How Progressive Visualizations Affect Exploratory Analysis

Emanuel Zgraggen, Alex Galakatos, Andrew Crotty, Jean-Daniel Fekete, Tim Kraska

How Progressive Visualizations Affect Exploratory Analysis

Emanuel Zgraggen, *Student Member, IEEE,* Alex Galakatos, Andrew Crotty, Jean-Daniel Fekete, *Senior Member, IEEE,* and Tim Kraska

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

Index Terms—Exploratory analysis, interactive visualization, progressive visualization, scalability, insight-based evaluation

1 Introduction

The literature [1], [2], [3], [4] often states that a delay of one second is the upper bound for computer responses after which users lose focus on their current train of thought. In order to ensure a highly interactive environment for data analysis, a visual data exploration system should therefore strive to present some actionable and understandable artifact for possible query over any dataset within a one second threshold. It is important to note that this artifact does not need to be the complete or most accurate answer, but it should be an answer that allows users to keep their attention on their current task.

Traditionally, visual data exploration systems employ a strategy whereby user-issued queries are offloaded to a database management system (DBMS), and the results are displayed once the complete answer is computed. We call this the *blocking* approach: a user's current train of thought is blocked until the query result is fully computed. Even with modern hardware and state-of-the-art DBMSs that can process millions of data points per second, this traditional approach suffers from the basic issue that some datasets will still be too "big" to yield results for interactive user queries within one second.

A wide variety of strategies, including precomputation, prefetching, sampling, and progressive computation, have been proposed to overcome this fundamental limitation. However, each of these approaches comes with its own set of advantages and challenges. In particular, *progressive computation*, where data is processed incrementally in small chunks, offers an interesting tradeoff between result accuracy and computation speed. Moreover, unlike many of the other approaches, progressive computation also provides a number of natural opportunities to incorporate user feedback and computational steering. The research community has recently regained interest in progressive computation, attempting to analyze and exploit some of the peculiarities of this approach [5], [6], [7], [8], [9], [10]. However, the effects of progressive computation—and progressive visualization—on user behavior and knowledge discovery in exploratory settings have not been studied in detail.

The aim of this work is to investigate how progressive visualizations affect users in exploratory settings. To this end, we design and conduct an experiment where we compare three different visualization conditions: (1) instantaneous, (2) blocking, and (3) progressive. The instantaneous visualization condition acts as a stand-in for hypothetical systems where all queries, independent of the dataset size, return and display accurate results within a strict latency constraint. On the other hand, blocking visualizations simulate traditional systems where results are displayed only after the full dataset has been processed. Blocking visualizations are limited by the throughput of the underlying computation engine; that is, the wait time increases proportionally to the size of the dataset. Finally, progressive visualizations, where data is processed incrementally in small chunks, present approximate results to the user at different points during the computation. To compare each of these strategies, we capture interaction logs and verbal data from the participants using a think-aloud protocol. We then extract several knowledge discovery and user activity metrics from these recordings, and we analyze these metrics to test how progressive visualizations compare to the alternatives. For

- *E. Zgraggen, A. Galakatos, A. Crotty and T. Kraska are with the Computer Science Department, Brown University, Providence, RI, 02912. {emanuel_zgraggen, alexander_galakatos, andrew_crotty, tim_kraska}@brown.edu*
- *Jean-Daniel Fekete is with INRIA, Orsay, FR 91405. jean-daniel.fekete@inria.fr*

Manuscript received April 14, 2016; revised April 14, 2016.

the purpose of our study, we picked simple uncertainty visualizations and update strategies and are specifically not testing different variations in those.

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

2 Related Work

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

2.1 Big Data Visual Analytics

Visual data analysis, or the task of gaining insights from a dataset through visualizations, is an interactive and iterative process where users must frequently switch between a wide range of distinct but interrelated tasks. While the specific set of tasks that recur in visual data analysis, as well as the tools that support them, are relatively well understood [11], [12], the constantly increasing volume of data has forced the interaction paradigm away from interactive approaches back to large-scale batch processing [13].

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

The research community has proposed several approaches to address this problem, each with its own set of advantages and challenges. We introduce a simple use case to illustrate these different approaches. Imagine a visual data exploration system that processes tabular data, allowing users to create simple aggregated histograms over this data by selecting different attributes. Furthermore, histograms are linked together, where selections in one visualization trigger filtering operations in others, such as in GraphTrail [14], PanoramicData [15], and Vizdom [16]. Figure 1 shows an example where the user only wants to see the histogram of income for a specific age range.

Conceptually, the simplest way to implement a data exploration system to support these interactions is through a *blocking* approach. After the user requests a particular histogram, the system scans the full dataset to perform the required aggregation and displays the result only after processing all of the data points. Using this approach, the user cannot see results until after the entire dataset has been processed. Response times of blocking systems are therefore

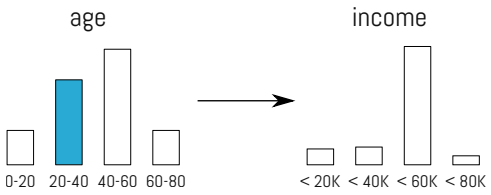


Fig. 1. Two coordinated visualizations. Selections in the left filter the data shown in the right.

proportional to the size of the dataset. Several commercially available visual analysis tools, including Tableau [17] (and its research predecessor Polaris [18]), Spotfire [19], and Microsoft Power Pivot [20], use a blocking approach.

Precomputation provides an opportunity to reduce the latencies of the blocking approach by performing some of the computation up front before a user begins exploration. The system can perform this precomputation during a loading phase and simply return a result from its cache when the user issues a request. This approach requires a substantial time and processing investment during the initial loading phase but can completely eliminate visualization latency during data exploration. However, a major drawback of precomputation is the potentially enormous number of both visualizations and exploration paths the user can take. In particular, the number of possibilities depends both on the characteristics of the data as well as the exploration tasks supported by the system. Permitting arbitrarily filtered histograms (e.g., as shown in Figure 1) drastically increases the number of required precomputations. Even with these issues, many systems have successfully used precomputation to improve the user experience. For instance, some commercial DBMSs support the creation of online analytical processing (OLAP) cubes [21], where the data is pre-aggregated along selected dimensions. Other research systems introduce improvements to these techniques using more advanced algorithms and data structures [22] or by exploiting modern hardware (e.g., GPUs) [23].

