Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations

Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer

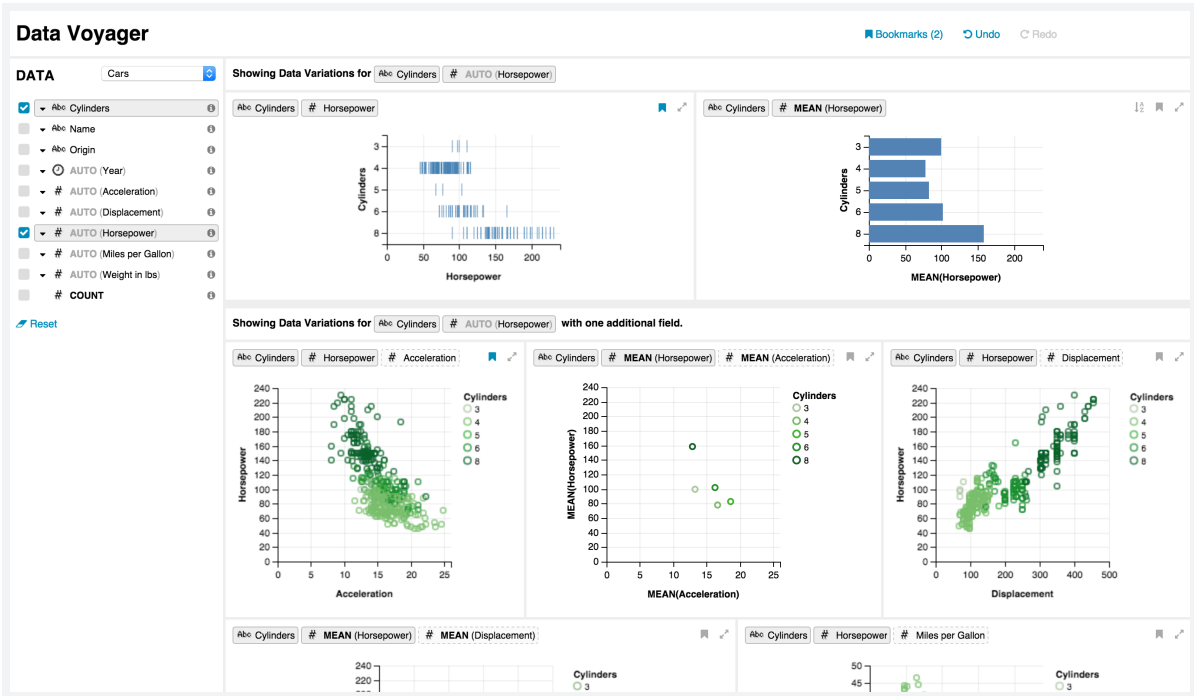

Fig. 1. Voyager: a recommendation-powered visualization browser. The schema panel (left) lists data variables selectable by users. The main gallery (right) presents suggested visualizations of different variable subsets and transformations.

Abstract—General visualization tools typically require manual specification of views: analysts must select data variables and then choose which transformations and visual encodings to apply. These decisions often involve both domain and visualization design expertise, and may impose a tedious specification process that impedes exploration. In this paper, we seek to complement manual chart construction with interactive navigation of a gallery of automatically-generated visualizations. We contribute Voyager, a mixed-initiative system that supports faceted browsing of recommended charts chosen according to statistical and perceptual measures. We describe Voyager's architecture, motivating design principles, and methods for generating and interacting with visualization recommendations. In a study comparing Voyager to a manual visualization specification tool, we find that Voyager facilitates exploration of previously unseen data and leads to increased data variable coverage. We then distill design implications for visualization tools, in particular the need to balance rapid exploration and targeted question-answering.

Index Terms—User interfaces, information visualization, exploratory analysis, visualization recommendation, mixed-initiative systems

1 Introduction

Exploratory visual analysis is highly iterative, involving both open-ended exploration and targeted question answering [16, 37]. Yet making visual encoding decisions while exploring unfamiliar data is non-trivial. Analysts may lack exposure to the shape and structure of their data, or begin with vague analysis goals. While analysts should typically examine each variable before investigating relationships between them [28], in practice they may fail to do so due to premature fixation on specific questions or the tedium of manual specification.

The primary interaction model of many popular visualization tools (e.g., [35, 44, 45]) is manual view specification. First, an analyst must *select variables* to examine. The analyst then may apply *data transformations*, for example binning or aggregation to summarize the data. Finally, she must *design visual encodings* for each resulting variable set. These actions may be expressed via code in a high-level language [44] or a graphical interface [35]. While existing tools are well suited to depth-first exploration strategies, the design of tools for breadth-oriented exploration remains an open problem. Here we focus on tools to assist breadth-oriented exploration, with the specific goal of promoting increased coverage of a data set.

To encourage broad exploration, visualization tools might automatically generate a diverse set of visualizations and have the user select

- *Kanit Wongsuphasawat, Dominik Moritz, Bill Howe, and Jeffrey Heer are with University of Washington. E-mail: {kanitw,domoritz,billhowe,jheer}@cs.washington.edu.*
- *Anushka Anand and Jock Mackinlay are with Tableau Research. E-mail: {aanand, jmackinlay}@tableau.com.*

Manuscript received 31 Mar. 2015; accepted 1 Aug. 2015; date of publication 20 Aug. 2015; date of current version 25 Oct. 2015. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org. Digital Object Identifier no. 10.1109/TVCG.2015.2467191

among them. However, for any given data table the choice of variables, transformations and visual encodings leads to a combinatorial explosion. Appropriate filtering and recommendation strategies are needed to prune the space and promote relevant views. Further, automation is unlikely to succeed on its own: as exploration proceeds, users will inevitably wish to focus on specific aspects of the data, requiring a browser that enables interactive steering of recommendations.

We present *Voyager*, a mixed-initiative system that couples faceted browsing with visualization recommendation to support exploration of multivariate, tabular data. Voyager exchanges specification for browsing, providing an organized display of recommended visualizations and enabling user input for both chart refinement and recommendation steering. To enable breadth-oriented exploration, Voyager privileges *data variation* (different variable selections and transformations) over *design variation* (different visual encodings of the same data). Underlying Voyager is the *Compass* recommendation engine, which enumerates, clusters and ranks visualizations according to both data properties and perceptual principles.

Voyager and Compass describe visualizations using *Vega-lite*, a new high-level specification language. Following in the footsteps of the Grammar of Graphics [44, 45] and Tableau's VizQL [35], Vega-lite provides a convenient formalism for enumeration and reasoning of visualization designs. It also enables hand-offs between different visualization tools (*e.g.,* for breadth- or depth-oriented exploration).

In this paper we describe Voyager's motivating design principles, interface design, and system architecture. We also present a controlled user study focused on exploratory analysis of previously unseen data. We compare Voyager with PoleStar, a state-of-the-art view specification tool modeled on Tableau. Through analysis of both user performance and preference ratings, we find that Voyager better facilitates initial exploration and leads to increased data variable coverage, while PoleStar is preferable for answering more specific questions. We discuss resulting implications for visualization tools, in particular the need to integrate rapid exploration and targeted question-answering.

The systems described in this paper are all available as open-source software. In addition to the contributions of the present work, we hope these components will provide a shared platform for continued research on visual analysis and visualization recommendation tools.

2 Related Work

Voyager draws on and extends prior research on exploratory search interfaces, visualization tools, and automated visualization design.

2.1 Exploratory Search

Voyager is partly inspired by work on *exploratory search* [26, 43], which shares a number of characteristics with *exploratory data analysis* (EDA) [15, 37]. Both involve activities of browsing (gaining an overview and engaging in serendipitous discovery) and searching (finding answers to specific questions). Users must clarify vague information needs, learn from exposure to information, and iteratively investigate solutions. In either exploratory search or EDA, people may be unfamiliar with the resources at hand (*e.g.,* specific datasets), in the midst of forming goals, or unsure about how to best achieve their goals.

