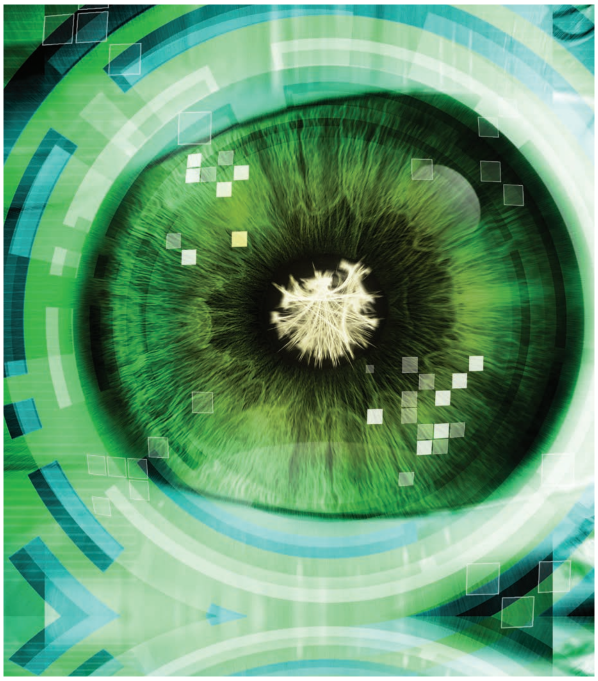
COVER FEATURE



Visual Analytics Infrastructures: From Data Management to Exploration

Jean-Daniel Fekete, *INRIA*

Analysts exploring big data require more from information visualization, data analysis, and data management than these components can now deliver. New infrastructures must address the nature of exploration as well as data scale.

Arguably the grand challenge of visual analytics is to support the exploration of and interaction with big data. In that role, visual analytics must contend not only with big data management and analysis, which have their own significant obstacles, but also with the nature of analysts' exploration demands. When facing large amounts of unfamiliar complex data, humans need overviews, summaries, and the capability to drill down to a level that will make it easier to look for patterns and correlations and to discover emergent phenomena, empirical models, and theories related to the data. The "Visual Analytics in Action" sidebar gives a flavor of these demands in exploring credit card transaction data to uncover fraud.

From the software developer's view, implementing a visual analytics application is difficult because traditional software and hardware layers lack important services that the human cognitive system needs in processing complex information. Even state-of-the-art analytics and data management do not yet meet visual analytics' requirements. These formidable issues require work to resolve so that visual analytics can fully support the interactive exploration of big data.

As a step toward that resolution, the Aviz project team investigated and qualified the human cognitive abilities that define analysts' exploration requirements and looked at how information visualization must change to meet these demands. Two open problems are the competing constraints of limited human cognition and the increasing strain that data is causing on software systems. An underlying cause of this strain is that analytics and data management software layers are not aligned with visual analytics. Correcting this mismatch will benefit not only visual analytics tools but also analytics and data management systems in general.

SOFTWARE AND HARDWARE LAYERS

Figure 1 shows the three layers of software and hardware that visual analytics must rely on in fulfilling an analyst's requirements.[1] Tools and techniques in all three categories—data management, analytics, and visualization—have greatly improved in the past decade, albeit largely in isolation. Recently, however, data management and analysis tools have begun to converge through the use of multiple technologies, including grids, cloud computing, and general-purpose graphics processing

Visual Analytics in Action

Discovering fraudulent schemes in credit card transaction data is one example of the demands on visual analytics support. Data consists of unexpected events in billions of daily transactions worldwide, with new schemes appearing frequently as thieves actively and creatively work against banks' automatic detection mechanisms. Bank analysts must explore transaction data continuously to identify and explain anomalies—most of which are not the direct result of fraud. The focus is on narrowing anomalies to fraudulent transactions, understanding the transaction's mechanisms, backtracing the transaction, and applying any corrective measures quickly.

The system's job is to raise alerts on nonstandard transactions, a type of transaction mining based on user profiling. The analytical processing applied to every transaction must be sensitive enough to detect important problems but not so sensitive that it creates many false positives, which could overwhelm the analyst. Once the system finds a fraudulent transaction, the analyst must not only cancel it but also understand what made it possible. To meet both requirements, processing must find similar frauds using a notion of similarity that is highly scheme specific—where it occurred, the people and goods involved, and so on.

