Info Vis

Article

Information Visualization
2021, Vol. 20(4) 207–228

DOI: 10.1177/14738716211038604
journals.sagepub.com/home/ivi
SAGE

Predicting intent behind selections in scatterplot visualizations

Kiran Gadhave[1], Jochen Görtler[2], Zach Cutler[1], Carolina Nobre[3], Oliver Deussen[2], Miriah Meyer[1], Jeff M. Phillips[1] and Alexander Lex[1]

Abstract

Predicting and capturing an analyst's intent behind a selection in a data visualization is valuable in two scenarios: First, a successful prediction of a pattern an analyst intended to select can be used to auto-complete a partial selection which, in turn, can improve the correctness of the selection. Second, knowing the intent behind a selection can be used to improve recall and reproducibility. In this paper, we introduce methods to infer analyst's intents behind selections in data visualizations, such as scatterplots. We describe intents based on patterns in the data, and identify algorithms that can capture these patterns. Upon an interactive selection, we compare the selected items with the results of a large set of computed patterns, and use various ranking approaches to identify the best pattern for an analyst's selection. We store annotations and the metadata to reconstruct a selection, such as the type of algorithm and its parameterization, in a provenance graph. We present a prototype system that implements these methods for tabular data and scatterplots. Analysts can select a prediction to auto-complete partial selections and to seamlessly log their intents. We discuss implications of our approach for reproducibility and reuse of analysis workflows. We evaluate our approach in a crowd-sourced study, where we show that auto-completing selection improves accuracy, and that we can accurately capture pattern-based intent.

Keywords

Provenance, reproducibility, intent, brushing, selections

Introduction

When experts interact with a visual analysis system, they are frequently guided by a domain-specific analysis question, such as identifying a gene that could be a drug target. To answer this question, they execute a series of intermediate tasks, such as selecting a set of correlated items for detailed analysis. In contrast to the high-level goal of answering a domain-specific question, these intermediate tasks are based on patterns in the data: for example, selecting outliers, clusters, or correlations. Such a carefully constructed selection of items based on a domain-agnostic structure reflects a reasoning process – an **intent** – by the analyst. We refer to the motivation behind these actions as the *pattern-based intent* of an analyst. Pattern-based intents are distinct from higher level intents in that they are free of context and based solely on the data. They are also distinct from low-level intents, such as hovering over an item to read its label. In this paper, we introduce methods to infer these pattern-based intents for brushes in scatterplots. *We define pattern-based intents as the reasoning behind selections based on statistical patterns*

[1]University of Utah, Salt Lake City, UT, USA
[2]University of Konstanz, Konstanz, Germany
[3]Harvard University, Cambridge, MA, USA

Corresponding author:
Kiran Gadhave, University of Utah, 201 S PRESIDENTS CIR RM 411, Salt Lake City, UT 84112-9200, USA.
Email: kiran@sci.utah.edu

or structures in a dataset. These selections can then serve as the basis for more sophisticated actions, such as filtering, querying, aggregating, or labeling, so that semantic knowledge about the purpose of the selection can be applied to these actions.

Why is capturing pattern-based intents important? First, inferring intents based on partial selections can be used to **auto-complete** selections. To select outliers, for example, analysts would have to brush only a few examples and could then auto-complete the selection, instead of painstakingly brushing the examples individually. Auto-complete can also be used to correct a selection. For example, if an analyst intended to select a cluster, reviewing the predicted cluster might reveal points that should be added to the selection.

Second, making pattern-based intents available in provenance data improves the **recall and reproducibility** of analytic processes conducted with visualization tools. By capturing such processes at a higher level of abstraction than just low-level interactions, they become more transparent when revisited either by the original analyst or a collaborator. Hence, analysis sessions that capture intents are more justifiable and likely to increase trust in the process.

Down the line, such rich provenance data also have the potential to enable **reusing visual analysis sessions** on modified or updated data. For example, when an analyst first removes outliers before proceeding with an analysis, that action could be translated into a rule, which then could be used to automatically remove outliers from an updated dataset.

We use scatterplots and tabular data as common and important representatives of visualization techniques and data types to demonstrate the feasibility of predicting intents from selections. To identify the types of patterns that map to these pattern-based intents, we conducted formative interviews with scientists who regularly use scatterplots in their research.

Our primary contribution is **a set of methods to detect and capture these pattern-based intents** for brushes and selections. We select data mining algorithms that are suitable to detect patterns in a dataset, and compute a large set of potential patterns for a dataset. We introduce methods to address the potentially large space of dimensions and parameters. Finally, we develop three approaches to score and rank the output of the algorithms relative to an analyst's selections.

Our secondary contribution is **an implementation of these methods in an interactive visualization technique**, thereby demonstrating how they can be leveraged for auto-complete and provenance tracking. By showing ranked predictions of patterns for a selection, we create a mixed-initiative approach that lets analysts easily capture their pattern-based intent by verifying a prediction. We provide the means to annotate these intents to tie them to higher level domain goals and capture this information in a provenance graph.

We demonstrate the usefulness of our approach in a set of examples. We also show that we can successfully predict pattern-based intents in a large, crowd-sourced quantitative study.

Background and related work

Our work is related to predicting intents in different contexts, data-aware brushes and selections, provenance tracking, and annotation of visual analysis processes, which we discuss in the following subsections.

Theoretical background

Selection is one of the fundamental interactions found in visualization systems.[1] Selections are typically communicated by manipulating the appearance of items, in which case they are also called a brush. In multiple coordinated view systems, linked brushing is frequently used to highlight the same items in multiple views.[2] However, selections can also serve as the first step in more complicated actions, such as filtering, extracting, querying, aggregating, grouping, manipulating, or labeling items. Hence, understanding the intent behind a selection is also useful to understanding the intents behind these derived operations.

Selections are commonly discussed in task analysis for visualization. Brehmer and Munzner,[3] for example, classify selections as a manipulation method in the *how* part of their typology. Rind et al.[4] classify tasks along a cube using the dimensions *abstraction* (*concrete* to *abstract*), composition (*low-level* to *high-level*), and *perspective* (*how* and *why*). In their design space, a selection is an abstract, low-level task of the how perspective.

Our goal, however, is to understand the *why* behind a selection and hence bridge the actions and the objectives: why has an analyst chosen to select a particular set of points? We attempt to infer pattern-based intents from domain-agnostic patterns in the data, and give analysts the opportunity to annotate their actions to capture domain-specific reasoning. Our ultimate goal is to realize Knuth's vision of literate programming for interactive visual data analysis, just like notebook formats such as Jupyter Notebooks or Observable have done for scripting-based data analysis.

Our pattern-based intents are related to *insights into the data*, as defined by Karer et al.:[5] "Insights affecting the viewer's knowledge about statistical and other structural information about the data." A difference in our definition of pattern-based intents is the

viewpoint: insight is about an analyst learning something, whereas intent is the reasoning behind an action.

Predicting intents

Inferring an analyst's intent has been studied in various contexts. For example, Myers[6] proposes methods for inferring operations and source code from demonstrations when implementing graphical user interfaces. More specific to data analysis, Gotz and Zhou[7] study analysts' activities and model them in four tiers, from high-level *tasks*, to *subtasks*, to *actions*, to *events*. Actions, which correspond to our pattern-based intents, are composed of a type, an intent, and parameters. They represent an executable, semantic step, such as a query, that bridges the high-level human cognitive ability and the low-level user interactions. Gotz and Zhou implement this framework in a prototype, named Harvest, that captures such actions. In contrast to our work, however, Harvest captures *that* an action was executed, but not *why*. A related tool that also captures actions is SensePath.[8] A key difference to Gotz and Zhou's work is that SensePath is optimized to support qualitative data analysis: it is made for analysts to use the log of semantic actions in qualitative coding.

Our approach is also related to query-from-example methods developed in the databases community. Dimitriadou et al.,[9] for example, infer range queries from a set of selected items. Cavallo and Demiralp[10] use an approach similar to ours for precomputing and predicting clusters. These approaches are limited to a single type of pattern. Our work, in contrast, considers diverse patterns and ranks the predicted pattern-based intents.

Dou et al.[11] argue that much of the reasoning process during a visual analysis session can be inferred by humans from inspecting user interactions, yet it is unclear whether a human's ability to do so can be leveraged by automatic methods.[12] Brown et al.[13] have shown that user performance and certain personality traits can also be inferred from analyzing user interactions. A thread of work is concerned with predicting future events in an analysis process to enable guidance.[14] Ottley et al.,[15] for example, predict future clicks on items based on an interaction history. Steichen et al.[16] and Gingerich and Conati[17] show that predicting lower level tasks, such as *retrieve value*, is possible using eye gaze data. This approach differs from our goal of predicting the intent of a current selection. Monadjemi et al.[18] propose a Bayesian approach to predict intents by ranking Gaussian distribution models based on user interactions. The ranked models can then be used to predict the next interaction, detect exploration bias, and summarize the analysis process based on click patterns. Battle et al.[19] propose ForeCache, a tool for the exploration of large datasets. ForeCache uses Markov chains and computer vision algorithms to model analysts' future actions based on their past moves. The system uses these predictions to pre-fetch data. In contrast to our work, the purpose of both these methods is to predict a future interaction or a region of interest.

A common goal for intent prediction is view specification, that is, the selection of data (sub)sets and suitable visual encodings. Systems such as Tableau's Show Me[20] use data properties to predict useful visual encodings. Natural language interfaces for view specification attempt to extract intents from language[21] and extract configurations for a view. Saket et al.[22] predict intents for view specification from demonstrations, such as assigning a color to a dot in a scatterplot, based on which their system infers the intent of mapping an additional variable in a dataset. Their follow-up work[23] demonstrates that analysts seamlessly switch between manual and mixed-initiative approaches. Demiralp et al.[24] compute patterns for a dataset and suggest visualizations for each of these patterns, but their approach is not reactive to analyst selections.

In summary, these related approaches usually target predicting of future interactions, visualization recommendation, query generation, or preloading of data subsets. Both our approach – mapping analyst selections to patterns and then ranking these patterns – and our applications – auto-complete and capturing intents to enable reproducibility and reusability – are different.

Data-aware brushes and selections

Selections, and the related concepts of brushes, queries, and filters, specify a subset of data items. Most selections are defined by explicit clicks on individual items, "paint-brushes" that select all elements under a brush tool, geometric brushes, such as rectangles or lassos, or textual queries. More advanced, data-driven brushes have also been proposed. For example, Fan and Hauser[25] introduce a method for fast brushing based on neural networks, where they estimate an intended selection based on simple sketches. Although they do not predict intents based on these brushes, a method like theirs could be used to improve brushing in our system.

