Better Logging to Improve Interactive Data Analysis Tools

S. Alspaugh
University of California, Berkeley
alspaugh@eecs.berkeley.edu

Archana Ganapathi
Splunk, Inc.
aganapathi@splunk.com

Marti A. Hearst
University of California, Berkeley
hearst@berkeley.edu

Randy Katz
University of California, Berkeley
randy@eecs.berkeley.edu

Abstract

Interactive data analysis applications have become critical tools for making sense of our world. We present a set of recommendations to improve the quality and quantity of user activity data logged from interactive data analysis systems. Such data is invaluable for improving our understanding of the data exploration process, for implementing intelligent user interfaces, for evaluating data mining and visualization techniques, and for characterizing how the broader ecosystem of data analysis tools are used in practice.

Currently, 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. As a result, researchers have to rely on labor-intensive techniques for extracting useful information from low-level event streams, or on collecting data through observation, interviews, experiments, and case studies.

We present recommendations – derived from personal experience as well as examples from the literature – for logging user activity in interactive data analysis tools, to ensure that better information is collected, and ultimately, to enhance human problem-solving abilities and speed the pace of discovery. We illustrate these recommendations using examples from three widely-used but distinct systems for analyzing data: Tableau, an interactive visualization product, Excel, a spreadsheet application, and Splunk, an enterprise log management and analysis platform.

1. INTRODUCTION

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 [12, 16, 28]. As Horvitz et al. note in their paper on Bayesian user modeling, "... 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." [16] 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 [14].

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 [4, 6, 17, 33] or on observational studies using manually-recorded and synthesized information that is hard to share and compare [18, 19]. This lack of automatically logged activity records hinders research into improving tools for data exploration and analysis.

Currently, to understand what users are doing, build user models to improve interfaces, yield predictions about user actions, and make recommendations to users, researchers have few options. One option is to take great pains to extract high-level information out of low-level event logs [12, 14, 16, 28]. 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 their tools [1, 17, 22, 27].

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 [3, 11, 13, 32]. As Hilbert and Redmiles conclude, "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 ... can be expected to yield meaningful results." [14]

This position paper encourages better practices for logging data to enable studying user behavior in interactive data and visualization systems. Our recommendations are based primarily on our personal experiences trying to build user models using traces from an enterprise-scale log analysis system, but also on a review of the literature and conversations with industry personnel. Our recommendations can be summarized as follows:

- Design to capture high-level user actions.
- Capture provenance of all events.
- Observe intermediate user actions.
- Obtain the analyzed data's metadata and statistics.
- Work towards log standardization.
- Collect user goals and feedback.

In Section 2 we first provide motivating examples of research and applications that could be enabled if better user activity data were logged from interactive data analysis systems. We then give our recommendations for collecting this improved data in Section 3. Section 4 discusses ramifications and other issues.

2. WHY DO WE NEED BETTER LOGGING?

This section motivates the need for better logging of interactive data analysis systems: characterizing the exploration process, implementing intelligent user interfaces, evaluating analysis tools and interfaces, and understanding the analysis ecosystem as a whole. These purposes are not only of interest to researchers who wish to understand these topics, but also to industry practitioners, who can use this information to design their products to make them better suited to their users. The section concludes with examples from the related area of web behavior mining, which is further along and could hold useful lessons.

Characterizing the exploration process: Two interesting theoretical models of the data exploration process have been put forward recently that would benefit greatly from better logging as proposed here.

(1) De Bie and Spyropoulou propose a formalization to unify the concept of interestingness and help automate data exploration across a range of data mining techniques [8]. In their formalization, users express interests and beliefs about the data in terms of mathematical patterns and probability distributions. To put this formalization into practical use, rather than asking end-users to specify these directly, data exploration tool developers will likely want to determine the beliefs and patterns users find useful for a particular domain and then expose those to the user in a more easily interpretable form. Doing so would require detailed and annotated records of exploratory activities in a wide variety of scenarios.

