ProReveal: Progressive Visual Analytics with Safeguards

Jaemin Jo, Sehi L'Yi, Bongshin Lee, and Jinwook Seo

Abstract—We present a new visual exploration concept—Progressive Visual Analytics with Safeguards—that helps people manage the uncertainty arising from progressive data exploration. Despite its potential benefits, intermediate knowledge from progressive analytics can be incorrect due to various machine and human factors, such as a sampling bias or misinterpretation of uncertainty. To alleviate this problem, we introduce *PVA-Guards*, safeguards people can leave on uncertain intermediate knowledge that needs to be verified, and derive seven PVA-Guards based on previous visualization task taxonomies. PVA-Guards provide a means of ensuring the correctness of the conclusion and understanding the reason when intermediate knowledge becomes invalid. We also present ProReveal, a proof-of-concept system designed and developed to integrate the seven safeguards into progressive data exploration. Finally, we report a user study with 14 participants, which shows people voluntarily employed PVA-Guards to safeguard their findings and ProReveal's PVA-Guard view provides an overview of uncertain intermediate knowledge. We believe our new concept can also offer better consistency in progressive data exploration, alleviating people's heterogeneous interpretation of uncertainty.

Index Terms—Progressive visual analytics, intermediate knowledge representation, hypothesis testing, scalability, uncertainty.

1 Introduction

We propose *Progressive Visual Analytics with Safeguards*, a novel visual exploration concept that helps people manage the uncertainty arising from progressive data exploration. Progressive visual analytics (PVA) allows people to access the partial results of visualization queries in the middle of computation, helping them make data-driven decisions faster even with large-scale data. However, such *intermediate* knowledge can be incorrect due to various machine and human factors. For example, many PVA systems build and use samples of raw data to estimate results, which leaves a discrepancy between the precise results and the results based on the samples. Another reason can be a human factor such as misinterpreting the uncertainty of intermediate knowledge and making a hasty decision. To address this issue, we introduce *PVA-Guards*, which can be used to validate intermediate knowledge during or after exploration and ensure its correctness.

We are especially interested in exploration scenarios where it is infeasible to obtain precise results during a single session due to a long computation time. In this case, people can rely only on partial and uncertain results to steer their exploration and draw conclusions. As an illustrative example, suppose an analyst, Zoey, is progressively exploring a large sales dataset of an online bookstore. She is interested in gathering useful information about the country that published the most books. Therefore, she first creates a bar chart of the number of books published in each country. The system shows the early estimates of publication counts, and she finds the USA has the highest publication count. However, as the bar chart gets updated (with more data processed), she notices that the difference in the publication counts for the USA and China is small.

Zoey needs to make a decision based on her uncertain intermediate finding—choosing either the USA or China as having the highest publication count—to look into the country next. One common strategy to handle this uncertainty is to wait longer until the visualization looks more certain (e.g., wait for narrower confidence intervals or larger height difference between the two bars). However, few systems guarantee how long she needs to wait to achieve a certain level of trustworthiness, which can decrease the benefits of PVA, that is, allowing early decision making. Using another strategy, Zoey can instead take the risk of proceeding with the uncertainty; she chooses one of the two countries and proceeds to her next visualization. In this case, she needs to consider the worst case of such an optimistic strategy: her choice turning out to be wrong.

Our concept, Progressive Visual Analytics with Safeguards, provides a means of managing such uncertainty resulting from progressive visual analytics by allowing people to leave PVA-Guards on uncertain intermediate knowledge. A PVA-Guard is a hypothetical representation of intermediate knowledge that people garnered during progressive data exploration. In the example of Zoey's exploration, her intermediate knowledge is that the publication count for the USA is greater than that of China, which can be represented as a `Comparative` PVA-Guard, such as $PubCount(USA) > PubCount(China)$, on a bar chart. Once a PVA-Guard is placed, the system statistically estimates the validity of her intermediate knowledge and gives continuous feedback on its validity. In case the PVA-Guard becomes invalid, the system notifies her so that she can manage the incorrect intermediate knowledge. Therefore, with the PVA-Guard on, Zoey can continue her exploration at a desired pace and confidence.

- *J. Jo, S. L'Yi, and J. Seo are with the Department of Computer Science and Engineering, Seoul National University, Korea, Republic of. E-mail: {jmjo, shlyi}@hcil.snu.ac.kr, jseo@snu.ac.kr*
- *B. Lee is with Microsoft Research, Redmond, Washington, USA. E-mail: bongshin@microsoft.com*

Manuscript received July 14, 2019; revised November 1, 2019.

The main contributions of our work are as follows:

- We define PVA-Guards, present concrete examples, and discuss design considerations to realize our new progressive visual analytics concept (section 3);
- We design and implement a proof-of-concept PVA system, ProReveal (Figure 1), integrating seven types of PVA-Guards: `Value`, `Rank`, `Range`, `Comparative`, `Power Law`, `Normal`, and `Linear` (section 4);
- We report a qualitative user study with 14 participants conducted to investigate how people use and interact with PVA-Guards for their progressive exploration (section 5).

2 Related Work

In this section, we elaborate on previous attempts to investigate user human factors in PVA and cover models, systems, and algorithms for enriching the coverage of PVA.

2.1 Human Factors in Progressive Visual Analytics

PVA is meant to address the long latency of interactive systems resulting from large data volumes or the complexity of analytical algorithms [1]. PVA systems aim to provide intermediate results in a reasonable time, which approximate the precise results, to help users make early decisions. The acceptable time limit for intermediate results varies depending on users' situations, dataset sizes, and analysis types; however, a few seconds up to 10 seconds has been regarded as a reasonable and feasible time limit in many progressive systems and applications, aligned with previous guidelines on latency [2], [3], [4].

Human factors and user experiences in PVA have been studied in the human-computer interaction field. Fisher et al. [5] studied how people interpret and use incremental approximate visualization. Their study suggested that people are capable of employing incremental visualizations for faster decision making, advocating the benefits of PVA. To understand how progressive visualizations influence human behavior during exploratory data analysis, Zgraggen et al. [6] compared instantaneous, progressive, and blocking visualizations using insight-based metrics. They found that progressive visualizations outpace blocking ones and show comparable performance to instantaneous ones.

Zhao et al. [7] devised heuristics for eliciting hypotheses from a visualization, and developed multiple hypothesis testing controls to allow people to manage false discoveries. In our study, we opt for a more explicit approach where people present their hypotheses (or intermediate knowledge) to test directly on a visualization. Regarding interactions on progressive interfaces, Wu et al. [8] emphasized the importance of cumulatively rendering asynchronous results to improve the perceived speed and usability of interactive visualizations.