Data-aware selections are actions that are defined in *data space*.[1,26,27] For a selection, for example, data-aware selections mean that it is described by conditions, not by a list of items. Dynamic queries[28] are commonly realized in a data-aware way: all items that fit certain conditions, defined, for example, via sliders, are considered to be in the query results. Certain types

of brushes[2] can be realized in a data-aware way. A rectangular brush in a scatterplot, for example, easily translates into the necessary conditions. Many selections (and other actions) are, however, realized by direct reference, for example, by pointing at items, and hence they are defined in *item space*. Actions that are defined in item space have several disadvantages: they cannot be generalized to apply to updating data, and they cannot be used to semantically explain a selection. Data-aware actions, in contrast, are robust to changes, can be used to explain and justify an action, and can be used in various ways to support an analyst, for example, by relaxing a selection,[29] or for reuse in a different context.[30] Most data-aware selections are realized by deriving rules directly from a rectangular brush. In more general cases, rules for data-aware selections are harder to derive. However, deriving the pattern of a selection (what makes the item in a selection belong to each other and different from everything else) is possible algorithmically. Xiao et al.,[31] for example, create "knowledge representations" of selections in communication networks. This approach is similar in spirit to our work, yet, Xiao et al.'s knowledge representations are limited to simple clauses and are not concerned with higher level patterns in the data.

Provenance

Our goal is not only to capture the intent behind selection-based actions, but also to explicitly track the intents and their constituting interactions for the purpose of reproducibility and, eventually, reuse of analysis actions. We use provenance tracking to achieve this goal. Provenance in the context of data analysis refers to the history of an artifact, such as a dataset, a computational workflow, or an insight. Ragan et al.[32] discuss different purposes of provenance, including recall, replication (reproducibility), presentation, and collaboration (among others), but do not discuss reuse. Ragan et al. also characterize the different types and purposes of provenance. They distinguish the provenance of data, the provenance of visualization, the provenance of interaction, the provenance of insights (which captures analytical findings), and the provenance of rationales (which captures the reasoning behind any decisions made). Most provenance-tracking techniques are limited to the former three, whereas insight and rational provenance can currently be achieved only using manual annotation.

Provenance tracking has two distinct approaches: (1) tracking the history of an analysis to achieve provenance (process-based), and explicitly modeling a visualization workflow (workflow-based).[33]

Workflow-based approaches are common in large-scale scientific data processing[34] in systems such as SCIRun.[35] Workflow approaches are also common for specifying the visualization pipeline, for example for volumetric data,[36] networks,[37] and tabular data.[38] A benefit of workflow-based systems is that they explicitly capture rules and thus can be reused easily. However, even these rules do not typically capture higher level semantics or intents.

Process-based approaches are the alternative to explicitly modeling workflows. They provide analysts with an interactive visualization systems while tracking the analysis process in the background.[12,39] Many visualization systems support the tracking of a history for the purpose of action recovery (undo/redo), so we limit our discussion to systems that explicitly target provenance. Examples include the graphical histories by Heer et al.[40] or CzSaw;[41] both render prior states as thumbnails. Various tools also represent histories as node-link diagrams,[42–45] and some methods automatically detect key states in an analysis process,[46] or retrieve prior states using search.[47] The provenance tracking in these systems is realized in an ad hoc way. However, recent papers have introduced software libraries for process-based provenance tracking,[48] including the Trrack library.[39] Trrack is developed by our team at the Visualization Design Lab. However, in all of these cases the tracked information is based on interaction logs and lacks higher level semantics.

Annotation

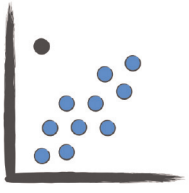
One approach to capture intents and semantics is through **note taking and annotation**. Annotations are common in visualizations designed for presentation, but are not frequently integrated in exploratory visualization tools, with notable exceptions.[40,45,49–53]

Manual notes, documentation, and annotations can capture analysts' reasoning and insights, but creating and maintaining them is associated with a burden on the analyst and thus a lack of scalability.[7] Hence, in this paper, we associate annotations with the corresponding provenance step, which can also be tied to pattern-based intents. This approach allows analysts to elaborate on their intentions, bridging the gap between pattern-based intents and domain-specific, higher level intents.

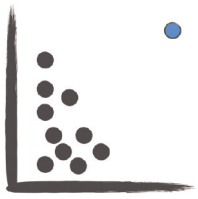
Patterns for selections

When analyzing data, analysts have intentions at different levels of abstraction. We are specifically interested in the pattern-based intents behind brushes or selections of data items in scatterplots, which are still semantically rich but domain agnostic.[54] To define a set of patterns that map to these intents, we first developed an initial classification based on the

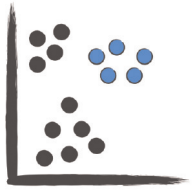
literature[55,56] and our own experiences working with scatterplots. We then validated and extended the initial classification through interviews with six scientists at the University of Utah who regularly use scatterplots in their data analysis. We used a convenience sample of domain experts we had interacted with professionally. Our inclusion criteria were: (1) regular use of scatterplots, and (2) and a willingness to share scatterplots or data used in scatterplots. The interview protocol was reviewed by the Institutional Review Board and classified as exempt from full review. We identified six participants from nursing, astrophysics, chemical engineering, psychiatry and population health, and surgery. The participants included one graduate student, one research scientist, and four faculty members. We provided participants with paper printouts of scatterplots of their own data and asked them to describe and highlight the kinds of patterns they find interesting. The goal of the interviews was to validate our initial classification of patterns based on the literature, and to identify patterns we might have missed. The interviews were video recorded and then transcribed. The transcriptions were coded by two independent coders using a seeded codebook developed from the initial classification of patterns: outliers, clusters, categories, multivariate optimization, and range queries. A table in the Supplemental Material shows the code frequencies from both coders for each interview. Both coders identified many instances of outliers, clusters, categories, and range queries. Only one of the two coders identified two cases of multivariate optimization. Both coders frequently identified correlation analysis, which we originally had not included in our set of patterns. Based on this process, we identified the following data patterns that match the analyst intents when analyzing data in scatterplots.



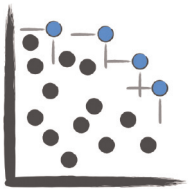
Correlations. Correlations are associations between two or multiple dimensions. They were mentioned as a target pattern in five of our six interviews with domain experts. Frequently, analysts were looking to identify correlations in the overall datasets or parts of the data, but also attempted to find points that do not fit the correlations. They had the intent to **identify subsets of data that correlate**, but also **identify items that do not fit the correlation**. In several interviews, these points were identified as "bad data." We found that participants did an approximate visual regression analysis, identifying both linear and nonlinear trends.



Outliers and inliers. Outliers are data points that differ significantly from other items. They were brought up as a pattern of interest in all six interviews. Frequently, analysts wanted to understand what causes the data points to be outliers, relying on their background knowledge. Outliers are also related to, but distinct from, the points that do not fit a correlation: for example, an item can be an outlier in its magnitude but perfectly fit the correlation. Outliers were also mentioned as bad data that should be filtered out. We consider both outliers and "inliers," that is, the set of points that are not outliers, as target patterns.



Clusters and groups. Clusters or groups of data points are items that are similar to each other, but distinct from the rest of the dataset. They were mentioned as a pattern that analysts look for in three of six interviews. Clusters were frequently not well defined in the data the experts we interviewed analyzed.

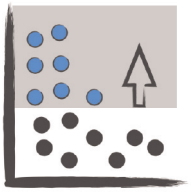


Multivariate optimization. One goal when analyzing data is to find data points that are dominant over multiple dimensions. A typical example is to find a hotel that is both close to the city center and affordable. The set of such points is often called a skyline.[57] Hotels in the skyline are such that no other hotel is both cheaper and closer to the center. Skylines were brought up in two of our six interviews, and hence are the least frequently mentioned pattern.

Categories. An observed pattern can sometimes be traced back to the items being of distinct categories. Four of our six expert participants mentioned they intend to select elements by category. For example, one expert wanted to separate data points based on categories, where one category corresponded to an experimental condition, and the other category was made up of unmanipulated controls.

Figure 1. Scatterplots showing three dimensions of a dataset. An analyst has brushed points in the right scatterplot based on a pattern they see (orange points). Our system predicts possible intents and ranks them by their match to the current selection. The points in green show a cluster that is recommended by our system based on the selection. When an analyst accepts this suggestion, a semantically rich log entry is stored in a provenance graph, shown on the right. [Interactive Figure].

Ranges. Four of the six experts mentioned they select data based on numerical ranges. Several experts stated these ranges can be based on domain conventions for setting cut-offs. We observed range selections based on single or multiple dimensions, implying that ranges can be combined for more complicated queries.



Discussion. We believe that the described patterns cover a broad range of use cases, but we do not argue that our list of patterns is exhaustive. For example, domain-specific patterns might be meaningful in certain contexts. Sarikaya and Gleicher[56] describe tasks for analyzing scatterplots. Each of our patterns can be mapped to one or multiple tasks from their work. For example, they mention tasks like *identify anomalies*, *identify correlation*, and *search for known motif*, which can be mapped to the *outlier*, *correlations*, and *cluster* pattern, respectively. Their list of tasks, however, goes beyond patterns, including, for example, *explore data*, and they do not explicitly mention some of our patterns, such as multivariate optimization.

Our pattern classification is limited to tabular data in scatterplots. We expect that other patterns, such as rankings, would be common in different representations. Finally, we have sometimes included a pattern and its antipattern, such as outliers and nonoutliers as separate patterns, but we have not done so consistently for all patterns. We have included anti-patterns for those cases where they were explicitly mentioned in our interviews (outliers and correlation). However, anti-patterns could also be considered for other cases.

Mapping patterns to intents

Most patterns that we identified in our formative study are also commonly targeted in data mining, which implies that various algorithms can be used to identify them. We leverage this diversity to calculate a broad set of patterns using different algorithms, combinations of dimensions, and parameters. We then compare the computed patterns with analysts' selections and rank them according to that match. Whereas our initial step creates a large set of patterns, the subsequent ranking makes these patterns manageable. We explain the details of the algorithms used and our ranking approaches in this section. Figure 1 gives an overview gives an overview of our method.

Up to this point, we have implicitly assumed that the patterns we discussed appear in two-dimensional space. In practice, however, many datasets have much higher dimensionality. Hence, a key question we have to answer is: For which dimensions should we calculate predictions? We considered calculating patterns for all pairs of dimensions, all dimensions that are actively brushed in the system, all dimensions that are visible in the system, all dimensions in the dataset, and any combination of these options. Calculating all possible options is computationally expensive, if not

prohibitive, but also not necessary. As we aim to predict the intent of analysts interacting with (possibly multiple) 2D scatterplots, and not to reveal high-dimensional patterns, we decided to limit predictions to (1) pairwise dimensions and (2) the dimensions that are actively brushed. We believe that predicting patterns on pairs of dimensions is the most appropriate choice for 2D scatterplots, as these patterns match what is visible in the plot. This restriction to pairs of dimensions is also supported by the fact that we did not encounter examples where experts wanted to select items based on more than two dimensions. However, we also do not want to exclude the possibility of analysts selecting higher dimensional patterns. Hence, we also calculate all patterns for all dimensions that are actively brushed, as the brushes indicate that an analyst is explicitly interested in a combination of these dimensions. Consequently, in a set of two 2D scatterplots visualizing dimensions A/B and C/D, and with active selections in both scatterplots, we calculate and predict patterns in two dimensions for A-B and C-D; and patterns in 4D space for A-B-C-D.

