Towards a Benchmark for Interactive Data Exploration

Philipp Eichmann, Emanuel Zgraggen, Zheguang Zhao, Carsten Binnig, Tim Kraska
Brown University, Providence, RI, USA

Abstract

Existing benchmarks for analytical database systems such as TPC-DS and TPC-H are designed for static reporting scenarios. The main metric of these benchmarks is the performance of running different SQL queries over a predefined database. In this paper, we argue that such benchmarks are not suitable for evaluating modern interactive data exploration (IDE) systems, which allow data scientists of varying skill levels to manipulate, analyze, and explore large data sets, as well as to build models and apply machine learning at interactive speeds. While query performance is still important for data exploration, we believe that a much better metric would reflect the number and complexity of insights users gain in a given amount of time. This paper discusses challenges of creating such a metric and presents ideas towards a new benchmark that simulates typical user behavior and allows IDE systems to be compared in a reproducible way.

1 Introduction

There exists an ever *growing set of data-centric systems that allow data scientists of varying skill levels to manipulate, analyze and explore large data sets*. For example, tools like Tableau or Cognos allow users to quickly analyze high-dimensional data using an interactive and visual interface. Research prototypes like imMens [27], DICE [16, 17] or IDEA [9] aim to improve upon systems like Tableau by using specialized data structures, pre-fetching, and/or approximation techniques to guarantee interactive latencies over big data sets. Other research projects like SeeDB [20] or Data Polygamy [5] help users during the exploration process by providing recommendations for interesting visualizations or correlations, whereas systems like DataWrangler [19], Trifacta [45] or Paxata [33] assist users in data wrangling and cleaning.

Although many systems and techniques have been proposed, there is currently no systematic way to evaluate and compare them. For instance, DICE [17] and IDEA [7, 9, 12] both aim to allow users to interactively explore large data sets, but do so through different techniques. Both systems leverage speculative execution, but DICE uses a set of pre-defined exploration paths (e.g., drill-down, roll-up, pivot, etc.) to decide what to speculatively execute, whereas IDEA's model is based on a simpler set of operations which only depend on the current set of visualizations shown on screen. Similarly, both systems try to re-use results between interactions. While DICE finds overlap among different queries using standard relational algebra rewrites, IDEA rewrites queries on the probability level. This allows IDEA, in contrast to DICE, to also reuse incomplete results. Furthermore, IDEA's

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

goal is to support complex analytical workflows including predictive model building. Therefore, IDEA provides an unbounded 2D workspace and a set of pen/touch gestures to support complex analytical workflows. As opposed to DICE, which uses a more traditional interface and focuses mainly on the interactivity of traditional data-cube operations. Even though both systems are designed for interactive analytics, it is hard to draw a comparison due their different approaches and goals. Moreover, systematically comparing a system which provides approximate results, like IDEA, with a system which always calculates the full result, like Hyper [22], is a complicated question on its own. This is due to the fact that it requires determining when an approximate answer meets the same standards as the final answer to the user.

What is needed is a new benchmark for interactive data exploration systems. This is challenging because such a benchmark has to be user focused and the system's performance, which is the main metric of existing analytical benchmarks such as TPC-H [44] and TPC-DS [43], is no longer the most important metric. Moreover, in interactive data exploration systems users incrementally build queries over the course of a session, which is another factor that is not represented in the workloads of the previously stated analytical benchmarks. Arguably the only important metric for data exploration systems is how efficient a user can gain insights from a new data set such as *Insights per Minute*. Clearly, the time to execute a single query has an impact on this metric; the longer queries run, the longer users take to find an interesting insight. What makes it more complicated is that many other aspects have an impact too. First, as IDEA and DICE have demonstrated, reuse of intermediate results and idle time between interactions can be exploited therefore making workflow performance dependent on many other factors. Secondly, the time of fully running a single query matters less. For example, a system which provides 99% accurate answers for 10 queries is – in most cases – superior over a system which provides a 100% correct answer for a single query in the same amount of time. Third, a system that slowly executes some of its queries, but additionally recommends some interesting visualizations to the user or detects common data errors and other pitfalls is arguably the better tool. Finally, the user interface itself often can have a profound impact on how quickly users can make discoveries.

However, measuring the time to insights requires the design, implementation and evaluation of open-ended user studies, which can be time-consuming, expensive and hard to compare (see Section 3.1). Instead we argue for a benchmark that simulates common user behaviour during visual data exploration sessions. While this approach does not allow comparison of all facets of data exploration systems (e.g., the quality of a visual recommendation), we hope it will enable comparing techniques and systems across a multitude of common interactive exploration tasks. Furthermore, such benchmark results can then be augmented with individualized user studies to evaluate features outside of the scope of this benchmark.

