Incremental, Approximate Database Queries and Uncertainty for Exploratory Visualization

Danyel Fisher
Microsoft Research

ABSTRACT

Exploratory data visualization calls for iterative analyses, but very large databases are often far too slow to allow interactive exploration. Incremental, approximate database queries exchange precision for speed: by sampling from the full database, the system can resolve queries rapidly. As the sample gets broader, the precision increases at the cost of time. As the precision of the sample value can be estimated, we can represent the range of possible values. This range may be visually represented using uncertainty visualization techniques. This paper outlines the current literature in both incremental approximate queries and in uncertainty visualization. The two fields mesh well: incremental techniques can collect data in interactive time, and uncertainty techniques can show bounded error.

Keywords: Uncertainty visualization, incremental visualization, approximate visualization, very large data, exploratory data analysis.

Index Terms: H.2.4 [Database Management]: Systems—Query Processing; H.5.2 [Information Interfaces] User Interfaces—Graphical User Interfaces.

1 Introduction

We live in an era of ever-faster computing systems, but larger storage and information lead to slower database queries. This is unfortunate, as the field of visual analytics has often focused on the value of exploratory visualization: a process of iterating through queries to learn more about a dataset. Users expect to ask a question of the data, get a response, and then generate a new round of questions. Getting one result from a dataset serves as a cue for the next query.

Very large databases make interactivity much more difficult, as a full query across the entire dataset can be very slow. Several data analysis systems, such as Tableau, Vertica, and Microsoft's PowerPivot, have incorporated high-speed in-memory column-oriented storage in order to scale to interactive queries across millions of rows. Beyond this range, however, there is a more fundamental issue: a database simply cannot produce a full response to a query in interactive time.

In this paper, we take as a primary goal the idea of presenting *useful* results to the user at interactive speeds. We accept that there will always be some queries that are too slow to fully complete as rapidly as a user might wish; we look for ways to reduce the impact of these limitations. Enabling these scenarios will require both new designs for visualization and system design.

danyelf@microsoft.com
1 Microsoft Way, Redmond, Washington. 98052 USA

1.1 Interaction Assists Exploratory Visualization

Several projects have attempted to distinguish between exploratory visualization and presentation (e.g. [22]). In presentations, the audience and presenters expect precision: the visualization should be based on the best data possible.

In contrast, exploratory visualization can be understood as iterating through a series of visualizations rapidly enough to make a decision based on data. In many cases, knowing a correct answer within some percentage may be sufficient for decision-making. Consider a sales manager examining recent sales tables: learning that her branch sold approximately five times as many widgets as sprockets will allow her to dive further into the widgets category without worrying much about sprockets.

We might draw an analogy to a computer-graphics artist looking at a rapidly-generated wireframe, rather than a full render of an animated scene. The wireframe can be generated rapidly, and may give the artist enough information to move props or actors around.

Approximate results can help make decisions about datasets rapidly. In the data cleaning phase, an analyst can discover that some messy items swamp the dataset. In exploration, a quick overview can help an analyst realize they have issued the wrong query—and, once they have the right one, quick overviews can help decide which region of the data is worthwhile focusing on. In some cases, approximate results might allow a time-critical decision to be made faster without having to wait for a long processing job.

Today, analysts will sometimes separate a tractable subset of their database, and work off of that subset, creating data cleaning scripts and preliminary graphs. They then run those same scripts on the full database. The analyst must judge how large a sample of data to collect, and must decide whether the sample is representative. In few cases does the analyst expect to make decisions based on a sample. Incremental processing allows a user to work off of increasingly-large samples, and thus increasingly-certain results.

In general, we wish to visualize data rapidly enough to encourage the user to be able to string tasks together interactively. Card *et al* suggest a timing model which describes how long a user might stay on task [2] (and later work on timing by Seow, [25]). Responses less than 0.1 seconds are perceived as nearly instantaneous; anything less than one second is understood as an prompt response. On the other hand, if a system takes more than ten seconds to reply to a query, the user will perceive the questioning and answering as separate tasks.

1.2 Big Data: Hard for Databases and Visualizations

That sort of timing is an ambitious goal. Any compute job that touches large data—in the terabyte range and up—requires a substantial amount of time. Disks are limited in reading speed; networks are limited in throughput. In distributed networks, such as Hadoop, Amazon EC2, or Microsoft's Azure, there can be individual machines may run slowly; consolidating data from multiple machines may also saturate bandwidth. While increased bandwidth, parallelism, or clever storage schemes may temporarily achieve interactive rates against large datasets, ever-larger datasets keep pace with innovations.

In the visualization community, large data has also been a challenge. Information visualization research was forced to face issues of scale over a decade ago, when datasets grew larger than the number of pixels on screen: a threshold crossed around a million items, or a few megabytes of data. Pioneering work by Shneiderman [26] and others argued that it is critical to summarize data, either by ordering it hierarchically, or allowing users to filter and zoom deeper into the dataset.

