Noname manuscript No.
(will be inserted by the editor)

Evaluating Interactive Data Systems

Survey and Case Studies

Protiva Rahman · Lilong Jiang · Arnab Nandi

Received: date / Accepted: date

Abstract Interactive query interfaces have become a popular tool for ad-hoc data analysis and exploration. Compared with traditional systems that are optimized for throughput or batched performance, these systems focus more on user-centric interactivity. This poses a new class of performance challenges to the backend, which are further exacerbated by the advent of new interaction modes (e.g., touch, gesture) and query interface paradigms (e.g., sliders, maps). There is, thus, a need to clearly articulate the evaluation space for interactive systems.

In this paper, we extensively survey the literature to guide the development and evaluation of interactive data systems. We highlight unique characteristics of interactive workloads, discuss confounding factors when conducting user studies, and catalog popular metrics for evaluation. We further delineate certain behaviors not captured by these metrics and propose complementary ones to provide a complete picture of interactivity. We demonstrate how to analyze and employ user behavior for system enhancements through three case studies. Our survey and case studies motivate the need for *behavior-driven* evaluation and optimizations when building interactive interfaces.

P. Rahman, A. Nandi
Computer Science and Engineering, The Ohio State University, 2015 Neil Ave, Columbus, OH 43210
Tel.: +001-614-2922572
E-mail: {rahmanp,arnab}@cse.ohio-state.edu

L. Jiang
Twitter, Inc., 1355 Market St, San Francisco, CA 94103
Tel.: +001-415-2229670
E-mail: lilongj@twitter.com.

1 Introduction

The recent data deluge requires input from various stakeholders, such as business executives and physicians, who are not necessarily trained in computer science. To allow these users to perform ad-hoc, iterative analysis with instant feedback, a new class of interactive human-in-the-loop systems are becoming increasingly common [3, 16, 43, 50, 91, 130]. These include systems for data cleaning and preparation [101, 148], analysis [16, 88, 115, 150, 158, 194] as well as visualization recommendation systems [57, 167, 176, 177]. Each of these systems consist of a combination of query interface tools. For example, Tableau [16] allows its users to compose queries via text box (keyword search), slider (query by sliding), and map (query by zooming and dragging). Further, each of these query interfaces, e.g., crossfiltering [3], generates a unique workload, which needs to be considered, individually and in composition with other interfaces, when evaluating and optimizing it.

This growth of interactive systems has been accompanied with an increase in touch-based (e.g., iPad, Microsoft Surface), and gesture-driven devices (e.g., Kinect, HoloLens, Leap Motion). In 2018, 1.56 billion smartphones [11] and over 36.4 million tablets were shipped [22]. These patterns suggest that touch and gestures will likely be the primary mode of data interaction in the near future. In fact, recent work by multiple members of the community has shown that touch devices can be more usable and intuitive than mouse for data interaction, and that the performance of gestural devices can be considered comparable to mouse-driven interaction [95, 122, 137, 195].

Taking advantage of both touch and interactivity, commercial tools such as Tableau Vizable [17] and Microsoft Power BI [14] provide touch-based data analysis.

On the other hand, the rise of virtual and augmented reality has popularized gestural querying in multiple industries, including automobile [5,163] and health care [18, 151]. This trend towards non-keyboard based interactive data analysis motivates the need for a deeper study into the workloads generated by these systems along with methods for evaluating them. Evaluation is a critical component of building systems: optimizing the user experience requires capturing the correct performance metric.

Contributions and Outline: In this survey, we first discuss salient features of evaluating interactive systems which differentiate them from traditional database queries. These include themes such as querying on touch devices as opposed to a physical computer, interfaces that provide continuous action, imprecise query intent, and related querying in a session. Each of these features poses challenges for data processing. They often lead to a large number of queries being issued in a small time frame and motivate the need for *behavior-driven* optimizations. We discuss these issues and provide compelling examples from related work in Section 2. In Section 3, we provide a taxonomy of metrics used in the literature and introduce two new ones to provide a more complete picture of interactivity. We summarize the metrics used by different systems in Tables 1 and 2 and provide guidelines on when a metric is relevant in Table 3. The goal of this section is to introduce the reader to certain metrics which they might not be familiar with, especially from the HCI side. We cover a wide-range of applications to expose different metrics, and as such cannot compare systems to provide a *state-of-the-art* interactive system. In Section 4, we summarize user study design criteria and methods for avoiding bias during in-person studies.

Advanced readers, familiar with user studies and metrics, can skip to Section 5, where we dive into guidelines for implementing *behavior-driven* optimizations. We demonstrate these principles and the use of different metrics to inform system design through three case-studies. The case studies cover a combination of devices, interfaces and user behavior. They are meant to provide examples of techniques for *behavior-driven* optimizations and not as benchmark results. They are organized as follows: Section 6 discusses techniques for optimizing the user's inertial scrolling experience as they are browsing through large query results. Section 7 discusses optimizations for coordinated brushing and linking, an interface commonly used to get quick insights from large multidimensional datasets. Section 8 studies user behavior in composite interfaces containing map, slider and text box querying. Sections 9 and 10 discuss related work and conclusions.

2 Salient Features of Interactive Data Systems

Workloads generated by interactive systems have salient characteristics that distinguish them from traditional database workloads. These factors need to be taken into consideration when selecting metrics for evaluation. We describe these features here with relevant examples from literature.

2.1 Devices and Interfaces

Different input devices have different sensing rates for user input, which directly affect the *query issuing frequency*, i.e., number of queries issued per second. Further, each device-interface combination generates a unique workload. For example, one zoom action on a map interface triggers two predicate changes (longitude and latitude) in the `WHERE` clause. Range sliders tend to trigger more intensive workloads (number of queries per second) than text input since sliding typically takes less time than typing. Two systems that account for this in their evaluation are TouchViz [63] and DBTouch [122]:

- TouchViz [63] empirically evaluates two UI design alternatives for touch-based devices. In their limitations section, the authors acknowledge that since their experiments are done on a touch interface, they need separate experiments to see if the results hold on *mouse-based* devices. They also mention that additional studies are needed for expanding the design to a *multi-device interface.*
- DBTouch [122] allows users to query databases using touch gestures. They process queries on a sample of the data as the user slides through values in a column. The sample size is defined by the gesture speed as well as the zoom level or the size of the object. Thus, two parameters that are studied in their evaluation include the *gesture speed* and the *object size*, since changes to either affect the number of objects processed. Such design decisions further motivate the need for separate evaluations for devices with different sensing rates or sizes.

2.2 Continuous Action

For touch and gesture devices [137] used in conjunction with continuous manipulation query interfaces (e.g., slider, linking and brushing) [62], the user's query specification has to be treated as a continuous process, with each change triggering a query. In direct manipulation systems, the whole process needs to remain interactive, motivating the need for a strict latency constraint. The fluid nature of this interaction generates heavy query workloads. Consider for example the case of *crossfiltering*, where the user moves the slider continuously to filter the dataset. If the interface detects events every $20ms$, then every second about 50 queries are issued to the backend from a single user. A common technique for providing low-latency is prefetching. The following examples employ prefetching for the use cases of geographic systems and visual data exploration, respectively:

– In the geoinformatics community, Yesilmurat et al. [188] introduced retrospective adaptive prefetching (RAP), which leverages the user's historical navigation pattern to prefetch tiles. Measuring the effectiveness of the algorithm requires a diverse set of user behavior. To address this, user behavior is simulated as a sequence of navigations. In the database community, ForeCache [36] is a similar system.

– Prefetching has also been used for visual data exploration in Doshi et al.'s 2003 SIGMOD paper [62], which studies star glyphs, parallel coordinates and dimensional stacking. Their evaluation also consists of simulation studies with varying interaction patterns. However, they perform additional validation with 30 minute interaction traces from 20 users. Thus, evaluating continuous action requires a sequence of interactions which can be sometimes be simulated.

Another consequence of continuous action is that *adjacent queries*, i.e., those issued in adjacent timestamps, usually return related or similar results since there is usually only a difference of one condition (e.g., `where` clause predicate) between them. This dependence can be used for performance optimizations, as shown in Section 7.

2.3 Ambiguous Query Intent

In exploratory data analysis, users often do not know what they are looking for. This is characterized by statements such as: *"I can't tell you what I want, but I'll know it when I see it"*. The old assumption in database systems that the user has a well-formed query is no longer valid. Interactive systems thus need to guide users to insights via instantaneous feedback and cues, as demonstrated in the following examples.

– SeeDB [167] guides users to insights by showing them interesting visualization, where interestingness is measured as deviation from a reference visualization. Comments from their user study evaluation show that the system is able to provide a good overview and a starting point for further analysis of trends.

– GestureDB [137] is another gestural querying system, similar to DBTouch. It approaches the problem of maintaining interactivity in the presence of ambiguity by anticipating the intended query through classification of gestures.

Ambiguity of queries is further exacerbated by sensitivity and jitter when using gestural devices, such as HoloLens or LeapMotion. When a user interacts with the mouse and touch devices, they are manipulating physical objects. The presence of friction and force make these interactions more accurate. However, on gestural devices, due to the absence of friction the interaction process is highly variable and sensitive [95]. The user's difficulty in holding the cursor steady at a specific point is compounded with the sensor's ability to detect minor hand movements. These effects trigger unintended, noisy, and repeated queries, more frequently than mouse and touch devices. Figure 11 shows the trace for mouse, touch and gesture devices. Systems need mechanisms to deal with these unintended queries. We demonstrate one such mechanism in our case study in Section 7.

2.4 Session Behavior

In interactive systems, the concept of sessions can be leveraged. Compared to traditional database systems where single independent queries were executed, in interactive analysis, consecutive queries are often related. This is because the user might be looking for a particular insight, or formulating their current query based on results of the prior one. ForeCache [36] and Sesame [100] are two systems that leverage session behavior:

– ForeCache [36] employs prefetching for fast visualization of large datasets. Their user studies show that the users' actions are session/task dependent and that large groups of users have similar behavior. Thus by analyzing user study traces, the system can be optimized for most users.

– Sesame [100] is another session-aware system that achieves performance gains of up to *25x* by reusing results of previous queries, which is only available in session-based querying.

3 Metrics

The first step of evaluating a system involves the selection of metrics, i.e., measures for assessing the system.

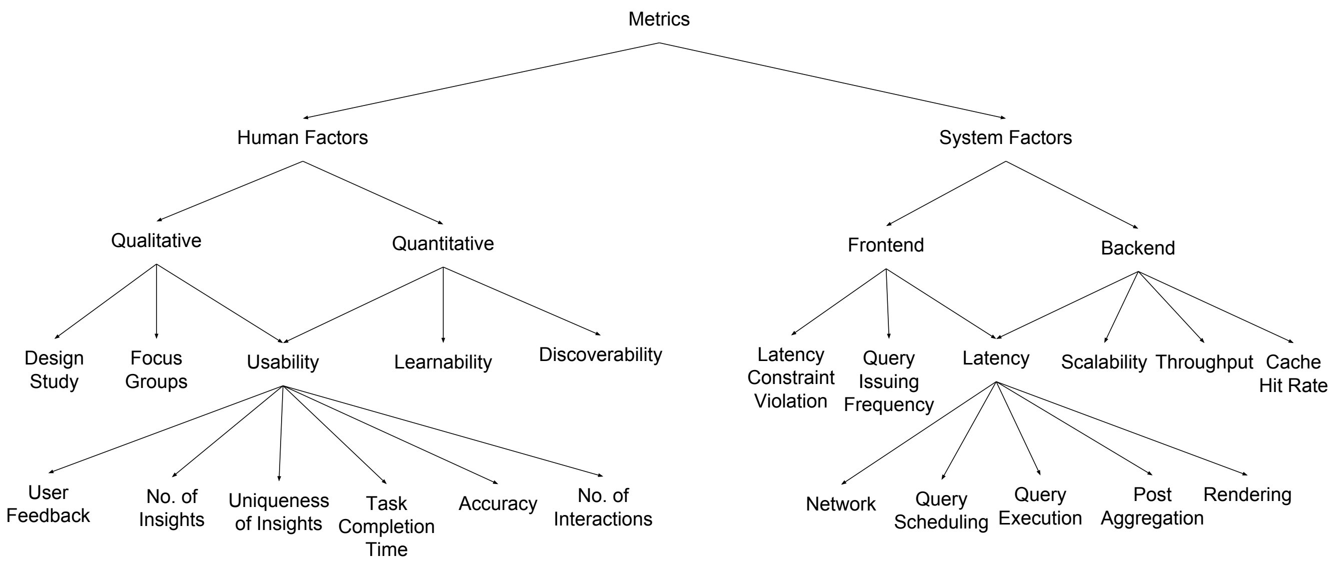


Fig. 1: Metrics

Due to the heterogeneity of interactive systems, a variety of metrics have been used in the past, some of which we discuss here. This list is by no means comprehensive but is meant to be a starting point, new systems might very well require additional metrics. We categorize metrics as human and system factors, depending on whether or not human involvement is required to measure them, as shown in Figure 1.

3.1 System Factors

The metrics in this category capture various aspects of the system and can be measured without involving humans. While the traditional TPC benchmarks [19] are ideal for transactional systems, they are inadequate for human-in-the-loop systems. Performance metrics remain a top priority of interactive systems but measuring max, median, or mean latency is not enough and a richer set of metrics is required. We group system metrics by frontend and backend.

Prior system metrics focus on backend factors, but as mentioned in the previous section, for interactive systems the frame rate of the frontend affects performance. To capture this, we introduce two novel frontend metrics: *Latency Constraint Violation* and *Query Issuing Frequency*, and demonstrate their usage via case studies in the following sections.

3.1.1 Backend Metrics

Throughput: Throughput is a classic metric used to measure database output, included in TPC-C and TPC-E. It can be calculated as transactions, requests or tasks per second. This metric is appropriate when building a distributed system, such as the Atlas system [49]. Atlas employs load balancing among distributed servers in order to visualize large scale temporal data. In Atlas' evaluation, Chan et al. measure speedup as increase in query throughput (i.e., number of queries processed by database) over baseline (1 server) with increase in number of servers.

Scalability: Scalability is another traditional performance metric which refers to change in performance with increase in data. There are two approaches for increasing performance: one is scaling up which refers to increasing the performance on a single machine (by increasing CPU speed, moving to main memory) and the other is scaling out which refers to splitting data on multiple machines to parallelize computation. But both of these methods provide improvement only up to a certain level. This is seen in experiments of the DICE (Distributed Interactive Cube Exploration) system [98]. DICE performs speculative query caching based on the faceted cube traversal model for distributed systems. Kamat et al. do a scalabililty experiment comparing decrease in execution time with number of distributed nodes. Their figure (Fig. 7 in [98]) show that increasing past 8 nodes is enough to avoid thrashing, but further increasing nodes provides diminishing returns.

Splitting data across machines comes at the cost of combining and aggregating these results before presenting them to the user. The cognitive ability of the user is the bottleneck here. Even if the system is able to process large amounts of data, the user is only able to consume a screenful. Thus, if the result is too large or complicated, some form of summarization (e.g., group-by, aggregates, etc.) needs to be done before presenting it to the user. The bottle neck of summarization is also demonstrated in DICE (Fig. 6 in [98]), where the cost of evaluating additional `WHERE` clause conditions dominate the benefits of selectivity (i.e., fewer tuples to process), when increasing number of dimensions.

