Big Data Exploration Requires Collaboration Between Visualization and Data Infrastructures

Danyel Fisher
Microsoft Research
1 Microsoft Way
Redmond, WA 98052 USA
danyelf@microsoft.com

ABSTRACT

As datasets grow to tera- and petabyte sizes, exploratory data visualization becomes very difficult: a screen is limited to a few million pixels, and main memory to a few tens of millions of data points. Yet these very large scale analyses are of tremendous interest to industry and academia. This paper discusses some of the major challenges involved in data analytics at scale, including issues of computation, communication, and rendering. It identifies techniques for handling large scale data, grouped into "look at less of it," and "look at it faster." Using these techniques involves a number of difficult design tradeoffs for both the ways that data can be represented, and the ways that users can interact with the visualizations.

CCS Concepts

• Human-centered-computing→ Visualization theory, concepts and paradigms. •Information Systems → Database management system engines

Keywords

Data visualization; big data; data analysis

1. INTRODUCTION

We live in an era of ever-growing datasets, when an analyst may be called on to process and interact with terabytes, or even petabytes, of data. Yet with these large datasets, the analyst faces not just more data than pixels, but more data than fits in memory, or on a single disk.

Free-form exploratory data visualization has long been a backbone of data analysis: data analysts expect to be able to visualize their data in many different configurations, asking a variety of questions of it, in order to make decisions. Exploratory data visualization requires the ability to manipulate data easily. Products like Tableau and Spotfire, and libraries like GGPLOT and MatPlotLib, have encouraged data analysts to explore their data freely, interacting rapidly with multiple dimensions to try to clean noise, pinpoint interesting dimensions, and explain phenomena. This process, however, depends on easy-to-construct queries and rapid responses.

Unfortunately, data at large scale is not characterized by either of these. Manipulating large data is not yet a straightforward process. In this paper, we identify several different bottlenecks which make data interaction difficult: querying and communicating the data is slow; memory limits constrain the amount of manipulation that can happen; and we can only show on screen a fraction of the data that we process.

The notion of "large data"—and the need for visualization techniques to account for it—is changing over time. The change comes from a growing mismatch between computation and perception. Unlike the Moore's law growth that drives computation and storage, the human eye does not have an ability to take in exponentially-increasing amounts of information. Carr *et al* [3] express concern over both the rendering time and the overplotting issues of 50,000 points: while the rendering question seems laughable now, the overplotting issue is still real. Fekete and Plaisant rendered an impressive million points [15]; now that, too, seems small. Recent projects have processed and rendered a billion points interactively [29].

New techniques are emerging that can inform big data exploration. In the database community, research has looked at a variety of strategies to handle storing, querying and analyzing large datasets; in the scientific visualization community, research has focused on rapid, high quality rendering, and on controlling compute and network costs [1]. The information visualization community is beginning to embrace a number of promising techniques adapted from both areas.

Historically, the database and visualization communities worked apart from each other: database researchers would choose query loads to optimize; visualizers would try to build interactive systems atop them. Sometimes, knowledge of how those visualizations would be built would inform new query specifications. Some database systems were built explicitly with user interaction in mind; however, this is comparatively rare. This might not have been as much an issue in an era of smaller datasets: when a dataset largely fit in memory, and a query could be completed interactively. However, with growing data sizes, the problem becomes more challenging.

In this position paper, we outline a research agenda for exploratory Big Data visualization and analytics. We find these three major classes of challenges to interactivity—rendering, communication, and querying—and outline solution spaces that can address these issues. We argue that all three of these can be addressed by building database systems that are intimately aware of the visualizations they will inform; and visualizations that are informed about the underlying architecture of their back-ends.

This paper presents as an outline of challenges and opportunities for exploration of big data. It does not attempt to present canonical answers nor original research. Rather, by identifying issues common across multiple systems and contexts, it

HILDA'16, June 26 2016, San Francisco, CA, USA

ACM 978-1-4503-4207-0/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2939502.2939518

provides a design space to guide future system and visualization design that accounts for big data.

