Information Visualization
XX(X):1–14
©The Author(s) 2021
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Contextual In-Situ Help For Visual Data Interfaces

Pramod Chundury[1] and M. Adil Yalçin[2] and Jonathan Crabtree[3] and Anup Mahurkar[3] and Lisa M. Shulman[3] and Niklas Elmqvist[1]

Abstract
As the complexity of data analysis increases, even well-designed data interfaces must guide experts in transforming their theoretical knowledge into actual features supported by the tool. This challenge is even greater for casual users who are increasingly turning to data analysis to solve everyday problems. To address this challenge, we propose data-driven, contextual, in-situ help features that can be implemented in visual data interfaces. We introduce five modes of help-seeking: (1) contextual help on selected interface elements, (2) topic listing, (3) overview, (4) guided tour, and (5) notifications. The difference between our work and general user interface help systems is that data visualization provide a unique environment for embedding context-dependent data inside on-screen messaging. We demonstrate the usefulness of such contextual help through two case studies of two visual data interfaces: Keshif and POD-Vis. We implemented and evaluated the help modes with two sets of participants, and found that directly selecting user interface elements was the most useful.

Keywords
Contextual help, help systems, visualization, guidance, guided tour.

Introduction

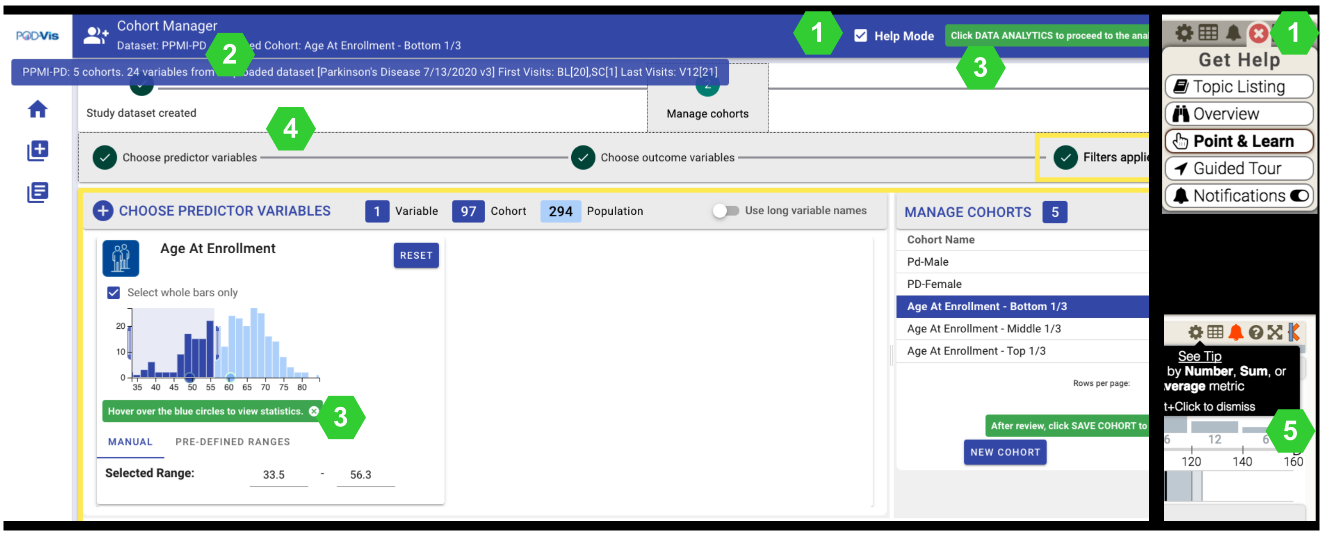
Using computer applications effectively can be demanding for both first-time and experienced users. While general user interface improvements, better interaction models, and increased familiarity have made applications easier to use, using new interfaces and learning new concepts always pose challenges.[1] Unfortunately, we may have become victims of our own success; users today expect to use new applications—even complex ones—immediately with no or minimal training, and to learn and troubleshoot as they go. In particular, designing self-instructional interfaces for data science tools faces many challenges because of the complexity of sensemaking. As a consequence, visualization tools such as Tableau, Spotfire[2], or Keshif[3] must guide experts in translating their analytical knowledge into actual tool features. This step is even more challenging for casual users or novices, who may have limited data analysis vocabularies yet are increasingly searching for data-driven answers in their everyday lives. However, traditional help materials based on static datasets and fixed application settings are a poor fit for current data analysis environments by requiring the user to translate abstract information into their task at hand. Contextual and integrated help systems have the potential to provide this crucial help and training guidance. Specifically, data interfaces constitute an unprecedented opportunity for *data-driven contextualization* where the features of the underlying dataset—such as variable types or distributions—and analysis settings—such as chart types and data selections—can be used to guide the user to learn the tool and perform data analysis.

We present a contextual, data-driven, and in-situ help framework for visualization. We implemented this framework in two visual data interfaces: (1) *HelpIn* for the multidimensional data browser Keshif,[4] and (2) Help Mode for *POD-Vis*, a visual analysis system to explore longitudinal medical outcomes. With contextual integration of help instructions using visual callouts, superimposed labels, and dynamic annotation into a live visual data interface (Figure 1), our framework responds to active data and application context to reduce the physical distance of help material to the interface, thus minimizing the need for the user to split their attention between tool window and help system.[5] The features of data, visualizations, queries, and application and task history help users to quickly find help material of interest by contextual filtering and ranking, and to comprehend dynamic narrative answers. We introduce five modes of help-seeking across the pull/push model (help initiated by the user vs. system)[6]: *Point&Learn, Topic Listing, Overview, Guided Tour, and Notifications.* While updating interface design can quickly make screenshots or videos outdated, our framework allows help material to be adjusted incrementally during development, enabling iterative maintenance.

We evaluated our contextual help framework as follows. For Keshif, we compared HelpIn to its stripped-down version with non-contextual topic index and non-integrated answers

[1]University of Maryland, College Park, MD, USA
[2]Microsoft Corporation, Redmond, WA, USA
[3]University of Maryland Medical School, Baltimore, MD, USA

Corresponding author:
Niklas Elmqvist, University of Maryland, College Park, College of Information Studies, 4130 Campus Drive, MD 20742, College Park, USA.
Email: elm@umd.edu


Click DATA ANALYTICS to proceed to the ana

PPMI-PD: 5 cohorts. 24 variables from loaded dataset [Parkinson's Disease 7/13/2020 v3] First Visits: BL[20],SC[1] Last Visits: V12[21]

Hover over the blue circles to view statistics.

After review, click SAVE COHORT to

Figure 1. Overview of contextual help in POD-Vis (left) and Keshif (right). 1 Users can toggle the help options in POD-Vis—"Help Mode", and Keshif—"HelpIn". 2 Hovering over interface elements triggers the Point&Learn help feature. 3 While Help Mode is enabled, in-context messages are displayed to help learn interface and data actions. 4 POD-Vis includes a step-by-step Guided Tour of the data analysis process. 5 Tips are sent as Notifications to Keshif users, and accessed when needed.

using shared instructional material. Similarly, for POD-Vis, we evaluated its Help Mode as part of our iterative design process. While our participants showed similar task progress across the help system conditions, the *Point&Learn* mode was found the most useful in their feedback, and objectively lead to higher task performance while also increasing time spent on help. Given high-quality help instructions, the preference across static vs. integrated topic answers were split. We also present help-seeking behaviors for visual analytics.

Background

Using new or rich interfaces can be a demanding task for users with a variety of backgrounds. Therefore, the design of effective help systems and documentation is an integral part of human-computer interaction research. Here we summarize the motivating related work, existing approaches, and the differences of our contributions.

Basic Help Techniques

As one of the most common help methods, *help topic indices* are commonly used to offer alphabetical, hierarchical, and search-based access to help. However, empirical studies suggest that users often avoid using both paper and online help manuals, and are frustrated by navigation, terms of indexing, and level of explanations.[7] In other words, users are increasingly eschewing this kind of traditional help mechanisms, favoring instead integrated help.

As a common UI pattern, *tooltips* (callouts) are simple text labels that offer brief information next to a UI component on demand (such as on mouse hover). However, they generally present static (non-contextual) descriptive information, and they are not indexed for navigation.

Guided tours use a sequence of tooltips as a fixed, step-by-step introduction to various interface components and tasks. Some guided tours are coupled with *overlays* consisting of multiple tooltips that can describe multiple components at once (for example Keshif[3]). However, both of these approaches are intrinsically static and precanned, and thus cannot provide help on-demand and on targeted questions.