Moritz et al.'s research on optimistic visualization [9] is one of the studies that motivated this research. In optimistic visualization, people explore data "optimistically," trusting early uncertain results from a visualization. Later, when the precise result is ready, people return to the visualization and check if there is a big difference between the earlier one and the precise one, which can be a signal of errors. In our work, however, we consider analysis scenarios where the precise result is not obtainable in a single session due to a long computation time. Therefore, rather than facilitating the comparison between the two results, we provide continuous feedback on the validity of intermediate knowledge that is statistically estimated if possible, and notify people when intermediate knowledge becomes invalid. Furthermore, we allow people to formulate their intermediate knowledge in a structured form (i.e., PVA-Guards) rather than plain text, which is used to delegate the validation process to the system and enable people to continue their exploration.

2.2 Models, Systems, and Algorithms for PVA

A body of research provides theoretical bases and useful models for PVA. For example, Fekete and Primet [10] defined progressive computation as a computation that gives a sequence of partial results with a bounded time limit between two consecutive results, distinguishing it from other computation paradigms, such as online computation [11]. Schulz et al. [12] defined partitioned data and visualization operators to model intermediate visualization updates in their incremental visualization model. Focusing on users, Mühlbacher et al. [13] formalized strategies for increasing user involvement in existing algorithms through two axes: direction of information and entity of interest. Extensive discussion regarding PVA such as its definition, tasks, users, and evaluation methodology also took place in a recent Dagstuhl seminar [14].

Angelini et al. [15] presented a comprehensive survey of existing progressive systems, especially characterizing the requirements and challenges in PVA, and Micallef et al. [16] characterized PVA users by their roles, tasks, and foci. Both studies identified that judging the uncertainty of partial results and handling fluctuating progressions are two of the major challenges in PVA. To tackle these issues, we put forward PVA-Guards as a means of managing the validity of intermediate knowledge from PVA. Angelini et al. [17] proposed ten fundamental quality indicators for indicating the progression, stability, and certainty of intermediate results. ProReveal provides indicators for absolute progress (i.e., the amount of data processed so far), relative progress (i.e., the amount of data processed in the last iteration), and absolute certainty (e.g., a confidential interval for a bar).

With the benefits of PVA, many researchers have endeavored to adopt progressive computation in domain-specific scenarios, expanding the border of PVA. Stolper et al. [1] showed how PVA can facilitate visual exploration of large-scale sequences by implementing a progressive sequential pattern mining algorithm. Based on Apache Spark [18], Jo et al. [19] designed and evaluated a scalable visualization system for exploring large-scale data on a distributed cluster. Enabling PVA on deep neural networks, Pezzotti et al. [20] presented DeepEyes, which supports detailed analysis while training a neural network. Sperrle et al. introduced the concept of Speculative Execution [21] that allows effective model optimization and refinement through progressive computation. Finally, Badam et al. [22] introduced InsightFeed for exploring Twitter data at scale and presented a set of interface design guidelines that can help people understand progressive updates and intermediate results. We believe PVA-Guards can be extended to allow people to verify domain-specific high-level findings.

Progressive exploration of multidimensional data is another important research area. Pezzotti et al. [23] introduced approximate t-Distributed Stochastic Neighbor Embedding (A-tSNE), which significantly lowered the long latency of the t-SNE algorithm [24]. Jo et al. [25] focused on further reducing the initial delay of t-SNE through progressive neighbor computation and presented a responsive t-SNE algorithm. Turkay et al. [26] proposed DimXplorer that integrates the incremental PCA and mini-batch k-means clustering algorithms with data exploration. For more efficient and robust sampling, Rahman et al. [27] proposed sampling-based incremental visualization algorithms that capture the salient feature of a visualization quickly.

3 Progressive Visual Analytics with Safeguards

While PVA allows people to access intermediate results in the middle of computation, the trustworthiness of the results remains equivocal. Such uncertainty can make analysts wait for more certain results or lead them to a hasty decision. We approach this problem by allowing analysts to explicitly leave a PVA-Guard on uncertain intermediate knowledge that needs to be verified. These PVA-Guards not only provide means for testing the correctness of the conclusion drawn from exploration even when analysts are unavailable, but also can be used to trace the provenance of intermediate knowledge when some of them turned out to be incorrect. We envision that PVA-Guards can provide the following benefits:

- *Speed*: People can continue to explore data even when ongoing results are not certain enough, leaving the task of validating intermediate knowledge to the system.
- *Correctness*: The set of PVA-Guards can be used to validate the correctness of intermediate knowledge gathered from exploration later or in the middle of analysis.
- *Trace*: When some intermediate knowledge turned out to be wrong, PVA-Guards can serve as traces of exploration that enable people to alter or re-run the analysis.

3.1 Definition

A PVA-Guard (hereafter, a Guard) is a hypothetical representation of intermediate knowledge that an analyst gathers during progressive data exploration. It can be formally defined as follows:

$$\langle \textit{PVA-Guard} \rangle := \langle variable \rangle \ \langle operator \rangle \ \langle operand \rangle \quad (1)$$

where

$$\langle operand \rangle := \ empty \ | \ \langle variable \rangle \ | \ \langle constant \rangle.$$

For example, a Guard where $\langle variable \rangle$ = $PubCount(USA)$, $\langle operator \rangle$ = $>$, and $\langle operand \rangle$ = $PubCount(China)$ indicates the USA's publication count is greater than China's.

In the definition, the first $\langle variable \rangle$ part refers to the subject of intermediate knowledge that the Guard tests. It can be a single value, such as the value of a cell of a heatmap, the rank of a bar in a bar chart, or even the distribution of values in a histogram. In the middle of progressive exploration, $\langle variable \rangle$ is uncertain and estimated through a statistical procedure if applicable.

The $\langle operator \rangle$ part indicates the type of intermediate knowledge and an operation a Guard performs. For example, common comparison operators (e.g., $\leq$) are useful when we want to compare a variable to a constant or another variable. Other operator can be $\sim$ (i.e., follows) and $\propto$ (i.e., is proportional to) that state a variable (i.e., a distribution of values) follows a certain distribution or is proportional to another variable, respectively.

The last $\langle operand \rangle$ part refers to the object of intermediate knowledge. The type of $\langle operand \rangle$ is determined by the operator. For example, for comparison operators, $\langle operand \rangle$ can be either a constant or another variable. $\langle operand \rangle$ can also be a specific distribution whose parameters are known; for example, when the operator is set to $\sim$, $\langle operand \rangle$ can be a normal distribution such as $\mathcal{N}(20, 10^2)$. Finally, $\langle operand \rangle$ can be unspecified when the operator does not require any operand such as an existence operator that checks whether a variable exists.

