Lytic: Synthesizing High-Dimensional Algorithmic Analysis with Domain-Agnostic, Faceted Visual Analytics

Edward Clarkson[1], Jaegul Choo[2], John Turgeson[1], Ray Decuir[1] and Haesun Park[2]

Georgia Tech Research Institute
925 Dalney St., Atlanta, GA 30332-0834
{edward.clarkson, john.turgeson, ray.decuir}
@gtri.gatech.edu

School of Computational Science & Engineering
Georgia Institute of Technology
266 Ferst Drive, Atlanta, GA 30332-0765
{jaegul.choo, park}@cc.gatech.edu

ABSTRACT

We present *Lytic*, a domain-independent, faceted visual analytic (VA) system for interactive exploration of large datasets. It combines a flexible UI that adapts to arbitrary character-separated value (CSV) datasets with algorithmic preprocessing to compute unsupervised dimension reduction and cluster data from high-dimensional fields. It provides a variety of visualization options that require minimal user effort to configure and a consistent user experience between visualization types and underlying datasets. Filtering, comparison and visualization operations work in concert, allowing users to hop seamlessly between actions and pursue answers to expected and unexpected data hypotheses.

Categories and Subject Descriptors

H.4.2 [Information Systems]: Types of Systems–Decision Support

General Terms

Design, Human Factors

Keywords

Visual analytics, scientific intelligence, infovis

1. INTRODUCTION

General-purpose data analysis tools—spreadsheets—were at the forefront of the personal computer revolution in the late 1970s and early 1980s. As a so-called "killer app" that showcased PC utility, the benefits of digitized data helped to drive widespread corporate adoption of personal computing [7]. In the 2010s, the wide availability of large amounts of data from a variety of sources has shown the potential for a similar revolution centered on data, and more importantly the *knowledge* that can be extracted from it by algorithms and human ingenuity.

And yet, there are relatively few general-purpose tools for such large-scale data analysis, particularly for exploratory purposes. Broadly, visual analytics addresses this problem through the synthesis of sophisticated computational algorithms with human initiative, and especially our high-functioning visuospatial perception capabilities.

This paper presents *Lytic* (from the word Ana***lytic***), a flexible, domain-independent visual analytic (VA) system for interactive exploration of large datasets. Its design focuses on facilitating key data analysis tasks that occur across disciplines: filtering, transformation, comparison, and visualization. Data can be formatted in convenient (character-delimited flat files) formats, with the UI adapting to the data characteristics (# of items, data types, etc.). Users create visualizations (scatter plots, line charts, parallel coordinates, etc.) simply by choosing variables from dropdown lists.

Lytic also integrates dimension reduction and clustering (DR/C) algorithms to facilitate analysis of high-dimensional data (e.g., text document term frequencies). It thus contributes to the literature a scalable system suitable for domain experts (rather than visualization or algorithm experts) to explore a wide variety of datasets[1].

IDEA '13, August 11th, 2013, Chicago, IL, USA.

2. RELATED WORK

Several related lines of research have informed the Lytic system. Faceted classification is a library science concept derived from colon classification [24], which in turn has been independently replicated in database systems (cf. dimensional modeling [19]) and information visualization (e.g., in the Attribute Explorer [26]). Its core concept is that of classifying items of interest along multiple independent axes, allowing more powerful and descriptive classifications than strict hierarchical organizations. When implemented in software, this pattern (which is both a UI and a data model pattern) is commonly referred alternately as *faceted search*, *browsing,* or *navigation.*

This design pattern has also become common in e-commerce sites, for example: customers can browse for clothes by filtering or comparing available items based on their size, style, color, price, manufacturer, etc. Because users can easily determine what characteristics (or *facets*) are available and hop between them to perform filters, this method has proven to be powerful, flexible and user-friendly to a wide audience. This commercial popularity was preceded by their use in many research systems [5,30,31]. Notably, faceted UI systems commonly display information about the values within each facet (e.g., the number of items matching a given category) and dynamically update it with successive queries [12].

There has long been recognition of the importance of the exploratory search and its distinction from targeted information seeking [22]. Information visualization (*infovis*) systems in particular are concerned with that distinction, as much of their value comes not just from allowing users to answer specific questions, but from harder-to-quantify attributes such as allowing

[1] A 64-bit windows binary release is publicly available from http://tmt.gtri.gatech.edu/lytic.html