Automated wizards aim to complete specific tasks on behalf of the user with minimal interruption. However, as argued above, wizards are static and tend to not provide data-specific context. This contrasts with how people are taught to carry out data analysis under different datasets and a rich range of configurations; describing general strategies while contextualizing them in concrete data.

A more dynamic approach to presenting help and guidance is to use a *multi-layered approach*[8] that structures help material from simple (on first-use) to complex (on continued use). In a similar fashion, the *training wheels* strategy[9] blocks complex actions and error states on introductory use. We borrow inspiration from both of these ideas in our work in this paper.

Video-based Training

Videos can introduce multiple interface features in a recorded sequence, often using verbal explanations. The research on video-based training commonly aims to allow navigation by video-content. To provide a content-annotated timeline, ToolScape[10] uses crowdsourcing to extract annotations, and Waken[11] identifies events and interface components by image processing. Nguyen and Lie[12] propose controlling the video playback by making the videos partially interactive within the captured video frame, while Pongnumkull et al.[13] propose synchronizing a tutorial video to a live interface when the user aims to achieve the same task on video.

However, videos fundamentally present a fixed linear flow using static material that cannot adjust to an active application. Users can disengage from video training for reasons including long segments, abstract conceptual information, inconsistencies within and compared to other documentation, or extensive zooming[14]. Changes in interface design can easily outdate existing videos. Therefore, producing and maintaining high-quality videos remains demanding; with videos offering limited integrated and contextual help support.

Context-aware Help Systems

AmbientHelp[15] uses a secondary monitor to continuously and ambiently present help material (videos and manuals) outside of the primary work monitor, with relevance detected using most recent user actions. Targeting web-search applications, Ekstrand et al.[16] propose context-profiles including recently used tools, actions, and open interface components. Our help modes, on the other hand, provide descriptions of data elements with an interpretation of actual live data.

Myers et al.[17] focus on answering why and why-not questions in user interfaces. Their query model can extract topics from pointed elements or recent actions, and present answers with textual description and relevant interface components highlighted. Yeh et al.[18] use screenshots to overlay the help on the interface directly. However, image targeting rules can result in false positives/negatives, and is not robust to changes in interface design. Also, this system cannot be aware of the full application state or underlying data, or control the application. A key distinguishing element of our help modes is that it provides descriptions of live data with explanations of how to interpret and act on that data in context of the data interface itself.

Help and Training for Visual Data Interfaces

Existing studies on visualization help commonly focus on providing training for a single visualization design or technique. Recently, Kwon and Lee[19] studied the effectiveness of different learning approaches for scatterplots (static, video, and interactive). Other recent approaches include converting visualizations to natural language descriptions of data features and potential insights, such as the recent Wordsmith[20] and Narratives[21] tools developed for dashboards created with Tableau software. While our framework also features customized narrations, these come in response to help seeking rather than detecting and presenting potential insights. Our method also enables finding relevant help topics rather than insights.

To our knowledge, there exists no comprehensive, integrated, and responsive help system developed for rich visual data interfaces similar to our work. Closely tied to help and training, literacy and knowledge have received attention in visual data analytics community. For assessing visualization literacy, Boy et al.[22] propose a principled approach based on Item Response Theory. In the Cognitive Exploration Framework[23], knowledge is modeled to influence cognitive activities in visual data exploration as a dynamic construct that can be extended with new knowledge of data and the application over use. These discussions on visual literacy and sensemaking further motivate our work towards improving help for data interfaces.

Design Rationale

The principles of minimalist documentation[24] motivates the design and contributions of our work: (i) getting started fast, (ii) training on real tasks (and real data), (iii) reading in any order, (iv) coordinating system and training, (v) using the situation (context), (vi) leveraging prior knowledge, and (vii) supporting reasoning, and improvisation. While our implementation also aims to (viii) support error recognition and recovery, as well as (ix) develop optimal training designs, we do not claim contributions on these principles. Based on these principles, we derive the following design characteristics (DCs) and motivate their use:

DC1 **Bridge to the familiar.** New help systems should build on existing help system conventions. Thus, our design and contributions reflect Caldwell and White's help-system design goals[25] of navigability, consistency, relevance, coherence, conciseness, reuse, and fidelity.

DC2 **Minimize separation.** Earlier studies have empirically shown that physical and temporal separation of information sources undermine learning, i.e. the split-attention effect.[5]

DC3 **Enable rapid switching.** Our design enables rapid switching between consulting help and using the interface (analyzing data).[26]

DC4 **Coexistence.** Help systems should avoid interference with the main interface and should remain unobtrusive while the user focuses on the original task.[27]

DC5 **Demonstrate first.** We guide the user through complex operations by demonstration in the context of the interface[28]—showing rather than telling.[29]

Design Space: Contextual Help in Visual Data Interfaces

A *help system* for a software application is a form of documentation integrated into the software that is designed to aid the user in correctly operating the software.[7] For visualization software, a help system instructs and guides the user in utilizing the visualizations, interactions, and data transformations to the best effect.[8] *Contextual help*, accordingly, uses the current software application state to provide customized and targeted help to facilitate comprehension.[16] Such contextual features can be used to filter and rank help material by relevance, and also to present dynamic and integrated answers.

Here we discuss how to design a contextual help system for data-visualization that draws on the design constraints DC1–DC5 above. In doing so, we identify two orthogonal axes that we can use when reasoning about such systems:

- **Initiating help:** Is the help explicitly invoked by the user (*pull*), or is it initiated by the computer system upon detecting specific context (*push*)?
- **Type of context:** The help context ranges from (i) *data-driven*, (ii) *application-driven*, and (iii) *history-driven* contextual features.

Below, we discuss these factors at length. In each category, we exemplify finding and integrating relevant help content. We will be using these three categories—DDC, ADC, and HDC—when discussing concrete help mechanisms later on in the paper.

Initiating Help: Push vs. Pull Models

Help systems can be organized into traditional *pull* mechanisms, that are initiated by the user, and less common *push* mechanisms, that are initiated by the computer system. These terms are drawn from *mixed-initiative interaction*,[6] where user-initiated action is blended with computer-initiated action. In other words, this classification deals primarily with how help is initiated. Context is used to determine how to actually deliver the help.

A pull-based help system requires the user to request the help, whereas a push-based one will detect situations where the user may be in need of help and show it automatically. The pull model is unobtrusive and lets the user focus on their primary task. The push model, on the other hand, may enable the user to carry out future tasks more efficiently through opportunistic learning.[30] Both mechanisms have their use; a sophisticated user application likely requires both.

Data-Driven Context (DDC)

This context category describes features of the underlying data, such as data types, distributions, and relations, and the states of data visualization and queries.

Help Seeking. Relevance of help content can be defined by existing data types, features and query states. For example, help content concerning computing temporal characteristics, such as extracting month, would be relevant only when data has a temporal component. If the data is not filtered, content on clearing filters would not be relevant. Help content can reflect the existing data visualization types as well. For example, selecting data by geographical regions would be relevant only when a map is visible. Ranking help content on relevance can reflect frequency of data types as well. For example, if numeric data type is common, tasks on numeric data can have higher priority (see Topic Relevance and Ranking above). To further support data-driven help seeking, data glyphs, or visualization components can be directly selected to retrieve their contextual information. Topic names can also reflect visualization states. For example, if application allows two modes for visual scale, the topic name can reflect the alternative (target) setting.

Help Comprehension. To try to aid comprehension, help descriptions can highlight appropriate data types, features, or distributions. A heuristic approach may select a data aggregate that includes about half of the data. For example, in our POD-Vis system, we suggest aggregating data into halves, tertiles, and quartiles. In addition, the descriptions of help can include information about the data distribution and features. For example, description of a record can include its sorted rank, or multiple encodings in a visual glyph can be clarified with legends and exact values. The answer can also respond to the visualization state and visual encodings. For example, in a scatterplot, requesting help information on a filtered-out dot (record) can describe why it is filtered out (i.e. which query it fails). If points are color coded by category, the description can describe the color mapping and the category of the point.

Application-Driven Context (ADC)

This context category describes the application state, as well as UI components (such as widgets, buttons, menus, etc.) that are visible or are reachable through interaction.

Help Seeking. The help material relevance can reflect active application settings. For example, if none of the panels in the interface is collapsed, the "uncollapse panel" topic would not be relevant. Relevant help material can be requested by interface components, either by direct interaction (pointing), or using a textual list of components. For example, pointing to a sorting icon can suggest "Change sorting criteria" and "Sort in reverse" topics. Considering help-system as application status, presenting related help topics to a selected help material can expand users' repertoire and provide supporting information. In addition, the position of help panels (and tooltips) can be adjusted to avoid, or minimize, overlap with highlighted components. Location-aware presentations has been shown to increase training performance[31].

