set-default-aggregation`	field
`tabdoc:set-default-color`	aesthetics
`tabdoc:set-default-shape`	aesthetics
`tabdoc:set-filter-context`	filter
`tabdoc:set-filter-shared`	filter
`tabdoc:set-item-color`	aesthetics
`tabdoc:set-parameter-ctrl-display-mode`	unknown
`tabdoc:set-parameter-value`	field
`tabdoc:set-primitive`	edit
`tabdoc:set-row-display-count`	unknown
`tabdoc:set-sheet-formatting`	aesthetics
`tabdoc:set-story-point-caption-width`	dashboard-or-story
`tabdoc:set-zone-is-fixed-size`	tableau
`tabdoc:show-aliased-fields`	tableau
`tabdoc:show-cat-ctrl-quick-filter`	filter
`tabdoc:show-col-totals`	annotation
`tabdoc:show-header`	aesthetics
`tabdoc:show-hidden-fields`	datasource
`tabdoc:show-legend-on-dashboard`	dashboard-or-story
`tabdoc:show-me`	showme
`tabdoc:show-quant-ctrl-quick-filter`	filter

Table C.4: Categories for each type of Tableau event.

Command	**Category**
`tabdoc:show-reference-line-editor`	annotation
`tabdoc:show-reference-line-formatter`	aesthetics
`tabdoc:show-sv-at-default`	unknown
`tabdoc:show-trend-line-editor`	annotation
`tabdoc:show-trend-line-formatter`	aesthetics
`tabdoc:simple-command-list`	aesthetics
`tabdoc:sort`	sort
`tabdoc:sort-datagrid-by-column`	datasource
`tabdoc:sort-datagrid-fields`	sort
`tabdoc:sort-from-indicator`	sort
`tabdoc:special-values`	unknown
`tabdoc:swap-rows-and-columns`	edit
`tabdoc:synchronize-axis`	edit
`tabdoc:theme`	aesthetics
`tabdoc:toggle-drop-lines`	annotation
`tabdoc:toggle-field-blending`	unknown
`tabdoc:toggle-freeform-zone`	dashboard-or-story
`tabdoc:toggle-legend-item-selection`	filter
`tabdoc:toggle-legend-title`	aesthetics
`tabdoc:toggle-lock-quick-filters`	filter
`tabdoc:toggle-mark-labels`	aesthetics
`tabdoc:toggle-parameter-ctrl-title`	tableau
`tabdoc:toggle-quick-filter-title`	filter
`tabdoc:trend-lines`	annotation
`tabdoc:try-set-aliases`	aesthetics
`tabdoc:undo`	undo-or-redo
`tabdoc:ungroup-layout-container`	unknown
`tabdoc:update-area-annotation`	annotation
`tabdoc:update-dashboard-alignment`	dashboard-or-story
`tabdoc:update-mark-labels-settings`	aesthetics
`tabdoc:update-point-annotation-pullback`	annotation
`tabdoc:update-selection-delta`	unknown
`tabdoc:view-quick-filters`	filter
`tabdoc:zoom-controls-toggle`	pan-zoom-highlight
`tabdoc:zoom-level`	pan-zoom-highlight
`tabui:about`	tableau
`tabui:actions-edit-dashboard`	dashboard-or-story
`tabui:actions-edit-worksheet`	animate
`tabui:add-qc-to-sheet-ui`	unknown
`tabui:add-reference-line-ui`	annotation
`tabui:add-to-schema-field-folder-ui`	datasource
`tabui:add-to-sheet-ui`	edit
`tabui:annotate-ui`	annotation
`tabui:cat-mode-quick-filter`	filter
`tabui:change-aggregation-ui`	field

Table C.5: Categories for each type of Tableau event.

Command	Category
tabui:change-field-type-ui	field
tabui:change-tab-color-sheet	sheet
tabui:clear-sheet-ui	edit
tabui:clear-viz-ui	edit
tabui:close-datasource	datasource
tabui:close-datasource-ui	datasource
tabui:close-workbook	tableau
tabui:color-legend-edit-ui	aesthetics
tabui:commit-pill-edit-ui	unknown
tabui:connect-datasource	datasource
tabui:connection-add-new-table-ui	datasource
tabui:connection-analyze-data-ui	datasource
tabui:connection-data-grid-copy-ui	datasource
tabui:connection-duplicate-relation-ui	datasource
tabui:connection-edit-connection-ui	datasource
tabui:connection-extract-ui	datasource
tabui:copy-crosstab	tableau
tabui:copy-data	unknown
tabui:copy-fields-defn-ui	unknown
tabui:copy-formatting	aesthetics
tabui:copy-image	unknown
tabui:copy-image-dashboard	tableau
tabui:copy-image-worksheet	tableau
tabui:create-calculation-ui	field
tabui:create-categorical-bins-ui	field
tabui:create-combined-field-ui	datasource
tabui:create-connection-ui	datasource
tabui:create-group-selection-ui	field
tabui:create-group-ui	field
tabui:create-numeric-bins-ui	field
tabui:create-parameter-ui	field
tabui:create-set-ui	field
tabui:custom-date-aggregation-ui	datasource
tabui:deactivate-dashboard	dashboard-or-story
tabui:delete-fields-ui	datasource
tabui:delete-sheet-ui	sheet
tabui:delete-story-point-ui	dashboard-or-story
tabui:describe-field-ui	datasource
tabui:describe-trend-model	annotation
tabui:dismiss-go-to-worksheet-tooltip-ui	tableau
tabui:display-encoding-type-menu	unknown
tabui:document	tableau
tabui:domain-quick-filter	filter
tabui:drop-ui	edit
tabui:duplicate-sheet-or-crosstab-ui	sheet

Table C.6: Categories for each type of Tableau event.