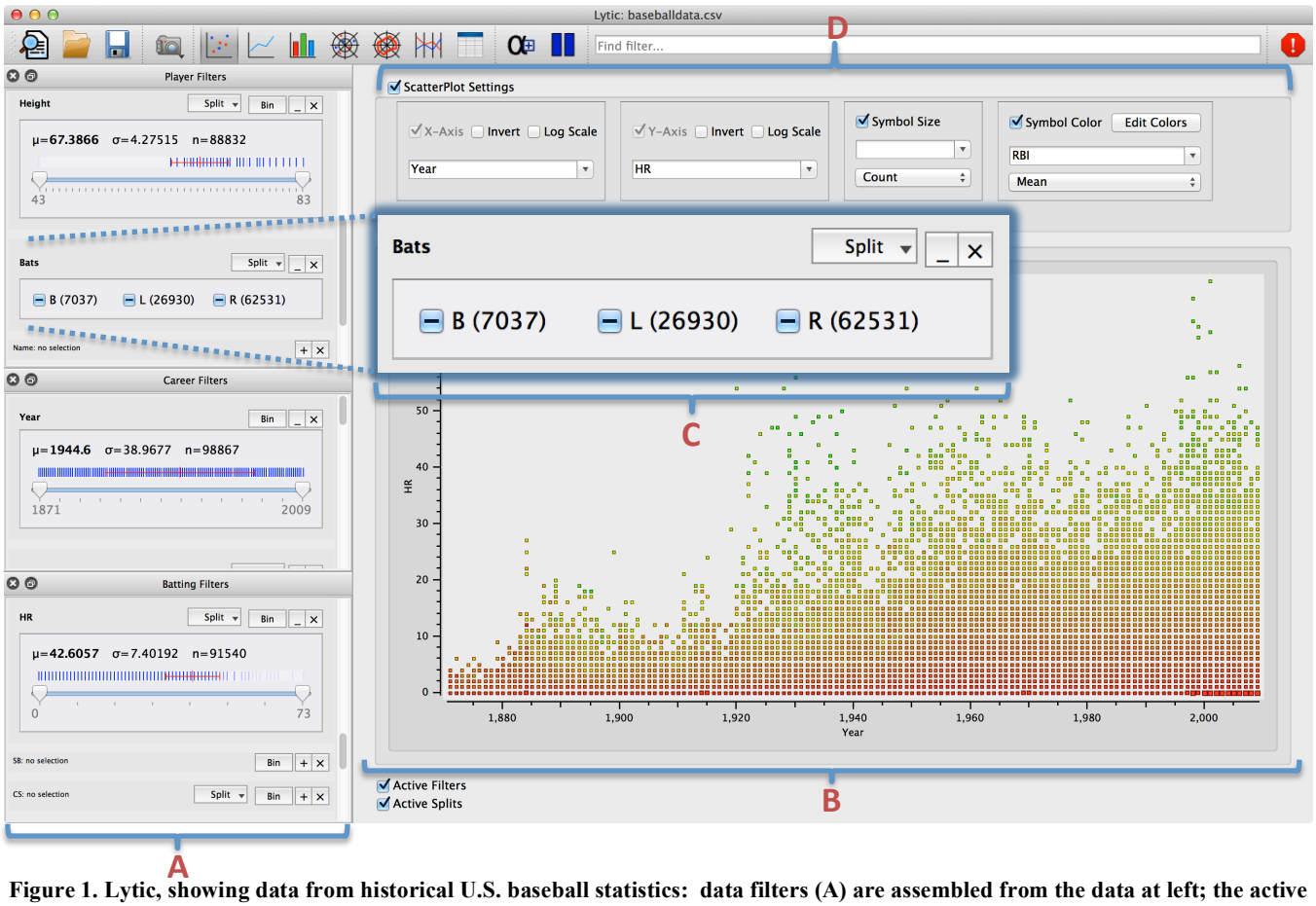


Figure 1. Lytic, showing data from historical U.S. baseball statistics: data filters (A) are assembled from the data at left; the active visualization (B) is at right. Filter choices update the active visualization and the choices available in other filter widgets. Inset (C) shows detail of a filter for the nominal variable *Bats* indicating a player's handedness. The view shows the number of home runs hit by players over time, configured by the common view controls (D).

users to ask *better* questions, or prompting users to ask questions they did not know they had in the first place [14].

Because software must first be derived from some implicit or explicit set of requirements, infovis work has therefore been concerned with the difficult question of what kind of end-user tasks it should support in order to help users achieve ill-defined, mutable, vague goals. One of the best known is Shneiderman's Visual Information Seeking mantra: "Overview first, zoom and filter, then details-on-demand" [25]. While the mantra concerns software, other work in both the infovis and VA communities characterizes the nature of both high- and low-level human activities and goals (Yi et al. provide a recent survey [32]). But informally, such activities include data filtering, comparisons, transformation, and visualization.

Infovis and VA systems have sought to support these kinds of tasks implicitly or explicitly in general-purpose analysis tool. Systems like Jigsaw [27], Apolo [8], and IN-SPIRE [29] all provide varying levels of support for these kinds of tasks in service of exploratory search goals. The overlapping domain of data warehousing within the database community also contributes to this space and commonly refers to these kinds of tools as a part of a *business intelligence* software environment.

Polaris [28] and its commercial successor Tableau exemplify this type of work, extending the Excel pivot table concept to provide a wide variety of filtering and visualization capabilities for arbitrary data. TIBCO Spotfire also began as an academic work [1] before becoming a commercial products. Many other purely commercial products also exist, such as SAS JMP, Pentaho, and many others.

DBMS-centric visualization environment tools generally act as an SQL abstraction layer, reducing the amount of programming expertise required. The Tioga-2 visualization environment [2] was an early example of providing a programming abstraction (boxes-and-arrows) specifically for visualization purposes. Zloof proposed a rendering-by-example [20] environment (building off his well-known query-by-example work [33]) that would allow users to build interactive renderings via a direct manipulation style interaction. More recent system include specialized designs such as building queries within a radial visualization [13] and querying electronic health record (EHR) data [15].

Influential infovis systems often employ brushing and linking interaction facilities to assist interactive exploration. The attribute explorer [26] consists of a set of linked histograms that show the distribution of the dataset along many dimensions, and dynamically update with filtering operations. Brushing scatterplots [4] visualize multidimensional data by forming a scatterplot matrix of arbitrarily many dimensions, and dimensional relationships can be explored by brushing items in the visualization. Both of these systems tightly couple a particular visualization technique with data exploration (filtering, comparison, interaction) actions: in contrast, Lytic decouples such operations from the visualization because 1) as the number of dimensions grows large, such small multiples-style approaches

become problematic and 2) it allows for operations on dimensions that are not being visualized.

Other systems have approached data exploration and visual analytics from a more algorithmic perspective, using machine learning or other data mining-oriented computation. The FODAVA Testbed system provides a library of old and new dimension reduction and clustering algorithms and provides a visualization environment to evaluate their performance on different datasets [10]. The VisIRR system uses the same library to implement a system for dynamic exploration of large-scale document corpora [9].

Lytic is distinct from previous work in two ways. First, it approaches its UI design from a faceted navigation rather than a RDBMS or pivot table orientation. Second, it directly integrates a scalable, general-purpose data exploration-and-visualization tool with algorithmic (DR/C) preprocessing.

3. DESIGN

Lytic is based on the data analysis component of a modeling and simulation management framework [11], and thus its origins are in analyzing large, high-dimensional simulation datasets for scientific and engineering applications. As a result, we think of it as a sort of *scientific intelligence* tool, complementing the existing business intelligence tools. It is implemented in C++ using the Qt framework, which provides the ability to compile a desktop application for Windows, Linux and OS/X.

As such, Lytic is designed as a generic data analysis tool to support common analytic activities that exist across application domains (e.g., filter, compare, transform, visualize). Its design imperative is to make common tasks as easy and fast as possible to complete, while providing facilities for more complex exploration through extension and customization. The focus on optimizing common tasks accelerates the human-in-the-loop analytic cycle, providing more opportunities for opportunistic and goal-driven insights.