Finally, as discussed before, recent interactive data exploration tools not only focus on the aspect of visual data browsing using simple analytical functions but also offer a broad range of other extensions ranging from interactive model building and machine learning over producing visual recommendations to interactive data cleaning. Covering all these aspects is a major challenge when designing a new benchmark.

In this paper, we make the first step towards defining such a new benchmark. We discuss the challenges of simulating users for benchmarking the different components of an interactive data exploration system. We also suggest potential metrics to measure the performance of these systems. The remainder of this paper is outlined as follows: Section 2 describes a potential data exploration session and its various facets; Section 3 talks about insights per minute as a metric and why it is problematic to measure in a benchmark; Section 4 discusses challenges and initial solutions for benchmarking the core functionality of an interactive data exploration systems. Sections 6, 5, and 7 extend upon these ideas to also include other aspects of IDE including visual recommendations, model building, risk evaluation, and data cleaning issues. Section 8 surveys ways to evaluate the overall user experience, and finally, Section 6 concludes.

2 A Motivational Example

To motivate various aspects an IDE benchmark should take into consideration, we outline a fictional data exploration scenario inspired by projects that have recently emerged in this field, such as Vizdom/IDEA [8, 9], SeeDB [20], QUDE [2], and imMens [27].

2.1 Use Case

Eve is a medical researcher at a major Boston area hospital. She obtained a new data set containing information about intensive care unit (ICU) patients including demographics, physical information, disease codes as well blood test results, as available in the MIMIC II data set [29]. She wants to get an overview of the data and gain insights. Eve starts off by visualizing different attributes containing physical and demographic information by creating visualizations depicting the distribution of all ages, weights and heights (Figure 1 A). She notices that patient ages are frequently set to -1, indicating that those values were not correctly recorded. She applies a data cleaning step that replaces all such occurrences with *null* (Figure 1 B) such that they are treated by the system as missing values. Weight and height have the same issue. However, because weight and height are strongly correlated, she decides not to substitute the missing values with *null*, but to perform a different data-cleaning operation that estimates missing values based on correlated attributes (Figure 1 C).

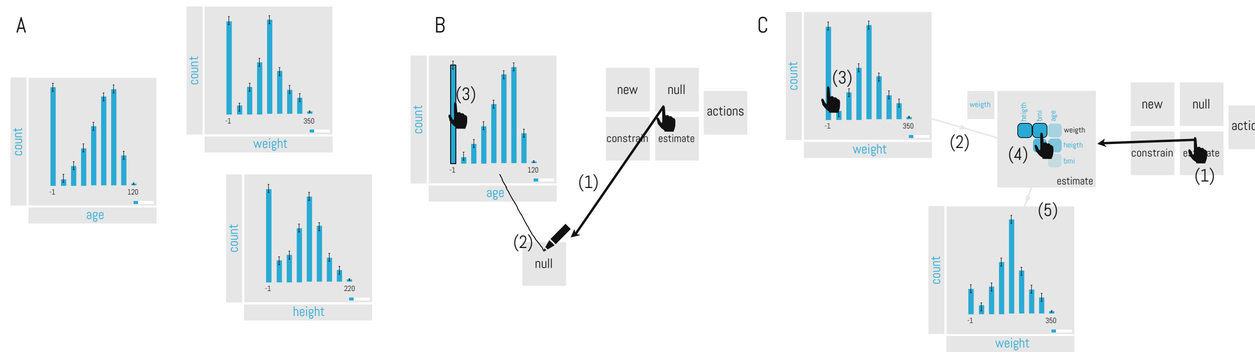

Figure 1: A: visualizations of patients' ages, weights and heights. B: Eve drags out a data-cleaning operator (1), links it to the plot of ages (2) and finally selects the range of age values that she wants to replace with *null*. C: Eve drags out a different data-cleaning operator (1). She links it to the weight visualization (2) and again selects the range of weight values she wants to clean (3). The system shows her the attributes that are most correlated with weight and Eve decides to use height and BMI (4) as the base attributes to estimate missing weight values. She creates a new visualization that shows the resulting weight distribution after this cleaning step.

Next, she is interested in finding out more about age distributions of different diseases. She creates a chain of visualizations that allows her to inspect age distributions of patients with a metabolic disease, and of patients with a heart failure (Figure 2). She then realizes that the two distributions are dissimilar and decides to test the difference for statistical significance. Because Eve substituted "-1' values with *null*, the system automatically knows that it should not consider the *null*-values as part of the permutation test. Optionally, based on these interactions the system could automatically recommend other attributes (i.e., other diseases) that might be of interest to Eve.

Eve now decides to take it a step further and test to see if she can train a classifier to predict whether a patient has a metabolic disease with a set of hand-picked attributes such as age and BMI (Figure 3). The system displays accuracy and other statistics about this model and additionally proposes modifications, such as adding further indicative attributes or changing the underlying prediction algorithm, that will increase the classifier's performance.