2. BIG DATA EXPLORATION

This paper is interested in interactive, exploratory [17][24][33] visualization of large-scale data. Handling large-scale data at interactive speeds has been proposed as a 'grand challenge' for visual analytics; clearly, interactive analysis is a growing and important area [36]. There are other forms of visualization that do not share all of these issues: in visualization for presentations, for example, the system may pre-cache the visualizations; in a dashboard, the refresh rate may be reduced instead. Both presentation and dashboard scenarios may have some aspects in common with these goals.

A system that accommodates exploratory analysis must fulfil several different goals. First, it must create interpretable, meaningful views that make sense within the context of the data. Second, it must flexibly allow the analyst to ask many different types of questions about a dataset, including ones that they did not initially anticipate. Last, it must work at interactive speeds, so that the analyst can iterate through different questions and pursue paths through the data [31].

The requirement of flexibility is worth noting as a potential concern. A user working in a command line interface, with data in R or Python, expects have very extremely flexible access to data. On the other hand, in many visualization systems, a database administrator chooses a set of dimensions across which they expect to ask questions. Indeed, to get the "overview" suggested by Shneiderman's mantra ("Overview; zoom and filter; details on demand") [30], some initial choices must be made to decide what dimensions will be visible in the overview; other dimensions may not even be available for analysis.

3. SOURCES OF COST

We sidestep the question of where "large" begins. Our working definition for this paper is "more data than is convenient to work with." (If necessary, add a zero or two at the end to the scale of the current state of the art.) At the least, that means more data than fits in memory, in its raw form, and so a few terabytes is a reasonable starting point. At that scale, it is expensive and difficult to look at all the data. Dealing with this sort of big data makes life harder. It has been common to discuss the issues of rendering speed as well as overplotting and clutter [13].

We consider three different forms of cost as we discuss different techniques for exploring and visualizing big data sets:

- The front end is limited: it runs into limitations on the number of points it can render; or it cannot visualize all the points without substantial overplotting.
- The query takes a long time to return
- The query returns a lot of data, which needs to be communicated to the front-end

3.1 Rendering Challenges

The first round of challenges in visualization was encountered in the question of rendering horsepower. There are a series of interrelated challenges in rendering: the

Rendering speed: Even with the assistance of GPUs and optimized data streams, rendering will be slow if the system is forced to render each and every data point of a dataset. This has become more visible in recent years, with the rise of web-based visualizations; as D3.js [3] rendered to an SVG surface has become a de-facto standard for creating and publishing visualizations, it has become increasingly visible that many browsers cannot scale to more than a few tens of thousands of shapes in an SVG.

Overplotting and clutter: Overplotting is the difficulty of having so many datapoints that they are drawn atop each other. Clutter is

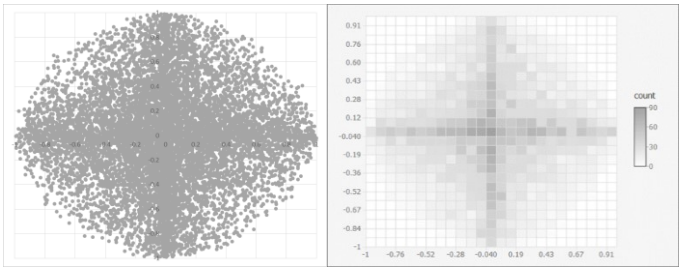

Figure 1: 10,000 points plotted as a scatterplot (left, showing overplotting) and a heatmap (right)

the related problem that it is very difficult to recognize the locations of datapoints when overplotting begins [6]. Figure 2 (left) illustrates the hazards of overplotting: there are enough data points that it is extremely difficult to understand the behavior of the underlying data. While it is clear that there are four branches, it is far less clear how wide the branches are, and whether points at the middle form a tighter inner ring. Nor can we solve this simply by adding more pixels. Both screen sizes and perceptual constraints limit the sheer number of individual data points that we can render: screens remain at a few million pixels--even very large or very high resolution displays [37] add merely an order of magnitude to size, while reducing the ability of users to read it all. There will need to be some form of data reduction to get a large dataset on screen without overplotting and clutter.