Command	**Category**
`tabui:duplicate-sheet-ui`	sheet
`tabui:edit-aliases-ui`	field
`tabui:edit-annotation`	annotation
`tabui:edit-axis-ui`	annotation
`tabui:edit-caption`	annotation
`tabui:edit-combined-field`	unknown
`tabui:edit-datasource`	datasource
`tabui:edit-datasource-date-properties-ui`	datasource
`tabui:edit-datasource-filters-ui`	filter
`tabui:edit-datasource-ui`	datasource
`tabui:edit-default-sort-ui`	sort
`tabui:edit-drop-lines-ui`	annotation
`tabui:edit-filter-quick-filter`	filter
`tabui:edit-filter-ui`	filter
`tabui:edit-group-ui`	field
`tabui:edit-legend-member-alias-ui`	field
`tabui:edit-legend-title-ui`	aesthetics
`tabui:edit-member-alias-label`	aesthetics
`tabui:edit-param-control-title`	aesthetics
`tabui:edit-parameter`	field
`tabui:edit-reference-line`	annotation
`tabui:edit-schema-calculation-ui`	field
`tabui:edit-schema-caption-ui`	annotation
`tabui:edit-schema-categorical-bins-ui`	field
`tabui:edit-schema-field-alias-ui`	field
`tabui:edit-schema-numeric-bins-ui`	field
`tabui:edit-text`	aesthetics
`tabui:edit-title`	annotation
`tabui:edit-title-quick-filter`	filter
`tabui:edit-tooltip`	annotation
`tabui:edit-trend-lines-ui`	annotation
`tabui:edit-user-description-ui`	datasource
`tabui:edit-zone-param`	dashboard-or-story
`tabui:exit-app`	tableau
`tabui:exit-application`	tableau
`tabui:export-data`	tableau
`tabui:export-packaged`	tableau
`tabui:export-workbook-sheets-ui`	sheet
`tabui:extract-ui`	unknown
`tabui:filter-field-label-ui`	filter
`tabui:format-annotation`	aesthetics
`tabui:format-axis-ui`	aesthetics
`tabui:format-caption-title`	aesthetics
`tabui:format-field-label-ui`	aesthetics
`tabui:format-label`	aesthetics

Table C.7: Categories for each type of Tableau event.

Command	Category
`tabui:format-legends-ui`	aesthetics
`tabui:format-reference-line`	aesthetics
`tabui:format-ui`	aesthetics
`tabui:get-open-pane-sample-workbooks`	tableau
`tabui:goto`	tableau
`tabui:goto-url`	tableau
`tabui:hide-caption`	aesthetics
`tabui:hide-filmstrip-ui`	sheet
`tabui:hide-title`	annotation
`tabui:hide-zone-ui`	dashboard-or-story
`tabui:import-workbook-ui`	tableau
`tabui:include-exclude-values-quick-filter`	filter
`tabui:manage-map-services`	tableau
`tabui:map-layers-ui`	tableau
`tabui:map-options-ui`	aesthetics
`tabui:new-dashboard-ui`	dashboard-or-story
`tabui:new-schema-drill-path-ui`	field
`tabui:new-storyboard-ui`	dashboard-or-story
`tabui:new-workbook`	tableau
`tabui:new-worksheet-ui`	sheet
`tabui:non-ranged-edit-reference-line`	annotation
`tabui:open-other-files`	tableau
`tabui:open-workbook`	tableau
`tabui:page-setup`	animate
`tabui:paste`	sheet
`tabui:presentation-mode`	tableau
`tabui:print`	tableau
`tabui:prompt-for-datasource-file-ui`	datasource
`tabui:properties-datasource-ui`	datasource
`tabui:publish-workbook-to-workgroup`	tableau
`tabui:quick-filter-field-label-ui`	filter
`tabui:quick-table-calc-ui`	field
`tabui:refresh-datasource`	datasource
`tabui:refresh-datasource-ui`	datasource
`tabui:remove-annotation`	annotation
`tabui:remove-mru`	tableau
`tabui:remove-sheet-from-dashboard-ui`	dashboard-or-story
`tabui:rename-datasource`	datasource
`tabui:rename-sheet-ui`	sheet
`tabui:reorder-sheets-ui`	sheet
`tabui:replace-field-ui`	edit
`tabui:replace-story-point-ui`	dashboard-or-story
`tabui:reset-caption-ui`	annotation
`tabui:reset-cards`	tableau
`tabui:save-as-workbook`	tableau

Table C.8: Categories for each type of Tableau event.