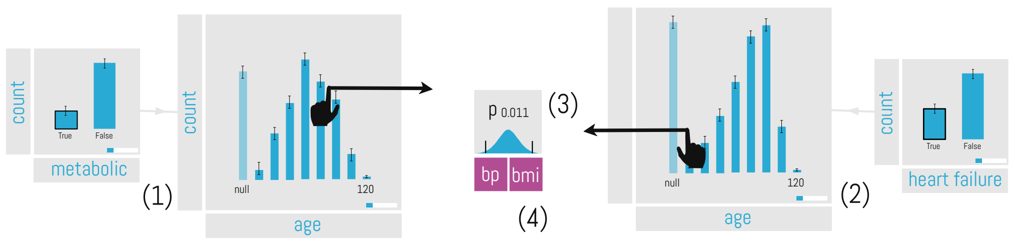

Figure 2: Chains of visualizations that show age distributions of patients with a metabolic disease (1) and heart failure (2). By dragging the two visualizations closer together the system performs a permutation test and displays significance levels (3). The system also displays additional attributes (4) that Eve might be interested in, as they might exhibit similar significance levels when comparing such sub-populations.

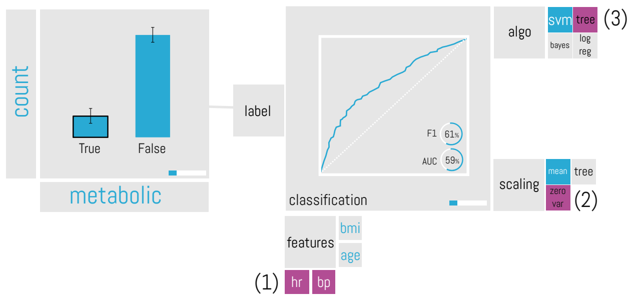

Figure 3: Classifier that predicts if a patient has a metabolic disease or not based on a patient's BMI and age. (1, 2 and 3) shows modifications that the system recommends to increase the model's performance.

2.2 Discussion

This use case exemplifies the potential of modern interactive data exploration tools. It also raises the question of whether or not Eve would have gained a higher number or more valuable insights, if she had used a different system. For instance, the results that were presented to her are approximate visualizations as indicated through the error-bars in Figure 1. Would she have gained more insights if they had been exact but took longer to compute? Would the system's time spent on finding correlations (see Figure 2) be better spent on computing more accurate approximations? Would a system that automatically implies that "-1" should be treated as a *null* value, lead to more discoveries over time?

Conducting user studies that measure the number and complexity of gained insight over time in order to compare different systems or variants thereof may shed more light on these and many other questions. However, such insight-based studies are not well suited for evaluating IDE systems, as we describe in the following section.

3 Benchmarking Interactive Data Exploration Systems

In the following, we first discuss why *Insights per Minute* or time it takes to gain the first insight are problematic as a metric for a benchmark for an IDE benchmark and provide an overview of alternative metrics that capture a similar notion.

3.1 The Problem of Using Insights in a Performance Metric

Ultimately, the goal of interactive data exploration is to extract insights from data. Thus, a system that allows to extract more insights than another system within a given time frame is preferable. However, creating a measure that captures this notion in a comparable and reproducible way is hard. What is an insight? Do different users have different notions of insights? How do we measure the complexity and value of an insight? Are they domain-dependent?

Recent work in the information visualization community has tried to address these questions. Various studies and guidelines [13, 26, 39] proposed to conduct user studies that include open-ended tasks with domain experts. However, evaluating data exploration systems with user studies is problematic for the following reasons: (1) These studies are expensive to conduct and require a lot of manual coding. (2) The results are hard to compare and reproduce. These studies usually have a small number of participants and use within-subject designs making the measured metrics (e.g., insights per minute) unusable for comparisons across different studies. Additionally, different levels of user expertise and that different studies sample from different populations, e.g., students vs. domain experts, further complicate comparisons of study results. (3) It is difficult to design controlled user studies for entire systems. Different user interfaces (UIs) might incentivize different types of insights, e.g., bar chart vs. pie chart, supported complexity of workflows. (4) The value of an insight is unclear. What is more valuable: a simple count comparison or knowing we could build a classifier to predict a label with high accuracy?

3.2 Benchmark Overview

We advocate for a reproducible IDE benchmark that does not factor in the variability of insights, domains, users and user experiences. The core idea is to provide a benchmark that simulates typical user behavior for common data exploration tasks [21, 32], such as filtering, projecting, as well as model building or reacting to recommendations. However, the richer the operations are, the harder they become to benchmark. We therefore suggest a core-set oriented benchmark design where each core-set aims to analyze a different functional aspect of a interactive data exploration system; e.g., one core set only tests simple analytical operations, whereas another one tests more complex model building tasks. Which core-sets are used to evaluate a system, therefore, depends on the supported functionality.

