Intelligent Support for Exploratory Data Analysis

Robert St. Amant and Paul R. Cohen
Computer Science Dept., LGRC
University of Massachusetts
Box 34610
Amherst, MA 01003-4610
stamant@cs.umass.edu, cohen@cs.umass.edu

June 7, 1996

Abstract

Exploratory data analysis (EDA) is as much a matter of strategy as of selecting specific statistical operations. We have developed a knowledge-based planning system, called AIDE, to help users with EDA. AIDE strikes a balance between conventional statistical packages, which need guidance for every step in the exploration, and autonomous systems, which leave the user entirely out of the decision-making process. AIDE's processing is based on artificial intelligence planning techniques, which give us a useful means of representing some types of statistical strategy. In this article we describe the design of AIDE and its behavior in exploring a small, complex dataset.

Keywords: exploratory data analysis, artificial intelligence

1 Strategic aspects of exploration

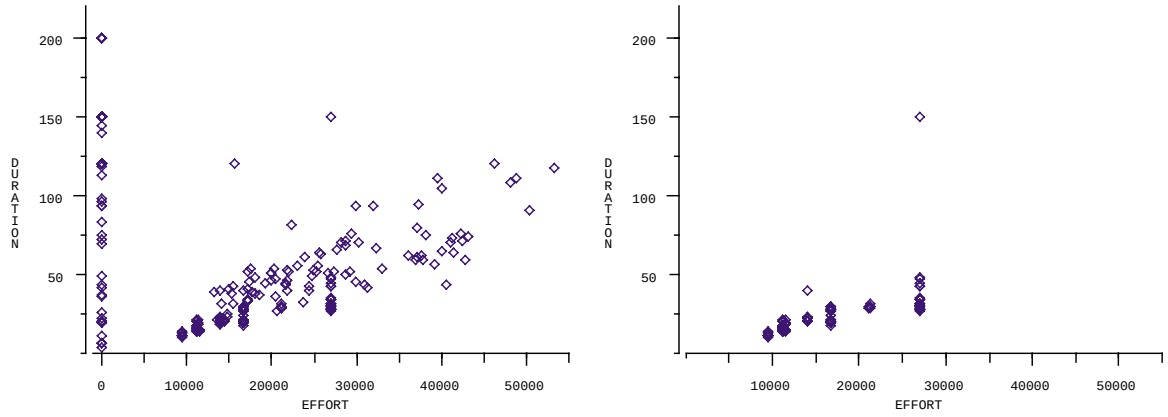
Exploring data is as much a matter of strategy as of selecting specific statistical operations. In his introduction to exploratory data analysis, for example, Tukey begins by laying out some general guidelines for describing data: anything that makes a simpler description possible makes the description easier to handle; anything that looks below the previously described surface makes the description more effective [25, p. v]. These strategic considerations guide our application of specific techniques for fitting lines, examining residuals, and so forth. Though most introductions to the field of EDA [27, 11, 12, 6] lay heavy emphasis on appropriate statistical techniques, none slights the importance of the strategies for their use.

In modern computer-based statistics packages we find a rich set of operations, suitable for almost any EDA application. These systems are nevertheless limited; they are almost completely lacking in strategic ability. Imagine a statistical system with both a set of basic exploratory operations and a set of strategies for applying them. A user might say, "Generate a linear fit for this bivariate relationship." The system then generates a least-squares or perhaps a resistant fit, checks the residuals for indications (e.g. curvature, outliers, unequal variance), performs appropriate transformations, iteratively refits the data if necessary, and reports all interesting results. The user might say, "There are clusters in this relationship," prompting the system to search for potential relationships between the clusters, to examine the behavior of the data internal to each cluster, to search other variables and relationships for similar behavior, and to present any findings. Further, the same system might initially suggest one of these lines of analysis, based on its own evaluation of the data. This system, instead of being a repository of statistical tools and techniques, comes closer to acting as an automated statistical assistant.

Two properties let us call a system an assistant rather than a sophisticated toolkit. First, an assistant is at least partially autonomous. We can give an assistant general instructions and let it make its own decisions about how to carry them out. Second, an assistant responds to guidance as it works. An automated system will inevitably make mistakes from time to time, so its reasoning process (past decisions as well as current ones) must be available to the user for approval or modification. A responsiveness to the guidance provided by human knowledge of context has been termed "accommodation" [15]. An accommodating system takes advantage of human knowledge to augment its own necessarily limited view of the world. The combination of autonomy and accommodation let the human data analyst shift some of the routine or search-intensive aspects of exploration to an automated system, without giving up the ability to review and guide the entire process.

We have developed an assistant for intelligent data exploration called AIDE. AIDE is a knowledge-based planning system that incrementally explores a dataset, guided by user directives and its own evaluation of indications in the data. Its plan library contains a varied set of strategies for generating and interpreting indications in data, building appropriate descriptions of data, and combining results in a coherent whole. The system is mixed-initiative, autonomously pursuing high- and low-level goals while still allowing the user to inform or override its decisions.

In this article we describe the design and behavior of AIDE. We begin with an example exploratory session, which lets us describe in concrete terms the capabilities we expect of an automated assistant. We discuss the internal data structures and control structures that represent exploratory statistical strategies in AIDE, and then turn to issues in the interaction between AIDE and the user. We end with a discussion of related work.



(a) All data (b) Clusters

Figure 1: Patterns in Planner Effort and Trial Duration

2 An EDA example

By "EDA" we mean techniques for fitting linear and higher-order functions to relationships, for composing and transforming variables with arithmetic functions, for separating relationships into partitions and clusters, for extracting features through statistical summaries, and the like. 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. EDA techniques are commonly distinguished from techniques for modeling and hypothesis confirmation; in EDA the emphasis lies on preliminary, informal probing of the data for underlying structure [9, 12].