Command	**Category**
`tabui:save-public`	tableau
`tabui:save-workbook`	tableau
`tabui:scroll-sheet`	sheet
`tabui:select-parent-zone`	tableau
`tabui:set-active-connection-ui`	datasource
`tabui:set-default-date-format-ui`	datasource
`tabui:set-default-location-ui`	datasource
`tabui:set-default-mapsource-ui`	datasource
`tabui:set-default-text-format-ui`	aesthetics
`tabui:set-filter-shared-ui`	filter
`tabui:set-free-form-dashboard-ui`	dashboard-or-story
`tabui:set-worksheet-sidepane-mode`	tableau
`tabui:set-zone-fixed-size-ui`	dashboard-or-story
`tabui:shape-legend-edit-ui`	tableau
`tabui:sheet-sorter`	sheet
`tabui:show-cat-ctrl-quick-filter`	filter
`tabui:show-connect-tab`	tableau
`tabui:show-connection-auth-ui`	tableau
`tabui:show-custom-split-dialog-ui`	tableau
`tabui:show-field-format-ui`	aesthetics
`tabui:show-field-in-schema`	tableau
`tabui:show-filmstrip-ui`	tableau
`tabui:show-in-out-sets`	datasource
`tabui:show-map-options-dialog`	tableau
`tabui:show-me-cycle-ui`	showme
`tabui:show-me-hide-ui`	tableau
`tabui:show-measures-ui`	tableau
`tabui:show-members-ui`	datasource
`tabui:show-parameter-control`	field
`tabui:show-quickfilter`	filter
`tabui:show-quickfilter-ui`	filter
`tabui:size-legend-edit-ui`	aesthetics
`tabui:sort-ascending-field-label-ui`	sort
`tabui:sort-descending-field-label-ui`	sort
`tabui:sort-field-label-ui`	sort
`tabui:sort-ui`	sort
`tabui:split-field-ui`	unknown
`tabui:stack-marks-ui`	aesthetics
`tabui:start-page`	tableau
`tabui:start-pill-edit-ui`	unknown
`tabui:switch-to-document`	sheet
`tabui:switch-to-sheet-sorter`	sheet
`tabui:table-calc-ignore-ui`	field
`tabui:table-calc-ordering-ui`	field
`tabui:table-calc-ui`	field

Table C.9: Categories for each type of Tableau event.

Command	**Category**
`tabui:toggle-connect-pane`	tableau
`tabui:toggle-filmstrip-ui`	sheet
`tabui:toggle-start-page`	tableau
`tabui:trend-lines-ui`	annotation
`tabui:upgrade-datasources-ui`	datasource
`tabui:view-caption-card`	tableau
`tabui:view-card`	aesthetics
`tabui:view-columns-shelf`	tableau
`tabui:view-dashboard-title`	dashboard-or-story
`tabui:view-data`	datasource
`tabui:view-datasource-data`	datasource
`tabui:view-legend`	annotation
`tabui:view-marks-shelf`	tableau
`tabui:view-quick-filters`	filter
`tabui:view-rows-shelf`	tableau
`tabui:view-side-pane`	tableau
`tabui:view-summary-card`	aesthetics
`tabui:view-summary-detail`	unknown
`tabui:view-table-data`	tableau
`tabui:view-title-card`	annotation

Table C.10: Categories for each type of Tableau event.

Bibliography

[1] W. van der Aalstand, H. Reijers, A. Weijters, B. van Dongen, A. De Medeiros, M. Song, and H. Verbeek. “Business process mining: An industrial application.” In: *Information Systems* (2007).

[2] A. Abouzied, J. Hellerstein, and A. Silberschatz. “DataPlay: interactive tweaking and example-driven correction of graphical database queries.” In: *ACM Symposium on User Interface Software and Technology (UIST)*. 2012.

[3] E. Agichtein, E. Brill, and S. Dumais. “Improving web search ranking by incorporating user behavior information.” In: *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 2006.

[4] J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, and E. Turricchia. “Similarity measures for OLAP sessions.” In: *Knowledge and Information Systems* (2014).

[5] S. Alspaugh and A. Ganapathi. “Towards a data analysis recommendation system.” In: *USENIX OSDI Workshop on Managing Systems Automatically and Dynamically (MAD)*. 2012.

[6] S. Alspaugh, B. Chen, J. Lin, A. Ganapathi, M. Hearst, and R. Katz. “Analyzing Log Analysis: An Empirical Study of User Log Mining.” In: *Large Installation System Administration Conference (LISA)*. 2014.

[7] S. Alspaugh, A. Ganapathi, M. Hearst, and R. Katz. “Better logging to improve interactive data analysis tools.” In: *ACM KDD Workshop on Interactive Data Exploration and Analytics (IDEA)*. 2014.

[8] S. Alspaugh, A. Ganapathi, M. Hearst, and R. Katz. “Building blocks for exploratory data analysis tools.” In: *Workshop on Interactive Data Exploration and Analytics (IDEA). Co-located with KDD.* 2013.

[9] R. St. Amant and P. Cohen. “Intelligent support for exploratory data analysis.” In: *ASA Journal of Computational and Graphical Statistics* (1998).

[10] R. Amar, J. Eagan, and J. Stasko. “Low-level components of analytic activity in information visualization.” In: *IEEE Information Visualization Conference (InfoVis)*. 2005.

[11] N. Andrienko, G. Andrienko, and P. Gatalsky. “Exploratory spatio-temporal visualization: an analytical review.” In: *Journal of Visual Languages & Computing* (2003).

[12] *Aqua Data Studio*. `http://www.aquafold.com/aquadatastudio.html`.

[13] R. Atterer, M. Wnuk, and A. Schmidt. “Knowing the user’s every move: user activity tracking for website usability evaluation and implicit interaction.” In: *ACM International Conference on World Wide Web (WWW)*. 2006.

[14] M. Aufaure, N. Kuchmann-Beauger, P. Marcel, S. Rizzi, and Y. Vanrompay. “Predicting your next OLAP query based on recent analytical sessions.” In: *International Conference on Data Warehousing and Knowledge Discovery*. Springer. 2013.

[15] E. Ayers and J. Stasko. *Using graphic history in browsing the World Wide Web*. Tech. rep. Georgia Institute of Technology, 1995.