These techniques are limited by data constraints: a treemap may be a good overview technique, but requires that each node have an accurate count of the sizes of its children. Depending on the size and storage of a dataset, that can be a very expensive computation. In this paper, we identify classes of visualization techniques that work well when our ability to collect data rapidly is limited.

1.3 Approximate Visualizations of Incremental Data

We can use incremental data sampling to generate approximate answers for many types of queries. While it is possible to generate a visualization based on a sample of a dataset, we wish to allow users to interactively choose to trade *time* for *accuracy*. For appropriately-formed queries, many queries should be able to produce an approximate answer rapidly; as the data request progresses, the user can get a more detailed, and more accurate, response. The user can choose to interrupt at any moment, when they have enough information to *act on the data*, *create a new query*, or *decide to wait* for more detailed data.

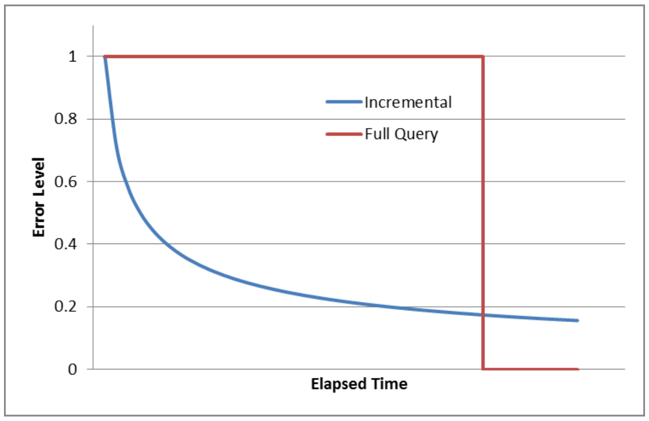

Figure 1. Hypothetical comparison on quality of answers for a system computing incremental results. While the traditional, fixed system generates a *complete* answer sooner, the incremental system has fairly tight bounds much earlier.

Figure 1 illustrates a (hypothetical) accuracy-vs-time trade-off curve. The Y axis represents the degree of accuracy of the query response, expressed as a fraction of the true value. In the incremental case, the system produces useful results almost immediately (solid line); at a later time, the full query (dotted line) returns. At that moment, the full query is more accurate (but only slightly) than the incremental partial results had been.

These techniques apply to data and queries where a sample can be visualized usefully, such as aggregate visualizations. In contrast, visualizations that emphasize individual datapoints will be much less successful.

Two fields of research converge to make this possible. On the one hand, the database community has carried on a tradition of research looking at incremental and online data querying techniques, examining systems and situations where it works well. These techniques have not yet become mainstream. On the other hand, the data visualization community has wrestled with concepts of collecting and visualizing data that is subject to 'uncertainty'; however, the field seems to have had limited success in finding useful examples of uncertain data. This paper unifies these literatures, suggesting interactions between them.

Aspects of this have been mentioned before. The CONTROL project [9][11] is a major inspiration for the area of online, interactive data analysis. The CONTROL project is motivated in part by allowing users to interactively generate visualizations. However, the authors have not presented appropriate visualizations that would go with their interactive database. Olston and Mackinlay [20] mention incremental sampling as a potential motivator for their visualization of uncertainty, and refer back to the CONTROL project, but provide little detail on how those techniques might be carried out in practice. In this paper, we attempt to overview both literatures, in an attempt to describe the costs and benefits of incremental visualizations.

We propose a workflow as in Table 2: the user begins a computation process that iteratively updates a visualization with ever better estimates and confidence bounds.

This paper intends to motivate incremental analysis from both the data side and the visualization side, and argues that the two should be brought together. First, the paper contributes as a review of techniques for handling large data, including pre-aggregation and online querying; it discusses statistical bounds that help describe the range of the data, and discusses the implications for arranging data in disks and networks. Second, it reviews the visualization literature on uncertainty and on aggregation, and links the notion of uncertainty to the probabilistic results produced in the previous section. Last, it introduces a prototype system, "TeraSim," that generates iterative, convergent data for visualization. While TeraSim is still in construction, its design illustrates how a pipeline from data through visualization can be brought together.

Table 2. Workflow for incremental visualization system. SQL syntax is for illustration only.

Incremental Visualization System	
1.	**Large Data is stored on disk or distributed system**

2.	**User selects a dataset within their visualization system; system issues aggregate query**
	`SELECT APPROXIMATE CATEGORY, SUM(*)`
	`FROM Table T`
	`WHERE Condition=C`
	`GROUP BY CATEGORY`

3.	**Database begins to produce histograms of (CATEGORY, SUM, CONFIDENCE BOUNDS).**
4.	**Visualization updates to reflect categories and values from the database.**
5.	**If the user chooses to interrupt, then the visualization stops; otherwise, repeat step 3 with more detail.**
6.	**User issues new query, or waits for computation to complete.**

2 Dealing with Large Data

The notion of "large" data changes with computer technologies. Papers in the VLDB ("Very Large Databases") conference of 1975 referred to tables of millions of entries [27]; current work looks at analyses of web-scale datasets—a handful of orders of magnitude larger. During the intervening 35 years, the use of multiple parallel data stores and networked databases has stayed constant. Separating data onto multiple disks and network means that any query against the data will be costly.