Help Comprehension. Help can be presented by highlighting relevant interface components, such as where to click to change a setting, to minimize the distance between answer and action. The help descriptions can be responsive, describing the current state, and the role and use of alternative states.

History-Driven Context (HDC)

This category is based on the actions performed by the user.

Help Seeking. The help topics can be ranked by recency or frequency of usage, emphasizing either more/less or most/least frequently used features. The action history information can be used to refresh the user's memory or clarify most recent interactions, or to enable discovery of new (or unused) features. User actions can also be used to infer high-level behavior. For example, if the user scrolls frequently in a visualization panel, the help system may suggest to maximize it. Or, when user frequently highlights two categories, the system may suggest comparing the two by a locked selection. Modeling user actions through methods such as clickstream analysis[32] can lead to powerful help personalization.

Help Comprehension. The answer may exemplify the most recently used components if there are alternatives to achieve the task.

Implementing Contextual In-Situ Help

Our contextual help system is based on five modes for help-seeking based on the contexts defined earlier: (i) Topic Listing, (ii) Overview, (iii) Point&Learn, (iv) Guided Tour, and (v) Notifications. The framework is designed as an overlay on top of a visual data interface. It blends a semitransparent help overlay with the underlying interface in the background, enabling the user to stay oriented. To demonstrate our ideas, we integrate our contextual help systems in two separate web-based data exploration tools, Keshif[4] and POD-Vis.

For Keshif, the help features were implemented as a plugin called *HelpIn*. All five help modes were implemented as part of the plugin. Oftentimes, visual data interfaces are developed using programming frameworks—as defined by the project or team needs. The POD-Vis tool, on the other hand, currently includes the Point&Learn and Guided Tour. For brevity we explain the help modes using HelpIn, but include images from POD-Vis when applicable.

Both HelpIn (Keshif), and Help Mode (POD-Vis) feature a stencil approach[31] to highlight interface components that are selected by the user (Figure 1), or to present part of a help topic answer (Figure 3). We propose the following components (with design constraints addressed for each):

- **Overview** presents a short narrative summary of active data analysis state (DC1, DC2, DC5);
- **Topic Listing** reflects an explicit pull action with the user controlling the topic search by keywords (DC1);
- **Point & Learn** makes it easier to pull help based on the pointed interface area (DC2, DC3, DC4);
- **Guided Tour** is initiated (pulled) by the user, yet the sequence of material is pushed by the help system (DC1, DC2, DC4, DC5); and
- **Notifications** reflect the explicit push mode by monitoring application use, and suggesting specific help directly (DC4, DC5).

We emphasize minimalism and simplicity in the help language, to reduce verbosity, and to maintain consistency across all components.[24] Our material reflects the design language of the underlying application, such as using the same icons and color design, thus reducing the extraneous cognitive load.[33]

Contextual Help Modes

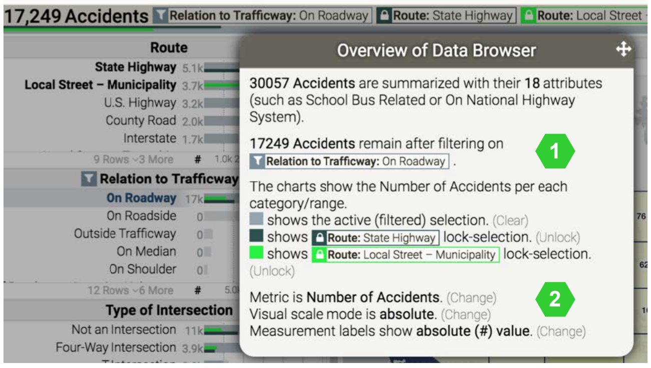
In this section, we describe the design of five help-seeking modes and the topic answer, which provides instructions.

Overview (pull)

The Overview mode (Figure 2) shows a narrative high-level summary of the active data analysis and interface state. This approach minimizes separation (DC2), bridges to the familiar (DC1), and demonstrates first (DC5). It orients the user in data analysis and exploration by describing multiple relevant features that affect the active view (such as active selections, and visualization modes). It also allows the user to see how these modes can be changed by linking to individual help topics. The Overview mode uses both data-driven (DDC) context—by integrating specific data from the current dataset selection—and application-driven (ADC) context—by highlighting relevant interface components.

Topic Listing (pull)

The Topic Listing mode (Figure 3) lists all help topics, ranked (and filtered) by relevance given the current context.



17,249 Accidents | Relation to Trafficway: On Roadway | Route: State Highway | Route: Local Street

Overview of Data Browser

30057 Accidents are summarized with their 18 attributes (such as School Bus Related or On National Highway System).

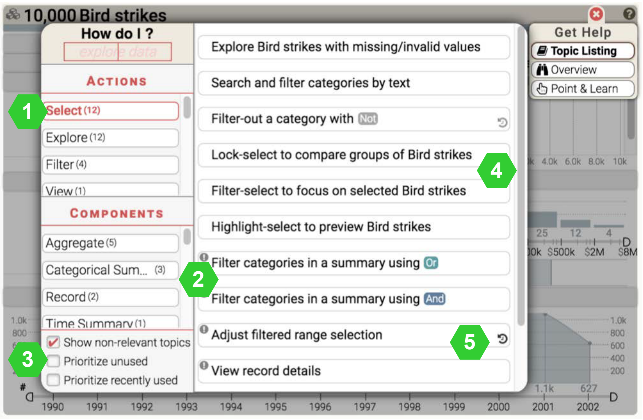
17249 Accidents remain after filtering on Relation to Trafficway: On Roadway .

The charts show the Number of Accidents per each category/range.
▨ shows the active (filtered) selection. (Clear)
■ shows Route: State Highway lock-selection. (Unlock)
🟩 shows Route: Local Street – Municipality lock-selection. (Unlock)

Metric is Number of Accidents. (Change)
Visual scale mode is absolute. (Change)
Measurement labels show absolute (#) value. (Change)

Figure 2. The Overview mode. 1 The interface state is briefly described using the active settings and data features. 2 The user can interact to learn how to change the related states (for example, changing selections or visualization modes).

While this mode reflects the traditional pull approach (DC1) with tag-based filtering and text-search to navigate through help topics, our context-aware ranking improves upon the static help listings and navigation of topics. In contrast, a static topic listing uses only static application-driven context (ADC). In addition, the system provides contextual options to hide (or show) non-relevant topics, and to prioritize unused (or most recently used) features. Providing paths to topics that may be currently irrelevant can help users learn about extended tool capabilities.



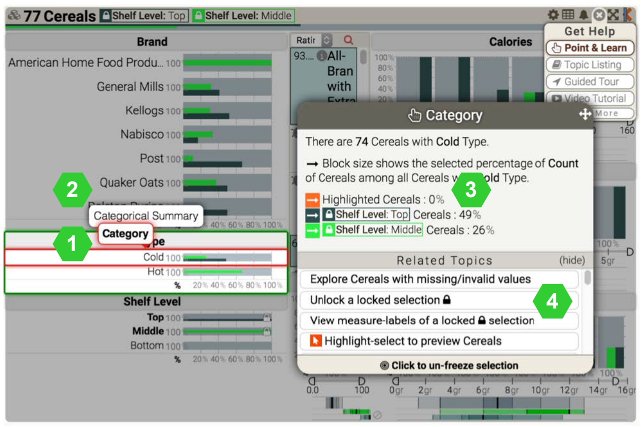
10,000 Bird strikes
How do I ?
explore data
Actions
Select (12)
Explore (12)
Filter (4)
View (1)
Components
Aggregate (5)
Categorical Sum... (3)
Record (2)
Time Summary (1)
✓ Show non-relevant topics
Prioritize unused
Prioritize recently used
Explore Bird strikes with missing/invalid values
Search and filter categories by text
Filter-out a category with Not
Lock-select to compare groups of Bird strikes
Filter-select to focus on selected Bird strikes
Highlight-select to preview Bird strikes
Filter categories in a summary using Or
Filter categories in a summary using And
Adjust filtered range selection
View record details
Get Help
Topic Listing
Overview
Point & Learn
1
2
3
4
5

Figure 3. The Topic Listing mode. 1 Topics are filtered to those relating to Select action, and ranked by contextual relevance. 2 Non-relevant topics are shown with (!), and are ranked below relevant topics. 3 Ranking options can be modified. 4 Topic names reflect the dataset (Records are Bird Strikes) and application state (ex: absolute vs. relative visual scales - not visible in this screenshot). 5 Recently used topics are marked with a clock icon.

