An Interactive Visualization Environment for Data Exploration

Mark Derthick and **John Kolojejchick** and **Steven F. Roth**
{mad+, jake+, roth+}@cs.cmu.edu
Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213

Abstract

Exploratory data analysis is a process of sifting through data in search of interesting information or patterns. Analysts' current tools for exploring data include database management systems, statistical analysis packages, data mining tools, visualization tools, and report generators. Since the exploration process seeks the unexpected in a data-driven manner, it is crucial that these tools are seamlessly integrated so analysts can flexibly select and compose tools to use at each stage of analysis. Few systems have integrated all these capabilities either architecturally or at the user interface level. Visage's information-centric approach allows coordination among multiple application user interfaces. It uses an architecture that keeps track of the mapping of visual objects to information in shared databases. One result is the ability to perform direct manipulation operations such as drag-and-drop transfer of data among applications. This paper describes Visage's Visual Query Language and visualization tools, and illustrates their application to several stages of the exploration process: creating the target dataset, data cleaning and preprocessing, data reduction and projection, and visualization of the reduced data. Unlike previous integrated KDD systems' interfaces, direct manipulation is used pervasively, and the visualizations are more diverse and can be customized automatically as needed. Coordination among all interface objects simplifies iterative modification of decisions at any stage.

Keywords Data Visualization, Interactive Data Exploration, Visual Query Language, Automatic Presentation

Introduction[1][2]

Exploratory data analysis has been called *data archaeology* (Brachman *et al.* 1993) to emphasize that it requires a human analyst tightly in the loop who has high level goals in seeking useful information from the data. For goals such as "find buying patterns we can use to increase sales" it is impossible to formally characterize what constitutes an answer. A surprising observation may lead to new goals, which lead to new surprises, and in turn more goals. To facilitate this kind of iterative exploration, we would like to provide the analyst rapid, incremental, and reversible operations giving continuous visual feedback.

In some ways, the state of the art 30 years ago was closer to this goal of a tight loop between exploration, hypothesis formation, and hypothesis testing than it is today. In the era of slide rules and graph paper, the analyst's forced intimacy with the data bred strong intuitions. As computer power and data sets have grown, analysis retaining the human focus has become on-line analytical processing (OLAP). In contrast, data mining has emerged as a distinct interdisciplinary field including statistics, machine learning and AI, and database technology, but that does not include human-computer interaction (HCI). KDD recognizes the broader context in which data mining systems are used, and that the majority of the user's effort is in preparatory and evaluatory steps. HCI is thus an important part of an end-to-end KDD system(Fayyad, Piatetsky-Shapiro, & Smyth 1996).

Visage provides a scripted rapid-prototyping environment for experimenting with interaction and visualization techniques. Unfortunately, the overhead of interpretation and first-class graphical objects limits the dataset size that can be manipulated rapidly. What we propose here are ways to attain initial intuitions about a large dataset from small samples; to create descriptions of the interesting subsets and attributes for input to cleaning and data mining applications; and to evaluate the discovered rules. We also offer a framework for how these distinct applications can be seamlessly integrated in a single information-centric interface.

A key innovation is the combination of extensional direct manipulation navigation with intensional query construction. The former allows fast opportunistic exploration of the sample, while the latter allows these operations to be carried out automatically on the full dataset. Below we survey the state of the art in interactive visualization systems and illustrate the use

[1]Copyright ©1997, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

[2]Color versions of the figures are at, eg, http://www.cs.cmu.edu/ sage/KDD97/figure1.gif

of Visage (Roth *et al.* 1997) on KDD tasks for a real-estate domain. We conclude by discussing the possibility of a fully integrated KDD system based on a system like Visage.

Interactive Visualization Systems

Previous Interactive Visualization Systems

The interactive aspects of previous visualization systems that we make use of affect how datapoints in the visualization can change their appearance based on attributes of the domain objects they represent. In *filtering* the points corresponding to houses in a certain neighborhood could be made invisible. In *brushing* (Becker & Cleveland 1987) the color of the points is changed. Datapoints can also be dragged and dropped to add or remove them from datasets, or to examine them in different tools. *Direct manipulation* interfaces are those in which operations are carried out by interacting with interface objects representing the semantic arguments of the operation. For instance, dragging an icon representing a file to a trash can is direct manipulation, while dialog boxes are not.

There are no previous systems that allow query construction by direct manipulation on domain-level data as well as schema-level data. This is largely because it is viewed as the job of the database management system (DBMS) to create a table with a given schema, and the visualization system's job to show all and only the domain-level data in the table. Visualizations are not viewed as full exploration interfaces themselves. Although some drill-down capability may be available, changing a query requires a distinct interface.