A *validity measure* of a Guard provides an estimate about how certain the intermediate knowledge is (Table 1). It can be a boolean value, indicating whether the Guard holds or not, but is not limited to, especially when the intermediate knowledge itself does not have an dichotomous answer (e.g., knowledge that the distribution of values follows a specific distribution). In some cases, a validity measure can be computed during exploration, which would be useful because it can help people judge the trustworthiness of the intermediate knowledge and steer exploration. For example, we can provide statistical significance or a p value on the difference of $PubCount(USA)$ and $PubCount(China)$ through a Student's t-test using sample statistics.

When computing such statistical significance during progressive computation, we can consider the given dataset as a finite population. In this case, processed rows can be regarded as a sample drawn from the finite population without replacement. Because the population is finite, we can obtain a definitive answer on the statistical significance when the entire dataset is processed; for example, a p value will converge to either 0 or 1 in the end. On the other hand, we can view the dataset as a sample drawn from a hidden infinite population (i.e., the world). In this case, even after the whole dataset is processed, the result remains uncertain. In this work, we take the former perspective; we estimate the final result using the statistical procedures for a finite population of size N (i.e., the number of rows in data). Note that by setting N to ∞, we can compute p values and confidence intervals from the latter perspective. Details on the procedures can be found in supplementary materials.

For both cases, however, p values obtained in the middle of computation do not guarantee to faithfully indicate the precise result, so interpretation of these values should be done with care and often requires prior knowledge, considering the importance of the decision.

3.2 Examples

To elicit candidates for meaningful Guards, we started from identifying what knowledge people can gain from a single visualization. We inspected a low-level task taxonomy that consists of the tasks that people perform on a visualization and possible outcomes from the tasks. Amar et al. [28] identified ten low-level tasks of analytic activity in information

TABLE 1
Validity measures of PVA-Guards

Name	Domain	Examples in ProReveal
Probability	$[0, 1]$	p value from a t-test
Quality	$[0, 1]$	Kolmogorov-Smirnov statistic
Error	$[0, \infty)$	Root mean square error (RMSE)
Boolean value	$\{true, false\}$	Estimates on MIN and MAX

visualization: Retrieve Value, Filter, Compute Derived Values, Find Extremum, Sort, Determine Range, Characterize Distribution, Find Anomalies, Cluster, and Correlate. In this section, we demonstrate how the knowledge gained by performing each of these tasks can be represented as a Guard. We categorized these ten tasks into four sets depending on 1) whether intermediate knowledge from the task can be stated and validated as a Guard and 2) the type of a possible validity measure for the Guard.

Tasks as Guards with Statistical Significance. The first set of tasks allows people to represent intermediate knowledge as a Guard, and there is also a statistical test for the validity of the knowledge which gives a statistical significance, such as a p value. For example, the **Retrieve Value** and **Compute Derived Value** tasks are to identify an attribute of a single data item and a derived value (e.g., a mean) of a set of data items, respectively. The intermediate knowledge from these tasks can be an estimate of the target value, and Guards for this value include hypotheses such as the value is greater or less than a threshold, or is in a specific range. For all the cases, p values can be given as validity measures through a t-test, considering the processed data items as a sample of the entire dataset, although those measures should be interpreted with care. Similarly, the results of the **Find Extremum** and **Sort** tasks are a data item and its rank, respectively, which can be described as a Guard stating that the rank of the item is equal to a number, or higher or lower than a threshold.

Tasks as Guards with Validity Measures. The intermediate knowledge of tasks in the second set can be described as a Guard with an interpretable statistic as a validity measure. For example, the intermediate knowledge for the **Characterize Distribution** task can be a specific distribution that data values are expected to follow. In a Guard form, $\langle variable \rangle$ is the distribution of values, $\langle operator \rangle$ is $\sim$, and $\langle operand \rangle$ is the distribution one identified, such as $\mathcal{N}(\mu, \sigma^2)$. To measure how similar the actual distribution of values is to the expected distribution, one can use the Kolmogorov-Smirnov statistic [29] as a validity measure, which is defined as the maximum difference between the cumulative probability functions of two distributions. For the **Correlate** task, the process of determining the relationship between two attributes, both $\langle variable \rangle$ and $\langle operand \rangle$ are the values of two different attributes. $\langle operator \rangle$ will vary depending on the relationship one found; for example, if a linear relationship is of interest, one can use a $\propto$ operator and the error from linear regression between two attributes (e.g., root mean square error, RMSE) as a validity measure.

Tasks as Guards without Validity Measures. The third set of tasks consists of the tasks from which quantifying the validity of intermediate knowledge is infeasible in the middle of computation. For example, the **Determine Range** task is an activity of finding the span (i.e., minimum and maximum) of a given attribute. The intermediate knowledge can be expressed by describing an acceptable range in a similar way to the Retrieve Value task. However, in this case, the validity measure would be a boolean value (*true* or *false*), since it is challenging to reliably estimate the minimum and maximum values of an attribute due to the sensitivity of these values. Similarly, the **Filter** task is designed to find the data items that satisfy given conditions. Possible intermediate knowledge from the task is whether such data items exist, that is, $\langle operator \rangle$ will be $exist$ with $\langle operand \rangle$ of $empty$, but it is also hard to predict the existence robustly before processing the entire dataset.

Tasks as Ill-defined Guards. The last set of tasks includes high-level tasks such as **Find Anomalies** and **Cluster**. One can create a Guard for these tasks such as $Num\ of\ Clusters\ in\ Heatmap = 3$, but it is hard to validate such a Guard even after the entire dataset is processed because these tasks require a choice of complex algorithms and parameters. Guards for these tasks would be useful since they can capture higher-level knowledge, but we leave designing and validating such Guards as future work.

3.3 Design Considerations

In this section, we discuss design considerations in realizing progressive visual analytics with PVA-Guards on interactive visualization systems.

Input: Explicit vs. Implicit. Guards need to capture intermediate knowledge people want to verify, and thus we need to consider the explicitness of interactions that people use to present their intermediate knowledge to a system. One end of the explicitness continuum is a fully explicit input where people clearly write down a Guard, for example, using a programming language. In this case, people may articulate their knowledge most accurately, but such an approach can be cumbersome and interrupt exploration. On the opposite end of the continuum, we can imagine a fictional system that automatically identifies intermediate knowledge without any user intervention (e.g., eye tracking technology can be used to capture the user intention [30]). This non-intrusive approach, however, can be error-prone, leading people to spend more time fixing the incorrectly captured Guards.

Other interaction methods can be placed in the middle of the continuum. One example is semantic interactions [31] where user interactions are associated with an intention, such as moving two document icons closer for presenting similarity between the documents. In this work, we designed a user interface to allow users to explicitly present their knowledge obtained from a visualization to the system.