A different approach to large data analysis is presented by Keim *et al [27]*. Keim's techniques for mining large data are largely pixel-oriented, painting one pixel or glyph per data point. These therefore require that the dataset has been reduced to under a few million points. While that is certainly a stage that some computations may reach, it seems overly restrictive in an era of petabyte inputs.

3.2 Computation and Communication Time

In a big data environment, a second set of challenges comes in communication time for datasets. Data processing and communication also take more time, detracting from interactivity. Indeed, disk reading time and database processing time dominate rendering as the major form of delay. The most naïve visualization technique extracts data from a database and forwards it directly to the rendering system, which is responsible for choosing precisely how to render this information. In a contemporary client/server system, this would mean passing terabytes over a standard computer network to be processed by JavaScript—an intractable choice, for the disk, network and client.

Equally problematic is the cost of computation and disk speed. While some algorithms can be carried out rapidly, many clustering and data cleaning techniques can be extremely costly. Of course, as Ellsis and Dix note [13] algorithms that execute in $O(n^2)$ and even $O(n\log(n))$ time counts as "extremely costly" in the context of a very big dataset. Even something as simple as taking a second linear pass through a terabyte dataset might turn out to be too expensive, for sufficiently-large datasets.

4. ADDRESSING THE CHALLENGES

A series of approaches are building in common across these challenges: first, taking on the problem of rendering visualization as an **aggregate** challenge. Aggregate visualizations reduce both the number of pixels to be visualized and the amount of data to be transferred. Second, a series of techniques in **reducing the computation** to be done—by pre-computing, by running queries in parallel, or by running progressive queries—can help accelerate the time to get a response. These techniques, however, come with

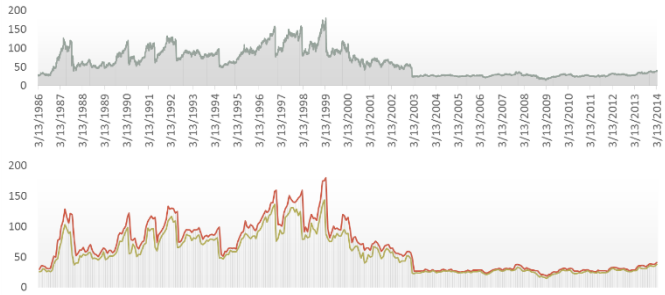

Figure 2: Bucketing over time. Chart (above) compressed by the charting program from 7000 values to 600 pixels; (below) keeping minimum and maximum values within each of 225 buckets.

tradeoffs to the user experience, which need to be carefully considered.

It is worth examining how and when data gets processed as it moves from database, to server, to rendering client; and that we consider visual representations that will allow rapid computation, efficient transfer, and interactive rendering without occlusion. As Elmqvist and Fekete point out [14], aggregate visualizations can resolve both rendering and occlusion problems by limiting the number of items that must be shown on screen.

4.1 Bucketed Aggregate Visualizations

These challenges suggest that we will need to think carefully about how to process large data sets.

Wickham's Bin-Summarise-Smooth framework [35] follows this idea of bucketed aggregations. Wickham provides an elegant way to conceptualize these challenges: the visualization designer chooses a binning function that makes sense for the granularity of changes in the dataset's independent variables; a summarization function that highlights the data relative to the task; and possibly a smoothing function to help see pattern and shape.

The 'summarize' function has many options, depending on user task: while "average value" or "number of values" are common choices, "maximum value" or "does it have more than one item" or even "how many modes are there" are also valid choices to bring out.