Further, each time a query is changed and re-run, a new table is sent to the visualization system, which won't know how to coordinate visualizations from the different tables even if the underlying conceptual objects are identical. For instance, if objects in one visualization are brushed/filtered it's difficult to identify the corresponding objects in other visualizations. The problem is compounded if the various visualizations originate from different database joins where the number of records differ.

Research is being done to increase the amount of exploration possible within a visualization via direct manipulation, such as brushing. Dynamic Query sliders (Ahlberg, Williamson, & Shneiderman 1992) filter interface objects representing data objects whose attribute value falls outside the range selected by the slider. They are called "queries" because they not only allow interactive exploration, but also make explicit the range of selection. Further, by including histograms on the sliders, useful feedback occurs even in the absence of visualizations displaying every data point, so larger datasets can be handled interactively (Tanin, Beigel, & Shneiderman 1996). Related work has addressed filtering hierarchical information (Kumar, Plaisant, & Shneiderman 1995) and the logical operators AND, OR, and NOT (Young & Shneiderman 1992). For KDD, where we want to reuse queries on different data, having an explicit query is very important. However these queries don't involve arithmetic functions of attributes and, more fundamentally, don't involve multiple objects and quantification, nor do they involve aggregation and aggregate functions.

There are some visual query languages, such as GQL (Papantonakis & King 1995), that can express these more complicated queries, but they are not integrated in a drag and drop manner with visualization systems. Thus the exploration loop requires the analyst to recall her sequence of exploration operations and reproduce them in the query language, rerun the query, and recreate any visualizations. The problem is much the same as with any other type of disconnected application used in the KDD process. With very few exceptions, the user is forced to manage the transmission of data from one application to another, increasing the overhead of exploration.

Visage

In addition to capabilities of previous interactive visualization systems, Visage also includes the Sage expert system (Roth *et al.* 1994) for automated design of sophisticated visualizations integrating many attributes. Visage is *information centric* (Roth *et al.* 1996), which means that data objects are represented as first class interface objects, which can be manipulated using a common set of basic operations, such as drill-down and roll-up, applicable regardless of where they appear: in a hierarchical table, a slide show, a map, or a query. Furthermore these graphical objects can be dragged across application boundaries. Allowing the visualization system direct access to the underlying database, rather than just isolated tables derived from it, is key to interface unification of visualization application with the other components of a KDD system.

A user can *navigate* from the visual representation of any database object to other related objects. For instance from a graphical object representing a real estate agent, all the houses listed by the agent could be found. It is also possible to *aggregate* database objects into a new object, which will have properties derived from its elements. For instance, we could aggregate the houses listed by John, and look up their average size. However these operations are only specified procedurally, that is by the sequence of direct manipulation operations – there is no explicit query. Once the user has navigated from John to his houses and aggregated them, he must repeat the operations to do the same for Jill's houses, or to repeat for John's houses next month when the data may have changed. Just as sliders visually indicate the current filtration range for parameterized queries, we would like a declarative and manipulable query representation that also applies to these structural operations. VQE is a visual query

environment within Visage where operations are represented explicitly, allowing the analyst to construct complex queries during the analysis process for later reuse.

Example

Schema Browsing

To illustrate the system we use a fictitious database of house sales based on data collected by a group of Pittsburgh real-estate agents describing clients and sales information for three neighborhoods for 1989. Figure 1(left) shows an entity-relationship (ER) diagram of the database. Data types are shown as rectangular nodes, and attributes of objects are listed inside the node. Relations between objects are shown as links.[3] The substructure of the links is used to show functional dependencies. Links with three diverging lines indicate one-many relations; those with five lines diverging in both directions indicate many-many relations. A link with a single line would indicate a one-one relation. For instance, the way we have modeled the domain, only one person can be the buyer agent in a given sale, although many sales can have the same person as the buyer agent. The role served by a relation for an object is labeled next to the object. For instance, the relation between sales and houses serves the house-sold role of the sale, while it is the sold-in role of the house.

The remainder of the example, the query proper, illustrates binding multiple objects so that their properties may be visualized together, selecting those properties to be visualized and the method by which they are visualized, filtering objects based on their properties or on the properties of other objects, coordination among visualizations derived from distinct queries, and definition of new properties on the fly.

Data Cleaning

We begin by randomly selecting 100 houses from the database to scan for obvious problems. Figure 1(right) shows a copy of the House data type being dragged from the ER diagram into the VQE frame,[4] where it becomes a *dynamic aggregate*, so called for reasons to be explained below. The dynamic aggregate represents a set of houses, and serves as a locus for browsing and querying operations on the elements of the set.

