In *Working Notes of the Symposium on Systematic Methods of Scientific Discovery, AAAI-95 Spring Symposium Series.*

Issues in Automating Exploratory Data Analysis

Robert St. Amant and Paul R. Cohen

Computer Science Technical Report 95-55

Experimental Knowledge Systems Laboratory
Computer Science Department, Box 34610
Lederle Graduate Research Center
University of Massachusetts
Amherst, MA 01003-4610

Abstract

Exploratory data analysis often plays a central role in the early stages of scientific inquiry. Models of complex phenomena are built incrementally, based on suggestive patterns in data and iterative refinement of plausible explanations. Unfortunately, exploration poses an explosive search problem. In this paper we present a planning approach to the problem. We outline the motivation for this approach and describe a planning system called Aide which we are developing to assist human analysts in exploring data.

This research is supported by ARPA/Rome Laboratory under contract F30602-93-C-0100.

1 Exploratory Data Analysis

Exploration often plays a central role in the early stages of scientific inquiry. One can rarely produce models of complex, unfamiliar phenomena on first contact with data. One must interpret suggestive features of the data, observe patterns these features indicate, and generate hypotheses to explain the patterns. Successive steps through the process can lead gradually to a better understanding of underlying structure in the data [Hoaglin *et al.*, 1983; Good, 1983].

Exploratory data analysis (EDA) encompasses a wide range of statistical tools [Tukey, 1977]. Simple exploratory results include histograms that describe discrete and continuous variables, schematic plots that give general characterizations of relationships, partitions of relationships that distinguish different modes of behavior, functional simplification of low-dimensionality relationships, and two-way tables such as contingency tables. These partial descriptions give different views of the data for a more complete, refined picture of underlying patterns.

EDA techniques have found application across a variety of scientific domains. In well-known studies, researchers have used EDA to attack problems in grouping corporations [Chen *et al.*, 1974], reducing TELSTAR data [Mallows, 1983], testing validity of approaches to ozone reduction [Cleveland *et al.*, 1974], and examining disease characteristics [Diaconis, 1985]. Our own use of EDA has led us to a better understanding of complex AI systems [Cohen, 1995; St. Amant and Cohen, 1994].

Viewed as search, EDA poses a difficult problem. Suppose we define search operators to be the menu operations in a statistics package. We now have a range of flexible, powerful possibilities available: arithmetic composition of variables, such as those used in function finding; model-based variable decomposition, as performed by linear regression; partitioning and clustering operations, such as those used in numerical and conceptual clustering systems; feature extraction operations such as statistical summaries; various transformation and data reduction operations. Unfortunately, this power is obtained at a price. The branching factor is large—think of the number of menu options active at any time. Furthermore, some operators may be repeated indefinitely, giving an unbounded search space.

Though difficult and painstaking, exploration is often manageable in human hands. Basic techniques and strategies for EDA can be communicated through textbooks in fairly straightforward terms. Specific characteristics of exploration make this possible: relatively few general principles guide exploratory procedures; difficult problems are often decomposed into smaller or simpler parts; exploration is constructive, often relying on partial results and incremental improvement to reach solutions. We can draw a natural analogy between exploration and *planning*.

We are developing an Assistant for Intelligent Data Exploration (AIDE) to assist human analysts with EDA [St. Amant and Cohen, 1995]. AIDE adopts a script-based planning approach to automating EDA. Data-directed mechanisms extract simple observations and suggestive indications from the data. Scripted combinations of EDA operations are then applied in a goal-directed fashion to generate simpler, deeper, or extended descriptions of the data. Control rules guide the EDA operations, relying on intermediate results for their decisions. The system is mixed-initiative, capable of autonomously pursuing high- and low-level goals while still allowing the user to guide or override its decisions.

The work presented here is based on a partially complete implementation of AIDE. All the specific mechanisms discussed have been implemented, and we have used AIDE to perform preliminary exploration on real data. We have not yet begun a systematic evaluation, however.