Topic Relevance and Ranking by Context. Several different ranking methods are possible. Ranking by application-driven context (ADC) would merely rank help topics based on their overall importance. Ranking by the current data selection would favor operations that are specifically related to the current data-driven context (DDC). Finally, a history-driven context (HDC) ranking would promote recently used actions; or, inversely, ones less recently used (or unused).

Ranking and filtering help topics using contextual information can offer more relevant options up-front and improve navigation. Each topic defines a list of context features (such as data, application, or UI state) to compute topic relevance. This is done by adding the weights of satisfied context features. The richer the required context features of a topic, the heavier the topic weighs. The context feature weights are defined in help material, and reflect the importance and commonality of the feature within expected interface use. The ranked topic relevance is computed using the following strategies):

- **Ranking by data (DDC):** The currently visible data dimensions, data items, or data types (such as the number of numeric attributes) can be used to control the weight. This means that more common features are given a higher ranking.
- **Ranking by UI components (ADC):** If topics reflect multiple targeted UI components (such as by recognizing UI hierarchy), the topic that relate to more specific components are ranked higher based on component specificity.
- **Ranking by history (HDC):** If topics are ranked by recency of use (history), a score that reflects if and how recent the feature was used is added. When ranking for the most recent first, the score is inversely-proportional to how recent the feature was used. When ranking for unused first, the score is highest for topics that have not been used, and lowest for the most recently used.
- **Static:** Topic weight (if defined) is added. This allows adjusting ranking per-topic irrespective of context.



There are 74 Cereals with Cold Type.

→ Block size shows the selected percentage of Count of Cereals among all Cereals with Cold Type.

Highlighted Cereals : 0%
Shelf Level: Top Cereals : 49%
Shelf Level: Middle Cereals : 26%

Explore Cereals with missing/invalid values
Unlock a locked selection
View measure-labels of a locked selection
Highlight-select to preview Cereals

Click to un-freeze selection

Figure 4. The Point&Learn mode. 1 A category is selected by pointing. 2 Its parent, categorical summary, is also highlighted. 3 Descriptions of the category is responsive to data, visualization and selection states. It includes a basic description, the visual encoding, and for each visual feature, describes the encoded value and how to read the interface. 4 Related topics include those pertaining to the category and categorical summary component.

Point & Learn (pull)

In the Point & Learn mode (Figure 4), the user selects an interface or visualization component by hovering their mouse over it. This directly minimizes the separation often inherent in many help systems (DC2) while enabling rapid switching (DC3) and coexisting with the existing data interface (DC4). The help panel shows the information relevant to the pointed element, including its name, description (along with visual encodings and settings), and related help topics, while the pointed element is highlighted using a stencil window and tooltip in the semi-transparent overlay. In other words, this is a fundamentally application-driven help mechanism (ADC), but the help can be extended with data-driven context (DDC) to connect its use to the user.

The hover-action provides responsive interaction design to quickly learn about multiple components. Clicking freezes the selected element, and enables interaction with the help panel (such as activating a related help topic). The freeze-action can also trigger updating help material, such as showing connected components of the selected item (such as a data record) on the interface. The selection can be unfrozen by clicking outside the help panel. The framework recognizes the hierarchical composition of UI elements on pointer-based selection.

For example, a measure label appears inside a category (glyph), which appears inside categorical summary, which appears inside a panel (of the data browser). While the description follows the most specific element (such as the measure label), the help topics and stencils can reflect multiple layers in hierarchy. We limit the hierarchy to two components (self and parent) to retain focus on the material, and not overwhelm the user.

Guided Tour (pull)

The Guided Tour mode (Figure 5) aims to quickly familiarize the user with the interface using a pre-determined sequence of help material (topics or interface components). The user controls the pace by explicitly stepping through the sequence. Related topics to the active step are available on request. The system displays the progress through a dot pattern, and clicking on a dot jumps to the tour to a specific step. If the user exits or changes help mode during the tour, they can later resume from the last active step.

Guided tours bridge to the familiar (DC1) yet minimize separation (DC2); while they are typically **not** rapid, but instead constitute an interface mode of their own, this design also means that they coexist (DC4) well with the existing interface. They also promote demonstration (DC5). While normal guided tours are purely application-driven (ADC), our approach also populates the tour with data from the current dataset, i.e., using data-driven context (DDC).

Notifications (push)

The Notifications mode suggests relevant help topics on-the-fly using an explicit push-model. To not disrupt to the user, we followed a subtle design that uses on the corner to present incoming notifications. On mouse-over, the icon reveals the related task name—an ADC approach—and allows for dismissing the notification. In our current prototype, we enable notifications on a per-topic basis over an extended period of time if the user has not used the relevant feature yet. The notifications can also be used as a tip-of-the-day feature to suggest new topics for revisiting the interface. Generating relevant notifications require detecting user behavior by

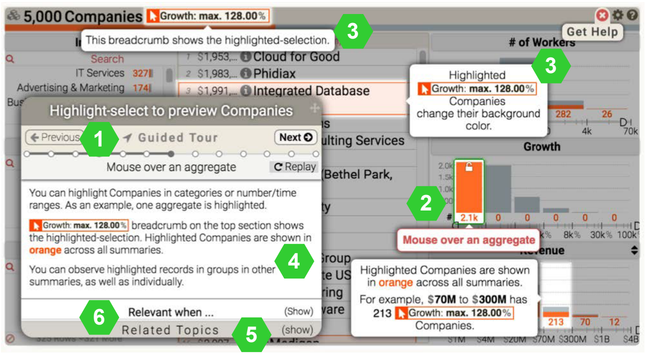

5,000 Companies
Growth: max. 128.00%
This breadcrumb shows the highlighted-selection.
Get Help
Highlighted Growth: max. 128.00% Companies change their background color.
Highlight-select to preview Companies
Previous
Guided Tour
Next
Mouse over an aggregate
Replay
You can highlight Companies in categories or number/time ranges. As an example, one aggregate is highlighted.
Growth: max. 128.00% breadcrumb on the top section shows the highlighted-selection. Highlighted Companies are shown in orange across all summaries.
You can observe highlighted records in groups in other summaries, as well as individually.
Relevant when ... (Show)
Related Topics (show)
Mouse over an aggregate
Highlighted Companies are shown in orange across all summaries.
For example, $70M to $300M has 213 Growth: max. 128.00% Companies.
1
2
3
4
5
6

Figure 5. The Guided Tour mode. 1 The tour progress is visible, and user can control it forward, backward, or to a specific step. This step shows answer to a help topic, "Highlight–select to preview Companies." 2 The tooltip of the main action, which is a mouse-over on a visual glyph, is highlighted by color. 3 Additional tooltips describe the effect of this action on other interface components. 4 Detailed description of the topic presents an easy-to-read summary of tooltips and additional information. 5 Related topics, and 6 the context under which this topic applies can be viewed on demand as well.

tracing their actions (HDC), in addition to taking data (DDC) and application (ADC) context into account.

Notifications promote coexistence (DC4) by subtly integrating help with the existing data interface, and proactively demonstrate the interface to the user (DC5).

While earlier intelligent help systems such as Microsoft's Clippy have not proven to be effective,[34] finding the right content and presentation design for notifications can enable opportunistic learning (DC5). In other words, more semantics and less intrusiveness (DC4) is desired.[28] To achieve unobtrusiveness and usefulness, the notifications should not be frequent (avoid false positives), and help the user when appropriate (avoid false negatives). We present Notifications as a design prototype that covers the explicit push model for help, and we claim no contributions on identifying when to raise notifications.

Topic Answers (pull)

A contextual topic answer aims to ease help comprehension (rather than help seeking), and can be reached through the topic listing or relevant topics of a pointed component. The topic answer is presented directly on the interface by highlighting all the UI components that can achieve or affect the task using a stencil window and tooltips (Figure 6). Help descriptions include not only how to perform the task, but also how it affects the rest of the interface, such as in coordinated-views design pattern for selection tasks.[35] When a dynamic demonstration of an answer is appropriate, the system can present an animated sequence of steps, highlighting information relating to each step directly on the interface. The user can replay these animation sequences to better attend to interaction details and sequences. Clicking on a highlighted UI component passes the mouse-click through overlay and executes the action.

The help overlay closes if there are no other actions to execute for the task, or shows the next action step if other steps remain. When multiple components can achieve the same task, a single tooltip is shown for each component group (Figure 6). The contextual features of a help topic are shown under "Relevant when..." part of the help panel. The selected topic can be non-relevant contextually if one or more context features are not met in live interface, such as when input data does not include the relevant data type, or when data is not filtered for a topic modifying an existing filter selection.