(2) As another example, Perer and Shneiderman propose a framework, called SYF (Systematic, Yet Flexible), for guiding users through data exploration [27]. It operates within interactive analysis and visualization interfaces, guiding users by providing an overview of recommended analysis steps, suggesting unexplored states, and allowing users to annotate and share a record of their activities. To implement SYF within a given tool, developers must register their recommended exploration steps with the SYF framework. To derive these systematic steps, Perer and Shneiderman suggest that developers try "[i]nterviewing analysts, reviewing current software approaches, and tabulating techniques common in research publications.". An additional useful approach for establishing these steps would be to mine detailed activity records from data analysis and visualization systems.

Implementing intelligent user interfaces: SYF is one example of an intelligent user interface. These assist the user by offloading some of the complexity in working with the tool at hand, often by automated means. Other examples include adaptive or adaptable [5, 24], predictive [29], and mixed-initiative interfaces [15], as well as automated user assistants [16, 23]. Automated interfaces often rely on statistical models of user behavior and thus require accurate accounting user actions at a level that corresponds to the variables being modeled.

For example, Wrangler has a mixed-initiative interface that makes suggestions to help users clean their data based on frequencies of user actions [17]. Wrangler was originally based on a transformation language with a small number of operators. To identify this list of transforms and pair them with interface gestures, the authors were able to capitalize on their extensive first-hand experience, as well as prior work on languages for data cleaning.

However, for data exploration rather than data cleaning, it is not clear what set of transforms and visualizations should be supported. Previous work has relied on author intuition and experience with particular situations to determine what actions to support [4, 6, 33]. However, these could be better determined by having detailed activity records from data exploration and visualization tools with direct-manipulation interfaces, logged at an appropriate level of granularity [12].

Evaluating analysis tools and interfaces: More generally, researchers and practitioners evaluate interfaces to understand user behavior, performance, thoughts, and experience, compare design alternatives, compute usability metrics, and certify conformance with standards [14]. To achieve these goals using events logged from current UI systems, researchers have devised a wide range of techniques: synchronizing data gathered from different sources, transforming, comparing, summarizing, and visualizing event streams, and abstracting low-level log events into high-level modeled events.

An alternative to these automated techniques is to perform carefully controlled laboratory evaluations or focused long-term studies of specific tools in isolation [18, 31, 20, 26, 19]. These usually involve watching videos of study subjects performing a task, interviewing subjects about their experience, and evaluating how well they performed the task.

While this research is valuable, some drawbacks of these techniques are that they don't scale well, they generate results that are not amenable to comparison or combination with data from other studies, and the process of recording the data is too open to subjective interpretation. High-quality automatically logged interaction data would circumvent each of these problems, although at the expense of missing the big picture that these techniques provide.

Understanding the analysis ecosystem: In addition to improving upon individual tools and interfaces, developers and researchers want to understand the entire data analysis pipeline. In practice, users leverage multiple tools to explore and visualize their data depending on their needs. For example, a data scientist might use Hadoop and R for statistical work. A product manager might create their visualizations in Tableau using web analytics reports generated in Splunk. A customer analyst might extract numbers from Salesforce to crunch in Excel. To get a complete picture of each of these user's exploration and visualization needs, it would help greatly to be able to track their activities across each of the tools they use. However, researchers are currently limited to gathering cross-tool data via long-term time-intensive interviews with industry practitioners [25, 30].

Inspirations from other domains: When it comes to automating data collection about user behavior and optimizing interfaces in response, work in mining website interaction data and search engine clickstreams may help point the way forward. Much effort has gone into designing tools to collect extensive data about web users and analyze it in increasingly sophisticated ways. A full survey of such work is beyond the scope of this paper; here we provide a few examples for illustration. Researchers have designed advanced and unobtrusive tracking software that can be implemented using standard web technologies [3]. Information gleaned from this type of data can be used to infer user goals to determine, for instance, if the user is interested in purchasing a product or merely researching it [13]. Such data can also

be used as implicit feedback on the presented interface, for example, regarding the quality of a ranking returned by a search engine [11].
We hope that in the near future, interactive data analysis and visualization users will benefit from similar efforts. We contribute to this vision through our detailed recommendations regarding what data to collect from interactive data analysis and visualization tools.