We envision the following four core-sets: *Core-Set I* focuses on Interactive Visual Analysis [21,32] and consists of operations like building filter chains, aggregations, drill-downs, pivoting, etc., as currently supported by systems like Tableau. However, it will exclude interactive model building, which is part of *Core-Set II*. *Core-Set I* and *II* are tightly coupled as it seems unreasonable to assume that one would build a model without having the possibility to efficiently inspect the data set. *Core-Set III* is concerned with benchmarking the recommendation part of a system, whereas *Core-Set IV* outlines metrics to compare the interactive cleaning and risk evaluation parts of a data exploration tool. While a system that supports a higher core-set typically includes the functionality of the core-sets below, we envision that each core-set can be tested individually as discussed before. Furthermore, the higher the core-set number the harder it is to define a benchmark since the sheer complexity of supported operations is increasing and their comparability becomes more difficult. We therefore focus most of our discussion on the lower core-sets and discuss initial ideas for the higher ones.

4 Core-Set I - Interactive Visual Analysis

The core of any IDE system is the capability to browse data through visual interfaces. Techniques like linking and brushing as well as OLAP-like aggregations, traditional statistics, and attribute derivation can help understand complex relationships in the data. The level of support for these operations is determined by various factors including the user interface, the chosen set of default visualizations, and how fluid, i.e., interactive, the system is. As a matter of fact, interactivity is one of the most important aspects as recent studies show. For instance, in [26] the authors argue that latencies of more than $500ms$ can already have a profound impact on discovery rates. In the remainder of this section, we focus on benchmarking the interactivity during Interactive Visual Analysis and discuss more qualitative aspects (e.g., how to evaluate the user interface) in Section 8.

4.1 Objective

The main objective of Interactive Visual Analysis from an interactivity point of view is to ensure that at no point the user is blocked and can always make an informed decision on what to explore next [14, 15]. This requires the system to be fast and responsive, independent of the complexity of a query. Consider a system *A* that provides a visualization to a query within $500ms$ and another system *B*, which takes $10s$ to process the same query. Clearly, *A* is superior over *B*. At the same time, a system which is only $50ms$ faster than another system does not add much value as it makes less of a difference to users. Similarly, a system which provides an approximate answer in $500ms$ and a complete answer for the same query in $10s$, is likely to be superior to a system which forces the users to wait and only returns the complete result after $8s$. On the contrary, short response times (e.g., $500ms$) are barely noticeable to users and therefore only marginally affect the user experience.

To that end, the questions a benchmark should consider are: How does a system, which never provides the full answer but a good approximation compare to a system that eventually provides the complete answer in a reasonable time-frame? What is a reasonable time-frame? How can progressively refining approximate answers be evaluated? How much time is the system given in between interactions? In the following, we outline potential design choices for a benchmark to answer these questions.

4.1.1 Metric

Since we cannot efficiently measure insights per minute directly, we propose a proxy metric called the *Interactivity Performance* metric. One approach would be to measure the time the system takes to provide a good quality estimate for the results of all interactions. Thus, a system, which on average has shorter response times to interactions, would be considered superior. However, as argued before, response times below a certain latency requirement (e.g., $500ms$) are barely noticeable to users and therefore only marginally improve the user experience. Therefore, we suggest a performance metric P that reflects how often and by how much a system violates the latency requirement:

$$P = \sum_{i \in I} max(0; T_q(i) - t_u) \tag{1}$$

Where I is an ordered set of all executed interactions, $T_q(i)$ is the time the system takes to execute interaction T to q-degree of quality and t_u the latency requirement. The smaller P, the better the performance.

Response times in computer systems have been studied extensively for problem solving tasks [38]. It was found that user productivity increases as response time decreases. But systems with a high variability in response time negatively impact user efficiency. However, it remains unclear whether a system which almost always meets the latency requirement with only few major exceptions performs better than one which consistently violates interaction latencies by a little. Potential variations might include different thresholds and weighting schemes (e.g., logarithmic, exponential, etc.) to correctly penalize different system behaviors such as high variability.

There are several ways to automatically determine the quality of an (approximate) visualization. [23] defines a *relationship-based* quality metric as a good approximation in which the relationships between data groups no longer change, i.e., the point in time where in a bar-chart diagram the relationship between two bars (where one is higher than the other) does not change. Another suggested quality metric is the *just noticeable difference* (JND) [42]; a good visualization cannot be visually distinguished from the final complete visualization. In some sense, the JND-based quality metric can be seen as an *error-based* quality metric with a specific error constant. That is, a good visual approximation lies within a pre-defined error-bound of the ground-truth visualization. More specifically, $T_q(i)$ is the time it takes the system to produce a visualization for interaction i within an error-bound of ϵ. We plan to conduct a user study to evaluate which of these quality metrics works best as a placeholder for real-world users.