At this stage, analysis amounts to finding a similarity measure in the data, which involves multiple queries to the transaction database. One technical difficulty is how to manage query latency. When an analyst queries the system with a particular hypothesis in mind, the system should send back an answer within a few seconds; otherwise the analyst may forget the hypothesis and need to retrieve it when the results arrive. The requirement for near-real-time results, even at the expense of accuracy, is due to the unique limitations on human cognition—but that is all humans currently have to make sense of the data. Most database queries produce just the opposite; results are entirely accurate but not bounded in time.

Continuing the scenario, the analyst will likely perform not only direct database queries but also more complex analytical queries involving machine learning and statistics. The investigation might even involve more complex video processing; perhaps the fraud involves cash machines that take videos of the operator, for example. Again, current analytics systems would perform these more complex analyses as accurately as possible, but in unbounded time.

The idea is to avoid making the analyst wait for results. In the traditional model, an analyst would start a series of analyses and then have to defer exploration to a later stage to avoid waiting for the system to provide results. In contrast, visual analytics uses faster, less-accurate mechanisms to provide initial results quickly and improves accuracy later if needed.

For example, if the fraud is restricted to a few cash machines in the same area, and the analyst wants to collect and classify the faces of possible accomplices, she can start processing videos from these machines. During the processing, if she realizes that the fraudulent operator is always wearing a motorcycle helmet—easier to recognize on video than human faces—she will want to change the recognition algorithm on the fly.

Analytical systems that run to completion with no or little feedback and do not let analysts steer the computation are not suitable for this kind of analysis. Instead of allowing dynamic interaction, the system restarts from scratch, ignoring any information gathered so far—even though most of the analysts' queries are refinements of previous queries. Consequently, much analysis work is lost. One promise of visual analytics is the elimination of such restarts.

units. High-performance computing on large amounts of data relies on the conjunction of new distributed data management technologies coupled with distributed computations.

Unfortunately, exploration has not been taken into account in these new infrastructures, and it seems daunting to introduce it without deep changes in both data management and analysis computation.

VISUALIZATION

For most researchers and practitioners, information visualization is the use of interactive visual representations of abstract data to amplify cognition.[2] Information visualization provides compact graphical presentations and user interfaces that enable the interactive manipulation of vast numbers of items, possibly extracted from far larger datasets.

All information visualization applications follow an abstract reference model that has evolved and matured to the version in Figure 2.[2]

As the figure shows, data flows from left to right. The system first transforms raw data into proper data tables that visual mapping then turns into visual structures.

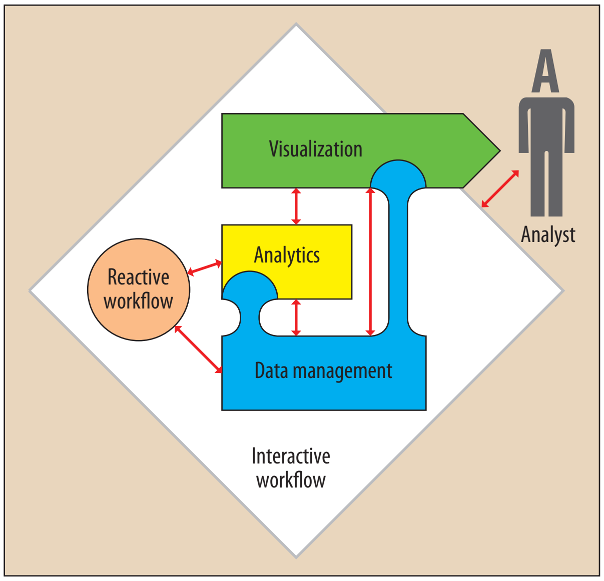


Figure 1. Three layers of software and hardware and their relationships with higher-level workflows and the analyst. Red arrows represent control mechanisms, solid blue areas represent data sharing and distribution, and the green area represents visualization as a communication medium between the system and the analyst.