Figure 1 shows the high-level organization of the tool, which is populated by an initial **data ingest and processing** step. **Data filter** widgets, which are created for each facet/dimension, are grouped at left. **Data view** choices (e.g., scatterplot visualization) at right. Filter operations update the backing **data model**, which is structured as a SQLite database, and in turn update the data view as well as other filters. **Data comparisons** notify the view that it should provide a partitioning of its representation along some dimension. The user may perform several types of **data transformations** on the data, including unit conversions and customized formulations of new variables. The following sections provide details of each of these topics.

3.1 Data Ingest and Processing

In keeping with this lowest-common-denominator support, Lytic reads row/column-oriented character-separated value (CSV) files. The convention used by Lytic and this paper is to treat columns as data dimensions or variables, and rows as distinct data items. A key component of Lytic is the integration of the algorithmic suite developed for the FODAVA testbed [10], which provides a variety of dimension reduction and clustering algorithms[2]. An end user can specify one or more algorithms to selected dimensions so that the use can leverage the algorithms to better analyze the data or use the tool to analyze the algorithms themselves. For very high-dimensional data (to us, more than ~1000 dimensions), the user can also augment the main data file with sparse matrix syntax auxiliary (high-dimensional data can also be specified in the CSV file).

Integrating DR/C algorithms directly helps analysts deal with high-dimensional data more effectively, which often occurs from automated processing of some source data, such as term frequencies in document data or image characteristics. DR/C algorithms thus summarize those dimensions efficiently, reducing the complexity of the data space for the analyst.

High-cardinality data (i.e., large number of data items/rows) also presents a scalability challenge for most DR/C algorithms, which are commonly O(N^2) or worse in processing time, memory space, or both. Lytic is intended as an interactive end-user tool on commodity PC hardware, so it is important that ingest and load be limited to a reasonable amount of time—for us, a few minutes in the common case and no more than a few hours. As such, "high-cardinality" is a relative term for DR/C scalability; we have found

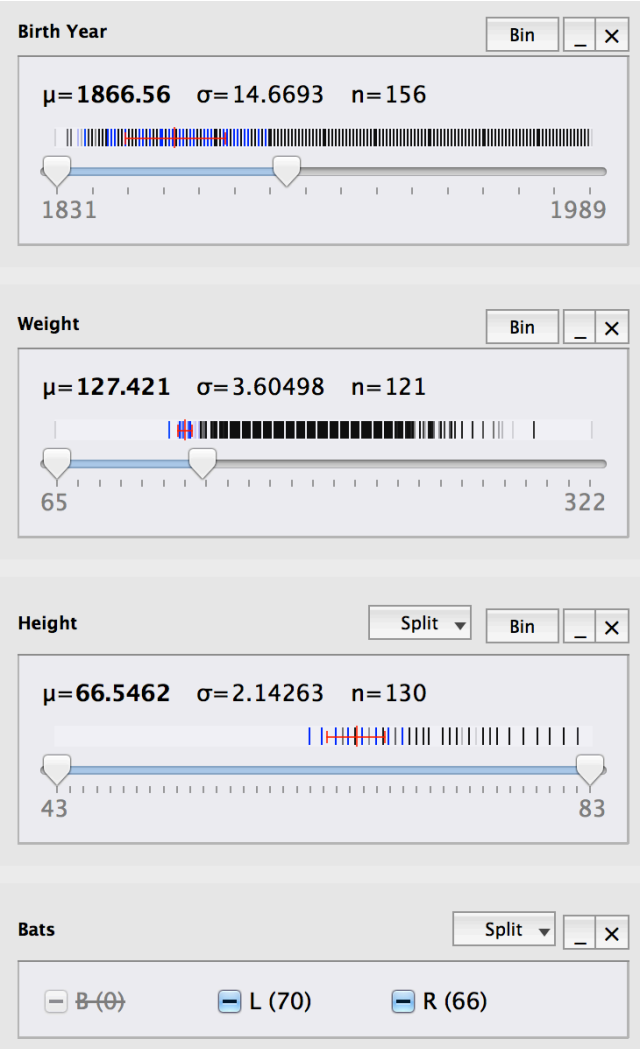
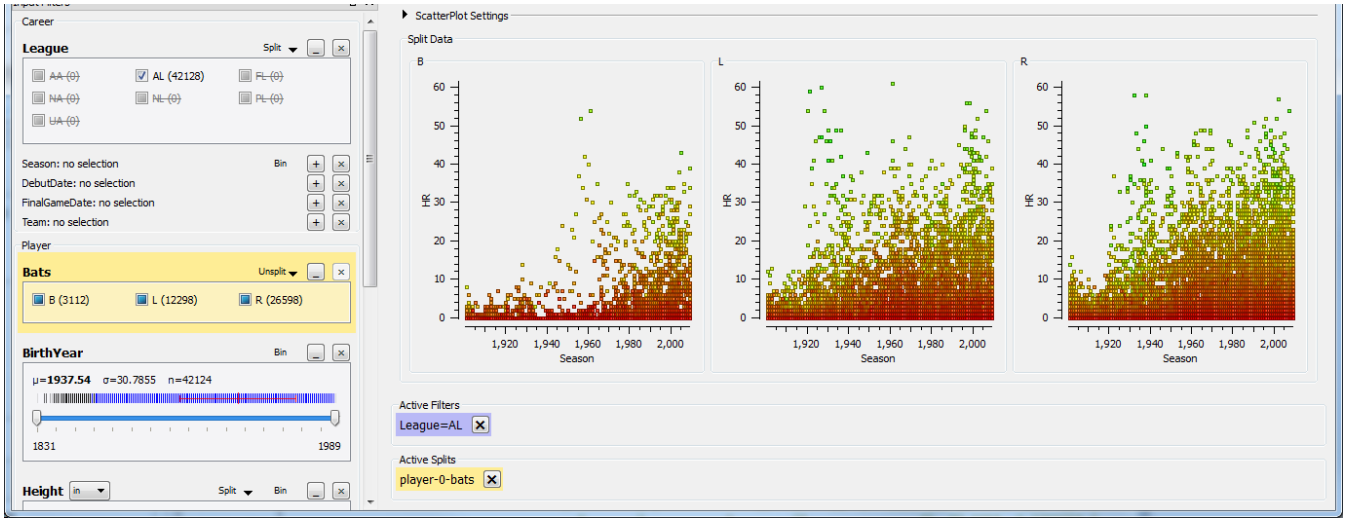


Figure 2. Three filter widgets for scalar data, which use double-ended sliders rather than checkboxes and render visualizations of the data distribution for that variable. Actions which would yield no data are disabled (Bats = 'B').