First the analyst examines the attributes of the houses. Histograms of attribute values are so useful that Visage includes a pre-built visualization for this purpose that also includes a slider to control selection. In the figure, histogram sliders have been dropped on all the attributes except address. Often these displays show peaks corresponding to bad data, for instance a peak at zero indicating missing data. Here, however, the only surprise is the uneven distribution of latitude and longitude. Given the analyst's domain knowledge that the three neighborhoods are adjacent and compact, a nearly uniform distribution is expected. To examine the house locations in more detail, the analyst wants to visualize them on a map, color coding by neighborhood. She uses the SageBrush tool (Roth *et al.* 1994) to sketch the desired visualization (figure 2, upper left). The Create Picture button sends a design request to the Sage expert system, which returns an instantiated picture (figure 2, right). (In the absence of a sketch, or with a partial sketch, Sage uses heuristics to design pictures to facilitate specified analysis tasks on specified attributes.) Except for a few outliers, the agent recognizes that the colors of the graphical representation of the houses match her knowledge of the location of the neighborhoods. In figure 2(lower left) the analyst has dragged a copy of one of the outliers to a table, to drill down textually to see what's wrong. The house's address, 7516 Carriage, is recognized to be a Point Breeze address, so why is the graphical mark misplaced? Zooming into the map shows that the house is indeed on Carriage street. It turns out that there are two Carriage streets in Pittsburgh, and the address-to-location server randomly chose the wrong one. The other outliers have the same problem. The analyst adjusts the latitude and longitude sliders to filter them out. Figure 3 shows that the outliers have disappeared from the map, the cardinality of the filtered set (96) has been updated in the header of the House dynamic aggregate, and the filtered (conditional) distribution of the other attributes is shown with the dark histograms. The unconditioned histogram remains as a light backdrop. All these updates happen continuously as the slider is moved, making it easy to find good cutoff points.

Similar examination of sales and people would be done next, but for the sake of brevity we skip them in this paper.

Creating the Dataset

In our scenario, the analyst is interested in the effects of market timing on buyers' decisions. That is, she is interested in seeing the relationships among attributes of sales (eg selling-price), attributes of people (eg age), and attributes of houses (eg neighborhoods). To make this possible, she must first specify which houses go with which sales, and which people go with which sales. In the navigation paradigm, where exploration is done by following links between related objects, she would select a particular house, use a menu to traverse to a particular sale, and then traverse to a particular buyer. VQE represents this schematically (figure 4) with a directed graph structure, where the nodes are dynamic aggregates representing each prototypical object and the links are the traversals. The direction of the arrow indicates the direction of navigation, and

[3] The knowledge representation system we use has only binary relations; in general relations in ER diagrams are more complicated.

[4] Visage objects which can contain other objects are called "frames."