[16] J. Bargas-Avila and K. Hornbæk. “Old wine in new bottles or novel challenges: a critical analysis of empirical studies of user experience.” In: *ACM Conference on Human Factors in Computing Systems (CHI)* (2011).

[17] L. Bavoil, S. Callahan, P. Crossno, J. Freire, C. Scheidegger, C. Silva, and H. Vo. “VisTrails: enabling interactive multiple-view visualizations.” In: *IEEE Visualization (VIS)*. 2005.

[18] A. Bernstein, F. Provost, and S. Hill. “Toward intelligent assistance for a data mining process: an ontology-based approach for cost-sensitive classification.” In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2005).

[19] A. Bhardwaj, A. Deshpande, A. Elmore, D. Karger, S. Madden, A. Parameswaran, H. Subramanyam, E. Wu, and R. Zhang. “Collaborative data analytics with DataHub.” In: *International Conference on Very Large Databases (VLDB)* (2015).

[20] L. Bitincka, A. Ganapathi, S. Sorkin, and S. Zhang. “Optimizing data analysis with a semi-structured time series database.” In: *OSDI Workshop on Managing Systems via Log Analysis and Machine Learning Techniques (SLAML)*. 2010.

[21] I. Boyandin, E. Bertini, and D. Lalanne. “A qualitative study on the exploration of temporal changes in flow maps with animation and small-multiples.” In: *Computer Graphics Forum*. 2012.

[22] K. Brodlie, A. Poon, H. Wright, L. Brankin, G. Banecki, and A. Gay. “GRASPARC: a problem solving environment integrating computation and visualization.” In: *IEEE Conference on Visualization (VIS)*. 1993.

[23] E. Brown, A. Ottley, H. Zhao, Q. Lin, R. Souvenir, A. Endert, and R. Chang. “Finding Waldo: learning about users from their interactions.” In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2014).

[24] A. Buja, D. Cook, and D. Swayne. “Interactive high-dimensional data visualization.” In: *Journal of Computational and Graphical Statistics* (1996).

[25] S. Card, J. Mackinlay, and B. Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[26] S. Card, A. Newell, and T. Moran. *The psychology of human-computer interaction.* L. Erlbaum Associates Inc., 1983.

[27] S. Casner. “Task-analytic approach to the automated design of graphic presentations.” In: *ACM Transactions on Graphics (TOG)* (1991).

[28] R. Chang, C. Ziemkiewicz, T. Green, and W. Ribarsky. “Defining insight for visual analytics.” In: *IEEE Computer Graphics and Applications* (2009).

[29] S. Chasins, S. Barman, R. Bodik, and S. Gulwani. “Browser record and replay as a building block for end-user web automation tools.” In: *ACM International Conference on World Wide Web (WWW)*. 2015.

[30] G. Chatzopoulou, M. Eirinaki, S. Koshy, S. Mittal, N. Polyzotis, and J. Varman. “The QueRIE system for personalized query recommendations.” In: *IEEE Data Engineering Bulletin* (2011).

[31] Y. Chen, S. Alspaugh, and R. Katz. “Interactive analytical processing in big data systems: a cross-industry study of MapReduce workloads.” In: *International Conference on Very Large Databases (VLDB)* (2012).

[32] E. Chi, P. Pirolli, K. Chen, and J. Pitkow. “Using information scent to model user information needs and actions and the web.” In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 2001.

[33] M. Chiarini. “Provenance for system troubleshooting.” In: *USENIX Conference on System Administration (LISA)*. 2011.

[34] M. Chuah and S. Roth. “On the semantics of interactive visualizations.” In: *IEEE Information Visualization Conference (InfoVis)*. 1996.

[35] K. Cook, N. Cramer, D. Israel, M. Wolverton, J. Bruce, R. Burtner, and A. Endert. “Mixed-initiative visual analytics using task-driven recommendations.” In: *IEEE Visual Analytics Science and Technology (VAST)*. 2015.

[36] P. Cowley, L. Nowell, and J. Scholtz. “Glass box: an instrumented infrastructure for supporting human interaction with information.” In: *IEEE Hawaii International Conference on System Sciences (HICSS)*. 2005.

[37] P. Cowley, J. Haack, R. Littlefield, and E. Hampson. “Glass box: capturing, archiving, and retrieving workstation activities.” In: *ACM Workshop on Continuous Archival and Retrival of Personal Experiences*. 2006.

[38] Inc. Distributed Management Task Force. *Common Information Model*. `http://www.dmtf.org/standards/cim`. 2016.

[39] A. Dix and G. Ellis. “Starting simple: adding value to static visualisation through simple interaction.” In: *ACM Conference on Advanced Visual Interfaces (AVI)*. 1998.

[40] W. Dou, D. Jeong, F. Stukes, W. Ribarsky, H. Lipford, and R. Chang. “Recovering reasoning process from user interactions.” In: *IEEE Computer Graphics and Applications* (2009).

[41] M. Drosou and E. Pitoura. “YmalDB: a result-driven recommendation system for databases.” In: *ACM International Conference on Extending Database Technology (EDBT)*. 2013.

[42] C. Dunne, N. Riche, B. Lee, R. Metoyer, and G. Robertson. “GraphTrail: analyzing large multivariate, heterogenous networks while supporting exploration history.” In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 2012.

[43] J. Foreman. *Data smart: using data science to transform information into insight.* John Wiley and Sons, Inc., 2014.

[44] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. “Evaluating implicit measures to improve web search.” In: *ACM Transactions on Information Systems (TOIS)* (2005).