Exploratory search is typically supported through browser interfaces. Faceted browsing [47] is a popular approach for exploring collections in which users specify filters using metadata to find subsets of items sharing desired properties. Interactive query refinement — by up-voting or down-voting metadata or items [21, 22] — can further facilitate exploration. In addition, recommender systems (sometimes in the form of collaborative filtering [18]) can be used to populate a browser with ostensibly relevant items, particularly when the number of items renders manual inspection intractable.

Here we seek to adapt these approaches to the domain of exploratory visual analysis. We contribute a browser interface for statistical graphics of a single relational table, and support navigation using facets such as the data schema, applicable data transformations, and valid visual encodings. As the set of possible charts is typically too large to manually inspect, we also contribute a visualization recommender system that attempts to produce relevant and perceptually effective views based on a user's current exploration state.

2.2 Tools for Visualization Construction

Visualization tools offer various levels of expressivity for view construction. Chart typologies, such as the templates provided by spreadsheet programs, are a common form of specification. While easy to use, they typically support a limited range of charts and provide little support for iterative view refinement, a crucial component of EDA.

Visualization toolkits (*e.g.,* [5, 6]) and design tools (*e.g.,* [30, 32]) enable intricate designs but require detailed specification, hindering rapid exploration. Higher-level grammars, such as Wilkinson's *Grammar of Graphics* [44, 45], can generate a wide-range of statistical graphics, but still require textual specification.

On the other hand, Tableau (formerly Polaris) [35] enables similar specification of visualizations using a graphical interface. Users drag-and-drop data variables onto visual encoding "shelves"; the system then translates these actions into a high-level grammar (VizQL), enabling rapid view creation for targeted exploration of multidimensional databases. Voyager adopts a similar grammar-based approach to represent visualizations; however, it automatically generates views and allows users to browse a gallery of recommended views.

2.3 Visualization Recommendation

Much existing research on visualization recommendation focuses on suggesting visual encodings for an ordered set of user-specified data variables. Mackinlay's APT [24] proposes a compositional algebra to enumerate the space of encodings. It then applies a set of *expressiveness* and *effectiveness* criteria based on the work of Bertin [4] and Cleveland [8] to prune and rank the set of visualizations. Sage [31] extends APT with a taxonomy of data properties for recommending visualizations. Tableau's Show Me [25] introduces a set of heuristics to aid in the construction of small multiples and recommend chart types. Voyager draws on this line of work, for example using expressiveness and effectiveness criteria to evaluate visual encoding options. Voyager extends this prior research by contributing methods for also recommending data variables and transformations, and enabling interactive browsing and refinement of multiple recommendations.

After creating valid views, some tools [33, 46] rank views based on statistical properties to recommend interesting relationships between variables in the dataset. Other tools like SemViz [10] and VISO [41] recommend data to visualize using knowledge ontologies from the semantic web. They rely on data having extensive semantic labels, which may not always be available. Other systems [7, 11, 48] recommend visualizations based on analytical tasks and handle a small number of predefined tasks by design. Inferring the user's task or asking the user to select one may preempt the iterative examination process at the heart of EDA. In the absence of perfect knowledge about the user's task, Voyager presents visualizations of appropriate yet diverse view types that cover a variety of data variables for analysts to examine.

Multiple visualizations are often presented in a gallery to facilitate data exploration. The classic Design Galleries work [27] shows alternatives of user-generated views by varying the choice of encoding parameters. Van den Elzen [38] similarly allows users to browse a small number of parameter variants using small multiples of alternative views. Both allow users to explore a small neighborhood of the visualization specification space. In contrast, Voyager presents both data variations and design variations to facilitate broader data exploration.

VizDeck [29] presents a gallery of recommended charts based on statistical properties of interest. The system includes a voting mechanism by which users can adjust the ranking and supports keyword queries to search for charts. Voyager is instead designed to support browsing, which is more suitable for exploratory tasks [43]. Voyager seeks to promote broader coverage of the search space and navigation by including or omitting selected data variables.

3 Usage Scenario

We first motivate the design of Voyager with a usage scenario. We illustrate how an analyst can use the system to examine data about cars [17]. The dataset contains 406 rows (cars) and 9 columns (variables).

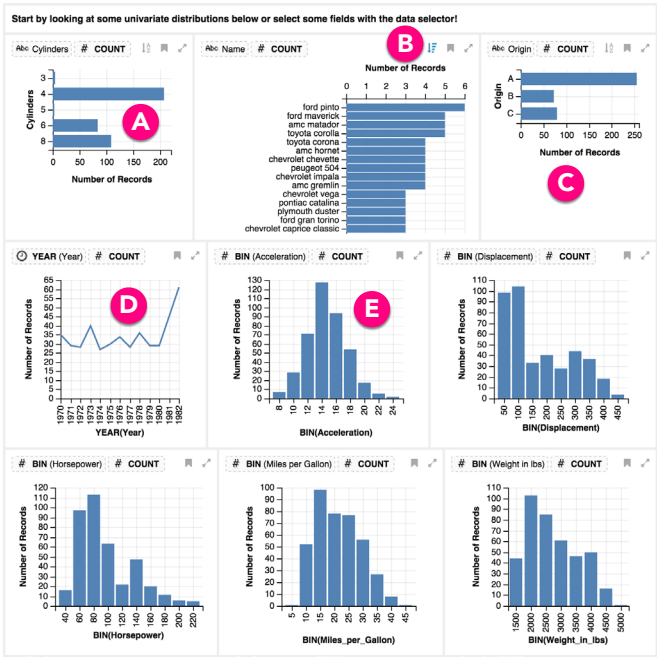

Fig. 2. The main gallery shows univariate summaries upon loading.

Fig. 3. Selecting *horsepower* updates the main gallery. (a) The *exact match* section shows different transformations for *horsepower*. (b) The *suggestion* section shows charts with suggested variables in addition to *horsepower*. (c,d) Each section's header bar describes its member views. (e) Hovering over a point reveals a tooltip with more information.

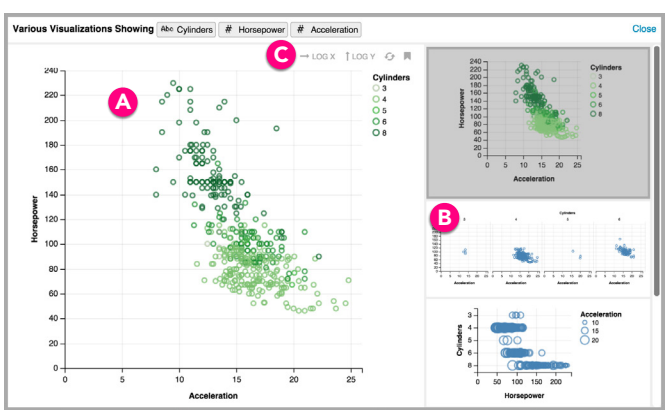

Fig. 4. The expanded gallery for *cylinder*, *horsepower*, and *acceleration*. (a) The main panel presents the selected chart in an enlarged view. (b) The sidebar shows alternative encodings for the expanded data.

Upon loading the data, the analyst examines the list of variables in the schema panel and their univariate summaries in the main gallery (Figure 2). Starting from the top left, she observes that most of the cars have 4, 6, or 8 *cylinders* (Figure 2a). Using the toggle button (⇅) to sort the *name* histogram by *number of records*, she notices that Ford Pinto has the highest frequency, with 6 records (Figure 2b). The majority of the cars are from *origin* A (coded information, Figure 2c) and the *years* 1970-1982 (Figure 2d). Most of the quantitative variables appear to have log-normal distributions except for *acceleration*, which looks normally distributed (Figure 2e).

Intrigued by *horsepower*, the analyst clicks that variable in the schema panel. The system in turn updates the gallery with relevant visualizations (Figure 3). The *exact match* section (Figure 3a) lists charts with varied transformations of *horsepower*. The analyst inspects the dot plot of *horsepower* and hovers over the maxima (Figure 3e) to discover that the car with highest *horsepower* is a Pontiac Grand Prix. She then glances at the *suggestion* section (Figure 3b), which shows charts with additional variables. She notices a correlation between *horsepower* and *cylinder*, and bookmarks the view so she can revisit it for targeted question answering after she completes her initial exploration.