Carr *et al* [6] note that for large amounts of data, a bucketed **heatmap** can reduce both rendering time and overplotting, compared to a scatterplot. In Figure 2, we compare two different representations of 10,000 points. The left side scatterplot exhibits overplotting; while it quite clear that some areas are much more crowded than others, a number of questions are left unanswered. Figure 2, right, shows a heatmap of the same data; it makes the shape quite clear. While Carr refers mainly of pairs of quantitative dimensions, we should note that categorical dimensions can be placed into the same sorts of axes. The same work also suggests using hexagonal bins, which—at the cost of some programming and computational complexity—may present a more realistic approximation of density information.

A classic **line chart** can be transformed into an aggregate form by choosing an aggregation function over each time period. In Figure 3, we see a monthly line chart, heavily reduced from single-day data. The aggregation function keeps both the minimum and maximum values at each bucket. This allows the reader to track how much variation occurred within each month. The figure (above) also shows a version of the chart generated by Microsoft Excel. Excel has its own reduction for times when there are more data points than pixels: in this case, we have 7000 data points, but only 600 pixels. Excel chooses a single value. The Excel version above fails to capture the range within some of the more dramatic spikes (although it does choose a slightly higher resolution).

Jugel *et al* [22] take advantage of this concept to speed query time while maintaining the same quality of line chart that would be created by visualizing high resolution data on a low-resolution screen. M4 creates specialized queries in standard SQL that bucket the data at pixel levels; by returning aggregated values, M4 is able to massively reduce the data used for a visualization. The authors then extend this work to other visualization types with VDDA [23]. One can imagine further expanding on this work by enhancing the visualization with additional information about the range and diversity of the data within the aggregation bin.

Many visualization types work well with bucketed aggregates. A ThemeRiver [19] or StreamGraph [5] counts the number of items that occurred per category per time period; it would be entirely reasonable to bucket along the time dimension and category. While traditional network diagrams are poor choices for aggregates, where each node usually represented by a single glyph, Wattenberg's PivotGraph [34] is already a bucketed aggregate. (In fact, there are two sets of buckets: the counts associated with each aggregate node; and the counts associated with each aggregate relation).

4.1.1 Bucketed Aggregate Visualizations

This scheme has limitations, though. The greatest is a constraint on interaction. In classic brushing and linking, the system tracks which elements have been selected, and then renders them across multiple visualizations. This requires, however, access to all of the items in the dataset. There are two possible implementations, as noted by Liu *et al* [29]. First, the buckets can store as many dimensions as needed for every visualization on screen simultaneously. Each visualization then merges buckets as needed to find the elements that go other. The cost to this is an exponential explosion in the number of buckets that must be collected, stored, and communicated. At one extreme, the choice to collect all of these dimensions in their partial summarization reduces back into storing an OLAP cube.

Alternately, brushing and linking can be transformed into a call back into the data source; the bounds of the bucket becomes a query against the data source. The new aggregates that are returned from this query are used for the linked records. Of course, this does us little good if queries are slow and expensive.

5. Look at Less Data; or Look at Data Faster

The bucketed aggregate and data flow allow us to articulate strategies for handling large-scale data visualization. Solutions in this space will happen at all different stages of the aggregate pipeline; solutions that come earlier in the pipeline will have better flexibility for exploration.

In part, here, we are limited by physics. A single processor, looking at one (or several) hard disks, can only access data so fast. A few SSDs filled with data can more than saturate the processor, and it will take a matter of minutes—or hours—to look at more than a few gigabytes. To get interactive speeds, there are several possible strategies, which can be selected in various combinations.

We group these strategies into two general goals: "look at less data," which emphasizes techniques for minimizing the amount of data processed, and "look at data faster," which emphasizes techniques for maximizing the amount of data that can be seen. (Table 1, below, lists the techniques).