Finally, it is worth noting that the combination of the error-based quality metric and the Interactivity Performance P allows for a direct comparison of visualization error and the maximum time allotted to compute it. Varying the error bound-threshold ϵ and the maximum latency t_u could help to better understand the trade-offs between computation time and latency. For instance, while it takes one system to compute an approximate visualization within $500ms$ and error threshold 5%, another system may constantly perform better after $1s$.

4.2 Workload

Having the main metric defined, we still need a way to simulate the user. As described earlier, in IDE systems users incrementally build queries over the course of a session. Idle time between interactions is often used by systems to prepare for the next interaction and in contrast to many existing benchmarks, the time users take between interactions varies a lot. In many cases interactions depend on each other and allow for reuse of partial results among other things. As a result, a benchmark for visual data exploration has to somehow simulate users' behavior with their typical think-time between interactions.

One method to derive realistic workflows is to study actual user behavior and synthesize them to exemplary workflows as done in [9]. By providing a set of potential IDE sessions the simulated users can capture common user behavior. This allows for adjustments to the synthesized session (e.g., from more novice to expert users). One idea is that the think-time between interactions could be scaled up or down similar to scaling the data size to increase the level of difficulty in a benchmark. While creating the different workflows it is important to cover different aspects of the data. For instance, one workflow could consist of mainly unrelated browsing queries, where users just look at different attributes in isolation, while another workflow could comprise the creation of a deep analytics pipeline. Similarly, as for other benchmarks, simulated users should vary their interactions (e.g., as [9] shows users tend to investigate outliers as they might contain interesting information).

Finally, a benchmark for visual data exploration should also allow to vary the data and data size. To test the different properties of the system it is important that the generated data follows different types of distributions for various attributes and contains random and correlated data. One way to generate such a data set is to take an existing data set (e.g., the flight data set from [1]) and provide ways to automatically scale it to the desired size while preserving the most important data properties.

4.3 Reporting

An important aspect of a benchmark is to define the main configurations for which the result should be reported. We suggest that users of the benchmark can select a scale-factor of the benchmark just as in traditional benchmarks. In addition, the benchmark should define different configurations such as the *browsing* configuration or the *no think-time* configuration, which have different values for the latency threshold t_u, the think-time between interactions and the error-based quality metric. These configurations allow the user to configure the benchmark to the targeted use case of the IDE system, while still making all systems comparable using the standard settings.

5 Core-Set II - Interactive Model Building

Apart from supporting users in visual analysis tasks, modern IDE systems increasingly help to interactively test the predictive power of attributes and build models [8, 9, 24, 41]. Some of the common tasks in model building, such as visually inspecting attributes for feature selection, are covered by Core-Set I. Others, such as model selection, hyper-parameter tuning etc., need to be evaluated separately and require IDE systems to potentially train and test hundreds of different configurations. While some progress has been made on how to do this with interactive latency guarantees [9, 37, 40], it still remains a challenge to compare different interactive model training systems and techniques.

5.1 Metric

Arguably, the most relevant metric is Time to Model, the time it takes users to derive a model with a satisfying model quality. This, yet again, requires user studies with all its difficulties. Thus, analogous to the ideas for Core-Set I, a better metric for Core-Set II captures how long it takes a simulated user to derive a model with satisfying quality.

Interestingly, estimating the model quality can also be considered a result approximation. Thus, like for Core-Set I, we use the *Interactivity Performance* metric (see Equation 1). However, in this case we must measure the quality of a computed model rather than the quality of a visualization. More specifically, given a workload that exactly specifies which features and model class the system must use (e.g., train an SVM to predict if a patient has a metabolic disease based on its age and BMI, as done in Figure 3), with a pre-defined validation method (e.g., 10-fold cross-validation), we can compute the expected "ground-truth" model quality, e.g., the F-score, offline. This allows us to compare the required time to achieve a certain model quality of a system to the best known model quality, again represented as an F-score.

Furthermore, this metric also allows comparisons of more complex model search strategies. For example, it might be possible to allow the system to do automatic algorithm and feature selection, or even feature transformation. In this case, we envision that the workload only defines a high level task and the system reports the model quality over time. This is then compared to the best known model quality, again represented as an F-score.

It should be noted, though, that different time weighting schemes as well as using the F-score as a quality model have implicit assumptions, which may or may not reflect the users intention. For instance, the F-score is the harmonic mean of precision and recall, yet especially in the medical context often precision is more important than recall.