We build our description of AIDE around an example taken from an experiment with PHOENIX, a simulation of forest fires and fire-fighting agents in Yellowstone National Park [5]. We first describe the experiment and our exploration of the data, originally carried out by hand, and then discuss the role an automated assistant could play in the process.

The PHOENIX experiment involved setting a fire at a fixed location and specified time, and observing the behavior of the fireboss (the planner) and the bulldozers (the agents that put out the fire). Variability between trials is due to randomly changing wind speed and direction, non-uniform terrain and elevation, and the varying amounts of time agents take in executing primitive tasks. In this experiment we collected forty variables over the course of some 340 PHOENIX trials, including measurements of the wind speed, the outcome (success or failure), the type of plan used, and the number of times the system needed to replan. We were mainly interested in about ten of the variables, especially in a comparison of the time it takes the planner to put out a fire (Duration) and the amount of fireline built during the trial (Effort). Figure 1a shows a scatter plot of these two variables.

We begin by observing that the relationship can be partitioned into two parts: a vertical partition at zero on the Effort axis and a separate, approximately linear partition. We call the gap in the distribution of Effort an *indication*, a suggestive characteristic of the data [16]. Examining other variables we find that the vertical partition corresponds to trials in which the outcome was `Failure`. Turning to the `Success` partition, we note that the correlation is positive, as expected. We fit a line to the `Success` partition, but note that there are two outliers from the general pattern. Examining the residuals of the fit, we find no indications of further structure, such as curvature,

# Replans =	0	1	2	3	5
clustered Effort	121	3	0	0	0
non-clustered Effort	4	63	20	2	2

Table 1: Clustered and non-clustered Effort for successful trials

WindSpeed =	Low	Medium	High
PlanType = X	11.0	15.5	18.8
PlanType = Y	14.5	26.7	22.1
PlanType = Z	19.5	29.9	29.7

Table 2: Median Duration by WindSpeed and PlanType for clustered data

that would render our linear description incorrect.

Looking more closely at the `Success` partition, we notice small, vertical clusters in the lower range of Effort. In a histogram, or a kernel density estimate, these would appear as peaks. The clustered points are isolated in Figure 1b. We can describe the behavior of the clusters in terms of their central location, by reducing each cluster to its median Effort and Duration value. These medians are also linear, with approximately the same slope as the line fitting the entire partition.

We then try to explain why some observations fall into clusters while others do not. We find a plausible explanation in the variable #Replans. #Replans records the number of times the PHOENIX fireboss was forced to abandon its current plan of putting out a fire and develop a new one. It turns out that the clustered data correspond largely to trials in which the planner did not need to replan; that is, observations fall into clusters mainly when #Replans = 0. This is shown in Table 1. If we further associate each cluster with a unique identifier, we can look for reasons why observations fall into specific clusters. We find that the two variables WindSpeed and PlanType strongly predict cluster membership. We can summarize the relationship with respect to Duration by a table of medians, as shown in Table 2. This gives a close view of the behavior of Duration as WindSpeed and PlanType take on different values. We can analyze this behavior in more detail if desired, using a median polish or similar technique [11].

This brief account gives the flavor of EDA. To see how an automated assistant might contribute to the process, consider the following dialog, which describes a detailed portion of the analysis above.

USER: Select relationship (Effort, Duration).
AIDE: (Effort, Duration) has these indications: high correlation ($r = 0.93$); outliers; clustering. Suggested courses of action are (1) generating a linear fit and (2) exploring clusters.
USER: Why the linear fit?
AIDE: A linear fit can be appropriate for relationships with high correlation.
USER: Why the given ordering of actions?
AIDE: When there is a high correlation and no evidence of extreme outliers, we fit a line before trying other possibilities.
USER: Okay.

AIDE: (Generates linear fit.) A three-group resistant line is appropriate; its intercept is -0.96, its slope 0.0017. (Presents graphical display of linear fit.) There are outliers in the residuals, as well as evidence of clustering.

USER: Show recent decisions.

AIDE: (1) With the goal of exploring (Effort, Duration), we decided to fit a line. The active alternative is to explore clusters. (2) With the goal of fitting a line, I decided without consultation on a resistant fit. Alternative is a regression fit.

USER: Back up a step.

AIDE: In fitting a line to the relationship, the possibilities are (1) a resistant fit, which is in progress, or (2) a regression fit, which has not started.

USER: Execute the regression fit.

AIDE: (Generates regression fit.) The regression line has an intercept of -2.73 and a slope of 0.0019; $r^2 = 0.57$. (Presents graphical display of linear fit.) There are outliers in the residuals, as well as evidence of clustering. The slope of the regression line is 1.13 times the slope of the resistant line.

USER: Make a note: I am confident that the relationship (Effort, Duration) is linear, though there are outliers. I prefer the regression fit.

AIDE: (Accepts note.)

USER: Okay.

AIDE: (Considers linear fit phase to be complete.) The plans remaining to explore (Effort, Duration) are (1) the linear fit, which is in progress, and (2) cluster exploration, which has not started.

USER: Okay.

AIDE: (Begins cluster exploration.) There are several different ways (Effort, Duration) can be broken into clusters. (Displays identifiers for potential clustering criteria: single-linkage clusters in Effort, regions of constant values in Effort, etc. Also displays similarity between resulting clusters.)

USER: Show single-linkage clusters in Effort.

AIDE: (Displays clusters.)

USER: Show constant Effort clusters.

AIDE: (Displays clusters.)

USER: Select constant Effort clusters.

...

This dialog captures the kind of interaction we desire between a user and an automated assistant, interaction actually supported by AIDE. We have described the exchange as a natural conversation; in reality, the dialog takes the form of mouse clicks, menu selections, typed input, and graphical displays. The information exchanged on the part of both conversants, however, is accurately rendered. Ideally our automated assistant has these features:

- The assistant performs a heuristic evaluation of the data. This lets it decide for itself which procedures are appropriate, rather than relying entirely on the user. In general, the assistant takes advantage of indications in the data as as more detail is uncovered, to focus its exploration on promising areas.
- The assistant does more than simply react to user commands. While the familiar "submit command $\rightarrow$ view result" cycle is still supported, the system anticipates the user's actions with knowledge of common statistical practice.
- The assistant does not go to the other extreme of being completely autonomous; to this end

the assistant makes its decision process accessible to the user. The user can thus review the assistant's decisions concerning statistical procedures, parameter settings, and variable selection. Further, the user is given a means of navigating through the exploration process, reevaluating and modifying decisions whenever necessary.