3. RECOMMENDATIONS

In this section we detail a set of proposals for collecting data from interactive data analysis and visualization systems to support the research and applications described in Section 2. We illustrate our proposals with examples from three systems: Tableau, an interactive data visualization product, Excel, a spreadsheet application, and Splunk, an enterprise log management and analysis platform.

3.1 Overview of logging basics

Here we review the basic types of information that should be logged.
Events: Units of information in a log are often called events, even if they weren't generated from an event-driven program. However, GUIs and other interactive programs are usually event-driven.
An event in a log is some piece of information that is recorded any time work of interest is run on the system. Work of interest could include functions called, queries run, GUI handlers triggered, threads executed, and so on. Exactly what information is logged for each event and the format it appears in varies widely – it may include information such as function parameters, execution duration, caller, and source code location. For example, an event logged by Tableau is shown in Table 1. Such events are typically logged for debugging and performance monitoring purposes. Later we discuss specifically what types of events and associated information should be logged for user modeling.
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 reliably accurate reflections of when an event truly occurred. This is usually not a problem when dealing with logs from a single machine but can be extremely challenging to deal with in a distributed setting. A discussion of how to deal with this problem is beyond the scope of this paper; we refer the reader to other work [14].
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.

3.2 Design to capture high-level user actions

As 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 [1] 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" [21]. In their paper describing this functionality, the authors 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 Me Alternatives*. These commands are implemented with the same code and the log entry is generated when the command is successfully executed." 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.
In other cases, logging user-level actions may be more difficult. For example, the authors of the Lumière project to build an automated assistant in Excel found transforming system events into modeled events to be challenging [16]. To establish the link between low-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.
Figure 1 and Figure 2 illustrate this problem graphically, using Excel as an example. Figure 1 shows the steps required to perform k-means clustering on data in Excel from a user's perspective [10]. The user may be doing this in order 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 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 2. In practice, the information captured tends to be low-level system events – the lowest level of Figure 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.
HARVEST is a visual analytics system specifically designed to deal with this problem and capture the provenance of "in-

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

Table 1: Example of an event logged by Tableau. This event records that a function was called, giving its location in the source.

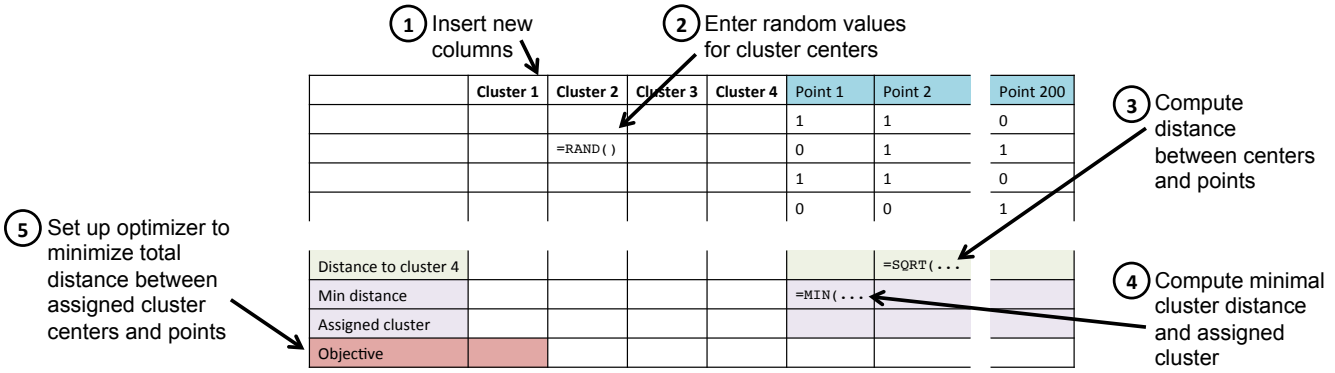

Figure 1: Shown above are the steps a user takes to perform k-means clustering on their data in Excel. Ideally, the interface would be designed to easily capture the user's task, which is clustering. Barring that, the task may have to be inferred from the sequence of user actions, shown here.

sights" [12]. It does this by exposing 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. 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 [28]. 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.