For example, if an analyst would like to select the species in Fisher's prominent Iris flower dataset, a selection based on 2D combinations of dimensions would be difficult as the feature are not well separated in any combination of 2D plots. In our system, they could start by plotting sepal width and sepal length and brushing a group of similar plants. They can further narrow the selection down by brushing in a second plot showing petal length and petal width. This combination of selections triggers a prediction considering all four dimensions. They can then select the cluster prediction that best matches their intended selection, leveraging patterns computed on higher dimensional space. A live version of this example is available [here].

Algorithms

Many algorithms can extract the patterns we describe. In our system, we deliberately rely on standard algorithms that are robust and simple, although more sophisticated versions might exist. One reason for this is generality: Many data mining algorithms require careful choices of hyperparameters, but choosing good parameters requires expertise and trial and error, which is not acceptable for our use case. Instead, we choose parameters for these simple algorithms by sampling the parameter space or rely on defaults. For example, we run k-means with a k of 2–7, but use defaults for all other parameters. We do not use evaluation approaches for the quality of the outputs; instead, we let our ranking approach reveal the most suitable results. We also assume that the visualization uses linear scales. However, an extension to logarithmic or power scales would be straightforward. We use algorithms provided by *Scikit-learn*[58] unless noted otherwise, and normalize the data before the analysis.

We use two different algorithms for **clusters**, DBSCAN and k-means, that have complementary strengths, since different algorithms pick up different kinds of clusters, such as circular clusters or concave-shaped clusters. DBSCAN is based on a (parameterized) measure of density (clusters are clouds of dense points of arbitrary shape), whereas k-means assumes roughly spherical clusters and requires the cluster number as a parameter. If no clusters are present, DBSCAN considers the whole dataset as one cluster (except for outliers), whereas k-means always provides a segmentation of the dataset. We solve each formulation multiple times with different parameterizations

For **outliers**, we use two algorithms: the local outlier factory and the outliers identified by DBSCAN. We treat inliers provided by the local outlier factory as a separate prediction named *nonoutliers*.

Multivariate optimization is used to find values that are optimal across multiple dimensions. Although a general optimization would require weighting the value of each dimension, **skylines**[57] are a generic approach that determines the items that are not dominated by other points. As a skyline requires a definition of what is considered "good" in each dimension (e.g. a *low* price, but a *high* customer rating is considered good for a hotel), we compute skylines for all high/low permutations of the 2D cases. We limit predictions to all-low or all-high for higher dimensional cases, because calculating all possible permutations would be computationally expensive.

If a dataset contains categorical values, we treat each **category** as a separate pattern. Individual categories could conceivably be shown in the scatterplot, but predicting an overlap between a selection and a category is especially important if a dataset has many categories that cannot be shown at the same time.

Finally, we use regression as a framework to analyze **correlations** in the data. To identify the sets of points that are part of a linear or quadratic correlation pattern, we run the following algorithm on linear and quadratic regression datasets, where X is the entire dataset, I are points marked as inliers, R is the regression model built with Scikit-learn, r_i is the residual for a point x_i calculated from R, and *iters* is a constant for the maximum number of iterations we execute if the algorithm does not converge earlier.

1. First we assume all the points in the dataset X are inliers I and build a Scikit-learn regression model R on the I.

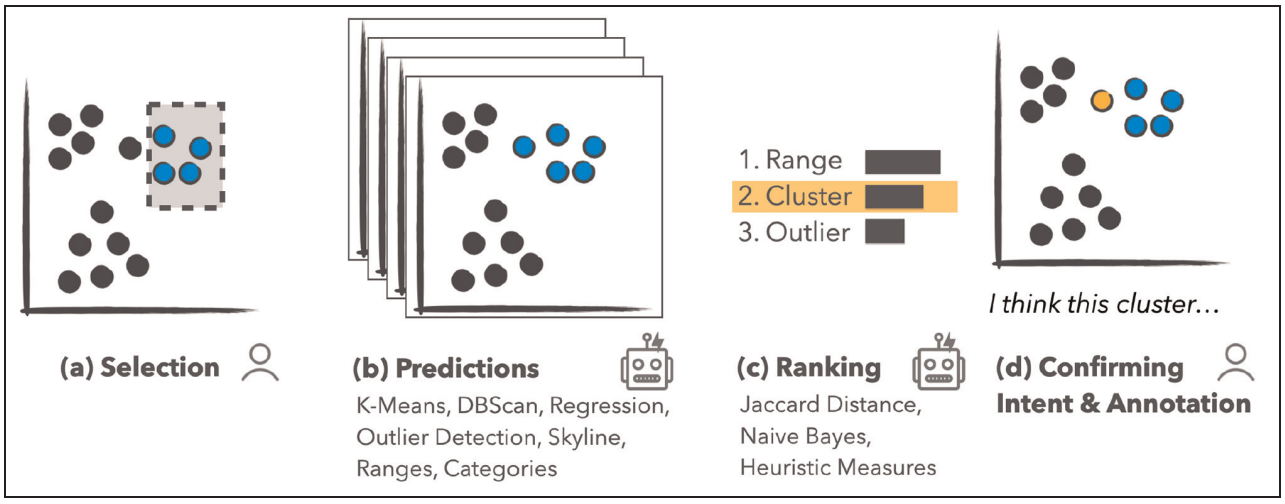


Figure 2. Overview of our method for mapping patterns to intents: (a) an analyst makes a selection in a scatterplot, (b) the system calculates many different patterns using various algorithms that we use for prediction, (c) we rank how well the analysts' selection matches the predicted patterns, and (d) the analysts select their intended pattern and provides annotations to capture their thoughts.

2. Then we calculate residuals r_i using R for all points x_i in I.
3. Next we define $\bar{r} = median(r_i | x_i \in I)$.
4. Then we redefine I as all the points $x_i | x_i \in X$ where $r_i < 2\bar{r}$.
5. We repeat points 1 to 4 for a predefined number of iterations *iter*, stopping early if inliers do not change between iterations.

The pseudocode for the above algorithm is expressed in algorithm 1.

Algorithm 1 Calculate inliers for a correlation pattern.

```
I ← X                                   ▷ initially mark all points as inliers
while iters > 0 and I is changed do
    R ← Regression(I)                   ▷ building regression model
    r̄ ← median(r_i | x_i ∈ I)           ▷ median residual over I
    I ← x_i | x_i ∈ X ∧ r_i < 2r̄        ▷ update inliers
    iters ← iters − 1
end while
```

Ranking predictions

All the described patterns result in classifications for each item in the dataset. To rank the predictions in our system, we compare these patterns with a binary classification representing an analyst's selection. Figure 3 shows an overview of our method. Some algorithms, like clustering, produce a multiclass prediction, which we first transform into a set of binary classifications using one-hot encoding. We can then use a similarity metric to rank each of the predictions. We use a preprocessing step to remove duplicate predictions for the same pattern from the set of predictions to rank. Duplicate predictions occur frequently if a pattern is robust to different parameterizations of the same algorithm.

In the following subsection, we discuss three ways to rank the predicted patterns that are optimized either to infer intent for an existing selection, or to predict intent of a partial selection, plus a special case for ranking range queries.

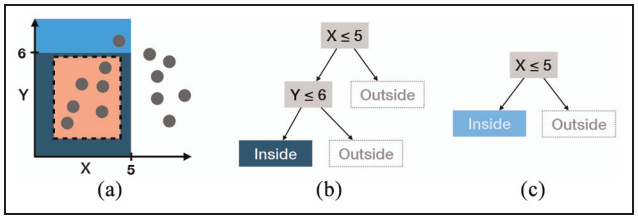


Figure 3. Using decision trees to capture range-based queries. (a) A brush is shown in red. The brush geometry can be described with four rules. (b) The decision tree simplifies the brush to two rules, illustrated in dark blue in (a). (c) A simplified decision tree, where one level has been removed. The result is a simple rule, which also includes a point that was not in the original selection, contained in the light-blue area in (a).

Ranking for inferring intent. Our baseline metric is the Jaccard index, which is a measure of similarities between sets. We consider the set of selected items S, and the set of items in a candidate pattern C. The

Jaccard index $\mathcal{J}(S, C)$ between those two sets is then defined as

$$\mathcal{J}(S, C) = \frac{|S \cap C|}{|S \cup C|} = \frac{|S \cap C|}{|S| + |C| - |S \cap C|}$$

Here, a value of 1 corresponds to a perfect match, and a value of 0 indicates no overlap. The Jaccard index is well suited to infer the intent of an existing, complete selection.

Ranking for auto-complete. The tasks of auto-completing and inferring intent differ with respect to ranking a possible pattern: In the case of inferring intent for a completed selection, finding the best match overall is necessary. In contrast, for auto-completion, the selection is partial, as the goal of the task is to complete the selection. Hence, we needed to develop a ranking approach that does not penalize incomplete selections. To do this, we rank the predictions using a modified Jaccard index $\mathcal{J}_m$. We define the similarity between sets S and C as,

$$\mathcal{J}_m(S, C) = \frac{|S \cap C|}{|S \cap C| + |C \backslash S| + w \times |S \backslash C| + r \times |X|}$$

Here X is the complete dataset. The modified similarity metric reduces the penalty for points in S that are not present in C by down-weighting $|S \backslash C|$ using a factor $w < 1$, reflecting the goal of a partial selection to be automatically completed. The metric also adds a regularization parameter of r to prevent boosting ranks in cases where few correct points are selected. Empirically, we found that $w = 0.2$ and selecting r such that $r \times |X| = 3$ gives good results for datasets that are suitable to be visualized in scatterplots. Due to the regularization, the metric never reaches 1, but 0 still indicates no overlap.

Ranking ranges. Our range-based queries rely on a decision tree of arbitrary depth; hence, the pattern captured by that decision tree is always a perfect match to the selection. Consequently, the range query would always rank at the top if we ranked it using the Jaccard index. However, this ranking is inconsistent with what humans perceive as a good prediction of their intent: when analysts create complex selections, they tend not to think of them as long lists of rules. Instead, they likely select a pattern based on a higher level relationship in the data. To address this inconsistency, we assign a score R to the range-based query using a heuristic based on the depth d of the decision tree: $R = \frac{1}{d^2}$. Our heuristic relies on the assumption that simpler queries are more likely to match an analyst's intent than complex queries that require deep decision trees to represent them. The resulting score is on the same scale as the Jaccard index, and hence can be easily integrated.