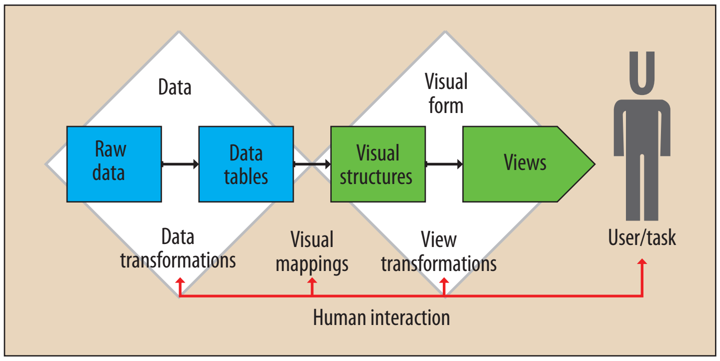


Figure 2. The latest version of the information visualization reference model. On top of the pipeline, the system adapts data to be visualized and uses visual mapping to convert the data into visual structures and views. User interaction (bottom) enables the control of all pipeline stages with immediate feedback.

Through view transformation, the visual structures then become views (images), which the system presents to the user through a viewport.

Interaction flows in the other direction. The user controls the view transformation parameters (essentially zooming and panning), the visual mapping parameters (color mappings, layout for networks, and so on), and database filtering.

MONOLITHIC VS. POLYLITHIC

Most traditional information visualization toolkits model visual structures as black-box components that manage their state internally and map views (rendering) directly to a window. This *monolithic* component model contrasts with the *polylithic* model, which represents visual structures as regular tables with a specific schema in which each data item has a geometry (or shape), an identifier, and a set of graphic attributes (fill color, transparency, stroke color, and so on).[3]

Although there is yet no consensus on preferred model type, one strong advantage of the polylithic model is its scalability; it is easy to implement on top of standard databases. Monolithic approaches suffer from the so-called impedance mismatch—the difficulty of converting their object-oriented structure into a database structure.

Tools that implement the information visualization reference model generally have interfaces that can rapidly propagate an interactive operation or view change. Human-computer interaction researchers have learned much about how humans work, and those insights have become part of visualization work as well. Ideally, to appear instantaneous, interactive operations should happen within 100 ms, and actions should complete within 10 seconds. Some operations, such as the computation of a large network layout, are much slower and must rely on strategies such as iterative solutions to minimize idle waits for the user.

To comply with these requirements, information visualization systems typically rely on ad hoc in-memory databases to hold the active data and use a column-oriented architecture to enhance performance. With few exceptions, the existing implementations are not based on standard databases. This lack of standardization is a clear call to action, and database and visualization researchers must join forces to ensure standardization.[4,5] Managing larger amounts of data will require faster and more sophisticated databases that are too complex for nonspecialists to implement. Conversely, existing databases will need extensions to manage the visualization data structures, which are likely to become a standard functionality that database systems will provide.

In addition, recent visualization research has focused on improving analytical algorithms to match the reactivity requirement. Some graph layout algorithms are special kinds of multidimensional projection algorithms, and researchers have devoted much effort to implement these projections efficiently and in a way that lets users see how projection computation is progressing. More sophisticated extensions let users steer iterative algorithms toward the computations of the greatest interest. These steerable algorithms are again variations of traditional analytical algorithms, such as multidimensional scaling enhanced to match the user's cognitive process.

Compared with information visualization, visual analytics involves a larger data scale, and the analyses to make sense of the data are more exhaustive. Consequently, software solutions must be both fast and reliable as well as support exploration.

DISPLAYS

In an attempt to increase the amount of visualized data, several organizations are exploring the use of wall-size displays, or power walls, as Figure 3 shows. However, enlarging displays is not trivial; developers must carefully consider screen size and resolution, the number of screens connected to each machine, how to distribute graphical applications to multiple machines, and how to handle input devices and interaction.[6]

Moreover, increasing pixels and improving input devices will not solve the problem of visualizing big data because perception is a major bottleneck. Human visual perception is limited by the eye's physiology and the brain's capacity to process images. The eye comprises about 125 million rods and 6 million cones as low-level receptors. A user could not perceive more than a few million features and is more likely to max out at considerably fewer.

Adding more pixels might enable more people to collaborate in the same environment, but it will not empower one user to perceive more visual information. In addition, many datasets—notably, those maintained by Facebook and Google—exceed the capacities of even a wall display by several orders of magnitude. Consequently, visual analytics must rely on analysis to summarize data and guide exploration.