3.3 Capture provenance of all events

Information about the point in the application where an event occured and how it was 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 2. 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 versus selecting a scatter plot from the *Show Me Alternatives* menu, it should be recorded not only that the user created a scatter plot but also how the user created the scatter plot i.e., the provenance.

Even then, 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-

```
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 2: Example of an event logged by Splunk. A query is highlighted in bold. Queries represent useful records of user activity. These queries are written in the Splunk query language, modeled after UNIX pipes and utilities.

based GUI (i.e., a bot). But this should be rare if the system is designed to encourage correct and easy-to-track use, such as providing programmatic access to its analysis capabilities via an API, if that is what users need.

3.4 Observe 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, and
- 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, however, 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 [2].

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 Obtain analyzed data's metadata & stats

If possible, with the user's permission, metadata and statistics about the data over which the user is operating should be 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

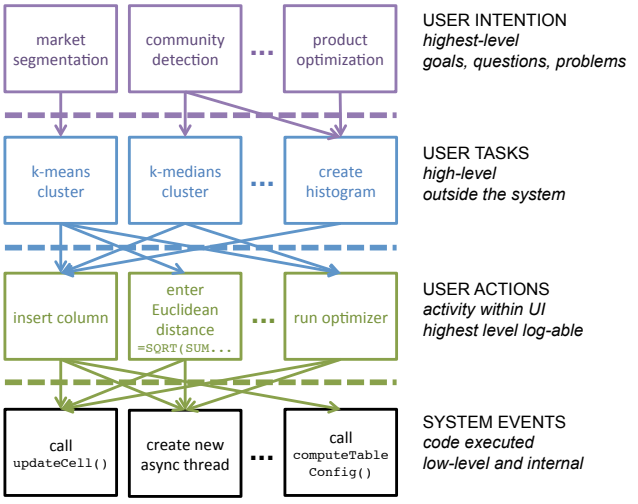

Figure 2: Users intentions motivate their actions, but may be hard to know (top row). When trying to understand user behavior especially with respect to data exploration and visualization, we are often interested in the high-level task the user is performing (second row). In the best case, we can log 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).

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 [21]

3.6 Work towards log standardization

Effort should be made where possible to log information that facilitates combination with and comparison to other systems. This would allow researchers and developers to understand what functionality is missing from existing tools, better characterize portions of the analysis pipeline that are cur-

rently 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 [9]. Examples of standard schemas in other domains that may serve as inspiration includes IEC61850 for electric grids and SensorML for sensor data.

3.7 Collect user goals and feedback

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 [7]. Where possible, information about goals, expertise, and other relevant context could be solicited from the user. However, if this solicitation requires the user do additional work that does not benefit them, it is highly unlikely to be successful. 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. The information we recommend collecting in the rest of this paper could facilitate such research.

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 [27]. 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.

4. CONCLUSION

We conclude our recommendations 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." [16] 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 [2].

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." [14] 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 Section 2 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 paper. 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.

5. REFERENCES

[1] Visual analysis best practices. Technical report, Tableau Software, 2014.

[2] Sara Alspaugh, Beidi Chen, Jessica Lin, Archana Ganapathi, Marti Hearst, and Randy Katz. Analyzing log analysis: An empirical study of user log mining. In *Conference on Large Installation System Administration (LISA)*, 2014.

[3] Richard Atterer, Monika Wnuk, and Albrecht Schmidt. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In *ACM Conference on World Wide Web (WWW)*. ACM, 2006.

[4] Abraham Bernstein et al. Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *Knowledge and Data Engineering*, 2005.