[2] The FODAVA suite includes newer techniques like linear discriminant analysis (LDA) and non-negative matrix factorization (NMF) as well as older algorithms like principal component analysis (PCA) or K-means clustering.



on modern laptop processors with SSD storage and 8-16 GB RAM this threshold to be ~2000 items.

Lytic itself can scale 2-3 orders of magnitude larger (see the following section), so we attempt to deal with this disparity by sampling and interpolation [3]: we randomly select a designated number (1000 by default) of rows and use only that data as input to the DR/C algorithms. Those results are then extrapolated to the extra-sample data items.

3.2 Data Model

The Lytic data model is similar to that of many OLAP-style tools. The combined CSV and synthesized DR/C data are imported into a single SQLite table, trading space efficiency for speed. All columns are indexed. This monolithic construction deviates from data warehousing and previously-described norms [12] because Lytic was initially targeted at scientific datasets with largely numeric, non-hierarchical data.

We define the data *scope* as the items/rows in the data model that have not been filtered out (see the following section): initially, all data are in scope. Filter operations are essentially SQL query generators the successively limit the scope of the data to smaller subsets. Since all columns are indexed, queries are executed efficiently, and become more efficient the more filters are put in place since the scope is necessarily reduced. Fields generated by DR/C algorithms are treated as any other filter, thus integrating synthetic and first-order characteristics of data items.

Lytic's scalability is dependent on the memory and speed of the user's workstation as well as the nature of the test data (the type and number of variables). On modern workstation-level hardware, we have found practical limits with reasonable interaction rates [6] to be between 100,000 and 1 million items with up to about 300 dimensions (after dimension reduction).

Lytic maintains a clean separation (with one minor exception, see §3.5) between its model and view, so other SQL implementations can easily replace the current Sqlite implementation (e.g., a column-oriented store like MonetDB), or even replaced by a non-SQL alternative.

3.3 Data Filtering and Comparison

Lytic's interaction design is focused on minimizing actions necessary to accomplish common operations: filtering, comparing, and visualizing. A faceted navigation-style UI makes as much data visible as possible and removing opportunities to make

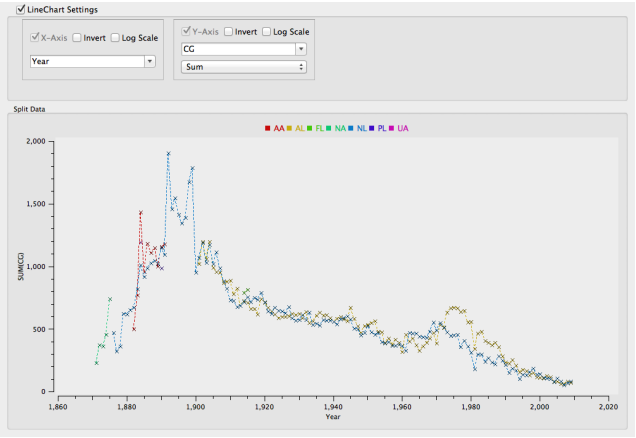


Figure 3. Comparison of split scatterplot (top) and line chart (bottom) views.

unproductive choices. Streamlining these operations reduces the barriers to iterative investigation through visual hypothesis testing and analysis. Filtering operations are accomplished by single actions: clicking a checkbox or dragging a slider handle. Dragging a slider handle dynamically queries the data view (subject to performance constraints).

Lytic forms a filter UI element for each field/column in the data model adapted to the nature of the data in the field: high-variation numeric data use a slider GUI element; nominal and low-variation scalars have a series of checkboxes (cf. Figure 1 inset) with parenthetical values indicating the number of data items matching that filter value. In Figure 1, there are 7037 B (baseball players who hit with **B**oth hands) players—many fewer than either **L**eft or **R**ight handed players.

For numeric (scalar) data, basic statistics (mean, std. dev., count) are calculated about the variable distribution. If there are sufficient distinct values, the UI includes a double-ended slider element to make range-based filters and renders a simple visualization of the data distribution). Blue marks indicate items that are in scope; black marks have been filtered out. Figure 2 shows three scalar filters (and one nominal); the user has filtered out all items with Weight greater than 135 and Birth Year greater than 1899, so all marks above those values are black in their respective distribution visualizations. However, there are black marks in the height filter as well, indicating how the filters from