2 Themes in Exploration

Existing approaches to exploration fall into two classes: autonomous machine discovery systems that perform clustering or function-finding [Gennari *et al.*, 1989; Biswas *et al.*, 1991; Langley *et al.*, 1987; Schaffer, 1990], and user-driven statistics packages. These approaches occupy opposite endpoints on a spectrum of control. In applying a machine discovery system to a task, the user specifies the goals of the analysis implicitly in the form of the data input to the system. Opportunities for changing the goals that guide exploration are limited. In a user-driven statistical system, in contrast, the user has complete control. Unfortunately, the burden is completely on the user to guide the system through every decision in the analysis.

We argue that a mixed-initiative system, in which the user and a semi-autonomous system share control, can produce better results than either acting alone. Flexible control is the essential problem in exploration. We can attribute much of the success of the field of statistical consulting to the generality of statistical problem solving strategies—in our terms, the generality of control knowledge for data explo-

ration. Because statistical experts can often proceed with surprisingly little information about the domain of the data [Thisted, 1986], we believe that control knowledge, as opposed to domain knowledge, is a large part of their expertise.

Furthermore, planning offers powerful insights into the EDA process. While planning may initially seem unsuited to a problem in which the nature of desired results is often unknown, there are nevertheless strong similarities to EDA. EDA makes use of general procedures and explicit intermediate goals, derives results constructively, and relies strongly on problem decomposition. Consider the alternative: however sophisticated a statistical package may be, it inevitably treats operations selected by the user as independent of one another, though each result may depend on the entire structure of earlier results. EDA involves the selection of operations in light of dependencies imposed by earlier actions and requirements of potential future actions—a fair description of AI planning.

If we view EDA as a planning problem, we find that the goals of exploratory procedures fall into four general categories: description, simplification, deepening, and extension. Procedures that meet description goals generate results directly interpretable by human analysts, standard descriptive statistics and structures such as means, histograms, and tables. Simplifying procedures facilitate description: removing outliers from a relationship, for example, simplifies the application of a functional description to the relationship. Deepening procedures look beneath surface descriptions for further structure. Applying a deepening procedure is analogous to looking through a microscope, trading a global perspective for local detail. Searching for structure within clusters and examining residuals for patterns are two common examples of deepening. Finally, extension procedures combine or extend local descriptions to a larger context. Observing that one variable falls into five clusters and another into four clusters is part of description; observing that the clustering criteria are similar and then consolidating the two descriptions is a form of extension. We briefly return to these ideas later in this paper.

3 Mechanics of Data Manipulation

At the lowest level of representation we find symbols, strings, and numbers, the atomic data collected during an experiment. These elements are stored in relational form. Exploration often exploits implicit knowledge about structure in data, knowledge inconvenient to represent in pure relational form. In the relational representation used in AIDE, dataset attributes may directly contain other datasets. Attributes and datasets may be annotated with relevant (e.g. meta-data) information. Specializations of the basic dataset give variables (univariate datasets), relationships (multivariate datasets), graphs (datasets/variables containing bivariate relationships), and more complex structures.

Three classes of primitive data manipulation operations apply to a dataset: attribute transformations, dataset compositions/decompositions, and dataset reductions. These operations are parameterized and may be combined in different ways for surprisingly complex data manipulations.

An attribute transformation generates new attributes for a dataset through the combination of existing ones. Suppose that the "Persons" dataset contains records of different persons, including attributes "earned income" and "unearned income". Then we can generate a new attribute by addition, calling the new attribute "total income". While this transformation combines two attributes to make a third, we may also break down a single attribute into two or more. We could find an example of the latter type of attribute transformation in an attempt to predict income as, say, a linear function of age. "Total income" would then be separated into "age-predicted" and "age-residual" values. Univariate transforms, such as log transforms for symmetry, are also useful attribute transformations.

A dataset decomposition divides the rows of a datset into separate partitions, generating several datasets from a single one. A dataset composition performs the inverse operation. We may decompose the "Persons" dataset, for example, by the attribute "sex" for a closer view of the similarities and differences between the sexes with regard to income, as well as the behavior of income among men and among women. The newly generated datasets resulting from a decomposition are usually stored in a new attribute of a new dataset.

A dataset reduction combines the elements of a dataset into a single value, or a fixed number of values. For example, the mean statistic is a reduction of a variable to a single value, while Pearson's r reduces a bivariate relationship to a single value. Reductions can be more complex: we can view a simple linear regression as reducing a relationship to a slope, an intercept, a significance value, and so forth.