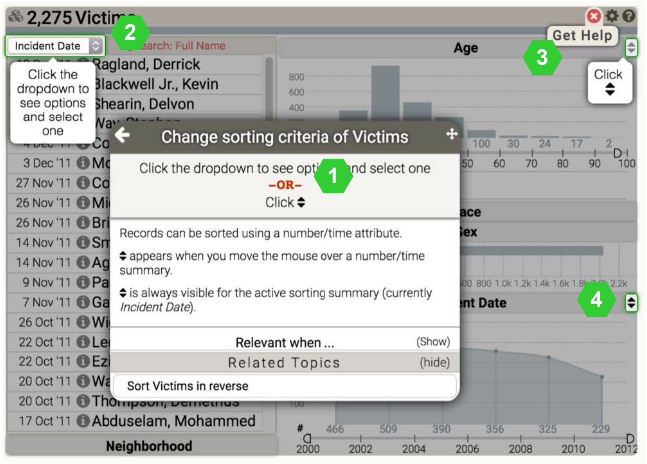

2,275 Victims
Incident Date
Get Help
Age
Click the dropdown to see options and select one
Click
Change sorting criteria of Victims
Click the dropdown to see options and select one
–OR–
Click
Records can be sorted using a number/time attribute.
appears when you move the mouse over a number/time summary.
is always visible for the active sorting summary (currently Incident Date).
Relevant when ... (Show)
Related Topics (hide)
Sort Victims in reverse
Neighborhood
1
2
3
4

Figure 6. The Topic Answer mode. 1 Two distinct actions can satisfy this task, either by 2 using the dropbox, or 3 clicking an icon in a numeric summary. 4 Notice that all relevant icons are highlighted, yet "Click" action tooltip is shown only for one. Clicking on the stencil boxes records the mouse event; the action is executed and HelpIn closes.

Case Study 1: Keshif

Keshif[3] is a data visualization browser designed around coordinated multiple views (CMV),[35] faceted browsing,[36] and brushing and highlighting.[37] The tool can be used to view multivariate tabular datasets, such as relational databases, as well as sets and network data. It is implemented as a web-based application using modern web technologies.

We implemented contextual help as the HelpIn plugin for the Keshif data visualization browser. The program logic of HelpIn is implemented in JavaScript, and help material is also described as JavaScript objects. The material includes lists of contextual features, help topics, UI components (for Point&Learn mode), and guided tour steps. To evaluate the context, HelpIn accesses the DOM of the webpage and/or accesses the underlying application state and dataset in JavaScript directly. It can also modify the application state through this direct code access. The stencil areas used for answers and Point&Learn components are expressed as DOM class names, which also enable detecting help topics for tracking historical context of use (for example, HelpIn can track a click on .summaryCollapse button to "Collapse summary" topic).

To understand how HelpIn influences the help-seeking and learning performance, behavior, and experience of first-time users for data analysis tasks, we conducted a laboratory experiment. For comparison to the contextual in-situ help system (HelpIn), we used non-contextual help topics with non-integrated topic answers (Baseline).' We present a quantitative analysis on performance, and the interactive help system use to answer tasks. We also present subjective feedback of our participants regarding the observed usability and efficiency of the help system and help materials.

Pilot Studies

We ran pilot studies with 4 participants to refine the study protocol. We observed significant variations for help use and analytical reasoning between participants, which limited effectiveness of between participant design protocol across help systems. Within participant design also allowed us to collect feedback on subjective preferences. We also noted that without a brief introduction to help system, the participants could not make informed decisions during the study since they were not aware of help system features.

Participants

After the pilot study, we recruited 14 participants (7 male, 7 female) using university public mail-lists and message-boards. All participants were first-time users of Keshif. Two participants had experience creating visual dashboards using SAP or D3.js. Other participants did not have visual data analytics training beyond basis statistics courses, and most previous experience were related to coursework. They all had some experience with drawing charts using Excel. Thus, the majority were novices in visual data analytics, as well as in Keshif. We compensated participants with $10 cash.

Study Design

We used a within participant design with the help system (Baseline vs. HelpIn) as the independent variable. The ordering of help systems shown to the participants were counterbalanced, i.e. 7 participants completed tasks with Baseline first, and the other 7 started with HelpIn. The system conditions were as the following:

- The Baseline condition included a traditional (non-contextual) topic listing with alphabetical sorting, and did not integrate answers into the interface, i.e. did not include stencil highlights or tooltips. The answers included static media (images, animated gifs) using samples from other datasets.
- The HelpIn condition used contextual and integrated help with Topic Listing, Point & Learn, and Overview modes.

Baseline was created using a stripped-down version of HelpIn to eliminate other differences across the systems. The help material used across the system conditions were the same except the help modes, the use of context, and integration of answers. The material, with 32 topics and 50 components, focused on the exploratory use of Keshif (i.e. did not include authoring data visualizations). We disabled Notifications as its efficiency depends on inferring user behavior with minimal false positives/negatives, which is not among our contributions. We used the Guided Tour mode for training only.

Task Description

The participants were given 12 tasks across three task types and four datasets. The three task types (Explain, Re-target, Analyze) cover both understanding the data interface, and executing actions to achieve desired outcomes. Specifically:

Explain (T1): We asked participants to "Focus on the summary and explain the chart, including the meaning of each color, numbers, and trends you identify." This task is aimed to assess data comprehension. The charts included different data selections and modes, and measured different characteristics across the datasets.

Re-target (T2): We provided a current configuration of the data interface, and a targeted configuration as a screenshot. We asked the participant to "Modify the page on the computer to exactly match the one shown in the screenshot." The target included 2-4 reconfigured settings and adjustments, different for each dataset. This task required understanding multiple differences across two configurations, finding relevant help topics to learn how to make necessary changes (if needed), and executing correct actions.

Analyze (T3): We asked the participant to answer a specific analysis question, such as finding the company with a minimum number of workers, or illnesses count in outbreaks within two states.

The questions required interacting with the interface and changing multiple settings. We used four datasets (companies, bird strikes on airplanes, foodborne outbreaks, and traffic accidents; all tabular datasets of comparable sizes and features) to limit the effects of learning the features of the underlying data. For each dataset, participants answered all three task types in the order noted above. Dataset were also presented in the order noted above. Participants were shown a timer; with 2.5 minutes to complete each task. The tasks across different datasets used targeted features of Keshif, and were of comparable difficulty based on our pilot studies and earlier experience evaluating Keshif. The features considered challenging which would benefit from the help system use included: linked selections, measure metric (count/ sum/ average), visual scale mode (absolute/ relative), label mode (absolute/ percentage), changing histogram axis scale (linear/log), and the use of percentile charts. We created a grading rubric on a [0,5] scale for each task. Zero noted no progress, and five noted a correct answer with all expected outcomes. Since tasks involved changing or describing multiple interface features, the rubric helped assess task progress and differences in performance with more granularity. The task duration limit and complexities created a challenging, yet inviting scenario, where participants had to use the help to best complete the tasks.

Procedure

We asked participants to follow the below procedure during the experiment:

- **Training.** After initially completing a background survey, all participants were introduced to Keshif, and the help system (HelpIn) in approximately 8 minutes. Participants completed a self-paced 12-step Guided Tour for Keshif. Then, the facilitator gave a 1-minute demonstration on how the help system can be opened/closed, and the three help modes: Topic Listing, Overview, and Point&Learn. If participants completed the Guided Tour early, they could choose to explore the tool and help system in the remaining training time. The training involved a separate dataset—homicides in Washington D.C.

- **Tasks.** Next, participants attempted 12 tasks in sequence; with a brief introduction to the task-specific dataset. We did not enforce a specific use of help system—i.e., the participants were free to choose when and how to seek help, and to interpret the material. However, we encouraged participants to use the help system for each task. If the help system was not used before an answer, we gently reminded them about its availability before participants finalized their answer. We encouraged participants to use the help system when they felt stuck, but we did not help them directly—both during training and the tasks. Each task was followed by a task survey on subjective task performance, and usefulness of the help content and system features. After attempting all tasks, participants completed an overview survey and a short interview on effectiveness of various help techniques and materials.
- **Data Collection and Metrics.** Sessions were held in a university lab. Participant sessions lasted for about one hour, and were screen and audio recorded for further analysis. We used Google Chrome on a Macbook Pro with a 15-inch Retina display and a mouse for interaction. We quantitatively measured system use by recording time spent on tasks, help, help mode usage count, task progress, and response time. We also collected survey responses and qualitative data from semi-structured interviews. One of the authors used the grading rubric by reviewing audio and screen recordings to code the interaction features involved while using HelpIn and while performing tasks.