2.1 Pre-Aggregation is Fast, but Inflexible

This problem has been recognized for a very long time. OLAP [3] is a technology that partially pre-aggregates data into 'cubes', stored in a 'data warehouse'. The cubes are chosen to expose major facets of the data, while reducing the number of individual elements. For example, a retailer might store individual sales transactions in a transactional database; the warehouse might then store all transactions aggregated by product, sales date, and retail outlet. This warehouse would be very quick to reply to queries like "how many widgets were sold on Wednesday?" However, it would be unable to address queries that require dimensions not stored in the cube, such as "how many customers from ZIP code 98052 purchased a widget?" While an offline data warehouse is a fast way to handle data, we are particularly concerned that the user be able to formulate queries based on dimensions that were not previously expected.

As an example from the Information Visualization field, the Hotmap [8] project visualized where users downloaded tiles from Microsoft's Bing Maps. While the datasource started with raw HTTP logs, they were pre-processed offline into a fixed data structure to allow for quick retrieval. Hotmap needed to know the number of hits at each latitude and longitude, as a result, location was used as a primary key. While this new format allowed for interactive navigation speeds, the pre-computation phase was very long (eliminating the possibility of looking at daily changes) and allowed only a certain set of queries. For example, the system could address queries about the date of the web hit, but not the IP address of the requestor (a field that was not indexed). This pre-aggregation limited the analyses that could be done with the system without extensive manual labor and slow computation.

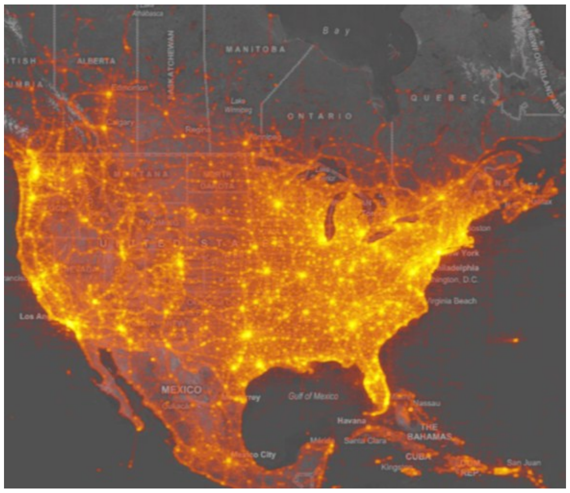

Figure 2. The Hotmap project. Brighter spots have had more people look at them.

Some issues with speed can be alleviated by carefully planning out a sequence of queries. Chan et al [4] show a system which keeps ahead of viewers by adaptively prefetching data near where they currently are. This means that users will always have the illusion of available data. This solution, like other forms of pre-aggregation, works best when the user's options are limited: if the user can choose from a very broad selection of choices, then the amount of data to be prefetched will be similarly large.

2.2 Parallel Computation for Big Data

One way to handle very large datasets is to parallelize storage and computation. Rather than storing all the data on a single machine—limited by the one machine's memory, disk bandwidth, and computation speed—large datasets can be spread across a distributed database. In the last few years, the MapReduce [5] concept has become a popular way to store large data in parallel. Distributed machines each store a portion of the dataset, and a query is run against each distributed machine separately. Each machine runs the query locally ("map"), and then an aggregation site brings together all of these results into a single result set ("reduce"). One major advantage of this scheme is that data size is no longer a problem: each machine handles a constant amount of data.

Several query languages have been written specifically against MapReduce-type clusters. Sawzall [21] is a specialized language for MapReduce, which allows users to precisely express each step as simple syntax. DryadLINQ [34] extends C#'s LINQ ("Language INtegrated Query") syntax to allow users to write queries that have a parallel component; the system then propagates the query across multiple machines in parallel. DryadLINQ also computes a full directed graph of maps and reduces, allowing multiple-step mappings and reductions.

Dremel [18] is also designed as a rapid query language against large databases; while other systems are agnostic to underlying data structures, Dremel requires that the data table be formatted as a particular column-oriented database; while the pre-processing for this column-orientation can be slow, the results allow for fast queries.

These parallel systems make linear queries fast, limited only by the read speed of their system. However, contemporary implementations do not have mechanisms for reporting incremental results: they touch every row before returning complete results. Of course, any of these systems might be modified to allow progressive queries: recently, MapReduce Online [6] has offered online incremental queries against Hadoop architectures; it allows increasingly-accurate results as further time runs.

In general, this paper addresses precisely this issue: what would the cost, and the value, be of adding progressive callbacks to MapReduce-type systems and large data queries? In the next section, we look first at approximate queries against large datasets.

3 APPROXIMATE QUERIES FOR LARGE DATASETS

As we have noted above, serial scans of large datasets look at individual records very rapidly. Rather than waiting until the scan has touched the entire database, it might be possible to return partial results. If that partial result is based on a fair random sample of the dataset, it will have statistical bounds and properties that can be used to approximate the remainder of the dataset. Unfortunately, this is frequently not the case: data often comes ordered on disk. In the next sections, we will discuss the operations that are possible on such a sample, the challenges of getting data into randomized order, and the sorts of statistical statements that we can make about these aggregations.

