Trust Me, I'm Partially Right: Incremental Visualization Lets Analysts Explore Large Datasets Faster

Danyel Fisher[*], Igor Popov[†], Steven M. Drucker[*], mc schraefel[†]

[*]Microsoft Research
1 Microsoft Way,
Redmond, WA USA
{danyelf, sdrucker}@microsoft.com

[†]Electronics and Computer Science
University of Southampton
Southampton, Hampshire, UK SO17 1BJ
{mc+w, ip2go9}@ecs.soton.ac.uk

ABSTRACT

Queries over large scale (petabyte) data bases often mean waiting overnight for a result to come back. Scale costs time. Such time also means that potential avenues of exploration are ignored because the costs are perceived to be too high to run or even propose them. With *sampleAction* we have explored whether interaction techniques to present query results running over only incremental samples can be presented as sufficiently trustworthy for analysts both to make closer to real time decisions about their queries and to be more exploratory in their questions of the data. Our work with three teams of analysts suggests that we can indeed accelerate and open up the query process with such incremental visualizations.

Author Keywords

Incremental visualizations, large data, exploratory data analysis, online aggregation.

ACM Classification Keywords

H.5.2. Information interfaces and presentation: Miscellaneous. H.2.8. Database Applications: Data mining.

General Terms

Experimentation, Human Factors

INTRODUCTION

The increased capacity to capture data from systems and sensors that generate it, from social networks to highway traffic flows, gives us an unprecedented opportunity to interrogate behavior from the individual to the complex system. Unfortunately, the speed at which this data can be explored, and the richness of the questions we might ask are currently compromised by the cost in time and resources of running our queries over such vast arrays of data. We have reverted to a batch-job era, where users formulate a query, wait for some time, and evaluate the results—a step backwards from the interactive querying that we expect in exploratory data analysis.

Of course the database community has attempted to reduce query costs in a variety of ways. Strategies to accelerate large scale data processing are represented in systems like Dremel [9] and C-Store [18] that churn through large collections of data by pre-structuring the data and moving the computation closer to the data.

So while computational and storage approaches make large scale queries possible, they still often restrict either the number and types of queries that might be run, or avenues that might be explored because the queries must be designed with such care to be worth the wait and the cost of queuing for the resource.

One possible technique, proposed by Hellerstein and others [7], is to query databases incrementally, looking at ever-larger segments of the dataset. These samples can be used to extrapolate estimated final values and the degree of certainty of the estimate. The analyst would get a response quickly by considering a large, initially unclear range of values that rapidly converge to more precise values. This approach may let an analyst iterate on a query with substantially decreased delay and increased flexibility: if the way forward is sufficiently clear from the samples, they can quickly refine queries or explore new parameters.

There is an important interaction issue here. Analysts are accustomed to seeing precise figures, rather than probabilistic results, and may not be willing to act on partial information. Confidence intervals add a degree of complexity to a visualization, and may simply be confusing. In order for incremental analysis to be a viable technique, it will be important to understand how analysts interact with incremental data.

Most research in incremental queries has gone into the technical aspects of the back-end [3,6]; we complement that technical agenda with an investigation of the interaction design challenges involved. Our exploration presented in this paper is two-fold: the production of an application with sufficient fidelity that will allow users to experience converging iterative estimates of their own data, and, in particular, to understand how this interaction enables an exploratory analysis process.

In this paper, we present *sampleAction*, a tool that allows us to simulate the effects of interacting with very large datasets while supporting an iterative query interaction for large aggregates. Our simulator allows us to examine how

CHI '12, May 5-10, 2012, Austin, Texas, USA.

both user interfaces and data storage concepts may be effectively redesigned to be amenable to our incremental interaction approach. In our evaluation, we carried out in-depth interviews with members of three different teams of data analysts, working in three different areas. The analysts reconstructed a series of queries from their own data in our system. We found that, after they had examined only a small fraction of the database with our interface, they overall found our representations of the incremental query results sufficiently robust that they were prepared either to abandon that query, refine it further, or create new ones previously unconsidered. Significantly, they were able to make these iterations rapidly, in real time.

We contribute, first, a methodology for simulating aggregate queries against large data back-ends; we hope that this will allow researchers to more broadly explore the interaction issues that arise in this area. Second, we contribute observations of expert analyst behavior in interacting with approximate queries.

Our paper draws on past research about incremental, approximate queries [4], as well as visualizations of uncertainty [15]. We present past work, followed by a rationale for the design and implementation approach and an overview and analysis of our sessions with the analysts. We discuss how our system enables analysts to make their decisions on incremental samples and the implications of our design approach for enhancing flexible data exploration.

BACKGROUND AND RELATED WORK

In this project, we visualize estimates on incremental data. Incremental analysis is an alternative to other techniques that are more familiar, but have disadvantages compared to our method. In this section, we first discuss these techniques in order to motivate incremental data analysis. We then discuss techniques for visualizing uncertainty, which we adapt for our visualization.

Background on Handling Large Data for Visualization

Information visualization is a popular way to help analysts make sense of large datasets. It allows an analyst to overview data quickly by seeing summary statistics, compared easily, through a selection of charts.