Validation: Online vs. Offline. Another tension exists regarding when to validate Guards, since verifying the Guards themselves consumes computational resources. Seeking the greatest accuracy, one can validate Guards during progressive data exploration (i.e., online) at the cost of sacrificing some computational resources that could be spent on exploring data. In contrast, one can validate the Guards after the exploration (i.e., offline), which can be done even when one is offline (e.g., after work). In this case, since Guards that

need to be validated are already known, offline computation can speed up the validation process, for example, by testing related Guards together. A hybrid approach is also feasible to balance accuracy and efficiency, for example, by allowing people to choose which Guards to be validated online or offline based on their importance. In ProReveal, we validate Guards online and show their validity measures in a PVA-Guard list (subsection 4.3).

Violation: Passive vs. Active. When a Guard turns out to be wrong (e.g., the value of a variable is out of an expected range, the error of linear regression is too large), a system needs to report such a violation to analysts. Simply, the system can take a passive approach, for example, by alerting the analysts and waiting for actions. This is similar to common monitoring systems such as intrusion detection systems for network security [32]. Analysts can prescribe how to react to such violations in advance, which allows more active and complex operations. More advanced systems would be able to suggest new values or parameters for the broken Guard and automatically re-run the analysis even when analysts are offline. In all the cases, the Guards can indicate the provenance of intermediate knowledge and can be used to manage the violation. In ProReveal, we employ a passive approach; we present the validity measures of PVA-Guards and let analysts manage violations.

4 ProReveal

To provide a clear example of our concept in practice, we designed and implemented a proof-of-concept system, ProReveal. ProReveal integrates PVA-Guards into progressive data exploration, allowing people to seamlessly articulate their findings as Guards in the middle of exploration. With ProReveal, we realize seven important Guards: `Value`, `Rank`, `Range`, `Comparative`, `Power Law`, `Normal`, and `Linear`. In this section, we elaborate on our design rationale and the challenges we confronted while integrating PVA-Guards into a progressive data exploration system.

4.1 Design Rationale

DR1: Seek Simplicity but Include Essentials. Serving as a proof-of-concept system for demonstrating the seven Guards, ProReveal is designed to provide an initial platform for observing how people use and interact with Guards in progressive data exploration. To this end, we sought simplicity while including the features essential in progressive visual analytics to our design. For example, ProReveal implements important requirements for PVA, such as uncertainty visualization, feedback on progress, execution control, prioritization, and providing quality measures. Nonetheless, we keep the remaining parts of ProReveal as simple as possible. For example, we decided to provide only two types of visualization (i.e., a gradient plot and a heatmap with Value-Suppressing Uncertainty Palettes [33]) and limit the types of visualization queries.

DR2: Allow Explicit Presentation of PVA-Guards. On a continuum of the explicitness of Guard presentation, we designed our system to support explicit presentation of Guards; people explicitly demonstrate directly on the visualization what intermediate knowledge they want to keep. In ProReveal, people can click on a visual element (i.e., a bar in a bar chart or a cell in a heatmap) and select the type of Guard that they want to leave on the element (Figure 2b). This can allow people to create Guards in a familiar and accurate manner, preventing potential errors that can occur when a more implicit method is employed.

DR3: Respect Intermediate Results. Theoretically, people can make Guards that are inconsistent even with the current intermediate results. For example, suppose a bar whose value is 50 with a standard error of 10, estimated by randomly sampling 10% of data. One can make a Guard that confirms the value of the bar is greater than 100, but it is unlikely to happen. In ProReveal, the use of such overly optimistic Guard is discouraged by permitting only Guards that respect the current intermediate results. For instance, in a `Value` Guard that compares the value of a bar or a cell with a given constant, the operator (i.e., $\geq$ or $\leq$) is automatically set by comparing the current value and the constant. Another example can be a `Range` Guard where the center of the target range is always set to the current value of a bar (i.e., only the width of the range can be controlled).

DR4: Keep PVA-Guards Visible. The collection of Guards can serve as an overview of progressive data exploration and the provenance of knowledge generated. Moreover, an unexpected value for their validity measures can be a signal for further inspection or altering the direction of exploration. To gain these benefits, we keep Guards and their validity measures always visible on the interface (i.e., in a PVA-Guard view, Figure 1d). We also compute and update the validity measure of a Guard online each time its source visualization (i.e., one on which the Guard is created) is updated to facilitate such steering.

4.2 Progressive Visualization

Visualization in PVA systems is required to be scalable and effectively encode uncertainty [19]. We chose to use two visualization techniques from literature: gradient plots [34] for univariate visualization and heatmaps with the Value-Suppressing Uncertainty Palettes (VSUP) [33] for bivariate visualization. Both visualization techniques can provide perceptually effective ways to encode uncertainty; gradient plots show the confidence intervals of values as gradients and VSUPs represent uncertainty through the lightness and saturation channels.

In ProReveal, people are allowed to select up to two fields that they want to include in a new visualization. The type of a new visualization is determined depending on the types of the selected fields: a gradient plot is used for a single categorical field (*C*), a single quantitative field (*Q*), and a pair consisting of a categorical field and a quantitative field (*CQ*), while a heatmap is used for two fields of the same type (*CC* and *QQ*). Hereafter, we use the letters *C* and *Q* before the name of fields and visualizations to denote the types of the fields and the types of the selected fields for the visualization (e.g., a *C* gradient plot for a gradient plot with a categorical field), respectively.

Gradient Plots. A gradient plot encodes the confidence interval of a value using opacity; 95% two-tailed t confidence intervals are shown fully opaque, while outside of the intervals, the opacity decays with respect to the cumulative