Probabilistic ranking. The Jaccard index considers each possible pattern independently. However, an analyst's intent is rarely independent, and some predicted patterns are more likely than others. To address this, we propose a probabilistic framework that models these effects. We denote predicted patterns with $C_i \in \mathbf{C}$ and the Boolean vector representing the analysts' selection as S. Finding a probabilistic ranking of the predicted patterns is the same as determining the conditional probability $P(C|S)$ for each pattern. Framing the problem using probabilities also gives us more interpretability as it relates the different intents to one another: the probabilities for each intent add up to one:

$$\sum_{C_i \in \mathbf{C}} P(C_i|S) = P(C|S)$$

$$\sum_{C_i \in \mathbf{C}} P(C_i|S) = P(C_{\text{Outliers}}|S) + P(C_{\text{Clusters}}|S) + \ \ldots \ = 1$$

To compute $P(C_i|S)$ we can use Bayes theorem. $P(S|C_i)$ models how a particular intent explains the current selection of the analyst. It is scaled by the term $P(C_i)$, which is called the prior. It describes the probability of each intent, without considering additional information. Finally, $P(S)$ acts as a normalizing constant that ensures that the result is a probability. To make this equation computationally tractable, we make use of two observations. First, if we do not consider the order of selections, the problem that we are trying to solve is very similar to text classification. Our description of the analysts' selection is almost identical to a *bag-of-words (BOW)* model, which is often used in this domain. The difference is that typically, in text classification, the bag-of-words model describes the frequency of each word. In our method, the BOW model simplifies to a constant frequency of one if a point is part of the selection. Second, by assuming that each feature (selected point) is independent of another, we can compute $P(S|C_i)$ using the *naive Bayes* method. In particular, we use a *multinomial naive Bayes* classifier to compute the conditional probabilities. For each selection by the analyst, we train such a classifier on the output vector of each of the intents C_i. Given a selection S as an input, the classifier yields the corresponding probability. Our prediction is then the intent that maximizes this probability. Sometimes, selected points are not part of any of the training samples, which leads to zero probabilities for

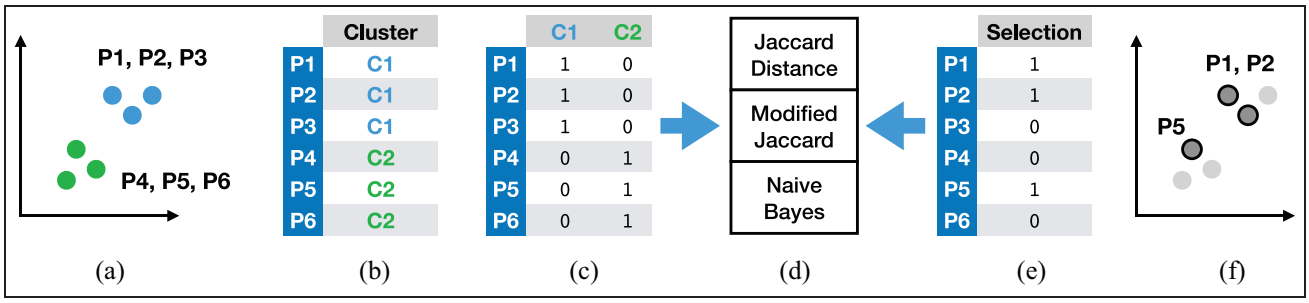


Figure 4. Overview of our method for ranking patterns for an analyst's selection based on a cluster example. (a) A dataset exhibits two clusters, shown in blue and green. (b) A clustering algorithm detects the clusters and assigns labels to the points. (c) We use one-hot encoding to transform the output of each algorithm into disjoint Boolean vectors. (f) An analyst's selection results in (e) another Boolean vector. (d) These Boolean vectors act as inputs to compute Jaccard indices and the naive Bayesian classifier, which are then used as scores for ranking.

Figure 5. The prediction interface shows ranked patterns based on the three scores. The "Category" prediction for a selection (orange points, rectangle brush) is shown in green in the scatterplot. Hovering over a row in the prediction interface shows a preview of the prediction. Clicking the row replaces the current selection with the predicted selection. The M, NP, and NS columns show the number of matching items (M), not predicted items (NP), and predicted but not selected items (NS). Hovering over a cell highlights the corresponding items in the scatterplot in green. The "Dims" column displays the dimensions considered for calculating a pattern. The "Probability" column encodes the probabilistic ranking. The provenance visualization (right) shows the steps that lead to the current selection and prediction. Insights (orange) are used to group and aggregate steps that lead to them. [Interactive Figure].

the intents. This is a common problem when using naive Bayes classifiers. We use *Laplacian smoothing* to avoid this effect.

Visualization and interaction design

In this section, we describe how we implemented our methods in an interactive visualization system and explain our visualization design decisions. The interface allows analysts to add scatterplots as desired. Categories can be visualized using glyph types (see Figure 4). We provide a paint-brush feature[2] with three brush sizes, rectangular brushes, and individual, click-based selections. The items in multiple rectangular brushes can be treated as unions or as intersections within or between multiple plots. Points that are selected individually or using the paint-brush are always treated as part of the intersection. The labels of the items in a selection are shown in a separate view (see Figure 5), where we also break down the number of items in the union and intersection of multiple brushes.

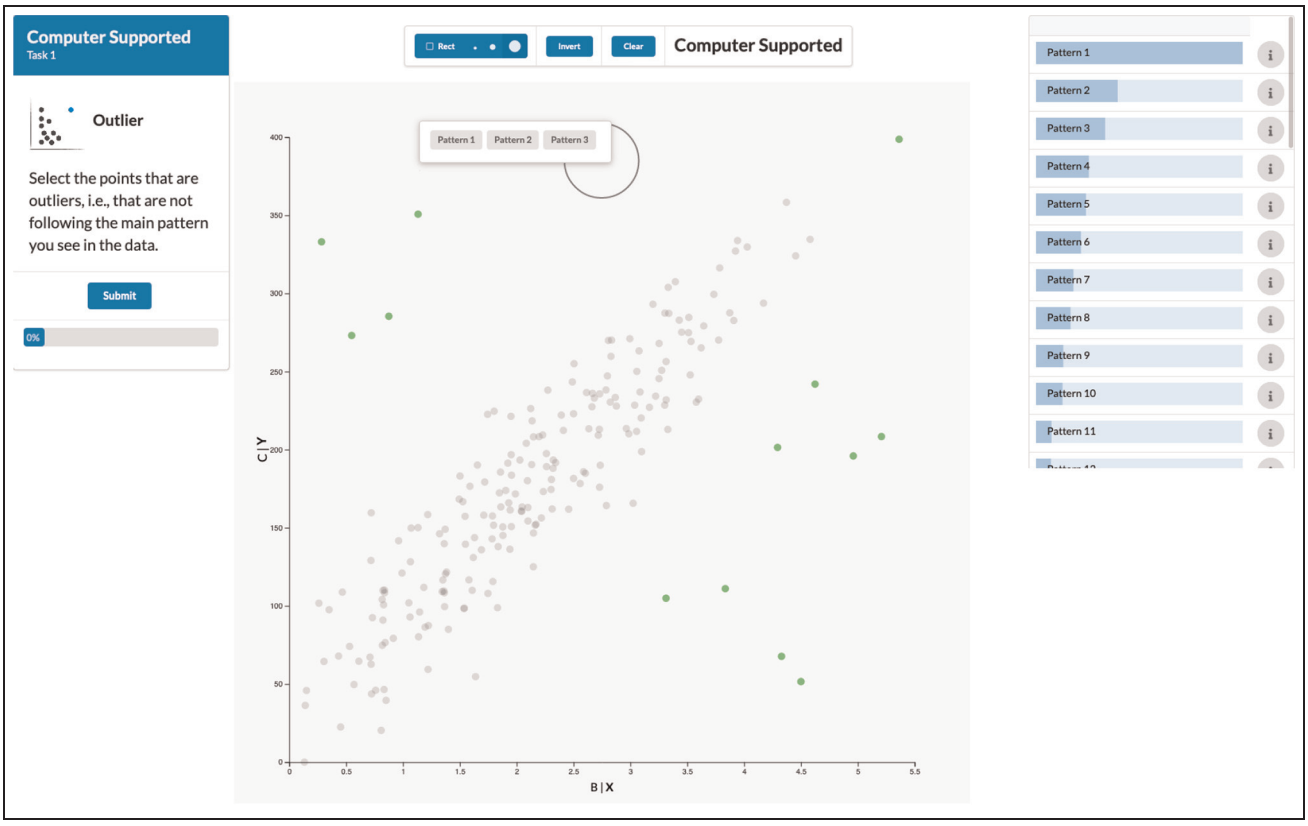


Figure 6. The study interface for the computer-supported condition for an outlier task. The user-driven condition was identical, except for the absence of the ranking on the right and the prediction pop-up. [Interactive Figure].

We designed the **prediction interface**, shown in Figure 5, as a ranked table with scores shown as bar charts.[59] Each predicted pattern is a row. Hovering over a prediction shows a preview, and clicking it replaces the selection with the prediction. The different scores are shown as bar charts in the columns as "Intent Rank" (the Jaccard index), "Auto-Complete Rank" (the Jaccard index modified to be sensitive to partial selections), and "Probability." The table can be sorted based on these scores. Other columns denote the "Matches (M)," that is, the number of points that the prediction and selection share; the "Not Predicted (NP)" items, that is, the number of items in the selection but not in the prediction; and the "Not Selected (NS)" items, that is, the number of items in the prediction but not in the selection. Combined with the similarity scores, these numbers give analysts a sense of how well each prediction matches the selection. Hovering over each of the M, NP, or NS numbers highlights the corresponding items in the scatterplot in green (see Figure 5).

Each prediction also shows on which dimension it was calculated (and their order) in the "Dims" column. We use short labels, which we replicate on the axes of the scatterplots to identify the dimensions. For range queries, we display the dimensions that are used in the decision tree.

When using **auto-complete**, analysts can sort by the auto-complete score. In addition, a pop-up appears right next to a selection in the scatterplot (Figure 6) showing the top three predictions for the current selection according to the auto-complete score. This popup can be used as a short-cut to complete selections.

To enable **reproducibility and recall**, a provenance graph is visualized in the history view (Figure 5).[39] Every persistent action, such as adding a plot or making a brush, is logged in the interface and can be retrieved at a later time. The provenance graph supports branching analysis histories. A prediction can be logged as a semantically meaningful insight, which can be supplemented with an annotation (see the annotation interface in Figure 4). Textual annotations are designed to connect the pattern-based intent in the data to the high-level, domain-specific goals. We use insights to group and aggregate the provenance graph: All actions that were in service of a particular insight are grouped together and can be collapsed. This grouping allows us to show a concise and semantically meaningful analysis history, while still storing a complete history of interactions. The example in Figure 5

shows one expanded group, indicated with an orange frame, and one aggregated insight in the inactive branch on the left.

The provenance graph contains all the information that is necessary to reconstruct the semantics of a selection, which means that a selection is not just a list of IDs, but contains, for instance, the explicit range query, or the cluster centroid and the algorithm configuration that can be used to reproduce a specific pattern on updated data. In the future, we plan to export the intents into machine-readable form, so that an interactive analysis and filtering session can be used, for example, in computational notebooks.

We chose to use scatterplots and point/brush-based selections for our prototype, because Scatterplots are a commonly used and widely understood visualization technique, and are well suited for brushing items. Combined with highlighting, scatterplots allow us to demonstrate analyst selections and system predictions clearly. We chose to focus on selections because they are not only important by themselves, but also are frequently precursors to more complex interactions like filtering, grouping, labeling, or segmentation. Our goal was to demonstrate the capturing of intent at the selection stage.

The visualization system described here is a technology demonstration that we developed with the goal of showing and validating the methods to detect and capture intent. We expect that a production system using our approach will use a simplified user interface for ranking intents, potentially closer to the simplified selection interface we used for our study (shown in Figure 6).

Results

We have implemented our prediction approach in an open-source prototype, and also have provided a variety of real and simulated datasets. An online version of the tool is available at [here], and the source code is available at [redacted for review].