Similar to precomputation, *prefetching* incrementally precomputes results during user exploration rather than computing all possible results a priori. Prefetching approaches typically either limit the degrees of freedom given to the user or employ an intelligent oracle that predicts the user's most likely subsequent actions, such as in a map application where users can perform only pan and zoom actions. Since the user can only transition to a very limited number of possible next states (i.e., panning left/right/up/down or zooming in/out), the system can therefore use the time that a user spends examining a specific region to prefetch neighboring tiles that might be requested next. Both ForeCache [24] and Semantic Windows [25] use prefetching by exploiting locality in the user's exploratory behavior to predictably prefetch chunks of data in anticipation of subsequent. Other prefetching based systems use models to estimate the most likely action the user will perform next. These predictive models can be built using a wide variety of different information. For example, Doshi et. al. [26] propose and compare different techniques that incorporate a user's interaction history, while Ottley et. al. [27] show that certain personality traits affect user exploration strategies.

Prefetching systems need to provide a fallback strategy for cases when the prediction fails, most commonly by reverting to a blocking approach for cases when the user issues a query for which the result has not yet been prefetched. Furthermore, even with prefetching, the result might take too long to compute if the user only spends a short amount of time between interactions.

While all of the approaches discussed thus far are guaranteed to return completely accurate results for all queries, *sampling* takes a fundamentally different approach by trading speed for accuracy. Instead of computing the completely accurate result, the system uses a small sample of the data to compute the histogram and presents it to the user with some quality metric for the approximation (e.g., confidence intervals, error bars). Systems that use sampling can return results for all user queries within the specified interactivity time constraints without needing to precompute any query results. The data management community has heavily explored the development of approximate query engines. For example, BlinkDB [28] is a SQL engine that uses sampling to answer queries over large datasets and allows users to specify accuracy or response time requirements. DICE [29] similarly supports subsecond latencies for large datasets using a variety of optimizations including sampling.

Although sampling-based approaches can provide users with a quick overview of the dataset, they also introduce a completely new set of challenges. For instance, rare (but potentially important) datapoints might not be captured in the sample. Additionally, even experts sometimes have trouble interpreting statistical accuracy metrics [30], [31]. Ferreira et. al. [32] introduced specialized visualizations that mitigate some of these issues, but further research is necessary in order to apply their findings to different visualization types and analysis tasks.

Progressive systems are an extension of sampling-based approaches that incrementally compute results over increasingly larger samples in order to provide more accurate results to the user over time. This concept is well known in the graphics domain and widely used on websites to improve user experience when loading high-resolution images [33]. A down-sampled, low-resolution image is displayed quickly and replaced with the high-resolution version when fully downloaded. This concept was previously explored in the data management community [34], where most of the subsequent research focused on creating progressive versions of well known DBMS operations (e.g., joins [35], aggregations [37]). More recently, progressive approaches have gained interest among the HCI and visualization communities [5], [6], [7], [8], [10], [38]. Progressive Insights [6] is an example of a progressive system that allows the user to analyze common patterns in event sequence medical data. Fisher et. al. [5] presented *sampleAction*, a tool that simulates progressive queries over large datasets, and discuss and analyze the implications of such queries and different confidence metrics through quantitative case studies. While this approach has similar drawbacks to sampling (e.g., bad for finding outliers, does not work with ordered data, requires statistical sophistication from users), many authors [7], [8], [16], [38] advocate for its usefulness in exploratory data analysis. For example, it allows users to decide what level of accuracy they need and provide opportunities to inject user-steerability into algorithms and computations. However, the effects of this approach in terms of user performance and behavior have not yet been analyzed in detail.

2.2 Latency in Computer Systems

As many have argued [12], [39], the goal of visual data exploration systems is to operate at a rate that matches the pace of data analysts. Systems should therefore attempt to keep query response times below thresholds that will make users lose focus on their current task. A study by Liu et. al. [40] shows that latencies of 500ms have significant effects on user performance in data exploration scenarios. In other domains, including web search [41] and video games [42], even smaller latencies (300ms and 100ms, respectively) can negatively influence user performance. Frameworks proposed by Nielsen and others [1], [2], [3], [4] suggest that responses within one second allow users to stay on their current train of thought, while response times over ten seconds exceed the average attention span. We use these models to justify the different delay times used in our experiment.

3 Experimental Design

The aim of this work is to investigate how progressive visualizations influence users during exploratory data analysis, as well as how these techniques compare to blocking and instantaneous visualizations. We use a full-factorial 3 visualization conditions (blocking, instantaneous, progressive) $\times$ 2 dataset-delay conditions (6s, 12s) $\times$ 2 dataset-order conditions (123, 312) experiment. Our experiment is based on work by Liu et. al. [40] and Guo et. al. [44], which suggest using a hybrid evaluation approach that uses system logs and insight-based metrics coded from think-aloud protocols in order to analyze both (1) user interactions and (2) analysis performance. We expect that users will generate more insights per minute with instantaneous visualizations than with progressive ones ($H1$) and that users will generate more insights per minute with progressive visualizations than with blocking ones ($H2$). Furthermore, we anticipate user activity levels to be higher with instantaneous visualizations than with progressive ones ($H3$) as well as higher with progressive visualizations than with blocking ones ($H4$). This section provides a detailed description of the experimental design.

3.1 Visualization Conditions

In order to understand how progressive visualizations influence users in terms of knowledge discovery and interaction behavior, we compare them against two baseline visualization conditions: (1) blocking and (2) instantaneous. The instantaneous condition represents a hypothetical ideal scenario where the system always return query results within the time constraint regardless of dataset size. The blocking condition represents the other extreme, which is how many current visual data exploration systems operate. For blocking visualizations, query results are displayed only after the computation over the entire dataset concludes, with visualization latencies scaling with dataset size. The progressive

condition bridges the instantaneous and blocking conditions by displaying an approximate result as soon as possible and then incrementally refining the results over time. Eventually, the full and completely accurate result is displayed, as in the blocking approach, but initial approximate results are still returned as soon as possible.

Figure 2 shows a schematic illustration of the differences between these three conditions. Assuming that a given hardware platform requires x seconds to compute and display the full query result, then a system operating in the hypothetical instantaneous mode will return results immediately; a blocking system will use the full x seconds before displaying any results; and the progressive system will show inaccurate results that are incrementally refined throughout the x seconds.

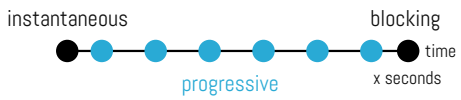


Fig. 2. Schematic time scale showing when different visualization condition display results.