5.2 Workload

Since model building is strongly intertwined with the tasks discussed for Core-Set I, the workload for Core-Set II should also include operations such as browsing, inspecting attributes, etc. In the workload of this core-set model building tasks should thus be intertwined with simple OLAP-style operations to inspect the data. As most IDE systems are not yet capable of full automatic model building as proposed in MLbase [24], the workload should also support different levels of specificity. Highly specific configurations dictate which algorithm and parameters to use, whereas non-specific ones merely require the system to return predictions regardless of models class, parameters, and features used.

5.3 Reporting

Core-Set I and Core-Set II essentially use the same metric. Therefore, results can be reported as described in Section 4.3. In cases where a strict latency requirement matters less, we suggest reporting the F-score metric for each trained model within the simulated workflows.

6 Core-Set III - Recommendations

There has been an increasing interest in automatically presenting recommendations to the user when using IDE systems. These recommendations often come in different forms; e.g. suggesting visualizations [20], or queries [28], recommending data cleaning steps [45], pointing out potential correlations [5] etc. Evaluating and comparing the quality of such recommendations is difficult as they typically depend on the data domain and the history of interactions.

6.1 Metric

Prior work in this area often use their own metric of success. The visual recommendation system SeeDB [20], for instance, rewards recommendations that are orthogonal to what the user has looked at so far, thus it tries to maximize the coverage of the data space (breadth first). Other approaches put more weight on recommendations that are based on a sub set of the data that is currently being looked at (depth first). Depending on the task at hand one of these objectives will be favorable over the other. We advocate that a benchmark for recommendations be flexible enough to accommodate for multiple metrics and objectives.

The actual benchmark for recommendations as part of IDE can be designed along the lines of matrix completion problems as used by movie recommendation engines. That is, given a labeled set of "good"' recommendations it is the system's task to identify them. Being able to situationally identify such "good"' recommendations is not the only important aspect. No user wants to inspect 1,000 recommendations. Instead, like with search engines, only the top-k recommendations really matter. The best recommendations are not useful if they are displayed too late. Because of this a quality metric like Mean Average Precision (MAP), as used within the Information Retrieval community, might be the right choice to compare the quality of two recommendation systems. As before with our Interactive Performance Metric we suggest to penalize individual MAP scores for recommendations that take longer to compute than some latency threshold.

6.2 Workload

The workload for benchmarking the recommendations produced by the IDE system can be similar to the workload of Core-Set I. In fact, the recommendations should not be benchmarked in isolation as it will be important to see how the system computes them based on ongoing user interactions and how the system prioritizes the process of creating the visualizations vs. supporting the ongoing user interactions.

The challenge in creating the benchmark lies in pre-labeling the best set of recommendations during every single step of the predefined workflows. Here, the large amount of publications on how to create benchmarks for search engines that tackle a similar problem might be relevant.

6.3 Reporting

While we are potentially able to use the same Interactivity Performance metric as for the other core-sets, it is still important to provide detailed results on the quality of recommendations in form of the individual MAPs, ROC curves, etc.

7 Core-Set IV - Risk and Data Quality

The last proposed core-set of the benchmark is concerned with evaluating the capability of the system to clean and integrate data and to detect risk factors (e.g., [6, 46]). While there has been some work on creating benchmarks for certain sub-problems related to data quality (e.g., XBenchMatch [11] and RODI [34] for schema mapping or TPC-DI [36] for data integration in general), other sub-problems are still evaluated in a more ad-hoc manner. For de-duplication, people often use a few "known" data sets, such as the restaurant data set in [25, 47], but to the best of our knowledge no standardized benchmarking suite exist. Similarly, there is no good benchmark for "data wrangling" [18], which is surprising given the importance it has for data scientists.

Furthermore, there has been very little work helping the user avoid common pitfalls during the analysis process. For example, visualizations can be misleading as they sometimes hide certain details. When analyzing the age of the patients in the MIMIC-II dataset we observed that a significant number of patients were between the ages of $0-20$-years-old. We originally believed that this is normal as kids, especially infants, are quite often in the emergency room. Only later we discovered, that many records which had an age of 0 since many doctors apparently use 0 if the age is unknown (like the previous mentioned -1 in Figure 1).

Similarly, high-level statistics can be misleading (e.g., resulting in the infamous Simpson Paradox), or recommendations can be extremely dangerous if not done with care. For example, in a recent paper [3] we showed that visual recommendation or correlation finders can significantly increase the risk of finding insights, which just appear by chance [2] (i.e., caused by the multi-hypothesis problem). In fact, interactive data exploration tools, even without recommendations, increase the chance of false discoveries as they enable the user to test many more hypothesis than ever before.

As part of the last core-set we plan to evaluate the system's capability to clean and integrate data as well as its ability protect users from making (common) mistakes. For example, we envision that as part of the benchmark requirements, the system lists its data integration capabilities and the types of common mistakes, the system tries to protect the user from. A more advanced benchmark design could contain certain types of data quality issues and potential pitfalls (e.g., a Simpson Paradox). That way systems could be evaluated in how many of these problems are detected as part of a workflow or how many system's operations were needed to correct the data error. This requires a very careful design of the data generator and might even entail adjustments to the attribute distributions based on the scale factor. Another idea is, that the system is tested with risk-control on and off as part of the Core-Sets I-III. This way the computational overhead of these protections can be factored in and measured.