The **"Look at Less Data"** techniques involve either aggressively filtering data—so that we can examine fewer rows, or columns; or pre-aggregating data—so that queries can look at fewer, already-aggregated rows. Recently, this strategy has been updated for visualization-specific applications. ImMens [29] observes that visualizations tend to use only three or four dimensions of any dataset; rather than creating large cubes of all possible dimensions, ImMens instead creates multiple cubes representing just a few dimensions. These smaller cubes can be

Table 1: Summary of performance techniques

	Technique	Probabalistic?	Advantages	Costs
Look at less data	Sample Incrementally	Y	Maintains flexibility	Doesn't see whole dataset. Requires randomized data
	Sample Stratified	Y	Good flexibility, fast and bounded cost	Requires stratified pre-sample and sample queries
	Reduce Dimensions	N	Reduces complexity of buckets	Reduces flexibility
	Filter rows	N	Reduces rows to be bucketed	
	Index data	N	Increases query speed	Comparatively small effect; must be precomputed
	Pre-aggregate	N	Very fast queries	Reduces flexibility greatly
Look at data faster	Column-oriented database	N	Increase query speed	Must be pre-computed
	One-pass techniques	N	Reduce memory impact	Some queries cannot be formulated
	Database sketches	Y	Reduce memory impact	Imprecise results
	Distributed computation	N	Parallelize computation	Requires amenable problems

transferred to the client in a compressed binary format, and so allows users to rapidly interact with the data dimensions. Nanocubes [28] uses a similar strategy; it heavily compresses the results of OLAP operations to create an aggregate data structure that can very efficiently store large numbers of datapoints in memory for fast recall.

One interesting technique is incremental, random sampling. In exploratory scenarios, analyst may ask questions that aren't well-represented in the sample—or may have chosen a sample that matches the needs of the data poorly. One possible response to this is to grow samples incrementally [17][18][20][21]. The user can choose how to trade off time for accuracy, watching as the sample refines further, and cut off computation when they have enough detail to proceed. AQP (Approximate Query Processing) [7] and BlinkDB [2] extend this process: they prepare a series of weighted samples in advance and so natively produce probabilistic responses. BlinkDB is built over a distributed network; rather than selecting the precise number of rows, users may select either the amount of computation time or the desired precision as part of the query. More recent systems are beginning to explore samples with appropriate stratification to bound errors and give users faster, less-expensive responses: STORM [9] provides interactive computation across spatio-temporal data; Ding *et al* present an AQP system with bounded error guarantees [12]. Quickr [25] creates distributed samples to reduce computation costs across parallelized queries; while it does not reach interactive response rates, its techniques amortize query cost impressively.

Of course, these techniques now require their own subtleties for visualization: how does one deal with a visualization where the x axis might be continually changing? How does one compute a histogram when the bounds might shift? These questions call for new visualization techniques.

The "Look at Data Faster" techniques, in contrast, focus on reducing the number of passes through the data. Database sketches are powerful techniques for estimating the number of values or approximate estimates in a large dataset; the visualization field has not yet caught up in presenting visual techniques that are good to show these sorts of approximations.

One way to overcome limitations on computation and bandwidth is to distribute a job across multiple machines; this is a particularly popular solution with the recent rise of MapReduce [10], as well as the continuing popularity of parallel processors, ranging from networked machines to GPUs.

A major advantage of breaking data into smaller, tractable parts is that a second pass across data becomes tractable and affordable: as we noted above, quantitative histograms may require two passes across a dataset. As Chauduri *et al* note, a parallel computer can afford that second pass to collect bounds; a central reduce operation coalesces the bounds together and chooses buckets; and then individual machines bucket the data [8].

For these sorts of 'embarrassingly parallel' operations—ones where jobs can be easily segmented into individual units, and those parts can straightforwardly put back together—structures like MapReduce are very effective. VisReduce [21] extends MapReduce to rapidly bin and bucket data that will be seen only in aggregate. Because the bucketed representation of the data is so much smaller than the original source, it can place reduced pieces back together very late in the process. Budiu [4] *et al* extend this concept by (like M4) taking advantage of screen resolution to choose how much data to send.

5.1 New Challenges in Visualization

A number of the topics we have discussed in this paper raise new issues for visualization researchers. We have already noted the phenomenon of categorical data that gains too many categories, a scenario, which can happen in streaming data or incremental samples; and the challenges of adding brushing and linking to otherwise-simple visualizations.