ANALYTICS

Analytics is the science or process of drawing conclusions from data analysis, so it makes sense that visual analytics systems frequently reuse existing data analysis systems, languages, and libraries and occasionally extend them.

Data analysis systems and languages are entrenched and diverse. As of May 2013, Wikipedia referenced 87 numerical analysis software systems, which include general-purpose languages such as Matlab or R, as well as libraries for more specialized computations such as OpenCV for video analysis (http://opencv.org) or GATE for text analysis (http://gate.ac.uk).

All these systems follow the same simple workflow: load a data file, process it, compute some analytical quantities, and export result files or statistical charts. Computation and analyses are often seen as black boxes that take tables as input and output, along with a set of parameters, and run to completion or error without interruption. This concept fits well when managing reasonable amounts of data according to a streamlined process to obtain a final result. The system will eventually get a result, and the analyst does not have to wait idly even if the computation takes a long time.

For exploration, however, this workflow model breaks down. The analyst might need to try several processing methods to search for interesting results. With small amounts of data, the system can compute analyses quickly, and the analyst can use charts to visualize the results or export the results to visualization systems for exploration. Indeed, both academic and industry analysts use this pairing of visualization and analytical environments extensively.

With big data, the loose coupling between visualization and analysis presents problems. Data transfer time exceeds the reactivity requirement—moving data back and forth takes minutes, sometimes hours. To avoid costly data transfers, the developers of many analytical environments have added basic visualization functionality to their tools. However, the visualizations in most of these environments are less sophisticated than dedicated information visualization tools because the aim was visualization for publication and communication, not exploration.

Another issue is that the algorithms in these analytical environments are not set up to provide early results quickly because, again, exploration is not their intended application. Adapting the design requires rewriting processing and analysis algorithms to account for the

Figure 3. Visual analytics on a wall display. Even displays of this size cannot handle the capacities of many current datasets, nor overcome the limitations of human perception (www.youtube.com/watch?v=5i3xbitEVfs).

analyst in the loop. So far, the dominant strategy has been to optimize existing analytical environments that often rely on scripting languages. Riposte[7] provides a just-in-time compiler for the R language that has achieved speedups of 50× or more using the multicore capabilities of modern processors to accelerate R's standard vector operations. The same approach could be used to optimize Matlab or Python code. The drawback of acceleration is that it merely pushes the limit of data size that a user can effectively explore in an analytical environment; it does not directly address how to scale visual analytics.

Solutions to the scalability of standard analytics have traditionally involved high-performance computing—first with expensive computers and more recently with affordable multimachine architectures that include computer grids and cloud computing. Although computation throughput is high, it comes at the cost of high latency, which makes these architectures ill-suited to the exploratory tasks of visual analytics.

To manage large amounts of data, visual analytics researchers and practitioners have had to reimplement existing algorithms. Popular reimplemented algorithms, which are typically available in statistics and machine-learning libraries, include hierarchical clustering and principal component analysis computation, among other clustering and projection methods.

As is true of information visualization experts who have had to reimplement in-memory databases, reimplementing algorithms is a suboptimal use of skills and resources. Fortunately, as visual analytics becomes more visible, researchers have begun collaborating to design and implement a new generation of analytical tools with better support for humans in the loop. Examples of such collaboration were evident from reports at the 2012 Dagstuhl seminar on information visualization, visual data mining, and machine learning, as well as in ongoing work at the University of British Columbia (www.cs.ubc.

ca/labs/imager/tr/2004/mdsteer) to explore interactive multidimensional projections and at Stanford University to investigate large-scale text visualization and analysis (http://vis.stanford.edu/papers/designing-model-driven-vis). This support will require anytime algorithms that trade accuracy for speed and that can repair and continue computations instead of stopping and restarting from scratch.

DATA MANAGEMENT

The Data Management International Association defines data resource management as "the development and execution of architectures, policies, practices, and procedures that properly manage the full data lifecycle needs for an enterprise" (www.dfwdama.org/aboutUs.asp). Clearly, visual analytics applications need a reliable and efficient data management system layer to manage their data life cycle, but the current offerings are not compatible.

Visual analytics applications should rely on standard data management systems instead of building their own.