Designing a system around such general structures and operations is not simply an attempt at conciseness or elegance. Rather, the design is geared

specifically toward the kinds of operations appropriate for EDA. Consider generating a histogram for a categorical variable x in dataset d. We select a dataset decomposition that generates partitions of d, one for each value of x. Our operations now work on the aggregation of these partitions in a new, composite variable x' in a new dataset d'. An attribute transformation of x', which reduces each partition by the count statistic, gives the height of each bin; another transformation of x' based on the mode statistic gives a label to associate with each bin. These distinct values and their counts can be displayed directly in histogram form.

We have developed a scripting language to implement procedures like this one, based on work in knowledge-based signal processing [Carver and Lesser, 1993]. The histogram procedure is implemented by the script in Figure 1.

Producing structures in this way has two abstract benefits: complex procedures can often be seen as natural extensions of existing procedures, and natural connections between conceptually similar structures become clear. Consider now producing a contingency table for x and y. We follow essentially the same procedure as for the histogram, this time simultaneously decomposing x and y, which produces a two-dimensional dataset of partitions. Again we transform by the count statistic. Calculating the x, y values for each bin involves an additional step but is straightforward. The desired contingency table data is the result, as shown in Figure 2.

By combining operators in higher-level scripts, these procedures restructure the search space. The problem shifts from selecting an appropriate primitive operator to selecting an appropriate macro operator. The introduction of macro operators alleviates the search problem but unfortunately does not eliminate it.

4 Plan Level Control

The scripts we have presented so far are more structured than the discussion has indicated. Scripts are applied to satisfy specific goals. The histogram and contingency scripts, for example, meet goals of describing data with specific properties. Beyond simple sequences of primitive operations, scripts combine operations and subgoals using control constructs for sequencing, mapping, conditionalizing, and iteration. These constructs give scripts the flavor (and power) of a high level programming language.

Given the addition of subgoals and appropriate control constructs, we can view the histogram and contingency scripts as instances of a more general plan. We first break a relationship (univariate in the case of the histogram) into smaller or simpler components through a data decomposition. We then describe each component, by application of one or more reductions. We recombine the results (managed automatically by the representation) to provide a description of the data. This plan is shown in Figure 3.

This plan is more than a consolidation of the two perhaps trivial procedures for building histograms and contingency tables. It appears in different guises in a variety of related situations. If we were to build a box plot for a relationship between a discrete and a continuous variable, we would simply use the calculation of letter values when describing each component, rather than the "count" statistic used in the original scripts. If we wished to see the behavior of a continuous variable with respect to two discrete variables, in a two-way table of means, then we would simply bring this variable into the original data decomposition, and calculate the mean of the continuous variable per component in the description phase. Note the advantage of being able to mix primitive operations, such as data decomposition, with subgoals, such as selecting a descriptive reduction. More complex constructions are managed in similar ways.

Even at the more abstract level of planning we still face a search problem. A large number of plans may apply at any point. Two complementary sources of knowledge come into play to guide plan selection: indications and directives.

Indications are suggestive characteristics of the data, most often involving evaluation of a statistic or descriptive structure. Indications establish goals for exploration. For example, curvature in the residuals of a linear fit indicates that a higher order functional fit may be appropriate. The indication establishes the goal of finding this fit, which may grow into quite an involved process. Indications constrain plan selection based on *internal* characteristics of data.

Directives, in contrast, impose *external* considerations on exploration. Directives can supply knowledge about properties that are not explicit in the data. For example, a directive may inform AIDE that a set of observations collected at hourly intervals in a research lab can be viewed as a time series, or that a plausible grouping of the observations is into daytime and nighttime values, or that observations at 10:00AM may be unusual because of coffee breaks. Directives can also supply knowledge about desirable properties of descriptive results of explo-

```
(define-script variable-histogram-script (the-variable)
  :satisfies   (describe the-variable)
  :constraints (. . .<discrete values>. . .)
  :bindings    ((partition-ds (partitions count value)))
  :script      (:sequence
                (DECOMPOSE the-variable
                           #'(lambda (x) x)
                           :output partition-ds)
                (TRANSFORM partition-ds
                           #'(lambda (partition)
                               (REDUCE partition 'count)))
                (TRANSFORM partition-ds
                           #'(lambda (partition)
                               (REDUCE partition 'mode)))))
```