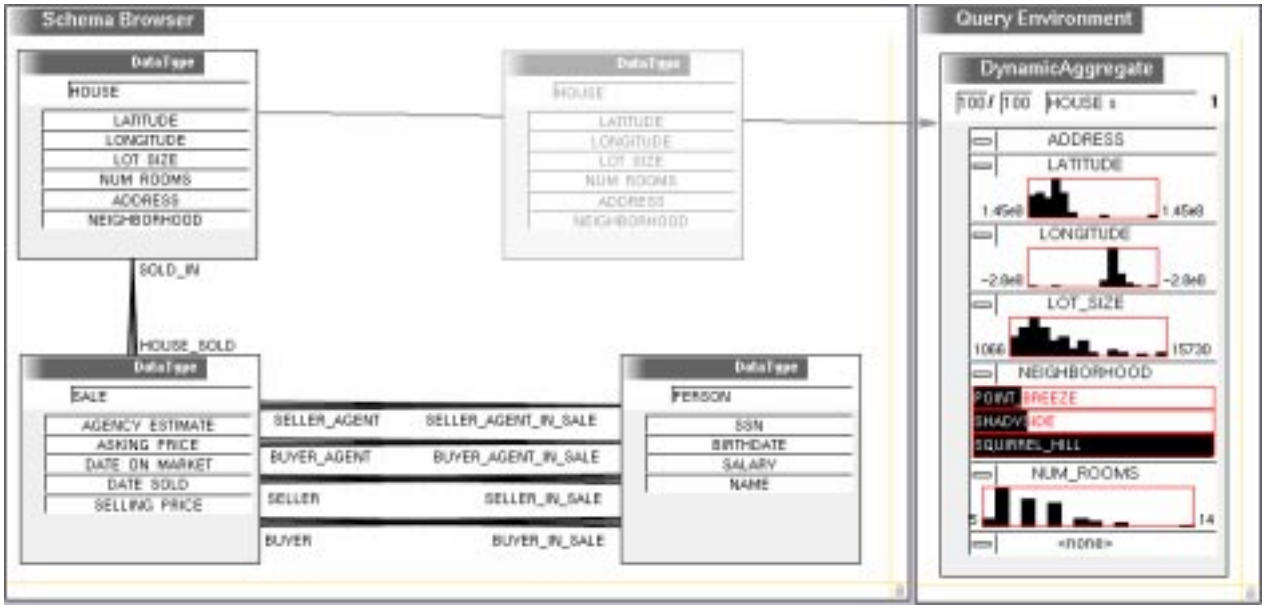


Figure 1: Left: Entity-Relationship diagram for the Real Estate domain. Right: The House data type has been dragged to QueryEnvironment (VQE), which creates a dynamic aggregate containing all (100) houses in the dataset being explored. Histogram sliders have also been dropped on most attributes, but no filtering has been done, so all 100/100 houses are selected.

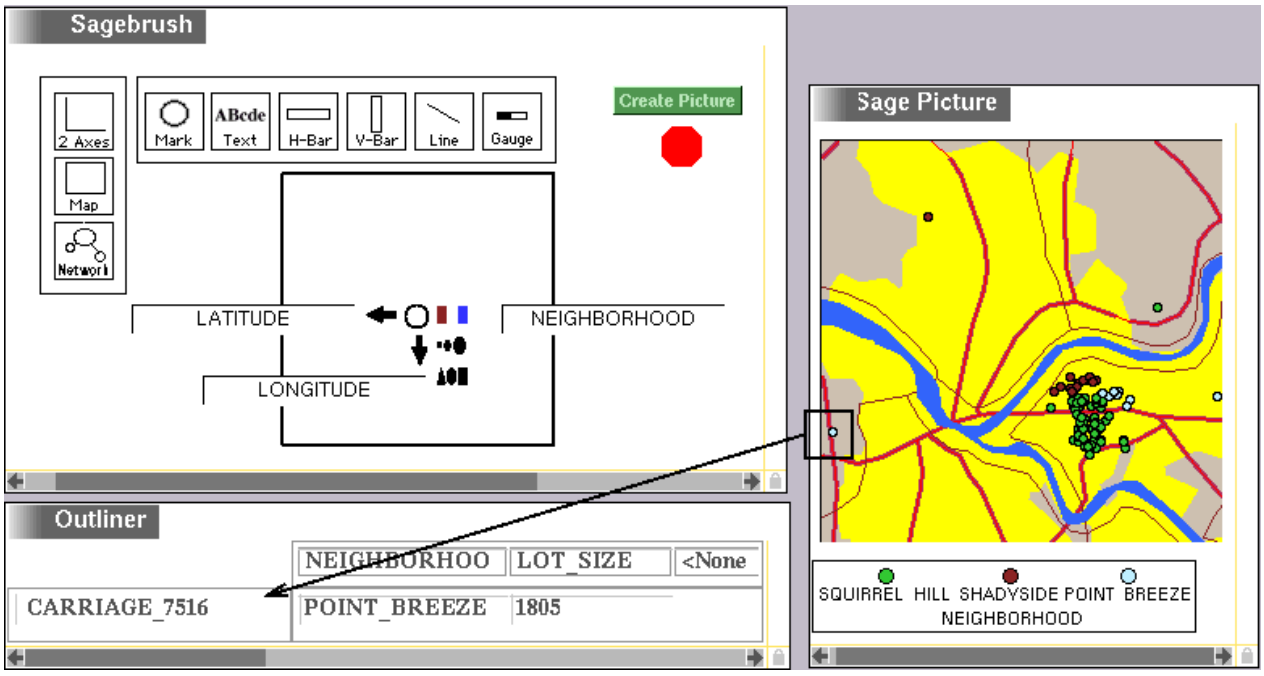


Figure 2: Upper Left: SageBrush sketch for creating a map. The big square icon representing a map was dragged from the Map icon at left; the round mark icon was dragged from the Mark icon at top. Attributes from the House dynamic aggregate were dropped on the mark's property icons. The Create Picture button then instantiated the SagePicture in the frame on the right. Lower Left: One of the outliers on the map was copy-dragged to an outliner to examine its address.