However, the feasibility of such a benchmark for Risk and Data Quality control has yet to be determined and many questions remain unanswered.

8 Evaluating the User Experience

Evaluating the effectiveness of user interface designs, information visualizations and interaction techniques for data exploration is an ongoing area of research [4,30]. On one hand, the composition of a set of tasks, e.g., load a dataset, plot the distribution of some attribute and filter based on another attribute, can favor one tool over another [35]. This bias is introduced through different factors in the complexity, the design, and the usability of the software, such as menu design, interaction techniques and defaults like color encoding. On the other hand, it is hard to conduct "fair" evaluations because tasks used in laboratory user studies don't normally reflect real-world tasks executed by domain experts. The tasks often need to be shortened and simplified so they can be accomplished in a given amount of time. However, simple tasks might lead to fewer discoveries.

In order to compare design choices within the same system, Munzer et al. [30] presented a model that defines various levels of design, each with its corresponding evaluation methodology. TouchViz [10], for instance, conducted a controlled study that compared the effect of two different interaction paradigms (gestural vs. *W*indows-*I*cons-*M*enus-*P*ointer) by measuring task completion times and task accuracy. In cases where these measures are not relevant (e.g., open-ended data exploration), like [31], we advocate for an open-ended protocol where researchers observe what insights domain experts gain while exploring the data in a self-directed manner over the course of a long-term study.

9 Conclusions and Future Work

In this paper, we presented challenges and initial ideas towards a benchmark for IDE systems. We divided the benchmark into four Core-Sets: (I) Interactive Visual Analytics, (II) Interactive Model Building, (III) Recommendations, and (IV) Risk and Data Quality. For each Core-Set we proposed a potential workload and a metric. As part of [9] we already started to develop an initial benchmark for Core-Set I and currently work on releasing that part of the benchmark publicly. In addition, we are actively looking for collaborators to design a concrete benchmark for the other Core-Sets. We further hope to achieve a standardized benchmarking suite like the TPC-benchmarks, which have been driving innovation for years and define a whole industry.

References

[1] Airline dataset. `http://www.stat.purdue.edu/~sguha/rhipe/doc/html/airline.html`. Accessed: 2016-10-17.

[2] C. Binnig, L. De Stefani, T. Kraska, E. Upfal, E. Zgraggen, and Z. Zhao. Towards sustainable insights, or why polygamy is bad for you. In *To appear in CIDR*, 2017.

[3] C. Binnig, L. De Stefani, T. Kraska, E. Upfal, E. Zgraggen, and Z. Zhao. Towards sustainable insights or why polygamy is bad for you. In *CIDR 2017*, 2017.

[4] S. Carpendale. Evaluating information visualizations. In *Information Visualization*, pages 19–45. Springer, 2008.

[5] F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire. Data polygamy: the many-many relationships among urban spatio-temporal data sets. In *SIGMOD 2016*, pages 1–15. ACM, 2016.

[6] Y. Chung, M. L. Mortensen, C. Binnig, and T. Kraska. Estimating the impact of unknown unknowns on aggregate query results. In *SIGMOD 2016*, pages 861–876, 2016.

[7] A. Crotty, A. Galakatos, K. Dursun, T. Kraska, C. Binnig, U. Çetintemel, and S. Zdonik. An architecture for compiling udf-centric workflows. *PVLDB*, 8(12):1466–1477, 2015.

[8] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska. Vizdom: interactive analytics through pen and touch. *Proceedings of the VLDB Endowment*, 8:2024–2027, 2015.

[9] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska. The case for interactive data exploration accelerators (IDEAs). In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, page 11. ACM, 2016.

[10] S. M. Drucker, D. Fisher, R. Sadana, J. Herron, et al. Touchviz: a case study comparing two interfaces for data analytics on tablets. In *SIGCHI 2013*, pages 2301–2310.

[11] F. Duchateau, Z. Bellahsene, and E. Hunt. Xbenchmatch: a benchmark for XML schema matching tools. In *VLDB 2007*, pages 1318–1321.

[12] M. El-Hindi, Z. Zhao, C. Binnig, and T. Kraska. Vistrees: fast indexes for interactive data exploration. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics, HILDA@SIGMOD 2016*, page 5, 2016.

[13] H. Guo, S. R. Gomez, C. Ziemkiewicz, and D. H. Laidlaw. A case study using visualization interaction logs and insight metrics to understand how analysts arrive at insights. *IEEE Trans. Vis. Comput. Graph.*, 22:51–60, 2016.