Results

Here we report on the overall performance of our contextual help system in Keshif, as well as specific details about its use.

Performance. We observed no performance differences, measured by task progress, across attempts with HelpIn vs. Baseline conditions (total progress scores 252 vs. 248, given 84 attempts each). We found no major performance difference across those who used HelpIn first, or Baseline first (total progress scores 256 vs. 244, given 84 attempts each). However, we found that participants performed significantly better in attempts where they used Point&Learn, compared to those where they used Topic Listing (with average progress 3.15 vs. 2.51, sample sizes of 82 and 53 attempts). The total progress per participant ranged between 18 to 45 (60 points total), showing significant individual variations in how participants performed. The total scores per task were distributed mostly in [43-57] range (for 9 tasks), while 3 outlier tasks had total scores 14, 18 and 34—indicating that tasks were of comparable difficulty

Time on Help. Of 168 task attempts (12 tasks by 14 participants), only 30 (18%) were finished before time-out, i.e., participants used all allocated time in 82% of their attempts. Thus, we focus our time-analysis on the use of help system. Our participants spent significantly more time with HelpIn than with Baseline (52 vs. 30 second average, sample size: 84 attempts each). This was mainly contributed by Point&Learn (54 sec average, based on 53 attempts where this mode is used), compared to Topic Listing mode (35sec average, based on 82 attempts with this mode). In addition, participants who used Baseline first spent significantly more time on help system compared to those who used HelpIn first (34 vs. 50 second in average, sample size 84). Using HelpIn first reduced total time spent on help, without major differences in task performance.

When Help Is Not Needed. Among 168 total attempts, 42 (25%) did not use help system, which also lead to higher average performance (3.57), compared to attempts with help use (2.77). This suggests that when participants felt confident in taking on the tasks, they did not seek help, and performed objectively better overall. In regards to not seeking help, a participant noted, "If it is a slightly familiar system, and I feel I can get about exploring things on my own, I prefer that than the help." In other reasons, one noted, "I wasn't sure it could really pinpoint what I wanted," and another said, "Because my time is so limited." Of the 30 attempts that were finished before timeout, 15 (50%) did not involve any use of the help system. The 6 remaining help use cases (40%) were to confirm an answer; not to search for answers.

The Characteristics of Help System Use. Figure 7 shows the distributions of the number of times the help system was used. Help was sought in HelpIn more than Baseline (58% vs. 42%). When all modes were available, Point&Learn was used significantly more than Topic Listing, and Overview was only used a few times. Distribution of help use across different datasets shows that tasks on different datasets were of comparable challenge. Participants used the help system 7-16 times in total through the study. Help seeking per task is also distributed between 11 to 21 uses. The outlier task is where the participants could not find answers to necessary steps with ease. We observed that help system was opened 20 times to confirm the answer or observation. 18 (90%) of these cases were with Point&Learn, while 2 were with Topic Listing. This demonstrates Point&Learn can also support the user to confirm or clarify the meaning of data visualizations.

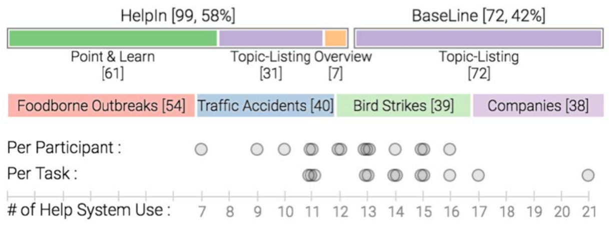
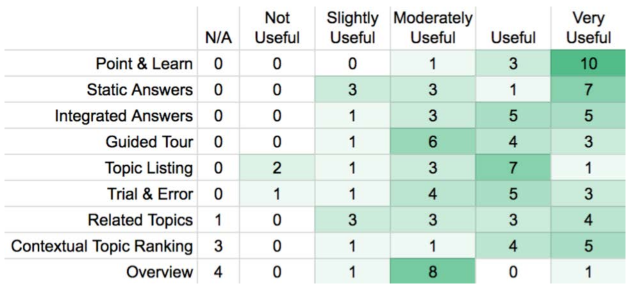


Figure 7. Help mode usage. Top: The distribution of the number of times the help system was used (of 171 total). The distribution across systems (HelpIn vs. Baseline), and help modes. Middle: The distribution across datasets. Bottom: The distributions per 14 participants, and 12 tasks, shown with jitter on overlaps.

Help Topic Listing Search Behavior. Of the 82 attempts that used Topic Listing, topics were searched by tags 41 times (50%) and by text 27 times (33%). Of all the tags selected (63), the majority (55) were action-tags (verbs), instead of component-tags (nouns). Our prototype used strict text matching with topic names, which frequently (22/35) did not return the relevant topics. The failed queries included names of the data attributes (9 cases, such as querying “workers” to find a topic that applies to “number of workers” attribute), as well as synonyms (10 cases, such as querying “combine” to add multiple filters, or “reorder” to sort). These interactions show that participants preferred to search by action rather than component (component names may have been unfamiliar), and that text query search needs to be flexible with synonyms, and match attribute names with components.

Subjective Preferences. Participants rated the help system features at the end of the study (Figure 8). Point&Learn was found to be most useful, as exemplified by the quote: *“(it was) my favorite part of the tool”*. When asked about preferences in static or integrated answer presentation, 8 preferred integrated, and 6 preferred static. A participant noted, *“Integrated answer is definitely tremendously more useful as it showed you on the page itself where to be looking for (. . .) It was able to point you in the right direction”*. In favor of static answers, another noted, *“My attention is so concentrated over (main help box) that I just might miss out on (tooltips) (...) (On integrated answers) I don’t know what the expected outcome would be (...) I really don’t know if I did something right, or if that I am in a wrong state and I have to do something more.”* Therefore, neither approach surpassed the other in our prototype. Preferences are also likely to be shaped by quality and content of help material and personal preferences. We observed that animated GIFs were good for demonstrations, and that Keshif’s integrated answers could provide more animations to demonstrate changes to the interface after user actions. We also noticed that some participants faced challenges while translating questions into relevant topics, i.e., about what to get help. A non-native English speaker noted, *“(English is) not a native language for me, so it’s just a little bit too long, so it’s just slightly helpful”*, while another one contrasted, *“I don’t know if it is possible for the text to be more succinct.”*.



	N/A	Not Useful	Slightly Useful	Moderately Useful	Useful	Very Useful
Point & Learn	0	0	0	1	3	10
Static Answers	0	0	3	3	1	7
Integrated Answers	0	0	1	3	5	5
Guided Tour	0	0	1	6	4	3
Topic Listing	0	2	1	3	7	1
Trial & Error	0	1	1	4	5	3
Related Topics	1	0	3	3	3	4
Contextual Topic Ranking	3	0	1	1	4	5
Overview	4	0	1	8	0	1

Figure 8. Participant feedback. Feedback on feature usefulness by the participants.

Case Study 2: POD-Vis

POD-Vis (*Probing Outcomes Data with Visual Analytics*) is a web-based visualization tool that was developed for visual exploration of large datasets that are often collected by scientists, clinicians, health systems, and private and government stakeholders. POD-Vis users may be employed by large organizations such as health systems and pharmaceutical companies, or small practices such as think tanks and group practices. Users can have diverse data analysis practices with differences in statistical and data exploration abilities, and varying access to resources such as human expertise (Statisticians, Analysts, Clinicians) and data analysis tools. POD-Vis was built to be simple and support preliminary data exploration compared to other visual analytics or statistical tools.

Overview. The POD-Vis tool was designed so that any user can explore a dataset easily—by choosing predictors and outcomes of interest to find meaningful patterns and associations (Figure 1). Data exploration is made possible by generation of quick statistical summaries, automatically generated visualizations such as bar charts, box plots, and spaghetti plots. Users can create cohorts or sub-groups of subjects through interactive data filtering of variables in the datasets. Cohorts are compared by viewing results of statistical analyses, and also through visual exploration of data visualizations. We conducted two rounds of usability testing using two datasets collected by the Michael J. Fox Foundation Parkinson’s Progression Markers Initiative (PPMI).