Cache Hit Rate: Cache hit rate refers to the number of times results of a query were found in the cache. This is an appropriate metric when designing caches or employing speculative prefetching. Things to consider include the location of the cache: frontend caches reduce load on the database but they are hard to maintain due to challenges in cache invalidation [131]. Backend caches still have to pay for network latency, but have constant execution time if an item is found in the cache.

In terms of caching strategies, eviction based policies such as least recently used and first-in first-out are not as effective as predictive caching [36,98,164]. Cache hit rate is measured by the Scout system [164]. Scout proposes methods for content aware prefetching of spatial scientific data and compares cache hit rate against baselines in prior works. Additionally, they report results of sensitivity analysis of different parameters on the cache hit rate.

Latency: While the above metrics are important, latency is more critical for interactive systems since it is directly perceived by the user [126]. Latency encompasses a lot more than just query execution time. It is calculated from the moment the user hits submit till they get back results. Depending on the system it can include:

– **Network Latency**: Network latency encapsulates the time it takes to send the request to the server as well the time to send the response back to the client. It can play a factor in distributed systems for deciding where certain computations are done, i.e., if the raw data or if the results of applying a function to it should be transmitted [125].

– **Query Scheduling Latency**: This refers to the time between when the system receives a query request to the time it starts executing the query. This usually depends on the query planner and can lead to cascading failures if new queries are issued before prior ones finish executing [182]. There are opportunities for optimizations here, if the user has moved on in their analysis before their previous query has begun executing (Section 7).

– **Query Execution Latency**: The traditional definition of latency, this refers to the actual time taken to execute the query. When reported by itself, it can be misleading since the total time can include the other factors discussed here.

– **Post-aggregation Latency**: This refers to the time taken to summarize, rank, bin, highlight, etc. results before they are presented to the user [134]. In interactive systems, presenting aggregated results often comes at the cost of maintaining lineage information of the raw data points which led to the aggregate [146].

– **Rendering Latency**: This is the time taken to render the results on the screen (Section 8).

By measuring latency at a finer granularity, optimizations such as prefetching (Section 8) or progressive rendering can be applied [70]. The latter refers to incrementally presenting partial results to the user until the complete results are available. Examples of this include online aggregation [85], where approximate results with increasing accuracy over time (by increasing sample size) are presented to the user. Incvisage [150] is another such example, where a coarse approximate of a visualization is shown to the user, which gradually refines to the final visualization, while maintaining the salient features of the visualization at every step.

The goal of measuring latency in interactive systems is to ensure that the user has a seamless experience and is not kept waiting for results. The above metrics measure the **system latency**, but to define an acceptable threshold for what is considered interactive, the user's **perceptual latency** needs to be quantified. There have been multiple studies in the HCI and visualization communities to quantify this, however the threshold is often task specific. These numbers can be used to avoid wasting computational resources if there is no benefit to the user.

– **Visual Analysis System:** Liu and Heer [126] find that adding a $500ms$ delay to interactive visual analytic systems is noticeable to users and has a negative effect on the user's analyses. Further, exposure to the delay in initial stages has a detrimental effect which remains even when the delay is removed at a later stage. Thus, an additional $500ms$ is perceivable, but it is unclear if a lower threshold is also perceivable.

– **Head Mounted Devices:** Nelson et al. [138] study the effects of time delay for head mounted devices.

Participants are instructed to visually follow stimuli presented to them for three conditions: base time, base time+50ms, and base time+100ms. After each experiment, participants are asked to fill a sickness severity questionnaire (to assess feelings of nausea). They find that base+50ms had the lowest sickness score, and that total time and not time delay is the main factor in user experience with these devices.

- **Target Acquisition:** Pavlovych et al. [143] study the effects of latency and jitter on target acquisition and tracking with mouse. They find that target acquisition accuracy drops with latency above 50ms and tracking accuracy drops above a latency of 110ms. This threshold might be relevant if users issue queries that require drawing or moving a slider.
- **Pointing Tasks on Touchscreens:** Jota et al. [97] study the effects of latency on direct pointing on touchscreens. Participants are asked to touch a target on the screen and then note the latency of the appearance of a rectangle on screen. They are further asked to identify the lower latency interface in pairwise tests. They find that participants are able to identify a difference of 20ms in latency, but are not able to perceive differences below that.

3.1.2 Frontend Metrics

Latency Constraint Violations (LCV): A key metric which we find missing in prior work is perceived latency violations. Currently, the state of the art involves measuring the mean or max latency of queries. However, in interactive systems the user issues a sequence of queries, where a query is often dependent on results of the prior one. For example, the user first scrolls left on a map and then zooms in before results of the left scroll has completed. There are two things that can happen, either the system abandons the scroll and zooms in at the point when the query is issued or the system first completes the scroll and then zooms. In the first case, the user has zoomed into an incorrect point, since the scroll did not complete. In the second case, the latency perceived by the user is the time to scroll plus the time to zoom. The latency of individual scroll and zoom actions might meet the interactive latency requirements, but added together they are violated.

Thus, measuring latency is not enough and we need a metric that captures the violations which are perceived by the user. In order to capture this stricter notion, we introduce *latency constraint violation (LCV)*. Specifically, this metric counts the number of times the zero latency rule is violated, i.e., the user perceives a delay. We describe this metric in the context of two of our case studies (Sections 6 and 7).

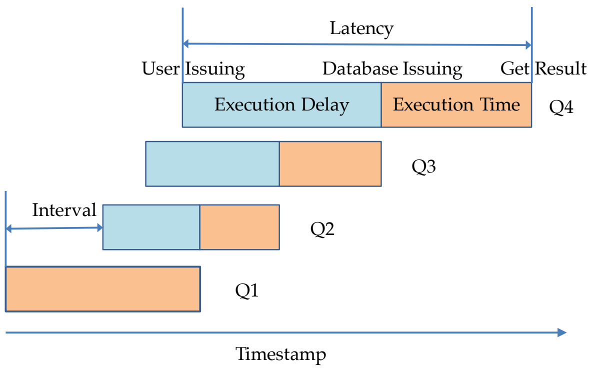


Fig. 2: Latency Constraint Violation: Before Q1 finishes its execution, Q2, Q3 and Q4 are already issued by the user in the front-end. When executing Q4, there is already the execution delay caused by previous queries.

Query Issuing Frequency (QIF): We refer to the number of queries issued per second as *query issuing frequency (QIF)*. Overtime the sensing rate for devices has gone up: the iPad used to be at 30Hz, but has now gone up to 120Hz with the Apple Pencil. This means that the frontend has the capacity to send 120 queries per second to the backend! While this provides smooth interactions for the user, it comes with a trade-off. In an ideal scenario, the backend should be able to handle the high frequency. However, if the backend is slow, then the high QIF can render it unresponsive. There is then, a need to throttle the number of queries being sent to match the backend capacity. This is because even if the user issues queries at a high rate, they are limited in the amount of information they can process, so progressively presenting them with results is adequate. QIF is thus a metric that should be measured for each system on each device to make sure it matches performance of the backend. We demonstrate this metric in Section 7.

3.2 Human Factors

Since interactive systems have a human-in-the-loop, it is important to have metrics that capture the user experience. We refer to these as human factors since they require user interaction for measurement. They include quantitative as well as qualitative metrics.

3.2.1 Qualitative Metrics

Qualitative metrics such as open-ended comments or feedback questionnaires can often provide deeper in-

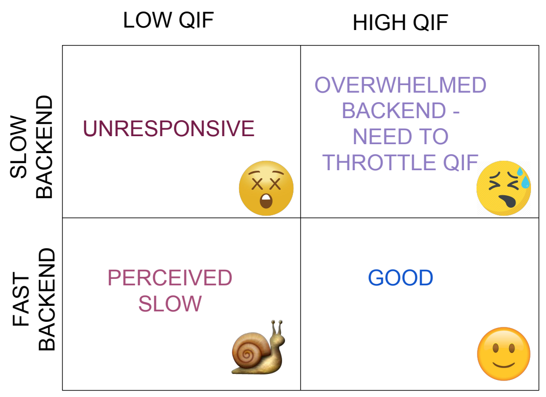


Fig. 3: Trade-offs with backend and frontend performance

sights than just quantitative metrics, especially in the preliminary development stages. However, they should be accompanied by quantitative metrics to provide a complete picture.

Design Study: Rather than a method of evaluation, design studies consist of extended interviews with practitioners for task definition, i.e., in order to articulate the problem space. These studies are meant for specifying user study tasks that simulate real world cases, and sometimes even for gathering system requirements. This is employed in the evaluation of Zenvisage [158], where Siddiqui et al. first found a taxonomy of tasks in the literature. Next, they met with seven data analysts for an hour to go over every step of their workflow and to verify that the tasks defined in the taxonomy matched experiences of the experts.

Focus Groups: Commonly used in market research, focus groups consist of small groups of people with a specific background or skill set whose opinion on a product is required. Similar to design studies, such feedback can be helpful in intermediate stages of system development to get collective feedback on system features or design decisions.

The focus group methodology is employed by Patterson et al. [142] for ranking antibiotics for an empiric antibiotic decision support tool [82]. The focus group met for 60 minutes during which participants were required to use their expertise to come to consensus on their antibiotic preferences for a hypothetical patient condition.

User Feedback: User feedback through comments, suggestions and survey can provide detailed insight on future improvements and new features. They can be reported as insight-based anecdotal comments, as reported in Tensorboard evaluations [178] or as quantitative values through Likert scale scores. The Scented Widgets system [175] employed a custom quantitative survey and report the mean and standard deviation of scores. The system usability scale [45] is commonly used for measuring system usability [31]. Recently, the ICE-T survey has been proposed for measuring visualizations [170].

3.2.2 Quantitative Metrics

Quantitative metrics consist of learnability, discoverability and usability, however, there are multiple flavors of measuring usability.

Learnability: Learnability of a system measures how quickly users are able to learn the functionalities of a system *after being taught to use it.* A system that is usable is not necessarily learnable. For example, the controls in a cockpit are very usable, but training a novice user to use it is a difficult task. Not all systems are required to be learnable, it is dependent on target audience. A database administrator, for example, would prefer writing SQL queries over doing joins with gestures, hence gesture-based query interfaces have a higher learnability requirement.

Gesture-based interfaces, such as GestureDB [137] and DBTouch [122] are targeted towards non-technical users and hence should be learnable. While this metric is reported in GestureDB, it is not tested in DBTouch. Another example of learnability is found in Kinetica [152], a touch-based data exploration system. They compare their system against Microsoft Excel, and for both cases, users received a tutorial on functionalities of the system they were testing, even if they were familiar with Excel. Thus, for learnability experiments, it is important to ensure that the training is equalized across conditions.

Discoverability: Discoverability measures how quickly users are able to find user actions *without instructions.* An everyday example of this is found in self check-in airport kiosks, where the required information and interactions are clear, even to first time fliers. For more complicated systems, affordances, i.e., usage clues, can be provided on the interface to make it easier for the user to discover actions. Examples of affordances include blinking buttons or showing the direction of a slide with a glimmer or color gradient. It goes without saying that the same group of users can not be used for learnability and discoverability experiments. An example of a discoverability experiment can be found in GestureDB [137].

Table 1: Metrics for Data Interaction 1997-2012

	Human Factors				System Factors		
	User Feedback	No. of Insights	Task Completion Time	Number of Interactions	Latency	Scalability	Throughput
Online Aggregation, 1997 [85]					X		
Igrarashi et al., 2000 [92]	X		X				
Fekete and Plaisant [71]					X		
Yang et al., 2003 [187]	X						
Plaisant, 2004 [144]	X						
Yang et al., 2004 [186]	X						
Seo and Schneiderman, 2005 [155]	X						
Kosara et al., 2006 [113]	X						
Mackinlay et al., 2007 [128]	X						
Scented Widgets, 2007 [175]	X	X					
Faith, 2007 [68]		X					
Jagadish et al., 2007 [93]			X				
Yang et al., 2007 [185]			X				
Nalix, 2007 [121]				X			
Heer et al., 2008 [83]	X						
LiveRac, 2008 [130]	X						
Basu et al., 2008 [34]				X			
Atlas, 2008 [49]					X		X
Liu and Jagadish, 2009 [124]	X						
Woodring and Shen, 2009 [179]	X						X
Facetor, 2010 [104]	X			X	X		
Wrangler, 2011 [102]	X		X				
Dicon, 2011 [47]	X		X				
Yang et al., 2011 [184]			X				
Kashyap et al., 2011 [105]				X			
Fisher et al., 2012 [74]	X						
GravNav, 2012 [94]	X		X				
Wei et al., 2012 [173]	X						
Dataplay, 2012 [23]	X		X				
Zhang et al., 2012 [196]			X				
VizDeck, 2012 [109]			X				

Usability: Usability is a catch-all metric that is meant to capture the ease of use of the system and is measured in a variety of ways:

- **Task Completion Time:** Measured as the time taken by the user to complete a system specific task, an example of this is found in the Dataplay system [23]. Their goal is to compare two proposed features: autocompleted query correction and user manipulation of the query tree. They use three tasks with varying complexity which they quantify as the number of tweaks. In the first task users are presented with a query tree of students who got As in some courses and are asked to fix the query to find students who got As in all courses. The second and third tasks build on this with additional constraints.

- **Accuracy:** The old contract with databases was that there is unbounded query execution time but the results have to be accurate. In interactive systems, this is flipped, where we now have strict latency requirements but accept approximate answers. This metric is again only applicable for systems employing sampling or providing approximate answers, which is one technique for handling interactivity in large databases. It is used to capture the difference of the approximate values from true values. There are multiple methods for measuring accuracy including precision, recall [148], mean-squared errors, etc.
  Accuracy is measured in Incvisage [150], a system that uses sampling to show the user incrementally improving visualizations. They compare the average mean squared error across iterations against baselines. They also evaluate accuracy in their user study where users are asked to find the min and max or estimate average values from the visualization. The scored accuracy measures the difference from the correct value weighted by the time at which the user submitted.

- **Number of Interactions:** The number of interactions can be measured coarsely as iterations to

Table 2: Metrics for Data Interaction 2012-present

	Human Factors						System Factors		
	User Feedback	No. of Insights	Task Completion Time	Accuracy	Learn-ability	Discover-ability	Latency	Scal-ability	Cache Hit Rate
Skimmer, 2012 [160]		X	X						
Scout, 2012 [164]									X
Martin and Ward, 1995 [129]						X			
Bakke et al., 2011 [29]			X			X			
GestureDB, 2013 [137]	X		X		X	X			
Basole et al., 2013 [33]	X		X		X				
Biswas et al., 2013 [44]	X		X						
MotionExplorer, 2013 [41]	X								
Yuan et al., 2013 [190]	X								
Ferreira et al., 2013 [73]	X								
Cooper et al., 2010 [55]								X	
Immens, 2013 [127]							X	X	
Nanocubes, 2013 [123]							X		
Kinetica, 2014 [152]	X	X			X				
DICE, 2014 [98]	X						X	X	X
Lyra, 2014 [154]	X		X						
Dimitriadou et al. [61]			X				X	X	
SeeDB, 2014 [167]	X						X	X	
SnapToQuery, 2015 [95]				X		X	X		
Kim et al., 2015 [111]								X	
ForeCache, 2015 [36]									X
Zenvisage, 2016 [158]	X			X			X		
FluxQuery, 2016 [64]							X		
Voyager, 2016 [176]	X								
Moritz et al., 2017 [132]	X								
Incvisage, 2017 [150]	X			X			X	X	
Data Tweening, 2017 [110]	X						X		
Icarus, 2018 [148]	X		X	X			X		
Datamaran, 2018 [39]	X								
Tensorboard, 2018 [178]	X							X	
DataSpread, 2018 [39]								X	
Sesame [100]							X		X
Transformer, 2019 [149]	X		X				X		
ARQuery, 2019 [46]			X						

complete a task, as in the Icarus [148] system which amplifies user's input for filling in unreported data, or as number of times an operator is applied, as in the Facetor system [104] that reduces the user's navigation cost in faceted data exploration and reports the number of times the expand and refine operators are used. Similarly, the guided data repair system [183] measures effort as the number of times the user needs to give feedback to the system, albeit via simulations. It can be used in combination with accuracy for computation on uncertain or ambiguous data [72, 191, 192, 193].