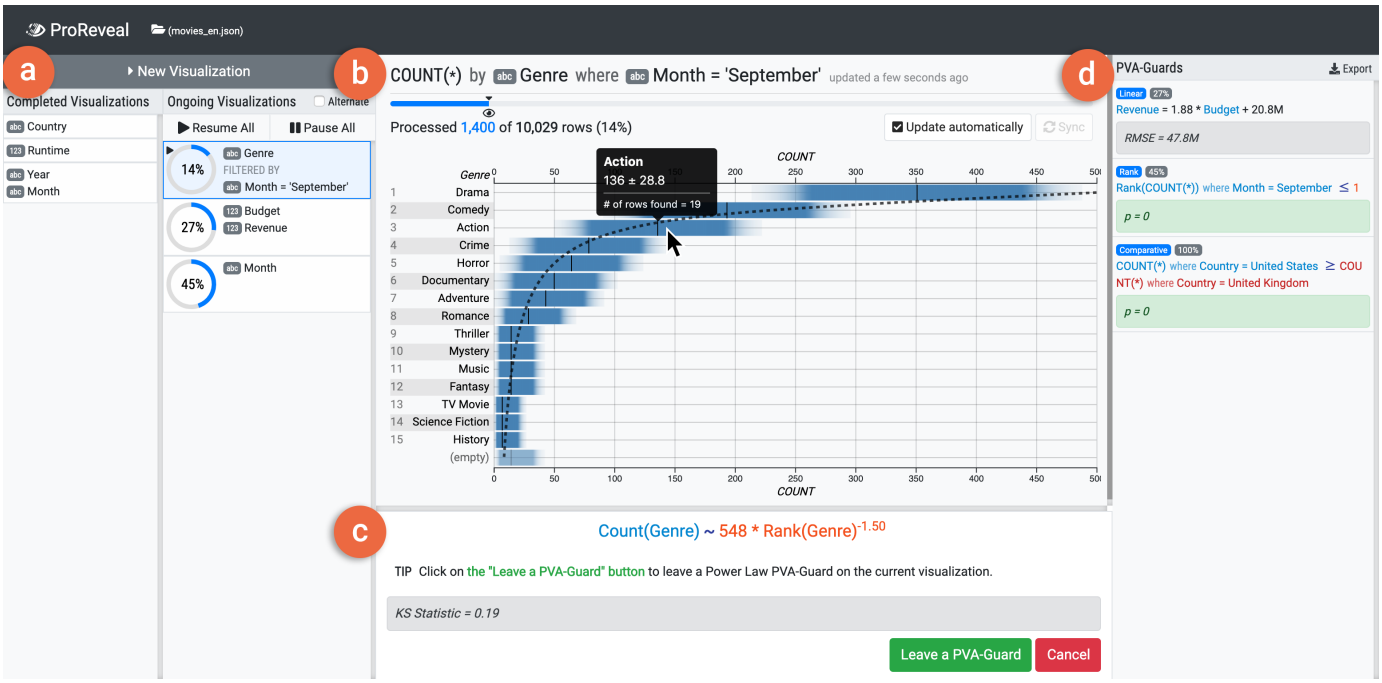


Fig. 1. ProReveal allows people to leave a safeguard, which we call *PVA-Guard*, on the intermediate knowledge found during progressive data exploration. a) Two visualization lists, one for ongoing visualizations and the other for completed visualizations, provide feedback on progress and the ability to control the execution. b) In the main view, ProReveal uses two types of progressive visualizations, gradient plots and heatmaps to visualize the uncertainty underlying intermediate results. A gradient plot is showing the estimated counts of movies of each genre. c) In the PVA-Guard panel, an analyst is creating a Power Law PVA-Guard to leave intermediate knowledge that the distribution of values follows a power law distribution. d) The PVA-Guard view gives an overview of the created PVA-Guards and shows the estimates of their validity (i.e., validity measures).

probability of an underlying t distribution [34] (Figure 1). We call a gradient in a gradient plot a *bar*, and the center of a gradient (i.e., an estimated value) the *value* of a bar.

In ProReveal, we display gradient plots horizontally for better scalability on the number of displayable categories. C gradient plots show the number of data items each category of a C field has. We sort the categories in descending order by their counts, so that the category with the most data items is shown on the top. When C and Q fields are chosen, the Q field is aggregated over the C field through an aggregation function (i.e., MEAN, SUM, MIN, or MAX). Then, the result is visualized in a CQ gradient plot where the categories are sorted by the value of the aggregated Q field in descending order, consistent with C gradient plots.

Q gradient plots bin a Q field and visualize the number of data items in each bin, which can be seen as a histogram with uncertain counts. In contrast to C gradient plots, we do not sort the bins by their counts to maintain the order of bins. The bin size is determined by an initial sample; we first compute the range of a fixed number of sample values (i.e., 200) and divide the range into a specific number of bins (i.e., 40). We adjust the number of bins, if needed, to ensure the edges of the bins are nice numbers. When a value that lies outside of the range is found during computation, the range is stretched and new bins are created to encompass the value (as was done in a previous study [19]).

Heatmaps. Heatmaps show the number of data items for each combination of categories (for CC heatmaps) or bins (for QQ heatmaps) as a matrix (Figure 2). The color of a cell represents the estimated value (i.e., count) and its standard error using a VSUP [33]. A VSUP encodes a value using the hue of a color (i.e., through the viridis colormap [35]) and the uncertainty of the value using the luminance and saturation of the color. CC heatmaps place the categories of each C field on the horizontal and vertical axes respectively, and similar to C and CQ gradient plots, the categories are sorted by the number of data items in each category. Instead of categories, QQ heatmaps create bins for each Q field and show the count of data items in each 2D bin. For both Q fields, we use the same procedure as Q gradient plots to determine the number and size of bins.

4.3 The ProReveal Interface

ProReveal employs a web-based user interface that enables progressive visual analytics with PVA-Guards (Figures 1 and 2). ProReveal supports features that are essential in progressive data exploration, such as feedback on progress, execution control, and prioritization (DR1). In this section, we briefly describe the ProReveal interface and interactions. For more detail, please refer to the videos in the supplementary materials. A self-contained demonstration of ProReveal is also available on the Web (see subsection 4.5). The ProReveal interface consists of three parts: visualization lists, a main view, and a PVA-Guard list.

Visualization Lists. In ProReveal, all visualizations are listed in one of the two juxtaposed lists: one for completed visualizations on the left and the other for ongoing visualizations on the right (Figure 1a). When an ongoing visualization is completed, it moves to the completed visualization list. Both lists display a summary of each visualization by presenting the fields used in the visualization, including those for filters. On the ongoing visualization list, a progress