The analyst wonders if other variables might be correlated with both *horsepower* and *cylinder*, so she selects *cylinder* in the schema panel. The display updates as shown in Figure 1. Looking at the first view in the suggestion section (Figure 1, leftmost view in the bottom section), she sees that *acceleration* is correlated with both variables. The analyst would like to see other ways to visualize these three variables, so she clicks the view's expand button (⤢). This action opens the expanded gallery (Figure 4), which shows different encodings of the same data. She selects a small multiple view grouped by *cylinder* (Figure 4b), so she can easily spot outliers in each group (Figure 5).

At this point, the analyst wants to explore other parts of the data. She clicks the reset button to clear the selection and starts selecting new variables of interest to look at relevant visualizations. As her exploration proceeds, she bookmarks interesting views for future investigation in the bookmark gallery (Figure 6).

4 The Design of Voyager

In this section we present our motivating design considerations and describe the design of the Voyager user interface. We defer discussion of technical implementation details to the next section.

4.1 Design Considerations

While creating Voyager we faced many design decisions. The interface should not overwhelm users, yet must enable them to rapidly browse collections of visualizations with minimal cognitive load. To guide our process, we developed a set of considerations to inform visualization recommendation and browsing. These considerations were informed by existing principles for visualization design [24], exploratory search [14, 43], and mixed-initiative systems [19], then refined through our experiences across multiple design iterations.

C1. Show data variation, not design variation. We adapt this well-known maxim from Tufte [36] to the context of visualization galleries. To encourage breadth-oriented exploration [28], Voyager prioritizes showing *data variation* (different variables and transformations) over *design variation* (different encodings of the same data). To discourage premature fixation and avoid the problem of "empty results" [14], Voyager shows univariate summaries of all variables prior to user interaction. Once users make selections, it suggests additional variables beyond those explicitly selected. To help users stay oriented, avoid combinatorial explosion, and reduce the risk of irrelevant displays, Voyager currently "looks ahead" by only one variable at a time.

C2. Allow interactive steering to drive recommendations. Analysts' interests will evolve as they browse their data, and so the gallery must be adaptable to more focused explorations. To steer the recommendation engine, Voyager provides facet controls with which analysts can indicate those variables and transformations they wish to include.

C3. Use expressive and effective visual encodings. Inspired by prior work on automatic visualization design [24, 25], Voyager prevents misleading encodings by using a set of *expressiveness* criteria and ranks encodings based on perceptual *effectiveness* metrics [4, 8].

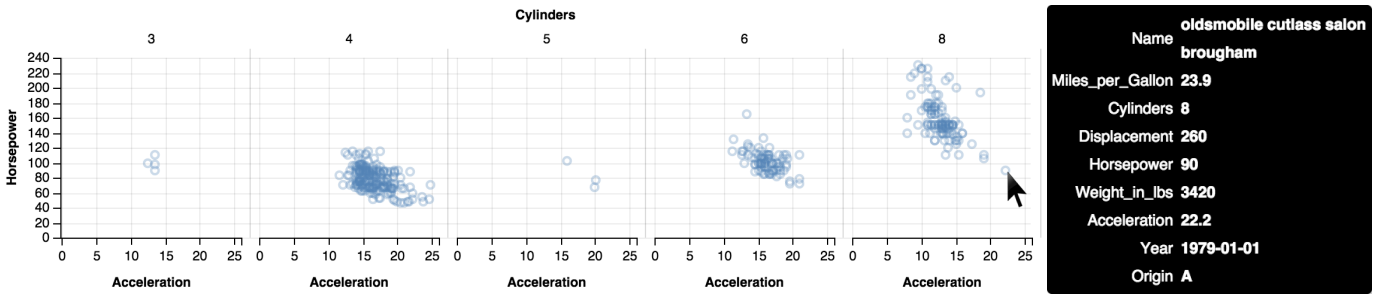

Fig. 5. Scatter plots of *horsepower* vs. *acceleration*, partitioned by *cylinder*. An analyst hovers the mouse over an outlier to view details-on-demand.

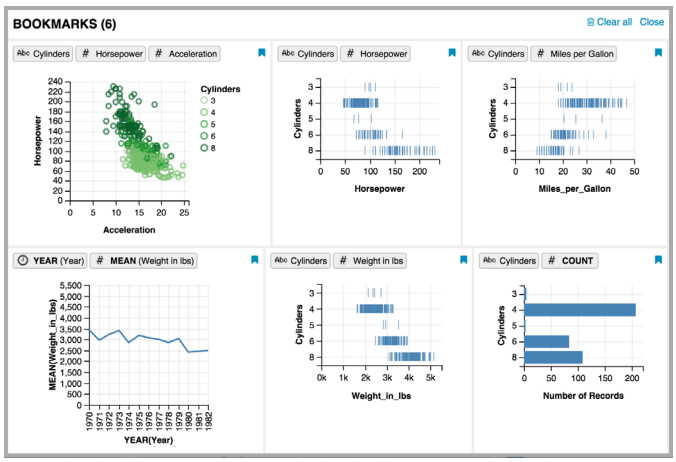

Fig. 6. A bookmark gallery of visualizations saved by an analyst.

C4. Promote reading of multiple charts in context. Browsing multiple visualizations is a complex cognitive process, arguably more so than image or product search. We must consider not only the comprehension of charts in isolation, but also in aggregate. When possible, Voyager consistently orders related charts such that effort spent interpreting one chart can aid interpretation of the next. For example, Voyager aligns axis positions and uses consistent colors for variables (Figure 1). Voyager organizes suggested charts by clustering encoding variations of the same data and showing a single top-ranked exemplar of each cluster. If desired, users can drill-down to browse varied encodings of the data. Voyager also partitions the main gallery into a section that involves only user-selected variables and a section that includes additional (non-selected) variables recommended by the system.

C5. Prefer fine-tuning to exhaustive enumeration. Even a simple chart might have a number of important variations, including the choice of sort order, aspect ratio, or scale transform (*e.g.,* linear vs. log). Rather than using up space in the gallery with highly similar designs, Voyager collapses this space of options to a single chart with default parameters, but supports simple interactions to enable fine-tuning.

C6. Enable revisitation and follow-up analysis. Successful explorations may result in a number of insights worthy of further study. Exploratory tools should assist the transition to other stages of analysis. Voyager provides a bookmarking mechanism to allow analysts to revisit interesting views or to share them with collaborators. By representing all visualizations in a high-level grammar (*Vega-lite*), Voyager can easily export visualizations for publishing or sharing with other tools.

4.2 The Voyager User Interface

Voyager's interface (Figure 1) consists of a schema panel (left) and a visualization gallery (right). Analysts can select variables and desired transformations in the schema panel; these selections become input for the recommendation algorithm. The main gallery presents recommended visualizations. Each chart supports interactive refinement, bookmarks, and expansion to increase the chart size and see related views. Undo buttons are provided in the top panel (Figure 1, top).

4.2.1 The Schema Panel

The schema panel (Figure 1, left) presents a list of all variables in the data table. By default the list is ordered by data type and then alphabetically. For each variable, the schema panel shows the following items from left to right: (1) a checkbox representing inclusion of the variable in the recommendation, (2) a caret button ▾ for showing a popup panel for selecting transformations, (3) a data type icon, (4) variable name and function, (5) and a basic information button ⓘ, which upon hover shows descriptive statistics and samples in a tooltip.

To steer the recommendations (C2), users can click a variable to toggle its inclusion or can select transformation functions in the popup panel revealed by clicking the caret. Selected variables are also highlighted with a surrounding capsule. Similar capsules are used in the gallery to facilitate comparison (C4). Data transformation functions are indicated using bold capitalized text (*e.g.,* **MEAN**).

4.2.2 The Main Gallery: Browsing Recommendations

The main gallery presents views that represent different data subsets relevant to the selected variables. To prioritize *data variation* over *design variation* (C1), each view in the main gallery shows the top-ranked encoding for each unique set of variables and transformations. To help provide meaningful groups (C4), the gallery is divided into two sections: *exact match* and *suggestion*. The top of each section (Figure 3c-d) contains a header bar that provides a description of its member views. The exact match section (Figure 3a) presents views that include only selected variables. In contrast, the suggestion section (Figure 3b) includes suggested variables in addition to selected variables. If the user has not selected any variables (as in Figure 2), only the suggestion section is shown, populated with univariate summaries (C1).