We demonstrate our results through examples of brushes and the matching prediction. Figure 4, for example, shows a partially selected cluster that is also predicted as a cluster. Figure 5 shows a brush that closely matches a category and a range. Figure 6, which shows the study stimulus, gives an example of how our system can be used to auto-complete complex brushes. The plot, overall, shows a strong linear correlation between X and Y. Here, a participant has selected four points in a dataset (the four points in the top-left corner), and intended to select outliers. Our system recommended a list of predictions on the right and shows the top three predictions on the plot itself. Selecting a prediction in the pop-up or the ranked table on right reveals the points recommended for auto-completing the selection in green. Here, the first pattern matches the outliers above the main trend, and the third pattern matches all outliers, including those above and below the main trend. We provide further examples for all patterns in the Supplemental Material and refer to our prototype for an interactive demonstration.

Evaluation

We explored various approaches to evaluate our methods. First, we decided to focus our evaluation on the primary contribution of this paper: a set of methods to detect and capture analysts' pattern-based intents behind selections. We do not evaluate details of our visualization design, as they are meant to be technology demonstrations in service of our primary contribution. To validate our primary contribution, we have to determine how well our predicted intents match the mental models of analysts. To do this, we considered qualitative evaluation with experts using our system, case studies, usage scenarios, and quantitative evaluation. We ultimately chose a two-pronged approach: a demonstration of our results through a prototype with many preloaded datasets illustrated by a discussion in the previous section and the Supplemental Material, and a quantitative evaluation. For our quantitative study, we had to make trade-offs among ecological validity (realism), internal validity (isolating factors), and external validity (generalization), and we opted for a controlled crowdsourced study with a simple interface to have control over the factors (internal validity) and to include diverse participants, beyond just experts (external validity).[60]

We validate our predictions using auto-complete as an application scenario (see Figure 7(a)). This choice is pragmatic: Even though our ability to capture mental models is at least equally important to auto-complete, evaluating auto-complete is significantly easier, because it enables us to run a large-scale study and cover various types of patterns. We argue that successes in predicting correct pattern-based intents in an auto-complete scenario is transferable to capturing pattern-based intents for the purpose of reproducibility and reuse.

In our study, participants were instructed to select a specific pattern in two conditions: either using only brushes, or using brushes and auto-complete based on our prediction system (see Figure 7(b)). The study is designed to test the validity of our approach using two primary measures: **accuracy** and **match between intended and predicted pattern**.

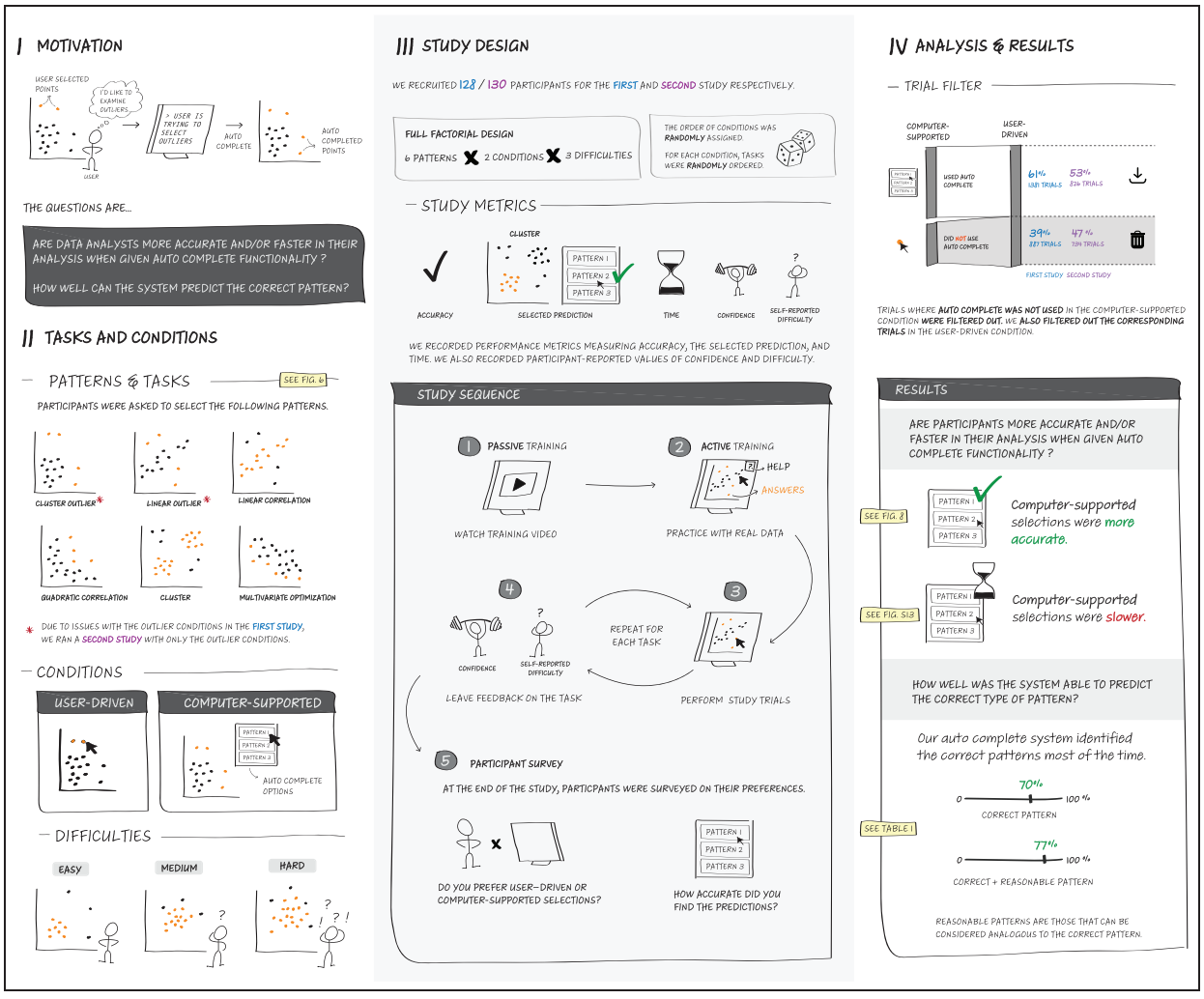


Figure 7. Data comic[61] showing the (a) motivation, (b) tasks and conditions, (c) study design, (d) analysis and results of our study.

We chose a subset of our patterns: correlations (linear and quadratic), outliers, clusters, and multivariate optimization. We excluded ranges, since they cannot be used for auto-complete, and categories, since selecting elements belonging to categories would be tedious in our system without auto-complete, and yet an alternative user interface design that enables participants to explicitly select categories would solve that problem trivially.

We also describe the study in a data comic, shown in Figure 7, because data comics are an effective way to communicate the complex procedures of a study.[61]

Procedure

We used a within-subjects design for two conditions: **user-driven**, using only manual brushes, and **computer supported**, which adds a simplified version of our prediction interface. Participants were instructed to select points that belong to a specified pattern. The interface, shown in Figure 6, was simplified to show only the rankings tailored to auto-complete. The names of the predicted patterns were not shown to avoid biasing participants. To counterbalance any learning effects, the conditions were assigned in random order, the task order in each condition was randomized, and the datasets were randomized. We recruited 128 participants for the study on Prolific, a crowdsourcing platform with a research focus. Based on completion times of pilot experiments, each participant was paid $6.25 USD, for an estimated duration of 25 min, resulting in an hourly rate of about $15 USD. All participants viewed and agreed to an IRB-approved consent form. To be

eligible, participants had to use a laptop or desktop device and either the Chrome or Firefox browsers.

Our procedure consisted of five phases (see Figure 7(c)) and followed guidelines on training participants for complex analysis tasks:[62] *passive training*, in the form of an 8-min video introducing the types of patterns and the interface; *active training*, where they had to complete representative tasks, but could use a help-feature to reveal the answer; *trials* in the two conditions; and a short *poststudy survey*. The full study with all phases is available online.

Our initial study had a negative result for outlier detection. Upon investigating the reasons behind this result, we found that our outlier prediction algorithms did not perform adequately. The algorithm we used at that time was Local Outlier Factor (LOF), which compares the local density of the object to density of its neighbors. This local density is given by the *k*-nearest neighbors algorithm. The algorithm is good at detecting local outliers, that is, data points that are some distance from a dense cluster are considered outliers, whereas a point that is far from a sparse cluster might be considered a part of the cluster, due to the cluster being sparse. We added another outlier detection algorithm, DBSCAN, to the interface. DBSCAN clusters the given data into density-based clusters and marks the points lying in low density region as outliers, which ensures that points that are outliers with respect to the entire dataset are correctly marked as outliers most of the time. We re-ran the study only for the outlier task. We edited the instructional video to remove explanations on other tasks; otherwise, the procedure remained the same. We recruited 130 participants for the revised outlier-only study. Participants were paid $3.52 for an estimated study duration of 14 min. For transparency, we report on both our original outlier results (referred to as "outlier old" going forward) and the revised outlier condition ("outlier revised").

Data and tasks

We generated synthetic two-dimensional datasets with between 200 and 222 items for (1) linear correlations, (2) quadratic correlations, (3) outliers combined with a linear correlation, (4) outliers combined with a single cluster, (5) clustered datasets with three or four clusters, and (6) datasets for multivariate optimizations, each in three levels of difficulty: easy, medium, and hard. The levels of difficulty were driven by how apparent a pattern is. For example, an easy clustering dataset had fully separated clusters, whereas a hard dataset had clusters that significantly overlap. We generated two variations of each combination (to be used in the different experimental conditions), for a total of 36 datasets for the study and 6 datasets for training tasks. For each dataset, we **generated ground-truth through human labeling**. Patterns such as clusters or outliers can be ambiguous, and our goal was to match the human perception of those patterns. Hence, we chose to ask expert coders to label the datasets. Our coders were five doctoral students in visualization not involved with this paper, with experience analyzing these patterns. We instructed them to carefully label each dataset for a specific pattern, with no algorithmic support. We then treated all points that 4–5 of our coders selected as correct, the points that 2–3 coders selected as ambiguous (neither correct nor incorrect), and the points that only a single or no coder selected as incorrect. The Supplemental Material contains images of the datasets, the ground-truth labels, and the code used to generate them.

The **tasks** instructed participants to select one of the patterns they learned about during training. As an example, for outliers, the prompt was: "Select the points that are outliers, that is, that are not following the main pattern you see in the data" (see Figure 6). For clusters, a specific cluster was marked in the plot with a red cross, and the prompt was "Select the points which belong to cluster centered around the cross."

Measures

We **measured accuracy, time to completion, the type and rank of a predicted pattern chosen by a participant, and survey responses**. After each question, we also elicited confidence and perceived difficulty on a five-point Likert scale and asked for comments. We also logged detailed interactions in a provenance graph. We **calculated the accuracy** of the participant's responses by using the **Jaccard index** of the response overlapping with the ground-truth, where we first removed the ambiguous points (hence, selecting ambiguous points neither benefits nor penalizes a score). For the time measures, we subtracted the times where the browser window showing the study was inactive. The final survey asked about the satisfaction with different features, and experience with visualization and statistics. Demographic data is provided through Prolific participant profiles.