– **Number of Insights:** In order to simulate a real world use case, Liu and Heer [126] ask users to perform exploratory analysis and report the number of insights found. This metric should be used with caution since what counts as an insight is subjective.

– **Uniqueness of Insights:** For exploratory systems, allowing the user to make unique discoveries has high value and this can be captured by uniqueness of insights, as done in Scented Widgets [175].

Additional metrics for evaluating visualization systems can be found in [117, 136].

3.3 Metric Selection

Given the variety of metrics, it can be difficult to identify the most appropriate one for a new system. Metric selection is application dependent and should highlight

Table 3: Guidelines for Selecting Metrics

Type	Metric	When to Use
Human Factors	Design Study	For formulating system specifications and evaluation tasks
	Focus Group	To get consensus feedback from a group
	User Feedback	Always
	No. of Insights	Exploratory systems that provide user guidance
	Uniqueness of Insights	Exploratory systems that provide user guidance
	Task Completion Time	Task-based systems
	Accuracy	Approximate and speculative systems
	Number of Interactions	Systems that aim to reduce user effort for a specific task; usually in comparison to a baseline
	Learnability	Complex systems that will be used frequently by experts
	Discoverability	Systems designed for everyday use by naive/untrained users
System Factors	Latency Constraint Violation	Systems where multiple queries are issued consecutively in a short time frame (Section 7)
	Query Issuing Frequency	Devices with high frame rate. (Section 7)
	Latency	Always
	Scalability	Systems that deal with large amounts of data
	Throughput	Distributed systems
	Cache Hit Rate	Systems that perform prefetching

the contributions of the system. While there is no optimal method for metric selection, there are certain best practices for interactive systems:

1. It should cover atleast one metric from system and human factors.
2. Systems targeted at domain specific tasks should perform design studies and focus groups with end-users to formalize needs and requirements.
3. End-users should be able to provide qualitative open-ended feedback at different stages of development.
4. Approximate systems should evaluate accuracy trade-offs with user effort and/or latency of the system. Accuracy or cache hit rate is also recommended for speculative prefetching systems.
5. Depending on if the system is being designed for novice vs. experts, discoverability and learnability should be measured respectively.
6. Systems aimed towards solving a specific task should measure user effort in completing the task, which can be in the form of task completion time, no. of interactions or quality of insights (for exploratory systems).
7. Distributed systems dealing with a large number of datapoints should measure throughput and scalability, along with summarization latency and cognitive load on the user.
8. Gesture and touch-based devices with high-frame rates, where multiple queries are issued one after the other, should measure query issuing frequency and latency constraint violations.

While we have covered a diverse set of metrics, it is possible that newer systems and devices will require novel metrics which have not been used before.

3.4 Takeaways

Tables 1 and 2 show metrics as used by different systems and is meant to provide an overview of which metrics occur together. For example the trade-off in accuracy and latency is exemplified by the fact that latency is always measured with accuracy (at least in the limited number of papers that report it and have been included in our survey). However, these tables should not be used to compare systems in terms of their evaluation coverage since each system solves a different problems and hence requires different metrics to convey its contributions. Table 3 summarizes the appropriateness of each metric. In later sections we demonstrate metric selection and evaluation via case studies.

4 Confounding Factors during User Studies

As mentioned above, interactive systems often require user studies as part of their evaluation. User studies can be more complicated than system evaluations since humans bring with them various biases and inconsistencies which need to be accounted for, to draw sound and reproducible conclusions. There is a wide body of research in HCI and cognitive science that study these factors in detail [27, 32, 69], here we discuss some common pitfalls and factors to consider when designing studies. The interested reader should look deeper at relevant references.

4.1 Study Design

There are various considerations when designing a user study. We describe different design decisions, followed by how they impact the user.

4.1.1 In-person vs. Remote

User studies can take place in-person or remotely. In an **in-person** setting, the study takes place in front of the researcher. This provides the maximum amount of control and observations to the researcher. The device used to test, external conditions such as noise, the participant's hand position, etc., can all be controlled (e.g., [149]). Think aloud protocols also usually require in-person studies (e.g. [126]). The main shortcoming of in-person studies is that it limits the number and population of participants. Researchers usually do not have resources to find diverse participants (non-college students might require higher compensation for their time).

In a **remote** setting, the study is done online, often through crowdsourcing platforms such as Amazon's Mechanical Turk [84]. This has opposite trade-offs to the in-person case. It is possible to recruit a large and diverse population, however, there is limited control on external conditions. Another shortcoming is that there could be cases of participants randomly selecting answers, which need to be filtered post-hoc [59]. This option is ideal when studying a specific population phenomenon without comparing against a control (e.g., [56, 180]).

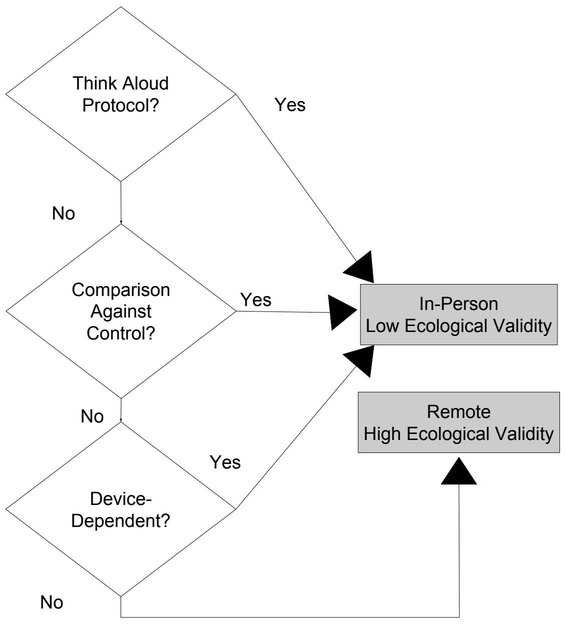

Think Aloud Protocol?
Yes
No
Comparison Against Control?
Yes
In-Person Low Ecological Validity
No
Remote High Ecological Validity
Yes
Device-Dependent?
No

Fig. 4: Guidelines for in-person vs. remote study design.

4.1.2 Within-Subject vs. Between-Subject

Another important design decision is whether to do a within-subject or a between-subject study. In a **within-subject** study, the same group of users are exposed to a condition more than once such as doing the same task in two systems, interacting with the same dataset in two systems, or doing different tasks/datasets in the same system. Such a design is needed when the task at hand depends on some inherent ability of the user. For example, when doing exploratory analysis, the definition of an insight is user-dependent. Thus, when comparing exploratory analysis systems, e.g., evaluation of the Voyager system [176], the same set of users are required.

On the other hand, in a **between-subject** experiments (e.g., Related Worksheets [29]), two sets of users are employed such that each set is exposed to a unique set of conditions. This design should be preferred whenever possible as it reduces carry-over effects (discussed below). Users should be evenly split in a random manner, to avoid any biases based on their demographics. Effort should be made to equalize instructions and conditions as much as possible between the control and the test set.

4.1.3 Simulation

In certain situations, user traces can be used to simulate interactions on systems. This is appropriate when results depend only on plausible user interaction sequences (e.g.,RAP [188], BinGo [38]). The time for each interaction can then be estimated via various HCI models such as Fitts', GOMS and ACT-R [42,48,75,96,161]. Different versions of the models are available for different input modes [67, 87, 119], devices [25], and users ability [26, 147]. The interactions in combination with the appropriate model can be used to simulate user behavior and measure metrics such as session duration and system latency for each interaction. This is demonstrated in evaluation of the Usher system (Chapter 2 in [52]).

Recently, there have been efforts in the community towards creating a benchmark for interactive systems by simulating user behavior. IDEBench [65] allows users to create workloads, i.e., user interactions, based on predefined navigation patterns, but this is not adequate for all use cases. On the other hand, collecting and sharing real user traces has its own challenges as outlined in [35]. Thus, there are many caveats to the validity of simulation studies, however, if appropriate, they are more efficient than user studies.

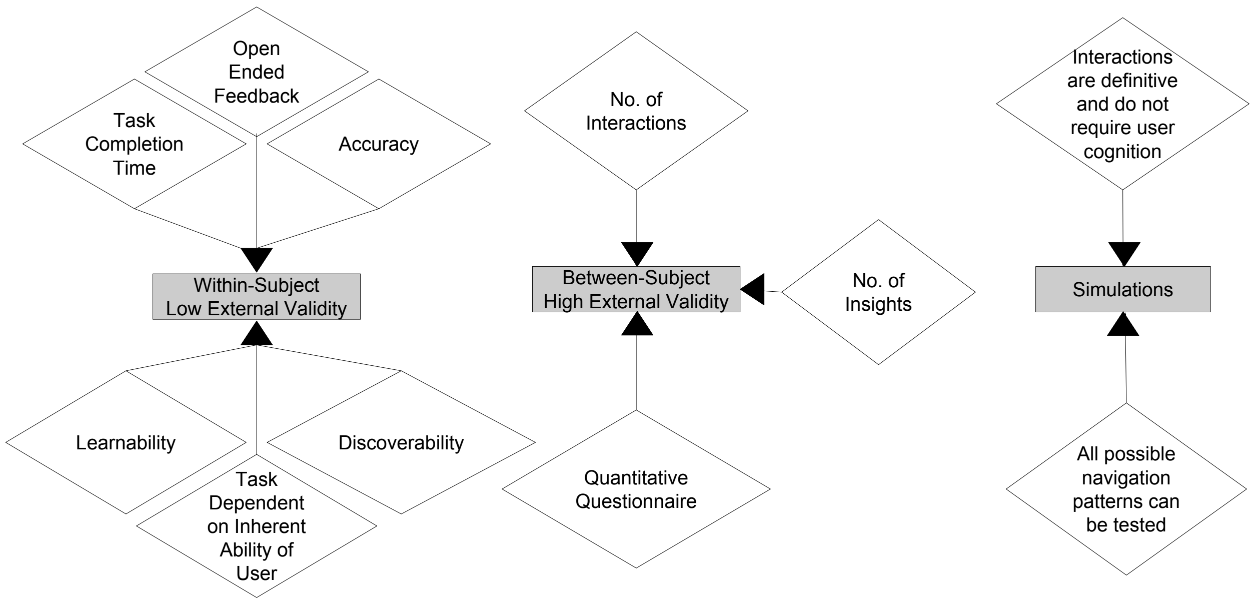

Open Ended Feedback
Task Completion Time
Accuracy
Within-Subject Low External Validity
Learnability
Discoverability
Task Dependent on Inherent Ability of User
No. of Interactions
Between-Subject High External Validity
No. of Insights
Quantitative Questionnaire
Interactions are definitive and do not require user cognition
Simulations
All possible navigation patterns can be tested

Fig. 5: Guidelines for study design based on metrics.

4.2 Study Validity

The design of a study affects its validity, i.e., the soundness of its results. Padilla's [141] position paper discusses three aspects to the validity of a user study.

4.2.1 Ecological Validity

Ecological validity refers to how closely the study conditions match real-world use cases. There is a trade-off between the ecological validity of a study and the amount of experimental control available to the researcher. A study with high ecological validity would involve deploying the system and logging user interactions during real usage. In this case, the researcher is not able to control for external conditions such as distractions, lighting, study device, etc. Whereas if the study is done in a controlled environment, it might not reflect real-world usage scenarios. Identifying the ideal amount of control while maintaining ecological validity depends on availability of resources and contributions of the system. If the system is an improvement on an existing baseline, which is validated via a user study, then a setting with finer control is more apt (e.g., [46]). On the other hand, if the goal of the evaluation is to demonstrate the usefulness of the various features of the system, then a real-world study might be more appropriate (e.g., [178]). In conclusion, real-world studies with high ecological validity should be conducted if possible, i.e., external conditions do not have a significant effect on study tasks.

4.2.2 External Validity

External validity refers to the ability to generalize the results of the study to populations outside of the ones tested in the study. There is a trade-off between external validity and convenience of population sampling. College students are a common population for in-person studies due to their availability. In cases where a certain level of expertise is required, a qualifying task can be used in the recruitment process. Proxies such as enrollment in a course [137] or self-reported skills via questionnaires [110] can be used to stratify users via skill level. High external validity is required to ensure generalizability to the general public. However, certain systems are targeted towards specific users, such as domain experts [148]. In these cases, it is enough to ensure that the results are applicable to set of targeted users, and efforts should be made to recruit only from the target population [148]. Some threats to external validity include:

– **Learning**: As mentioned earlier, in a within-subject study, each user is exposed to the same condition (task or system) twice. This leads to the user doing slightly better on the second system simply because they are familiar with the task and due to no merit of the system. In order to combat this, randomization or counterbalancing can be used. In randomization, the order in which a participant sees the control and the test is randomly assigned(e.g., [158]). In counterbalancing, participants are randomly assigned to group 1 or group 2, and everyone in group 1 sees the

Table 4: Cognitive Biases during User Studies, definitions in [60]

type	bias	mitigation measures
Participant's Biases	**Social Desirability Bias**: tendency of humans to perform actions that will make them likable to others, e.g., support the researcher's hypothesis	Follow externally approved scripted language when interacting with study participants and make every attempt to not disclose tested hypothesis.
	Anchoring Effect: fixating on a specific piece of initial information and basing all decisions on it, e.g., preference for the first system the user is exposed to	Randomization and counterbalancing of conditions
	Halo Effect: positive characteristics based on positive appearance. e.g., in a user study questionnaires, participant rates all aspects of a system highly because it looks nice or has one good feature.	Break down study tasks into fine-grained task and attempt to have every participant evaluate a single feature.
	Attraction Effect: clustering of points in a scatter plot affect the user's ability to choose between items on the pareto front [59]. This can affect a user's accuracy in visualization studies with scatter plots.	Modify study procedure: Example for mitigating effect in scatterplots described in [59]
Experimenter's Bias	**Framing Effect**: selecting an option because of how the sentence is framed, e.g., researcher can influence the participants' choices by framing questions to favor the system being tested.	Have study verbiage externally reviewed
	Selection Bias: selecting participants that are likely to perform favorably on the condition being tested. For example, selecting only iPhone users when the study device happens to be an iPhone, even though this is not representative of the general population.	Randomly assign participants to the study, before collecting any demographics/background information which might affect their performance in study
	Confirmation Bias: tendency of the researcher to see results that confirms their hypothesis	Practice high transparency in results, e.g., make study material/all user comments available

control and the test in the same order, while group 2 sees the opposite order (e.g., [176]). If different metrics are being tested for the same system, different users should be employed as in the learnability and discoverability experiments for SnapToQuery [95].

– **Interference**: In a within-subject studies, this refers to the user's performance on the second task being affected due to exposure to the first task. This differs from learning in that their performance in the second task could diminish. For example, if both the control and the experimental system are new to the user, they may perform poorly on the second by confusing functionalities with the first. Again, randomization or counterbalancing can help account for this. However, if the effects are asymmetric, i.e., one user shows improvement while the other one shows deterioration, it makes it hard to draw conclusions.