AIDE has all these features, which fall naturally out of its design as a mixed-initiative planner. The remainder of this article explains how AIDE's planning representation can generate such scenarios.

3 Planning

AIDE treats EDA as a search problem, more specifically a planning problem. Planners formulate the search problem in terms of states, goals, and sequences of actions to achieve goals. A planner solves a problem by constructing a step-by-step specification of actions that move from the initial conditions (the start state) through intermediate states to the desired conclusions (the goal state). The construction is complicated by constraints on the application of actions, uncertainty about the effects of actions, and unforseen interactions between goals, among other difficulties. Planners rely on task decomposition to solve complex problems, using knowledge about states, actions, and goals to structure the search at different levels of abstraction [19]. The simplest planners work by backward-chaining. Given a goal state, a planner begins by examining those actions that achieve the goal state. By treating the preconditions of these actions as goals to be satisfied in turn, and taking note of potential interactions between actions, the planner recursively generates a sequence of appropriate actions.

Traditional AI planners construct plans from primitive actions, starting from scratch for each new problem. Script-based planning takes a different approach [8, 3]. If a planning system repeatedly encounters similar problems, it becomes wasteful to generate similar solutions from scratch each time. Instead, a script-based planner can rely on a library of plans that apply to common cases. Rather than choosing an appropriate action at some point during the planning process, the planner can choose an existing plan in which many or all of the actions have already been decided on. If the library has sufficient coverage, planning largely becomes a matter of choosing which plan to apply at which time.

AIDE is a script-based planner. It draws from a library of about a hundred plans for its analysis, relying on a comparable number of indication definitions, rules, and other control constructs to guide its plan selection. In the next sections we discuss the actions that constitute basic plan primitives, the representation of AIDE plans, and the control of their execution.

3.1 Primitive operations in Aide

Following suggestions in the literature (e.g. [4]), we represent datasets as relations. Variables correspond to relation attributes, while observations correspond to relation tuples. Bivariate and multivariate relationships are relations with two or more attributes. A variable may be considered either an attribute of a relation or a relation consisting of a single attribute. In contrast to the usual relational database conventions, the domain of an attribute may contain other relations as well as atomic values.

We define three types of operations: reduction, transformation, and decomposition. These operations are higher-order functions that take functions and relations as input. They constitute a language for manipulating exploratory structures, in the same sense that the relational algebra constitutes a language for manipulation of relations in a database [22].

A *reduction* operation maps a relation to an atomic element. A reduction takes a function and a relation as input. Computing the mean of a sequence of numbers is an example of a reduction of a relation whose single attribute has a numeric domain. Letter values are reductions, as is the correlation between two sequences of numbers. All statistical summaries of this type fall into the class of reduction operations. We write `(reduce (function mean) ?sequence ?output)` where `?sequence` is a batch of numbers and `(function mean)` is the function called on the data. Where appropriate we may add bookkeeping arguments to specialize the types of new datasets and to associate names with new attributes.

Reductions summarize data. The functions by which we reduce batches are most often measures of location, scale, shape, and magnitude. Functions that compute location include the mean, the median, the trimean, different levels of trimmed mean, and weighted averages. For the other categories of functions we have similar ranges of possibilities. Reductions of relationships include measures of covariation (Pearson's r, Spearman's rank correlation, etc.) along with measures analogous to those we use for batches.

A *transformation* operation maps a function over the tuples of a relation. A transformation takes a function f and a relation R as input. The operation generates a new relation S, such that for each tuple r_i over the given attributes in R, S contains a corresponding tuple $s_i = f(r_i)$. Taking logs of a batch of numbers is a simple example of transformation: `(transform (function log) ?sequence)`.

The most familiar transformation is probably the generation of residuals from a linear fit. In this case the transformation function is subtraction, the input a relationship containing y and $\hat{y}$. The generation of residuals is an example of transformations that make patterns more easily visible. This kind of transformation also includes power functions that increase symmetry or that separate densely clustered regions in the data. A variety of specialized, ad hoc transformations become useful in special situations. For example, the reciprocal of a duration measurement is often meaningful. Ratios, sums, and differences of related measurements can be meaningful as well.

A *decomposition* operation breaks a dataset down into smaller datasets. A decomposition takes a relationship R and a mapping function M as input. The mapping function is applied to individual tuples, and returns a subset of $1, \ldots, k$ for each tuple. This set of integers can be viewed as a set of assignments of the tuple to a newly generated relation. The decomposition of R by M generates a new relation S, of cardinality k, with a single attribute whose values are new relations themselves. The contents of the ith relation in S are those tuples r_j in R such that $M(r_j) = i$. Decompositions may partition a dataset or generate overlapping subsets. Separating a dataset into clusters is a decomposition, as is isolating outliers in a relationship. For decompositions we write `(decompose (function mapping) ?dataset ?output)`.

Decompositions are mainly of two types. The first kind decomposes a relation into mutually exclusive subsets depending on a variety of criteria. We separate outliers from the main body of data, distinguish clusters, and separate partitions with different decompositions. The second kind of decomposition divides a relation into overlapping subsets, in order that a function may be applied within a limited "window" of the data. Smooths, kernel density estimators, and forms of robust regression rely on this form of decomposition.