3.1 Operations on Samples

Working from a sample of data implies that we can only carry out some operations: we can use the limited sample to extrapolate what the rest of the database looks like. We can then predict bounds on the remainder of the dataset, and visualize these bounds.

3.1.1 Using Aggregate Queries

Aggregate queries are of particular interest precisely because they can reasonably be predicted from a limited subset of a database. Computing the sum, count, and average are very common aggregate queries, and can be computed both easily and precisely online. For sum, count, and average queries against a random sample, the size of the confidence interval inversely proportional to the square root of the size of the sample, and proportional to the variance of the dataset. Thus, as the sample grows, the confidence interval decreases, allowing the user to see values converging.

These queries can be applied well to a variety of different types of visualizations. A histogram, for example, can be understood as a count query, separated by categories. Many popular visualizations are one- or two-dimensional histograms: faceted browsers, such as FacetMap [29], are simply histograms of categories across different dimensions. Hotmap, discussed above, is a two-dimensional histogram across a fairly large number of possible buckets. While both of these projects showed precise results, they would have worked well as approximations.

Some queries are harder. Percentile and median queries can be estimated incrementally, but with diminished precision and ill-defined bounds. Operations like finding the largest or smallest values, or top-N lists, requires looking at every point, and so cannot be reasonably approximated with samples.

We can also approximate some queries with others. While finding a particular outlier is difficult, finding values that fall outside a percentile range is entirely plausible.

3.1.2 Appropriate Data Characteristics

Beyond the type of query, the type of data matters, too. It is desirable to have a finite number of categories: many of the techniques discussed here cannot apply to histograms that are wider than memory. In the example looking at corporate sales, a histogram of product by number of people who bought it would be easy to generate. On the other hand, a histogram of street address by number of purchases made from it may well overwhelm the histogram.

As the confidence level is a function of the variance of the data, the distribution of the data columns also matter: the bounds of a highly skewed distribution will converge much more slowly than a more balanced distribution.

3.2 Randomly Selecting from Databases

The question of efficient random sampling from database files is well-known ([19] has a survey of techniques from 1990). It is a slow process to individually read random rows of a database; the database system is optimized for reading continuous chunks of memory. The quality of the random values can be traded for speed, however: taking random pages from a disk table introduces some bias, but is much faster than taking random rows. A second trade-off is whether to sample with or without replacement. Sampling with replacement can be statistically desirable; however, sampling without replacement ensures that the process will eventually cover all rows and eventually converge at the true answer. (For small numbers of rows—a few percent of the database—sampling with and without replacement are nearly identical.)

Random sampling is implemented in major commercial databases: the TABLESAMPLE keyword is now available in DB2 [10], SQL Server, and Oracle. Each of these allows a user to specify a fraction of the result set to return. This random selection can then be joined with other tables; the query could be repeated to get a broader random sample. The TABLESAMPLE function takes a random selection of pages in the database.

It might be possible to get rapid, partial answers by pre-computing a large random sample of the database, especially in cases where a full OLAP cube would still be too big (or too slow) to be practicable. Recent work [14] explores creating and maintaining these random samples.

3.3 Joining Sampled Data

The join operation is critical to sophisticated data analysis. To understand the importance of the 'join' operation, recall the marketing scenario from before. To examine customers who returned for a second visit, or those who bought a t-shirt in the same visit as a pair of pants, they will need to construct a join statement: two different tables will need to be linked together. Join operations are also needed to fill in metadata about columns, navigate graph data structures, and link together different sources of data.

While joining is a fundamental relational database operation, joining can be difficult on both Map-Reduce and in sampled datasets. On parallel Map-Reduce systems, join operations often require substantial network bandwidth and computation to first hash, then join entries together.

Joins cut substantially down on the size of samples, too: a random 10% sample of two datasets to be joined together has just a 1% chance of finding a row that matches between both datasets. A system will need to keep intermediate join results in memory, looking for matches—and data collection will be especially slow if one of the joined attributes is highly filtered.

Much of the research from the CONTROL project tried to establish ways to handle joining rapidly. One CONTROL technique was the Hash Ripple Join [9]. In a typical join, a database looks at all of the index values of one dimension into memory (perhaps in portions), and then matches them against the other dimension. The hash ripple join, in contrast, alternately reads sets of rows from each dimension, incrementally building a larger pool of rows that might match each other. The Ripple Join suffers when it runs out of memory, however; the Scalable Hash Ripple Join [17] addresses issues with both memory usage by falling back to disk storage, and adds a capability to join across a distributed system.

The Sort-Merge-Shrink Join [13] is an alternate strategy that, similarly, works when the set of keys is larger than memory; in addition, the Sort-Merge-Shrink Join offers robust statistical estimates of both progress and running totals. However, it requires the data on disk to be arranged in random order.