Kandel *et al* [25] note that there is value to a color scale that visibly separates "no data" from "at least one item." In almost any continuous color scale, one item will fall close to empty; however, in a big data context, there is a substantial difference between the two: with millions of data items, a place where *nothing* has ever landed is probably out of the domain of useful points; a place where *anything* has landed is a clear indicator of an area of at least potential interest. InMems recommends placing the "one or more" items at the 15% level on the color scale. A different way to reach the same point is to use the histogram-dependent color scales of Thompson *et al* [32].

One theme that has come up several different times is probabilistic data. If a system has computation based on samples, if it utilizes database sketches or adaptive histograms, the results will be probabilistic rather than definitive. There is an opportunity here to look carefully at techniques for representing uncertain data across visualization to convey these sorts of uncertainty: times when the dataset has an answer embedded in it, but it is impractical to find. Merely showing confidence intervals is insufficient; it is remarkably hard for users to interpret intervals for many critical tasks [10]. Task-oriented overlays might help alleviate some of these difficulties [16], but there are many opportunities for improving these techniques.

6. REFERENCES

[1] L. Battle, M. Stonebraker, R. Chang. Dynamic reduction of query result sets for interactive visualization, *2013 IEEE International Conference on Big Data* , vol., no., pp.1,8, 6-9 Oct. 2013

[2] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: Queries with bounded errors and bounded response times on very large data. In Proceedings of EuroSys. ACM, 2013.

[3] M. Bostock, V. Ogievetsky, J. Heer. D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011

[4] M. Budiu, R. Isaacs, D. Murray, G. Plotkin, P. Barham, S. Al-Kiswany, Y. Boshmaf, Q. Luo, A. Andoni. Interacting with Large Distributed Datasets Using Sketch. University of Wisconsin CS Technical Report TR1817. http://digital.library.wisc.edu/1793/70467

[5] L. Byron and M. Wattenberg. Stacked Graphs – Geometry & Aesthetics. *IEEE Trans. on Visualization and Comp. Graphics* 14, 6 (Nov 2008), 1245-1252. DOI=10.1109/TVCG.2008.166

[6] D. B. Carr, R. J. Littlefield, W. L. Nicholson, and J. S. Littlefield. Scatterplot matrix techniques for large n. *Journal of the American Statistical Association*, 82(398), 1987.

[7] A. Chaudhuri, T. Lee, B. Zhou, C. Wang, T. Xu, H. W. Shen, and Y. J. Chiang. Scalable computation of distributions from large scale data sets. In *2012 IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, (pp. 113-120). IEEE.

[8] S. Chaudhuri, G. Das, and V. Narasayya. A Robust, Optimization-Based Approach for Approximate Answering of Aggregate Queries. *ACM SIGMOD 2001*

[9] R. Christensen, L. Wang, F. Li, K. Yi, J. Tang, N. Villa. STORM: Spatio-Temporal Online Reasoning and Management of Large Spatio-Temporal Data. *SIGMOD 2015*. pages 1111-1116

[10] G. Cumming. *Understanding the new statistics: Effect sizes, confidence intervals, and meta-analysis*. New York, Routledge, 2012.

[11] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In Proc. of the 6th OSDI (Dec. 2004), pp. 137-150.

[12] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang. Sample + Seek: Approximating Aggregates with Distribution Precision Guarantee. *SIGMOD 2016*

[13] G. Ellis and A. Dix. A Taxonomy of Clutter Reduction for Information Visualisation. *IEEE Trans. on Visualization and Comp. Graphics* 13, 6 (November 2007), 1216-1223. DOI=10.1109/TVCG.2007.70535

[14] N. Elmqvist and J.-D. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Trans. on Visualization and Comp. Graphics*, 16(3):439–454, May 2010.

[15] J.-D. Fekete and C. Plaisant. Interactive information visualization of a million items. In *Proc. of the IEEE Symp. on Information Visualization (InfoVis'02)*, Washington, DC, USA, 2002. IEEE.