[14] P. Hanrahan. Analytic database technologies for a new kind of user: the data enthusiast. In *SIGMOD 2012*, pages 577–578.

[15] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *Queue*, 10:30, 2012.

[16] P. Jayachandran, K. Tunga, N. Kamat, and A. Nandi. Combining user interaction, speculative query execution and sampling in the dice system. *VLDB 2014*, 7:1697–1700.

[17] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *ICDE 2014*, pages 472–483.

[18] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck, and P. Buono. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization*, 10(4):271–288, 2011.

[19] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *SIGCHI 2011*, pages 3363–3372.

[20] M.-T. Ke, S. Fujimoto, and T. Imai. Seedb: a simple and morphology-preserving optical clearing agent for neuronal circuit reconstruction. *Nature Neuroscience*, 16:1154–1161, 2013.

[21] D. A. Keim. Information visualization and visual data mining. *IEEE Trans. Vis. Comput. Graph.*, 8(1):1–8, 2002.

[22] A. Kemper and T. Neumann. Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots. In *ICDE 2011*, pages 195–206.

[23] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid sampling for visualizations with ordering guarantees. *VLDB 2015*, 8:521–532.

[24] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan. Mlbase: A distributed machine-learning system. In *CIDR*, volume 1, pages 2–1, 2013.

[25] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, T. Kraska, T. Milo, and E. Wu. Sampleclean: Fast and reliable analytics on dirty data. *IEEE Data Eng. Bull.*, 38(3):59–75, 2015.

[26] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Trans. Vis. Comput. Graph.*, 20:2122–2131, 2014.

[27] Z. Liu, B. Jiang, and J. Heer. immens: Real-time visual querying of big data. In *Computer Graphics Forum*, volume 32, pages 421–430. Wiley Online Library, 2013.

[28] P. Marcel and E. Negre. A survey of query recommendation techniques for data warehouse exploration. In *Actes des 7èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne, EDA 2011*, pages 119–134.

[29] Mimic-iii, a freely accessible critical care database. `https://mimic.physionet.org/`. Accessed: 2016-10-17.

[30] T. Munzner. A nested model for visualization design and validation. *IEEE Trans. Vis. Comput. Graph.*, 15:921–928, 2009.

[31] C. North. Toward measuring visualization insight. *IEEE Computer Graphics and Applications*, 26:6–9, 2006.

[32] S. Oeltze, H. Doleisch, H. Hauser, and G. Weber. Interactive visual analysis of scientific data. In *Presentation at IEEE VisWeek 2012*, 2012.

[33] Paxata. `http://www.paxata.com`. Accessed: 2016-10-17.

[34] C. Pinkel, C. Binnig, E. Jiménez-Ruiz, W. May, D. Ritze, M. G. Skjæveland, A. Solimando, and E. Kharlamov. RODI: A benchmark for automatic mapping generation in relational-to-ontology data integration. In *European Semantic Web Conference, ESWC 2015*, pages 21–37.

[35] C. Plaisant. The challenge of information visualization evaluation. In *ACM Working Conference on Advanced Visual Interfaces 2014*, pages 109–116.

[36] M. Poess, T. Rabl, and B. Caufield. TPC-DI: the first industry benchmark for data integration. *PVLDB*, 7(13):1367–1378, 2014.

[37] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[38] B. Shneiderman. Response time and display rate in human performance with computers. *ACM Computing Surveys (CSUR)*, 16(3):265–285, 1984.

[39] B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *ACM AVI Workshop on Beyond Time and Errors: novel evaluation methods for information visualization 2006*, pages 1–7.

[40] E. R. Sparks, A. Talwalkar, D. Haas, M. J. Franklin, M. I. Jordan, and T. Kraska. Automating model search for large scale machine learning. In *ACM SoCC 2015*, pages 368–380.

[41] E. R. Sparks, A. Talwalkar, V. Smith, J. Kottalam, X. Pan, J. E. Gonzalez, M. J. Franklin, M. I. Jordan, and T. Kraska. MLI: an API for Distributed Machine Learning. In *ICDM 2013*, pages 1187–1192, 2013.

[42] M. K. Stern and J. H. Johnson. Just noticeable difference. *Corsini Encyclopedia of Psychology*, 2010.

[43] TPC-DS. `http://www.tpc.org/tpcds/`, 2016. Accessed: 2016-10-17.

[44] TPC-H. `http://www.tpc.org/tpch/`, 2016. Accessed: 2016-10-17.

[45] Trifacta. `http://www.trifacta.com`. Accessed: 2016-10-17.

[46] B. Trushkowsky, T. Kraska, M. J. Franklin, and P. Sarkar. Crowdsourced enumeration queries. In *ICDE 2013*, pages 673–684, 2013.

[47] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.