Many visualizations are based on aggregate queries against of datasets. A dataset can be thought of as a table of data, made up of *measures*—the values being visualized—and *dimensions*, the categories into which the measures are divided. For example, in a sales database, an analysts might choose to create a bar chart (the visualization) showing average sales per customer (the measure) divided by different products (a dimension). In exploratory data visualization, it is common to rapidly iterate through different views and queries about a data-sets. In contrast, visualizations for reporting or presentations are usually prepared in advance and allow limited interaction.

In a very large dataset, exploratory visualization becomes onerous: each query can take hours or days to compute before a result is ready to be seen on screen. There are several ways to deal with visualizing very large datasets. The simplest technique is to wait through a long processing job, allowing the job to run overnight. This has the virtue of *precision*, but loses out on *speed.* In particular, by waiting for hours for each query, a user writes off the possibility of iteratively exploring their dataset.

Dremel [9] and other scale-out architectures have massively increased the speed of accessing data rows. These architectures are expensive, though, both in money and energy. An incremental database can help save computation costs by looking at fewer rows and spinning fewer disks. In addition, even large-scale architectures can be overwhelmed by ever-larger datasets, and datasets where it is expensive to access rows.

The user can save computation time on queries by building an OLAP (Online Analytical Processing) cube [1]. An OLAP cube pre-aggregates a database by specifying dimensions and measures in advance. This allows users to explore those aggregated dimensions, at the cost of another long processing job. The results also limit flexibility, as users cannot easily add new dimensions after the cube has been built.

If the dataset is well-organized as tabular data, a user might take a fixed-size sample and use exploratory visualization on the sample, using off-the-shelf software like Tableau[1]. Indeed, Tableau has an ability to handle samples from a large dataset, selected randomly. Tableau then allows rapid queries against the in-memory portion of the dataset. These queries can be interactive, but, as they are based on a sample, they cannot be precise—and the system does not provide a way for the user to know how good an approximation is the sample of the full dataset. Extending the idea of samples, Infobright [16], has explored the idea of using approximate SQL to allow for more responsive queries, although the estimates do not improve incrementally.

With incremental, approximate analysis we avoid the difficulties of these approaches. Incremental analysis collects ever-larger samples in the back-end, and uses them to estimate the true value of a query. In addition, approximate queries can present confidence bounds: the region in which the final value is likely to fall.

The system can respond quickly and flexibly as it acts on samples; over time it gains accuracy. Because the system is based around samples, it computes estimated values, rather than definitive results. In addition to the estimate, the system can compute a confidence interval for many types of

[1] http://www.tableausoftware.com/

aggregate queries. This interval predicts the possible range of the true value.

The CONTROL project initiated the area of online incremental data analysis; in the course of a series of papers [3,4,5], the project laid out an agenda for incremental analysis and laid out a technical infrastructure. Later projects by Jermaine and colleagues further explored incremental database queries [6,7]. All of these projects produce output including estimates and confidence intervals; several have prototyped possible visualizations, although none of them evaluate these visualizations.

Incremental Visualization

This project uses uncertainty visualization techniques to monitor estimates on incremental data. Neither of these component ideas are new individually: both Olston and Mackinlay [11] and Fisher [2] argue that uncertainty visualization would be a valuable output for incremental techniques. Neither paper on this work, however, offers an implementation nor examines how users would respond to the combined system.

Uncertainty Visualization

The research community has had a standing interest in visualizing uncertain data: researchers have addressed data that is uncertain in its values, quality, provenance, and even in its structure [15,20]. Researchers have experimented with a number of different visualizations of uncertainty, including error bars, translucency and fuzzy regions [8,18]. However, several evaluations [14,20,21] have found that more exotic schemes can be difficult for users to interpret. The most applicable approach from this work is 'statistically uncertain visualization.' Both Olston and Mackinlay [11] and Sanyal *et al* [14] refer specifically to uncertain data values that have known properties, such as bounds or probability ranges. In an original CONTROL paper [4], the authors suggest both a bar chart with error bars, and a "cloud" plot to represent multidimensional data.

We leverage these past approaches by representing confidence intervals as error bars: both confidence intervals and error bars refer to an expected range of values, and error bars are familiar representations in statistics.

METHOD

The purpose of the work we present here has been to understand whether data-intensive users would be willing to make decisions on the fly using incremental visualizations in order to engage in exploratory data analysis. We wanted to understand whether they found the concept of dealing with confidence bounds confusing, and at what point a visualization's bounds are "good enough" to act upon. We also wanted to understand whether providing an interactive exploratory front-end encouraged them to learn about their datasets in new ways.

Our hypothesis is that users working with incremental visualizations will be able to interpret the confidence intervals comfortably. We further hypothesize that this will allow them to act rapidly on their queries. Last, we hypothesize that incremental results will allow users to carry out exploratory queries.

Experimentation

To interrogate our hypotheses, we do not need to implement an incremental database system at full scale. Rather, we need to produce a realistic experience that allows users to understand what using an incremental database would be like. The system must, of course, incrementally update samples, showing ever-larger portions of the dataset and the estimates that emerge from them. In order to maintain ecological validity, the system should work with data that is familiar to the analyst, and should allow the analyst to experiment with a broad assortment of queries.