– **Fatigue**: Long tasks can lead to users performing poorly towards the end, due to fatigue. Hence, tasks need to be broken into small chunks, with adequate breaks in between (e.g., [30]).

4.2.3 Construct Validity

Construct validity refers to if the metric being used actually measures the correct factor. For example, in the Transformer system [149] form completion time is used as a proxy for physical effort, when a more fine-grained metric such as number of interactions could have been used. Similarly, the GOMS model [96] is often used for estimating total completion time, including mental effort. Other studies use task completion time and accuracy as a proxy for mental effort [141]. However, there are various standard methods to draw from the area of cognitive science such as neuro-imaging and the dual-task paradigm [162], which are more effective at measuring cognitive load. The dual-task paradigm refers to measuring the task productivity of a user in the presence of a secondary task in addition to the primary interface query task [162]. Measuring the user's physiol-

Table 5: Case Study Summary. The case studies span different devices, interfaces, interaction techniques, and queries.

name	**device**	**query interface**	**interaction techniques**	**trace**	**query**
inertial scrolling (Section 6)	touch(trackpad)	scroll	browsing	{timestamp, scrollTop, scrollNum, delta}	`select`, `join`
crossfiltering (Section 7)	mouse, touch(iPad), gesture(leap motion)	slider	linking & brushing	{timestamp, minVal, maxVal, sliderIdx}	`count aggregation`
composite interface (Section 8)	mouse	textbox, slider, checkbox, map	filtering & navigating	{timestamp, tabURL, requestId, resourceType, type, status}	`select`, `join`

ogy using methods such as fMRI [166] and galvanic skin response [139] are also options. These methods are more expensive than measuring speed and accuracy, hence there is a trade-off between construct validity and convenient metrics.

4.3 Cognitive Biases

Finally, humans have many cognitive biases which can affect the validity of a study. Cognitive bias refers to distorted representations of reality in the minds of individuals, which occur systematically and involuntarily [60, 81]. There are over one hundred and eighty cognitive biases [40], only some of which might affect user studies, most often in an in-person setting. While there have been efforts to create frameworks to study them in visualization [60, 165], they have not been discussed in the context of user study evaluations. Cognitive biases from the experimenter's and the participant's side are summarized in 4, along with mitigation measures for them. It is good practice to follow these measures for all user studies.

5 Guidelines for Evaluation

There are a variety of metrics employed by systems based on their use-case, but no set of standard metrics are adequate for capturing the whole picture, which makes it difficult to compare two systems. Given the variety of interfaces and their uniqueness outlined in Section 2, this is understandable. However, comparisons become easier if certain principles, such as the following, are practiced when evaluating new systems.

1. System designers should take *behavior-driven* optimizations into consideration, leveraging the user's session characteristics when designing and evaluating interactive systems.

2. Metrics should maximize the coverage of query types (e.g., `select`, `join`, `aggregation`) and interaction techniques [107, 156, 174, 189] (e.g., filtering, linking & brushing), since each of these generate unique workloads.

3. Evaluation should be done from a human as well as a system perspective (Figure 1).

4. User study tasks should simulate *real-world* use cases on *real* datasets to ensure high ecological validity, because interactive systems are used everyday as opposed to data warehouses used for business transactions.

5. Participant order should be randomized between tasks to minimize learning effects or interference and ensure high external validity.

6. Tasks should be granularized and its language externally reviewed to mitigate experimenter and participant biases.

7. When studying user behavior, some studies recommend a minimum of 10 users [69, 118]; however, this number strongly depends on the nature of tasks and variability of the interaction itself.

8. Evaluations should cover a variety of workloads, e.g., different scenarios, data distributions, data sizes, etc.

We demonstrate these principles (except for the last one which can be controlled synthetically [65]) by walking through three case studies. The behaviors studied and metrics used in the case studies are summarized in Table 6. Since our user studies do not require specific expertise, we do not report user profiles (e.g., age, gender, major, etc.) for any of them.

5.1 Limitations of Case-Studies

Our case-studies are meant to provide concrete examples of applying *behavior-driven* optimizations in interactive systems and to demonstrate the new metrics we proposed (latency constraint violation and query issuing frequency). Consequentially, the following should be kept in mind:

– The generalizability of the study results to other users requires additional validation studies, and remains to be tested. We report standard deviation and ranges, wherever applicable, to give the reader a sense of the variability of results.
– The case studies were selected to cover a variety of devices, interface widgets and queries (Table 5). The results are not meant to be representative of other interactive systems and applications. In fact, each application requires its own study to inform *behavior-driven* optimizations based on its target users. Our aim is to show how user behavior can be translated to optimizations.

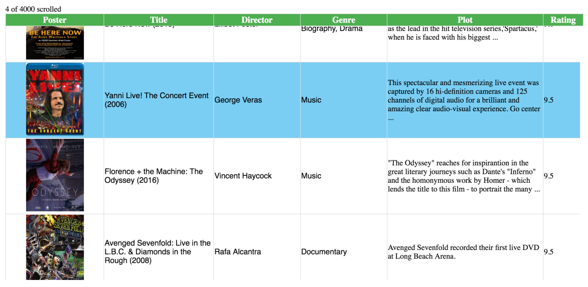
Table 6: Behaviors and Metrics in Case Studies

interface	**behavior**	**performance**
inertial scrolling (Section 6)	scrolling speed	latency constraint violation
	no. of backscrolls	latency
crossfiltering (Section 7)	sliding behavior	query issuing frequency
	querying behavior	latency
		latency constraint violation
composite (Section 8)	query interface	exploration time
	zooming	
	dragging	data request time
	filter conditions	

6 Case Study 1: Inertial Scrolling

Inertial scrolling is used for browsing information when the result set does not fit on screen [92, 160], which is common for query results in large data warehouses as well as in search engines. Inertial or momentum scrolling [135] is a feature that helps users scroll smoothly and is widely adopted in touch based devices, e.g., smartphones, trackpads, etc. Its salient feature is that there is an acceleration when the user scrolls. Further, when the user stops scrolling, the screen stops gradually as opposed to immediately as in traditional scrolling.

In scrolling interfaces, large result sets are impossible to load at once since they do not fit in memory. Moreover, the user cannot digest the entire result set at once. Different loading strategies are used to address this, such as lazy loading implemented using `LIMIT` and `OFFSET`. We conduct a user study to collect scrolling traces under ideal conditions, i.e., without loading latency, and then use these traces to optimize loading strategies.


4 of 4000 scrolled

Poster	Title	Director	Genre	Plot	Rating
			Biography, Drama	as the lead in the hit television series,'Spartacus,' when he is faced with his biggest ...	
	Yanni Live! The Concert Event (2006)	George Veras	Music	This spectacular and mesmerizing live event was captured by 16 hi-definition cameras and 125 channels of digital audio for a brilliant and amazing clear audio-visual experience. Go center ...	9.5
	Florence + the Machine: The Odyssey (2016)	Vincent Haycock	Music	"The Odyssey" reaches for inspirantion in the great literary journeys such as Dante's "Inferno" and the homonymous work by Homer - which lends the title to this film - to portrait the many ...	9.5
	Avenged Sevenfold: Live in the L.B.C. & Diamonds in the Rough (2008)	Rafa Alcantra	Documentary	Avenged Sevenfold recorded their first live DVD at Long Beach Arena.	9.5

Fig. 6: Inertial Scrolling Experimental Interface. Highlighted section denotes movie selected by user.

Dataset: We selected the top rated 4000 tuples (similar to the number of tuples used in prior studies [160]) from the IMDB [7,12] movie dataset, with 6 attributes (poster, title, director, genre, plot, rating). All tuples are preloaded to the browser to avoid loading latency.

Task & Users: 15 users were asked to *skim* all 4000 tuples and select interesting movies by scrolling through a MacBook [21] trackpad to increase ecological validity of the study. Each user is trained on scrolling on the touchpad before the task.

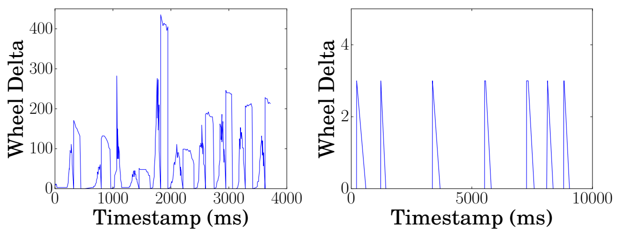
Trace & Query: For each scroll and wheel event, we store the current timestamp, scrollTop [15] which is the number of pixel scrolled upward, number of tuples scrolled, and delta [4] which is the accelerated scroll amount. The query can be a simple `select` query (Q1), which selects 100 tuples every time:

```
SELECT poster, title || '(' || year || ')',
director, genre, plot, rating
FROM imdb
LIMIT 100 OFFSET 100
```

or a complex streaming `join` query (Q2) [103,168] when the information comes from different streaming sources.

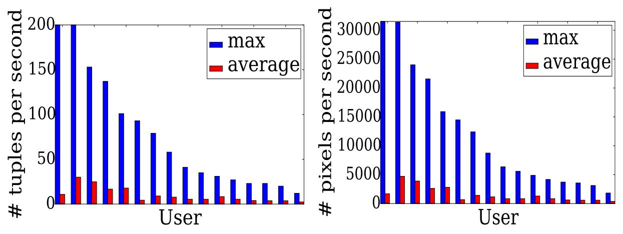
```
SELECT poster, title || '(' || year || ')',
director, genre, plot, rating
FROM (
(SELECT id, rating
FROM imdbrating
LIMIT 100 OFFSET 100) tmp
INNER JOIN movie ON
tmp.id = movie.id
)
```

In the workload, we vary the `limit` and `offset` numbers for both queries.



(a) with inertia. (b) without inertia.

Fig. 7: Scrolling with / without inertia: The wheel delta indicates the distance scrolled. As can be seen from the plot, the user scrolls much faster with inertial scrolling (y-axis scale: 400 vs 4), rendering lazy loading ineffective.



(a) Scrolling Speed in # tuples (b) Scrolling Speed in # pixels

Fig. 8: Scrolling Speed: Maximum and average scrolling speed measured as the number of tuples and as pixels per second for each user sorted by the maximum. Figure b can be transformed into one with the number of tuples per second when the height of tuple is different.

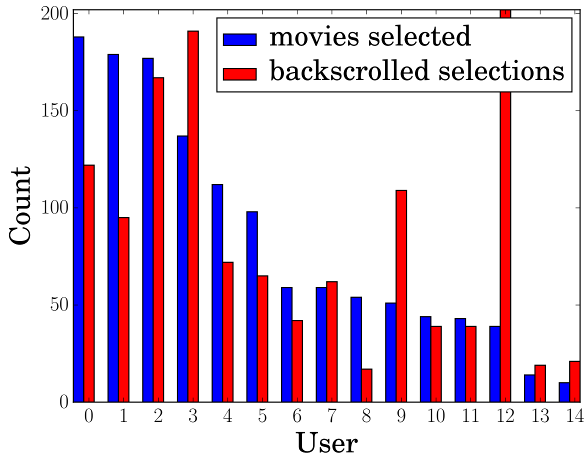


Fig. 9: Comparison of number of selected movies and number of movies selected with backscrolls. In some cases, the number of backscrolls is larger than the number of selected movies, indicating that the user had to scroll back and forth multiple times to select a movie. This shows the need for additional usability optimizations with inertial scroll.

6.1 Behavior Analyses

We study the scrolling speed and number of back scrolls to capture the user's difficulty in selecting targets.

Scrolling Speed: Figure 7 shows a representative wheel delta (i.e., distance scrolled) against the timestamp from part of one user's trace. The figure indicates that the user scrolls larger distances (y-axis scale: 400 vs 4) with inertial scrolling than regular scrolling. Hence, techniques such as lazy loading [37, 78], i.e., fetching more items once the user reaches the bottom of the results do not work. The user often reaches the end of the page before items are loaded, which increases wait time and reduces usability.

No. of Backscrolls: We also observed that the increased momentum often causes users to scroll past a movie that they want to select, requiring them to scroll back to select it (Figure 9). These findings verify the needs of designing interfaces that reduce information loss, as discussed in previous research [92, 160].

6.2 Evaluation

In this section, we evaluate query workloads generated from the user study over PostgreSQL. For each query, the reported query execution time is the maximum of 10 runs. To decide on number of tuples to fetch, we compared {12, 30, 58, 80} tuples. These values correspond respectively to lower bound of maximum, upper bound of average, median of maximum, and mean of maximum scrolling speed (measured in no. of tuples) of 15 users (Table 7). We compare two alternative prefetching strategies to lazy loading:

– *Event fetch:* For every scroll event, the system checks whether there are enough prefetched items in cache.

Table 7: Statistics for Scrolling Behavior

	range, mean, median of maximum scroll speed	**range, mean, median of average scroll speed**
# pixels / sec	[1824, 31517], 12556, 8741	[369, 4717], 1580, 848
# tuples / sec	[12, 200], 80, 58	[2, 30], 10, 5

If not, then more items are prefetched. Since a scroll event is triggered every 15–20 ms, this method adds heavy computation burden on the browser. For our experiments, we set cache limit to product of tuples to fetch and query execution time.

– *Timer fetch:* Items are prefetched at regular time intervals, e.g., 20 items every second. We set the fetching interval to 1 second and fetch a fixed number of tuples.

Latency: An obvious metric for comparing these strategies is latency in loading tuples, shown in Figure 10. As can be seen *event fetch* is insensitive to the number of tuples fetched and keeps at a reasonable latency ~ 80 ms. In *timer fetch*, when number of tuples fetched is low, faster users can wait up to ~ 60 seconds. However, it decreases with increase in number of tuples and achieves zero latency when the tuples fetched equals the median of max scroll speed.

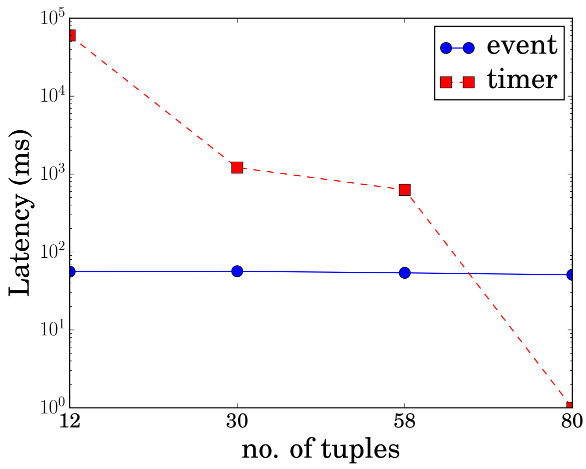


Fig. 10: Average latency of 15 users over different number of tuples fetched. *Event fetch* is insensitive to the number of tuples. *Timer fetch* decreases linearly reaching zero latency at median of max scrolling speed.