3.2 Datasets

We use three datasets from different domains in our experiment. The first dataset (DS1) contains information about cars [45] (8 attributes: 7 quantitative, 1 nominal), such as "acceleration" and "horsepower." The second one (DS2) includes data about wines (7 attributes: 5 quantitative, 2 nominal), with attributes including "type", "country of origin" and several different ratings. Finally, the third one (DS3) is a subset of the 1994 US census [45] (9 attributes: 3 quantitative, 6 nominal). We use a fourth dataset (Titanic) to introduce participants to the system.

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

3.3 System

We created an experimental system specifically for our study. The UI, shown in Figure 3, consists of a list of the current dataset's attributes (a) and four visualization panels ($v1$-$v4$). Users can drag and drop attributes onto the axes of the visualization panels, and tapping on an axis cycles through different aggregation functions (e.g., count, average). Our system supports two visualization types: (1) bar charts and (2) 2D-histograms. Visualizations are always binned and never show individual data points, which allows us to fix the time required to display a visualization, even for arbitrarily large datasets. That is, the number of visual elements to render is decoupled from the size of the underlying dataset.

Selecting a bin in a visualization triggers a brushing operation in which the selected data points are highlighted in all other visualizations. In the example (Figure 3), the user selected the rightmost bar in $v2$. All other visualizations now shade bins in two different colors: blue indicating the overall amount and purple the amount of data points that corresponds to the user's selection. Bar height and rectangle area are scaled to reflect the number of data points that match the selection. A textbox (c) permits finer-grained brushing control through arbitrary Boolean statements (e.g., highlight all wines where $price < 35$ and $vintage > 2010$). Similarly, a second textbox (b) supports filtering operations through Boolean statements to select a specific subset of the data. Brushing and filtering operations require all affected visualizations to recompute their results from scratch, regardless of the current visualization conditions. For example, selecting an additional bar in $v2$, and thereby changing the currently applied brush, will force all other visualizations (i.e., $v1$, $v3$, and $v4$) to recompute. Depending on the visualization condition, $v1$, $v3$, and $v4$ will either show the results of the new brushing action instantaneously, a loading animation until the full result is available, or incrementally updated progressive visualizations. The system does not perform any form of result caching.

Upon startup, the system loads the selected dataset into memory. The system randomly shuffles the data points in order to avoid artifacts caused by the natural oder of the data and to improve convergence of progressive computations [46]. We approximate the instantaneous visualization condition by computing queries over entire dataset as quickly as possible. Initial microbenchmarks for a variety of queries over the small datasets used in the experiment yielded the following measurements: time to compute a result $\approx 100ms$ and time to render a visualization $\approx 30ms$. Although not truly instantaneous, a total delay of only $\approx 130ms$ is well below the guideline of one second, therefore allowing us to simulate the instantaneous condition. We simulate the blocking condition by artificially delaying the rendering of a visualization by the number of seconds specified through the dataset-delay factor. In other words, we synthetically prolong the time necessary to compute a result in order to simulate larger datasets. While a blocking computation is ongoing, we display a simple loading animation, but users can still interact with other visualization panels, change the ongoing computation (e.g, selecting a different subset), or replace the ongoing computation with a new query (e.g., changing an axis). Figure 4 (top) shows an example of a blocking visualization over time.

We implemented the progressive condition by processing the data in chunks of 1,000 data points at a time, with approximate results displayed after each chunk. In total, we refresh the progressive visualization 10 times, independent of the dataset-delay factor. We display the first visualization as quickly as possible, with subsequent updates appropriately delayed so that the final accurate visualization is displayed after the specified dataset-delay

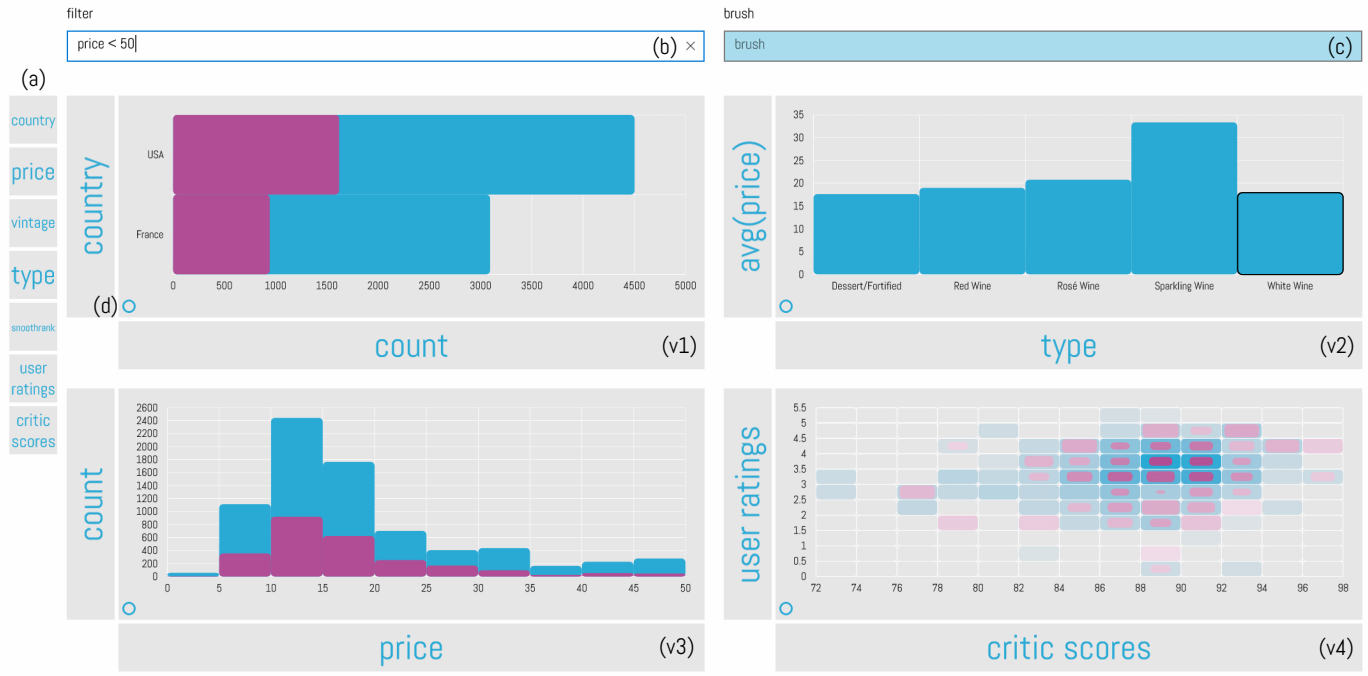


Fig. 3. Screenshot of our experimental system.

condition. Note that the initial min and max estimates might change afters seeing additional data, in which case we extend the visualization by adding bins of the same width to accommodate new incoming data. Throughout the incremental computation, we display a progress indication in the bottom left corner of a visualization (Figure 3 (d)). Progressive visualizations are augmented with error metrics indicating that the current view is only an approximation of the final result. Even though 95% confidence intervals based on the standard error have been shown to be problematic to comprehend in certain cases [47], we still opted to use them for bar charts due to their wide usage and familiarity. We render labels with margins of error (e.g., "$\pm 3\%$") in each bin of a 2D-histogram. Figure 4 (bottom) shows an example of a progressive visualization and how it changes over time. Note that the confidence intervals in the example are rather small, but their size can change significantly based on the dataset and query.