[45] E. Gamma. *Design patterns: elements of reusable object-oriented software.* Pearson Education India, 1995.

[46] *Gene Ontology.* `http://www.geneontology.org`.

[47] S. Gomez and D. Laidlaw. “Modeling task performance for a crowd of users from interaction histories.” In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 2012.

[48] S. Gomez, H. Guo, C. Ziemkiewicz, and D. Laidlaw. “An insight-and task-based methodology for evaluating spatiotemporal visual analytics.” In: *IEEE Visual Analytics Science and Technology (VAST)*. 2014.

[49] D. Gotz and M. Zhou. *An empirical study of user interaction behavior during visual analysis.* Tech. rep. IBM Research, 2008.

[50] D. Gotz and M. Zhou. “Characterizing users’ visual analytic activity for insight provenance.” In: *IEEE Information Visualization Conference (InfoVis)* (2009).

[51] D. Gotz, Z. Wen, J. Lu, P. Kissa, M. Zhou, N. Cao, W. Qian, and S. Liu. “HARVEST: situational visual analytics for the masses.” In: *Workshop on Intelligent Visual Interfaces for Text Analysis (IVITA)*. 2010.

[52] J. Gray. *Why do computers stop and what can be done about it?* Tech. rep. Microsoft Research, 1985.

[53] W. Gray, B. John, and M. Atwood. “Project Ernestine: validating a GOMS analysis for predicting and explaining real-world task performance.” In: *Human-computer Interaction* (1993).

[54] T. Grossman, J. Matejka, and G. Fitzmaurice. “Chronicle: capture, exploration, and playback of document workflow histories.” In: *ACM Symposium on User Interface Software and Technology (UIST)*. 2010.

[55] H. Guo, S. Gomez, C. Ziemkiewicz, and D. Laidlaw. “A case study using visualization interaction logs and insight metrics to understand how analysts arrive at insights.” In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2016).

[56] P. Guo, S. Kandel, J. Hellerstein, and J. Heer. "Proactive wrangling: mixed-initiative end-user programming of data transformation scripts." In: *ACM Symposium on User Interface Software and Technology (UIST)*. 2011.

[57] Q. Guo and E. Agichtein. "Ready to buy or just browsing?: detecting web searcher goals from interaction data." In: *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 2010.

[58] H. Haugerud and S. Straumsnes. "Simulation of user-driven computer behaviour." In: *USENIX Conference on System Administration (LISA)*. 2001.

[59] M. Hearst. *Search User Interfaces*. Cambridge University Press, 2009.

[60] J. Heer, F. Viégas, and M. Wattenberg. "Voyagers and voyeurs: supporting asynchronous collaborative information visualization." In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 2007.

[61] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. "Graphical histories for visualization: Supporting analysis, communication, and evaluation." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2008).

[62] D. Henschen. "Splunk leads tiny big data market." In: *Information Week* (2015). `http://www.informationweek.com/big-data/big-data-analytics/splunk-leads-tiny-big-data-market/a/d-id/1319295`.

[63] D. Hilbert and D. Redmiles. "An approach to large-scale collection of application usage data over the Internet." In: *IEEE International Conference on Software Engineering (ICSE)*. 1998.

[64] D. Hilbert and D. Redmiles. "Extracting usability information from user interface events." In: *ACM Computing Surveys (CSUR)*. 2000.

[65] H. Hoppe. "Task-oriented parsing-a diagnostic method to be used adaptive systems." In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 1988.

[66] *Horizontal merger guidelines*. Tech. rep. U.S. Department of Justice and the Federal Trade Commission, 2010. URL: `{https://www.justice.gov/atr/horizontal-merger-guidelines-08192010#5c}`.

[67] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. "The Lumière project: Bayesian user modeling for inferring the goals and needs of software users." In: *Conference on Uncertainty in Artificial Intelligence (UAI)*. 1998.

[68] B. Howe, A. Key, D. Perry, and C. Aragon. "VizDeck: a card game metaphor for fast visual data exploration." In: *ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA)*. 2012.

[69] *International Electrotechnical Commission (IEC) Standard 61850 Power Utility Automation*. `http://www.iec.ch/smartgrid/standards`.

[70] T. Jankun-Kelly, K. Ma, and M. Gertz. “A model and framework for visualization exploration.” In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2007).

[71] D. Jannach, M. Jugovac, and L. Lerche. “Adaptive recommendation-based modeling support for data analysis workflows.” In: *ACM Conference on Intelligent User Interfaces (IUI)* (2015).

[72] B. Jansen, A. Spink, J. Bateman, and T. Saracevic. “Real life information retrieval: a study of user queries on the web.” In: *SIGIR Forum* (1998).

[73] D. Jeong, W. Dou, H. Lipford, F. Stukes, R. Chang, and W. Ribarsky. “Evaluating the relationship between user interaction and financial visual analysis.” In: *IEEE Visual Analytics Science & Technology (VAST)*. 2008.

[74] S. Kaasten, S. Greenberg, and C. Edwards. “How people recognise previously seen Web pages from titles, URLs and thumbnails.” In: *People and Computers XVI-Memorable Yet Invisible*. Springer, 2002, pp. 247–265.

[75] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. “Enterprise data analysis and visualization: an interview study.” In: *IEEE Visual Analytics Science & Technology (VAST)*. 2012.

[76] S. Kandel, R. Parikh, A. Paepcke, J. Hellerstein, and J. Heer. “Profiler: integrated statistical analysis and visualization for data quality assessment.” In: *ACM Conference on Advanced Visual Interfaces (AVI)*. 2012.