Each entry in the gallery contains an interactive visualization. The top of each view lists its member variables in capsules. The capsules for user-selected variables (solid border, darker background, Figure 7a) are visually differentiated from capsules for suggested variables (dashed border, lighter background, Figure 7b). The top right of each view (Figure 7c) contains *bookmark* and *expand view* buttons. During exploration, analysts can bookmark views they wish to share or revisit (C6); bookmarked visualizations can be viewed in the bookmark gallery (Figure 6). Analysts can also hover over data points to view details-on-demand (Figure 5).

Voyager attempts to parameterize and layout charts such that reading one chart facilitates reading of subsequent related charts (C4). To do so, Voyager places charts with shared axes in close proximity to each other. Moreover, Voyager suggests the same visual encoding (axis position, sorting, spacing, palettes, etc.) for the same variable to aid scanning and reduce visual clutter. For example, it uses a consistent color palette for *cylinders* and aligns y-axes for *cylinders* and *horsepower* in Figure 1). To further aid comparison, all capsules for the same variable are highlighted when the user hovers over a capsule.

Fig. 7. The top of each view shows user-selected variables (a), suggested variables (b), and buttons to bookmark or expand the view (c).

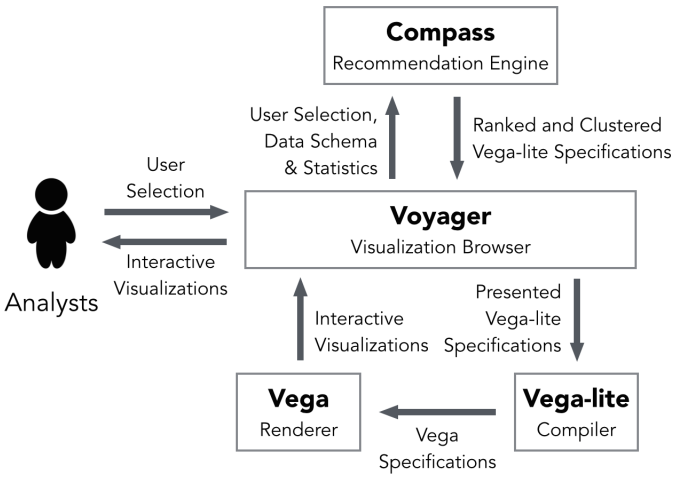

Fig. 8. Voyager's system architecture. Voyager uses Compass to generate clustered and ranked Vega-lite specifications. These specifications are translated to Vega and rendered in the Voyager interface.

4.2.3 The Expanded Gallery: Inspecting Alternative Encodings

An analyst can click a chart's *expand view* button to invoke the expanded gallery (Figure 4). This mode allows analysts to interact with a larger visualization and examine alternative visual encodings of the same data. The top-right corner of the main panel (Figure 4c) includes controls for interactive refinement (C5): transposing axes, sorting nominal or ordinal dimensions, and adjusting scales (*e.g.,* between linear and log). Thumbnails of alternative encodings are presented in a sidebar. Analysts can click a thumbnail to load the chart in the main panel.

5 The Voyager System

We now describe Voyager's system architecture. Figure 8 depicts the relationships between the major system components. Voyager's browser interface displays visualizations and supports user navigation and interaction. Visualizations are specified using *Vega-lite*, a declarative grammar that compiles to detailed *Vega* [40] visualization specifications. The *Compass* recommendation engine takes user selections, the data schema and statistical properties as input, and produces recommendations in the form of Vega-lite specifications. The recommendations are clustered by data and visual similarity, and ranked by perceptual effectiveness heuristics. Each of these components is implemented in JavaScript, and is individually available as an open-source project.

5.1 Vega-lite: A Formal Model For Visualization

We developed the Vega-lite specification language to provide a formal model for representing visualizations in Voyager. Vega-lite is modeled after existing tools and grammars such as Tableau's VizQL [35], ggplot2 [44], and Wilkinson's Grammar of Graphics [45]. Vega-lite specifications consist of a set of mappings between visual encoding channels and (potentially transformed) data variables. Like other high-level grammars, these specifications are incomplete, in the sense that they may omit details ranging from the type of scales used to visual elements such as fonts, line widths and so on. The Vega-lite compiler uses a rule-based system to resolve these ambiguities and translate a Vega-lite specification into a detailed specification in the lower-level *Vega* visualization grammar [40]. Though initially developed for Voyager, Vega-lite can serve as a model for other tools. For example, we built the PoleStar visualization specification tool (§6.1) using Vega-lite.

A Vega-lite specification is a JSON object (see Listing 1) that describes a single data source (`data`), a mark type (`marktype`), key-value visual encodings of data variables (`encoding`), and data transformations including filters (`filter`) and aggregate functions. Vega-lite assumes a tabular data model: each data source is a set of records, where each record has values for the same set of variables.

Vega-lite currently supports Cartesian plots (with mark types `points`, `bars`, `lines` or `areas`), and pivot tables (with mark type `text`). Available encoding channels include position (`x`, `y`), `color`, `shape`, `size`, level of detail (`detail`) for specifying additional group-by values, and facets (`row`, `column`) for creating trellis plots [3, 36]. A specification of each encoding channel (`encoding`) includes the assigned variable's name, data type, scale and axis properties, and transformations. Vega-lite supports nominal, ordinal, quantitative, and temporal data types [34]. Supported transformations include aggregation (`summarize`), binning (`bin`), sorting (`sort`), and unit conversion for temporal variable (`timeUnit`). For example, `year`, `month`, and other time abstraction values can be derived from temporal variables.

```
{
  "data": {"url": "data/cars.json"},
  "marktype": "point",
  "encoding": {
    "x": {
      "name": "Miles_per_Gallon",
      "type": "Q",
      "summarize": "mean"
    },
    "y": {
      "name": "Horsepower",
      "type": "Q",
      "summarize": "mean"
    },
    "row": {
      "name": "Origin",
      "type": "N",
      "sort": [{"name": "Horsepower",
          "summarize": "mean", "reverse": true}]
    },
    "color": {"name": "Cylinders","type": "N"}
  }
}
```

Listing 1. A Vega-lite specification of the visualization shown in Figure 10. The JSON object specifies a trellis of scatter plots for a data about cars. Each plot shows for one origin (`row`) the mean miles per gallon (`x`) and mean horsepower (`y`), broken down by the number of cylinders (`color`). Origin and number of cylinders are nominal while miles per gallon and horsepower are quantitative. The scatter plots are sorted by the mean horsepower per origin.

Vega-lite makes default assignments for parameters such as axis scales, colors, stacking, mark sizes, and bin count. These parameters can be explicitly specified to override default values. Nominal variables are mapped to ordinal scales by alphabetical order unless an explicit order is provided. When assigned to `color`, nominal variables are mapped to hue using Tableau's categorical color palette, while other variables are mapped to luminance. When a `color` channel is used with a `bar` or `area` mark type, Vega-lite creates a stacked chart. The band size of an ordinal scale is automatically adjusted based on the assigned variable's cardinality. Vega-lite determines properties such as bin count based on the encoding channel: the default max bin count is 7 for `color` and 20 for positional encodings.

In the future, we plan to extend Vega-lite with additional features such as cartographic mapping, polar coordinates, and layering multiple variables (including dual axis charts). The goal of this work, however, is to investigate different modes of visual exploration and the current implementation of Vega-lite is sufficiently expressive for this purpose.

5.2 The Compass Recommendation Engine

The goal of the Compass recommendation engine is to support rapid, open-ended exploration in Voyager. Compass generates an expressive set of visualization designs (C3) represented using Vega-lite specifications. Compass also prunes the space of recommendations based on user selection (C2) and clusters results into meaningful groups (C4).