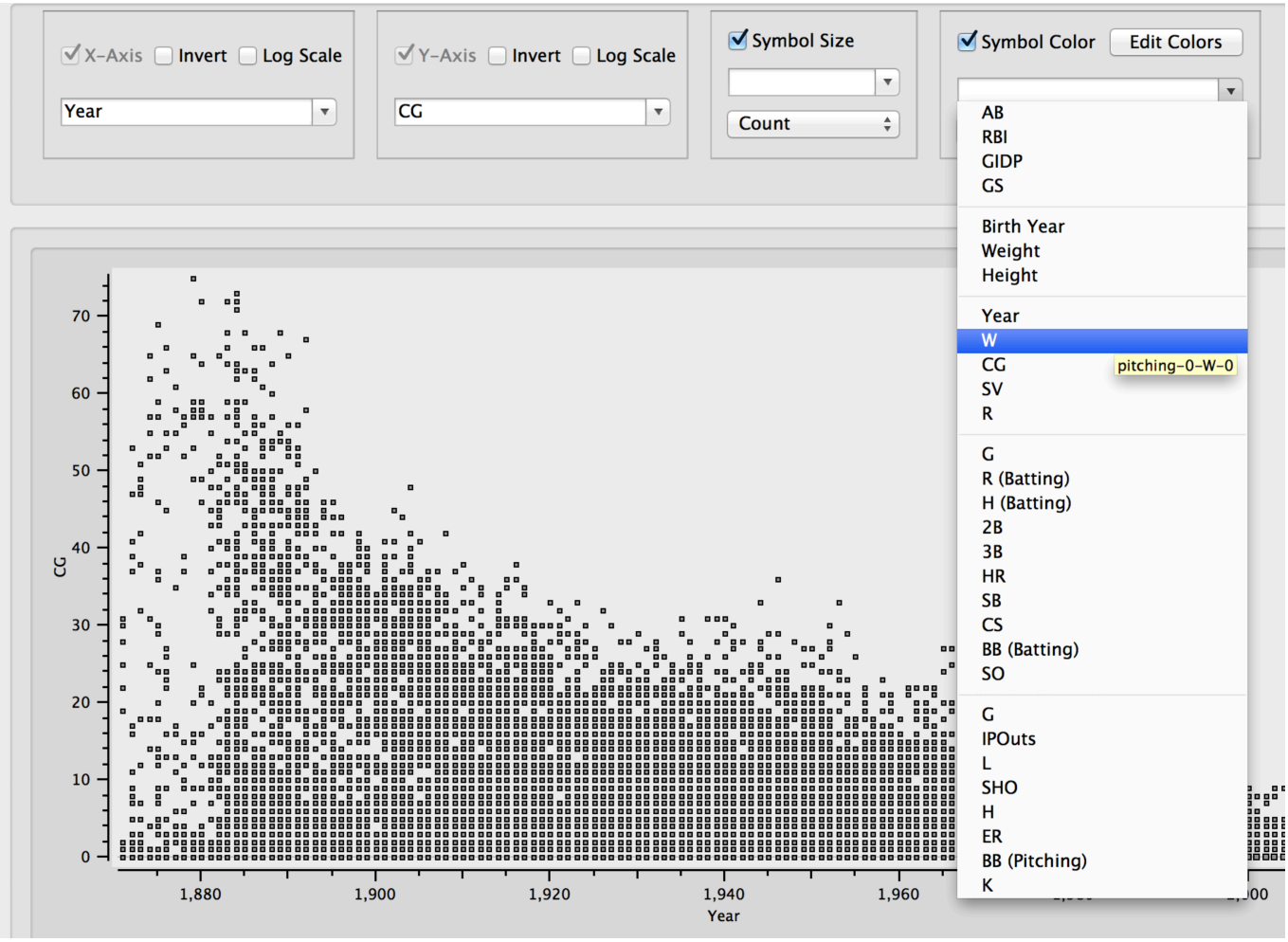


Figure 4. Plot slot detail view. Different slots have different options, and variable selection dropdowns are populated only with appropriate types.

one variable affect the distribution of data in another: as one might expect, lightweight players born in the 19th century are relatively short. Furthermore, the user is prevented from making unproductive UI choices: there are no switch hitting players left in scope, so the *B* value in the Bats filter is disabled.

The *Split* operator—available on nominal filters—compares data by partitioning it on the filter values. Data views are responsible for visualizing the comparison as appropriate for each view type (see Figure 3 and the following section).

3.4 Data Views

A *data view* is a fundamentally a method of mapping data to the characteristics of a visual *glyph*. We term these characteristics *slots,* similar to the Polaris/Tableau *shelf* concept). Examples of glyph slots are the two position dimensions of a mark on an X/Y Cartesian coordinate system, a mark's color, size, orientation, etc. Thus, a data view is described by how glyph characteristics are used in the data mapping, and a user creates a specific view instantiation by choosing what actual data to map to those slots. Lytic is currently limited to a single view at a time, and there are currently no brushing operations from the view to filter widgets. The lack of brushing operations is due to computational obstacles: recent work shows promise for scalable brushing using precomputed "data tiles", but even so scales only to modest numbers of dimensions [21].

All data views reflect current filters and splits. Views only display data within the current filter scope, and any split comparisons are handled in a view-appropriate manner: line charts use color to denote different split values within the same chart (cf. Figure 3, top), while scatter plots display multiple instances of the view with the distinct data partitions (cf. Figure 3, bottom). The latter allows the easy construction of small multiple views of the dataset. There are currently five major view types implemented: scatterplots, line charts, bar charts, parallel coordinates [16], and a tabular data view, with polar variants of the scatter and line charts.

For example, the Lytic scatterplot view can map data to the glyph X/Y position, color and size; a user defines a specific scatter plot view by selecting variables from her dataset to map to each slot: time to the X-axis; distance to the Y-axis, etc. Other views have fewer (e.g., line chart has only X and Y axis slots) or different slots (the parallel coordinates view has a dynamic number of slots, all for parallel Y axes).

Different slots have different properties: scalar-only slots can be inverted or log-scaled; some have custom transformations (e.g., color mapping); some views have dynamic slots that can be added or removed. All data views list what slots are used at the top of the visualization area, and within each slot a drop-down list of variables is available to choose which to map to that slot. Thus, to

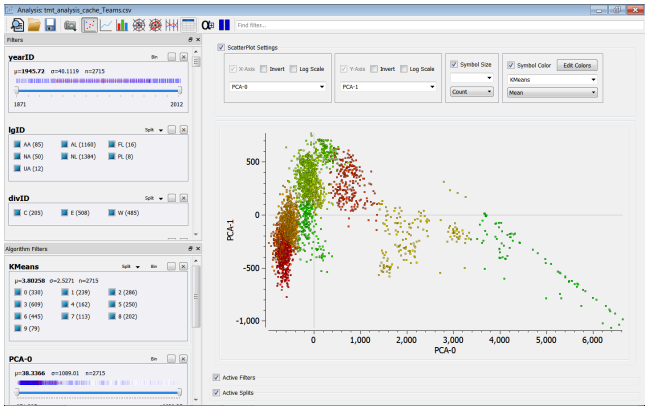


Figure 6. PCA visualization of baseball team statistics.

create a new visualization, the user simply selects the desired variable from the list, and the view updates with the new data.

Only valid selections are shown or allowed from the view drop-down boxes: variables that are not the appropriate data type for a slot (e.g., nominal variables for a numeric axis) are not included. Some slots are *aggregate*, meaning the view will combine multiple data points (grouping by the same value the non-aggregate slots in that view); for those slots a second drop-down box is shown with the available aggregate functions (e.g., mean, standard deviation, etc.).

In Figure 4, the user has already selected to visualize Year vs. CG (Complete Games), and is preparing to make a selection for the Color slot. Variables such as Team or League are not present since they are nominal. As the user makes selections in the slots, the view updates immediately; the current view shows that over time, the number of pitcher Complete Games has decreased dramatically. If the user selects W (Wins) from the dropdown, she could investigate if the number of Complete Games is correlated with Wins in some way.

3.5 Data Transformation

Other analysis capabilities include data transformations via automated unit conversions, creating new computed variables, and custom analysis via a simple plugin API. Unit conversions are specified in an XML file containing relevant units and conversion factors. In the data ingest step, the user can specify units for a variable, and any valid conversions are automatically presented within the filter widgets. Users can also set global preferences for metric or imperial unit systems, and any non-conforming units will be transformed.