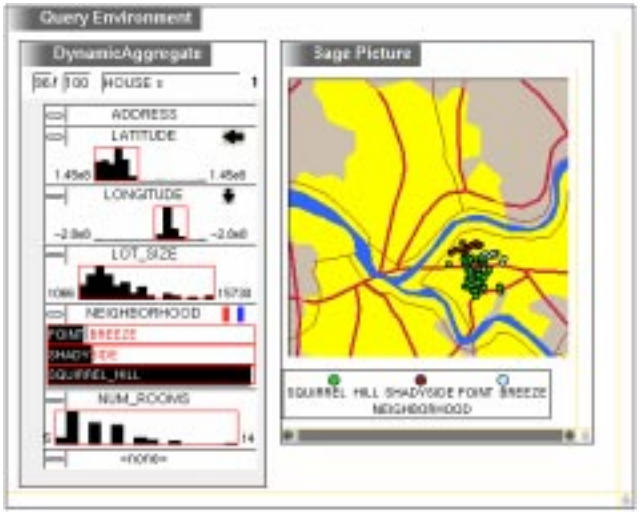


Figure 3: By moving the latitude and longitude sliders, the outliers have been deselected.

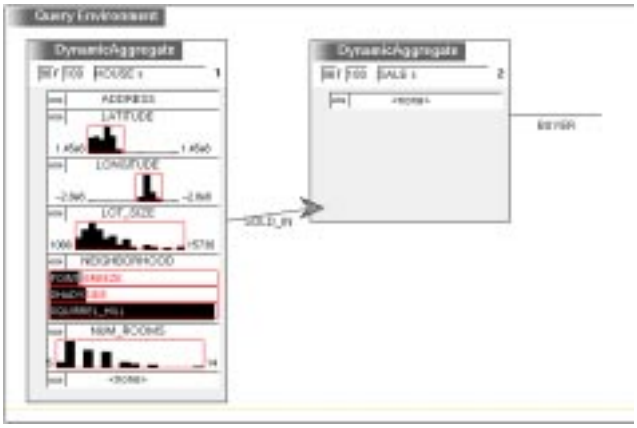


Figure 4: The analyst has already performed parallel navigation from the House dynamic aggregate to a Sale dynamic aggregate, and is now dragging a navigation arrow to a spot where a third dynamic aggregate will be placed.

the direction-appropriate label is chosen (eg "sold-in" rather than "house-sold"). Navigation operations on dynamic aggregates use interface actions similar to ordinary navigation in Visage, and we call it "parallel navigation," because the result node is the aggregate of the nodes reached by navigation from *each* constituent of the origin aggregate. Thus in figure 4 the analyst has navigated in parallel from 96 houses to 96 sales, and is about to navigate to the (126) buyers in those 96 sales.

Reusing the specification

During exploration the dynamic aggregates are most useful as extensional sets. However the graph structure tracing navigation paths forms an intensional description of a query. The nodes in the graph are analogous to relational calculus variables whose values are records, or to variables in an object-oriented query language. The extensional set represented by a dynamic aggregate is the set of values that the corresponding variable can be consistently bound to. The name *dynamic aggregate* indicates that it represents this aggregate set, whose extension is dynamically determined both by the graph structure and by the data objects that have been dropped onto it or dragged off of it. The intensional aspect allows reuse. All the houses could be dragged out of their dynamic aggregate, leaving an empty query and visualizations. The sliders retain their numeric range restriction. New data could then be dropped on any of the nodes or visualizations, repopulating the entire query. Useful queries can be stored in libraries, as can individual visualizations. The latter can even be customized automatically to match data from different schemata (Chuah *et al.* 1995). Below we suggest how queries and results might be passed to a data mining or other processing application.

Data Reduction and Projection

Data mining algorithms may be swamped by the entire dataset and attribute set. Usually some focus is needed, and often patterns are more apparent if the data is redescribed in terms of derived attributes. The analyst can find interesting subsets of data to mine with exploratory visualizations. Figure 5 shows a visualization the user has sketched that integrates a number of attributes of interest. The y-axis of both interval bar charts represents sales, and both are sorted by date-sold. The left chart shows the interval during which the house was on the market, with color encoding neighborhood. This data was collected for sales that occurred during 1989, so the right edge of each bar falls in this interval. The left edges of the bars indicate that the houses went on the market as early as early 1988. The right chart shows the interval between the asking price and the selling price of the house. By looking for correlations among these five variables, and by manipulating sliders on additional variables, the analyst can quickly identify obvious patterns in the data. Some of these will be of interest, while others are not.