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

3.4 Procedure

We recruited 24 participants from a research university in the US. All participants were students (22 undergraduate, 2 graduate), all of whom had some experience with data exploration or analysis tools (e.g., Excel, R, Pandas). 21 of the participants were currently enrolled in and halfway through an introductory data science course. Our experiment included visualization condition as a within-subject factor and dataset-delay and dataset-order as between-subject factors. Note that the dataset-delay has no direct influence on the instantaneous condition, but it is used as a between-subject

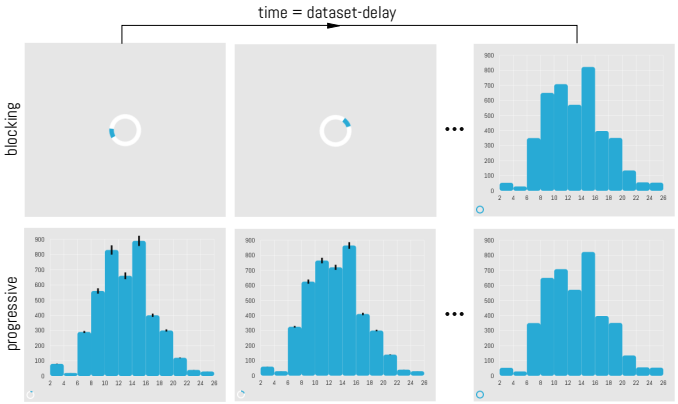


Fig. 4. The blocking and progressive visualization conditions.

factor to test if it affects the other visualization conditions. To control against ordering and learning effects, we fully randomized the sequence in which we presented the different visualization conditions to the user and counterbalanced across dataset-delay conditions. Instead of fully randomizing the ordering of datasets, we opted to create two predefined dataset-orderings and factored them into our analysis. The two possible dataset-ordering values were 123 and 312 (i.e., DS1 followed by DS2 followed by DS3 and DS3 followed by DS1 followed by DS2, respectively), and we again counterbalanced across dataset-ordering. We seed our system's random number generator differently for each session to account for possible effects in the progressive condition caused by the sequence in which data points are processed. While each participant did not experience all possible combinations of visualization, dataset-ordering, and dataset-delay conditions, all users saw all visualization conditions with one specific dataset-delay and dataset-ordering.

For example, participant P14 was assigned dataset-ordering 312, dataset-delay 6s, and visualization condition ordering [blocking, instantaneous, progressive]. In total, this adds up to four trial sub-groups, each with six participants: (dataset-delay = 6s & dataset-order = 123), (dataset-delay = 12s & dataset-order = 123), (dataset-delay = 6s & dataset-order = 312) and (dataset-delay = 12s & dataset-order = 312). Within each subgroup, we fully randomized the order in which we presented the different visualization conditions.

After a 15 minute tutorial of the system, including a summary of the visualization conditions and how to interpret confidence intervals and margins of error, we instructed the participants to perform three exploration sessions. In the case of participant P14, these three sessions included (1) blocking visualizations on DS3, (2) instantaneous visualizations on DS1, and (3) progressive visualizations on DS2 all with 6s delay. Each session was open-ended and we asked the participants to explore the dataset at their own liking and pace. We allotted 12 minutes per session, but participants were free to stop earlier We used a think-aloud protocol [48] where participants were instructed to report anything they found interesting while exploring the dataset. Throughout each session, we captured both screen and audio recordings and logged low-level interactions (mouse events) as well as higher-level events ("axis changed", "aggregation changed", "textbox brush / filter", "visualization brush," and "visualization updated / stopped / completed"). An experimenter was present throughout the session, and participants were free to ask any technical questions or questions about the meaning of dataset attributes. At the end of the three sessions, we asked participants to give feedback about the tool and whether they had any thoughts regarding the different visualization conditions.

3.5 Statistical Analysis

Our study is designed as a full-factorial 3 (visualization conditions) $\times$ 2 (dataset-delay conditions) $\times$ 2 (dataset-order conditions) experiment. We applied mixed design analysis of variance tests (ANOVA) with visualization condition as the within-subject factor and dataset-delay and dataset-order as the between-subject factors to assess the effects of our factors on the various metrics we computed. Note that the dataset-delay factor should have no influence on trials where the visualization condition is set to instantaneous.

We tested the assumption of sphericity using Mauchly's test, and we report results with corrected degrees of freedom using Greenhouse-Geisser estimates if violated. We report all significant (i.e., $p < 0.05$) main and interaction effects of these tests. For significant main effects, we conducted further analysis through Bonferroni corrected post hoc tests and for more nuanced interpretation, we opted to include Bayes factors for certain results and report BF_{10} factors along with corresponding significance labels [49]. Effect sizes are reported through Pearson's r coefficient, and significance levels are encoded in plots using the following notation: $* \; p < 0.05$; $* * \; p < 0.01$; $* * * \; p < 0.001$.

4 Analysis of Verbal Data

Inspired by previous insight-based studies [40], [44], [50], we manually coded insights from the audio recordings and screen captures. An insight is a nugget of knowledge extracted from the data, such as "France produces more wines than the US". We followed the verbal data segmentation approach proposed by Liu et. al. [40] and adopted their coding process in which the first author did the bulk of the coding, but we iteratively revised finished codes with collaborators to reduce bias. In our case, we decided to count observations that are within the same visualization, have the same semantics, and are on the same level of granularity as one insight. For example: "It looks like country 1 makes the most cars, followed by country 2 and country 3" was coded as one single insight, whereas an observation across two visualizations (through brushing) such as "Country 1 makes the most cars and seems to have the heaviest cars" was counted as two separate insights. We did not categorize insights or assign any quality scores or weights to insights nor did we quantify accuracy or validity of insights. For our analysis, all insights were treated equally.