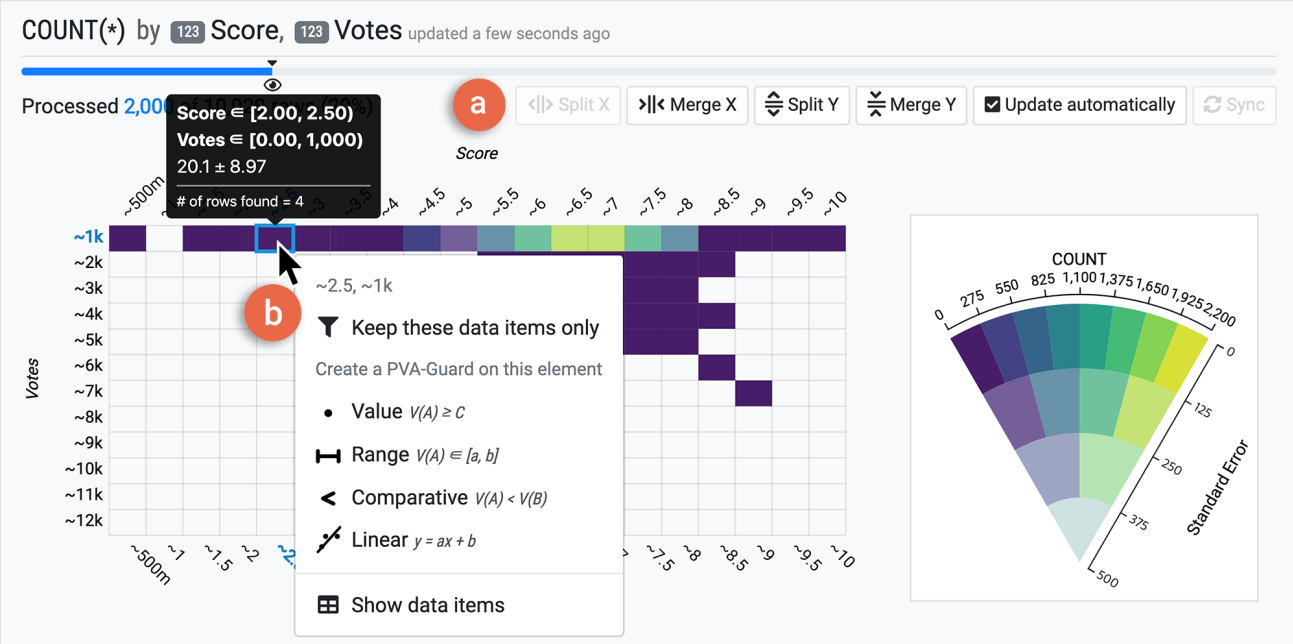


Fig. 2. The main view with a heatmap. a) A toolbox on the main view provides additional features of a visualization, such as changing the number of bins or postponing automatic updates of a visualization. b) The PVA-Guard menu, appearing when people click on a bar or a cell, enables three special operations on the visual element: 1) creating a new visualization only with the data items under the element, 2) leaving a PVA-Guard on the element, and 3) showing the underlying data items.

ring shows the progress, which becomes hidden when completed. People can pause, resume, and remove a visualization through the buttons that appear when they hover the mouse cursor on a summary (i.e., execution control).

Multiple visualizations can exist in the ongoing list, but only one visualization can be computed at any moment. The visualization being computed is marked with a play icon on its progress ring. By default, the topmost visualization on the ongoing list is processed first to the end. But, one can reorder the visualization on the ongoing list by drag-and-drop interaction. Alternatively, one can check an "Alternate" button to alternate computation between visualizations.

Visualization Creator. To create a new visualization, one can open a visualization creator by clicking on the "New Visualization" button above the visualization lists. The visualization creator shows the list of fields in a dataset labelled by type ("abc" for categorical and "123" for quantitative). One can choose up to two fields, and when one chooses one *C* field and one *Q* field (i.e., for a *CQ* gradient plot), a list of available aggregate functions (i.e., MEAN, SUM, MIN, and MAX) appears below the field list. Based on the types of the chosen fields, ProReveal automatically creates a new visualization (DR1) and adds it to the top of the ongoing visualization list.

Main View. The main view (Figures 1b and 2) shows the currently selected visualization. By default, a visualization is immediately updated each time a new partial result is available. However, one can postpone the automatic updates by unchecking the "Update automatically" button and manually refresh the visualization through the "Sync" button (Figure 2a). Due to a long-tail distribution of values, visualizations for large-scale data often suffer from excessive white space. For *C* fields, we limit the number of categories visible to a fixed number (i.e., 50), and people can see all the categories if they need. For *Q* fields, we allow people to change the granularity of bins on the fly, but the minimum bin size is determined based on a small sample (as mentioned in subsection 4.2), and one cannot split the bins smaller than that size. When one hovers the mouse cursor over a visual element (i.e., a bar or a cell) in the main view, a tooltip pops up, describing the corresponding categories or intervals of the element, the estimated value (i.e., an aggregate value for *CQ* gradient plots or a count otherwise) and its standard error, and the number of data items found for the corresponding categories or intervals.

TABLE 2
Types of applicable PVA-Guards for each field combination

	Visualization				
	Gradient Plots			Heatmaps	
Field Types	*C*	*Q*	*CQ*	*CC*	*QQ*
Value	O	O	O	O	O
Rank	O		O		
Range	O	O	O	O	O
Comparative	O	O	O	O	O
Power Law	O	O			
Normal		O			
Linear					O

PVA-Guard Menu. The PVA-Guard menu (Figure 2b), which can be opened by clicking on a visual element of interest (hereafter, target), shows three types of operations one can perform on the target: 1) creating a new visualization only with the underlying data items of the target (i.e., filtering), 2) leaving a Guard on the target, and 3) showing the underlying data items of the target in a pop-up

table. When the filter operation is chosen, the visualization creator appears on the visualization, but in this case, a new visualization is computed only for the data items of the target, not for all data items in the dataset. The types of available Guards vary depending on the visualization and the field types selected (Table 2), and only applicable Guards are shown in the PVA-Guard menu.

PVA-Guard Panel. When a Guard type is chosen on the PVA-Guard menu, the PVA-Guard panel appears (Figure 1c) below the main view, previewing the Guard and its validity measure. $\langle variable \rangle$ of the Guard is set to the target element on which the PVA-Guard menu was invoked. One may want to set $\langle constant \rangle$ of the Guard; we design the interactions for setting $\langle constant \rangle$ according to the Guard type (see subsection 4.4). $\langle operator \rangle$ of the Guard is automatically chosen based on the Guard type and $\langle constant \rangle$ (DR3). Finally, one can click on the "Leave a PVA-Guard" button in the panel to activate the Guard.

PVA-Guard List. The PVA-Guard list provides an overview of created PVA-Guards and their validity measures (DR4, Figure 1d). We decided not to provide a history of a validity measure, since it is not meaningful and can give the illusion that the measure is converging, especially for p values. We color-encoded p values to facilitate interpretation; a p value is shown in green when it is lower than a confidence level (i.e., $\alpha = 0.05$), red when it is higher than 0.5 (i.e., worse than a random guess), and yellow otherwise. When people hover the cursor over a Guard, the linked visualization (i.e., the visualization on which the Guard is created) is highlighted in yellow in the visualization lists, and when people clicked a Guard, the linked visualization is shown in the main view. Finally, people can export and download Guards in a JSON format for future use.

4.4 PVA-Guards

As shown in subsection 3.2, we identified what intermediate knowledge people can derive from our target visualizations (i.e., gradient plots and heatmaps) based on previous literature [28], [36] (Table 3). Based on our identification, we designed and implemented seven Guards in ProReveal: `Value`, `Rank`, `Range`, `Comparative`, `Power Law`, `Normal`, and `Linear`.