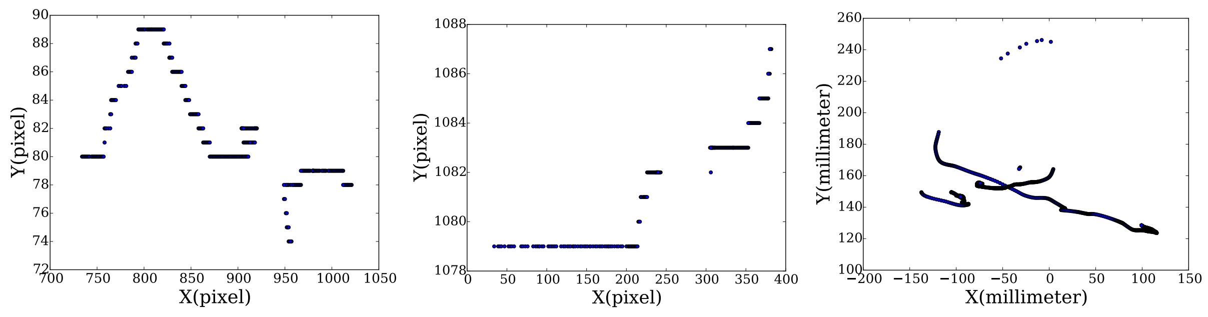
Latency Constraint Violation: For inertial scroll, a latency constraint violation happens if the number of tuples scrolled is greater than the cumulative number of tuples cached. This captures the number of times users observe a latency, i.e., users have to wait for tuples to load. Table 8 shows the number of users (out of 15) who observed latency constraint violation. The total number of constraint violations are also shown for each of the four values of cached tuple sizes. For *timer fetch*, violations only occur in few users' traces. When we increase the number of tuples fetched, the number of violations rapidly decreases. However, *event fetch* is less sensitive to the number of tuples fetched since it only fetches more tuples when the number of currently cached tuples is less than the cache limit. Because of the acceleration, at some points, the number of tuples scrolled may be greater than the number of cached tuples.

Table 8: Latency Constraint Violations for Event & Timer Fetch: Almost all users had a violation for all cache sizes for event fetch. For timer fetch, the number of violations decreases with number of tuples cached, and it performs better than event fetch in general.

# tuples fetched	**12**	**30**	**58**	**80**
# users (event)	15	15	15	14
# users (timer)	3	1	1	0
# no. of violations(event)	2203	840	457	167
# no. of violations (timer)	767	2	1	0

6.3 Takeaways

Through behavior analyses, we demonstrate that inertial scrolling has a much heavier query workload (200 tuples per second) than traditional scrolling, requiring optimizations in loading techniques. To address this, we compare two prefetching techniques and show that by picking the right parameters based on behavior analysis, we can achieve zero latency. The results reported here along with the generated workloads can be used to optimize prefetching parameters for different screen sizes and databases. The pixel statistics in Table 7 allows the developer to calculate tuple ranges for different layouts. Thus, new interfaces that change the result of user interaction, i.e., in this case queries generated, require a study of user behavior to inform design decisions.



(a) Mouse (b) Touch (c) Leap Motion

Fig. 11: Traces of a user specifying a range query on three devices: The leap motion presents more jitter than mouse and touch.

7 Case Study 2: Crossfiltering

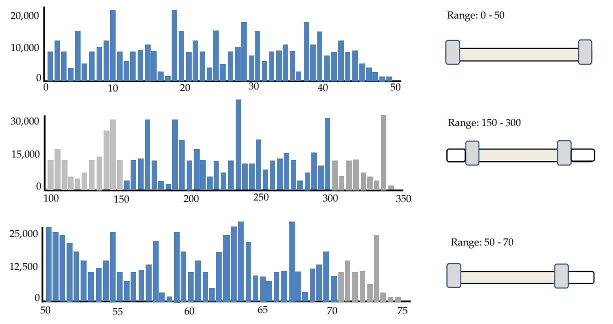


Fig. 12: Crossfiltering Interface. Each histogram corresponds to one attribute and range slider. The blue indicates the filtered range for each attribute. Histograms for other attributes are updated synchronously while the user is manipulating one slider.

Crossfiltering (or brushing-and-linking [107]) is a popular visualization concept which allows users to issue multi-dimensional filters to a subset and explore the dataset in a coordinated view arrangement [3, 95, 123, 127, 172]. This interface is used to quickly gain insights from large multidimensional datasets, instead of having to wait for hours to get results from a query [99, 133]. Figure 12 shows the UI for our experiments, where each histogram corresponds to one attribute and is controlled by a range slider.

Dataset: We used a 3D road network dataset from the UCI ML repository [20, 106], which has 3 attributes (`longitude`, `latitude` and `height`) and 434,874 tuples.

Task & Users: We recruited 30 users (10 per device) and asked them to specify range queries by moving the handle to a specific position with mouse, iPad or Leap Motion. The crossfiltering library [3] was used, which can support extremely fast (<30ms) interactions for up to one million records. Hence, users did not perceive any query latency.

Trace & Query: For each sliding event, we collect the current timestamp, value range, and slider index. At each timestamp, two queries are issued concurrently. Each query is used to calculate a histogram. One query example is shown below:

```
SELECT ROUND((y - 56.582) / ((57.774 - 56.582) / 20)),
COUNT(*)
FROM dataroad
WHERE x >= 8.146 AND x <= 11.2616367163
AND y >= 56.582 AND y <= 57.774
AND z >= -8.608 AND z <= 137.361
GROUP BY ROUND((y - 56.582) / ((57.774 - 56.582) / 20))
ORDER BY ROUND((y - 56.582) / ((57.774 - 56.582) / 20))
```

Configuration: We run queries on PostgreSQL [13] and MemSQL [9] on a Linux machine (Intel Core i5-4590 CPU @ 3.30GHz × 4, 15.6 GiB Memory). PostgreSQL is a popular disk-based database while MemSQL is an in-memory database. We choose these two databases since we wanted to compare the interactive performance for disk-based and in-memory databases. In Python, the global interpreter lock ensures that only one thread runs in the interpreter at once [6]. In order to issue multiple queries at the same time, we fork and execute multiple Python processes at the same time, where each process has an independent database connection.

7.1 Behavior Analyses

For the crossfiltering interface, we study the sliding behavior, i.e., the manner in which the user interacts with the sliders, and how this affects the querying behavior, i.e., the manner in which queries are issued.

Sliding Behavior: Figure 11 shows representative traces on three different devices from the same user. As seen on the trace, Leap Motion triggers unintended gestures, resulting in heavy workloads from noisy queries.

Querying Behavior: Since crossfiltering uses sliders as the query interface, queries are issued one after another as the handle is moved continuously. When the user interface frame rate per second is $20ms$, every second about 50 queries are issued. In coordinated views, such as crossfiltering, about $50(n-1)$ queries are issued, where n is the number of dimensions, leading to heavy workloads. We propose two optimization algorithms to reduce this: *Skip* and *KL Optimization*, which can be applied to any interface that has high query frequency.

Skip: In crossfiltering, no dependency exists between adjacent queries, since each represents a separate range query, and unlike inertial scroll where the user browses serially, a user does not necessarily look at ranges serially. Thus, queries can be *skipped* to improve performance. If the query execution time is high, the user might have issued a second query before the first one completes (Figure 2). Further, in exploratory data analysis, the user may abandon issued queries if initial findings or partial results [85,150] do not match the hypothesis being tested. In this case, previously issued queries for that hypothesis should also be abandoned, i.e., the current query run by the database may already have been skipped by the user. Based on this, a natural optimization would be to skip previous queries once a new one has been issued.

Algorithm 1 Optimization: Skip

```
1: for queryGroup in queryGroups do
2:     # busy wait
3:     while True do
4:         if curTimestamp > queryGroup['timestamp'] then
5:             # at same timestamp, multiple queries are issued in coordinated view
6:             sqls = queryGroup['sqls']
7:             # proc1 and proc2 is used for synchronization
8:             if proc1.value == 0 and proc2.value == 0 then
9:                 # fork multiple processes, run multiple queries concurrently, subprocesses don't block main process
10:                procPoll.map_async(issueQuery, sqls)
11:            break
12:            curTimestamp = time.time()
```

KL Optimization: Most adjacent queries have same or similar results, since users make iterative changes. Hence, an alternative optimization would be to only execute queries whose results differ to a certain extent from the previous query. Hash-based methods [79], sampling-based methods [51,76,145], wavelets-based methods [169] etc. can be used for fast approximation of histograms, i.e., result sets, without running the query. We use the Kullback-Leibler (KL) [116] divergence to measure the difference between two histograms, before queries are sent to the database. If KL = 0, then the two histograms are the same. We approximate KL by quantizing and comparing histograms $\mathcal{T}$ and $\mathcal{T}'$ at each of the n bins:

$$KL(\mathcal{T},\mathcal{T}') = \sum_{x=0}^{n} \mathcal{T}(\frac{x}{n}) \times ln(\frac{\mathcal{T}(\frac{x}{n})}{\mathcal{T}'(\frac{x}{n})}) \tag{1}$$

In [181], the authors present initial work on measuring human's perception error of with KL, which can be used to determine the KL threshold.

Algorithm 2 Optimization: KL Approximation

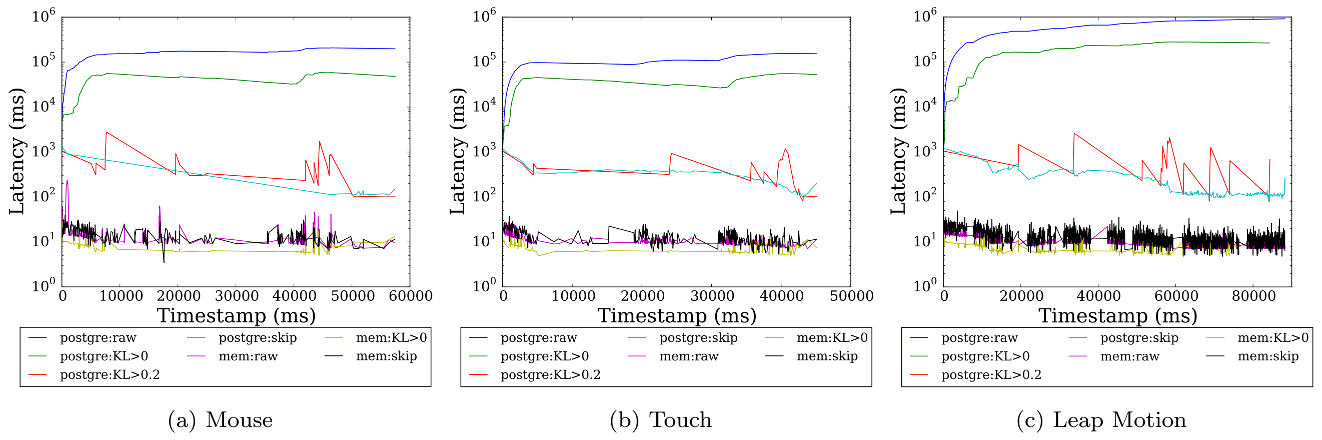
```
1: for queryGroup in queryGroups do
2:     # busy wait
3:     while True do
4:         if curTimestamp > query['timestamp'] then
5:             sqls = queryGroup['queries']
6:             id = queryGroup['id']
7:             # KL function is used to calculate the KL between current query and previous query
8:             if KL(id, preId) > KLThresh then
9:                 # fork multiple processes, run multiple queries concurrently, subprocesses block main process
10:                procPoll.map(issueQuery, sqls)
11:            break
12:            curTimestamp = time.time()
```

7.2 Evaluation

We evaluate *query issuing frequency and latency* over three variable factors: database (PostgreSQL, MemSQL), device (mouse, touch, leap motion), and optimization method (*raw*, KL>0, KL>0.2, *skip*), where *raw* means running all queries.

Query Issuing Frequency: To demonstrate the high frame rate in interactive systems, we plot the frequency histogram of query issuing intervals from a representative trace for each device. Figure 14 provides us with the following insights:

1. The number of queries issued by leap motion is much larger than mouse and touch (y-axis scale: 2500 vs 120), which verifies the instability and sensitivity of this device.



(a) Mouse (b) Touch (c) Leap Motion

Fig. 13: Latency for Different Devices under Different Factors. MemSQL can maintain a latency 10∼50ms. After some optimization (KL=0.2 or skip), PostgreSQL can maintain a latency 100∼1000ms. Leap motion has more dense workload than the mouse and touch.

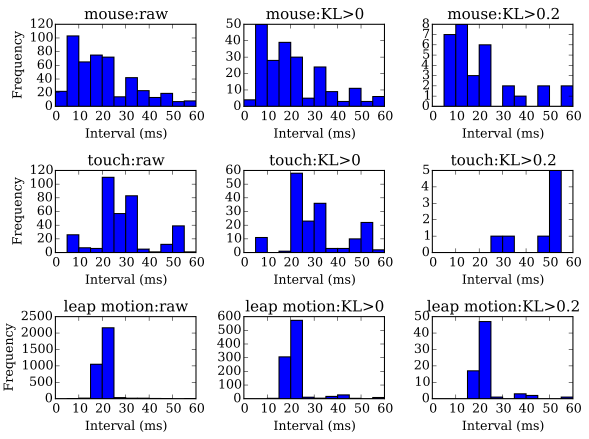


Fig. 14: Frequency Histograms of Query Issuing Intervals

2. If we only issue queries when there is a difference in result sets (KL>0), we can drastically reduce the number of queries issued.

3. Compared to the leap motion whose frequencies concentrate on 20ms - 25ms, the shape of the bell is broader for mouse and touch. This indicates that leap motion has a stricter latency requirement than mouse and touch.

Latency: We plotted the latency of each user under each condition and device, and show a representative one in Figure 13. It is clear from the figure that MemSQL can maintain a latency of 10∼50ms for all three algorithms. For KL>0, it can maintain an interactive performance ∼10ms. The *skip* strategy doesn't work better than the *raw* since the latencies of most queries are below the query issuing intervals. On the other hand, PostgreSQL's latencies are beyond 10*s* for *raw* and KL>0. When we skip queries and run queries with KL>0.2, it can keep a latency 100∼1000ms (< 1sec).

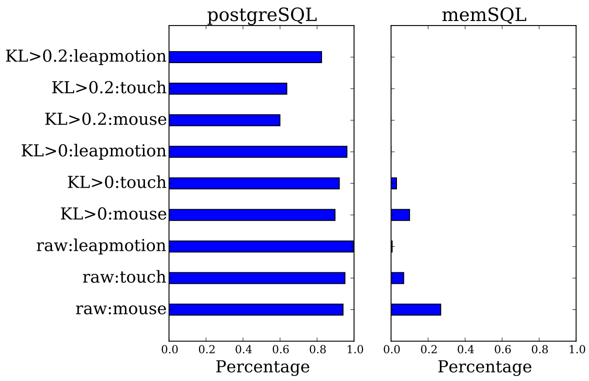


Fig. 15: Percentage of Queries that Violate Latency Constraint. KL>0 is enough to improve performance in MemSQL, but KL>.2 is required for observable differences in PostgreSQL.

Latency Constraint Violation: For crossfiltering, a latency constraint is violated if the user issues a second query before the results of the first are returned. This is shown in Figure 2, where Q1, Q2, Q3 all violate the constraint since their execution time is beyond the query issuing interval. Further, these violations cascade, since the delay in Q1's execution adds to the delay in Q2's execution and so on.

We evaluated the percentage of violations for the skip and KL optimizations on the three devices and two databases, shown in Figure 15. As expected, MemSQL has a lower percentage of violations than PostgreSQL. When we issue queries with KL>0, we can reduce about half of violated queries for MemSQL. For PostgreSQL, however, queries have to be issued with KL>0.2, for observable differences. In fact with KL>0.2, we achieve 30% decrease for mouse and touch and 17% decrease for leap motion for PostgreSQL. We find that the running times of the violated queries are between $150 - 500ms$ for PostgreSQL, and < 25ms on MemSQL.