These methods might potentially be parallelized or run on a distributed system. This line of research suggests that incremental queries against joined data may be possible, and can be made both fast and memory efficient.

3.4 Convergent Bounds on Values

An estimate is of limited utility if it does not allow us to also understand how good an estimate it is. We can use confidence bounds to control the visualization. Both the Ripple Join [9] and Sort-Merge-Shrink [13] papers provide discussions of their statistical estimators. These estimators are complex, as they need to account for the statistical properties of the JOIN operation from two different, dependent samples.

The notion of a 'central limit theorem' states that the distribution of the mean of a sample of data is normally distributed, with a variance proportional to the variance of the dataset. This means that the mean of the full dataset can be estimated from the mean of the sample. In general, a variety of different central limit bounds can be applied straightforwardly to these sampling problems; different properties of the data may allow tighter bounds, depending on how the underlying data is distributed. We may therefore expect that an approximate, incremental operation should return both a value, and a probability distribution function (such as 95% confidence bounds).

3.5 Sketch Estimators

A different approach to trading speed for accuracy is through database "sketch estimators." Database sketches attempt to rapidly aggregate data by keeping summaries that are smaller than the number of items. Sketches maintain probabilistic aggregations of values, which can be queried rapidly. Ruso and Dobra [23] compares several different sketch estimators, and provides their statistical properties, including evaluating the quality of the estimators that they produce.

4 Visualizing Aggregate and Incomplete Data

We have largely discussed the techniques that might be used to power and drive an interactive aggregate visualization system. In this section, we turn from examining databases to visualization research. There has largely been little work linking approximate database queries with uncertainty visualization, although the two seem like a logical obvious fit. Olston and Mackinlay are a dramatic exception: their work on bounded uncertainty explicitly mentions incremental data updates as one likely scenario [20]. However, their work focuses specifically on visualization techniques, and does not discuss database sampling issues. In this section, we discuss Infovis research on uncertainty, focusing in particular on visualizations that are likely to be appropriate for approximate databases.

4.1 Types of Visualizations

Only certain visualization techniques would be appropriate to incremental visualizations. In this section, we discuss several visualizations that can or do not apply to large data visualizations. In Shneiderman's discussion of "squeezing a billion points into a million pixels" [26], he distinguishes between "one point per pixel" techniques, which will require zooming, and aggregate visualizations. A scatterplot is an example of the former: a scatterplot of a billion points must plot (and overplot) a billion pixels.

4.1.1 Aggregate Visualizations

In contrast, aggregate visualizations count elements. Those same billion points might be drawn as a two-dimensional histogram, showing the number of points that occur within each region. This latter visualization can be imprecise without losing its meaning. A treemap is similarly a sort of histogram, with each cell as a counted group-by operation.

Elmqvist and Fekete [7] suggest a broad spectrum of ideas for aggregate visualizations. While the paper refers to these as "hierarchical" visualization, their technique is to ensure that individual data points are aggregated together. They suggest visualizations that can be aggregated by sum, average, median, and other statistical measures. They suggest using hierarchical clustering as a base techniques to modify well-known visualizations (such as scatter plots, parallel coordinates, and star glyphs) into aggregate visualizations by summarizing parts of graphs. Many of their techniques are appropriate for the aggregate visualizations we discuss here, although the navigation techniques they discuss to zoom in or peel away layers of data would, in all likelihood, trigger new queries.

4.1.2 Tag Clouds

Tag Clouds are a common visualization that does not hinge on precision [32]. As there are only so many possible sizes for text, a small uncertainty range makes little difference to the tags. It may be possible to render the most frequent words in a visualization using the constraints suggested by ManiWordle [15]. (As Kosara shows, however, adding blur to text may render it unreadable, rather than indicating—as one might hope—that the value associated with the word is uncertain [16]. Other types of annotations might help indicate uncertainty, however.)

4.2 Uncertainty Visualization

In addition to portraying the estimated value, it is desirable for the visualization to show the range of possible values or estimates. As noted above, there seems to be a natural match between some forms of 'uncertainty visualization' and the approximate values with ranges that these estimation techniques produce. The term 'uncertainty' has taken on many meanings within the field of information visualization; authors have used it to refer not only to uncertain data values, but to the quality, provenance, and even the structure of data (summarized in [28]). A broad typology of both sources and types of uncertainty in geospatial data [31] provides additional detail. Zuk and Carpendale [35] analyze several uncertainty visualizations using frameworks established by Bertin, Tufte, and Ware, finding that standard rules of visualization were only sparingly used within uncertainty visualizations, compromising reader's abilities to understand the visualization.

4.2.1 Statistical and Quantitative Uncertainty

Of this broad family of uncertainty, we are interested solely in quantitative uncertainty: places where we know a numerical result, possibly within a degree of precision (that is, statistical uncertainty). In [30], the authors produce a model for maintaining uncertainty information in a spreadsheet. They introduce three forms of uncertainty: *estimates* which are known to be inaccurate (but not by how much); *intervals,* in which a value is known to fall into a range; and probabilities, in which a value can be expressed as a probability curve. These allow them to create approximate line graphs, using translucency for probability, dotted lines for estimation, and filled regions for ranges (Figure 3, bottom). They also offer a 3D chart, which uses translucency at each point to represent probability.