Compass takes the following input: (1) the data schema, which contains a set of variables (D); (2) descriptive statistics for each variable including cardinality, min, max, standard deviation, and skew; (3) the user selection, which consists of a set of selected variables ($U \subset D$), preferred transformations for each variable, and a set of excluded variables.

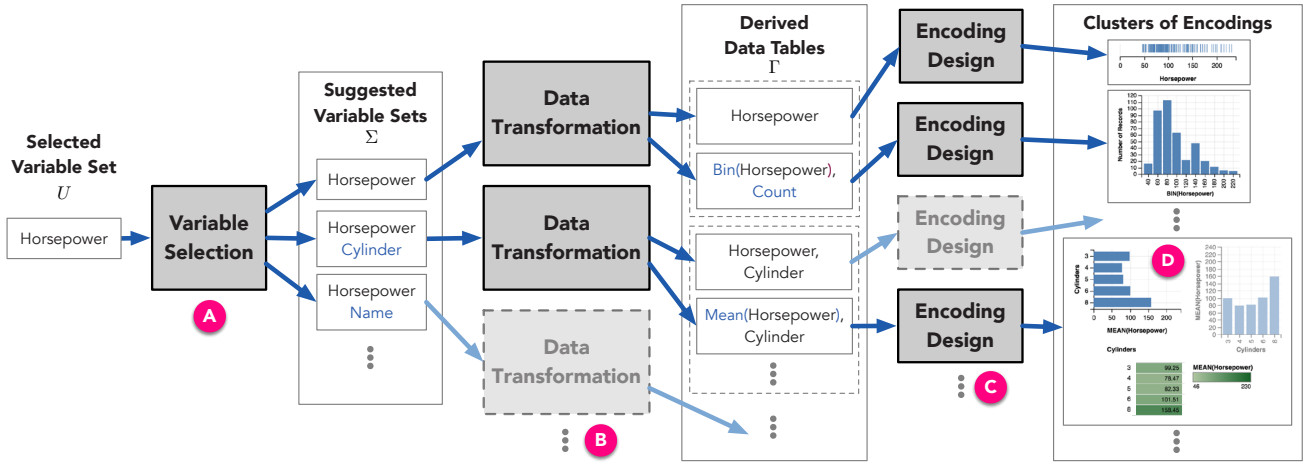

Fig. 9. Compass's 3-phase recommendation engine. (a) *Variable selection* takes user-selected variable sets and suggests additional variables (b) *Data transformation* applies functions including aggregation and binning to produce data tables for each variable set. (c) *Encoding design* generates visual encodings for each data table, ranks results by perceptual effectiveness score, and prunes visually similar results.

Compass enumerates, ranks and prunes recommendations in three phases, taking output from each phase as input to the next phase. The process is depicted in Figure 9. First, Compass *selects variables* by taking user-selected variable sets and suggesting additional variables. It then *applies data transformations*, including aggregation and binning, to produce a set of derived data tables. For each data table, it *designs encodings* based on *expressiveness* and *effectiveness* criteria (C3) and prunes visually similar results to avoid exhaustive enumeration (C5). The multiple phases of pruning and ranking allow Compass to constrain the search space early in the generation process, and to produce clusters of visualizations that are grouped by their underlying data tables.

Our initial Compass design is admittedly modest. Though more advanced recommender systems are possible, the primary goal of this paper is to develop and evaluate an overall approach to breadth-oriented data exploration. In lieu of more sophisticated methods, we intentionally limit ourselves to "single-step" variable additions and interpretable, deterministic heuristics for pruning and ranking. We view the design and evaluation of improved recommenders (likely expressible within the current Compass architecture) as important future research.

5.2.1 Selecting Variables

Compass first suggests variables beyond what the user has explicitly selected, producing new variable sets for encoding. The primary goals of this phase are to recommend additional variables that the analyst might otherwise overlook (C1) and to avoid the "empty results" problem [14].

Prior to user selection, Compass suggests a univariate summary of each variable. When a user selects a variable set U (of size $|U| = k$), Compass returns a sequence of variable sets $\Sigma = [U, V_1, V_2, ..., V_n]$, where U is the original user selection, $n = |D - U|$, and each V_i contains $k+1$ variables: the k user-selected variables U along with exactly one additional (non-selected) variable $v_i \in D - U$, such that $V_i = U \cup \{v_i\}$. For example, if a user selects {`horsepower`}, Compass may return the variable sets U = {`horsepower`}, V_1 = {`horsepower`, `cylinder`}, V_2 = {`horsepower`, `year`}, and so on.

Compass recommends all non-selected variables (all $v_i \in D - U$) by default, but analysts can interactively exclude variables from the suggestions to focus on a particular variable set of interest (C2).

The generated variable sets are returned in a sorted order. The user's selected variable set is always ranked first, as it is the most relevant to the user's specified intention. The remaining variable sets $[V_1, V_2, ..., V_n]$ are ordered by the type and name of the recommended variable v_i, consistent with the display order in Voyager's schema panel. This approach provides a predictable and consistent ordering, which works well for the common case of datasets with a sufficiently bounded number of variables to allow users to scroll through the recommendations. For datasets with a large number of variables, we plan to extend Compass to support multiple relevancy rankings based on statistical measures (e.g., [33]) and allow analysts to select measures that fit their interests.

5.2.2 Applying Data Transformations

For each suggested variable set $V \in \Sigma$ from the first phase, Compass enumerates applicable transformations for each variable $v \in V$ to recommend an ordered set of data tables Γ.

Compass produces both *raw tables* without aggregation to provide details and *aggregate tables* to provide summaries. For raw tables, each variable is untransformed by default, but users can perform binning if desired. For aggregate tables, variables either serve as *measures* (values amenable to aggregation) or *dimensions* (values to group by). By default, each quantitative variable is treated as a measure, while each ordinal, nominal, or temporal variable is treated as a dimension.

Compass averages (`MEAN`) quantitative measures by default; users can choose to apply other aggregation functions such as `SUM`, `MIN`, `MAX`. Averages may be affected by outliers or mix effects [1], but are also more likely to be generally useful. In our experience, defaulting to sums results in plots that are not always meaningful and skewed when the number of records varies across dimensions. Users can also choose to treat quantitative variables as dimensions by either using the untransformed values or binning. Compass determines the largest units of time within the extents of temporal variables. For example, if a variable spans within one year, `MONTH` is applied. Ordinal variables are untransformed by default.

Derived tables are first ordered by the rank of its corresponding variable set. (Tables derived from U come before tables from V_1, which in turn come before tables from V_2, and so on.) For tables from the same variable set, Compass then orders raw tables before aggregate tables to provide a consistent ordering (C4).

5.2.3 Designing Encodings

For each data table $T \in \Gamma$, Compass applies visualization design best practices drawn from by prior research [4, 9, 24, 25, 42] to generate and rank a set of encodings E_T (C3).

Generation. Compass first enumerates candidates for the encoding set E_T by composing permutations of data variables, visual encoding channels, and mark types. It first assigns each variable $v \in T$ to all permitted visual encoding channels (Table 1) to generate a set of mappings M_T. Then it generates each encoding candidate by combining each mapping $m \in M_T$ with each valid mark type.

Compass considers multiple criteria to determine whether a mark type is appropriate for a given mapping m. For a given mark type, some encoding channels are required, while some are disallowed (see Table 2). For example, Compass requires a mapping to have both `x` and `y` encodings if used with a `line` or `area` mark. Such constraints ensure the production of appropriate visualizations (here, a proper line or area chart). After determining the set of supported mark types, Compass assigns the mark type that best respects expressiveness criteria according to the rankings listed in Table 3, indexed by the data types of the `x` and/or `y` encodings.

Data Types	Encoding Channels
quantitative, temporal	x,y > size > color > text
ordinal	x,y > column, row > color > size
nominal	x,y > column, row > color > shape

Table 1. Permitted encoding channels for each data type in Compass, ordered by perceptual effectiveness rankings.

Mark Types	Required Channels	Supported Channels						
		X, Y	Column, Row	Color	Shape	Size	Detail	Text
point	x or y	✓	✓	✓	✓	✓	✓	
tick	x or y	✓	✓	✓				
bar	x or y	✓	✓	✓				
line, area	x and y	✓	✓	✓			✓	
text	text and (row or column)		✓	✓				✓

Table 2. Required and permitted encoding channels by mark type.