New computed variables can be created by the Bin dialog (which bins scalar variables per user input) or in the more general case, arbitrary SQL formulas using existing variables (this is the only Lytic feature that directly exposes the SQL data model underpinnings).

Lytic can be configured to launch external programs ("data handlers"), passing any subset of the current data scope (as a file) for customized analysis. Data is selected from the active view (e.g., lasso-selecting scatterplot points, or selecting rows in the tabular view), and a right-click context menu shows all available handlers. A built-in default shows the entire record for each selected data item, but generally the fields can be processed for any purpose. For example, a handler could read the contents of a file path that contains 3D scene data and render it, or could implement a customized statistical analysis with an R script.

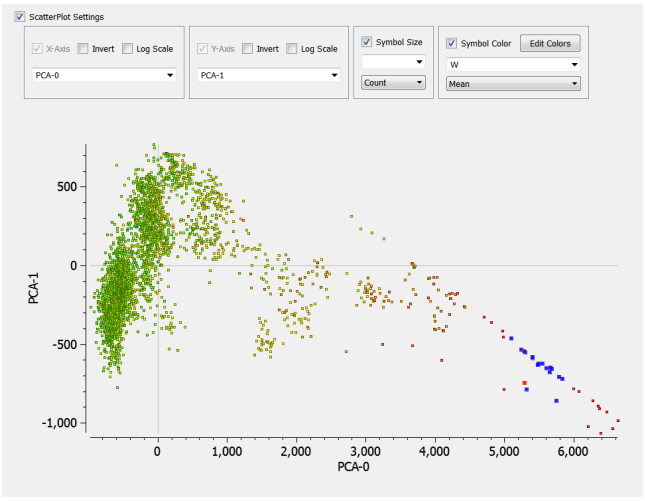


Figure 5. The same view as Figure 6, but with color mapped to Wins instead of cluster label. The user has selected a group of items in the lower left (bold blue items).

4. USAGE SCENARIO

To illustrate Lytic's end-to-end utility, we provide a high-level usage scenario prior to providing more details on system design and implementation. Sports data analysis is an interesting domain for visual analytics that has seen some recent attention [23], providing motivated users (fans, teams, and media) and a rich set of qualitative and quantitative data.

4.1 Narrative

Consider the perspective of a writer, Sally, doing research for a book on the general history and evolution of the game of baseball. She is aware of the Lahman Baseball Archive[3], which contains pitching, batting, fielding, managerial, and other statistics from U.S. professional baseball.

Sally starts Lytic and selects the Teams CSV file from the several dozen different collations in the archive, which she assumes has summary statistics from each team season. Aside from general research, she is interested in seeing if a machine learning tool can suggest any interesting new ways of categorizing or looking at historical teams, so she also adds a K-Means clustering and a PCA (principal component analysis) dimension reduction algorithm in the load dialog using the default parameters (10 clusters and 2 dimensions, respectively).

Lytic processes the data and presents the initial view of the team

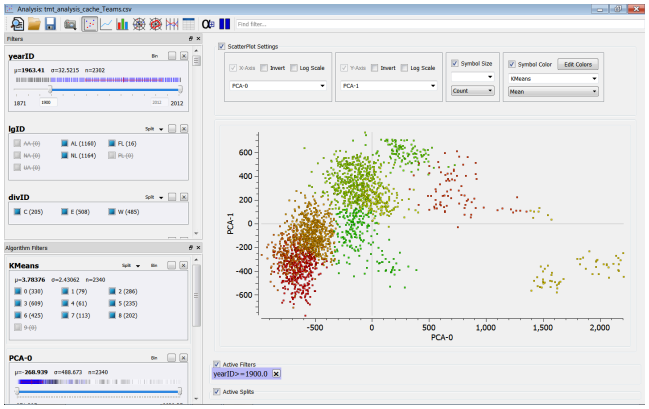


Figure 7. Same view as Figure 6 and Figure 5, but with a filter in place for seasons 1900 and after.

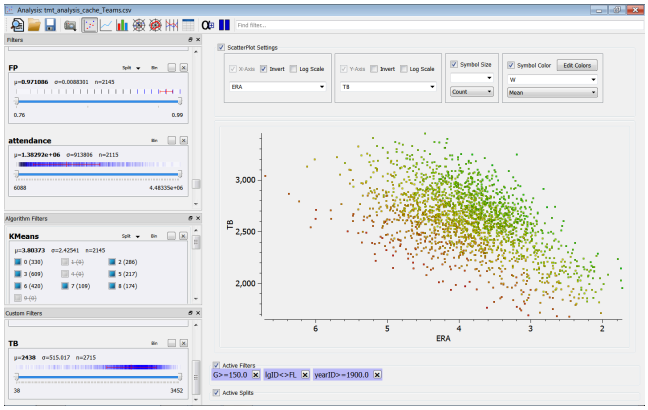


Figure 8. Scatterplot view of Earned Run Average (ERA) vs. Total Bases, with color mapped to Wins, showing the tradeoff between offense and defense and the advantage of being near the Pareto front.

data. Sally is first interested in looking at what makes a successful baseball team, but she first takes stock of the various filters built from the file. She first notices that there is data dating from 1871, when professional baseball was in its infancy, and that there is also data from older defunct professional leagues. There are also filters for the K-Means clusters and PCA dimensions. That kind of data is new to her, so she decides to see what it looks like using the default scatterplot view: she selects the PCA-0 and PCA-1 variables in the X- and Y-axis slots, and selects the K-Means variable in the color (see Figure 5).

She is intrigued by the arrangement of the teams in the plot, and wonders whether the algorithm results have any bearing on team quality or success, so she changes the color slot to Wins (see Figure 6). This seems to have some relationship with the reduced space, primarily in the lower right that all appear to be low-win teams (the default color mapping is a stoplight pattern). She lasso-selects several points in the lower-right section, and right-clicks to examine their details. She finds that they are all from the 19th century, so she immediately goes to confirm her guess by changing the color slot to Year.