Pilots, analysis, and experiment planning

We conducted several tests and pilots to evaluate tasks, system usability, data collection modalities, measures, and our procedure. We estimated the number of participants required to uncover effects based on a pilot run on Prolific with 10 participants. We used a power analysis to estimate the variance in our measures, which we

combined with our observed means to estimate the number of trials required. Due to the limitations of null hypothesis significance testing, we base our analysis on best practices for fair statistical communication in HCI[63] by reporting confidence intervals and effect sizes. We compute **95% bootstrapped confidence intervals**[64] and **effect sizes** using Cohen's d to indicate a standardized difference between two means. For the accuracy values, we also supplement our analysis by including p-values from Wilcoxon signed-rank tests (given the non-normal distributions of our data and the within-subjects design). We consider a Bonferroni-corrected threshold for significance of $p = 0.0083$.

Expectations

We expected that accuracy would be higher using computer-supported mode for the medium and hard datasets, and that accuracy would be about the same and consistently very high with the easy datasets. We assumed that the value of the prediction system would be greater on ambiguous patterns, and that obvious patterns would be easy to select manually, given the brushing tools we provided. We also expected that participants would perceive predictions as accurate and the interface as user-friendly, and they would prefer the computer-supported mode. Finally, we initially also expected computer-supported mode to be faster, but we realized during testing and pilots that this would unlikely be the case.

Results

The original study had 128 participants and the follow-up outlier study had 130 participants. After reviewing the provenance data using the reVISit system,[65] we realized that participants sometimes chose not to use predictions in the computer-supported condition. Since our goal was to measure the effects of using predictions, we removed trials that were not completed using predictions in the computer-supported mode. To avoid biasing our data by removing low-effort results in one condition, we also always removed the equivalent trial in the user-driven mode. We include data for all trials in our Supplemental Material. Based on these criteria, we retained 1381 of 2268 trials in each of the computer-supported and user-driven conditions (826 of 1560 for the second study). Hence, when auto-complete was available, participants chose to use it in 61% of cases (53% for the second study). We argue that the removal of these trials is justified and even necessary, but this argument leads to the question of why predictions were not used in many cases. We do not have definitive answers for that question, since conducting follow-up interviews with crowd participants is not possible. We do believe, however, that skipping the prediction interface is a sign of a crowd participant minimizing their effort, which is well known to be a challenge in crowdsourced studies.[66]

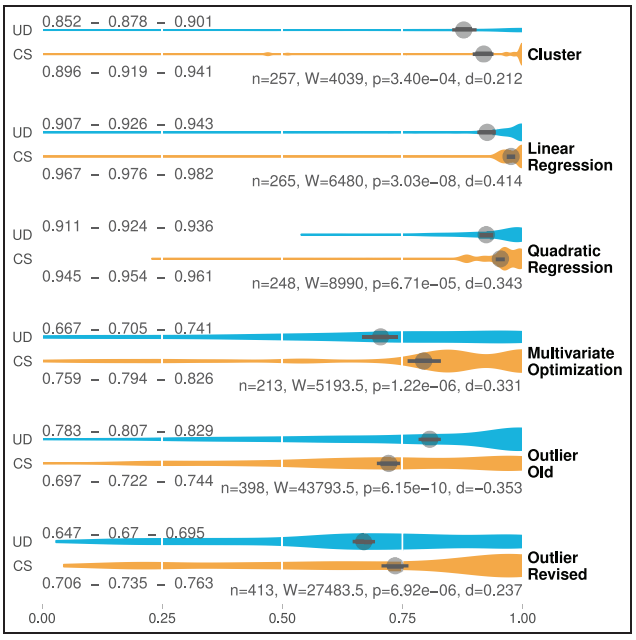


Figure 8. Task-specific accuracy shown as medians and 95% confidence intervals on a scale of 0–1. Blue (UD) encodes the user-driven condition, orange (CS) the computer-supported condition. Violin plots visualize the underlying distribution. The numbers on the left show the median and the extent of the 95% confidence interval. We also give the number of trials per condition for each task (n), Cohen's d for effect sizes (d), and p-values. All differences are significant. Note that the number of trials varies due to our exclusion criteria, and that the outlier tasks have higher numbers of trials as they group multiple outlier configurations (outliers on top of clusters, regressions, etc.) Also note that "outlier old" shows the result of our original study, whereas "outlier revised" shows the result of the separate study with an improved outlier detection algorithm.

We analyze easy, medium, and hard tasks together, resulting in 1785 valid trials across both studies. Figure 8 summarizes our main results. Accuracy and speed for every task are shown individually in Supplemental Figure S21 and Figure S22. We also break down results by levels of difficulty (see Supplemental Figures S14, S15, and S16). Accuracy was fairly high in both conditions for clusters, linear regression, and quadratic regression (median of 84%–98%), with a **small to medium, significant effect**

showing higher accuracy in the computer-supported condition. The computer-supported clustering condition shows a small "bump" at an accuracy of around 0.5. Analysis of provenance data has revealed that this bump is due to one of the clustering predictions aggregating two ground-truth clusters into one.

Overall accuracy for the multivariate optimization task was lower, with **accuracy in the computer-supported condition being significantly higher, with a small to medium effect size**. Interestingly, many of our coders omitted points that are contained in the formal definition of a skyline, resulting in a "bump" of accuracy scores at around 0.85, representing participants who have selected the formally correct skyline as recommended by the algorithm.

The accuracy for our original outliers condition ("outlier old," in Figure 8) was significantly lower in the computer-supported condition than in the user-driven condition. Inspection of the provenance data revealed that in many cases, applying a prediction for outliers made user selections worse. As previously discussed, we re-ran our study using a different outlier prediction algorithm (outliers as reported by DBSCAN). In the second study, we saw **significantly higher accuracy for computer-supported mode**, although overall accuracy had gone down. The reduced overall accuracy could be caused by reduced learning effects in the study with fewer tasks.

The difference in accuracy in favor of the computer-supported condition was more pronounced in medium and hard tasks. The accuracy in easy tasks was similar for both conditions (see Supplemental Figures S14, S15, and S16).

Our exit survey revealed that participants generally found predictions accurate (average score of 3.6 on a five-point Likert scale) and helpful (average score of 3.8 on a 5-point Likert scale). In terms of the interaction choices for selections, the paintbrush selection was rated more helpful (average 4.5/5) than the rectangular brush (average 2.3/5) and individual point selection (average 3.3). Every task was followed by a mini survey in which participants reported their confidence in their selection and self-reported the difficulty of the task. Confidence and difficulty were reported on 1–5 scale with 1 being confident, 5 being not confident, 1 being easy, and 5 being difficult. Confidence was higher and difficulty was reported lower for the computer-supported condition for all tasks except outliers, where they were about the same, suggesting that participants trusted the predictions when they matched their mental model.

We also analyzed whether the **type of predictions chosen by participants matched the patterns they were instructed to select**, which is a useful metric to judge the quality of our predictions and rankings. We see a strong overlap between prediction and target pattern (see Table 1); **participants selected the right type of pattern 70% of the time from our predictions**, and used the correct pattern or a reasonable substitute (e.g. "outside linear regression" instead of "outlier") 77% of the time. There were variations between patterns: clusters were almost perfectly matched, whereaas both regression patterns were less frequently correctly matched. Notably, quadratic and linear regression were frequently substituted, and non-outliers were also frequently chosen for regression tasks.

Time to completion was generally slower by 3–12 s (with completion times ranging from 21 to 37 s on average) for the computer-supported condition (see Supplemental Figure S13). Given the higher accuracy – overall, the median accuracy for computer supported mode was 3%–9% higher (excluding the old outlier condition) – but the slower response times, it is worth asking whether **this a trade-off worth making.** In retrospect, the longer response times for computer-supported work make sense. Previous work by Saket et al.[23] has shown that task completion times in multiparadigm interfaces can be higher compared to a single paradigm interface. However, Saket et al. also argue that optimizing efficiency is not a suitable goal in many contexts and that multiparadigm tools can make analysts think more carefully. How meaningful are 10 s of an analyst's time when trying to understand an important dataset? We argue that accuracy by itself is much more important than time, when the time difference is a few seconds, for most analysis scenarios. Furthermore, for our study specifically, we were able to show not only that the accuracy in computer-supported mode was higher, but also that we were able to correctly predict patterns based on participant selections in most cases, which has the benefit that this data is now available as structured information that can be leveraged for reproducibility and reuse.

Study discussion

Overall, our expectations were verified: accuracy was consistently higher in the computer-supported condition, and we were able to correctly predict a large percentage of patterns. In terms of predicted patterns, quadratic and linear regression showed lower accuracy in predicting correct patterns, even when including nonoutliers as a reasonable substitute. This result is likely due to the linear and quadratic regression algorithms using our thresholding being quite similar. Creating an umbrella intent "regression" would be one possibility to address this problem.

Table 1. To analyze the matches between patterns, participants were instructed to select (columns) and patterns they chose from our set of predictions.

Pattern asked for in tasks	Cluster (*N* = 257)	Linear regression (*N* = 265)	Quadratic regression (*N* = 248)	Multivariate optimization (*N* = 213)	Outlier old (*N* = 398)	Outlier revised (*N* = 520)	Average
Pattern predicted							
Cluster	**256 (223)**	6 (6)	82 (78)	13 (2)	43 (25)	44 (5)	–
Linear regression – within	0 (0)	**136 (136)**	54 (54)	1 (0)	4 (1)	1 (0)	–
Quadratic regression – within	0 (0)	82 (82)	**84 (84)**	0 (0)	2 (1)	0 (0)	–
Multivariate optimization	0 (0)	0 (0)	0 (0)	**175 (163)**	21 (10)	22 (22)	–
Linear regression – outside	0 (0)	1 (0)	0 (0)	0 (0)	**0 (0)**	**60 (44)**	–
Quadratic regression – outside	0 (0)	0 (0)	0 (0)	2 (0)	**0 (0)**	**13 (0)**	–
Non-outlier	1 (0)	**40 (39)**	**28 (28)**	0 (0)	1 (1)	0 (0)	–
Outlier	0 (0)	0 (0)	0 (0)	22 (2)	**327 (174)**	**380 (178)**	–
Correct pattern	99% (87%)	51% (51%)	34% (34%)	82% (76%)	82% (44%)	73% (34%)	70% (54%)
Correct + reasonable pattern	99% (87%)	66% (66%)	45% (45%)	82% (76%)	82% (44%)	87% (48%)	77% (61%)

The values in the cells show the number of times participants selected a certain predicted pattern for a pattern they were asked to select. For example, the first column, "Cluster" contains responses for all trials that instructed participants to select a cluster. The first row, also "Cluster," shows all trials where participants chose a cluster pattern from the predictions. Consequently, the first cell shows that when asked to select a "Cluster," in 256 trials, participants have also selected a predicted pattern of type "Cluster." Cells containing precise matches are highlighted in green. Cells highlighted in yellow show matches with analogous patterns. For example, the "outside linear regression" pattern is a reasonable substitute for the "outlier" pattern. The numbers in parentheses show how frequently a response resulted in a correct solution (assuming correct to be an accuracy >0.75). The bottom two lines give percentages for the correctly identified patterns and correct + analogous patterns combined. Again, the number of responses with an accuracy >0.75 is in parentheses. Overall, correct patterns were used frequently for clusters, multivariate optimization, and outliers. Results for linear and quadratic regression show a frequent mix-up with the other type of regression pattern, and for quadratic regression also with clusters. Interestingly, these mismatched patterns do not negatively influence accuracy. .