7.3 Takeaways

First, we demonstrate that Leap Motion has more jitter than mouse and touch, which can generate noisy, unintended, and repeated queries. Second, by analyzing the querying behavior, we propose two behavior-driven optimization techniques – interface (skip) and result-driven (KL>0 or KL>0.2). Through performance experiments, we show that MemSQL can maintain interactive performance even with raw workloads while PostgreSQL can maintain subsecond performance after some optimizations (skip or KL>0.2). This case study motivates and demonstrates the use of behavior analysis in system optimizations.

8 Case Study 3: Composite Interfaces

Most interactive systems [14, 16] allow users to explore datasets through composite interfaces, i.e., multiple query interface widgets (e.g., text box, slider, map, etc.), which we try to simulate in this case study. Multi-interface querying is popular for browsing geo-spatial data [1], which requires large-scale dataprocessing [36, 140]. A key technique for improving performance in interactive systems is speculative prefetching. Many methods have been proposed for prefetching, such as Markov chains [114, 120], heuristic methods [188], and data-driven characteristics [36]. However, they have either only been evaluated using synthetic workloads [114,120, 188] and prototype systems [114], or they have focused on one query interface, like map [36,188], which is easier to predicate than multiple query interfaces.

In this case study, we use a popular commercial accommodation website – Airbnb [1](shown in Figure 16) as the experimental setup since it satisfies the requirement of having multiple query interface widgets and is freely accessible. Our goal in this case study is to analyze traces of the user's interaction with different widgets on *a real commercial website*, in order to inform parameters on *behavior-driven* prefetching techniques.

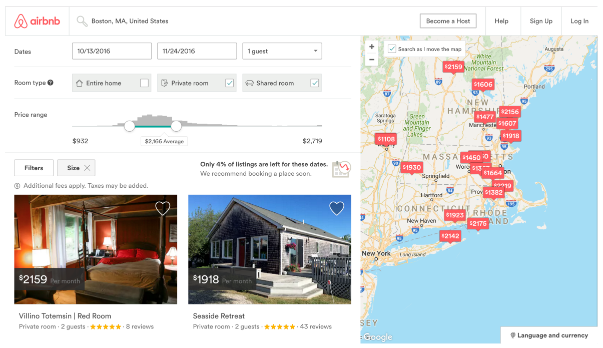


Fig. 16: Airbnb Interface which incorporates different query widgets, e.g., slider, map, etc.

They can also serve as a public benchmark to evaluate current speculative prefetching techniques.

Task & Users: We recruited 15 students and asked them to think of an ideal vacation and then use Airbnb to book short-term housing for it. In order to get enough traces, we asked users to spend at least 20 minutes on it.

Trace & Query: We wrote a browser extension to collect HTTP requests, tab URLs, and events. In order to get a clean requests collection, we only collected requests whose method was GET. For these requests, we collected data, image, and map requests, ignoring other requests such as status update. For each HTTP request, we further recorded its request id, timestamp before the request is made, timestamp after request is collected, resource type (e.g., image, XMLHttpRequest, etc.), and its URL. The Airbnb tab URL itself can be regarded as a query:

```
https://www.airbnb.com/s/Alabama--United-States?page=1
&source=map&airbnb_plus_only=false
&sw_lat=27.73476540653654&sw_lng=-91.1290339635413
&ne_lat=36.80215351286778&ne_lng=-82.0982722447913
&search_by_map=true&zoom=6&checkin=12%2F14%2F2016
&checkout=12%2F18%2F2016&guests=3
&room_types%5B%5D=Entire%20home%2Fapt&price_min=10
&price_max=56&allow_override%5B%5D=
```

From this URL, we can get query parameters such as place, check-in time, check-out time, map, southwest longitude.

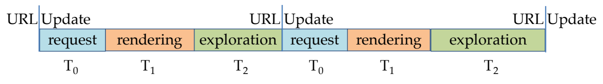


Fig. 17: Exploration Process

8.1 Behavior Analyses

Exploration Process: The user' s exploration process can be modeled as shown in Figure 17. The session starts when the tab URL is updated. Then the browser waits for the data and resources of the request (request time T_0), and renders them (rendering time T_1) once they are received. Finally, the user spends some time exploring the results and decides the next query (exploration time T_2). We estimate the rendering time by listening to mutation events (e.g., DOMNodeInserted, DOMNodeRemoved, DOMSubtreeModified) [10] and recording their timestamps.

Table 9: Percentages of Queries for Each Interface

interface	map	slider, checkbox	button	text box
percent	62.8%	29.9%	3.6%	3.6%

Query Interface: We report the percentage of queries issued using each interface widget in Table 9. It shows that the user prefers issuing queries with map as opposed to other interfaces. These percentages can be used for informing weights in prefetching strategies, e.g., more map tiles should be prefetched as opposed to the next range for slider in cases where both are available.

Zooming: Next, we analyzed the different zoom levels used by users' in their exploration, to find the most common levels for prefetching. Figure 18 shows the change of zoom levels over time, where each user is represented by a color. It shows that except for one, users' zoom navigate at most, three level from their starting point, so depths greater than three can be ignored when prefetching. The majority of users explore between levels 11 and 14. Most pre-computation efforts should thus concentrate on these levels.

Dragging: We further analyzed map dragging behavior for a particular zoom level, focusing on levels 11–14, since these were the most common. We facet over different zoom levels since, at different levels, the same viewpoint size represents a different area size. Figure 19 shows the longitude and latitude change of the bound center (i.e., average of the four bound corners) over the time for different users. Table 10 summarizes this in the form of ranges of latitude and longitude change. As we can see, at deeper levels, users move smaller distances.

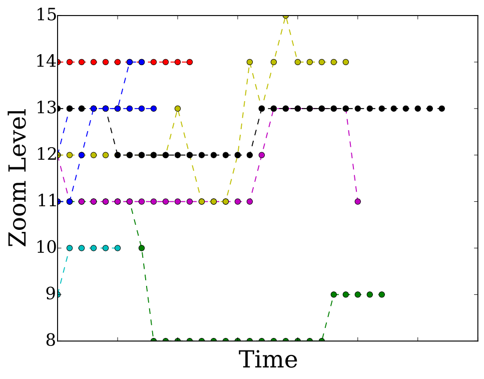


Fig. 18: Change of Zoom Levels over Time for each user, represented by a color. Zoom levels concentrate between 11 and 14.

These statistics can be used to determine number of adjacent tiles to prefetch as well as determine the appropriate tile size.

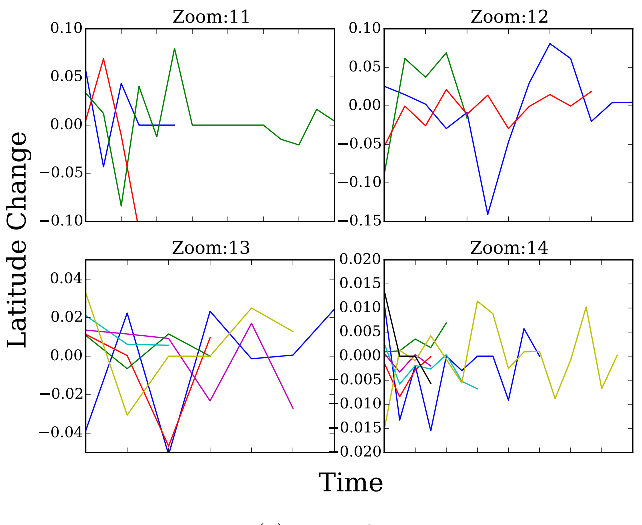
Table 10: Ranges for Center of Bounds

zoom	latitude	longitude
11	[-0.10, 0.07]	[-0.2, 0.2]
12	[-0.15, 0.07]	[-0.2, 0.2]
13	[-0.05, 0.03]	[-0.08, 0.05]
14	[-0.015, 0.013]	[-0.02, 0.02]

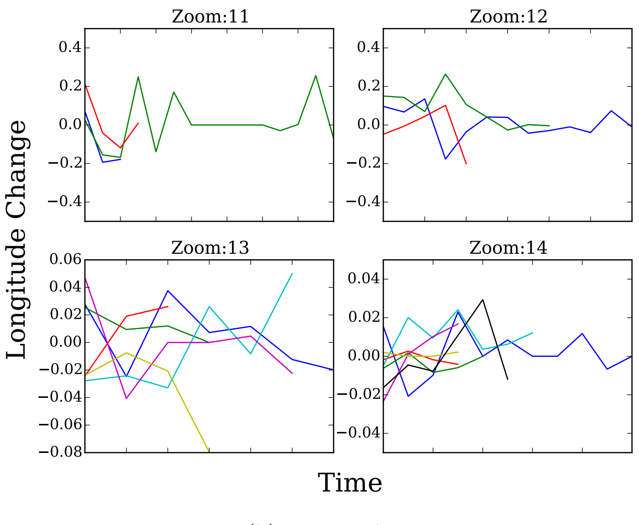
Filtering Behavior: Figure 20 shows the cumulative distribution for number of filter conditions, i.e., frequency of queries with 2 filters, 4 filters and so on. It shows that 70% of queries had four filters or less. Thus, it makes more sense to cache results with upto 4 filter predicates, which can be refined as more filters are used.

8.2 Evaluation

Figure 21 shows the cumulative distributions for the request and exploration times over all users. It shows that 80% of exploration time are greater than 1 second and 80% queries are completed in less than 1 second, which indicates that at least one query can be fetched in most cases. On average, the exploration time is 18.3 seconds and the fetching time is about 1.1 seconds, which indicates that about 18 adjacent queries can be fetched.



(a) Latitude



(b) Longitude

Fig. 19: Change of Latitude / Longitude of the Bound Center over Time. Each color represents a different user.

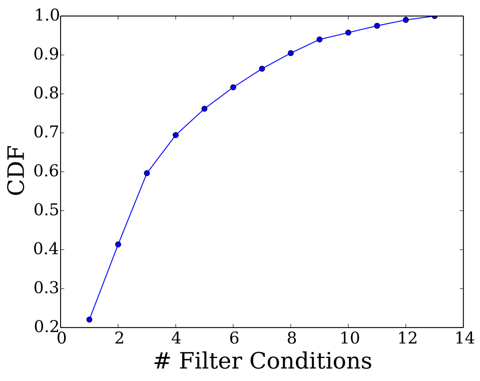


Fig. 20: Cumulative Distribution Function (CDF) of Number of Filter Conditions used.

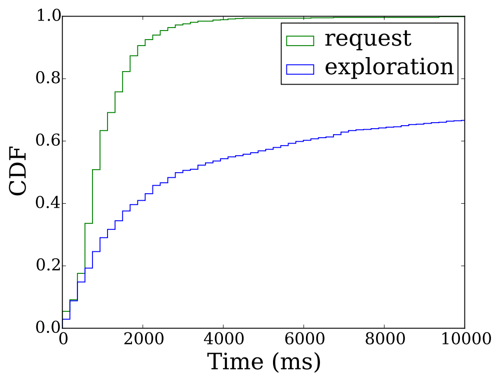


Fig. 21: Cumulative Distribution Functions (CDFs) for Request and Exploration Time: The request time is much lower than the exploration time indicating that multiple speculative queries can be prefetched.

8.3 Takeaways

We demonstrate that the user prefers map and slider widget over the others. When users manipulate the map, they focus on zoom levels between 11 and 14. These observations can be used to provide guidelines for behavior-driven techniques like prefetching. Our analysis of the request and exploration time show that the system can prefetch 18 adjacent queries on average and verify requirements to speed up query specification. This case study provides an example of the kinds of analysis required for composite interfaces.

9 Related Work

In this section we describe research that is related to our case studies.

Data Interaction with New Computing Devices: With the rise in popularity of multi-touch and gesture devices, there have been several works on interactive data exploration with multi-touch and gesture devices [86, 95, 122, 137, 152, 195]. Compared to mouse and keyboard, new computing devices provide better direct manipulation experience [157] and multi-finger interaction (e.g., zoom in / out), etc. These systems include the following:

– GestureDB [137] and DBTouch [122] are multi-touch query interfaces which translate gestures into database queries.
– Data3 [86] allows the user to manipulate the 3D cube with the Kinect.

- PanoramicData [195] is a multi-touch interface focusing on the machine learning and data analysis algorithms.
- Kinetica [152] is a multi-touch interface allowing the user to explore the visualization with physical affordances.
- SnaptoQuery [95] allows the user to issue range queries with gesture devices and shows that gesture devices are comparable to the mouse and touch devices.
- Tableau Vizable [17] allows interactive data visualization.

However, most of these works focus on usability. There are no existing works for capturing the characteristics of new query workloads, investigating possible behavior-driven optimizations with new computing devices, and evaluating the performance of databases with those interactive workloads.

Interactive Performance: Liu and Heer [126] show that an additional delay of 500ms can affect users' exploration behavior, i.e., it is necessary to keep an interactive performance. Many methods have been proposed to achieve this. Some systems fetch all data at once and build their own cache or data management systems. imMens [127] uses bin or grid aggregation and sustains 50 frames-per-second brushing & linking over billions of records. Nanocubes [123] is proposed to construct a data cube that is able to fit in a modern laptop's main memory. In SnapToQuery, a clustering-based method [95] is proposed to approximate the query result.

Most of theses optimizations are specific to the system and do not measure database performance, which our case studies explore. General optimization techniques have also been proposed, such as sampling-based [24, 98, 159], prefetching [114, 120, 164, 188], adaptive indexing [80, 89, 90], etc. However, these optimization techniques do not take the users' behavior into account.

Traditional Benchmarking & Optimizations: Benchmarks are used to measure the performance of systems. [77] surveys popular benchmarks and elaborates the need of domain-specific benchmarks. Traditional benchmarks focus on the transaction processing performance, such as TPC [19] and Wisconsin Benchmark [58]. They typically measure a throughput metric (work/second) and a price metric which is a five-year cost-of-ownership metric. Recently, with the increasing popularity of cloud serving systems (e.g., Hadoop, Spark, etc.), benchmarks have been proposed to evaluate their performance. Current cloud benchmarks usually focus on one specific application domain. For example, YCSB [55] focuses on the online service. Others [2, 53, 54] focus on the real-time analysis. BigDataBench [171] summarizes current status of big data benchmarks and proposes a comprehensive benchmark spans offline analytics, online service, and real-time analytics. Graph and RDF benchmarks have also been proposed, e.g., LinkBench [28], a benchmark based on the Facebook social graph. The Linked Data Benchmark Council (LDBC) [8] was founded a couple of years ago to establish benchmarks for evaluating graph data management systems.

Closer to our work, in their vision paper, Eichmann et al. [66] elaborate on use cases for interactive data exploration. They follow up on this in [65], which provides a benchmark for interactive systems. However, these benchmarks and optimizations do not account for the salient features of interactive query interfaces on non-keyboard devices, and further, they do not consider user-driven optimizations.

Also, as visualizations become a popular channel to communicate results, there are some efforts in evaluating the overlap between visualization and data systems [35]. Battle et al. [35] propose a standard model for collecting traces. Finally, with the popularity of mobile devices, benchmarks have been proposed to evaluate data management in mobile systems [108, 112].