4.1 Number of Insights per Minute

In order to get a quantifiable metric for knowledge discovery, we normalized the total insight count for each session by the duration of the session. Figure 5 shows this resulting "number of insights per minute" metric across different factors. Our results show that this metric was significantly affected by the type of visualization condition ($F(1.452, 29.039) = 6.701, p < 0.01$).

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

4.2 Insight Originality

Similar to [44], we grouped insights that encode the same nugget of knowledge together and computed the originality of an insight as the inverse of the number of times that insight was reported by any participant. That is, insights reported more frequently by participants received lower overall originality scores. A participant received an insight originality score of 1 if he or she had only unique insights (i.e., insights found by no other users). The lower the score, the less original the insights were on average. We averaged originality scores across insights for each session and show plots for this insight originality metric across different factors in Figure 6. We did not find any significant effects that

influence the insight originality metric, which indicates that the originality score seems unaffected by any of the factors for which we controlled.

5 Analysis of Interaction Logs

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

5.1 Visualization Coverage

We were interested in analyzing how much of the possible space of visualizations our participants covered. To create a metric for visualization coverage, we computed the set of unique visualizations possible within each dataset. We considered all attributes, both visualization types supported by our system, and all possible aggregation functions. However, we ignored the axis-to-attribute mapping (e.g., a visualization with attribute A on the x-axis and attribute B on the y-axis is considered the same as if the axes were flipped). We then extracted and counted up all visualizations a participant created during a session from the interaction logs. Our final visualization coverage metric is the percentage of possible visualizations a participant created per minute.

Figure 7 plots this metric across different factors. Our analysis shows that the type of visualization condition significantly affects the percentage of the total visualizations that a participant covered per minute ($F(2, 40) = 9.847, p < 0.001$). Post hoc tests revealed that the instantaneous condition showed a slight increase in percentage over the progressive condition ($1.553\% \pm 0.690\%$ vs. $1.311\% \pm 0.651\%$, respectively), which was not statistically significant ($p = 0.226, r = 0.251$). However, the blocking condition reduces the percentage of total visualizations covered per minute to $0.845\% \pm 0.449\%$, which differed significantly from both the progressive ($p < 0.01, r = 0.545$) and instantaneous ($p < 0.001, r = 0.740$) conditions. Participants explored more visualizations per minute with the instantaneous and the progressive condition and there is no significant difference between the two.

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

5.2 Number of Brush Interactions per Minute

While the coverage metric considers the number of possible static visualizations, it does not consider brushing. We measured the brushing interactions by counting the number of brush events from the interaction logs, which we then normalized by the duration of a session. The results, shown in Figure 8, demonstrate that the type of visualization condition significantly affected the number of brush interactions per minute ($F(1.335, 26.690) = 17.620, p < 0.0001$). Post hoc tests revealed that the instantaneous condition showed a significant increase in number of brush interactions per minute over the progressive condition (4.640 ± 4.095 vs. 2.108 ± 1.744, $p < 0.01, r = 0.316$), as well as over the blocking condition (1.190 ± 0.737 , $p < 0.001, r = 0.413$). Furthermore, brushing interactions per minute for the progressive condition were significantly different than for the blocking condition ($p < 0.05, r = 0.29$).

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

5.3 Visualizations Completed

We extracted the number of times a visualization was completed—the participant waited the full amount of time specified by the dataset delay until a visualization was completely accurate—from our log data. We then divided this count by the number of interactions that forced a visualization to recompute (e.g., axis change, brushing interaction). This gives us the percentage of times an interaction led to a fully accurate visualization. Figure 9 visualizes this metric for different factors. Note that, by definition, this metric is always 100% for the instantaneous condition and we therefore exclude it from post hoc tests. Our results show that the percentage of completed visualizations was significantly affected by the type of visualization condition ($F(2, 40) = 169.972, p < 0.0001$). Post hoc tests revealed that the blocking condition showed an increase of percentage of completed visualizations over the progressive condition ($56.22\% \pm 15.01\%$ vs. $45.99\% \pm 14.14\%$, respectively), which was statistically significant ($p < 0.05, r = 0.296$). In short, people often moved ahead without waiting for the full result.

5.4 Mouse Movement per Minute

Finally, to compute a simple metric of a participant's activity level, we calculated the distance in pixels the mouse was moved per minute and show the results in Figure 10. Our analysis shows that the type of visualization condition significantly affected the mouse movement per minute metric ($F(2, 40) = 5.431, p < 0.01$). Post hoc tests revealed that the instantaneous condition showed a slight decrease of mouse movement per minute over the progressive condition

(303.799 ± 128.243 vs. 313.912 ± 139.1299, respectively), which was not statistically significant ($p = 1.0, r = 0.051$). However, the blocking condition reduces movement per minute to 258.060 ± 131.536, which differed significantly from both the progressive ($p < 0.05, r = 0.315$) and instantaneous ($p < 0.05, r = 0.245$) conditions. Mouse movement is a crude indication of a user's activity level, and our test shows that these levels are lowest with the blocking conditions. Again we find no difference between instantaneous and progressive visualizations.

6 Perception of Visualization Conditions

During our exit interview, we asked participants to provide feedback about the different visualization conditions they experienced. Most participants liked the instantaneous visualizations best, but preferred the progressive ones over blocking. Below are quotes from our participants that describe these preferences:

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

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

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

7 Discussion

Our analysis provides evidence that overlaps with findings in previous work, such as by Liu et. al. [40], which finds that even small latencies have an impact on user performance in exploratory settings. Our results show that knowledge discovery and user activity measures are negatively influenced by the blocking condition when compared to instantaneous visualizations. This intuitively makes sense and it is widely acknowledged that low latency leads to improved user experience and user performance. The more interesting evidence we present in this paper arises when we compare blocking to progressive visualizations or instantaneous to progressive visualizations.