To that end, we have created a tool that allows us to simulate the experience of using a very large dataset. Analysts provide us with a table of data. In turn, we enable the analysts to execute a variety of queries, while incrementally displaying results based on ever-larger portions of the dataset. This allows users to work with their own data while exploring our interface. We called this system *sampleAction. sampleAction* allows a user to formulate a query visually. The system responds with a partial result, displaying a bar chart with confidence bounds; as the analyst waits, the system increases its sample size, narrowing the confidence intervals and producing more precise results (Figure 1).

sampleAction uses estimators that produce error bounds to predict the final values; these estimators base their results on the size of the sample, rather than the size of the database, and so can scale up with the data.

System Implementation

In the following sections we first describe the front end interaction that drives the queries and represents the incremental responses. We next describe the statistical reasoning used to present the confidence in the samples presented and close this section with the description of the database implementation supporting the simulation.

Interface for Formulating Queries

The front-end interface of *sampleAction* allows users to execute basic aggregate queries (averages, sums, counting) typically used in exploratory data analysis, in an environment that resembles analytics tools like Tableau. The user can connect to arbitrary database tables, and, without any knowledge of the underlying query language, use the graphical user interface to create visual summaries of database queries with filters, sorting, and groups.

An initial screen allows users to connect to an SQL server, select a table from that database and open a dashboard over which analytics over the data in the table can be performed. The major screen in *sampleAction* is the *Analytics* panel (Figure 1), which allows users to compose aggregate queries over the data table. An analyst can drop one or more *dimensions* onto the column box, and a *measure* onto the

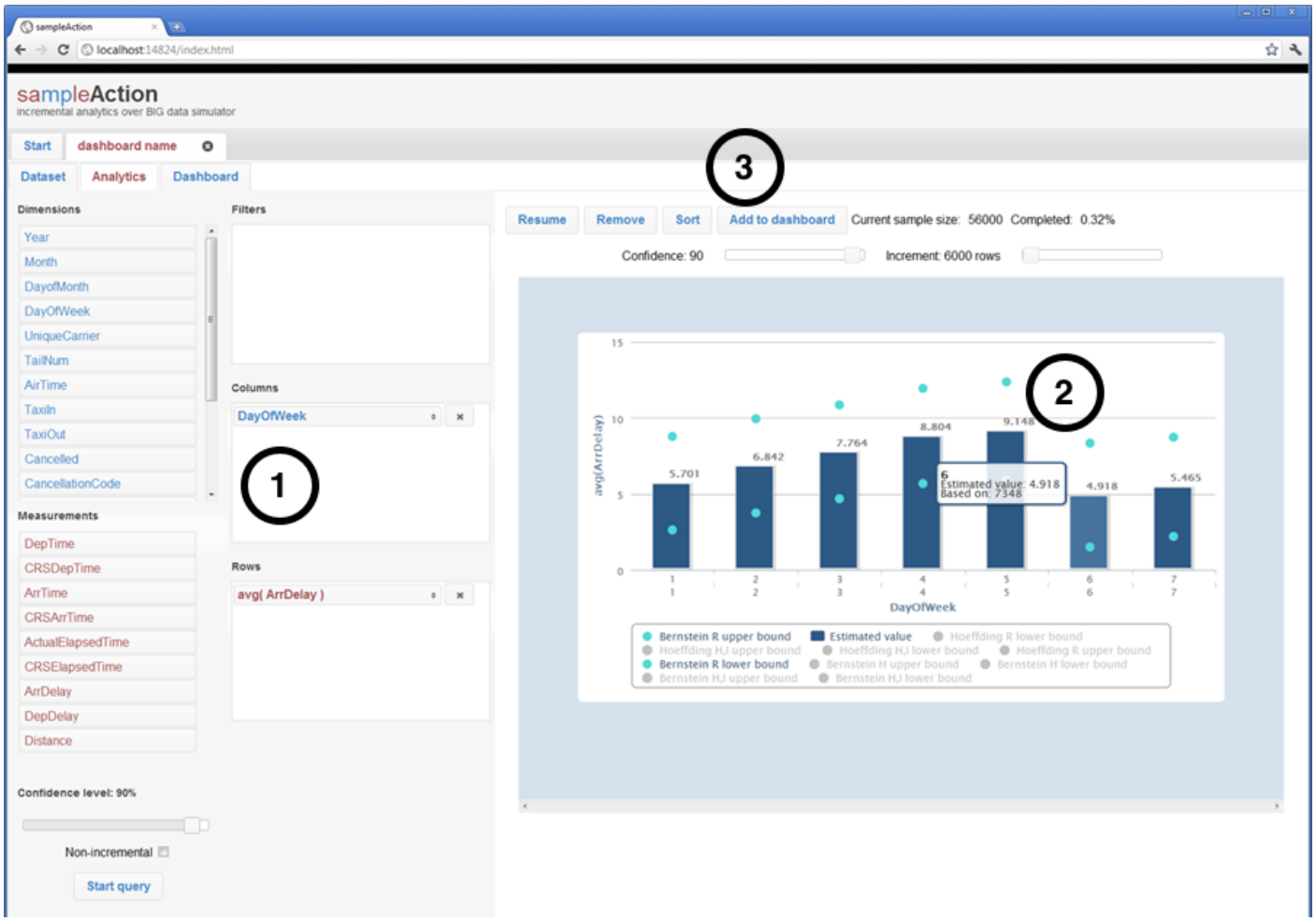

Figure 1. The Analytics panel in sampleAction showing an incremental visualization in progress. The analyst is looking at flight delays by day of week. (1) Selecting columns to be shown in (2) the visualization. Dark blue bars show current estimates; pale blue dots show the expected range of values. This prototype interface includes multiple selectable bounding algorithms. (3) A progress indicator showing that 0.32% of the database has been seen so far.

rows box (Figure 1-1). The "filter" box allows an analyst to create a filter on either a dimension or measures. In Figure 1, for example, the analyst performs a query over an FAA (US Federal Aviation Administration) database of flight delays, showing the average arrival delay by day of the week.