Figure 1: Histogram script

```
(define-script contingency-table-script (the-relationship)
  :satisfies   (describe the-relationship)
  :constraints (. . .<discrete values>. . .)
  :bindings    ((partition-ds (partitions count x y)))
  :script      (:sequence
                (DECOMPOSE the-relationship
                           #'(lambda (x y) (values x y))
                           :output partition-ds)
                (TRANSFORM partition-ds #'(lambda (partition)
                                            (REDUCE partition 'count))
                           :key (attributes partitions))
                (TRANSFORM partition-ds #'(lambda (partition)
                                            (REDUCE partition 'mode :key (attributes x)))
                           :key (attributes partitions))
                (TRANSFORM partition-ds #'(lambda (partition)
                                            (REDUCE partition 'mode :key (attributes y)))
                           :key (attributes partitions))))
```

Figure 2: Contingency table script

ration. For example, a common simplifying assumption made in causal modeling is that relationships between variables are linear [Glymour *et al.*, 1987]. Appropriate exploration directives check for gross departures from linearity and suggest appropriate transforms.

Indications help generate descriptions suggested by the data, while directives help generate descriptions appropriate for the goals of the analysis. Both guide plan selection by their presence in the goal and constraint forms of each plan. This gives a static control over plan selection: if a structure has a feature whose value is below a specific threshold, or if the user has specified that a linear description is desired, then some specific set of plans is selected. Often, however, we find that the results leading to some point in the analysis may influence the next step we take. Focusing heuristics dynamically manage plan selection.

Focusing heuristics take advantage of the sequence of operations leading up to a specific decision (the plan context) and the attributes associated with the data under consideration (the data context) to decide which of the applicable plans and structures should be pursued, which delayed, and which abandoned. These heuristics search through the planning and data hierarchy to provide the bridge from the high level guidance of indications and directives to

```
(define-plan decompose-transform-reduce-script (structure)
  :satisfies   (describe structure)
  :constraints (. . .<discrete values>. . .)
  :internal    (decomposition reduction components)
  :script      (:sequence
                (:subgoal (select-decomposition structure decomposition))
                (:subgoal (apply-decomposition structure decomposition components))
                (:iterate
                 (:sequence
                  (:subgoal (select-reduction decomposition reduction))
                  (:subgoal (transform-reduce components reduction))))))
```

Figure 3: Decomposition/description plan

the lower level of script operations. Focusing heuristics are implemented as structures that are passed from plan to plan, carrying relevant contextual information. Appropriate representation for their internal form is an open issue.

5 Discussion

We find that the planning representation captures to a large extent the notion of statistical strategy [Gale, 1986; Oldford and Peters, 1986; Hand, 1986]. In the AI and statistics literature, statistical strategy refers to a formal description of the actions and decisions to be made while using statistical methods in the course of a study. The simple example strategy we present here is one of several we are pursuing. It should suggest the more complex chains of reasoning AIDE is capable of carrying out.

Here is a more involved example, described at a coarse level: A user begins with a dataset describing the behavior of a system during an experiment. The user generates a graphical, partial causal model of the variables. As AIDE explores each relationship, it finds that $\langle x, y \rangle$, given as a direct relationship in the model, has a small negative correlation (in comparison with other correlations), but strong indications of clustering. AIDE partitions the relationship and explores each. In one partition AIDE produces a description of x and y as linear, with a high positive correlation; in the other partition x remains essentially constant. AIDE explores other model relationships, checking whether similar partitioning and descriptions hold, and finds that for several this is the case. AIDE furthermore finds a binary variable which closely matches the partitioned variable, as can be seen in a contingency table. AIDE partitions the entire dataset, with this evidence that the system behaves qualitatively differently in the two subsets of observations. All of these decisions are made visible to the user, to be adjusted or possibly overridden. Exploration continues.