The difference between the progressive and blocking conditions is that users can see approximate results while a query is ongoing rather than a loading animation. However, the overall delay until a final, 100% accurate visualization arrives is exactly the same for both conditions. Yet, our data shows that users generated more insights per minute, had a higher visualization coverage percentage, and displayed higher levels of mouse movements when given progressive rather than blocking visualizations. Additionally, the percentage of completed visualizations (i.e., visualizations where users waited the full number of dataset-delay seconds to get the final answer) is higher in the blocking than the progressive condition. This result suggests that users are efficiently using these in-between and approximate visualizations to either pre-process information, extract insights early, or to decide that the result is not what they were looking for and then move on to the next visualization. A participant expressed it this way: *"It's much easier to look at a rough picture of the final data rather than just to see nothing at all. You can start to get an idea of relationships and things like that."*. Based on our analysis we accept hypothesis $H2$ (more insights with progressive than blocking) as well as $H4$ (higher user activity levels with progressive than blocking).

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

However, there are some open questions that our study does not address. While our data suggests that approximate visualizations might help users grasp certain characteristics of a dataset rapidly, they simultaneously introduce a set of new challenges. For example, prior work in psychology [30], [31] has shown that even experts sometimes have trouble interpreting standard accuracy metrics (e.g., confidence intervals, error bars). While we did not observe any cases in our study where participants misinterpreted visualizations based on their uncertainty, these cases are also hard to capture. Our participants often reported high-level trends like *"these two categories seem about equal in counts"*, *"there is a slight negative correlation between these two variables"*. How much of a change would need to occur between the approximate and the final visualization before users would retract these insights? Progressive visualizations expose fundamental limitations when it comes to exploration sessions on datasets where it is important to capture insights that are based on outliers or small subsets of the data. It might take a long time to sample those rare datapoints, and visualizations can therefore take a long time to converge to a view that shows those features prominently.

The second open question is how refresh rates for progressive visualizations affect user interaction and whether or not, in the extreme case, a simple one-sample visualization provides the same benefits. For example, techniques such as the one presented by Stolper et. al. [6] that allow users to decide when to update a visualization might be less distracting than our approach of constant regular updates.

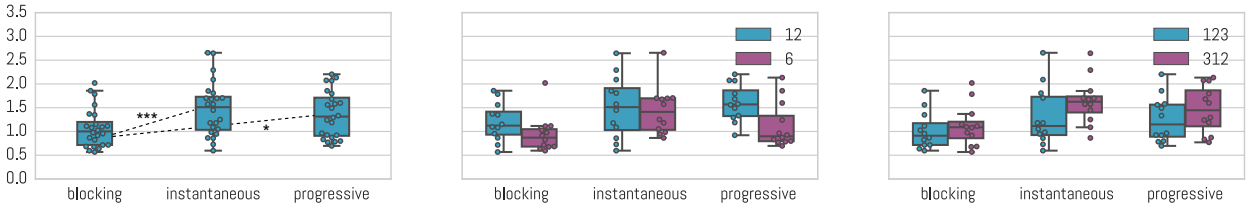


Fig. 5. Insights per Minute: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for insights per minute (left) overall, (middle) by dataset-delay, and (right) by dataset-order. Higher values indicate better.

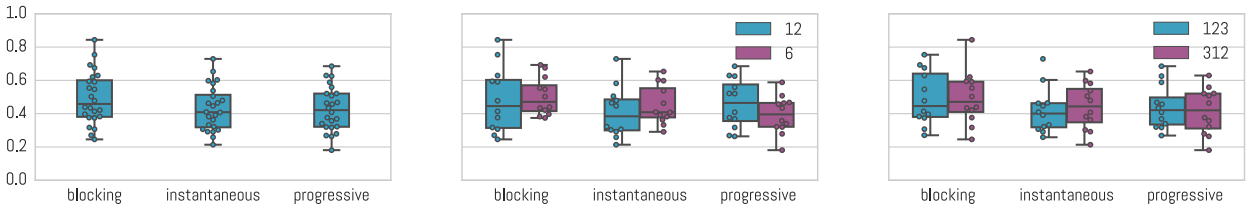


Fig. 6. Insight Originality: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for insight originality (left) overall, (middle) by dataset-delay, and (right) by dataset-order. Higher values indicate more original insights.

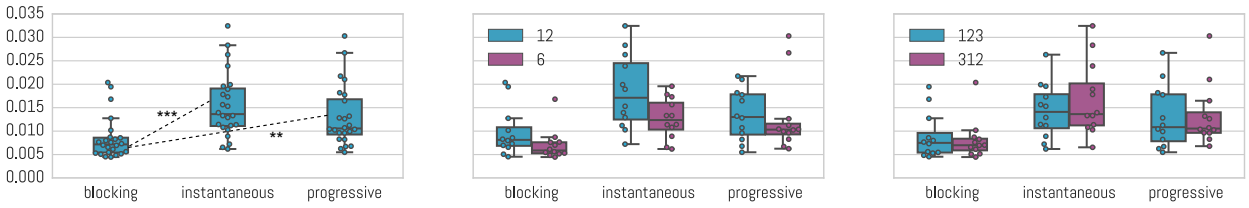


Fig. 7. Visualization Coverage Percentage per Minute: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for for visualization coverage % per minute (left) overall, (middle) by dataset-delay, and (right) by dataset-order. Higher values are better.

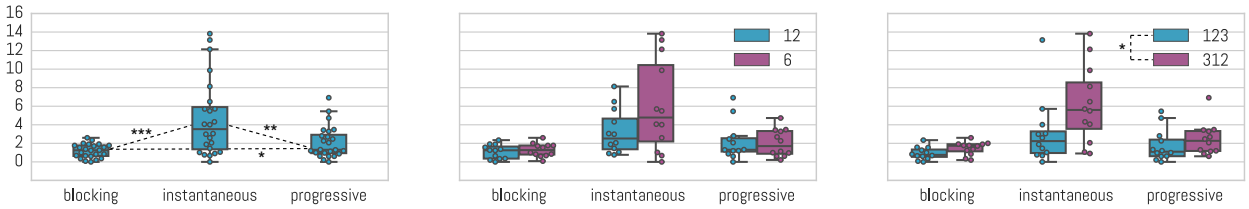


Fig. 8. Brush Interactions per Minute: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for brush interactions per minute (left) overall, (middle) by dataset-delay, and (right) by dataset-order.

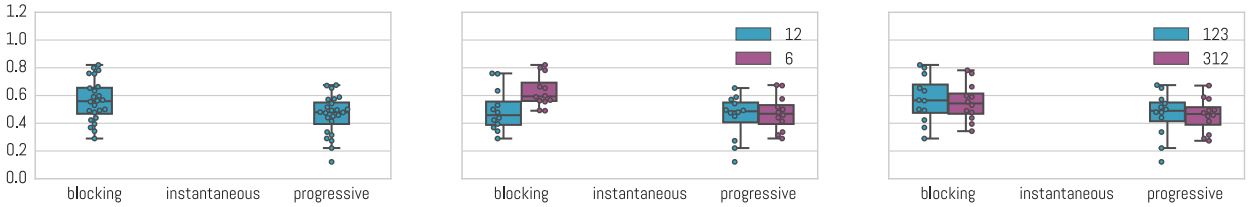


Fig. 9. Visualizations Completed Percentage: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for for completed visualization % (left) overall, (middle) by dataset-delay, and (right) by dataset-order.