These operations provide the primitives for data manipulations in the exploratory process. They offer surprising generality in representing common procedures. Consider a simple procedure for building a histogram, for a discrete-valued variable x: divide the range of the variable into its unique values; break the variable into subsets, one subset per unique value; count the number of elements in each subset. The resulting counts are the bar heights of the histogram, each bar associated with a different value of x. In other words, we `decompose` the variable, which generates

a new relation with an attribute that contains the subsets. We then apply a `transformation`, with an embedded `reduction`, which maps each subset relation to a single value, the "count" statistic of the subset. Now consider a procedure for building a contingency table for a relationship between x and some other discrete-valued variable y. We divide the relationship into subsets, one corresponding to each unique combination of x and y values. We record the number of observations in each subset, associating each count with a different x, y combination. The resulting values are the cell counts for the contingency table. A contingency table can thus be seen as a two-dimensional analog of a histogram.

Such similarities between statistical procedures are not unusual. Constructing a table of median Duration values for the PHOENIX dataset, for example, is similar to constructing a histogram or contingency table as above. A procedure for kernel density estimation has the same structure as the histogram procedure, but uses different decomposition and reduction functions. Such combinations of operations constitute primitive actions in the planning representation. These actions are the building blocks from which more complex plans are constructed.

3.2 Plans and goals in Aide

Above the level of primitive actions, AIDE supports two basic planning structures: goals and plans. AIDE begins with a single, top-level goal, the goal of describing a relationship, for example. To satisfy this goal, AIDE searches through its library for plans and actions that provide potentially relevant descriptions of the given relationship. Actions generate a result directly, while plans establish more detailed subgoals. These subgoals, when satisfied, combine to produce a final descriptive result that completes the plan to satisfy the higher-level goal.

A goal has a name, a form, and a set of plan variables contained in the form, as shown below. The form of a subgoal is a list of constants and plan variables. Plan variables are prefixed by the character "?", and may either have a binding or be unbound. Data are passed between goals and plans by unification.

```
(:SUBGOAL subgoal-name (generate-description ?data-structure ?result))
```

A plan has a name, a specification of a goal that the plan can potentially satisfy, constraints on variable bindings in the goal, and a body. The body of a plan is a schema of subgoals that must be satisfied for the plan to complete successfully. Control constructs in the schema allow sequencing (:SEQUENCE), iteration (:WHILE), conditionalizing (:IF, :WHEN) of subgoals, as well as other domain-specific forms. A simple plan is given below.

```
(define-plan fit-relationship ()
  "Describe a relationship by fitting it. Evaluate the fit."
  :satisfies  (generate-description :fit ?structure ?fit-operation ?fit-relationship)
  :features   ((?structure ((:dataset-type relationship)
                            (:cardinality 2))))
  :body       (:SEQUENCE
               (:SUBGOAL generate-fit
                         (generate-fit ?structure ?fit-operation ?fit-relationship))
               (:SUBGOAL evaluate-fit
                         (explore-by ?strategy ?activity ?model ?fit-relationship
                                     ?deepening-result))))
```

In words, this plan computes a fit for a bivariate relationship and then examines the fit for possible deviations. The `:features` form constrains the plan's applicability. The body of the plan

specifies that two subgoals must be satisfied in sequence for the plan to complete successfully. The `generate-fit` subgoal may be satisfied by different subplans: AIDE's current plan library contains plans for a least-squares linear fit, different varieties of resistant line fits, and smoothing fits. The `evaluate-fit` subgoal, in each of these cases, is matched by plans that explore the residuals for patterns not captured by the fit. While not an example of deep statistical knowledge, this plan nevertheless captures a basic exploratory strategy. As Tukey writes, "Anything that looks below the previously described surface makes [a] description more effective." [25, p.v] The `fit-relationship` plan makes explicit the dependence between the fitting and deepening procedures. Thus a pattern that appears in the residuals at the `evaluate-fit` step (outliers, say, for a regression fit) can be interpreted by the system as indicating that a different operation (perhaps a resistant fit) is called for at the `generate-fit` step. Plans like this one give us more than just a convenient way to implement statistical procedures. Their representation of explicit procedural knowledge lets the system (as well as the user) interpret results in the context in which they are generated.

The planning representation lets us build data descriptions of different scope and at different levels of abstraction. We have implemented different kinds of resistant lines, various box plot procedures, kernel density estimators, and smoothing procedures. We have found little difficulty in implementing most simple exploratory procedures. We have also developed more abstract sets of plans to implement forward selection algorithms for cluster and regression analysis [21], causal modeling algorithms [20], and incremental, opportunistic modeling procedures.

Planning works for EDA by reducing the search space through abstraction, problem decomposition, and identification of useful combinations of operations [14]. These are identifying characteristics of a planning problem, and are central to strategic reasoning in EDA.

Abstraction is ubiquitous in exploration. Fitting a straight line to a relationship involves deciding that variance around the line, evidence of curvature, outlying values, and so forth may be ignored at an appropriate level of abstraction. One fits a simple description, a line, before attempting to describe the residuals, i.e., those data that don't fit the abstraction well. The effect is of moving from higher to lower levels of abstraction.

Hierarchical problem decomposition plays a large part in exploration as well. Some relationships in the PHOENIX data, for example, can be partitioned into distinct, differently shaped clusters. We might describe such a relationship first in terms of the clusters locations, but then we would pursue the description of each cluster independently. Much of exploration can be viewed as the incremental decomposition of data into simple descriptions, which are then combined into a more comprehensive result.

Exploratory procedures often impose top-down structure on the exploration process. In other words, when we execute an exploratory operation we generally have a good notion of which operation, of many possible, to execute next. Further, common procedures often fall into a few basic families that process data in similar ways. It is easy to see, for example, that constructing a histogram involves the same procedures as constructing a contingency table: the contingency table is a two-dimensional analog of the histogram, with cell counts corresponding to bin heights. We can draw similar analogies between procedures for smoothing and for generating kernel density estimates, or between resistant line fitting and locally-weighted regression curves. While sometimes novel procedures are constructed from scratch, variations on existing procedures are much more common.