Visualization of Queries

When a query is issued, the system sends it to the back-end, which computes and returns an estimate and confidence bounds; the front-end displays a chart of the results. The estimate and bounds are updated every second with more rows of data.

sampleAction displays all aggregates with a column chart. We chose column charts for their familiarity and versatility. The column charts is annotated with error bars (Figure 1) which are shown around each column. The error bars show the confidence bounds around the resulting data value. The error bars show the range of values that may occur at the confidence levels, while column height itself shows the current estimated value. For example, in Figure 1, an analyst can conclude that—with 90% probability—the true average delay on Friday (day=5) is somewhere between 6 and 12 minutes, while on Saturday is between 2 and 8 minutes. These conclusions are drawn by looking at 56000 rows, just 0.32% of the full database.

sampleAction uses error bars to show the values of the estimate. However, there are new parameters that are not common in standard exploratory data visualization systems, which *sampleAction* is able to show (Figure 1). The display shows the number of rows of data examined so far, and how much of the total dataset this represents. A tooltip (Figure 1-2) allows the user to know the number of datapoints that were used to compute a given estimate. Last, sampleAction shows how the bounds are changing over time (Figure 2); this can help an analyst decide how much longer it is worth waiting for more data.

An analyst can pause or stop the incremental process at any time; in the current implementation, analysts can also start additional queries while the previous ones are still running.

Figure 2. Convergence of confidence bounds for a given column as the database reads in more columns based on two different formulae. We experimented with different bounds in order to better understand convergence behavior.

All queries will continue to add samples and slowly converge.

Bounded uncertainty based on samples

The back-end of *sampleAction* computes responses based on queries from the front-end. In order to supply the information in the UI, the responses that it sends back are somewhat more complex than standard SQL query result sets: in addition to returning a set of values, queries also return confidence bounds and the number of rows seen.

The choice of appropriate bounds is at the heart of the *sampleAction* system. Bounds should appropriately bound the data—that is, they need to represent the highest and lowest likely values of the true value. If the bounds are too wide, users will gain little information about the estimate. If bounds converge too slowly, incremental visualization will be little better than waiting overnight for a precise value.

Computing statistically accurate bounds requires random samples from the dataset in order to ensure that the sample is unbiased. As a result, incremental results need to be selected from a randomly-ordered stream. The efficient computation of random samples from a database is a well-known problem (Olken and Rotem [10] survey techniques from 1990). This sampling can be accomplished by selecting randomly from the table, which is computationally expensive. Alternately, we can randomly order the database, which potentially interferes with index structures or requires a redundant copy of the data. These tradeoffs are active areas of research in the database community. For the goals of this project, we randomly ordered the dataset.

Given its stream of samples, *sampleAction* computes an estimate of the expected value of an aggregate of the stream. The tool uses the rows processed so far in order to make an estimate of the value based on the full dataset, as illustrated in Figure 3. For the purposes of estimating a value, we treat each category of a group-by query as separate. For example, if we are querying for total sales, grouped by state, then *sampleAction* will estimate fifty different values. The tool attempts to estimate the true value of all rows that match the query (Figure 3-2); however, it has only seen a subset of rows (Figure 3-4). The estimate is based therefore on those rows which it has seen already and that match the query (Figure 3-3), which can be a much smaller subset. In a fixed-size sample, this fraction could mean that the estimated values might be very inaccurate; in an incremental system, it means that the user interested in a rare phenomenon can choose to wait for more samples.

For a tool like sampleAction to work against very large datasets we want the formula that provides the confidence bounds to be scalable. In particular, even for very large databases (that is, where (Figure 3-1) is large), we would not want that size to generate very broad confidence bounds. We also want an estimator in which the confidence bounds narrow monotonically as the sample size increases: as the number of rows that the stream has processed (Figure 3-3) grows, we expect the bounds to tighten.

The computation of appropriate bounds is an active area of research in probability theory, and different bounds are appropriate under different circumstances. In general, though, some estimators gain their strength by using additional information from the dataset beyond the sampled values. For example, it is common to examine the minimum and maximum values in the data column. The pace at which bounds shrink is determined by the size and variance of the sample: the bounds expand with the variance of the sample, and tighten in proportion to the square root of the number of samples. As a result, the choice of estimators, combined with the distribution in the results can produce very different bounds, changing at very different speeds.

In the *sampleAction* protoype, we computed several different sets of bounds in order to learn about their

convergence properties. These bounds are displayed in the captions of Figures 1 and 2, and were selectable by the user. We do not expect that this variety of bounds would be available, or desirable, in a final product; our user study did not emphasize multiple bound types.