[77] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. “Wrangler: interactive visual specification of data transformation scripts.” In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 2011.

[78] Y. Kang, C. Gorg, and J. Stasko. “Evaluating visual analytics systems for investigative analysis: deriving design principles from a case study.” In: *IEEE Visual Analytics Science & Technology (VAST)*. 2009.

[79] Y. Kang and J. Stasko. “Characterizing the intelligence analysis process: Informing visual analytics design through a longitudinal field study.” In: *IEEE Visual Analytics Science & Technology (VAST)*. 2011.

[80] D. Keim. “Information visualization and visual data mining.” In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2002).

[81] A. Key, B. Howe, D. Perry, and C. Aragon. “VizDeck: self-organizing dashboards for visual analytics.” In: *ACM International Conference on Management of Data (SIGMOD)*. 2012.

[82] R. Khemiri and F. Bentayeb. “Interactive query recommendation assistant.” In: *IEEE Workshop Database and Expert Systems Applications (DEXA)*. 2012.

[83] N. Khoussainova, Y. Kwon, W. Liao, M. Balazinska, W. Gatterbauer, and D. Suciu. "Session-based browsing for more effective query reuse." In: *International Conference on Scientific and Statistical Database Management*. Springer. 2011.

[84] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. "SnipSuggest: context-aware autocompletion for SQL." In: *International Conference on Very Large Databases (VLDB)* (2010).

[85] D. Klein. "A forensic analysis of a distributed two-stage web-based spam attack." In: *USENIX Conference on System Administration (LISA)*. 2006.

[86] S. Klemmer, M. Thomsen, E. Phelps-Goodman, R. Lee, and J. Landay. "Where do web sites come from?: capturing and interacting with design history." In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 2002.

[87] A. Kobsa. "An empirical comparison of three commercial information visualization systems." In: *IEEE Information Visualization Conference (InfoVis)*. 2001.

[88] D. Koop, C. Scheidegger, S. Callahan, J. Freire, and C. Silva. "VisComplete: automating suggestions for visualization pipelines." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)*. 2008.

[89] D. Kurlander and S. Feiner. "Editable graphical histories." In: *VL*. Citeseer. 1988, pp. 127–134.

[90] A. Laender, B. Ribeiro-Neto, A. da Silva, and J. Teixeira. "A brief survey of web data extraction tools." In: *SIGMOD Record* (2002).

[91] N. Lao, J. Wen, W. Ma, and Y. Wang. "Combining high level symptom descriptions and low level state information for configuration fault diagnosis." In: *USENIX Conference on System Administration (LISA)*. 2004.

[92] J. Lee. "A systems and process model for data exploration." PhD thesis. University of Massachusetts Lowell, 1998.

[93] J. Lee and G. Grinstein. "An architecture for retaining and analyzing visual explorations of databases." In: *IEEE Conference on Visualization (VIS)*. 1995.

[94] Z. Liu and J. Heer. "The effects of interactive latency on exploratory visual analysis." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2014).

[95] J. Lou, Q. Fu, Y. Wang, and J. Li. "Mining dependency in distributed systems through unstructured logs analysis." In: *SIGOPS Operating System Review* (2010).

[96] L. van der Maaten and G. Hinton. "Visualizing high-dimensional data using t-SNE." In: *Journal of Machine Learning Research (JMLR)* (2008).

[97] J. Mackinlay. "Automating the design of graphical presentations of relational information." In: *ACM Transactions on Graphics (TOG)* (1986).

[98] J. Mackinlay, P. Hanrahan, and C. Stolte. "Show Me: automatic presentation for visual analysis." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2007).

[99] A. Makanju, A. Zincir-Heywood, and E. Milios. "Clustering event logs using iterative partitioning." In: *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2009.

[100] S. Mangiafico. *Summary and Analysis of Extension Program Evaluation in R*. `http://rcompanion.org/handbook/index.html`. Rutgers Cooperative Extension, 2016.

[101] P. Marcel and E. Negre. "A survey of query recommendation techniques for data warehouse exploration." In: *EDA*. 2011, pp. 119–134.

[102] R. Marmorstein and P. Kearns. "Firewall analysis with policy-based host classification." In: *USENIX Conference on System Administration (LISA)*. 2006.

[103] C. Meng, M. Yasue, A. Imamiya, and X. Mao. "Visualizing histories for selective undo and redo." In: *IEEE Asia Pacific Computer Human Interaction*. 1998.

[104] B. Mittman and W. Dominick. "Developing monitoring techniques for an on-line information retrieval system." In: *Information Storage and Retrieval*. 1973.

[105] A. Moere, M. Tomitsch, C. Wimmer, B. Christoph, and T. Grechenig. "Evaluating the effect of style in information visualization." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2012).

[106] B. Mutlu, E. Veas, C. Trattner, and V. Sabol. "VizRec: a two-stage recommender system for personalized visualizations." In: *ACM Conference on Intelligent User Interfaces (IUI)*. 2015.

[107] B. Myers and D. Kosbie. "Reusable hierarchical command objects." In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 1996.

[108] K. Noriaki, M. Takeya, and H. Miyoshi. "Semantic log analysis based on a user query behavior model." In: *IEEE International Conference on Data Mining (ICDM)*. 2003.

[109] C. North, P. Saraiya, and K. Duca. "A comparison of benchmark task and insight evaluation methods for information visualization." In: *Information Visualization* (2011).

[110] Chris North. "Toward measuring visualization insight." In: *IEEE Computer Graphics and Applications* (2006).