Help Mode and Formative Testing. Similar to the HelpIn system, we included a *Help Mode* in POD-Vis that can be toggled on and off. The first version of POD-Vis included (i) Point&Learn, and (ii) a static, non-contextual Guided Tour. We conducted a remote task-based user study with 20 experts—colleagues of one of our authors—from across domains (Neurology, Epidemiology, Informatics, Pharmacology) to improve usability and identify new features. Participants were given a verbal, guided tutorial of the tool during this study, and then completed two representative tasks of increasing complexity. Each study session lasted no more than 60 minutes. We identified the need to improve the help features, and to create a standardized help repository.

Evaluation. Following an iterative design process, we improved POD-Vis features based on findings from our formative testing. We recruited 35 participants through personal connections of one of the authors from across domains, and conducted a second round of summative usability testing. Both the usability studies were conducted remotely due to the COVID-19 pandemic. 6 participants quit the study due to emergencies—leading to 29 participants, in total.

- **Training.** POD-Vis is a complex system, and our participants were expected to learn the interface, and interpret the underlying data within 60 minutes. To standardize training, we created a web tutorial containing annotated screenshots with sufficient information to complete the usability tasks. In the tutorial, we included tutorial tasks that were meant to help participants learn the POD-Vis interface. We explained the Point&Learn capability of POD-Vis, and we ensured that participants enabled the Help Mode during the tutorial tasks. Participants could ask the researchers for help during training. Some participants preferred a verbal version of the tutorial (8/29)—some were overwhelmed by all the features, some preferred a guided tour from the researcher "like a YouTube tutorial"; the researcher verbally read and explained the web tutorial to the participants.

- **Tasks.** After completing the tutorial, participants were asked to attempt 6 usability tasks that were designed such that each participant needed to interact with all the main features of POD-Vis. While attempting to complete the 6 tasks, participants had the choice to enable the Help Mode. Point&Learn features were always enabled, but the in-context, positional messages, that were part of the Guided Tour (see 3 in Figure 1).

- **Data Collection and Metrics.** We collected demographics, recorded task completion, and calculated time spent on the tutorial, and each individual tasks. We instrumented the POD-Vis testbed to collect interaction logs such as transition between POD-Vis features, clicks on specific buttons of interest (defined by the research team), and usage of the help mode. We rely on the clickstream timestamps and video analysis, to quantify completion times of various sections of the usability session. At the end of sessions, participants filled a questionnaire that was adapted from the validated System Usability Scale (SUS). Additionally, participants answered questions about the help features.

Results

Overall, 21/29 participants completed all 6 tasks, 4/29 completed 5 tasks, 3/29 completed 4 tasks, and 1/29 completed 3 tasks. 25/29 participants spent an average of 38 minutes attempting and completing the tutorial and five tasks. Task 6 included a discussion with the facilitator by viewing results of Tasks 4 and 5, and therefore could be completed by scrolling alone. We did not log scroll events. On average, participants spent 25.51 minutes on the tutorial alone. Only 13/29 participants chose to enable Help Mode while attempting the tasks. Enabling Help Mode was not influenced by time spent on the tutorial because average tutorial time for participants who enabled (26.52 minutes) Help Mode was very similar to those who did not (24.69 minutes). Similarly, we did not see any difference in task performance between people that used and did not use Help Mode. However, based on subjective feedback (SUS) 16/29 participants "strongly agreed", and 10/29 "somewhat agreed" that the tutorial and help messages were effective in learning the tool. Finally, during our qualitative debriefing at the end of the sessions, participants mentioned that it was easier to rely on the researcher for help during the session, but in their own time, the Help Mode features would be "extremely helpful."

Discussion

Here we discuss our results and their implications for visualization.

Experiment Results: Keshif and POD-Vis

The Baseline condition was a non-contextual version of HelpIn (Keshif) with non-integrated topic answers. To create a shared basis of training material, we avoided using fully-separated help material or videos which may lead to differences beyond help system design. Future studies may target evaluations across media types and designs. Our experiment also did not aim to measure long-term retention, or open-ended use. The effectiveness of the help system may be more pronounced during more ecologically valid settings, i.e., outside a lab setting.

Similar task performance across Baseline and HelpIn in the Keshif study, and in the POD-Vis study could be contributed by the training and study design. However, in both studies, the priming was minimal, and participants would not have been able to complete tasks within the session duration. In Baseline (Keshif), our participants strongly noted the absence of Point&Learn mode, and were less expressive on differences in the presentation of help topic answers and contextual topic ranking, although their final feedback was mostly positive. In addition, our participants showed more progress in tasks in which they used Point&Learn, compared to the tasks where they used Topic Listing. We believe that not yielding expected results (with and without help) is more a reflection of the shared instructional basis (tool design, usability, instructional content, tasks), and the challenges of high-level data analysis, rather than a failing of the help system.

Overall, we think that our studies qualitatively demonstrate the importance of implementing contextual help features in visual data interfaces. In the POD-Vis study, a static web tutorial required users to switch between tabs while attempting the tutorial tasks, which leads to a stronger split-attention effect.[5] Both studies helped us identify opportunities to improve help features, tool usability, and help instructions for Keshif, and POD-Vis. Based on participant feedback and system use, text query search can be improved to find more relevant topics by considering attribute names and synonyms, and Point&Learn components can be narrowed down to the level of glyphs used in visualizations. Videos can also provide additional benefits in explaining interfaces by using spoken (audio) narratives, which may compliment visual channels. Future work can integrate audio into the live interface help.

Generalizing Contextual In-Situ Help for Visualization

Our implementations of contextual in-situ help modes in two different visual data interfaces demonstrate that the core concepts of data, application and historical contexts, and the help modes can be generalized. Both implementations are tool-specific, and currently does not support targeting new interfaces (tools) easily. However, we believe that our help modes can be modularized for the web, and other applications in the future. High-quality material requires careful design and iterative improvements on content and its integration, beyond what a modular implementation may provide out-of-the-box. Our design space and implementation provides a structured basis to undertake similar task for other visual data tools.

Overall, the state of art in help systems for data visualization is curiously lacking. This may be because today's users are highly resistant against looking for help and reading documentation on their own. However, our work in this paper yields hope. We believe that help systems must evolve past the traditional help window with an index and a search feature, and instead become deeply integrated and contextualized in the data visualization application itself. Help should be immediate and unprompted, scaffolding memory for features, yet not obtrusive and disruptive. We believe there is much work to be done in this space to achieve this vision in the future.

Finally, we note that we designed our contextual help systems primarily for practiced use where the user is motivated to learn all of the features of the help system. Accordingly, our user study involved an 8-minute training phase to simulate such practiced use among our participants. It could argued that a help system should also be designed for novice and first-time users and thus require no training. While we agree with this, we first note that we believe our contextual help systems will work well even for a first-time user because of its clear and explicit visual design. After all, the training phase in our evaluation was conducted entirely using the HelpIn Guided Tour itself. Second, we would argue that complex visualization systems such as Keshif and POD-Vis are rarely used by casual first-time users. Still, we agree that understanding first-time use would be highly interesting, and leave understanding such settings for future work.

The Synergy Between Interface Design and Help Design

From the perspective of interface designers and developers, our integrated approach enables preparing and maintaining the training material along with the design and implementation of the interface. This can reduce time-consuming updates to existing material after interface changes, and can shift the preparation of the help material from post-implementation (waterfall model) to the course of interface development. In addition, the design of help material should build upon the design of the interface. While providing help and documentation is necessary to support the wide range of tasks or learning requirements, improving design of the underlying interface should be prioritized to minimize the need for help, and to push towards self-explanatory interfaces. In other words, the help material should not be the primary resource to enable usability.

Conclusion

We have presented a framework for contextual in-situ help systems for visual data interfaces. This framework uses data and visualization features, in addition to application and action history context, to find relevant help material, and to present answers that are integrated and responsive to the active interface and dataset. The approach is also based on both traditional user-initiated pull as well as computer-initiated push requests. We identified five modes to seek for help—Point&Learn, Topic Listing, Overview, Guided Tour, and Notifications—as well as contextual approaches to support both help seeking and help comprehension. While our experiment with participants of mostly data analytics novices show that full-featured contextual help did not improve task performance overall compared to a non-contextual version of the same help material, both performance and subjective feedback highlights the utility of using Point&Learn, one of the modes, to seek help and to perform data analysis.