Data Types	Mark Types
Q	tick > point > text
(O **or** N) × (O **or** N)	point > text
$Q \times N$	bar > point > text
$Q \times$ (T **or** O)	line > bar > point > text
$Q \times Q$	point > text

Table 3. Permitted mark types based on the data types of the x and y channels. N, O, T, Q denote nominal, ordinal, temporal and quantitative types, respectively.

Positions	x, y
Facets	column, row
Level of detail	color (hue), shape, detail
Retinal measures	color (luminance), size

Table 4. Encoding channel groups used to perform clustering.

In addition to Tables 1-3, Compass considers interactions among visual variables (*e.g.,* the perceptual separability of visual channels [42]) and avoids creating ineffective charts. It produces mappings that encode `color`, `size`, or `shape` only when the mappings also contain both `x` and `y`. In other words, Compass omits dot plots that use these encodings, as they would likely suffer from occlusion and visual clutter.

Ranking. Compass ranks the generated encodings using perceptual effectiveness metrics. Compass applies prior work by Cleveland [8] and Mackinlay [24] to rank the effectiveness of each visual channel based on a variable's data type (Table 1). Compass also considers the cardinality, or number of unique values, of a data variable. Encoding high cardinality variables with `color`, `shape`, `row`, or `column` can lead to poor color or shape discrimination or massive, sparse trellis plots. Moreover, the effectiveness of each visual channel is not measured in isolation. Since over-encoding can impede interpretation [42], Compass penalizes encodings that use multiple retinal encodings (*e.g.,* both `color` and `shape`, or both `color` and `size`).

Compass takes into account that the produced charts will be presented in a gallery, with the goal of promoting charts that are easier to read (C4). Vertical bar charts and histograms are preferred if their dimensions are binned quantitative or temporal variables. Otherwise, horizontal bar charts are preferred as their axis labels are easier to read. Horizontal line and area charts are favored over vertical ones. Compass also privileges encodings that use less screen space, and hence are more easily browsed in the gallery. For example, colored scatter plots (Figure 4a) are ranked higher than small multiple plots (Figure 5).

Compass maps all of the above features to scalar values and calculates a weighted sum to derive the effectiveness score $s(e)$ for each encoding candidate e. We have manually tuned the current weights and scores for each feature through a series of refinements and tests. Automatic determination of these parameters remains as future work.

Clustering. To prevent exhaustive enumeration (C5), Compass groups encoding candidates that map the same variables to similar encoding channels listed in Table 4, and suggests only the most effective view in each group. This includes variants caused by swapping variables in the positional or facet encodings (producing transposed charts as depicted in Figure 9d), or by selecting alternative retinal encodings (*e.g.,* `shape` instead of `color`). All the suggested views are also sorted by the effectiveness score s. Therefore, for each $T \in \Gamma$, Compass produces an ordered set of visually different visualizations E_T, ranked by their perceptual effectiveness.

As a result, Compass recommends clusters of visualizations grouped by their corresponding data tables. To privilege data variation over design variation (C1), the main gallery presents the top ranked visualization for each table $T \in \Gamma$. When the user expands a view that shows data table T, the expanded gallery displays a set of visually different views E_T and provides an interface for refining presented views (Figure 4c).

5.3 Implementation Notes

We implemented Voyager as a web application using the AngularJS framework. When the user selects a dataset, the application asynchronously loads the data, determines the variable types, and calculates descriptive statistics. The type inference procedure determines whether a variable is nominal, ordinal, quantitative or temporal based on primitive types (*e.g.,* integer, string, float), special formats (for dates), and statistics (*e.g.,* low cardinality integers are treated as ordinal).

In the main gallery (§4.2.2), views are laid out using HTML5's flex display. Each view has the same height and has a maximum width. A view that displays a visualization larger than the view size includes a local scroll bar, which is activated by hovering for 500ms in order to disentangle local and global scrolling. By default, Voyager loads up to a fixed number of views to safeguard application performance, but users can load more visualizations as they scroll down the page.

6 Evaluation: Voyager vs. PoleStar

We conducted a user study to contrast recommendation browsing with manual chart construction, focusing on exploratory analysis of previously unseen data. We compared Voyager with PoleStar, our own implementation of a visualization specification interface (Figure 10).

We hypothesized that Voyager would encourage breadth-first consideration of the data, leading to higher coverage of unique variable combinations. Given its direct control over visual encodings, we expected PoleStar to be better for targeted depth-first question answering.

6.1 Study Design

Our study followed a 2 (visualization tool) × 2 (dataset) mixed design. Each participant conducted two exploratory analysis sessions, each with a different visualization tool and dataset. We counterbalanced the presentation order of tools and datasets across subjects.

Visualization Tools. Participants interacted with two visualization tools: Voyager and PoleStar, a manual specification tool. Rather than use an existing tool such as Tableau, we implemented PoleStar (named in honor of Polaris [35]) to serve as a baseline interface, allowing us to control for external factors that might affect the study. Like Voyager, PoleStar models visualizations using Vega-lite. In fact, any visualization suggested by Voyager can also be constructed in PoleStar, ensuring comparable expressivity. PoleStar also features similar UI elements, including field capsules, bookmarks, and an undo mechanism.

Figure 10 illustrates PoleStar's interface. The left-hand panel presents the data schema, listing all variables in the dataset. Next to the data schema are the encoding shelves, which represent each encoding channel supported by Vega-lite. Users can drag and drop a variable onto a shelf to establish a visual encoding. Users can also modify properties of the data (*e.g.,* data types, data transformations) or the visual encoding variable (*e.g.,* color palette or sort order) via popup menus. The mark type can be changed via a drop-down menu. Upon user interaction, PoleStar generates a new Vega-lite specification and immediately updates the display.

Datasets. We provided two datasets for participants to explore. One is a dataset of motion pictures ("movies") comprising title, director, genre, sales figures, and ratings from IMDB and Rotten Tomatoes. The table has 3,201 records and 15 variables (7 nominal, 1 temporal, 8 quantitative). The other dataset is a redacted version of FAA wildlife airplane strike records ("birdstrikes"). The table has 10,000 records and 14 variables (8 nominal, 1 geographic, 1 temporal, 4 quantitative).

Fig. 10. PoleStar, a visualization specification tool inspired by Tableau. Listing 1 shows the generated Vega-lite specification.

We removed some variables from the birdstrikes data to enforce parity among datasets. We chose these datasets because they are of real-world interest, are of similar complexity, and concern phenomena accessible to a general audience.

Participants. We recruited 16 participants (6 female, 10 male), all students (14 graduate, 2 undergraduate) with prior data analysis experience. All subjects had used visualization tools including Tableau, Python/matplotlib, R/ggplot, or Excel.[1] No subject had analyzed the study datasets before, nor had they used Voyager or PoleStar (though many found PoleStar familiar due to its similarity to Tableau). Each study session lasted approximately 2 hours. We compensated participants with a $15 gift certificate.

Study Protocol. Each analysis session began with a 10-minute tutorial, using a dataset distinct from those used for actual analysis. We then briefly introduced subjects to the test dataset. We asked participants to explore the data, and specifically to "get a comprehensive sense of what the dataset contains and use the bookmark features to collect interesting patterns, trends or other insights worth sharing with colleagues." To encourage participants to take the analysis task seriously, we asked them to verbally summarize their findings after each session using the visualizations they bookmarked. During the session, participants verbalized their thought process in a think-aloud protocol. We did not ask them to formulate any questions before the session, as doing so might bias them toward premature fixation on those questions. We gave subjects 30 minutes to explore the dataset. Subjects were allowed to end the session early if they were satisfied with their exploration.

All sessions were held in a lab setting, using Google Chrome on a Macbook Pro with a 15-inch retina display set at 2,880 by 1,980 pixels. After completing two analysis sessions, participants completed an exit questionnaire and short interview in which we reviewed subjects' choice of bookmarks as an elicitation prompt.