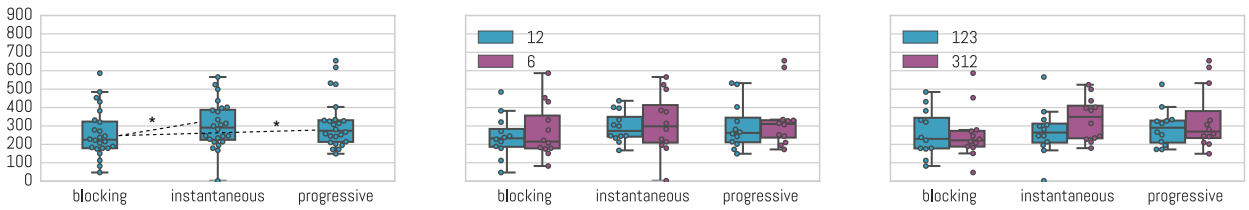


Fig. 10. Mouse Movement per Second: Boxplot (showing median and whiskers at 1.5 interquartile range) and overlaid swarmplot for mouse movement per minute (left) overall, (middle) by dataset-delay, and (right) by dataset-order.

There are reasons to believe that progressive visualizations have advantages over approximate visualization that are based on a single sample. For example, users can decide individually when a visualization is accurate enough for their liking or their current task. We observed that some users tend to hold off with reporting insights until after several progressions, especially for visualizations where there was a high visual variance between updates. In other cases, participants waited for uncertainty metrics to be in a range small enough for them to make judgments. One of our participants stated: *"I didn't want to draw any conclusion right away. Especially in this one visualization where the error bar was across the whole graph, I decided to wait."*. While these anecdotal findings make a case for progressive visualizations over simple sampling-based ones, our study does not provide the means for a quantitative comparison between the two.

8 Future Work & Conclusion

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

However, this study is just a first step towards understanding progressive visualizations. We studied only simple forms of uncertainty visualizations and we also excluded other factors from the study, such as if users are able to fully grasp the meaning of approximate answers and how distracting the visualization updates are. We plan to carry out several follow-up studies where we compare different update strategies and different types of uncertainty representations. Furthermore, we intend to get a better understanding of how a user's behavior changes between instantaneous and progressive visualizations through in-depth sequential interaction log analysis, potentially coupled with eye-tracking data. Gaining such understanding would help optimize visual representations and interactions with progressive visualizations.

Acknowledgments

The authors wish to thank Angelina Maric from the University of Zürich for her help with statistical analysis as well as Professor Andries van Dam, Robert Zeleznik, and the reviewers for their comments, feedback, and suggestions.

References

[1] S. C. Seow, *Designing and engineering time: The psychology of time perception in software*. Addison-Wesley Professional, 2008.

[2] J. Nielsen, "Powers of 10: Time scales in user experience," *Retrieved January*, vol. 5, p. 2015, 2009.

[3] S. K. Card, G. G. Robertson, and J. D. Mackinlay, "The information visualizer, an information workspace," in *Proceedings of the SIGCHI Conference on Human factors in computing systems*. ACM, 1991, pp. 181–186.

[4] B. Shneiderman, "Response time and display rate in human performance with computers," *ACM Computing Surveys (CSUR)*, vol. 16, no. 3, pp. 265–285, 1984.

[5] D. Fisher, I. Popov, S. Drucker *et al.*, "Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 1673–1682.

[6] C. D. Stolper, A. Perer, and D. Gotz, "Progressive visual analytics: User-driven visual exploration of in-progress analytics," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 20, no. 12, pp. 1653–1662, 2014.

[7] D. Fisher, "Incremental, approximate database queries and uncertainty for exploratory visualization," in *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*. IEEE, 2011, pp. 73–80.

[8] D. Fisher, S. M. Drucker, and A. C. König, "Exploratory visualization involving incremental, approximate database queries and uncertainty," *IEEE computer graphics and applications*, no. 4, pp. 55–62, 2012.

[9] J.-D. Fekete and R. Primet, "Progressive analytics: A computation paradigm for exploratory data analysis," *CoRR*, vol. abs/1607.05162, 2016. [Online]. Available: http://arxiv.org/abs/1607.05162

[10] R. Rosenbaum and H. Schumann, "Progressive refinement: more than a means to overcome limited bandwidth," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2009, pp. 72 430I–72 430I.

[11] R. Amar, J. Eagan, and J. Stasko, "Low-level components of analytic activity in information visualization," in *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*. IEEE, 2005, pp. 111–117.

[12] J. Heer and B. Shneiderman, "Interactive dynamics for visual analysis," *Queue*, vol. 10, no. 2, p. 30, 2012.

[13] D. Fisher, R. DeLine, M. Czerwinski, and S. Drucker, "Interactions with big data analytics," *interactions*, vol. 19, no. 3, pp. 50–59, 2012.

[14] C. Dunne, N. Henry Riche, B. Lee, R. Metoyer, and G. Robertson, "Graphtrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 1663–1672.

[15] E. Zgraggen, R. Zeleznik, and S. M. Drucker, "Panoramicdata: Data analysis through pen & touch," *IEEE Transactions on Visualization and Computer Graphics*, vol. 99, no. PrePrints, p. 1, 2014.

[16] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska, "Vizdom: interactive analytics through pen and touch," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 2024–2027, 2015.

[17] "Tableau," http://www.tableau.com/, accessed: 2015-06-03.

[18] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: A system for query, analysis, and visualization of multidimensional relational databases," *IEEE Trans. Vis. Comput. Graph.*, vol. 8, no. 1, pp. 52–65, 2002. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/2945.981851

[19] "Tiboco Spotfire," http://spotfire.tibco.com/, accessed: 2015-06-03.

[20] "Microsoft Power BI," https://powerbi.microsoft.com/en-us/, accessed: 2015-06-03.

[21] S. Chaudhuri and U. Dayal, "An overview of data warehousing and olap technology," *ACM Sigmod record*, vol. 26, no. 1, pp. 65–74, 1997.