Current SQL databases are a case in point. These databases provide scalable storage, fast queries, and efficient distribution to transparently load and save data through a network or shared memory, but standard versions are transactional; they assume that an application will select a small set of items for processing or write a small set at a time. In visual analytics, however, the analyst can ask to visualize an unbounded portion of a database at any time either directly or through an aggregate query. These requirements make it infeasible to use SQL databases alone, so most visual analytics applications use one in-memory database and a different external data management system. Some visualization systems, such as Tableau (www.tableausoftware.com), rely on a standard SQL database but use a custom in-memory database to hold visualization structures.

Work is progressing to address the mismatch between database technologies and visualization needs with an eye toward having visual analytics applications rely on standard data management systems instead of building their own.

Two trends have emerged in efforts to extend database capabilities. NoSQL systems have relaxed some of the mandatory properties of standard databases—atomicity, consistency, isolation, and durability (ACID)—as well as their reliance on SQL. Meanwhile, SQL systems have continued to extend their capabilities, adding high-performance features while keeping their ACID properties.

To shorten reaction time, some visual analytics applications have even gone as far as reimplementing their data management layer. One approach is to implement aggregation queries by returning running approximations.[8] Another effort explored the possibility of implementing fast incomplete queries on MonetDB, a high-performance standard SQL database.[9]

In the NoSQL world, Google has designed Dremel,[10] a query engine that aims to perform extremely fast queries on Google's infrastructure; according to Google, Dremel can run aggregation queries of more than a trillion-row tables in seconds. SAP's Hana-DB, an in-memory database system for SAP's business-analytics applications,[11] demonstrates that visual analytics is clearly of commercial interest but will require substantial infrastructure investments to provide an acceptable user experience.

Overall, data management technologies are evolving quickly to provide better support for exploration and interaction. A few, such as Hana-DB, provide core usability for visual analytics applications. The data management community has also begun to recognize the needs of these applications, and some tools to address issues such as high-performance in-memory databases are already usable. Even so, developers and data management providers need more time to develop infrastructure that fully supports analysis and standardization.

INTEGRATED ENVIRONMENTS

Although researchers have made some progress in integrating the three software layers in a consistent framework, additional issues have arisen, most of which center on how to handle the workflows in visual analytics applications.

Novel workflows

Workflows aim to carry out complex or repetitive computations by sequencing computational and flow-control modules and running them in a database. Workflows are popular in business applications, such as banking, in which they control automated operations instead of relying on humans to perform risky or time-consuming operational sequences. Scientific workflows, for example, control the operational sequence and flow in processing large bodies of scientific data. Because workflows are constructed in advance, multiple operations are often performed in parallel, and most computations run from start to end with no interruption and little feedback.

Two new kinds of workflows have been designed specifically for visual analytics: reactive workflows and interactive workflows.

Reactive. Traditional workflows run a sequence of processes with each database record modification, and scientific workflows run a set of processes on the entire database one time, but visual analytics must perform computations on the whole dataset each time something in it changes. This distinction is important in managing dynamic data, since users can opt to run operations ahead of time to prepare for interactive exploration. For example, a company interested in monitoring the reputation of its products can perform sentiment analysis for all the webpages that mention those products and continue performing the analysis whenever the system fetches new pages, rather than just at exploration time. An example of continuous analysis is the EdiFlow system, which lets analysts specify a set of operations that should be performed on the data each time it changes.[12]

A naive approach would be to run the whole workflow for the whole database for each update, which would consume resources and become impractical if changes occur faster than the system can process them. To avoid recomputing everything from scratch, EdiFlow tries to repair what the system has computed. Each computation module decides on the repair strategy according to that module's computational semantics. In the same vein, processes can decide to interrupt or repair their long computations when they are notified that the data table they operate on has changed.

Reactive workflows require nonstandard support from their underlying database. First, they need a notification mechanism, such as a trigger, to control when workflow restarts. Managing asynchronous notifications and communicating with an external workflow system each time a database modification occurs requires extending the notion of transactions to allow several processes to compute on the same tables without conflicts. In the EdiFlow prototype, every table record is explicitly time-stamped to support long transactions. In parallel, Oracle has also implemented this feature in Total Recall, an extension of its database.[13] Again, Oracle's effort shows that industry is becoming interested in satisfying visual analytics' requirements.