Central to AIDE is the opportunistic, incremental approach to discovery described by Tukey, Mosteller, and other advocates of EDA. There are some obvious difficulties with the approach: in many cases local techniques can miss simple global patterns [Daniel and Wood, 1980]; local techniques can lead to a plethora of spurious results [Schaffer, 1990]; maintaining consistency in an incrementally growing set of descriptions can be difficult. Nevertheless, most such objections apply only to a system that acts autonomously, focuses on data but not on goals of the analysis, and pursues exploration paths independent of external context and context supplied by its own actions. We have designed AIDE to address these concerns.

Acknowledgments

This research is supported by ARPA/Rome Laboratory under contract F30602-93-C-0010. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copright notation hereon.

References

Biswas, Gautam; Weinberg, Jerry; Yang, Quin; and Koller, Glen R. 1991. Conceptual clustering and exploratory data analysis. In *Proceedings of the Eighth International Conference on Machine Learning.*

Carver, Norman and Lesser, Victor 1993. A planner for the control of problem solving systems. *IEEE*

Transactions on Systems, Man, and Cybernetics, special issue on Planning, Scheduling, and Control 23(6).

Chen, H. J.; Gnanadesikan, R.; and Kettenring, J. R. 1974. Statistical methods for grouping corporations. *Sankhya, Series B* 36:1–28.

Cleveland, W. S.; Graedel, T. E.; Kleiner, B.; and Warner, J. L. 1974. Sunday and workday variations in photochemical air pollutants in New Jersey and New York. *Science* 186:1037–1038.

Cohen, Paul R. 1995. *Empirical Methods in Artificial Intelligence.* MIT Press. In press.

Daniel, Cuthbert and Wood, Fred S. 1980. *Fitting Equations to Data.* Wiley.

Diaconis, Persi 1985. Theories of data analysis: From magical thinking through classical statistics. In Hoaglin, David C.; Mosteller, Frederick; and Tukey, John W., editors 1985, *Exploring Data Tables, Trends, and Shapes.* Wiley.

Gale, W. A. 1986. Rex review. In Gale, W. A., editor 1986, *Artificial Intelligence and Statistics I.* Addison-Wesley.

Gennari, John H.; Langley, Pat; and Fisher, Doug 1989. Models of incremental concept formation. *Artificial Intelligence* 40.

Glymour, Clark; Scheines, Richard; Spirtes, Peter; and Kelly, Kevin 1987. *Discovering Causal Structure: Artificial Intelligence, Philosophy of Science, and Statistical Modeling.* Academic Press.

Good, I. J. 1983. The philosophy of exploratory data analysis. *Philosophy of Science* 50:283–295.

Hand, D.J. 1986. Patterns in statistical strategy. In Gale, W.A., editor 1986, *Artificial Intelligence and Statistics I.* Addison-Wesley. 355–387.

Hoaglin, David C.; Mosteller, Frederick; and Tukey, John W. 1983. *Understanding robust and exploratory data analysis.* Wiley.

Langley, Pat; Simon, Herbert A.; Bradshaw, Gary L.; and Zytkow, Jan M. 1987. *Scientific Discovery: Computational Explorations of the Creative Processes.* MIT Press.

Mallows, C. L. 1983. Data description. In Box, G. E. P.; Leonard, T.; and Wu, C.-F., editors 1983, *Scientific Inference, Data Analysis, and Robustness.* Academic.

Oldford, R. Wayne and Peters, Stephen C. 1986. Implementation and study of statistical strategy. In Gale, W.A., editor 1986, *Artificial Intelligence and Statistics I.* Addison-Wesley. 335–349.

Schaffer, Cullen 1990. Domain-independent scientific function finding. Department of Computer Science Technical Report LCSR-TR-149, Rutgers University, New Brunswick, NJ. PhD Thesis.

St. Amant, Robert and Cohen, Paul R. 1994. A planning representation for automated exploratory data analysis. In Fisher, D. H. and Buntine, Wray, editors 1994, *Knowledge-Based Artificial Intelligence Systems in Aerospace and Industry Proc. SPIE 2244.*

St. Amant, Robert and Cohen, Paul R. 1995. Preliminary system design for an EDA assistant. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics.*

Thisted, Ronald A. 1986. Representing statistical knowledge. In Gale, W.A., editor 1986, *Artificial Intelligence and Statistics I.* Addison-Wesley. 389–399.

Tukey, John W. 1977. *Exploratory Data Analysis.* Addison-Wesley.