A planning approach to EDA gives several further benefits. Building plans is analogous to the generation of descriptions; the process is inherently constructive. Plans provide an explicit representation for the strategic, procedural knowledge we apply in exploring data. Plans provide a structured means of communication between the user and the system.

3.3 Control in Aide

Planning reduces the search problem of exploration, by considering a set of procedures rather than the much larger set of all possible combinations of primitive operations. We must still address some further issues:

- Which course of action (i.e., which plan) should be followed?
- Which data should be examined, which operations applied? More specifically, to which values should each plan's internal variables be bound?
- When should the choice of a specific plan or value of a plan variable be reconsidered?
- When should one course of action be abandoned for a different, more promising one?

In AIDE, these issues are addressed by three mechanisms. Activation and preference *rules* determine which plans are active and which values a plan variable is bound to. *Focus points* manage plans as they execute, maintaining and evaluating different possible courses of action as more information becomes available. A *meta-planner* controls plan execution sequences, maintaining a broader perspective on the exploration process. These elements play complementary and interacting roles in exploration.

During exploration of the PHOENIX dataset, a goal was established to describe the (Effort, Duration) relationship. Several options arose, each represented by a different plan. These plans were activated by rules triggered by features and indications, ranging from judgments about linearity and curvature to tests for clustering and outliers. The rule below activates the plan `generate-linear-fit` when the input relationship meets a set of tests, in particular that the indication `low-correlation` is not present. The judgment that a correlation is low depends on a threshold set initially by the system but modifiable by the user.

```
(define-plan-evaluation ((the-plan generate-linear-fit)
                         (the-model t)
                         (structure t))
  (when (and (has-features structure
                           '((:dataset-type relationship)
                             (:cardinality 2)
                             (:component (:not (constant-valued-p t)))
                             (:not (trivial-sequence-relationship *))))
             (has-indications structure
                              '((low-correlation-indication nil))))
        :active))
```

Rules simply activate plans to satisfy goals; several may be activated at once. A *plan focus point* determines which plan is most appropriate; this decision may change depending on context. A plan focus point designates a single plan as its current focus. When this plan completes, the focus point can exit, indicating that the goal has been satisfied; it may alternatively change the current focus to another plan, to let it execute instead. In the (Effort, Duration) example, plans for the linear fit and generating clusters were both allowed to execute, in turn. The plan focus point managed the process by making changes to its current focus.

A *variable focus point* is analogous to a plan focus point. Just as multiple plans can satisfy a goal, there may be more than one way to bind plan variables in executing a plan. In exploring a relationship with clusters, for example, it often happens that there may be several different ways to interpret the clusters. Under one interpretation there may be four clusters, in another five, in yet another four clusters but with a different assignment of observations to specific clusters. A

plan for exploring the clusters contains a plan variable, e.g. `?cluster-mapping`, that represents one of these interpretations. A variable focus point maintains all the different interpretations, so that they may be examined individually. Its behavior is essentially the same as that of a plan focus point.

Together, rules and focus points determine which plans should be applied to satisfy a goal and which bindings are appropriate for plan variables. These control mechanisms act locally, taking advantage of planning context. For example, rules determine that residual examination plans are applicable only in the context of a linear fit plan. A focus point lets them all begin execution, so that relevant indications may be observed. Another rule prevents the activation of a further linear fit to the residuals. There is more to exploration than local control, however. Sometimes a pattern in one relationship may suggest a course of action in the exploration of different relationship. Control in this case must shift, not just from one plan to another in a single focus point, but from one focus point to another. This capability is provided by the meta-planner.

From the meta-planner's point of view, AIDE's expansion of the exploration search space is a branching tree. The root of each subtree represents the result of making a sequence of choices: plan P rather than plan Q to satisfy goal G, binding variable v to b instead of a in plan P, and so forth. Each branch point, i.e., each node with more than one child, is a focus point. The meta-planner uses this branching tree to implement a form of opportunistic backtracking, which is sometimes called "refocusing" [3].

Whenever a focus point is visited (or revisited, on plan completion) the meta-planner is invoked. The meta-planner is identical in design to the planner that executes the base-level plans described earlier. The meta-planner has a single directive, however; it is responsible for deciding which focus point should be active and which of its plans or plan variable bindings should be pursued. In some cases the meta-planner, given a specific focus point, will simply return the same focus point; this indicates that the current course of action is the one it judges best. In other cases the meta-planner will return a different focus point, indicating that another decision is more promising than the current one. The current set of meta-plans in AIDE's library implement a depth-first search in the default case. In some situations the agenda of a focus point are explored in breadth-first fashion; it is useful, for example, to partially execute all of the residual exploration plans for a linear fit, until any relevant indications are derived, rather than executing each one to completion. In other situations the meta-planner must search through the plan network for an appropriate focus point.

To summarize briefly, evaluation and preference rules decide which plans are relevant in trying to satisfy a goal. A focus point selects one plan from those possibilities and executes it; the focus point may switch dynamically between possible plans as knowledge about the problem changes. Meta plans decide which focus point is currently active, i.e., which goal the system should be trying to satisfy at the current time. These mechanisms give flexible, opportunistic control over the exploration process.

Managing interaction with the user at the meta-planning level gives the AIDE design much of its power. Relieved of the necessity of calling individual actions directly, the user can take a strategic view of the process. The user participates in the exploration by viewing the sequential presentation of these focus points and providing acknowledgement or guidance when appropriate. In the next section we show how the interaction is managed.