Note that the map, which was designed before the House dynamic aggregate was connected to the Sale and Person dynamic aggregates, is coordinated with the charts. Coordination happens with sliders, drag and drop, and brushing. Figure 5 illustrates brushing, as the houses sold before March 1989 were brushed with a bounding box in the right chart, and the corresponding sales and houses are colored black on the left chart and on the map. The same interface objects can be shared across stages of the discovery process, and in fact the demarcation between stages is blurred. The extreme case is when we change the dataset. For

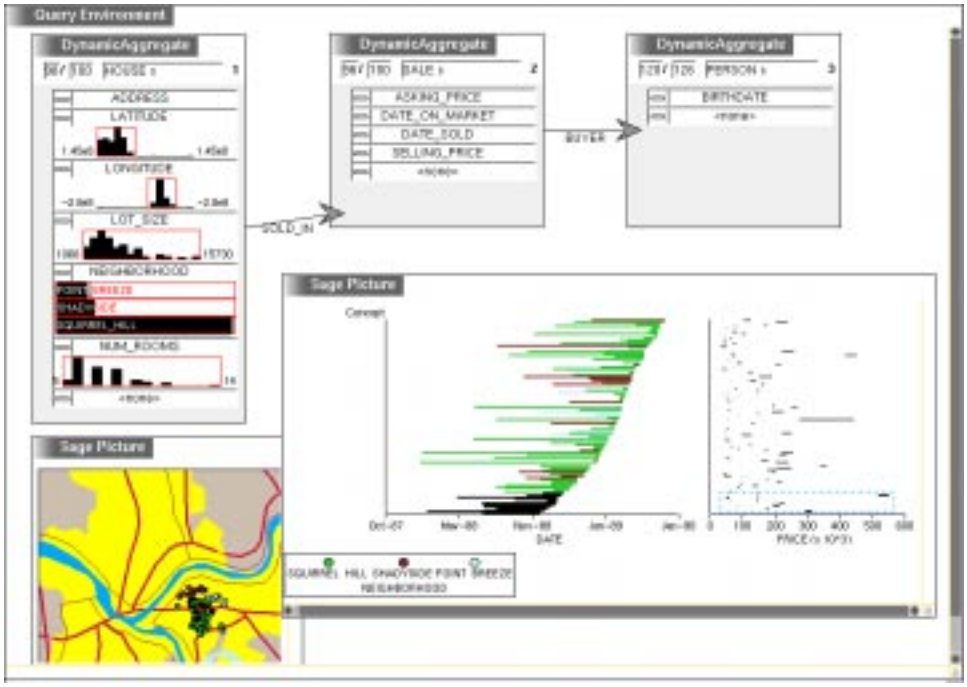


Figure 5: Visualization incorporating attributes from multiple objects. Using a bounding box (dashes), the analyst has *brushed* some bars in the right chart black. The corresponding bars in the left chart, and the marks on the map, also become black.

instance the analyst could switch from looking at buyers in these sales to looking at sellers as follows: drag the Person dynamic aggregate to the trash, leaving a 2-node graph just as in figure 4, and then navigating again from sale but this time across the seller relation. Dropping a slider on the sellers birthdate attribute would then control visibility of houses on the map and sales in the existing charts, but now for houses that were *sold* by the person, rather than houses that were *bought* by the person.

Obviously, Squirrel Hill sales dominate the dataset. This is apparent from the histogram, the map, and the bar colors. However the map shows that it is also the largest neighborhood geographically, so this pattern is really not surprising. As a fraction of all houses in the neighborhood, the number sold in Squirrel Hill may not differ from that in the other neighborhoods.

Another interesting feature is the shape of the curve formed by the right ends of the date bars. It is steep from about May 1989 to October 1989, and flatter during the winters (January-March 1989 and November-December 1990). Again, this is a very familiar pattern to real estate agents.

Potentially of more interest are the bottom few and top few rows of the price interval chart. While low price houses dominate overall, and are pretty evenly distributed vertically, there are fewer at the extremes, during the winter slowdowns. Moving the selling-price slider to show only expensive houses (not shown), this impression is confirmed, as more bars disappear from the middle of the charts, while the top and bottom extremes remain more dense with bars. By moving sliders to focus on various subsets of the data, the analyst finds that many variables are correlated with selling price; expensive houses take longer to sell, they seem to drop more in price (even as a percentage), they are exclusively in Squirrel Hill (in the northern half of Squirrel Hill in fact), and they are bought by older buyers. In fact, while looking only at older buyers, there is an even more pronounced increase in density at the date extremes than with high-priced houses. This is a likely deeper explanation for the price/date correlation. These patterns are difficult to convey in the static pictures in this paper. Figure 6 shows the reuse of this exploration session on a subset of the data including only older buyers. The age-dependence of the distribution of sale dates is apparent – it is almost linear for these buyers, with winters being, if anything, more active.