That appears to confirm her suspicion, so she filters out all seasons prior to 1900 (see Figure 7). Doing so dynamically filters the visualization, and when she releases the year slider it updates the other filters—whose distribution is visible by default—indicating that (as she already knew) almost all the historical professional leagues were gone by 1900, and more importantly

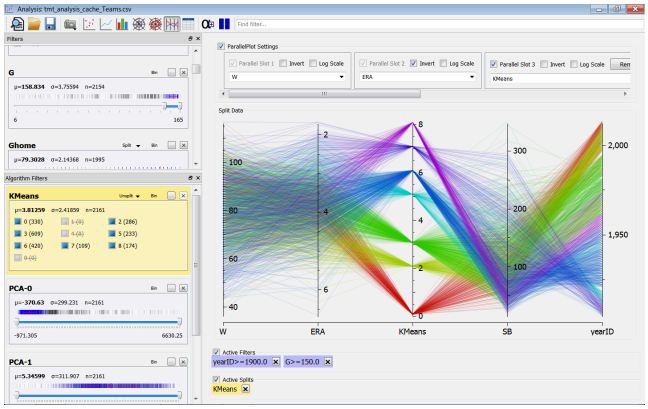


Figure 9. A parallel coordinates view of several different variables. Color is mapped to cluster label.

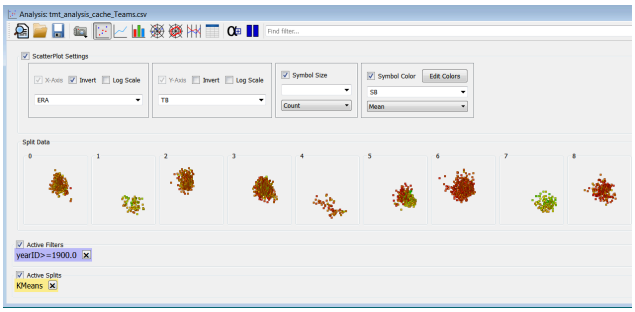


Figure 10. A small multiples scatterplot view of ERA vs. Total Bases (cf. Figure 8), but with color mapped to Stolen Bases and the view split on cluster value. This shows two clusters (#1, 2nd from left; #7, 2nd from right) that show similar teams having high stolen base totals, low ERAs and low Total Bases.

shows that one of the clusters has been removed from the focus data.

Sally then reverts to her original line of inquiry: do the clusters have any relationship to team quality as expressed by Wins? She maps color to Wins—there is still no obvious relationship on the high end, but again the clusters on the lower right are consistently on the lower end. She again lasso-selects and examines their details, finding that one sub-group are teams from 1994 and one from 1981. Sally might recognize these seasons as ones impacted by player strikes; otherwise, this would clearly prompt additional investigation outside Lytic.

This prompts Sally to wonder how much of the PCA components and cluster values are related to the number of games played in the season—she switches the color slot to G, and finds that a great deal of the first component is due to the number of games in the season; this leads her to wonder if the algorithmic data would be better applied to rate statistics (e.g., attendance/game, adjusted for population). She decides to investigate a few more items and leave the re-processing for later.

Still interested in team success, she uses the scatterplot view to look at team wins versus several statistics and finds some interesting tradeoffs between offensive variables (e.g., Home Runs and Triples are slightly negatively correlated). This causes her to wonder about overall team construction: is there a tradeoff between offense and defense? To test this, she creates a custom variable, Total Bases, which is calculated from Home Runs, Triples, Doubles and Hits; she plots total bases versus Earned Run Average (a measure of pitching/defensive effectiveness), and sets the color to wins (see Figure 8). Interestingly, there does appear to be a tradeoff at work—the more offense a team has, the worse its defense, and that the better teams are closer to the Pareto front. Sally realizes that teams are constructed under resource constraints (player supply, salaries), so this makes some sense

Piqued by this realization, she then uses the parallel coordinates view to look at several variables at once (see Figure 9). She happens to place the GHome (home games) variable next to the year, finding that there are no teams from 2006-2010 with any recorded home games. She suspects this must be a data collection problem, since she is certain baseball was played during those years.

She eventually backtracks to looking at the K-Means cluster data (also splitting on that variable to color-code the view), and notices a few clusters do have an association with low Stolen Base totals. That leads her to wonder: is the offensive/defensive tradeoff (as

measured by Total Base) mitigated by Stolen Bases, which are another way to generate offense?

She returns to the scatterplot view of ERA vs. Total Bases, but this time colors nodes by Stolen Bases and splits the view on the cluster value, creating a small multiples-style view (see Figure 10). She is excited to see that some of the clusters *do* have some additional meaning outside of the number of games or the year: two of the clusters appear to represent teams that steal a lot of bases but otherwise are successful due to better pitching.

4.2 Discussion

Sally demonstrated several benefits of visual analytics as supported by Lytic. First, outlier detection is a common task (either intended or serendipitous), and in the context of end-user data analysis, often takes the form of investigating data quality and provenance. If a data point is anomalous, in different contexts it is often the case that the data collection is faulty rather than encountering a truly novel case—a useful outcome in and of itself. Conversely, outliers that are really novel are the canonical case of sidetracking for opportunistic insight, regardless of the original goal or lack thereof.

The benefits of combining a general purpose VA tool with even basic DR/C algorithms are also significant, simply because their data provide a new jumping-off point for further investigation, potentially leading to insights only a human might not have considered.

A few points of Sally's story also pointed out areas for potential improvement. Sally is a writer or journalist, with no background in algorithms—as a result, it is somewhat of a stretch to assume that she would be asking for "PCA" or "K-Means" "algorithms", much less perform even basic configuration, such as selecting an appropriate *k* value. Revising the end-user nomenclature—e.g., *data map* for 2-dimensional DR results; *automatic categories* for cluster results—may be beneficial for some user audiences, as well as simplifying or eliminating configuration options. Conversely, if we have a user-friendly method of employing DR/C algorithms, it would be useful to have inline processing to either include new or exclude spurious features from within the analytic environment: in Sally's case, including more rate statistics or remove year as features.

Finally, although the scenario did not address truly high-dimensional data such as those from a document term frequency matrix, DR is crucial to enabling a manageable navigation environment for those types of datasets.

5. LIMITATIONS AND FUTURE WORK

Lytic has a number of general limitations that we should acknowledge and are implicit future work. Hierarchical facets and similar data—particularly that which can be automatically generated, such as a hierarchy of time intervals or geographic location granularities—are a key omission from the current tool. Similarly, Lytic has no first-class notion of either time or location data types (e.g., Date/Time or latitude/longitude information), and the current view have more options for numeric vs. categorical data. Using Lytic to monitor streaming data would be useful as well, but there is no current support for updating the data model once it is built.