Collected Data. An experimenter (either the first or second author) observed each analysis session and took notes. Audio was recorded to capture subjects' verbalizations for later review. Each visualization tool recorded interaction logs, capturing all input device and application events. Finally, we collected data from the exit survey and interview, including Likert scale ratings and participant quotes.

6.2 Analysis & Results

We now present a selected subset of the study results, focusing on data variable coverage, bookmarking activity, user survey responses, and qualitative feedback. To perform hypothesis testing over user performance data, we fit linear mixed-effects models [2]. We include visualization tool and session order as fixed effects, and dataset and participant as random effects. These models allow us to estimate the effect of visualization tool while taking into account variance due to both the choice of dataset and individual performance. We include an intercept term for each random effect (representing per-dataset and per-participant bias), and additionally include a per-participant slope term for visualization tool (representing varying sensitivities to the tool used). Following common practice, we assess significance using likelihood-ratio tests that compare a full model to a reduced model in which the fixed effect in question has been removed.

6.2.1 Voyager Promotes Increased Data Variable Coverage

To assess the degree to which Voyager promotes broader data exploration, we analyze the number of unique variable sets (ignoring data transformations and visual encodings) that users are exposed to. While users may view a large number of visualizations with either tool, these might be minor encoding variations of a data subset. Focusing on unique variable sets provides a measure of overall dataset coverage.

While Voyager automatically displays a number of visualizations, this does not ensure that participants are attending to each of these views. Though we lack eye-tracking data, prior work indicates that the mouse cursor is often a valuable proxy [12, 20]. As a result, we analyze both the number of variable sets shown on the screen and the number of variable sets a user interacts with. We include interactions such as bookmarking, view expansion, and mouse-hover of a half-second or more (the same duration required to activate view scrolling). Analyzing interactions provides a conservative estimate, as viewers may examine views without manipulating them. For PoleStar, in both cases we simply include all visualizations constructed by the user.

We find significant effects of visualization tool in terms of both the number of unique variable sets shown ($\chi^2(1, N = 32) = 38.056, p < 0.001$) and interacted with ($\chi^2(1, N = 32) = 19.968, p < 0.001$). With Voyager, subjects were on average exposed to 69.0 additional variable sets (over a baseline of 30.6) and interacted with 13.4 more variable sets (over a baseline of 27.2). In other words, participants were exposed to over 3 times more variable sets and interacted with 1.5 times more when using Voyager.

In the case of interaction, we also find an effect due to the presentation order of the tools ($\chi^2(1, N = 32) = 5.811, p < 0.05$). Subjects engaged with an average of 6.8 more variable sets (over the 27.2 baseline) in their second session.

6.2.2 Bookmark Rate Unaffected by Visualization Tool

We next analyze the effect of visualization tool on the number of bookmarked views. Here we find no effect due to tool ($\chi^2(1, N = 32) = 0.060, p = 0.807$), suggesting that both tools enable users to uncover interesting views at a similar rate. We do observe a significant effect due to the presentation order of the tools ($\chi^2(1, N = 32) = 9.306, p < 0.01$). On average, participants bookmarked 2.8 additional views (over a baseline of 9.7 per session) during their second session. This suggests that participants learned to perform the task better in the latter session.

6.2.3 Most Bookmarks in Voyager include Added Variables

Of the 179 total visualizations bookmarked in Voyager, 124 (69%) include a data variable automatically added by the recommendation engine. Drilling down, such views constituted the majority of bookmarks for 12/16 (75%) subjects. This result suggests that the recommendation engine played a useful role in surfacing visualizations of interest.

6.2.4 User Tool Preferences Depend on Task

In the exit survey we asked subjects to reflect on their experiences with both tools. When asked to rate their confidence in the comprehensiveness of their analysis on a 7-point scale, subjects responded similarly for both tools (Voyager: $\mu = 4.88, \sigma = 1.36$; PoleStar: $\mu = 4.56, \sigma = 1.63$; $W = 136.5, p = 0.754$). Subjects rated both tools comparably with respect to ease of use (Voyager: $\mu = 5.50, \sigma = 1.41$; PoleStar: $\mu = 5.69, \sigma = 0.95$; $W = 126, p = 0.952$).

[1] All participants had used Excel. Among other tools, 9 had used Tableau, 13 had used Python/matplotlib and 9 had used R/ggplot.

Participants indicated which tool they would prefer for the tasks of exploration vs. targeted analysis. Subjects roundly preferred Voyager for exploration (15/16, 94%) and PoleStar for question answering (15/16, 94%) – a significant difference ($\chi^2(1) = 21.125, p < 0.001$).

Finally, we asked subjects to rate various aspects of Voyager. All but one (15/16, 94%) rated Voyager's recommendations as "Helpful" or "Very Helpful". When asked if Voyager's inclusion of additional (non-selected) variables was helpful, 14/16 (88%) responded "Helpful" or "Very Helpful", with 2 responding "Neutral". We also asked participants to agree or disagree with the statement "The recommendations made by Voyager need improvement." Here, 8/16 subjects (50%) agreed with the statement, 5/16 (31%) were neutral and 3/16 (19%) disagreed. This last result surprised us, as we expected all subjects would request refined relevance rankings. In aggregate, these results suggest that though there remains room for improvement, the current Voyager system already provides a valuable adjunct to exploratory analysis.

6.2.5 Participant Feedback: Balancing Breadth & Depth

Participants' comments reinforce the quantitative results. Subjects appreciated Voyager's support for broad-based exploration. One said that *"Voyager gave me a lot of options I wouldn't have thought about on my own, it encouraged me to look more deeply at data, even data I didn't know a lot about"*. Another *"found Voyager substantially more helpful in helping me learn and understand the dataset,"* while a third felt Voyager *"prompted me to explore new questions in a way that didn't derail me from answering follow-up questions."*

Subjects also reflected on the complementary nature of Voyager and PoleStar for the tasks of breadth- vs. depth-oriented exploration. One user noted that *"with Voyager, I felt like I was scanning the generated visualizations for trends, while with PoleStar, I had to think first about what questions I wanted to answer, then make the visualizations for them."* Another wrote that Voyager *"is really good for exploration but cumbersome for specific tasks."* All but one subject wished to use a hybrid of both tools in the future. For example, one participant said that *"if I have to just get an overview of the data I would use Voyager to generate visualizations, and then dive in deep using PoleStar,"* while another envisioned that *"I would start with Voyager but want to go and switch to PoleStar to dive into my question. Once that question was answered, I would like to switch back to Voyager."*

7 Discussion and Future Work

We presented *Voyager*, a mixed-initiative system to facilitate breadth-oriented data exploration in the early stages of data analysis. Voyager contributes a visualization recommender system (*Compass*) to power a novel browsing interface that exchanges manual chart specification for interactive browsing of suggested views. In a user study comparing Voyager with a visualization tool modeled after Tableau (*PoleStar*), we find that Voyager encourages broader exploration, leading to significantly greater coverage of unique variable combinations. The vast majority of participants (15/16) expressed a preference for using Voyager in future exploration tasks. This result is encouraging and indicates the value of improved support for early-stage exploration: PoleStar is based on a popular interaction model backed by over a decade of research and industrial use, whereas Voyager is relatively new and untested.

That said, we view Voyager as a first step towards improved systems that balance automation and manual specification. First, multiple avenues for future work lie in perfecting the Voyager interface. Further iterations could support more navigational facets (e.g., based on chart types, statistical features or detected anomalies [23]) as well as additional visualizations (e.g., dots plots annotated with summary statistics, violin plots, cartographic maps) and data types (e.g., networks). While we have taken initial steps to support reading of multiple charts in context (C4), more work is needed to formalize and evaluate this goal.

Additional interaction techniques might further aid analysis. An obvious candidate is to support brushing & linking within Voyager's visualization gallery. Other possibilities may require additional research. For example, trellis plots are a valuable method for multidimensional visualization, but often require substantial screen real estate at odds with a rapidly scannable gallery. An open question (c.f., [13]) is how to best fit such large plots in small spaces, perhaps via paging and "scrubbing" interactions common to videos or image collections.