Back to figure 5, another interesting pattern is the distribution of date bar lengths. One might expect that this distribution would tail off for longer bars, but the long bars are quite salient and it looks like the distribution is almost uniform. There is no attribute corresponding to this interval, however, so there is no histogram to look at. So the analyst defines one, using a spreadsheet-like interface. The resulting histogram has a rather long tail (figure 6), but nothing as sur-

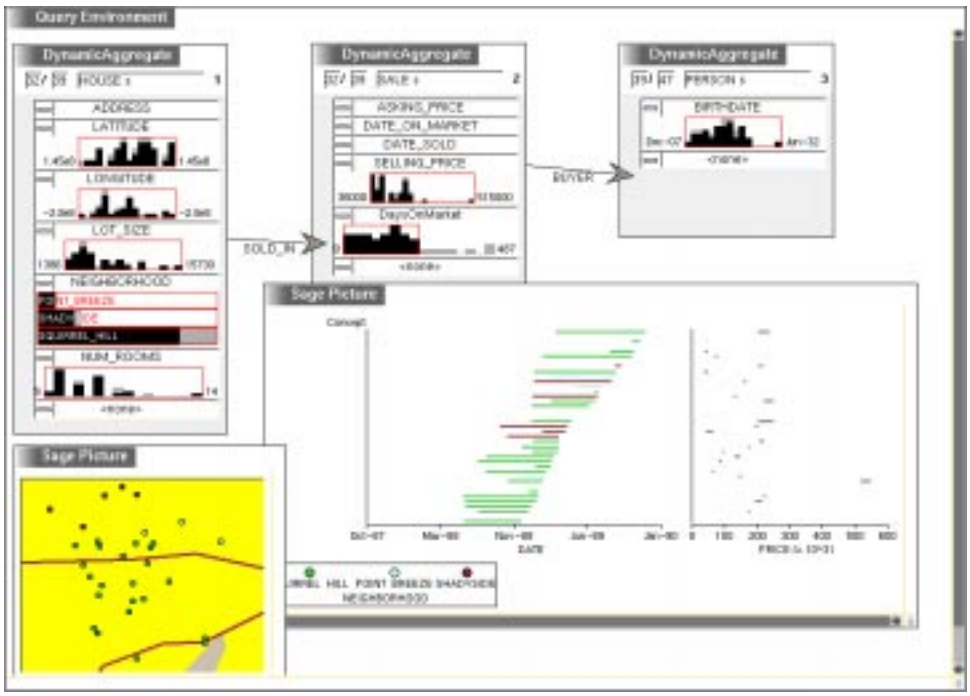


Figure 6: Noticing interesting patterns for older buyers, the analyst wants to reuse the previous visualizations, but for only the 47 (/126) older buyers she has currently selected with the Birthdate slider. There are two ways to remove the younger buyers from the QueryEnvironment, illustrating how negation is done in Visage. The first option is to move the slider to invert the selection, and then drag the newly selected younger buyers to the trash. Alternatively, the analyst can temporarily store the currently selected older buyers by dragging them to the desktop, leaving only 79 younger buyers. These are then dragged to the trash, leaving an empty query and visualizations. Finally the stored object representing the older buyers is dropped back onto the Person dynamic aggregate. The visualizations automatically zoom in to the new ranges of attribute values as shown.

prising as first appeared.

Mining for and Analyzing Rules

We can consider a VQE query as the beginning of a request to a data mining algorithm. The graph structure and slider settings define a dataset, and the visible attributes are those to be mined. The data mining tool's interface would be implemented in Visage and contain controls for the search. Dropping a VQE query could initiate the search. The results could be returned as a modified VQE query with new attributes defined for each discovered rule. For instance, a rule predicting DaysOnMarket would give rise to an attribute PredictedDaysOnMarket. The analyst could then do further exploration, seeking patterns to deviations from the rule. Previous visualizations are still in the workspace, and remain coordinated with the updated query.

Contributions

The primary point we want to illustrate is that interactive visualization systems have the potential to make it easy to explore data for tentative patterns. Here two visualizations immediately generated several hypotheses for deeper analysis, as well as patterns to steer a data mining algorithm away from. Coordination across multiple applications allows exploration to unfold continuously, without burdening the analyst with rerunning commands, regenerating visualizations at various stages, and still having them uncoordinated. These capabilities were enabled by a visual query language that forges an appropriate compromise between ease-of-use and expressive power, and by the dual extensional/intensional nature of dynamic aggregates. The queries required more expressive power than is available in previous visualization systems, yet were simple to construct by virtue of the analogy to direct manipulation operations in Visage. Queries involving multiple objects mirrors navigation, selecting properties to be visualized is done with a sketch, filtering is done with sliders and brushing, and arithmetic operations are entered as in spreadsheet formulas.

Related Work

IDEA

IDEA (1996) is an interactive tool for exploring and analyzing business marketing data. It uses a random subset of the data for exploration, and retains a history of user queries connected by derivational and subsumption relations. Each query is specified relative to a root relational table, and can thus be reused on the full dataset. IDEA emphasizes range selection and aggregation operations (termed segmentation), giving a careful taxonomy and semantics for the latter. However it can only explore a single relational table, and it merely imports and exports data to external tools, rather than offering a common front end for them.