10 Conclusion and Future Work

Interactive workloads present characteristics that are different from traditional workloads, especially for new devices and interfaces. In this paper, we talk about both existent and emergent characteristics of interactive workloads: difference between devices and interfaces, continuous actions, ambiguity in query intent, and session behavior for adjacent queries. We catalog different metrics used in the literature, both from a systems perspective and a HCI perspective. We further discuss various pitfalls for user studies such as cognitive biases and threats to validity. Based on these, we provide guidelines for *behavior-driven* optimizations and evaluation of interactive systems and demonstrate these through three case studies which span different interaction devices (mouse, touch, Leap Motion), query interfaces (e.g., map, slider), interaction techinques (e.g., linking & brushing), and queries (e.g., `select`, `aggregation`). Our case studies show that behavior analyses can help system designers to build more responsive UIs and highly performant backend systems.

There are several directions for future work. We introduced a new metric, latency constraint violation, to measure if a user perceives a delay from the system, and demonstrated it in two of our case studies. However, adapting this metric for other interactive systems requires understanding of the internals of each, and hence a general purpose suite of this metric remains an open problem.

The metrics discussed in this work concern performance of the interface and backend. The other aspect of evaluating the system involves its information presentation and cognitive effect on the user. For example, a system that provides *too much* information to the user may be considered performant from the system's standpoint, but could be considered suboptimal from a user's standpoint. In our case studies, this applies to the skip and the KL optimizations in crossfiltering. We need to conduct user studies to see if users loose information due to queries being skipped. This is especially true in the KL divergence case because if the user is looking for anomalies, the slight difference in result set might be important. We have to work on methods for measuring loss of information. Orthogonally, if an interface is not designed well, it can lead to the user being disoriented and cognitively overloaded as reported in [153]. While we briefly discussed cognitive effort with respect to construct validity (Section 4.2.3), measuring it in interactive systems remains an open research area.

References

1. Airbnb: Vacation Rentals, Homes, Experiences and Places. `https://www.airbnb.com/`.
2. AMP Benchmarks. `https://amplab.cs.berkeley.edu/benchmark/`.
3. Crossfilter Library. `http://square.github.io/crossfilter/`.
4. Delta. `https://developer.mozilla.org/en-US/docs/Web/Events/mousewheel`.
5. F 015 Luxury in Motion Concept Car: Interaction with the vehicle through gestures, eye tracking and high-res touch-screens. `https://www.mbusa.com/mercedes/future/model/model-All_New_F015_Luxury`.
6. GlobalInterpreterLock. `https://wiki.python.org/moin/GlobalInterpreterLock`.
7. IMDb. `http://www.imdb.com/`.
8. LDBC: The graph and RDF benchmark reference. `http://ldbcouncil.org/benchmarks`.
9. MemSQL: The Fastest In-Memory Database. `http://www.MemSQL.com/`.
10. Mutation Events. `https://developer.mozilla.org/en-US/docs/Web/Guide/Events/Mutation_events`.
11. Number of smartphones sold to end users worldwide from 2007 to 2020 (in million units). `https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/`.
12. OMDb API - The Open Movie Database. `http://www.omdbapi.com/`.
13. PostgreSQL: The world's most advanced open source database. `https://www.postgresql.org/`.
14. Power BI. `https://powerbi.microsoft.com`.
15. ScrollTop. `https://developer.mozilla.org/en-US/docs/Web/API/Element/scrollTop`.
16. Tableau. `http://www.tableau.com/`.
17. Tableau Vizable. `https://vizable.tableau.com/`.
18. TEDCAS uses the Myo armband to give surgeons gesture control. `http://blog.thalmic.com/myo-armband-surgery/`.
19. TPC Benchmark. `http://www.tpc.org/`.
20. UCI Repository of Machine Learning Databases. `https://archive.ics.uci.edu/ml/datasets.html`.
21. Use Multi-Touch gestures on your Mac. `https://support.apple.com/en-us/HT204895`.
22. Worldwide tablet shipments from 2nd quarter 2010 to 3rd quarter 2018 (in million units). `https://www.statista.com/statistics/272070/global-tablet-shipments-by-quarter/`.
23. A. Abouzied, J. Hellerstein, and A. Silberschatz. Dataplay: Interactive tweaking and example-driven correction of graphical database queries. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 207–218, New York, NY, USA, 2012. ACM.
24. 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*, EuroSys '13, pages 29–42, New York, NY, USA, 2013. ACM.
25. S. Al-Megren. A predictive fingerstroke-level model for smartwatch interaction. *Multimodal Technologies and Interaction*, 2(3):38, 2018.
26. S. Al-Megren, W. Altamimi, and H. S. Al-Khalifa. Blind flm: An enhanced keystroke-level model for visually impaired smartphone interaction. In *IFIP Conference on Human-Computer Interaction*, pages 155–172. Springer, 2017.
27. W. Albert and T. Tullis. *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.
28. T. G. Armstrong, V. Ponnekanti, D. Borthakur, and M. Callaghan. Linkbench: A database benchmark based on the facebook social graph. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 1185–1196, New York, NY, USA, 2013. ACM.
29. E. Bakke, D. Karger, and R. Miller. A spreadsheet-based user interface for managing plural relationships in structured data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2541–2550, New York, NY, USA, 2011. ACM.
30. E. Bakke and D. R. Karger. Expressive query construction through direct manipulation of nested relational results. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1377–1392. ACM, 2016.
31. A. Bangor, P. T. Kortum, and J. T. Miller. An Empirical Evaluation of the System Usability Scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.
32. C. M. Barnum. *Usability testing essentials: ready, set... test!* Elsevier, 2010.
33. R. C. Basole, T. Clear, M. Hu, H. Mehrotra, and J. Stasko. Understanding interfirm relationships in business ecosystems with interactive visualization. *IEEE Transactions on visualization and computer graphics*, 19(12):2526–2535, 2013.
34. S. Basu Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 13–22, New York, NY, USA, 2008. ACM.

35. L. Battle, R. Chang, J. Heer, and M. Stonebraker. Position statement: The case for a visualization performance benchmark. In *2017 IEEE Workshop on Data Systems for Interactive Analysis (DSIA)*, pages 1–5, Oct 2017.
36. L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. pages 1363–1375, 2016.
37. D. A. Bell, L. S. Deluca, D. J. Levinson, and R. Salem. Browser interaction for lazy loading operations, Dec. 15 2014. US Patent App. 14/570,430.
38. J. F. Beltran, Z. Huang, A. Abouzied, and A. Nandi. Don't just swipe left, tell me why: Enhancing gesture-based feedback with reason bins. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 469–480. ACM, 2017.
39. M. Bendre, V. Venkataraman, X. Zhou, K. Chang, and A. Parameswaran. Towards a holistic integration of spreadsheets with databases: A scalable storage engine for presentational data management. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 113–124. IEEE, 2018.
40. B. Benson. Cognitive bias cheat sheet. *Better Humans*, 2016.
41. J. Bernard, N. Wilhelm, B. Krüger, T. May, T. Schreck, and J. Kohlhammer. Motionexplorer: Exploratory search in human motion capture data based on hierarchical aggregation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2257–2266, Dec. 2013.
42. X. Bi, Y. Li, and S. Zhai. Ffitts law: modeling finger touch with fitts' law. In *SIGCHI*, 2013.
43. C. Binnig, A. Fekete, and A. Nandi. Hilda '16: Proceedings of the workshop on human-in-the-loop data analytics. New York, NY, USA, 2016. ACM.
44. A. Biswas, S. Dutta, H.-W. Shen, and J. Woodring. An information-aware framework for exploring multivariate data sets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2683–2692, Dec. 2013.
45. J. Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.
46. C. Burley and A. Nandi. Arquery: Hallucinating analytics over real-world data using augmented reality. In *CIDR*, 2019.
47. N. Cao, D. Gotz, J. Sun, and H. Qu. Dicon: Interactive visual analysis of multidimensional clusters. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2581–2590, Dec 2011.
48. S. K. Card. *The psychology of human-computer interaction.* CRC Press, 2017.
49. S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining interactivity while exploring massive time series. In *2008 IEEE Symposium on Visual Analytics Science and Technology*, pages 59–66, Oct 2008.
50. P. Chau, J. Vreeken, M. van Leeuwen, D. Shahaf, and C. Faloutsos. Proceedings of the acm sigkdd workshop on interactive data exploration and analytics. In *ACM SIGKDD 2016 Full-day Workshop on Interactive Data Exploration and Analytics.* IDEA'16, 2016.
51. S. Chaudhuri, R. Motwani, and V. Narasayya. Random sampling for histogram construction: How much is enough? In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 436–447, New York, NY, USA, 1998. ACM.
52. K. Chen. *Data-driven techniques for improving data collection in low-resource environments.* PhD thesis, UC Berkeley, 2011.
53. Y. Chen, S. Alspaugh, D. Borthakur, and R. Katz. Energy efficiency for large-scale mapreduce workloads with significant interactive analysis. In *Proceedings of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pages 43–56, New York, NY, USA, 2012. ACM.
54. Y. Chen, S. Alspaugh, and R. Katz. Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *Proc. VLDB Endow.*, 5(12):1802–1813, Aug. 2012.
55. B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 143–154, New York, NY, USA, 2010. ACM.
56. M. Correll, M. Li, G. Kindlmann, and C. Scheidegger. Looks good to me: Visualizations as sanity checks. *IEEE transactions on visualization and computer graphics*, 2018.
57. Ç. Demiralp, P. J. Haas, S. Parthasarathy, and T. Pedapati. Foresight: Rapid Data Exploration Through Guideposts. *arXiv preprint arXiv:1709.10513*, 2017.
58. D. J. DeWitt. The wisconsin benchmark: Past, present, and future, 1993.
59. E. Dimara, G. Bailly, A. Bezerianos, and S. Franconeri. Mitigating the attraction effect with visualizations. *IEEE transactions on visualization and computer graphics*, 2018.
60. E. Dimara, S. Franconeri, C. Plaisant, A. Bezerianos, and P. Dragicevic. A task-based taxonomy of cognitive biases for information visualization. *IEEE transactions on visualization and computer graphics*, 2018.
61. K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 517–528, New York, NY, USA, 2014. ACM.
62. P. R. Doshi, E. A. Rundensteiner, and M. O. Ward. Prefetching for visual data exploration. In *Eighth International Conference on Database Systems for Advanced Applications, 2003*, pages 195–202, March 2003.
63. S. M. Drucker, D. Fisher, R. Sadana, J. Herron, et al. Touchviz: a case study comparing two interfaces for data analytics on tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2301–2310. ACM, 2013.
64. R. Ebenstein, N. Kamat, and A. Nandi. Fluxquery: An execution framework for highly interactive query workloads. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1333–1345. ACM, 2016.
65. P. Eichmann, C. Binnig, T. Kraska, and E. Zgraggen. Idebench: A benchmark for interactive data exploration. *CoRR*, abs/1804.02593, 2018.
66. P. Eichmann, E. Zgraggen, Z. Zhao, C. Binnig, and T. Kraska. Towards a benchmark for interactive data exploration. *IEEE Data Eng. Bull.*, 39:50–61, 2016.
67. K. El Batran and M. D. Dunlop. Enhancing klm (keystroke-level model) to fit touch screen mobile devices. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, pages 283–286. ACM, 2014.
68. J. Faith. Targeted projection pursuit for interactive exploration of high- dimensional data sets. In *Information Visualization, 2007. IV '07. 11th International Conference*, pages 286–292, July 2007.

69. L. Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3):379–383, Aug 2003.
70. J.-D. Fekete. Progressivis: A toolkit for steerable progressive analytics and visualization. In *1st Workshop on Data Systems for Interactive Analysis*, page 5, 2015.
71. J.-D. Fekete and C. Plaisant. Interactive information visualization of a million items. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, INFOVIS '02, pages 117–, Washington, DC, USA, 2002. IEEE Computer Society.
72. S. Feng, A. Huber, B. Glavic, and O. Kennedy. Uncertainty annotated databases-a lightweight approach for approximating certain answers. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1313–1330. ACM, 2019.
73. N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva. Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2149–2158, Dec. 2013.
74. D. Fisher, I. Popov, S. Drucker, and m. schraefel. Trust me, i'm partially right: Incremental visualization lets analysts explore large datasets faster. ACM Conference on Human Factors in Computing Systems, May 2012.
75. P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381, 1954.
76. P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. volume 27, pages 261–298, New York, NY, USA, Sept. 2002. ACM.
77. J. Gray. *Benchmark Handbook: For Database and Transaction Processing Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.
78. N. R. Gujarathi and A. A. Shah. Parameterized computed scrolling for navigation of structured data, 2015.
79. D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Approximating multi-dimensional aggregate range queries over real attributes. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 463–474, New York, NY, USA, 2000. ACM.
80. F. Halim, S. Idreos, P. Karras, and R. H. C. Yap. Stochastic database cracking: Towards robust adaptive indexing in main-memory column-stores. *Proc. VLDB Endow.*, 5(6):502–513, Feb. 2012.
81. M. G. Haselton, D. Nettle, and D. R. Murray. The evolution of cognitive bias. *The handbook of evolutionary psychology*, pages 1–20, 2015.
82. C. Hebert, J. Ridgway, B. Vekhter, E. C. Brown, S. G. Weber, and A. Robicsek. Demonstration of the weighted-incidence syndromic combination antibiogram: an empiric prescribing decision aid. *Infection Control & Hospital Epidemiology*, 33(4):381–388, 2012.
83. J. Heer, M. Agrawala, and W. Willett. Generalized selection via interactive query relaxation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 959–968, New York, NY, USA, 2008. ACM.
84. J. Heer and M. Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 203–212. ACM, 2010.
85. J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Acm Sigmod Record*, volume 26, pages 171–182. ACM, 1997.
86. S. Hirte, A. Seifert, S. Baumann, D. Klan, and K.-U. Sattler. Data3 – a kinect interface for olap using complex event processing. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 1297–1300, Washington, DC, USA, 2012. IEEE Computer Society.
87. P. Holleis, F. Otto, H. Hussmann, and A. Schmidt. Keystroke-level model for advanced mobile phone interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1505–1514. ACM, 2007.
88. K. Hu, D. Orghian, and C. Hidalgo. DIVE: A Mixed-Initiative System Supporting Integrated Data Exploration Workflows. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, page 5. ACM, 2018.
89. S. Idreos. Database cracking: Towards auto-tuning database kernels. *CWI and University of Amsterdam*, 2010.
90. S. Idreos, S. Manegold, H. Kuno, and G. Graefe. Merging what's cracked, cracking what's merged: Adaptive indexing in main-memory column-stores. *Proc. VLDB Endow.*, 4(9):586–597, June 2011.
91. S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 277–281, New York, NY, USA, 2015. ACM.
92. T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, pages 139–148, New York, NY, USA, 2000. ACM.
93. H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 13–24, New York, NY, USA, 2007. ACM.
94. W. Javed, S. Ghani, and N. Elmqvist. Gravnav: Using a gravity model for multi-scale navigation. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 217–224, New York, NY, USA, 2012. ACM.
95. L. Jiang and A. Nandi. Snaptoquery: Providing interactive feedback during exploratory query specification. *Proc. VLDB Endow.*, 8(11):1250–1261, July 2015.
96. B. E. John and D. E. Kieras. Using goms for user interface design and evaluation: Which technique? *TOCHI*, 1996.
97. R. Jota, A. Ng, P. Dietz, and D. Wigdor. How fast is fast enough?: a study of the effects of latency in direct-touch pointing tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2291–2300. ACM, 2013.
98. N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *2014 IEEE 30th International Conference on Data Engineering*, pages 472–483, March 2014.
99. N. Kamat and A. Nandi. Infiniviz: Interactive visual exploration using progressive bin refinement. *arXiv preprint arXiv:1710.01854*, 2017.
100. N. Kamat and A. Nandi. A session-based approach to fast-but-approximate interactive data cube exploration. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(1):9, 2018.