[111] S. North, C. Scheidegger, S. Urbanek, and G. Woodhull. "Collaborative visual analysis with RCloud." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2015).

[112] A. Oliner, A. Ganapathi, and W. Xu. "Advances and challenges in log analysis." In: *ACM Queue* (2011).

[113] A. Oliner, A. Kulkarni, and A. Aiken. "Using correlated surprise to infer shared influence." In: *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2010.

[114] A. Oliner and J. Stearley. "What supercomputers say: a study of five system logs." In: *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2007.

[115] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. "Pig Latin: a not-so-foreign language for data processing." In: *ACM International Conference on Management of Data (SIGMOD)*. 2008.

[116] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. "Characteristics of internet background radiation." In: *ACM SIGCOMM Internet Measurement Conference (IMC)*. 2004.

[117] C. Papadopoulos, I. Gutenko, and A. Kaufman. "VEEVVIE: visual explorer for empirical visualization, VR and interaction experiments." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2016).

[118] A. Parameswaran, N. Polyzotis, and H. Garcia-Molina. "SeeDB: visualizing database queries efficiently." In: *International Conference on Very Large Databases (VLDB)*. 2013.

[119] F. Paterno. "Task models in interactive software systems." In: *Handbook of Software Engineering and Knowledge*. 2001.

[120] A. Perer and B. Shneiderman. "Systematic yet flexible discovery: guiding domain experts through exploratory data analysis." In: *ACM International Conference on Intelligent User Interfaces (IUI)*. 2008.

[121] L. Pevzner and M. Hearst. "A critique and improvement of an evaluation metric for text segmentation." In: *Association for Computational Linguistics (ACL)* (1994).

[122] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. "Interpreting the data: parallel analysis with Sawzall." In: *Scientific Programming Journal* (2005).

[123] E. Pinheiro, W. Weber, and L. Barroso. "Failure trends in a large disk drive population." In: *USENIX Conference on File and Storage Technologies (FAST)*. 2007.

[124] P. Pirolli and S. Card. "The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis." In: *Proceedings of international conference on intelligence analysis*. 2005.

[125] B. Poblete and R. Baeza-Yates. "Query-sets: using implicit feedback and query patterns to organize web documents." In: *ACM International Conference on World Wide Web (WWW)*. 2008.

[126] F. Ramsey and D. Schafer. *The statistical sleuth: a course in methods of data analysis*. Cengage Learning, 2012.

[127] K. Reda, A. Johnson, J. Leigh, and M. Papka. "Evaluating user behavior and strategy during visual exploration." In: *ACM Beyond Time And Errors: Novel Evaluation Methods For Visualization Workshop (BELIV)*. 2014.

[128] S. Reddy, Y. Gal, and S. Shieber. "Recognition of users' activities using constraint satisfaction." In: *ACM Conference on User Modeling, Adaptation, and Personalization (UMAP)*. 2009.

[129] Wolfram Research. *Wolfram Predictive Interface.* `http://reference.wolfram.com/mathematica/guide/WolframPredictiveInterface.html`. 2016.

[130] M. Richardson. "Learning about the world through long-term query logs." In: *ACM Transactions on the Web (TWEB)* (2008).

[131] S. Roth and J. Mattis. "Automating the presentation of information." In: *IEEE Conference on Artificial Intelligence Applications*. 1991.

[132] S. Roth, J. Kolojejchick, J. Mattis, and J. Goldstein. "Interactive graphic design using automatic presentation knowledge." In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 1994.

[133] D. Russell, M. Stefik, P. Pirolli, and S. Card. "The cost structure of sensemaking." In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 1993.

[134] P. Santos and A. Badre. "Automatic chunk detection in human-computer interaction." In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 1994.

[135] P. Saraiya, C. North, and K. Duca. "An insight-based methodology for evaluating bioinformatics visualizations." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2005).

[136] P. Saraiya, C. North, V. Lam, and K. Duca. "An insight-based longitudinal study of visual analytics." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2006).

[137] A. Sarvghad and M. Tory. "Exploiting analysis history to support collaborative data analysis." In: *Proceedings of the 41st Graphics Interface Conference*. Canadian Information Processing Society. 2015.

[138] A. Sarvghad, M. Tory, and N. Mahyar. "Visualizing dimension coverage to support exploratory analysis." In: 2017.

[139] M. Schiff. "Designing graphic presentations from first principles." PhD thesis. University of California, Berkeley, 1998.

[140] H. Schulz, T. Nocke, M. Heitzler, and H. Schumann. "A design space of visualization tasks." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)*. 2013.

[141] T. Sellam and M. Kersten. "Fast cartography for data explorers." In: *International Conference on Very Large Databases (VLDB)* (2013).

[142] *Sensor Model Language (SensorML).* `http://www.opengeospatial.org/standards/sensorml`.

[143] B. Shneiderman. "The eyes have it: a task by data type taxonomy for information visualizations." In: *IEEE Symposium on Visual Languages*. 1996.

[144] B. Shneiderman and C. Plaisant. "Strategies for evaluating information visualization tools." In: *ACM Beyond Time And Errors: Novel Evaluation Methods For Visualization Workshop (BELIV)*. 2006.

[145] B. Sigelman, L. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag. *Dapper, a large-scale distributed systems tracing infrastructure.* Tech. rep. Google, Inc., 2010.

[146] F. Silvestri. "Mining query logs: turning search usage data into knowledge." In: *Foundations and Trends in Information Retrieval* (2010).

[147] *Sloan Digital Sky Survey.* `http://www.sdss.org/dr8/`.