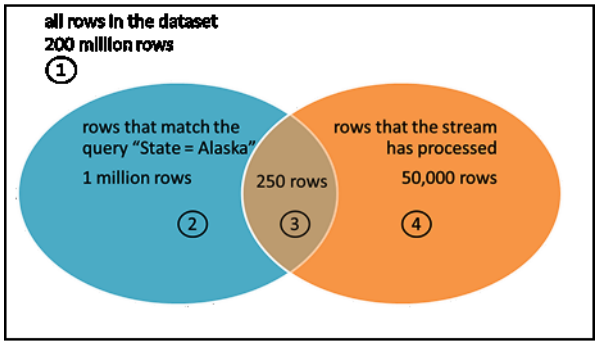

Figure 3. Schematic view of sampling against filters. A restrictive query, joined with a small sample, can make for a very small set of rows to inform estimates.

The Back-End Database

Industrial database management systems do not currently support incremental queries of the type required to test our hypotheses. Therefore, we constrained this initial evaluation to deploying *sampleAction* on a database small enough to query interactively: *sampleAction* takes samples from a database with several millions of records. We used a standard SQL database system to store the data. While these datasets are still relatively small in comparison with "big data" systems, they are nevertheless sufficiently large for our purposes: it is possible to extract samples, run aggregate statistics, and compute probability bounds over them. These smaller databases have the virtue that they are easy to query and return rapid results. Therefore, *sampleAction* is able to interactively run complete queries over the entire dataset, collect metadata, and compare estimates with ground truth results.

sampleAction stores data in its back-end SQL database in a randomized order. Putting the data in random order allows *sampleAction* to perform collection of random samples, merely by querying for the first few thousand rows. To simulate incremental results, we simply executed repeated queries of increasing size. For example, the first initial query requests results based on the first 5000 records; the second query based on the top 10000, and so on. Each of these queries could be resolved quickly and returned to the user. This scheme allows us to prototype the effects of an incremental query against a randomized dataset.

We note that because the queries for different groups are drawn from the same sample, the estimates are not fully independent. For the experiments we conducted, we found the sample sizes were typically large enough for these effects to be negligible.

We note that our sample is far from using SQL to its capacity: we would expect that in a production system, users might see updates of millions of rows at a time.

USER STUDY

To evaluate the effectiveness of our technique, we recruited three sets of experts who analyze data on a regular basis. All three work for a large, data-intensive corporation. We selected three very different groups of analysts, with very different types of data to see how they would respond to incremental visualization in their work. One team runs system operations on a large network, looking for network and server errors. The second team is part of a marketing organization looking at the marketing and use of network-connected games. The third participant is a researcher, studying social behavior on Twitter.

To create a familiar, real-world data experience with *sampleAction*, we collected sample data from each of the expert teams; they provided us with recent selections of their datasets. We asked them to provide us around a million rows of data in order to have a reasonably large dataset. We wished to ensure that the session asked real questions that the analysts might have encountered. Therefore, in preparation for the session, we asked them to recall a recent data exploration session, or to think of the sorts of questions that they frequently ask of their data. Because of the iterative query services facilitated by our interface, we expected our sessions to diverge from their usual queries, but this structure allowed us to start from a familiar place.

After introducing the system to the teams, we had the experts reconstruct the questions that they had encountered in the past. During this series, we asked them to think aloud through the charts they were seeing on screen, and asked them to describe points at which they would be able to make a decision. Periodically, we paused computation to ask them about how they would interpret the interrupted session. While the session with the Twitter researcher took place in person, the other sessions were carried out by remote conversation with a shared desktop session running the application. Voice and screen interactions for all sessions were recorded.

Bob: Server Operations

Bob is on a team that manages operations for a handful of servers. Their group has a logging infrastructure that is periodically uploaded into an SQL server; nightly, the server's results are produced into a static report, generated by Microsoft SQL Reporting Services. Bob's team both monitors the performance of a set of servers, and diagnoses error conditions that may occur. The report is not interactive; as a result, the team has created an interactive custom application that shows some results that the report cannot. However, they complain that the custom application has a very limited set of queries.

Bob was able to provide 200,000 rows in each of two tables: one that was oriented around error conditions; the other around successful interactions. Bob's dataset is fairly uniform: the back-end server behaves reliably, and the range of data is small. Therefore, he was able to get rapid and accurate estimates, and the confidence intervals converged very rapidly.

Bob started off by looking at number of errors divided by datacenter. After seeing the first set of results, he realized that the errors he was investigating were all in one datacenter: "Ok, we'll stop that, and we'll change over to the right variable this time."

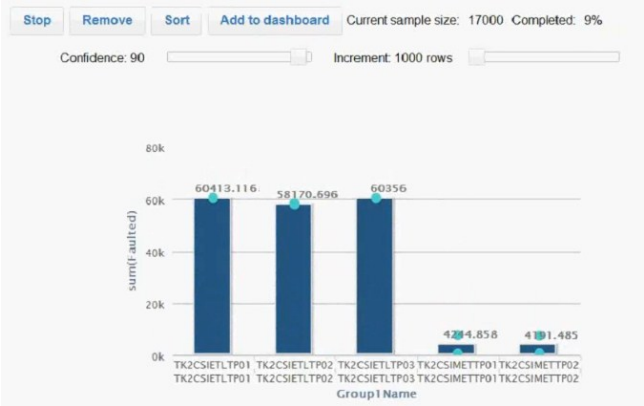

Figure 4. Consistent error behavior across three servers of one type, and two other servers of another.