An important avenue for continued research is the design and evaluation of more sophisticated (and more scalable [39]) visualization recommenders. Our current approach is intentionally conservative, seeking to provide useful recommendations while helping users stay oriented. The current system is deterministic, based on best practices formalized as heuristics in a rule-based system. In future work, we hope to explore probabilistic recommendation models that can learn improved ranking functions over time (e.g., by updating parameters in response to user input, including bookmarks and explicit ratings). Such work also opens up possibilities for studying personalization or domain adaptation. For example, might different data domains benefit from differing recommendation strategies?

A clear next step is to better integrate breadth-first and depth-first visual analysis tools. How might Voyager and PoleStar be most fruitfully combined? One straightforward idea is that Voyager users could further drill-down into plots to enable PoleStar-like refinement. However, it is not immediately clear if refinements should backpropagate into broad exploration. How might view refinements inform subsequent recommendations in a consistent, understandable fashion?

Finally, further empirical work is needed to better understand the analysis process, including breadth- and depth-oriented strategies. Our user study, for instance, resulted in rich event logs and think-aloud transcripts that are ripe for further analysis beyond the scope of this paper. By recording and modeling analysis activities, we might better characterize analysts' strategies and inform tool development.

To support these and other future research questions, the system components described in this paper (Voyager, Compass, PoleStar, and Vega-lite) are all freely available as open-source software at *http://vega.github.io*. We hope these systems will provide valuable building blocks and shared reference points for visualization research and development.

Acknowledgments

We thank the anonymous reviewers, Magda Balazinska, Daniel Halperin, Hanchuan Li, Matthew Kay, and members of the Interactive Data Lab and Tableau Research for their comments in improving this paper. This work was supported in part by the Intel Big Data ISTC, DARPA XDATA, the Gordon & Betty Moore Foundation, and the University of Washington eScience Institute. Part of this work was developed during the first author's internship at Tableau Research in 2013. We also thank the Noun Project and Dmitry Baranovskiy for the "person" icon used in Figure 8.

References

[1] Z. Armstrong and M. Wattenberg. Visualizing statistical mix effects and simpson's paradox. *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis)*, 20(12):2132–2141, 2014.

[2] D. J. Barr, R. Levy, C. Scheepers, and H. J. Tily. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of memory and language*, 68(3):255–278, 2013.

[3] R. A. Becker, W. S. Cleveland, and M.-J. Shyu. The visual design and control of trellis display. *Journal of computational and Graphical Statistics*, 5(2):123–155, 1996.

[4] J. Bertin. *Semiology of graphics: diagrams, networks, maps*. University of Wisconsin press, 1983.

[5] M. Bostock and J. Heer. Protovis: A graphical toolkit for bisualization. *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis)*, 15(6):1121–1128, 2009.

[6] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis)*, 17(12):2301–2309, 2011.

[7] S. M. Casner. Task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics (TOG)*, 10(2):111–151, 1991.

[8] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.

[9] S. Few. *Now you see it: simple visualization techniques for quantitative analysis*. Analytics Press, 2009.
[10] O. Gilson, N. Silva, P. W. Grant, and M. Chen. From web data to visualization via ontology mapping. *Computer Graphics Forum*, 27(3):959–966, 2008.
[11] D. Gotz and Z. Wen. Behavior-driven visualization recommendation. In *Proceedings of the 14th international conference on Intelligent user interfaces*, pages 315–324, 2009.
[12] S. Green, J. Heer, and C. D. Manning. The efficacy of human post-editing for language translation. In *Proc. ACM Human Factors in Computing Systems (CHI)*, 2013.
[13] R. Hafen, L. Gosink, J. McDermott, K. Rodland, K.-V. Dam, and W. Cleveland. Trelliscope: A system for detailed visualization in the deep analysis of large complex data. In *Proc. IEEE Large-Scale Data Analysis and Visualization (LDAV)*, pages 105–112, Oct 2013.
[14] M. Hearst. *Search user interfaces*. Cambridge University Press, 2009.
[15] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *Commun. ACM*, 55(4):45–54, Apr. 2012.
[16] J. Heer, F. Van Ham, S. Carpendale, C. Weaver, and P. Isenberg. Creation and collaboration: Engaging new audiences for information visualization. In *Information Visualization*, pages 92–133. Springer, 2008.
[17] H. V. Henderson and P. F. Velleman. Building multiple regression models interactively. *Biometrics*, pages 391–411, 1981.
[18] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.
[19] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proc. ACM Human Factors in Computing Systems (CHI)*, pages 159–166, 1999.
[20] J. Huang, R. White, and G. Buscher. User see, user point: gaze and cursor alignment in web search. In *Proc. ACM Human Factors in Computing Systems (CHI)*, 2012.
[21] S. Kairam, N. H. Riche, S. Drucker, R. Fernandez, and J. Heer. Refinery: Visual exploration of large, heterogeneous networks through associative browsing. *Computer Graphics Forum (Proc. EuroVis)*, 34(3), 2015.
[22] Y. Kammerer, R. Nairn, P. Pirolli, and E. H. Chi. Signpost from the Masses: Learning Effects in an Exploratory Social Tag Search Browser. In *Proc. ACM Human Factors in Computing Systems (CHI)*, pages 625–634, 2009.
[23] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proc. Advanced Visual Interfaces (AVI)*, pages 547–554. ACM, 2012.
[24] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.
[25] J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis)*, 13(6):1137–1144, 2007.
[26] G. Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4):41–46, 2006.
[27] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 389–400. ACM Press/Addison-Wesley Publishing Co., 1997.
[28] D. S. Moore and G. P. McCabe. *Introduction to the Practice of Statistics*. WH Freeman/Times Books/Henry Holt & Co, 1989.
[29] Perry, Daniel B and Howe, Bill and Key, Alicia MF and Aragon, Cecilia. VizDeck: Streamlining exploratory visual analytics of scientific data. In *Proc. iSchool Conference*, 2013.
[30] D. Ren, T. Hollerer, and X. Yuan. ivisdesigner: Expressive interactive design of information visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2092–2101, 2014.
[31] S. F. Roth, J. Kolojejchick, J. Mattis, and J. Goldstein. Interactive graphic design using automatic presentation knowledge. In *Proc. ACM Human Factors in Computing Systems (CHI)*, pages 112–117. ACM, 1994.
[32] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. In *Computer Graphics Forum*, volume 33, pages 351–360. Wiley Online Library, 2014.
[33] J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. *Information Visualization*, 4(2):96–113, 2005.
[34] S. S. Stevens. On the theory of scales of measurement, 1946.
[35] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.
[36] E. R. Tufte. *The visual display of quantitative information*, volume 2. Graphics press Cheshire, CT, 1983.
[37] J. W. Tukey. Exploratory data analysis. *Reading, Ma*, 231:32, 1977.
[38] S. van den Elzen and J. J. van Wijk. Small multiples, large singles: A new approach for visual data exploration. *Computer Graphics Forum*, 32(3pt2):191–200, 2013.
[39] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. SeeDB: Automatically generating query visualizations. *Proceedings of the VLDB Endowment*, 7(13):1581–1584, 2014.
[40] Vega: A visualization grammar. `https://github.com/trifacta/vega`.
[41] M. Voigt, S. Pietschmann, L. Grammel, and K. Meissner. Context-aware recommendation of visualization components. In *Proceedings of the 4th International Conference on Information, Process, and Knowledge Management*, pages 101–109, 2012.
[42] C. Ware. *Information visualization: perception for design*. Elsevier, 2012.
[43] R. W. White and R. A. Roth. Exploratory search: Beyond the query-response paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–98, 2009.
[44] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2009.
[45] L. Wilkinson. *The Grammar of Graphics*. Springer, 2005.
[46] G. Wills and L. Wilkinson. Autovis: automatic visualization. *Information Visualization*, 9(1):47–69, 2010.
[47] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proc. ACM Human Factors in Computing Systems (CHI)*, pages 401–408, 2003.
[48] M. X. Zhou and M. Chen. Automated generation of graphic sketches by example. In *IJCAI*, volume 3, pages 65–71, 2003.