4 User interaction

AIDE can be used like any other menu-based statistical package, in which one examines the data, selects a statistical operation, examines the result, and repeats the cycle. A key distinction between

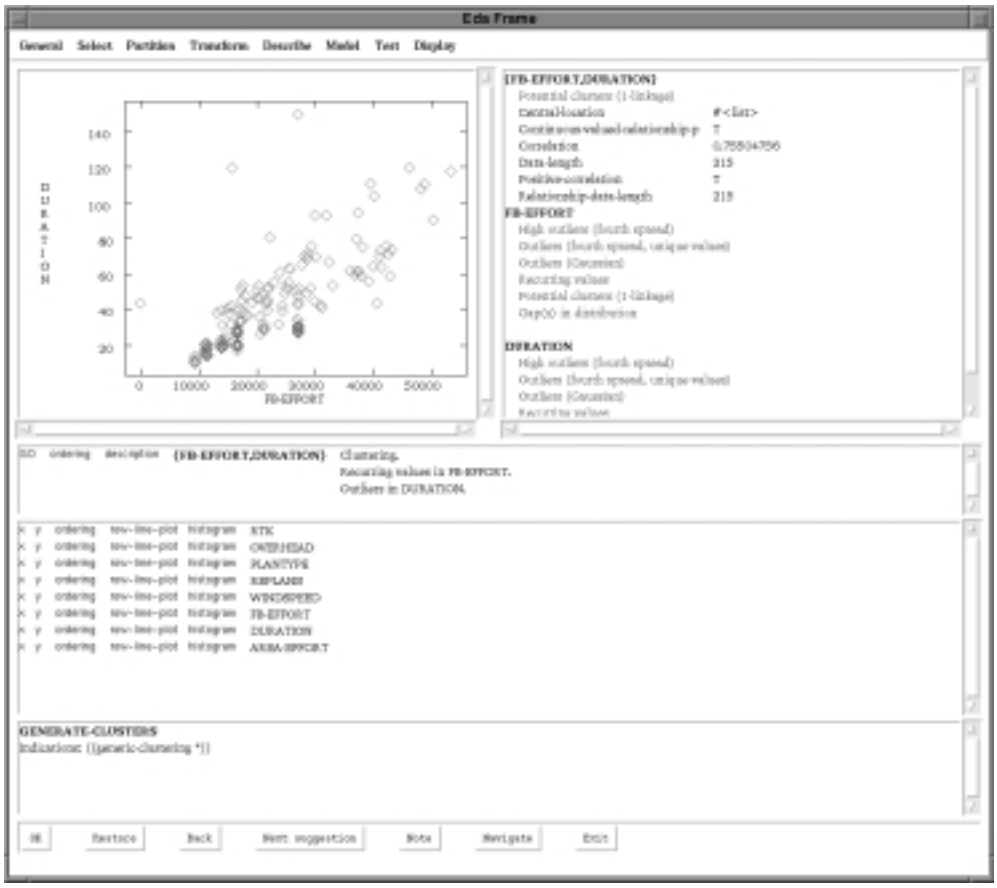


Figure 2: The AIDE frame

AIDE and a conventional statistical interface, however, is that in addition to displaying data and results, AIDE supports the exchange of information about the decisions that give rise to the results. This is more than simply recording and replaying a history of user commands: the user is able to move forward and backward through a growing hierarchy of decisions, reviewing and modifying choices where necessary. AIDE itself has the same capabilities. This lets interaction with AIDE exhibit a flexible balance between autonomous action and accommodation of human knowledge.

The AIDE interface is shown in Figure 2. Menu choices let the user load a dataset, compose variables into relationships, compute summary statistics, generate linear models, partition data, run statistical tests, and so forth. The upper left pane, the source pane, displays the variable, relationship, table, or other data structure under consideration, while the upper right pane, the result pane, contains indications or derived descriptions for the data structure. AIDE can thus be used as a conventional menu-based statistics package. The three remaining panes are the suggestion, selection, and documentation panes. They display, respectively, actions or choices that AIDE suggests the user take; other choices AIDE considers applicable but less appropriate than the one suggested; and documentation for the choices AIDE is considering and for the results AIDE has generated.

We can best see the interface at work by reviewing a portion of the dialog.

USER: Select relationship (Effort, Duration).

The user makes a selection by specifying, given a list in the selection pane of all dataset variables, a bivariate relationship with Effort as the `x` variable and Duration as the `y` variable. When the relationship is constructed, its scatter plot is displayed in the source pane and relevant features and indications appear in the result pane. For example, different indications of outliers are displayed, each generated by a different criterion. These indications are mouse-sensitive; clicking on one shows which points in the scatter plot can be considered outliers. Indications of clustering, curvature, skew, and so forth are similarly active.

AIDE: (Effort, Duration) has these indications: high correlation ($r = 0.93$); outliers; clustering. Suggested courses of action are (1) generating a linear fit and (2) exploring clusters.

When the user indicates that the construction of the relationship is complete, AIDE responds by displaying a set of relevant actions in the selection pane, with the linear fit action in the suggestion pane. Each action appears with an inline set of choices that allow the user to see, in the documentation pane, a justification for the action or the criteria by which the action was ranked among the other possibilities.

Later in the dialog AIDE generates a linear fit, in response to a user gesture of `Okay`. The `Okay` gesture prompts a context-dependent response from AIDE. It can be interpreted as an acknowledgement of a result AIDE has generated, as an acceptance of a suggested action, or as a command to proceed with an alternative action, among other more specific possibilities. Here the gesture is interpreted as acceptance of AIDE's suggested plan to fit a line to the relationship:

USER: Okay.

AIDE: A three-group resistant line is appropriate; its intercept is -0.96, its slope 0.0017. There are outliers in the residuals, as well as evidence of clustering.

A scatter plot of the relationship (Effort, Duration) is displayed in the source pane. The same scatter plot, with the linear fit superimposed, is displayed in the result pane. The parameters of the line are displayed in the documentation pane, along with indications of outliers in the residuals. These

indications are active: if the user selects one with the mouse, the relevant points are highlighted in the source pane. Several residual evaluation structures are automatically generated as part of fitting the line; actions to examine these further are presented in the selection pane.