Similarly, Olston and Mackinlay [20] render bounded and statistical uncertainty. They use error bars to indicate statistical ranges (as error bars indicate a 'likely region and estimator'); they render bounded (such as 4.5 +/- 3) as "graphical fuzz", greyed regions. Their techniques work well for scatterplots and line charts; they are forced to compromise for stacked-bar charts and pie charts (Figure 3, top-left).

4.2.2 Challenges in Quantitative Visualization of Uncertainty

Several user studies of uncertainty visualization have suggested that there are real challenges to making uncertain data easily readable to users. Kosara [16] argues that while blur might help cue depth, it is difficult to compare different amounts of blur.

Wittenbrink *et al* test several different glyphs that can be used to highlight uncertainty in vector fields [33]. Their research, which examines glyphs that show uncertainty in both angle and length, evaluates tradeoffs not raised by others. For example,

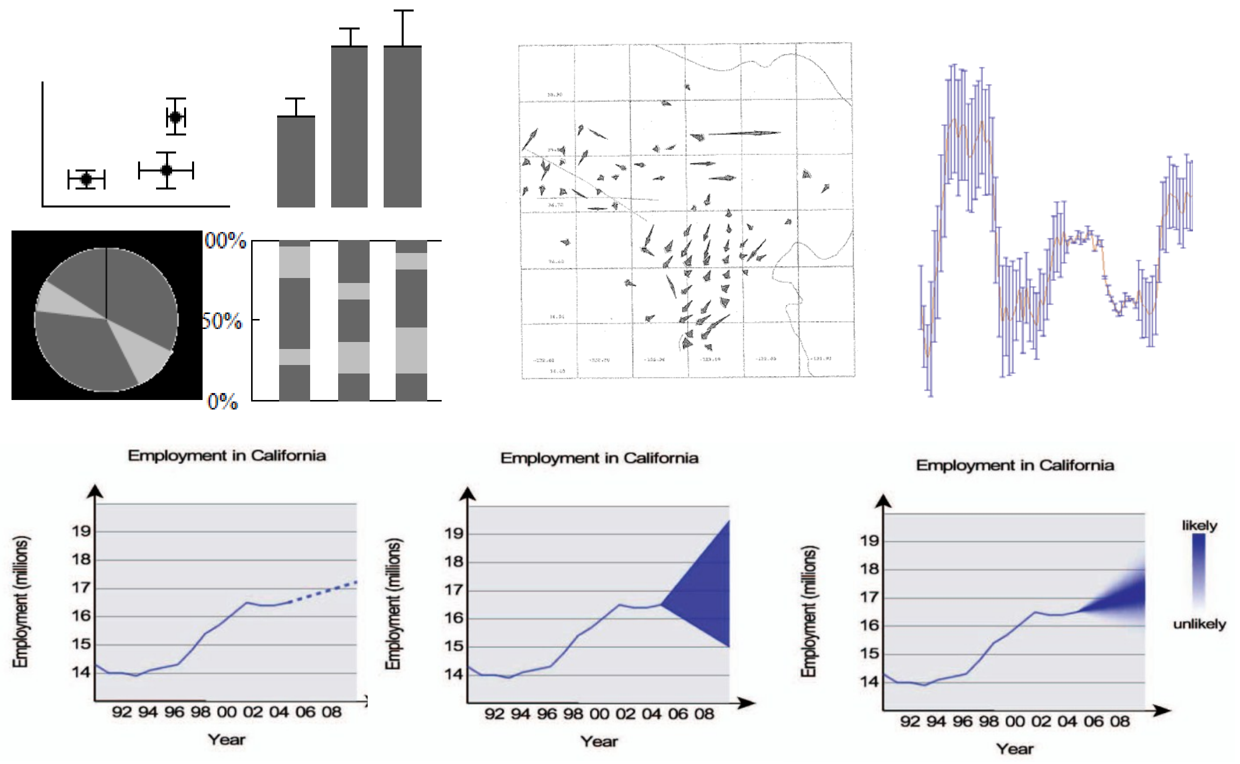

Figure 3. Techniques for visualizing quantitative uncertainty. (Top Left) Error bars for scatter plot and bar chart; ‘ambiguation’ areas for pie chart and stacked bar chart [20]. (Top Middle). Vector glyphs showing uncertainty in both length and angle. The area of the vector arrow shows uncertainty; a certain vector is a thin line [33]. (Top Right) A line chart showing error bars to indicate uncertainty; one of the techniques attempted in [24]. (Bottom) Showing approximate uncertainty, range uncertainty, and a probability distribution function across certainty on a line chart [30].

when a vector has some uncertainty in both magnitude and angle, many natural encodings would be drawn across a greater area, suggesting that the values were *bigger* or *more important*, rather than just more uncertain. Figure 3, top-middle, shows one of their uncertainty-encoded charts.