After changing to a display by server, he let it run for a moment (Figure 4). "What we're not seeing is any particular outliers. What this is telling me is that all the machines are performing about the same. The errors are high, but consistent. The pile of errors we're seeing is a site-wide issue, not a machine issue."

He then wished to drill down into the types of errors. He filtered down to just two servers and added the error type as a measure along the X axis. In these machines, most records were of the same error type. A very small number of rows were of other error types; these other types had few samples, and so displayed very wide confidence intervals.

Bob was interested in incremental visualization as an alternative to their current, index-heavy implementation of data management. As his team has attempted to scale upward, they have spent a great deal of effort optimizing their data, queries, and indices to be able to diagnose errors within a few minutes of their occurring. Finding these rare errors will not be helped by sample-based methods: sampling cannot find outliers.

Bob's team currently archives all data after a day in order to focus on new data—and infrequently carries out the costly queries that would be required to access their archive. He felt that incremental systems might help them explore their archives, understanding how system performance is gradually changing over time.

Allan: Online Game Reporting

Allan is in charge of maintaining the database reporting system for a large online gaming system. The core database records every session by every player logged into the system, as well as their purchasing history. Allan is regularly asked to prepare tremendously varying reports for a variety of stakeholders, ranging from marketing teams as well as game designers. In order to present these reports, Allan often creates an OLAP cube which summarizes relevant answers. Allan, therefore, is accustomed to having to clearly specify queries and is unused to exploring his data.

Allan suggested that we examine player session information. The player session table has the locations of players (on a national level), statistics about the players (such as their age), and which games they played on any given day. Allan provided two billion player records.

Allan had recently run an interesting statistic: the average age of a game player on the system. He began by looking at the sum of ages. After a moment, he realized that he wanted to see the *average* age, and stopped the query in order to correct it and issue a new one. Reassured that the data was showing the same result he had seen before, he terminated the query after a few seconds (looking at just thirty thousand rows) and began to explore new queries.

He looked at average age by country, before deciding that the many categories caused the error values to converge too slowly; cancelled the query, and instead looked at average age by region. For some regions where there are fairly few players, the system found few examples, and so generated very broad confidence intervals for those regions. Other regions, such as the United States and the UK, had very precise error bars due to the high number of players. In the current *sampleAction* implementation, the scale broadened to show the large confidence intervals which swamped the values. Consequently Allan turned off confidence intervals, feeling he now knew which columns he could trust.

He then wanted to see whether sports games have a different distribution then war games. He added a filter to the previous query, specifying only war games, and started it. He changed the filter again, and started another query, specifying only sports games. He scrolled back and forth, comparing the results to each other. A few moments later, he added another query, comparing the numbers of war-gamers to sports players by region.

Allan, accustomed to running reports, had not been able to explore his dataset before; he enjoyed exploring the dataset in ways that had not been accessible to him before.

Sam: Twitter Analytics

Sam is analyzing Twitter data to understand relationships between the use of vocabulary and sentiment. He works with Twitter data that is saved to a high-capacity distributed system. New data constantly streams into its ever-growing archive, which has stored several years' worth of data.

Sam's queries require several sets of keyword filters, which he frequently tunes. Sam provided us with a single day worth of data, with annotations labeling which filters would have affected which tweets. The result was approximately 10 million records.

Sam sometimes uses visualization: "I've generated my own charts in R, but it's based on small samples."

During Sam's interview, he created a series of bar charts, tweaking variables. He frequently made small errors, realizing that he had placed the wrong variable in the query or had failed to filter out 'null' values. In each of those cases, he observed this within the first few iterations, when we had seen less than 0.1% of the full dataset. Using his usual batch tools, he would not have caught this error until after the computation was done, several hours later.

In using *sampleAction*, Sam moved rapidly from query to query, exploring and testing different variations. Once, for example, he wanted to compare the relative frequency of keywords having to deal with emotions. When he generated the column chart, he was able to stop the iteration after 150,000 samples (about 30 seconds) and explore it. By the time he was at that phase, the differences were vivid. For this keyword, at least, the error margins were tight: "I didn't actually know before that 'hate' was so dominant."

He was aware of the limitations of looking at a sample: "the statistician in me is saying, I want to let this run a little longer before I make a total judgment call on these two sets." Nonetheless, the partial result was enough for him to continue to explore.

He decided to figure out why the keyword was so large. To do so, he needed to compare the word list under two different conditions. He created two filters—one for each condition—and started two queries. He compared the two runs to each other: "See how much bigger 'angry' was in the other one? These are hugely different."

Because his X axis had so many different keywords, some of which were rare, the results were distributed across a very large confidence interval. As happened for Allan, this large confidence interval distorted the scale on the rest of the image. He found the distortion to be too large to interpret the chart, and often distracting; he would turn them off to examine the values, then turn them back on to check how confident he could be in any value.

ANALYSIS

In this section, we collect some of the major insights from the three different user studies.

The value of seeing a first record fast

In all three studies, users found value in getting a quick response to their queries. Sam and Allan realized they had entered an incorrect query, and were able to repair it quickly by adding appropriate filters. Ordinarily, discovering and repairing these errors would have been a costly, even overnight process. Allan also realized that his X axis would be wider than he wanted, and changed his query to narrow his results. Bob's data was uniform enough that even the first view had a good confidence interval, and so he was able to draw conclusions from it.