These exchanges highlight the balance AIDE maintains between autonomy and accommodation. Because the selection of a variable or relationship often depends heavily on subject matter knowledge, AIDE makes suggestions rather than selecting data by itself. On the other hand, once a relationship has been selected and the user has decided on a general course of action, AIDE can take on more of the decision-making responsibility. In the last exchange, for example, the user accepts the suggestion of a linear fit, and AIDE pursues a linear fitting strategy which involves autonomous decisions about how to fit the line and how to examine the residuals (Residual examination is part of a larger set of plans, based on a regression strategy developed by Gale and Pregibon [7].) Other examples of autonomous behavior on AIDE's part include searching for predictive relationships involving categorical variables when clusters are explored, generating a linear fit for a relationship if it has been transformed to remove curvature, "sharpening" a relationship if no clear pattern is initially apparent [26], and performing various other data reductions and searches to bring out patterns. In some areas AIDE behaves much like a conventional statistical system but in other areas provides more autonomous assistance.

Maintaining this balance is harder than might appear at first glance. If the system can take actions without consulting the user, it can as a natural consequence drive the analysis into inappropriate areas, possibly losing the user in the process. Similarly, the user is not constrained to take only those actions the system finds reasonable; on the contrary, with better pattern-matching abilities and knowledge of what variable values actually mean, the user should be able to take the analysis in whichever direction appears appropriate. The system must neverthless be able to follow in the user's path, ready to contribute to the process once it again reaches a familiar area.

AIDE's plans alleviate this problem somewhat. They attempt to implement common statistical practice, relying implicitly on the user's knowledge of what actions are appropriate in a given situation. Thus if the system takes the initiative by executing a sequence of actions without consultation, the result will rarely be completely incomprehensible to the user. In the worst case, the user will think, "Why did the system choose to take that action in this situation?" while in the best case the response will be, "That's exactly what I would have done." The examination of residuals is a simple example of the best case: the exploration process continues in an expected direction. To handle cases in which AIDE's actions are more subtle, a set of navigation tools lets the user follow its decision-making and make changes when appropriate.

The navigation interface is shown in Figure 3. The interface is similar to the main exploration interface of Figure 2 but has a different function, namely, to "locate" the user in the "landscape" of the exploration process:

- The command history view displays a list of the commands that the user has selected from the menus. Selecting a command shows the operation, the data to which it was applied, and the result generated. This facility gives a history of the exploration process. It is a shallow history, though, in that it reflects only the actions taken, and not the decisions that led to their selection.
- The overview view presents a graph of the dataset, with variables represented as nodes and relationships as arcs, marked appropriately to show which have been explored and the types of descriptions that have been established for them. The user can review a variable or relationship along with its indications from this view, and can direct AIDE to return to the initial point of exploring that variable or relationship.

Figure 3: The navigation frame

- The data view displays the dataset in textual tree form. In addition to the variables and relationships of the dataset, this view also presents derived data (e.g., transformations, partitions) and results.
- The plan view displays the plan structure as it has been elaborated to the current point. Each focus point is presented, with documentation about the decision it manages. By browsing through this graph, the user can review past decisions. The user can also cause AIDE to return to an earlier decision to resume at that point.

  The plan view is a generalization of functions available directly from the main AIDE frame. From the main frame the user can enter a `Back` gesture, which causes AIDE to return to the previous decision. The user can also display recent decisions, from the current point back to the root of the exploration; these decisions, or focus points, may be revisited simply by selecting them.

The importance of a navigation facility is reflected by research in mixed-initiative planning. James Allen has drawn a useful analogy between mixed-initiative planning and dialog between problem-solving agents [1]. Allen identifies three distinguishing characteristics of the approach: flexible, opportunistic control of initiative; the ability to change focus of attention; mechanisms for maintaining shared, implicit knowledge. Mixed-initiative systems display these characteristics to a greater or lesser extent, depending on the domain in which they operate and the requirements on their behavior. AIDE implements them as follows.

AIDE's control of initiative changes with context. That is, whether a decision is presented to the user or is settled internally depends on situation-dependent factors. For example, if AIDE determines that only one plan is able to satisfy a given goal (i.e., all others rendered inactive by evaluation rules), then this decision point will not be presented to the user. An exception is made

in the case where a plan is being selected for the initial exploration of a variable or relationship. Choosing an appropriate initial plan for exploring a relationship can often depend heavily on external knowledge; thus the decision about how to proceed from the new point is presented to the user even if only one course of action seems appropriate. At any other point in the exploration, the user is free to take the initiative. Because AIDE supports the interaction style of a conventional statistical package, the user can simply select operations from the menu bar, entirely disregarding the suggestions and results AIDE generates.

When the user changes focus of attention, AIDE follows suit. For example, the user may decide to abandon exploration of relationship (x, y) for exploration of (z, w), simply by choosing `Select variable/relationship` from the menu bar and constructing the new relationship. AIDE interprets this as a shift in focus of attention, and runs the appropriate meta-level actions to match the shift. That is, it computes

AIDE also makes limited decisions on its own to change focus. For example, after fitting a line to a relationship and generating residuals, AIDE presents a set of residual examination and other plans to the user. An `Okay` gesture, which usually indicates that the top-rated plan for the current decision should be activated, rather in this context causes AIDE to refocus on other plans for exploring the relationship.

Part of the shared context between the user and AIDE is implicit in the plans stored in the library. These plans are intended to implement common statistical procedures, so that the user, when encountering a focus point, will find that it involves a plausible and perhaps even familiar decision. Navigation mechanisms further support the sharing of context. The user can view the data under consideration, results constructed, commands carried out, and planning structures leading to the current state. There is no complementary facility for AIDE to ask for clarification of user actions, however, which could clearly be helpful in some situations.

5 Related work

AIDE builds on work in the areas of statistical expert systems, statistical interfaces, and planning. In each of these areas, the cooperation between AI and statistical researchers has yielded useful results.