[148] R. Spencer. "Information Visualization: Design for Interaction." In: *Person Education* (2007).

[149] R. Springmeyer, M. Blattner, and N. Max. "A characterization of the scientific data analysis process." In: *IEEE Conference on Visualization (VIS).* 1992.

[150] J. Srivastava, R. Cooley, M. Deshpande, and P. Tan. "Web usage mining: discovery and applications of usage patterns from web data." In: *SIGKDD Explorations Newsletter* (2000).

[151] C. Stolte, D. Tang, and P. Hanrahan. "Polaris: A system for query, analysis, and visualization of multidimensional relational databases." In: *IEEE Visualization and Computer Graphics* (2002).

[152] V. Thost, K. Voigt, and D. Schuster. "Query matching for report recommendation." In: *ACM Conference on Information and Knowledge Management (CIKM).* 2013.

[153] D. Toker, C. Conati, G. Carenini, and M. Haraty. "Towards adaptive information visualization: on the influence of user characteristics." In: *ACM Conference on User Modeling, Adaptation, and Personalization (UMAP).* 2011.

[154] J. Tolle. "Performance measurement and evaluation of online information systems." In: *ACM Conference on Computer Science (CSC).* 1985.

[155] E. Tufte and G. Howard. *The Visual Display of Quantitative Information.* Graphics Press, 1983.

[156] L. Tweedie. "Characterizing interactive externalizations." In: *ACM Conference on Human Factors in Computing Systems (CHI).* 1997.

[157] C. Upson, T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. Van Dam. "The application visualization system: a computational environment for scientific visualization." In: *IEEE Computer Graphics and Applications* (1989).

[158] N. Velasquez, S. Weisband, and A. Durcikova. "Designing tools for system administrators: an empirical test of the integrated user satisfaction model." In: *USENIX Conference on System Administration (LISA).* 2008.

[159] F. Viegas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. "Many Eyes: a site for visualization at Internet scale." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2007).

[160] *Visual Analysis Best Practices*. Tech. rep. Tableau Software, Inc., 2014.

[161] M. Voigt, M. Franke, and K. Meißner. "Capturing and reusing empirical visualization knowledge." In: *UMAP Workshops 2013* (2013).

[162] M. Voigt, M. Franke, and K. Meissner. "Using expert and empirical knowledge for context-aware recommendation of visualization components." In: *International Journal On Advances in Life Sciences* (2013).

[163] M. Voigt, S. Pietschmann, L. Grammel, and K. Meißner. "Context-aware recommendation of visualization components." In: *Proceedings of the 4th International Conference on Information, Process, and Knowledge Management*. 2012.

[164] M. Ward. *Jack Ma: This is what to study if you want a high-paying job in the future.* June 2017. URL: `\url{http://www.cnbc.com/2017/06/21/jack-ma-this-is-what-to-study-if-you-want-a-good-job-in-the-future.html}`.

[165] M. Ward and H. Yang. "Interaction spaces in data and information visualization." In: *Joint EuroGraphics / IEEE TVCG Conference (VISSYM)*. 2004.

[166] S. Waterson, J. Hong, T. Sohn, J. Landay, J. Heer, and T. Matthews. "What did they do? understanding clickstreams with the WebQuilt visualization system." In: *ACM Conference on Advanced Visual Interfaces (AVI)*. 2002.

[167] L. Wilkinson. *The grammar of graphics*. Springer Science & Business Media, 2006.

[168] W. Willett, S. Ginosar, A. Steinitz, B. Hartmann, and M. Agrawala. "Identifying redundancy and exposing provenance in crowdsourced data analysis." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)*. 2013.

[169] A. Woodruff, A. Faulring, R. Rosenholtz, J. Morrsion, and P. Pirolli. "Using thumbnails to search the Web." In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 2001.

[170] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan. "Experience mining Google's production console logs." In: *OSDI Workshop on Managing Systems via Log Analysis and Machine Learning Techniques (SLAML)*. 2010.

[171] Yanpei Y. Chen, K. Srinivasan, G. Goodson, and R. Katz. "Design implications for enterprise storage systems via multi-dimensional trace analysis." In: *ACM Symposium on Operating Systems Principles (SOSP)*. 2011.

[172] H. Yang, Y. Li, and M. Zhou. "Understand users comprehension and preferences for composing information visualizations." In: *ACM Transactions on Computer-Human Interaction (TOCHI)* (2014).

[173] X. Yang, C. Procopiuc, and D. Srivastava. "Recommending join queries via query log analysis." In: *IEEE International Conference on Data Engineering (ICDE)*. 2009.

[174] J. Yi, Y. Kang, and J. Stasko. "Toward a deeper understanding of the role of interaction in information visualization." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2007).

[175] J. Yi, Y. Kang, J. Stasko, and J. Jacko. “Understanding and characterizing insights: how do people gain insights using information visualization?” In: *ACM Beyond Time And Errors: Novel Evaluation Methods For Visualization Workshop (BELIV)*. 2008.

[176] D. Yuan, S. Park, and Y. Zhou. “Characterizing logging practices in open-source software.” In: *IEEE International Conference on Software Engineering (ICSE)*. 2012.

[177] X. Zhang, H. Brown, and A. Shankar. “Data-driven personas: constructing archetypal users with clickstreams and user telemetry.” In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 2016.

[178] M. Zhou and S. Feiner. “Visual task characterization for automated visual discourse synthesis.” In: *ACM Conference on Human Factors in Computing Systems (CHI)*. 1998.