4.4.1 Value

One common task in a bar chart and a histogram is to read the value of a specific category or interval. This task has different names in the literature, such as Retrieve Value [28] or Identify Attribute [36]. In ProReveal, such tasks correspond to reading the value of a bar on a gradient plot or a cell on a heatmap. A `Value` Guard allows people to present intermediate knowledge obtained while performing those tasks: the value of a bar or a cell is less or greater than a specific constant (e.g., $Price(Apple) \leq \$2$).

- $\langle variable \rangle$:= value of a bar | value of a cell (e.g., $Price(Apple)$)
- $\langle operator \rangle := \geq \mid \leq$
- $\langle constant \rangle$:= a number (e.g., $2)

Interaction. When the `Value` Guard is selected on the PVA-Guard menu, an orange constant bar appears on the visualization space (for a gradient plot) or the legend (for a heatmap), showing the value of $\langle constant \rangle$. People can drag the constant bar left or right to change the value. By default, $\langle constant \rangle$ is set to the current value of $\langle variable \rangle$, and $\langle operator \rangle$ is set to $\leq$. $\langle operator \rangle$ is automatically chosen by comparing the value of $\langle variable \rangle$ and $\langle constant \rangle$ when the Guard is created (DR3).

4.4.2 Rank

A `Rank` Guard indicates that the rank of a category is higher or lower than a threshold (e.g., $Rank(Price(Apple)) \leq 10$). The `Rank` Guard is related to tasks of Find Extremum and Sort [28]. The `Rank` Guard is available in a *C* or *CQ* gradient plot where the target rank can be directly specified as a horizontal line on the visualization.

- $\langle variable \rangle$:= rank of a bar (e.g., $Rank(Price(Apple))$)
- $\langle operator \rangle := > \mid \leq$
- $\langle constant \rangle$:= a rank (e.g., 10)

Interaction. An orange horizontal bar (i.e., the rank bar) appears on the gradient plot, showing the rank of $\langle constant \rangle$. People can drag the rank bar up and down to set the desired rank. By default, $\langle constant \rangle$ is set to the current rank of $\langle variable \rangle$, and $\langle operator \rangle$ is set to $\leq$. $\langle operator \rangle$ is automatically chosen by comparing the rank of $\langle variable \rangle$ and $\langle constant \rangle$ when the Guard is created (DR3).

4.4.3 Range

A `Range` Guard is a two-sided version of a `Value` Guard. The `Range` Guard indicates that the value of a bar or a cell is in a certain range (e.g., $Price(Apple) \in [\$1, \$3]$). With DR3, we only allow symmetric ranges whose center is at $\langle variable \rangle$ when the Guard is created, preventing people from choosing an arbitrary range. Therefore, the `Range` Guard can be seen as representing acceptable bounds that $\langle variable \rangle$ lies on at the end.

- $\langle variable \rangle$:= value of a bar | value of a cell (e.g., $Price(Apple)$)
- $\langle operator \rangle := \in$
- $\langle constant \rangle$:= a range whose center is at $\langle variable \rangle$ (e.g., $[\$1, \$3]$)

Interaction. When the `Range` Guard is chosen, a gray brush appears on the visualization space (for a gradient plot) or the legend (for a heatmap), showing the range of $\langle constant \rangle$. The shape of the brush depends on the visualization type (i.e., rectangular-shape for a gradient plot and pie-shape for the heatmap legend). People can drag the left or right edge of the brush to change the range. The center of the range stays at the current value of $\langle variable \rangle$; as $\langle variable \rangle$ is progressively updated, the center of the range moves, maintaining its width. By default, $\langle constant \rangle$ is set to the 98% confidence interval of $\langle variable \rangle$.

4.4.4 Comparative

A `Comparative` Guard allows people to safeguard comparison between two values, a common task performed on visualization [36]. The `Comparative` Guard takes and compares two variables of the same type (e.g., values of two bars or two cells), e.g., $Price(Melon) \geq Price(Apple)$.

TABLE 3
Examples of PVA-Guards, validity measures, and their corresponding tasks

Name	Example ($\langle variable \rangle$ $\langle operator \rangle$ $\langle constant \rangle$)	Validity Measure	Relevant Tasks	
			Amar et al. [28]	Munzner [36]
`Value`	$Price(Apple) \leq \$2$	p or *Boolean value*	Retrieve Value & Compute Derived Value	Identify Attribute
`Rank`	$Rank(Price(Apple)) \leq 10$	p	Find Extremum & Sort	Identify Extremum
`Range`	$Price(Apple) \in [\$1, \$3]$	p	Retrieve Value & Compute Derived Value	Summarize Attribute
`Comparative`	$Price(Melon) \geq Price(Apple)$	p	-	Compare Attribute
`Power Law` `Normal`	$Prices\ of\ Fruit \sim a \times Rank^{-k}$ $Prices\ of\ Fruit \sim \mathcal{N}(\mu, \sigma^2)$	*Quality*	Characterize Distribution	Summarize Trend
`Linear`	$Prices\ of\ Fruit \propto Sizes\ of\ Fruit$	*Error*	Correlate	Identify Correlation

- $\langle variable1 \rangle$:= value of a bar | value of a cell (e.g., $Price(Melon)$)
- $\langle operator \rangle$:= $\geq$ | $\leq$
- $\langle variable2 \rangle$:= value of a bar | value of a cell (e.g., $Price(Apple)$)

Interaction. $\langle variable1 \rangle$ is set to the element (i.e., a bar or a cell) on which the PVA-Guard menu was invoked with click, while people can right-click on a bar or a cell to set $\langle variable2 \rangle$. ProReveal automatically chooses $\langle operator \rangle$ by comparing the value of $\langle variable1 \rangle$ and $\langle variable2 \rangle$ when the Guard is created (DR3).

4.4.5 Power Law and Normal

Identifying the distribution of values is another important task [28], [36]. `Power Law` and `Normal` Guards (hereafter, distributive Guards) present intermediate knowledge that the distribution of values follows a power law or normal distribution, respectively, e.g., $Prices\ of\ Fruit \sim \mathcal{N}(\mu, \sigma^2)$. The `Power Law` Guard is available for both *C* and *Q* gradient plots, but the `Normal` Guard is only available for *Q* gradient plots, since a normal distribution requires quantitative values for its domain. Following DR3, the parameters (e.g., μ and σ for a normal distribution) of the distributive Guards are automatically fit to the data at the moment of creation. Moreover, we chose to update the parameters to fit the most recent data, even after the Guard is created, since in exploratory analysis people often want to identify the shape of distribution rather than check if the distribution has specific parameters.