[5] Andrea Bunt, Cristina Conati, and Joanna McGrenere. Supporting interface customization using a mixed-initiative approach. In *ACM Conference on Intelligent User Interfaces (IUI)*, 2007.

[6] Stephen Casner. Task-analytic approach to the automated design of graphic presentations. *Graphics (TOG)*, 1991.

[7] Stephen Casner. Task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics (TOG)*, 1991.

[8] Tijl De Bie and Eirini Spyropoulou. A theoretical framework for exploratory data mining: Recent insights and challenges ahead. pages 612–616, 2013.

[9] Inc. Distributed Management Task Force. Common information model. `http://www.dmtf.org/standards/cim`.

[10] John Foreman. *Data Smart: Using Data Science to Transform Information into Insight*. John Wiley and Sons, Inc., 2014.

[11] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems (TOIS)*, 23(2):147–168, 2005.

[12] David Gotz et al. Characterizing users' visual analytic activity for insight provenance. *Information Visualization*, 2009.

[13] Qi Guo and Eugene 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.

[14] David M Hilbert and David F Redmiles. Extracting usability information from user interface events. In *ACM Computing Surveys (CSUR)*, 2000.

[15] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Human Factors in Computing Systems (CHI)*, 1999.

[16] Eric Horvitz et al. The Lumière project: Bayesian user modeling for inferring the goals and needs of software users. In *Conference on Uncertainty in Artificial Intelligence*, 1998.

[17] Sean Kandel et al. Wrangler: Interactive visual specification of data transformation scripts. In *Human Factors in Computing Systems (CHI)*, 2011.

[18] Youn Ah Kang et al. Evaluating visual analytics systems for investigative analysis: Deriving design principles from a case study. In *Visual Analytics Science & Technology (VAST)*, 2009.

[19] Youn Ah Kang et al. Characterizing the intelligence analysis process: Informing visual analytics design through a longitudinal field study. In *Visual Analytics Science & Technology (VAST)*, 2011.

[20] H Lam, E Bertini, P Isenberg, C Plaisant, and S Carpendale. Empirical Studies in Information Visualization: Seven Scenarios. In *Transactions on Visualization and Computer Graphics (TVCG)*, pages 1520–1536, 2012.

[21] J. Mackinlay et al. Show me: Automatic presentation for visual analysis. *Visualization and Computer Graphics*, 2007.

[22] Jock Mackinlay. Automating the design of graphical presentations of relational information. *Graphics (TOG)*, 1986.

[23] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 1994.

[24] Joanna McGrenere, Ronald M Baecker, and Kellogg S Booth. An evaluation of a multiple interface design solution for bloated software. In *Conference on Human Factors in Computing Systems (CHI)*, 2002.

[25] Sean Kand others. Enterprise data analysis and visualization: An interview study. In *Visual Analytics Science & Technology (VAST)*, 2012.

[26] Adam Perer and Ben Shneiderman. Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *Conference on Human Factors in Computing Systems (CHI)*, 2008.

[27] Adam Perer and Ben Shneiderman. Systematic yet flexible discovery: guiding domain experts through exploratory data analysis. In *Conference on Intelligent User Interfaces (IUI)*, 2008.

[28] Swapna Reddy, Ya'akov Gal, and Stuart Shieber. Recognition of users' activities using constraint satisfaction. In *Conference on User Modeling, Adaptation, and Personalization (UMAP)*, 2009.

[29] Wolfram Research. Wolfram predictive interface. `http://reference.wolfram.com/mathematica/guide/WolframPredictiveInterface.html`.

[30] Michael Sedlmair et al. Evaluating information visualization in large companies: challenges, experiences and recommendations. In *3rd BELIV Workshop (BELIV)*, 2010.

[31] Ben Shneiderman and Catherine Plaisant. Strategies for evaluating information visualization tools. In *Beyond Time And Errors: Novel Evaluation Methods For Visualization Workshop (BELIV)*, 2006.

[32] Jaideep Srivastava et al. Web usage mining: discovery and applications of usage patterns from web data. *SIGKDD Explorations Newsletter*, 2000.

[33] Robert St. Amant et al. Intelligent support for exploratory data analysis. *Journal of Computational and Graphical Statistics*, 1998.