Interactive. In visual analytics, exploration is essentially the construction of a workflow as a cascade of operations that filter, summarize, and analyze the data, asking such questions as:

- What are the right operations to apply?
- What parameters should be used for these operations?
- What is the difference between applying one sequence of operations versus another?

All these questions arise when exploring data, either to understand it or make it usable. In intermediate stages, visualization shows the results of data processing and provides additional feedback and quality control measures.

VisTrails interactively builds and runs scientific workflows that include visualizations.[14] Even as a mature platform, it continues to evolve quickly because so many application domains are using it, including physics, biology, medicine, and climate-data modeling. VisTrails is implemented in Python and can run native applications or use libraries once they are wrapped into Python modules. Writing these wrapper modules is easy, and already many packages have been integrated into VisTrails.

The novel part of VisTrails is its facility for tracking workflow changes, which it records using a provenance management subsystem. The provenance facility, which lets analysts iteratively improve their workflow while keeping track of the changes, is not limited to one person. Rather, analysts can exchange workflows for interactive and collaborative exploration to help others understand a process, reapply it, or continue it.

On the downside, the VisTrails architecture still relies on data exchange through standard input/output operations and does not yet enforce any data management mechanism—both obstacles to optimizing the workflow's data communication part. Consequently, data communication cost adds time to any effort to link the analysis and visualization of large datasets.

Exploration is essentially the construction of a workflow as a cascade of operations that filter, summarize, and analyze the data.

Abstraction layers

The cost of data transfer between analysis and visualization modules adds delays between modules and wastes memory by duplicating data between all the modules that need it. The Obvious metatoolkit uses an abstraction layer to help overcome this problem.[15] The creators of Obvious implemented it as a Java library and designed it to encapsulate other Java libraries. Obvious relies on the idea that all the classes to implement data tables in software libraries are very similar—if not identical—except for variation in method and class names, which is true for visualization systems as well as for analysis and database systems. Obvious provides an abstract class definition that serves as a pivot to share data for all encapsulated Java libraries. Wrapping a library requires only two classes—one to map a native data table to an Obvious data table and the other to map the tables in the other direction. For the most part, these mappings involve calling methods with other names; they do not introduce noticeable slowdowns in accessing data.

A wrapped library can exchange its data with other libraries almost for free, enabling database modules to produce Obvious tables that machine-learning modules can process in place and pass to visualization modules without additional input/output or data duplication.

The compatibility layer between all the data-table models is a superset of common functions that specifies a semantic of notification, which is usually inconsistent between libraries. Currently, Obvious wrappers are defined for four visualization toolkits, two machine-learning toolkits, and the standard Java API for database access. Obvious is an example of how designers can address the data communication issue through a data service that maps efficiently to multiple libraries. This library-neutral, in-memory data abstraction is an important data management component for visual analytics that to some extent generalizes across languages and systems.

OPEN OBSTACLES

Collaboration among visual analytics, visualization, analytics, and data management efforts has already begun, and commercial vendors are becoming more aware of the pressing issues. Companies such as SAP, Google, and Oracle implemented special database infrastructures suitable for visual analytics even before it became popular in research—evidence that visual analytics is already an identified market for these companies.

At the data management layer, vendors are beginning to offer in-memory databases with support for extended data types, such as geometric shapes, although these extra services are not yet standardized. Once the required services are available, visualization systems can adopt them instead of relying on their own limited implementations. Database researchers need to explore mechanisms that trade accuracy for speed in queries, support query refinement, and add new types of long transactions suitable for expensive computations.

Adding data abstractions atop database services, as in Obvious, might be a way to extend data sharing to additional languages such as Python or R, and could avoid data transfer altogether in visual analytics applications and workflows such as VisTrails or EdiFlow. This table-sharing and caching service would differ from the transaction-based service that most databases offer because it would allow the implementation of much more efficient optimizations from inside the database driver than are possible on top of transactions.[9]

At the analytics layer, research must be devoted to designing analysis modules that can repair computations when data changes, provide continuous feedback during computation, and be steered by user interaction whenever possible. At a higher level, balancing the tradeoff between speed and accuracy might call for new analysis strategies, as well as tools that can change between strategies depending on the analyst's current goal. Providing these high-level analysis strategies to users requires more research.

At the system level, integration of visualization, analytics, and data management will require more collaboration among researchers and practitioners across all three domains. Information visualization relies on a well understood and accepted reference model; visual analytics must do likewise. All domains must contribute more time and experience to design such a model.