[16] N. Ferreira, D. Fisher, and A. C. König. Sample-Oriented Task-Driven Visualizations: Allowing Users to Make Better, More Confident Decisions. In CHI 2014.

[17] D. Fisher. Incremental, Approximate Queries and Uncertainty for Exploratory Visualization. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)* 2011.

[18] D. Fisher, I. Popov, S. Drucker, and m. schraefel. Trust me, I'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proc. of CHI '12*, pages 1673–1682, New York, NY, USA, 2012. ACM.

[19] S. Havre, B. Hetzler, and L. Nowell. ThemeRiver: Visualizing Theme Changes over Time. In *Proc. of the IEEE Symp. on Information Visualization 2000 (INFOVIS '00)*. IEEE, Washington, DC, USA.

[20] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. SIGMOD Rec., 26(2):171–182, June 1997.

[21] J.-F. Im, F. Giguère Villegas, and M. McGuffin. VisReduce: Fast and responsive incremental information visualization of large datasets. 2013 IEEE International Conference on Big Data

[22] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. M4: a visualization-oriented time series data aggregation. *Proc. VLDB Endow.* 7, 10 (June 2014), 797-808. DOI=http://dx.doi.org/10.14778/2732951.2732953

[23] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. VDDA: automatic visualization-driven data aggregation in relational databases. *The VLDB Journal 25, 1* (February 2016), 53-77. DOI=http://dx.doi.org/10.1007/s00778-015-0396-z

[24] S. Kandel, A. Paaepke, J. Hellerstein and J. Heer. Enterprise data analysis and visualization: an interview study. IEEE *Trans. on Visualization and Comp. Graphics* 18, 12 (2012). 2917-2926.

[25] S. Kandel, R. Parikh, A. Paepcke, J. Hellerstein, and J. Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In Advanced Visual Interfaces, 2012.

[26] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, and B. Ding. Quickr: Lazily Approximating Complex Ad-Hoc Queries in Big Data Clusters. *SIGMOD 2016.*

[27] D. Keim, H.-P. Kriegel. Visualization Techniques for Mining Large Databases: A Comparison. *IEEE Trans. on Knowledge and Data Engineering*, Vol. 8, No. 6, Dec. 1996.

[28] L. Lins, J. Klosowski, and C. Scheidegger. Nanocubes for Real-Time Exploration of Spatiotemporal Datasets. Visualization and Computer Graphics, IEEE Transactions on 19, no. 12 (2013): 2456-2465.

[29] Z. Liu, B. Jiang, and J. Heer. (2013). imMens: Real-time Visual Querying of Big Data. *Computer Graphics Forum*, *32*(3pt4), 421–430.

[30] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proc. of the 1996 IEEE Symp. on Visual Languages (VL '96)*. IEEE, Washington, DC, USA, 336-.

[31] B. Shneiderman. Dynamic Queries for Visual Information Seeking. *IEEE Software*. 11, 6 (November 1994), 70-77.

[32] D. Thompson, J. Bennett, C. Seshadhri, and A. Pinar. A provably-robust sampling method for generating colormaps of large data, *IEEE Symp. on Large-Scale Data Analysis and Visualization (LDAV)*, pp.77,84, 13-14 Oct. 2013

[33] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley: 1977

[34] M. Wattenberg. Visual Exploration of Multivariate Graphs, *Proc. ACM Conf. Human Factors in Computing Systems (CHI '06)* pp 811-819. 2006.

[35] H. Wickham. Bin-summarise-smooth: a framework for visualising large data. *had.co.nz Technical Report*. http://vita.had.co.nz/papers/bigvis.html

[36] P. Wong, H. Shen, C. Johnson, C. Chen, R. Ross. The Top 10 challenges in extreme-scale visual analytics. *IEEE Computer Graphics & Applications*. 32(4):63-67. 2012

[37] B. Yost, C. North: The Perceptual Scalability of Visualization. IEEE Trans. Vis. Comput. Graph. 12(5): 837-844 (2006)