101. S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372. ACM, 2011.
102. S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3363–3372, New York, NY, USA, 2011. ACM.
103. J. Kang, J. F. Naughton, and S. D. Viglas. Evaluating window joins over unbounded streams. In *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, pages 341–352, March 2003.
104. A. Kashyap, V. Hristidis, and M. Petropoulos. Facetor: Cost-driven exploration of faceted query results. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 719–728, New York, NY, USA, 2010. ACM.
105. A. Kashyap, V. Hristidis, M. Petropoulos, and S. Tavoulari. Effective navigation of query results based on concept hierarchies. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):540–553, April 2011.
106. M. Kaul, B. Yang, and C. S. Jensen. Building accurate 3d spatial networks to enable next generation intelligent transportation systems. In *2013 IEEE 14th International Conference on Mobile Data Management*, volume 1, pages 137–146, June 2013.
107. D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, Jan. 2002.
108. O. Kennedy, J. Ajay, G. Challen, and L. Ziarek. Pocket data: The need for tpc-mobile. In R. Nambiar and M. Poess, editors, *Performance Evaluation and Benchmarking: Traditional to Big Data to Internet of Things*, pages 8–25, Cham, 2016. Springer International Publishing.
109. A. Key, B. Howe, D. Perry, and C. Aragon. Vizdeck: Self-organizing dashboards for visual analytics. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 681–684, New York, NY, USA, 2012. ACM.
110. M. Khan, L. Xu, A. Nandi, and J. M. Hellerstein. Data tweening: Incremental visualization of data transforms. *Proc. VLDB Endow.*, 10(6):661–672, Feb. 2017.
111. A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid sampling for visualizations with ordering guarantees. *Proceedings of the VLDB Endowment*, 8(5):521–532, 2015.
112. J.-M. Kim and J.-S. Kim. Androbench: Benchmarking the storage performance of android-based mobile devices. In S. Sambath and E. Zhu, editors, *Frontiers in Computer Education*, pages 667–674. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
113. R. Kosara, F. Bendix, and H. Hauser. Parallel sets: Interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, July 2006.
114. A. Kraiss and G. Weikum. Integrated document caching and prefetching in storage hierarchies based on markov-chain predictions. *The VLDB Journal*, 7(3):141–162, Aug 1998.
115. T. Kraska. Northstar: An Interactive Data Science System. *Proceedings of the VLDB Endowment*, 11(12), 2018.
116. S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.
117. H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. Empirical studies in information visualization: Seven scenarios. *IEEE transactions on visualization and computer graphics*, 18(9):1520–1536, 2012.
118. J. Lazar, J. H. Feng, and H. Hochheiser. *Research Methods in Human-Computer Interaction*. Wiley Publishing, 2010.
119. A. Lee, K. Song, H. B. Ryu, J. Kim, and G. Kwon. Fingerstroke time estimates for touchscreen-based mobile gaming interaction. *Human movement science*, 44:211–224, 2015.
120. D. H. Lee, J. S. Kim, S. D. Kim, K. C. Kim, K. Yoo-Sung, and J. Park. Adaptation of a neighbor selection markov chain for prefetching tiled web gis data. In T. Yakhno, editor, *Advances in Information Systems*, pages 213–222, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
121. Y. Li, H. Yang, and H. V. Jagadish. Nalix: A generic natural language search environment for xml data. *ACM Trans. Database Syst.*, 32(4), Nov. 2007.
122. E. Liarou and S. Idreos. dbtouch in action database kernels for touch-based data exploration. In *2014 IEEE 30th International Conference on Data Engineering*, pages 1262–1265, March 2014.
123. L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, Dec. 2013.
124. B. Liu and H. Jagadish. A spreadsheet algebra for a direct data manipulation query interface. In *2009 IEEE 25th International Conference on Data Engineering*, pages 417–428, March 2009.
125. F. Liu, N. Kamat, S. Blanas, and A. Nandi. To ship or not to (function) ship (extended version). In *2018 IEEE High Performance extreme Computing Conference (HPEC)*. IEEE, 2018.
126. Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2122–2131, Dec 2014.
127. Z. Liu, B. Jiang, and J. Heer. immens: Real-time visual querying of big data. 32(3pt4):421–430, 2013.
128. J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144, Nov. 2007.
129. A. R. Martin and M. O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Proceedings of the 6th Conference on Visualization '95*, VIS '95, pages 271–, Washington, DC, USA, 1995. IEEE Computer Society.
130. P. McLachlan, T. Munzner, E. Koutsofios, and S. North. Liverac: Interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1483–1492, New York, NY, USA, 2008. ACM.
131. C. Mohan. Caching technologies for web applications. In *VLDB*, volume 1, page 726, 2001.
132. D. Moritz, D. Fisher, B. Ding, and C. Wang. Trust, but verify: Optimistic visualizations of approximate queries for exploring big data. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2904–2915. ACM, 2017.
133. D. Moritz, B. Howe, and J. Heer. Falcon: Balancing interactive latency and resolution sensitivity for scalable linked visualizations. 2019.

134. T. Mostak. An overview of mapd (massively parallel database). *White paper. Massachusetts Institute of Technology*, 2013.
135. F. Moussavi. Hybrid inertial and touch sensing input device, Feb. 18 2010. US Patent App. 12/192,889.
136. T. Munzner. A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, 15(6), 2009.
137. A. Nandi, L. Jiang, and M. Mandel. Gestural query specification. *Proc. VLDB Endow.*, 7(4):289–300, Dec. 2013.
138. W. T. Nelson, M. M. Roe, R. S. Bolia, and R. M. Morley. Assessing simulator sickness in a see-through hmd: Effects of time delay, time on task, and task complexity. Technical report, AIR FORCE RESEARCH LAB WRIGHT-PATTERSON AFB OH, 2000.
139. N. Nourbakhsh, Y. Wang, F. Chen, and R. A. Calvo. Using galvanic skin response for cognitive load measurement in arithmetic and reading tasks. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, OzCHI '12, pages 420–423, New York, NY, USA, 2012. ACM.
140. B. Omidvar-Tehrani, A. Nandi, N. Meyer, D. Flanagan, and S. Young. Dv8: Interactive analysis of aviation data. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1411–1412. IEEE, 2017.
141. L. Padilla. A case for cognitive models in visualization research. *Proceedings of the Seventh Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, pages 143–151, 2018.
142. E. S. Patterson, C. M. Dewart, K. Stevenson, A. Mbodj, M. Lustberg, E. M. Hade, and C. Hebert. A mixed methods approach to tailoring evidence-based guidance for antibiotic stewardship to one medical system. In *Proceedings of the International Symposium on Human Factors and Ergonomics in Health Care*, volume 7, pages 224–231. SAGE Publications Sage India: New Delhi, India, 2018.
143. A. Pavlovych and C. Gutwin. Assessing target acquisition and tracking performance for complex moving targets in the presence of latency and jitter. In *Proceedings of Graphics Interface 2012*, pages 109–116. Canadian Information Processing Society, 2012.
144. C. Plaisant. The challenge of information visualization evaluation. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 109–116, New York, NY, USA, 2004. ACM.
145. V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 294–305, New York, NY, USA, 1996. ACM.
146. F. Psallidas and E. Wu. Smoke: Fine-grained lineage at interactive speed. *Proceedings of the VLDB Endowment*, 11(6):719–732, 2018.
147. A. Quezada, R. Juárez-Ramírez, S. Jiménez, A. Ramírez-Noriega, S. Inzunza, and R. Munoz. Assessing the target?size and drag distance in mobile applications for users with autism. In *World Conference on Information Systems and Technologies*, pages 1219–1228. Springer, 2018.
148. P. Rahman, C. Hebert, and A. Nandi. ICARUS: Minimizing Human Effort in Iterative Data Completion. *Proceedings of the VLDB Endowment*, 11(13), 2018.
149. P. Rahman and A. Nandi. Transformer: A Database-Driven Approach to Generating Forms for Constrained Interaction. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. ACM, 2019.
150. S. Rahman, M. Aliakbarpour, H. K. Kong, E. Blais, K. Karahalios, A. Parameswaran, and R. Rubinfield. I've seen enough: incrementally improving visualizations to support rapid decision making. *Proceedings of the VLDB Endowment*, 10(11):1262–1273, 2017.
151. G. M. Rosa and M. L. Elizondo. Use of a gesture user interface as a touchless image navigation system in dental surgery: Case series report. *Imaging science in dentistry*, 44(2):155–160, 2014.
152. J. M. Rzeszotarski and A. Kittur. Kinetica: Naturalistic multi-touch data visualization. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 897–906, New York, NY, USA, 2014. ACM.
153. R. G. Saadé and C. A. Otrakji. First impressions last a lifetime: effect of interface type on disorientation and cognitive load. *Computers in human behavior*, 23(1):525–535, 2007.
154. A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. In *Proceedings of the 16th Eurographics Conference on Visualization*, EuroVis '14, pages 351–360, Aire-la-Ville, Switzerland, Switzerland, 2014. Eurographics Association.
155. J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. *Information Visualization*, 4(2):96–113, July 2005.
156. B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–, Washington, DC, USA, 1996. IEEE Computer Society.
157. B. Shneiderman, C. Williamson, and C. Ahlberg. Dynamic queries: Database searching by direct manipulation. pages 669–670, 1992.
158. T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. Parameswaran. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *Proceedings of the VLDB Endowment*, 10(4):457–468, 2016.
159. L. Sidirourgos, M. L. Kersten, P. A. Boncz, et al. Sciborq: Scientific data management with bounds on runtime and quality. In *CIDR*, 2011.
160. M. Singh, A. Nandi, and H. V. Jagadish. Skimmer: Rapid scrolling of relational query results. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 181–192, New York, NY, USA, 2012. ACM.
161. R. St Amant, T. E. Horton, and F. E. Ritter. Model-based evaluation of cell phone menu interaction. In *CHI*, 2004.
162. J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285, 1988.
163. P. Tan. BMW Demonstrates Future iDrive with Touchscreen, Gesture and Tablet Control. *CES 2015*, 2015.
164. F. Tauheed, T. Heinis, F. Schürmann, H. Markram, and A. Ailamaki. Scout: Prefetching for latent structure following queries. *Proc. VLDB Endow.*, 5(11):1531–1542, July 2012.
165. A. C. Valdez, M. Ziefle, and M. Sedlmair. A framework for studying biases in visualization research. 2017.
166. L. F. Van Dillen, D. J. Heslenfeld, and S. L. Koole. Tuning down the emotional brain: an fmri study of the effects of cognitive load on the processing of affective images. *Neuroimage*, 45(4):1212–1219, 2009.

167. M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: Automatically generating query visualizations. *Proc. VLDB Endow.*, 7(13):1581–1584, Aug. 2014.
168. S. D. Viglas, J. F. Naughton, and J. Burger. Maximizing the output rate of multi-way join queries over streaming information sources. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 285–296. VLDB Endowment, 2003.
169. J. S. Vitter, M. Wang, and B. Iyer. Data cube approximation and histograms via wavelets. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, pages 96–104, New York, NY, USA, 1998. ACM.
170. E. Wall, M. Agnihotri, L. Matzen, K. Divis, M. Haass, A. Endert, and J. Stasko. A heuristic approach to value-driven evaluation of visualizations. *IEEE transactions on visualization and computer graphics*, 2018.
171. L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zheng, G. Lu, K. Zhan, X. Li, and B. Qiu. Bigdatabench: A big data benchmark suite from internet services. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 488–499, Feb 2014.
172. C. Weaver. Multidimensional visual analysis using cross-filtered views. In *2008 IEEE Symposium on Visual Analytics Science and Technology*, pages 163–170, Oct 2008.
173. J. Wei, Z. Shen, N. Sundaresan, and K.-L. Ma. Visual cluster exploration of web clickstream data. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 3–12, Oct 2012.
174. L. Wilkinson. *The grammar of graphics*. Springer Science & Business Media, 2006.
175. W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136, 2007.
176. K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics*, 22(1):649–658, 2016.
177. K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2648–2659. ACM, 2017.
178. K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mané, D. Fritz, D. Krishnan, F. B. Viégas, and M. Wattenberg. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE transactions on visualization and computer graphics*, 24(1):1–12, 2018.
179. J. Woodring and H.-W. Shen. Multiscale time activity data exploration via temporal clustering visualization spreadsheet. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):123–137, Jan 2009.
180. E. Wu, L. Jiang, L. Xu, and A. Nandi. Graphical perception in animated bar charts. *arXiv preprint arXiv:1604.00080*, 2016.
181. E. Wu, L. Jiang, L. Xu, and A. Nandi. Graphical perception in animated bar charts. volume abs/1604.00080, 2016.
182. Y. Wu, R. Chang, E. Wu, and J. Hellerstein. Programming with timespans in interactive visualizations. *arXiv preprint arXiv:1907.00075*, 2019.
183. M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. *Proceedings of the VLDB Endowment*, 4(5):279–289, 2011.
184. D. Yang, Z. Guo, E. A. Rundensteiner, and M. O. Ward. Clues: A unified framework supporting interactive exploration of density-based clusters in streams. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 815–824, New York, NY, USA, 2011. ACM.
185. D. Yang, E. A. Rundensteiner, and M. O. Ward. Analysis guided visual exploration of multivariate data. In *2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 83–90, Oct 2007.
186. J. Yang, A. Patro, S. Huang, N. Mehta, M. O. Ward, and E. A. Rundensteiner. Value and relation display for interactive exploration of high dimensional datasets. In *IEEE Symposium on Information Visualization*, pages 73–80, 2004.
187. J. Yang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In *Proceedings of the Ninth Annual IEEE Conference on Information Visualization*, INFOVIS'03, pages 105–112, Washington, DC, USA, 2003. IEEE Computer Society.
188. S. Yeşilmurat and V. İşler. Retrospective adaptive prefetching for interactive web gis applications. *GeoInformatica*, 16(3):435–466, Jul 2012.
189. J. S. Yi, Y. ah Kang, and J. Stasko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics*, 13(6):1224–1231, 2007.
190. X. Yuan, D. Ren, Z. Wang, and C. Guo. Dimension projection matrix/tree: Interactive subspace visual exploration and analysis of high dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2625–2633, Dec. 2013.
191. Y. Yuan, G. Wang, L. Chen, and H. Wang. Graph similarity search on large uncertain graph databases. *The VLDB JournalThe International Journal on Very Large Data Bases*, 24(2):271–296, 2015.
192. Y. Yuan, G. Wang, H. Wang, and L. Chen. Efficient subgraph search over large uncertain graphs. *Proc. VLDB Endowment*, 4(11):876–886, 2011.
193. Y. Yuan, G. Wang, J. Y. Xu, and L. Chen. Efficient distributed subgraph similarity matching. *The VLDB JournalThe International Journal on Very Large Data Bases*, 24(3):369–394, 2015.
194. K. Zarifis and Y. Papakonstantinou. ViDeTTe Interactive Notebooks. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, page 2. ACM, 2018.
195. E. Zgraggen, R. Zeleznik, and S. M. Drucker. Panoramicdata: Data analysis through pen & touch. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2112–2121, Dec 2014.
196. Z. Zhang, K. T. McDonnell, and K. Mueller. A network-based interface for the exploration of high-dimensional data spaces. In *Proceedings of the 2012 IEEE Pacific Visualization Symposium*, PACIFICVIS '12, pages 17–24, Washington, DC, USA, 2012. IEEE Computer Society.