[22] L. Lins, J. T. Klosowski, and C. Scheidegger, "Nanocubes for real-time exploration of spatiotemporal datasets," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 19, no. 12, pp. 2456–2465, 2013.

[23] Z. Liu, B. Jiang, and J. Heer, "*imMens*: Real-time visual querying of big data," *Comput. Graph. Forum*, vol. 32, no. 3, pp. 421–430, 2013.

[24] L. Battle, R. Chang, and M. Stonebraker, "Dynamic prefetching of data tiles for interactive visualization," 2015.

[25] A. Kalinin, U. Cetintemel, and S. Zdonik, "Interactive data exploration using semantic windows," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 505–516.

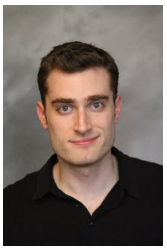
[26] P. R. Doshi, E. A. Rundensteiner, and M. O. Ward, "Prefetching for visual data exploration," *Database Systems for Advanced Applications, International Conference on*, vol. 0, p. 195, 2003.

[27] A. Ottley, H. Yang, and R. Chang, "Personality as a predictor of user strategy: How locus of control affects search strategies on tree visualizations," in *Proceedings of the 33rd Annual ACM Conference*

on Human Factors in Computing Systems. ACM, 2015, pp. 3251–3254.
[28] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica, "Blinkdb: queries with bounded errors and bounded response times on very large data," in *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 2013, pp. 29–42.
[29] N. Kamat, P. Jayachandran, K. Tunga *et al.*, "Distributed and interactive cube exploration," in *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*. IEEE, 2014, pp. 472–483.
[30] G. Cumming, "Inference by eye: reading the overlap of independent confidence intervals," *Statistics in medicine*, vol. 28, no. 2, pp. 205–220, 2009.
[31] ——, *Understanding the new statistics: Effect sizes, confidence intervals, and meta-analysis*. Routledge, 2013.
[32] N. Ferreira, D. Fisher, and A. C. Konig, "Sample-oriented task-driven visualizations: allowing users to make better, more confident decisions," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2014, pp. 571–580.
[33] C. Harrison, A. K. Dey, and S. E. Hudson, "Evaluation of progressive image loading schemes," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2010, pp. 1549–1552.
[34] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas, "Interactive data analysis: The control project," *Computer*, vol. 32, no. 8, pp. 51–59, 1999.
[35] P. J. Haas and J. M. Hellerstein, "Ripple joins for online aggregation," in *ACM SIGMOD Record*, vol. 28, no. 2. ACM, 1999, pp. 287–298.
[36] C. Jermaine, A. Dobra, S. Arumugam, S. Joshi, and A. Pol, "The sort-merge-shrink join," *ACM Transactions on Database Systems (TODS)*, vol. 31, no. 4, pp. 1382–1416, 2006.
[37] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "Mapreduce online." in *NSDI*, vol. 10, no. 4, 2010, p. 20.
[38] J.-D. Fekete, "ProgressiVis: a Toolkit for Steerable Progressive Analytics and Visualization," in *1st Workshop on Data Systems for Interactive Analysis*, Chicago, United States, Oct. 2015, p. 5. [Online]. Available: https://hal.inria.fr/hal-01202901
[39] P. Hanrahan, "Analytic database technologies for a new kind of user: the data enthusiast," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 577–578.
[40] Z. Liu and J. Heer, "The effects of interactive latency on exploratory visual analysis," *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2014. [Online]. Available: http://idl.cs.washington.edu/papers/latency
[41] J. Brutlag, "Speed matters for google web search," *Google. June*, 2009.
[42] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, "The effects of loss and latency on user performance in unreal tournament 2003®," in *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*. ACM, 2004, pp. 144–151.
[43] J. Deber, R. Jota, C. Forlines, and D. Wigdor, "How much faster is fast enough?: User perception of latency & latency improvements in direct and indirect touch," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 1827–1836.
[44] H. Guo, S. Gomez, C. Ziemkiewicz, and D. Laidlaw, "A case study using visualization interaction logs and insight," 2016.
[45] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[46] A. Dobra, C. Jermaine, F. Rusu, and F. Xu, "Turbo-charging estimate convergence in dbo," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 419–430, 2009.
[47] M. Correll and M. Gleicher, "Error bars considered harmful: Exploring alternate encodings for mean and error," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 12, pp. 2142–2151, 2014.
[48] K. A. Ericsson and H. A. Simon, "Protocol analysis," 1993.
[49] M. D. Lee and E.-J. Wagenmakers, *Bayesian cognitive modeling: A practical course*. Cambridge University Press, 2014.
[50] S. R. Gomez, H. Guo, C. Ziemkiewicz, and D. H. Laidlaw, "An insight-and task-based methodology for evaluating spatiotemporal visual analytics," in *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on*. IEEE, 2014, pp. 63–72.

Emanuel Zgraggen received his Fachhochschuldiplom in Informatik from HSR Hochschule für Technik Rapperswil in Switzerland and his MS in computer science from Brown University. He is currently a PhD candidate at Brown University working in the graphics group. His main research areas are Visual Analysis, Human Computer Interaction and Information Visualization.

Alex Galakatos received his BS in computer engineering from Lehigh University and his MS in computer science from Brown University. He is currently a PhD candidate at Brown University working in the data management group. His interests include new systems for interactive data exploration as well as large-scale machine learning.



Andrew Crotty received a BA in computer science from Boston College (2013) and MS in computer science from Brown University (2015). He is currently pursuing a PhD in computer science at Brown, where he works as a member of the data management group. His work mainly focuses on the development of new big data analytics tools, with a focus on complex computation like machine learning.

Jean-Daniel Fekete is the Scientific Leader of the INRIA Project Team AVIZ that he founded in 2007. He received his PhD in Computer Science in 1996 from University of Paris Sud, France. He joined the Human-Computer Interaction Laboratory at the University of Maryland in the USA for one year and was recruited by INRIA in 2002 as a confirmed researcher and became Senior Research Scientist in 2006. His main research areas are Visual Analytics, Information Visualization and Human Computer Interaction.

Tim Kraska is an Assistant Professor in the Computer Science department at Brown University. Currently, his research focuses on Big Data management systems for modern hardware and systems for interactive data exploration. Before joining Brown, Tim Kraska spent 2 years as a PostDoc in the AMPLab at UC Berkeley after receiving his PhD from ETH Zurich, where he worked on cloud-scale data management systems and stream processing.