We also believe that the matches between patterns and predictions will be much higher in practice when labels for the patterns are shown, so that analysts can pick the pattern that corresponds to their mental model.

Although we were able to show that we can successfully predict pattern-based intents for auto-complete, the question remains how useful real-life analysts will find the ability to track semantically meaningful pattern-based intents. To answer this question, we plan to develop our prototype system into a visual exploration tool that enables actions derived from selections, such as filters and groupings, and provides features such as sharing, replaying, and exporting the analysis process into other pipelines or tools such as Jupyter Notebooks. We can then design a more comprehensive evaluation strategy that can validate the efficacy of this system with regard to reproducibility and reusability.

Discussion

In this work, we demonstrate a method for semi-automatically detecting and capturing analysts' pattern-based intents. Detecting intents is useful for two scenarios: to auto-complete selections and to be able to semi-automatically record semantically rich insights in provenance data and therefore make visual analysis processes reproducible and justifiable. By capturing pattern-based intents, we can, for example, more easily create curated analysis stories by leveraging ideas from prior work on using provenance information to create interactive data stories.[45] The capability to capture pattern-based intents opens up numerous other prospects as well.

Integration in computational workflows and analysis reuse

Our interviews show that analysts frequently use scatterplots in combination with statistical modeling tools and computational notebooks, such as R-Markdown or Jupyter notebooks. Having semantically meaningful intents available means that we can generate robust analysis scripts based on interactive visualization, supporting more automatic computational workflows. For example, if an analyst uses our tool to select a specific cluster for downstream analysis, we will be able to generate code that will select this cluster even for updated data.

Learning from interaction

Through large-scale capturing of intents, we can empirically learn patterns that analysts select to further improve our predictions. Such a system could dynamically "auto-correct" analysis and allow large-scale feedback on the usefulness and effectiveness of various features within complex tools. For instance, a software tool with a diverse set of users and skill levels would allow intent to be trained on experienced users so that novices are guided quickly toward effective strategies.[14]

Generalization to other visualization techniques and data types

We chose to limit ourselves to scatterplots and tabular data because we believe that these are important cases that can be used to demonstrate the feasibility of our approach. There are numerous extensions and generalizations of our work, ranging from implementing more brushing tools, such as lasso selections, to allowing analysts to filter datasets. We argue that our framework could be extended to other visualization techniques, such as parallel coordinates, histograms, or tabular visualizations[67] with small adaptions. Other visualization techniques could also provide additional clues we could use for predicting intents. For example, in a tabular visualization, the action of sorting a table is likely important to understand the intent of a subsequent selection. Other data types, such as time-series or network data, are likely amenable to the same approach, but would require identifying appropriate patterns and the corresponding algorithms.

Higher dimensionality

Although we allowed analysts to explore multiple two-dimensional views, building a mental model of high-dimensional data can be difficult. A potential solution to this problem is dynamic dimensionality reduction. That is, given points already selected, the system could dynamically adjust a linear projection (e.g. PCA) to best capture those datasets in a 1-, or 2-dimensional subspace. Alternatively, given more complex selections, like clusters of relevant points, dimensionality reduction can use techniques such as Latent Discriminant Analysis to find the best linear projection to separate the clusters. Another approach is to label pairs of points that should be close (or far). Using these pairs, a similarity learning method could provide the best linear projection that satisfies those constraints. An intent-driven tool could suggest the most informative point-pairs to label.

Scalability

Our current system recalculates the predictions every time an analyst interacts with the system. This delay in

the prediction mechanism can be prohibitive for large datasets, large combinations of dimensions, or more parameterizations for algorithms. The prediction mechanism's two main phases are to run the machine learning algorithms on the datasets and to rank the results based on an analyst's selection. Only the second step has to be done in real time. The first, computationally expensive, step can be done once, on data upload, as an offline step. Precomputing would also allow us to include a larger combination of dimensions and add more algorithms and parameterization without substantially adding to the prediction time.

Active pattern exploration

Instead of just passively suggesting which pattern matches to a selection, we could also suggest various patterns in the data set as possible aspects to explore at the beginning of an analysis. In this way, analysts could, for example, see all computed skylines without ever using selections. The downside of such an approach is the potential for increased complexity in the UI: analysts would be confronted with many different analysis choices, and rankings or suggestions would be difficult to achieve without prior input from the analyst.

Multiverse analysis

As in any multi-step analysis process, analysts must make choices about their analysis paths, leaving other reasonable paths behind. As we capture analysis paths explicitly, it would be intriguing to also explore different analysis paths from the multiverse and visualize the results of these alternatives using a framework like Boba.[68]

Limitations

Even in a simple scatterplot environment, our work has identified numerous complexities. When more than four dimensions of a dataset are relevant in the exploration, the combinatorial complexity of all the possible intents we model is significant. One potential solution is to automatically filter entire classes of intents so that not all of them need to be explicitly explored.

Showing many scatterplots also raises the problem of fitting them visually on the screen. Predicting intents in higher dimensions based on selections in 2D scatterplots is tricky, because scatterplots do not provide a good visual representation of high-dimensional patterns. We plan on addressing this problem by providing additional visualizations for visualizing high-dimensional data, such as parallel coordinate plots or heat maps, or by using dimensionality reduction.

Our tool currently does not handle missing data. When working with our collaborators, we frequently encountered datasets that were generally well suited to our approach but contained invalid or missing cells. On the front-end, we plan to provide separate views for items with missing data. On the back-end, appropriate interpolation and fitting strategies could be a solution.

Our current approach to parameter space exploration is naive. We could improve our prediction by evaluating our classifications using methods such as silhouette analysis for clustering and varying the parameters accordingly.

In some cases, meaningful selections might not correspond to predicted patterns, yet our ranking system will still recommend a pattern, although with low scores. We considered including a "no pattern" prediction in the ranking, but ultimately decided against it, since it would be difficult to rank in either of our ranking frameworks. However, analysts can explicitly record "custom insights" that are not based on any ranked pattern to account for same.

As analysts interact with a visualization over a longer period of time, the provenance graph keeps growing. The Trrack library[39] can demonstrably handle a large number of actions, but the provenance visualization will become hard to navigate. As a partial remedy, we group the actions in a provenance graph when an insight has been generated, which allows the collapse of the provenance visualization to give a higher level overview. We can improve this approach by allowing analysts to manually group provenance actions and collapse the tree further. Finally, search functionality on the provenance graph may be a scalable solution to the problem.[47]

Conclusion

In this paper, we introduce the first approach to predict, capture, and annotate pattern-based intents of analysts as they interact with data in a scatterplot. We use a mixed-initiative approach, leveraging data mining methods to identify patterns in datasets, ranking potential matches based on selections, and allowing analysts to specify which (if any) of the predicted intents fit their actual intents. We discuss two application scenarios: auto-completing selections and increasing reproducibility. We believe that our work will form the foundation of many future projects. Immediate next steps are the application to different visualization techniques and data types. Other prospects include learning from interactions and integrating the output

of interactions in visualizations into computational workflows.

Acknowledgement

We thank the domain experts we interviewed for their time and their willingness to provide datasets, and Lane Harrison and members of the Visualization Design Lab for feedback.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: We gratefully acknowledge funding by the National Science Foundation (IIS 1751238) and by the Deutsche Forschungsgemeinschaft (251654672-TRR 161).

ORCID iDs

Kiran Gadhave https://orcid.org/0000-0001-6916-2583
Alexander Lex https://orcid.org/0000-0001-6930-5468

Supplemental material

Supplemental material for this article is available online.

References

1. Heer J and Shneiderman B. Interactive dynamics for visual analysis. *Commun ACM* 2012; 55(4): 45–54.
2. Becker RA and Cleveland WS. Brushing scatterplots. *Technometrics* 1987; 29(2): 127–142.
3. Brehmer M and Munzner T. A multi-level typology of abstract visualization tasks. *IEEE Trans Vis Comput Graph* 2013; 19(12): 2376–2385.
4. Rind A, Aigner W, Wagner M, et al. Task cube: a three-dimensional conceptual space of user tasks in visualization design and evaluation. *Inf Vis* 2016; 15(4): 288–300.
5. Karer B, Hagen H and Lehmann DJ. Insight beyond numbers: the impact of qualitative factors on visual data analysis. *IEEE Trans Vis Comput Graph* 2021; 27(2): 1011–1021.
6. Myers BA. Creating User Interfaces by Demonstration. PhD Thesis, University of Toronto, Canada, 1987.
7. Gotz D and Zhou MX. Characterizing users' visual analytic activity for insight provenance. *Inf Vis* 2009; 8(1): 42–55.
8. Nguyen PH, Xu K, Wheat A, et al. SensePath: understanding the sensemaking process through analytic provenance. *IEEE Trans Vis Comput Graph* 2016; 22(1): 41–50.
9. Dimitriadou K, Papaemmanouil O and Diao Y. Explore-by-example: an automatic query steering framework for interactive data exploration. In: *Proceedings of the 2014 ACM SIGMOD international conference on management of data. SIGMOD'14*, Snowbird, UT, 22–27 June 2014, pp.517–528. New York: ACM.
10. Cavallo M and Demiralp C. Clustrophile 2: guided visual clustering analysis. *IEEE Trans Vis Comput Graph* 2018; 25(1): 267–276.
11. Dou W, Jeong DH, Stukes F, et al. Recovering reasoning processes from user interactions. *IEEE Comput Graph Appl* 2009; 29(3): 52–61.
12. North C, Chang R, Endert A, et al. Analytic provenance: process + interaction + insight. In: *CHI'11 extended abstracts on human factors in computing systems. CHI EA'11*, Vancouver, BC, Canada, 7–12 May 2011, pp.33–36. New York: ACM.
13. Brown ET, Ottley A, Zhao H, et al. Finding Waldo: learning about users from their interactions. *IEEE Trans Vis Comput Graph* 2014; 20(12): 1663–1672.
14. Ceneda D, Gschwandtner T and Miksch S. A review of guidance approaches in visual data analysis: a multifocal perspective. *Comput Graph Forum* 2019; 38(3): 861–879.
15. Ottley A, Garnett R and Wan R. Follow the clicks: learning and anticipating mouse interactions during exploratory data analysis. *Comput Graph Forum* 2019; 38(3): 41–52.
16. Steichen B, Carenini G and Conati C. User-adaptive information visualization: using eye gaze data to infer visualization tasks and user cognitive abilities. In: *Proceedings of the 2013 international conference on intelligent user interfaces. IUI'13*, Santa Monica, CA, 19–22 March 2013, pp.317–328. New York: ACM.
17. Gingerich MJ and Conati C. Constructing models of user and task characteristics from eye gaze data for user-adaptive information highlighting. In: *Twenty-Ninth AAAI conference on artificial intelligence*, Austin, Texas, 25 January 2015, pp.1728–1734. Menlo Park, CA: AAAI Press.
18. Monadjemi S, Garnett R and Ottley A. Competing models: inferring exploration patterns and information relevance via Bayesian model selection. *IEEE Trans Vis Comput Graph* 2021; 27(2): 412–421.
19. Battle L, Chang R and Stonebraker M. Dynamic prefetching of data tiles for interactive visualization. In: *Proceedings of the 2016 international conference on management of data. SIGMOD'16*, San Francisco, California, 26 June–1 July 2016, pp.1363–1375. New York: ACM.
20. Mackinlay J, Hanrahan P and Stolte C. Show Me: automatic presentation for visual analysis. *IEEE Trans Vis Comput Graph* 2007; 13(6): 1137–1144.
21. Tory M and Setlur V. Do what i mean, not what i say! design considerations for supporting intent and context in analytical conversation. In: *2019 IEEE conference on visual analytics science and technology (VAST)*, Vancouver, BC, Canada, 20–25 October 2019, pp.93–103. New York: IEEE.
22. Saket B, Kim H, Brown ET, et al. Visualization by demonstration: an interaction paradigm for visual data