New Behaviors around Data

All three of our analysts were accustomed to seeing their data in a static, non-interactive form: they formulate a query (or cube), wait a period of time, and can explore the results. Most visibly with Sam, the opportunity to interact with the data without waiting was freeing: it changed the *sorts* of queries that he was able to make, as well as the *results* of those queries. Allan was excited to have the opportunity to ask new questions of his dataset without delay.

We observed real exploration of the dataset using our system. Sam was able to play with a hypothesis that he had not previously explored, in part because it required several different permutations of his query in order to find the interesting result. Allan was able to try a handful of different variations, exploring questions in depth. Bob was able to clean his queries on the fly, removing special cases and exploring the types of results returned. None of these were possible in the non-interactive case.

At the same time, the incremental aspects were helpful to the analysts. If the first few samples had not converged, they would decide whether it was worth the trade-off of waiting longer, sometimes checking the convergence view (Figure 2) to decide. In cases where the system seemed unlikely to converge, they would decide which columns of data to regard.

Difficulties with Error Bar Convergence

We did not anticipate the tremendous variance in confidence interval sizes. While Bob never saw a confidence interval much larger than his largest data point, Allan often could not see his data without hiding the confidence intervals. Past literature on visualizing uncertainty [11] has emphasized visualizations that fit the entire uncertainty range on screen; these were not sufficient for some of these preliminary bounds. It would be worthwhile investigating visualizations that can show the size of the interval even past screen borders.

In Allan's sample, some data points had noisy values: for example, the minimum 'age' listed was -100, while the oldest was 284. This threw off the "minimum" and "maximum" values; as the computation we used included these values, the bounds converged slowly. Incremental systems can be slowed by datasets that are not clean. Using additional domain knowledge during the execution—such as discarding values that fall outside meaningful constraints—would improve convergence, and would show more meaningful results.

Non-Expert Views of Confidence Intervals

While error bars are familiar indications of confidence, some of the users found them confusing. It was not initially

obvious to Allan, for example, that the interval would shrink toward a converged value.

The confidence interval is a complex indicator: it carries information about both the number of samples seen so far, and the variance of a column. As a result, two very different adjacent columns might have identical confidence intervals: one has a small variance but is fairly rare in the database; the one next to it is common, but has a high variance. Helping users distinguish these would be useful.

In all three cases, users had data that was unevenly distributed across the X-axis, with some categories having a great many entries, and others having very few. For example, in Allan's situation, countries with few players converged very slowly, causing estimates to be very large.

Sam and Allan were able to adapt to the error bars, regarding the numbers that converged faster as more certain than the ones that took longer.

Implications

Our work shows both that users seem to be able to interpret confidence intervals, and that this finding opens opportunities for using uncertainty visualization tied to probabilistic datasets. Creating *sampleAction* has allowed us to have a concrete feel for the experience of watching bounds shrink at different rates, which in turn is illuminating for visualization design of confidence intervals.

The major step that stands between simulators like *sampleAction* and true interactive techniques are limitations to databases. Currently, Big Data systems do not support the callbacks or partial results that would allow incremental results to be computed. Similarly, SQL tables allow sampling, but do not allow the user to progressively increase the size of their sample. Allowing these is a necessary back-end for future interactions.

Limitations of Incremental Visualization

sampleAction has helped us interpret how users interact with incomplete and incremental data. Even in a complete incremental system, however, there some genres of queries that are structurally going to be difficult. These are not limitations of our prototype, but are fundamental to the approach.

Outlier Values

This system only works for meaningful, aggregate queries. Thus, operations that depend on single items, such as outlier queries, cannot be supported. There is no probabilistic answer to "which item has the highest value". However, there might be ways to *rephrase* queries: for example, it might be possible to use order statistics in this context.

Table Joins

Joins are an important part of database interaction; past database projects like CONTROL [4] and others [6] have looked at the statistical and technical issues involved in incremental joins. As with outliers, some types of joins can be very difficult for incremental sampling techniques; in some cases, such as joins against a rare or unique key, using samples from joining tables may not work at all.

Future Work

The experience of exposing users to incremental queries and approximate visualizations motivates several lines of future work. First, it has highlighted the importance of exploring representations of confidence. While error bars are conventional, they are not necessarily easily comprehensible. In addition, they can only highlight one probability value at a time. The downsides of error bars, such as the difficulties they raise with scaling, argue that there could be an opportunity to find new ways to represent confidence intervals.

Our users also asked for more types of visualizations: clustered bar charts showing more than one measure; a line chart; and two-dimensional histograms. Each of these visualizations will raise new issues in presenting confidence intervals. It is worthwhile to explore new visualizations in order to enable rapid refinement for these more sophisticated query types.

Last, we would like to explore more types of data analysis, such as machine learning techniques. We believe that applying incremental techniques to a broad range of algorithms might help users anticipate their algorithm's progress before it comes out with its final result.

Conclusions

While the concept of approximate queries has been known for some time, the visualization implications have not been explored with users. In particular, it has been an open question whether data analysts would be comfortable interacting with confidence intervals. We hope that showing the utility of these approximations will encourage further research on both the front- and back-ends of these systems.