Visage/VQE uses this random subset idea, and has a direct-manipulation interface (not described here) for specifying aggregation operations. It does not yet have a history mechanism. Visage relies more heavily on a shared object database optionally including metadata than does IDEA, although less so than IDEA's predecessor, IMACS (Brachman *et al.* 1993).

GQL

GQL (Papantonakis & King 1995) is a fully visual conceptual level query interface with the expressiveness of SQL. VQE's visual representation of the query graph is adapted from GQL, with some interface modifications such as using containment to show attributes rather than links. The problem with GQL as it stands is that it is not integrated with a visualization system for displaying query results. Each query generates a static table, so the paradigm is batch processing rather than the incremental query and direct manipulation exploration of VQE.

Summary

We have described an information-centric interface architecture for unifying the subtasks of knowledge discovery, allowing the analyst to focus on the process rather than the tools. Dynamic aggregates form a bridge between bottom-up exploration and top-down specification of analysis tasks, allowing drag-and-drop construction of queries either extensionally or intensionally. We illustrated this synergy with a single scenario that combined data cleaning, creating a dataset, data reduction and projection, multiple coordinated interactive visualizations, and session reuse and modification, together with suggestions for interfacing with a data mining tool.

Acknowledgements

This project was supported by DARPA, contracts DAA-1593K0005 and N660061-96-C-8503. We are grateful to the members of the Visualization and Intelligent Interfaces Group at CMU, and the Visage team at Maya Design Group for numerous discussions and for valuable comments on this paper.

References

Ahlberg, C.; Williamson, C.; and Shneiderman, B. 1992. Dynamic queries for information exploration: An implementation and evaluation. In *Proceedings of the Conference on Human Factors in Computing Systems (SIGCHI '92)*-, 619–626. ACM Press.

Becker, R. A., and Cleveland, W. S. 1987. Brushing scatterplots. *Technometrics* 29(2).

Brachman, R. J.; Selfridge, P. G.; Terveen, L. G.; Altman, B.; Borgida, A.; Halper, F.; Kirk, T.; Lazar, A.; McGuinness, D. L.; and Resnick, L. A. 1993. Integrated support for data archaeology. *International Journal of Intelligent and Cooperative Information Systems* 2(2):159–185.

Chuah, M. C.; Roth, S. F.; Kolojejchick, J.; Mattis, J.; and Juarez, O. 1995. Sagebook: Searching datagraphics by content. In *Proceedings of the Conference on Human Factors in Computing Systems (SIGCHI '95)*, 338–345. ACM/SIGCHI.

Fayyad, U.; Piatetsky-Shapiro, G.; and Smyth, P. 1996. The kdd process for extracting useful knowledge from volumes of data. *Commumications of the ACM* 39(11).

Kumar, H. P.; Plaisant, C.; and Shneiderman, B. 1995. Browsing hierarchical data with multi-level dynamic queries and pruning. Technical Report CS-TR-3474, University of Maryland.

Papantonakis, A., and King, P. J. H. 1995. Syntax and semantics of GQL, a graphical query language. *Journal of Visual Languages and Computing* 6:3–25.

Roth, S. F.; Kolojejchick, J.; Mattis, J.; and Goldstein, J. 1994. Interactive graphic design using automatic presentation knowledge. In *Proceedings of the Conference on Human Factors in Computing Systems (SIGCHI '94)*, 112–117.

Roth, S. F.; Lucas, P.; Senn, J. A.; Gomberg, C. C.; Burks, M. B.; Stroffolino, P. J.; Kolojejchick, J. A.; and Dunmire, C. 1996. Visage: A user interface environment for exploring information. In *Proceedings of Information Visualization*, 3–12. IEEE.

Roth, S. F.; Chuah, M. C.; Kerpedjiev, S.; Kolojejchick, J. A.; and Lucas, P. 1997. Towards an information visualization workspace: Combining multiple means of expression. *Human-Computer Interaction* in press.

Selfridge, P. G.; Srivastava, D.; and Wilson, L. O. 1996. Idea: Interactive data exploration and analysis. In *Proceedings of SIGMOD 1996*.

Tanin, E.; Beigel, R.; and Shneiderman, B. 1996. Incremental data structures and algorithms for dynamic query interfaces. *SIGMOD Record* 25(4).

Young, D., and Shneiderman, B. 1992. A graphical filter/flow representation of boolean queries: A prototype implementation and evaluation. Technical Report CS-TR-2905, University of Maryland.