Sanyal et al [24] also study bounded uncertainty, evaluating different techniques for visualizing uncertain values. They generate uncertain data, and find that error bars provide disappointing results while other mappings (including glyph color and glyph size), while still having high error rates, are somewhat more accurate. Figure 3, top right, has the error bars condition from their study.

4.2.3 Changing Values

As the compute process progresses, it should repeatedly update the visualization with tighter bounds and better estimates. A user should expect that when the process converges, it will deliver a visualization with no uncertainty at all. These constraints require that the uncertain region is drawn in ways that are consistent with the final visualization, so they can be removed. Olston & Mackinlay’s implementation, for example, succeeds at this: their error bars and fuzzy regions would shrink as the computation progresses.

In order to smooth the process of transition, an animation engine (such as DynaVis [12]) could enable smooth transitions between states as the values update. The designer might also want to allow the user to track how the visualization is changing over time: past research has shown [22] that animation can be a poor way to allow users to see differences. For example, the user may wish to track the rate at which regions are converging, or see whether the mean estimate is staying within its estimators. Visualization techniques like showing tracks [22] or allowing old versions of the value to linger on screen (as in Phosphor [1] can allow a user to track changing values.

4.3 Conclusion: Uncertainty Visualization

The techniques discussed above have a number of properties in common: they show ranges of information, mapped as an extra dimension of the visualization: error bars, fuzz zones, and uncertainty-encoded glyphs take up additional space on the screen. In other work, researchers have looked at mapping color to certainty, allowing the visualization to maintain its standard space requirements. For the incremental visualizations we look at here, we may find that the uncertainty is not to be measured per data item, but rather in the visualization as a whole—that is, the whole dataset might be described with a given level of uncertainty, which might open up new alternative visualization types.

5 Implementing a Prototype Simulator System

This paper has so far discussed existing work. In this section, we discuss our system in progress, known as TeraSim, which allows us to simulate a large cloud-based data store. We have chosen to build this as a simulation, rather than as a full cloud-based system, in order to control variables such as latency, amount of data that can be read at once, and even network topology. TeraSim simulates a large-scale back-end server which stores a large set of data, and a front-end computation hub. The back-end server can be configured with the amount of data it controls, its disk read speed (expressed as number of rows per second), and its communication latency.

We are using TeraSim to explore a variety of different visualization techniques over this data. Our goals are to explore possible visualizations of confidence levels and approximate results; to better understand the database constraints needed to support them; and to explore how users interact with changing data.

In reality, TeraSim is implemented as a single application running over a single commodity database. The database contains pre-sampled data, with a few millions of rows. The results that it provides can be scaled up (simply by repeating values) to simulate billions or more of rows. The front-end can issue queries to the database, which returns appropriate statistical properties: usually means, counts, and standard deviations. The front-end then combines these counts to generate its estimates and error bounds, and presents them to the user. The front-end iterates its queries to the back end, getting back increasingly-large estimates.

TeraSim is designed to handle multi-dimensional histogram queries of the form

```
SELECT <G1, G2>, MEAN/SUM/COUNT (*)
FROM <TABLE>
WHERE <Condition>
GROUP BY <G1, G2>
```

where the condition is limited only on columns that are in the table and the group can take one or many columns. TeraSim cannot, however, handle join operations.

We are in the process of extending the back-end so that the system can simulate a cloud of computers, rather than a single machine. At that point, each back-end machine will pass back not its own confidence bounds, but rather components that will allow the front-end to compute the confidence bounds. For example, the SUM estimator is based on the sum of the entire back-end sample, multiplied out to match the size of the full dataset. Each back-end replies with the sum of the back-end sample and their fraction of the size; the system combines these across all of the machines to generate a single estimator. Similarly, the error bounds are computed based on the standard deviation of the sample; each simulated machine will compute a partial result, and the front-end will combine them.

5.1 Passing Incremental Data

In our current implementation, TeraSim passes entire histograms at a time. Each server computes its histogram and passes it to the front-end; the front-end, in turn, replaces its current histogram and re-renders. As we experiment with more (simulated) back-end servers and more data, we are beginning to encounter times when passing the entire histogram for a dataset gets large (such as when we group on vocabulary within a document.) We are designing a system for with passing incremental histograms: tables that represent only the difference from the previous data.

While TeraSim is in early stages yet, we are finding it a robust platform to explore several different types of data, and to motivate further research on building incremental and scalable back-ends.

6 Conclusion

In this paper, we have contributed four interrelated ideas:

First, we have argued for linking incremental data querying techniques to visualizations. We have shown applications and uses where rapid, less-accurate results are more valuable than slow, more-accurate results.

Second, we have highlighted the major issues in the data sampling literature, including the difficulty of getting a good random sample, and the challenges involved in join queries.

Third, we have discussed some visualization tools that help indicate how the visualization field could connect the uncertainty-laden data produced by these approximate queries.

Last, we have discussed TeraSim, a testbed for visualizing real-time results from dynamic queries across large data.