HCI researchers have also been limited in their ability to explore these concepts; our model for simulating large data systems may help them explore realistic front-ends without needing to build full-scale computation back-ends.

We have shown that it is both tractable and desirable to support incremental query interactions for data analysts. With such mechanisms in place, analysts can take advantage of the immediate feedback afforded by incremental queries by rapidly refining their queries, and more importantly, exploring new avenues which they would not have done before.

Our approach has validated the concept of incremental queries. We have shown that it is possible to use interaction strategies that analysts have desired, but not been able to pursue given the time required to complete queries of large scale databases. As our interviews show, even relatively simple representations of uncertainty using error bars progressively updating over time, allowed analysts to trust

their decision points, potentially saving days or weeks of effort, and exploring unimagined routes through their data for new discoveries and insights.

ACKNOWLEDGEMENTS

We thank our participants for their time and enthusiasm in working with our prototype. This project has benefitted from the expert advice of Christian Konig, who has provided us with valuable references, design, and sanity; and from the comments of the anonymous reviewers.

REFERENCES

1. S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*. 26(1):65-74. March 1997.
2. D. Fisher. Incremental, Approximate Database Queries and Uncertainty for Exploratory Visualization. In *Proceedings of 1st IEEE Symposium on Large-Scale Data Analysis and Visualization*. 2011.
3. P. Haas, J. Hellerstein. Ripple Joins for Online Aggregation. In *Proceedings of ACM SIGMOD International Conference on Management of Data*. 1999.
4. J. Hellerstein, R. Avnur, A. Chou, C. Olston, V. Raman, T. Roth, C. Hidber, P. Haas. Interactive Data Analysis with CONTROL. *IEEE Computer*, 32(8). August, 1999.
5. J. Hellerstein, P. Haas, and H. Wang. Online aggregation. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of data* (SIGMOD '97), J. Peckman, S. Ram, and M. Franklin (Eds.). ACM, New York, NY, USA, 171-182.
6. C. Jermaine, A. Dobra, S. Arumugam, S. Joshi, and A. Pol. The Sort-Merge-Shrink Join. *ACM Transactions on Database Systems* 31(4): 1382-1416. December 2006.
7. S. Joshi and C. Jermaine. Materialized Sample Views for Database Approximation. *IEEE Transactions on Knowledge and Data Engineering*. 20(3): 337-351. March 2008.
8. R. Kosara, S. Miksch, and H. Hauser. Semantic Depth of Field. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*. IEEE Computer Society. 2001.
9. S. Melnik, A. Gubarev, J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis. 2010. Dremel: interactive analysis of web-scale datasets. *Proc. VLDB Endow.* 3, 1-2 (September 2010), 330-339.
10. F. Olken and D. Rotem. 1990. Random sampling from database files: a survey. In *Proc.of the 5th international conference on Statistical and Scientific Database Management (SSDBM'1990)*, Zbigniew Michalewicz (Ed.). Springer-Verlag, London, UK, 92-111. 1990.
11. C. Olston and J. Mackinlay. Visualizing data with bounded uncertainty. In *Proceedings of IEEE Symposium on Information Visualization*, pp. 37-40. 2002.
12. C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: A Not-So-Foreign Language for Data Processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (SIGMOD '08). ACM, New York, NY, USA, 1099-1110.
13. R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. 2005. Interpreting the data: Parallel analysis with Sawzall. *Sci. Program.* 13, 4 (October 2005), 277-298.
14. J. Sanyal, S. Zhang, G. Bhattacharya, P. Amburn, and R. Moorhead. 2009. A User Study to Compare Four Uncertainty Visualization Methods for 1D and 2D Datasets. *IEEE Transactions on Visualization and Computer Graphics* 15(6): 1209-1218. November 2009.
15. M. Skeels, B. Lee, G. Smith, and G. Robertson. Revealing Uncertainty for Information Visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM, New York, NY, USA. 2008, 376-379
16. D. Ślęzak and M. Kowalski. Towards Approximate SQL – Infobright's Approach. In M. Szczuka *et al* (Eds.): Rough Sets and Current Trends in Computing (RSCTC). LNAI 6086, pp. 630-639. Springer-Verlag. 2010.
17. A. Streit, B. Pham, and R. Brown. A Spreadsheet Approach to Facilitate Visualization of Uncertainty in Information. *IEEE Transactions on Visualization and Computer Graphics* 14(1): 61-72. January 2008.
18. M. Stonebraker, D. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, and S. Zdonik. 2005. C-store: a column-oriented DBMS. In *Proceedings of the 31st International Conference on Very Large Data Bases* (VLDB '05). VLDB Endowment 553-564.
19. J. Thomson, E. Hetzler, A. MacEachren, M. Gahegan and M. Pavel, A typology for visualizing uncertainty. In *Proceedings of SPIE & IS&T Conference on Electronic Imaging, Visualization and Data Analysis 2005*, 5669: 146-157. 2005.
20. C. Wittenbrink, A. Pang, and S. Lodha. Glyphs for Visualizing Uncertainty in Vector Fields. *IEEE Transactions on Visualization and Computer Graphics*. 2(3):266-279. September 1996.
21. T. Zuk and S. Carpendale. Visualization of Uncertainty and Reasoning. In *Proceedings of the 8th international symposium on Smart Graphics (SG '07)*. Springer-Verlag, Berlin, Heidelberg. 2007.