The study of statistical strategies, or formal descriptions of the actions and decisions involved in applying statistical tools to a problem [10], has led to automated systems in several areas. For example, Gale and Pregibon's REX implemented a strategy for linear regression [7]. REX was one of the earliest attempts to apply AI techniques to the representation of statistical expertise. In another research area, Oldford and Peters developed a complex automated strategy for collinearity analysis [17]. With the system TESS, Lubinsky and Pregibon studied the accommodation of human knowledge in data analysis [15]; rather than automating data analysis, TESS is modeled after systems that perform tasks like interactive regression, in which human knowledge of the meaning of the data is used incrementally to build a model. AIDE concentrates on two issues not fully addressed by these earlier systems: a flexible representation for procedural knowledge about statistical processing, and closer, mixed-initiative cooperation with the user.

Work in statistical interfaces has also influenced AIDE's development. Modern statistical packages provide sophisticated interfaces to support EDA (in fact, the more sophisticated systems contain far more functionality than we can include in AIDE, which is as much a tool for AI research as for statistical analysis.) Statistical environments like LispStat [24] and S [2], for example, contain procedures for regression analysis, cluster analysis, factor analysis, ANOVA, and dozens of other techniques. Systems targeted at specific approaches, such as grand tour or projection pur-

suit, provide a great deal of interactive support for their designated tasks. Conventional statistical software is very good at giving the user appropriate tools, often including programmatic control over their application; nevertheless it has little to say about how the tools should be used at a more strategic level [10, 18, 13]. AIDE aims to take over some of the strategic reasoning that guides tool application.

Mixed-initiative systems have recently become interesting to researchers in AI planning. An extension of PRODIGY, for example, provides autonomous planning with consultation [23], by letting the user review each decision the planner makes. Allen's work in the TRAINS project provides a more formal basis for mixed-initiative interaction [1]. Not surprisingly, research in statistical systems has followed a similar path. REX, for example, might be described as an autonomous regression system with consultation; it gave users control over decisions when reaching difficult points in its analysis. TESS took the approach further by letting users opportunistically modify its search strategies and decisions. AIDE, by relying on similarities between planning and EDA, can take fuller advantage of progress in planning research.

In summary, we have described the task of EDA and how it can be cast as a planning problem. We have shown how AIDE contributes to EDA: in providing a representation well-suited to procedural statistical knowledge, and in allowing flexible, mixed-initiative interaction with the user. We have described a representative problem in the domain of EDA, and explained AIDE's role its solution. We have run pilot evaluations of AIDE; a more comprehensive series of experiments is currently in progress to evaluate the performance of users working with and without assistance from AIDE.

References

[1] James F. Allen. Mixed initiative planning: Position paper. Presented at the ARPA/Rome Labs Planning Initiative Workshop. URL http://www.cs.rochester.edu/research/trains/mips/, 1994.

[2] Richard A. Becker and John M. Chambers. *S : an interactive environment for data analysis and graphics*. Wadsworth Advanced Book Program, 1984.

[3] Norman Carver and Victor Lesser. 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), November 1993.

[4] John M. Chambers. *Computational methods for data analysis*. John Wiley & Sons, Inc., 1977.

[5] Paul R. Cohen, Michael L. Greenberg, David M. Hart, and Adele E. Howe. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):32–48, Fall 1989.

[6] John Fox and J. Scott Long. *Modern Methods of Data Analysis*. Sage Publications, 1990.

[7] W. A. Gale. REX review. In W. A. Gale, editor, *Artificial Intelligence and Statistics I*. Addison-Wesley Publishing Company, 1986.

[8] Michael P. Georgeff and Amy L. Lansky. Procedural knowledge. *Proceedings of the IEEE Special Issue on Knowledge Representation*, 74(10):1383–1398, 1986.

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

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

[11] David C. Hoaglin, Frederick Mosteller, and John W. Tukey. *Understanding Robust and Exploratory Data Analysis*. John Wiley & Sons, Inc., 1983.

[12] David C. Hoaglin, Frederick Mosteller, and John W. Tukey. *Exploring Data Tables, Trends, and Shapes*. John Wiley & Sons, Inc., 1985.

[13] Peter J. Huber. *Computational Statistics*, chapter Languages for Statistics and Data Analysis. Springer-Verlag, 1994.

[14] Richard E. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[15] David Lubinsky and Daryl Pregibon. Data analysis as search. *Journal of Econometrics*, 38:247–268, 1988.

[16] Frederick Mosteller and John W. Tukey. *Data Analysis and Regression*. Addison-Wesley Publishing Company, 1977.

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

[18] Daryl Pregibon. Incorporating statistical expertise into data analysis software. In *The Future of Statistical Software*, pages 51–62. National Research Council, National Academy Press, 1991.

[19] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., 1995.

[20] Robert St. Amant and Paul R. Cohen. Toward the integration of exploration and modeling in a planning framework. In *Proceedings of the AAAI-94 Workshop in Knowledge Discovery in Databases*, 1994.

[21] Robert St. Amant and Paul R. Cohen. A case study in planning for exploratory data analysis. In G. E. Lasker and X. Liu, editors, *Advances in Intelligent Data Analysis*, pages 1–5, 1995.

[22] David Stemple and Tim Sheard. Automatic verification of database transaction safety. *ACM Transactions on Database Systems*, 14(3):322–368, 1989.

[23] Peter Stone and Manuela Veloso. User-guided interleaving of planning and execution. In *European Workshop on Planning*, 1995.

[24] Luke Tierney. *LispStat: An Object-Oriented Environment for Statistical Computingand Dynamic Graphics*. John Wiley & Sons, Inc., 1991.

[25] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley Publishing Company, 1977.

[26] P. A. Tukey and J. W. Tukey. Graphical display of data sets in 3 or more dimensions. In Vic Barnett, editor, *Interpreting Multivariate Data*. John Wiley & Sons, Inc., 1981.

[27] Paul F. Velleman and David C. Hoaglin. *Applications, Basics, and Computing of Exploratory Data Analysis*. Duxbury Press, 1981.