References

1. Ben Shneiderman, Catherine Plaisant, Maxine S. Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, chapter Documentation and User Support (a.k.a. Help), pages 266–290. Pearson, 6th edition, 2016.
2. Christopher Ahlberg. Spotfire: An information exploration environment. *ACM SIGMOD Record*, 25(4):25–29, dec 1996. ISSN 0163-5808. doi: 10.1145/245882.245893. URL https://doi.org/10.1145/245882.245893.
3. Mehmet Adil Yalçın, Niklas Elmqvist, and Benjamin B. Bederson. Keshif: Rapid and expressive tabular data exploration for novices. *IEEE Transactions on Visualization and Computer Graphics*, 24(8):2339–2352, 2018. doi: 10.1109/TVCG.2017.2723393. URL https://doi.org/10.1109/TVCG.2017.2723393.
4. Keshif. Keshif: Data made explorable, 2022. URL https://keshif.me/.
5. Paul Ginns. Integrating information: A meta-analysis of the spatial contiguity and temporal contiguity effects. *Learning and Instruction*, 16(6):511–525, 2006. ISSN 0959-4752. doi: 10.1016/j.learninstruc.2006.10.001. URL https://doi.org/10.1016/j.learninstruc.2006.10.001.
6. Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, page 159–166, New York, NY, USA, 1999. ACM. doi: 10.1145/302979.303030. URL https://doi.org/10.1145/302979.303030.
7. David G. Novick and Karen Ward. Why don't people read the manual? In *Proceedings of the ACM Conference on Design of Communication*, pages 11–18, New York, NY, USA, 2006. ACM. doi: 10.1145/1166324.1166329. URL https://doi.org/10.1145/1166324.1166329.
8. Hyunmo Kang, Catherine Plaisant, and Ben Shneiderman. New approaches to help users get started with visual interfaces: Multi-layered interfaces and integrated initial guidance. In *Proceedings of the Annual National Conference on Digital Government Research*, page 1–6. Digital Government Society of North America, 2003.
9. John M. Carroll and Caroline Carrithers. Training wheels in a user interface. *Communications of the ACM*, 27(8):800–806, aug 1984. doi: 10.1145/358198.358218. URL https://doi.org/10.1145/358198.358218.
10. Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J. Guo, Robert C. Miller, and Krzysztof Z. Gajos. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, page 4017–4026, New York, NY, USA, 2014. ACM. doi: 10.1145/2556288.2556986. URL https://doi.org/10.1145/2556288.2556986.
11. Nikola Banovic, Tovi Grossman, Justin Matejka, and George Fitzmaurice. Waken: Reverse engineering usage information

and interface structure from software videos. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 83–92, New York, NY, USA, 2012. ACM. doi: 10.1145/2380116.2380129. URL https://doi.org/10.1145/2380116.2380129.

12. Cuong Nguyen and Feng Liu. Making software tutorial video responsive. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1565–1568, New York, NY, USA, 2015. ACM. doi: 10.1145/2702123.2702209. URL https://doi.org/10.1145/2702123.2702209.
13. Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F. Cohen. Pause-and-play: Automatically linking screencast video tutorials with applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, New York, NY, USA, 2011. ACM. doi: 10.1145/2047196.2047213. URL https://doi.org/10.1145/2047196.2047213.
14. Catherine Plaisant and Ben Shneiderman. Show me! Guidelines for producing recorded demonstrations. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 171–178, Piscataway, NJ, USA, 2005. IEEE. doi: 10.1109/VLHCC.2005.57. URL https://doi.org/10.1109/VLHCC.2005.57.
15. Justin Matejka, Tovi Grossman, and George Fitzmaurice. Ambient help. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, page 2751–2760, New York, NY, USA, 2011. ACM. doi: 10.1145/1978942.1979349. URL https://doi.org/10.1145/1978942.1979349.
16. Michael Ekstrand, Wei Li, Tovi Grossman, Justin Matejka, and George Fitzmaurice. Searching for software learning resources using application context. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 195–204, New York, NY, USA, 2011. ACM. doi: 10.1145/2047196.2047220. URL https://doi.org/10.1145/2047196.2047220.
17. Brad A. Myers, David A. Weitzman, Amy J. Ko, and Duen H. Chau. Answering why and why not questions in user interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, page 397–406, New York, NY, USA, 2006. ACM. doi: 10.1145/1124772.1124832. URL https://doi.org/10.1145/1124772.1124832.
18. Tom Yeh, Tsung-Hsiang Chang, Bo Xie, Greg Walsh, Ivan Watkins, Krist Wongsuphasawat, Man Huang, Larry S. Davis, and Benjamin B. Bederson. Creating contextual help for GUIs using screenshots. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, page 145–154, New York, NY, USA, 2011. ACM. doi: 10.1145/2047196.2047214. URL https://doi.org/10.1145/2047196.2047214.
19. Bum Chul Kwon and Bongshin Lee. A comparative evaluation on online learning approaches using parallel coordinate visualization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 993–997, New York, NY, USA, 2016. ACM. doi: 10.1145/2858036.2858101. URL https://doi.org/10.1145/2858036.2858101.
20. Automated Insights. Wordsmith, 2022. URL https://automatedinsights.com/wordsmith/.
21. Narrative Science. Quill for tableau, 2022. URL https://narrativescience.com/quill/tableau.
22. Jeremy Boy, Ronald A. Rensink, Enrico Bertini, and Jean-Daniel Fekete. A principled way of assessing visualization literacy. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1963–1972, 2014. doi: 10.1109/TVCG.2014.2346984. URL https://doi.org/10.1109/TVCG.2014.2346984.
23. M. Adil Yalçin, Niklas Elmqvist, and Benjamin B. Bederson. Cognitive stages in visual data exploration. In *Proceedings of the Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, page 86–95, New York, NY, USA, 2016. ACM. ISBN 9781450348188. doi: 10.1145/2993901.2993902. URL https://doi.org/10.1145/2993901.2993902.
24. John M. Carroll. *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*. MIT Press, Cambridge, MA, USA, 1990. ISBN 02620316390.
25. David E. Caldwell and Michael White. CogentHelp: a tool for authoring dynamically generated help for java guis. In *Proceedings of the ACM Conference on Computer Documentation*, pages 17–22, New York, NY, USA, 1997. ACM. doi: 10.1145/263367.263371. URL https://doi.org/10.1145/263367.263371.
26. Oscar D. Andrade, Nathaniel Bean, and David G. Novick. The macro-structure of use of help. In *Proceedings of the ACM Conference on Design of Communication*, page 143–150, New York, NY, USA, 2009. ACM. doi: 10.1145/1621995.1622022. URL https://doi.org/10.1145/1621995.1622022.
27. Katie Sherwin. Pop-ups and adaptive help get a refresh, Mar 2015. URL https://www.nngroup.com/articles/pop-up-adaptive-help/.
28. Tovi Grossman, George Fitzmaurice, and Ramtin Attar. A survey of software learnability: Metrics, methodologies and guidelines. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 649–658, New York, NY, USA, 2009. ACM. doi: 10.1145/1518701.1518803.
29. Ron Baecker. Showing instead of telling. In *Proceedings of the ACM Conference on Computer Documentation*, pages 10–16, New York, NY, USA, 2002. ACM. doi: 10.1145/584955.584957. URL https://doi.org/10.1145/584955.584957.
30. R Keith Sawyer. *The Cambridge Handbook of the Learning Sciences*. Cambridge University Press, 2005.
31. Caitlin Kelleher and Randy Pausch. Stencils-based tutorials: Design and evaluation. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, page 541–550, New York, NY, USA, 2005. ACM. doi: 10.1145/1054972.1055047. URL https://doi.org/10.1145/1054972.1055047.
32. Gang Wang, Xinyi Zhang, Shiliang Tang, Haitao Zheng, and Ben Y. Zhao. Unsupervised clickstream clustering for user behavior analysis. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, page 225–236, New York, NY, USA, 2016. ACM. doi: 10.1145/2858036.2858107. URL https://doi.org/10.1145/2858036.2858107.
33. Paul Chandler and John Sweller. Cognitive load theory and the format of instruction. *Cognition and Instruction*, 8(4): 293–332, 1991. doi: 10.1207/s1532690xci0804_2. URL https://doi.org/10.1207/s1532690xci0804_2.

34. Microsoft. Farewell Clippy: What's happening to the infamous office assistant in Office XP, Apr 2001.
35. Jonathan C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Proceedings of the International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 61–71, Piscataway, NJ, USA, 2007. IEEE. doi: 10.1109/CMV.2007.20. URL https://doi.org/10.1109/CMV.2007.20.
36. Bongshin Lee, Greg Smith, George G. Robertson, Mary Czerwinski, and Desney S. Tan. FacetLens: exposing trends and relationships to support sensemaking within faceted datasets. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1293–1302, New York, NY, USA, 2009. ACM. doi: 10.1145/1518701.1518896. URL https://doi.org/10.1145/1518701.1518896.
37. Richard A. Becker and William S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987. doi: 10.2307/1269768. URL https://doi.org/10.2307/1269768.