exploration. *IEEE Trans Vis Comput Graph* 2017; 23(1): 331–340.
23. Saket B, Jiang L, Perin C, et al. Liger: combining interaction paradigms for visual analysis. arXiv:190708345 [cs] 2019; 1907.08345.
24. Demiralp Ç, Haas PJ, Parthasarathy S, et al. Foresight: recommending visual insights. *Proc VLDB Endow* 2017; 10(12): 1937–1940.
25. Fan C and Hauser H. Fast and accurate CNN-based brushing in scatterplots. *Comput Graph Forum* 2018; 37(3): 111–120.
26. Martin AR and Ward MO. High dimensional brushing for interactive exploration of multivariate data. In: *Proceedings of the IEEE conference on visualization (Vis'95)*, Atlanta, GA, 29 October–3 November 1995, pp.271–278. Washington, DC: IEEE Computer Society Press.
27. Derthick M, Kolojejchick J and Roth SF. An interactive visual query environment for exploring data. In: *Proceedings of the 10th annual ACM symposium on user interface software and technology. UIST'97*, Banff, AB, Canada, 14–17 October 1997, pp.189–198. New York: ACM.
28. Shneiderman B. Dynamic queries for visual information seeking. *IEEE Softw* 1994; 11(6): 70–77.
29. Heer J, Agrawala M and Willett W. Generalized selection via interactive query relaxation. In: *Proceedings of the SIGCHI conference on human factors in computing systems. CHI'08*, Florence, Italy, 5–10 April 2008, pp.959–968. New York: ACM.
30. Su SL, Paris S and Durand F. QuickSelect: history-based selection expansion. In: *Proceedings of graphics interface 2009. GI'09*, Kelowna, BC, Canada, 25–27 May 2009, pp.215–221. Toronto, ON: Canadian Information Processing Society.
31. Xiao L, Gerth J and Hanrahan P. Enhancing visual analysis of network traffic using a knowledge representation. In: *2006 IEEE symposium on visual analytics science and technology*, Baltimore, MD, 31 October–2 November 2006, pp.107–114. New York: IEEE.
32. Ragan ED, Endert A, Sanyal J, et al. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE Trans Vis Comput Graph* 2016; 22(1): 31–40.
33. Freire J, Koop D, Santos E, et al. Provenance for computational tasks: a survey. *Comput Sci Eng* 2008; 10(3): 11–21.
34. Deelman E, Gannon D, Shields M, et al. Workflows and e-science: an overview of workflow system features and capabilities. *Future Gener Comput Syst* 2009; 25(5): 528–540.
35. Parker SG and Johnson CR. SCIRun: a scientific programming environment for computational steering. In: *Proceedings of the ACM/IEEE conference on supercomputing (SC'95)*, San Diego, CA, 8 December 1995, p.52. New York: ACM.
36. Bavoil L, Callahan SP, Scheidegger C, et al. VisTrails: enabling interactive multiple-view visualizations. In: *Proceedings of the IEEE conference on visualization (VIS'05)*, Minneapolis, MN, 23–28 October 2005, pp.135–142. New York: IEEE.
37. Dunne C, Henry Riche N, Lee B, et al. GraphTrail: analyzing large multivariate, heterogeneous networks while supporting exploration history. In: *Proceedings of the SIGCHI conference on human factors in computing systems (CHI'12)*, Austin,Texas, 5–10 May 2012, pp.1663–1672. New York: ACM.
38. Yu B and Silva CT. VisFlow – web-based visualization framework for tabular data with a subset flow model. *IEEE Trans Vis Comput Graph* 2017; 23(1): 251–260.
39. Cutler ZT, Gadhave K and Lex A. Trrack: a library for provenance tracking in web-based visualizations. In: *IEEE visualization conference (VIS)*, Salt Lake City, UT, 25–30 October 2020, pp.116–120. New York: IEEE.
40. Heer J, Mackinlay J, Stolte C, et al. Graphical histories for visualization: supporting analysis, communication, and evaluation. *IEEE Trans Vis Comput Graph* 2008; 14(6): 1189–1196.
41. Kadivar N, Chen V, Dunsmuir D, et al. Capturing and supporting the analysis process. In: *2009 IEEE symposium on visual analytics science and technology*, Atlantic City, NJ, 12–13 October 2009, pp.131–138. New York: IEEE.
42. Kreuseler M, Nocke T and Schumann H. A history mechanism for visual data mining. In: *Proceedings of the IEEE symposium on information visualization (InfoVis'04)*, Austin, Texas, 10–12 October 2004, pp.49–56. New York: IEEE.
43. Shrinivasan YB and van Wijk JJ. Supporting the analytical reasoning process in information visualization. In: *Proceedings of the SIGCHI conference on human factors in computing systems. CHI'08*, Florence, Italy, 5–10 April 2008, pp.1237–1246. New York: ACM.
44. Streit M, Schulz HJ, Lex A, et al. Model-driven design for the visual analysis of heterogeneous data. *IEEE Trans Vis Comput Graph* 2012; 18(6): 998–1010.
45. Gratzl S, Lex A, Gehlenborg N, et al. From visual exploration to storytelling and back again. *Comput Graph Forum* 2016; 35(3): 491–500.
46. Fujiwara T, Crnovrsanin T and Ma KL. Concise provenance of interactive network analysis. *Vis Inform* 2018; 2: 213–224.
47. Stitz H, Gratzl S, Piringer H, et al. KnowledgePearls: provenance-based visualization retrieval. *IEEE Trans Vis Comput Graph* 2018; 25(1): 120–130.
48. Camisetty A, Chandurkar C, Sun M, et al. Enhancing web-based analytics applications through provenance. *IEEE Trans Vis Comput Graph* 2018; 25(1): 131–141.
49. Groth DP and Streefkerk K. Provenance and annotation for visual exploration systems. *IEEE Trans Vis Comput Graph* 2006; 12(6): 1500–1510.
50. Wright W, Schroh D, Proulx P, et al. The sandbox for analysis: concepts and methods. In: *Proceedings of the SIGCHI conference on human factors in computing systems*, Montréal, Québec, 22–27 April 2006, pp.801–810. New York: ACM.
51. Wohlfart M and Hauser H. Story telling for presentation in volume visualization. In: *Proceedings of the symposium on visualization (EuroVis'07)*, Norrköping, Sweden, 23–25 May 2007, pp.91–98. Goslar, Germany: Eurographics Association.

52. Eccles R, Kapler T, Harper R, et al. Stories in GeoTime. *Inf Vis* 2008; 7(1): 3–17.
53. Kwon BC, Stoffel F, Jäckle D, et al. VisJockey: enriching data stories through orchestrated interactive visualization. In: *Poster compendium of the computation + journalism symposium*, New York, NY, 24–25 October 2014. New York, NY: The Brown Institute for Media Innovation, Pulitzer Hall, Columbia University.
54. Gotz D and Wen Z. Behavior-driven visualization recommendation. In: *Proceedings of the conference on intelligent user interfaces (IUI'09)*, Sanibel Island, Florida, 8–11 February 2009, pp.315–324. New York: ACM.
55. Amar R, Eagan J and Stasko J. Low-level components of analytic activity in information visualization. In: *Proceedings of the IEEE symposium on information visualization (InfoVis'05)*, Minneapolis, MN, 23–25 October 2005, pp.111–117. New York: IEEE.
56. Sarikaya A and Gleicher M. Scatterplots: tasks, data, and designs. *IEEE Trans Vis Comput Graph* 2018; 24(1): 402–412.
57. Borzsony S, Kossmann D and Stocker K. The skyline operator. In: *Proceedings 17th international conference on data engineering*, Heidelberg, Germany, 2–6 April 2001, pp.421–430. New York: IEEE.
58. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in python. *J Mach Learn Res* 2011; 12: 2825–2830.
59. Gratzl S, Lex A, Gehlenborg N, et al. LineUp: visual analysis of multi-attribute rankings. *IEEE Trans Vis Comput Graph* 2013; 19(12): 2277–2286.
60. Carpendale S. Evaluating information visualizations. In: Stasko JT, Fekete JD, North C, et al. (eds) *Information visualization: human-centered issues and perspectives*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp.19–45.
61. Wang Z, Ritchie J, Zhou J, et al. Data comics for reporting controlled user studies in human-computer interaction. *IEEE Trans Vis Comput Graph* 2021; 27(2): 967–977.
62. Nobre C, Wootton D, Harrison L, et al. Evaluating multivariate network visualization techniques using a validated design and crowdsourcing approach. In: *Proceedings of the 2020 CHI conference on human factors in computing systems*, Honolulu, HI, 25–30 April 2020, pp.1–12. New York: IEEE.
63. Dragicevic P. Fair statistical communication in HCI. In: Robertson J and Kaptein M (eds) *Modern statistical methods for HCI*. Cham: Springer International Publishing, 2016, pp.291–330.
64. Cumming G. *Understanding the new statistics: effect sizes, confidence intervals, and meta-analysis*. Oxfordshire, England: Routledge, 2013.
65. Nobre C, Wootton D, Cutler Z, et al. reVISit: looking under the hood of interactive visualization studies. In: *SIGCHI conference on human factors in computing systems (CHI) (to appear)*, Yokohama, Japan, 8–13 May 2021, pp.1–12. New York: ACM.
66. Borgo R, Lee B, Bach B, et al. Crowdsourcing for information visualization: promises and pitfalls. In: Archambault D, Purchase H and Hoßfeld T (eds) *Evaluation in the crowd. Crowdsourcing and human-centered experiments*. Cham: Springer, 2017, pp.96–138, Vol. 10264.
67. Furmanova K, Gratzl S, Stitz H, et al. Taggle: combining overview and details in tabular data visualizations. *Inf Vis* 2020; 19(2): 114–136.
68. Liu Y, Kale A, Althoff T, et al. Boba: authoring and visualizing multiverse analyses. *IEEE Trans Vis Comput Graph* 2021; 27(2): 1753–1763.