Working on Lytic has also convinced us of one of the conclusions of a recent survey of data analysts [18]: that "data wrangling" is a significant barrier to making full use of tools like Lytic, and that more work [17] is necessary to solve end-to-end visual analytic problems.

Mixed-initiative visual analytic tools that respond to users previous interaction with the same or similar datasets. For example, learning from user's data ingest choices to make better default selections in the future. Finally, we hope to make Lytic a more open tool—including a visualization plugin system—for the visual analytic community as both a research and practitioner platform.

6. ACKNOWLEDGMENTS

The work of these authors was supported in part by the National Science Foundation grant CCF-0808863. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

1. Ahlberg, C. Spotfire: an information exploration environment. *SIGMOD Rec. 25*, 4 (1996), 25–29.
2. Aiken, A., Chen, J., Stonebraker, M., and Woodruff, A. Tioga-2: A Direct Manipulation Database Visualization Environment. (1996), 208–217.
3. Bae, S.-H., Choi, J.Y., Qiu, J., and Fox, G.C. Dimension reduction and visualization of large high-dimensional data via interpolation. *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ACM (2010), 203–214.
4. Becker, R.A. and Cleveland, W.S. Brushing Scatterplots. *Technometrics 29*, 2 (1987), 127.
5. Capra, R.G. and Marchionini, G. The relation browser tool for faceted exploratory search. *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, (2008), 420–420.
6. Card, S.K., Robertson, G.G., and Mackinlay, J.D. The information visualizer, an information workspace. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (1991), 181–186.
7. Ceruzzi, P. and Grad, B. Guest Editors' Introduction: PC Software–Spreadsheets for Everyone. *IEEE Annals of the History of Computing 29*, 3 (2007), 4–5.
8. Chau, D.H., Kittur, A., Hong, J.I., and Faloutsos, C. Apolo: making sense of large network data by combining rich user interaction and machine learning. *Proceedings of the 2011 annual conference on Human factors in computing systems*, (2011), 167–176.
9. Choo, J., Lee, C., Clarkson, E., et al. VisIRR: Interactive Visual Information Retrieval and Recommendation for Large-scale Document Data. *In submission to IEEE Transactions on Visualization and Computer Graphics (TVCG)*, .
10. Choo, J., Lee, H., Liu, Z., Stasko, J., and Park, H. An interactive visual testbed system for dimension reduction and clustering of large-scale high-dimensional data. *Proceedings SPIE 8654, Visualization and Data Analysis*, (2013).
11. Clarkson, E., Hurt, J., Zutty, J., Skeels, C., Parise, B., and Rohling, G. Supporting robust system analysis with the test matrix tool framework. *Proceedings of the 2013 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ACM (2013), 23–34.
12. Clarkson, E., Navathe, S.B., and Foley, J.D. Generalized formal models for faceted user interfaces. (2009), 125–134.
13. Draper, G.M. and Riesenfeld, R.F. Who votes for what? A visual query language for opinion data. *IEEE transactions on visualization and computer graphics 14*, 6 (2008), 1197–1204.
14. Fekete, J.-D., Wijk, J.J., Stasko, J.T., and North, C. The Value of Information Visualization. In A. Kerren, J.T. Stasko, J.-D.

Fekete and C. North, eds., *Information Visualization: Human-Centered Issues and Perspectives*. Springer-Verlag, Berlin, Heidelberg, 2008, 1–18.
15. Huser, V., Narus, S.P., and Rocha, R.A. Evaluation of a flowchart-based EHR query system: a case study of RetroGuide. *Journal of biomedical informatics 43*, 1 (2010), 41–50.
16. Inselberg, A. The plane with parallel coordinates. *The Visual Computer 1*, 2 (1985), 69–91.
17. Kandel, S., Paepcke, A., Hellerstein, J., and Heer, J. Wrangler: interactive visual specification of data transformation scripts. *Proceedings of the 2011 Conference on Human Factors in Computing Systems (CHI)*, ACM (2011), 3363–3372.
18. Kandel, S., Paepcke, A., Hellerstein, J.M., and Heer, J. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2917–2926.
19. Kimball, R. A dimensional modeling manifesto. *DBMS 10*, 9 (1997), 58–70.
20. Krishnamurthy, R. and Zloof, M. RBE: Rendering by example. *Proceedings of the Eleventh International Conference on Data Engineering, 1995*, (1995), 288–297.
21. Liu, Z., Jiang, B., and Heer, J. imMens: Real-time Visual Querying of Big Data. *Computer Graphics Forum 32*, 3pt4 (2013), 421–430.
22. Marchionini, G. and Shneiderman, B. Finding Facts vs. Browsing Knowledge in Hypertext Systems. *Computer 21*, 1 (1988), 70–80.
23. Pileggi, H., Stolper, C.D., Boyle, J.M., and Stasko, J.T. SnapShot: Visualization to Propel Ice Hockey Analytics. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2819–2828.
24. Ranganathan, S. *Colon Classification*. Madras Library Association, Madras, 1933.
25. Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. *Visual Languages, 1996. Proceedings., IEEE Symposium on*, (1996), 336–343.
26. Spence, R. and Tweedie, L. The Attribute Explorer: information synthesis via exploration. *Interacting with Computers 11*, 2 (1998), 137–146.
27. Stasko, J., Görg, C., and Liu, Z. Jigsaw: supporting investigative analysis through interactive visualization. *Information Visualization 7*, 2 (2008), 118–132.
28. Stolte, C., Tang, D., and Hanrahan, P. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 52–65.
29. Thomas, J.J., Cowley, P.J., Kuchar, O., Nowell, L.T., Thompson, J., and Wong, P.C. Discovering knowledge through visual analysis. *Journal of Universal Computer Science 7*, 6 (2001), 517–529.
30. Wilson, M., Russell, A., and Smith, D.A. mSpace: improving information access to multimedia domains with multimodal exploratory search. *Communications of the ACM 49*, 4 (2006), 47–49.
31. Yee, K.-P., Swearingen, K., Li, K., and Hearst, M. Faceted metadata for image search and browsing. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2003), 401–408.
32. Yi, J.S., Kang, Y., Stasko, J.T., and Jacko, J.A. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1224–1231.
33. Zloof, M.M. Query-by-example - Operations on hierarchical data bases. *Managing Requirements Knowledge, International Workshop on*, IEEE Computer Society (1976), 845.