Existing software and hardware architectures in visualization, analytics, and data management are not yet suited to integration into visual analytics applications, primarily because big data exploration is relatively new. In particular, the constraints from human cognition are not well known outside information visualization research, but they have strong implications on the services required to effectively explore large amounts of data.

Some prototypes and solutions partially address these limitations and are mature enough to spread to production software. Other libraries will require more research or experimentation to mature. Eventually, visual analytics could become easier to implement and more widespread, but until then providing the mechanisms for exploration in databases and analysis systems will benefit any situation in which users are willing to trade accuracy for time. This tradeoff is important because time is rapidly becoming one of society's most important resources.

Acknowledgments

Thanks to Wesley Willett for his help in improving earlier drafts of this article and to Jeremy Boy for the original graphics.

References

1. J. Thomas and K. Cook, eds., *Illuminating the Path: Research and Development Agenda for Visual Analytics*, IEEE, 2005.
2. S. Card, J. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann, 1999.
3. B.B. Bederson, J. Grosjean, and J. Meyer, "Toolkit Design for Interactive Structured Graphics," *IEEE Trans. Software Eng.*, vol. 30, no. 8, 2004, pp. 535-546.
4. J.-D. Fekete and C. Silva, "Managing Data for Visual Analytics: Opportunities and Challenges," *IEEE Data Eng. Bull.*, vol. 35, no. 3, 2012, pp. 27-36.
5. J.-D. Fekete, "Infrastructure," *Mastering the Information Age—Solving Problems with Visual Analytics*, D. Keim et al., eds., Eurographics Assoc., 2010, pp. 87-108.
6. M. Beaudouin-Lafon et al., "Multisurface Interaction in the Wild Room," *Computer*, Apr. 2012, pp. 48-56.
7. J. Talbot, Z. DeVito, and P. Hanrahan, "Riposte: A Trace-Driven Compiler and Parallel VM for Vector Code in R," *Proc. 21st Int'l Conf. Parallel Architectures and Compilation Techniques* (PACT 12), ACM, 2012, pp. 43-52.
8. D. Fisher et al., "Trust Me, I'm Partially Right: Incremental Visualization Lets Analysts Explore Large Datasets Faster," *Proc. SIGCHI Conf. Human Factors in Computing Systems* (CHI 12), ACM, 2012, pp. 1673-1682.

9. P.A. Boncz, M.L. Kersten, and S. Manegold, “Breaking the Memory Wall in MonetDB,” *Comm. ACM*, vol. 51, no. 12, 2008, pp. 77-85.
10. S. Melnik et al., “Dremel: Interactive Analysis of Web-Scale Datasets,” *Proc. 36th Int’l Conf. Very Large Databases* (VLDB 10), IEEE CS, 2010, pp. 330-339.
11. F. Färber et al., “The SAP Hana Database – An Architecture Overview,” *IEEE Data Eng. Bull.*, vol. 35, no. 1, 2012, pp. 28-33.
12. V. Benzaken et al., “EdiFlow: Data-Intensive Interactive Workflows for Visual Analytics,” *Proc. 27th Int’l Conf. Data Engineering.* (ICDE 11), 2011, pp. 780–791.
13. Oracle Inc., “Total Recall White Paper,” Sept. 2012; www.oracle.com/technetwork/database/focus-areas/storage/total-recall-whitepaper-171749.pdf.
14. S.P. Callahan et al., “VisTrails: Visualization Meets Data Management,” *Proc. SIGMOD Int’l Conf. Management of Data* (SIGMOD 06), ACM, 2006, pp. 745-747.
15. J.-D. Fekete et al., “Obvious: A Meta-toolkit to Encapsulate Information Visualization Toolkits—One Toolkit to Bind Them All,” *Proc. Conf. Visual Analytics Science and Technology* (VAST 11), IEEE, 2011, pp. 89-98.

***Jean-Daniel Fekete** is the scientific leader and founder of INRIA’s Aviz Project team. His research interests are in visual analytics, information visualization, and human-computer interaction. Fekete received a PhD in computer science from Université Paris-Sud. He is a senior member of IEEE. Contact him at jean-daniel.fekete@inria.fr.*