Acknowledgements

My thanks to the many collaborators who have provided critical feedback on this work, including Steven Drucker, John Platt, Christian Konig, and the anonymous reviewers. Research on this project was motivated by probing questions from Sean Boon and Alex Gorev. Shankar Narayanan implemented the first steps toward TeraSim.

References

[1] P. Baudisch, D. Tan, M. Collomb, D. Robbins, K. Hinckley, M. Agrawala, S. Zhao, and G. Ramos. Phosphor: explaining transitions in the user interface using afterglow effects. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06).* ACM, New York, NY, USA, 169-178. 2006.

[2] S. Card, G. Robertson, and J. Mackinlay. The Information Visualizer, an Information Workspace. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology (CHI '91)*. ACM, New York, NY, USA, 181-186. 1991.

[3] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*. 26(1):65-74. March 1997.

[4] S.-M. Chan, L. Xiao, J. Gerth, P. Hanarhan. Maintaining interactivity while exploring massive time series. In *Proceedings of IEEE VAST 2008*: 59-66. 2008.

[5] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber. Bigtable: a distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)*, Vol. 7. USENIX Association, Berkeley, CA, USA, 15-15. 2006.

[6] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears. MapReduce online. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation (NSDI'10)*. USENIX Association, Berkeley, CA, USA, 21-21. 2010.

[7] N. Elmqvist and J.-D. Fekete. Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines. *IEEE Transactions on Visualization and Computer Graphics* 16(3):439-454. May 2010.

[8] D. Fisher. Hotmap: Looking at Geographic Attention. *IEEE Transactions on Visualization and Computer Graphics* 13(6): 1184-1191. November 2007.

[9] P. Haas, J. Hellerstein. Ripple Joins for Online Aggregation. In *Proceedings of ACM SIGMOD International Conference on Management of Data*. 1999.

[10] P. Haas. The Need for Speed: Speeding Up DB2 Using Sampling. *IDUG Solutions Journal*, 10:32–34. 2003.

[11] 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.

[12] J. Heer and G. Robertson. Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics* 13(6): 1240-1247. November 2007.

[13] 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.

[14] S. Joshi and C. Jermaine. Materialized Sample Views for Database Approximation. *IEEE Transactions on Knowledge and Data Engineering*. 20(3): 337-351. March 2008.

[15] K. Koh, B. Lee, B. Kim, and J. Seo. 2010. ManiWordle: Providing Flexible Control over Wordle. *IEEE Transactions on Visualization and Computer Graphics* 16(6): 1190-1197. November 2010.

[16] 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.

[17] G. Luo, C. Ellmann, P. Haas, and J. Naughton. A Scalable Hash Ripple Join Algorithm. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD '02)*. ACM, New York, NY, USA. 2002.

[18] S. Melnik, A. Gubarev, J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis. Dremel: interactive analysis of web-scale datasets. In *Proceedings of the VLDB Endowment*. 3(1-2):330-339. September 2010.

[19] F. Olken and D. Rotem. 1990. Random sampling from database files: a survey. In *Proceedings of the 5th international conference on Statistical and Scientific Database Management (SSDBM'1990)*, Zbigniew Michalewicz (Ed.). Springer-Verlag, London, UK, 92-111. 1990.

[20] C. Olston and Mackinlay, J. Visualizing data with bounded uncertainty. In *Proceedings of IEEE Symposium on Information Visualization*, pp. 37-40. 2002

[21] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the data: Parallel analysis with Sawzall. Scientific Programming Journal. 13(4): 277-298. October 2005.

[22] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko, Effectiveness of Animation in Trend Visualization. *IEEE Transactions on Visualization and Computer Graphics*. 14(6):1325-1332. November 2008

[23] F. Rusu and A. Dobra. Statistical analysis of sketch estimators. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*. ACM, New York, NY, USA, 187-198. 2007.

[24] 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.

[25] S. Seow. *Designing and Engineering Time: the Psychology of Timer Perception in Software*. Boston:Pearson Education. 2008

[26] B. Shneiderman: Extreme visualization: squeezing a billion records into a million pixels. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*. 3-12. 2007.

[27] W. Simonson and W. Alsbrooks. A DBMS for the U. S. Bureau of the Census. In *Proceedings of the 1st International Conference on Very Large Data Bases (VLDB '75)*. ACM, New York, NY, USA, 496-498. 1975.

[28] 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

[29] G. Smith, M. Czerwinski, B. Meyers, D. Robbins, G. Robertson, and D. Tan. FacetMap: A Scalable Search and Browse Visualization. *IEEE Transactions on Visualization and Computer Graphics* 12(5): 797-804. September 2006.

[30] 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.

[31] 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.

[32] F. Viegas, M. Wattenberg, and J. Feinberg. Participatory Visualization with Wordle. *IEEE Transactions on Visualization and Computer Graphics* 15(6):1137-1144. November 2009.

[33] 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.

[34] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. Kumar Gunda, and J. Currey. DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation (OSDI'08)*. USENIX Association, Berkeley, CA, USA, 1-14. 2008.

[35] 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.