- $\langle variable \rangle$:= values of bars (e.g., $Prices\ of\ Fruit$)
- $\langle operator \rangle$:= $\sim$
- $\langle constant \rangle$:= $PowerLaw(k, \alpha)$ | $\mathcal{N}(\mu, \sigma^2)$

Interaction. For consistency, people have to open the PVA-Guard menu first by clicking a bar to create the distributive Guards, even though those Guards are not for a single bar but for the whole gradient plot. Then, a dotted curve appears on the gradient plot (Figure 1b), showing the current distribution (i.e., $\langle constant \rangle$). As the gradient plot is updated progressively, the parameters are automatically changed to fit the most recent result.

4.4.6 Linear

Designed for tasks of identifying the correlation between two quantitative fields [28], [36], a `Linear` Guard indicates that two *Q* fields are linearly correlated, e.g., $Prices\ of\ Fruit \propto Sizes\ of\ Fruit$. The `Linear` Guard is only available for *QQ* heatmaps, and the values of the second *Q* field is linearly modeled by the first one.

- $\langle variable1 \rangle$:= a *Q* field (e.g., $Prices\ of\ Fruit$)
- $\langle operator \rangle$:= $\propto$
- $\langle variable2 \rangle$:= a *Q* field (e.g., $Sizes\ of\ Fruit$)

Interaction. Similar to distributive Guards, people open the PVA-Guard menu first by clicking any cell on a heatmap to create a `Linear` Guard. Then, a dotted line appears on the heatmap, showing the fitting line. As the heatmap is updated progressively, the parameters of linear fitting (i.e., the slope and the intercept) are automatically changed to fit the most recent result.

4.5 Estimation and Implementation

For progressive computation, ProReveal builds uniform samples of a dataset without replacement and processes the samples one by one to yield progressive results. For each sample, we compute the count (for both *C* and *Q* fields), sum, and squared sum (for *Q* fields) of data values and accumulate the numbers over samples. For visualizations that use COUNT, MEAN, or SUM aggregation functions, we statistically estimate the target value and its standard error using the accumulated statistics. For MIN and MAX aggregation functions, which are more sensitive to outliers, we only show the MIN or MAX value we found so far. For scalable computation, ProReveal processes large-scale data on a distributed computing engine, Apache Spark [37], with a similar architecture to a previous study [19]. Please refer to the videos in the supplementary materials to check out our system running on about 1.7 billion entries of the GAIA dataset [38], [39].

ProReveal employs four types of validity measures (Table 1): p values for `Value`, `Rank`, `Range`, and `Comparative`, quality (e.g., Kolmogorov-Smirnov statistic) for `Power Law` and `Normal`, error (e.g., root mean square error) for `Linear`, and truth values for Guards on visualizations with MIN and MAX aggregation functions. The

validity measures are computed using the sample statistics; therefore, if a visualization is paused, the Guards left on it are not updated. For the progressive computation of uncertainty and validity measures, we assumed that the number of rows in the dataset is known. For more detail, please refer to our supplementary materials.

The metadata of fields, such as type, are conjectured using the fixed number (i.e., 200) of data items, as mentioned in subsection 4.2. However, people can specify the metadata in a separate file, such as the range or desired bin size of a *Q* field. The ProReveal interface is implemented using TypeScript [40], D3.js [41], and Angular [42]. The source codes of the interface and backend are available at https://github.com/proreveal. An interactive demo is also available at https://proreveal.github.io/ProReveal.

5 Evaluation

We conducted a user study to understand if and how people use PVA-Guards and to evaluate the usability of ProReveal.

5.1 Study Design

Participants. We recruited 14 participants (3 females and 11 males) from a university, ranging in ages from 20 to 31 years. We screened the participants through a questionnaire to ensure that 1) they were familiar with using and interpreting common visualizations (e.g., bar charts) and 2) took at least one statistics class with understanding of statistical hypothesis testing and confidence intervals. They received about US$20 for their participation.

Tasks and Datasets. We designed two tasks to evaluate different aspects of data exploration with Guards. From Task 1, we wanted to assess the usability of our interface and interactions for creating Guards. We provided the participants with a structured form of interaction sequences (i.e., exploration recipes). The participants were asked to follow five exploration recipes and answer the question at the end of each recipe as accurately as possible in three minutes.

Consisting of three steps and one question, an exploration recipe describes interactions occurring in analysis where a participant creates a visualization (VIS1 step), leaves a Guard on a finding from the visualization (Guard step), creates another visualization by applying a filter from the first one (VIS2 step), and answers a specific data-driven question. In the VIS1 step, participants were asked to create a visualization by choosing one or two fields in the visualization creator. Then, in the Guard step, they were instructed to make a Guard on the visualization; the parameters for the Guard, such as type, variable, and constant, were described in the recipe, if needed. Next, the participants were asked to create another visualization (VIS2) by selecting a specific category or interval on the first visualization. Finally, they were asked to identify the category or interval with the most data items by answering questions with five choices. If the visualization was not certain enough, the participants needed to wait for a clearer answer.

We designed five templates for the exploration recipes (Table 4) that cover five types of Guards (`Value`, `Rank`, `Range`, `Comparative`, and `Linear`) and five field combinations (*C*, *Q*, *CQ*, *CC*, and *QQ*). To control the difficulty, all templates use the same number of fields in total (i.e., three in VIS1 and VIS2). Based on the templates, we created 10 exploration recipes, R1–R5 with a weather dataset for a tutorial and R6-R10 with a birdstrike dataset for Task 1.

TABLE 4
Five templates of exploration recipes

Name	VIS1	Guard	VIS2
R1, R6	*C*	Rank	*QQ*
R2, R7	*Q*	Range	*CC*
R3, R8	*CQ*	Comparative	*C*
R4, R9	*CC*	Value	*Q*
R5, R10	*QQ*	Linear	*C*

With Task 2, we wanted to investigate how participants use ProReveal in progressive data exploration and how they employ Guards when they are not explicitly instructed to use them. We asked participants to explore the given data to find meaningful and trustworthy insights that they want to share with colleagues, considering themselves data scientists, which is similar to the approach used in the evaluation of previous data exploration systems [43], [44]. To limit the potential effect of datasets, we used two different datasets; seven participants explored a movie dataset and the other seven explored a Korean SAT dataset. After the 10-minute exploration, we asked them to briefly explain the visualizations they created to check if they created the visualizations as they intended.

We used four datasets: a weather dataset (2,922 rows and 8 fields) [45] for tutorial videos, quizzes after each video, and exercise recipes; for Task 1, a birdstrike dataset (9,987 rows and 15 fields) [45]; for Task 2, a movie dataset (10,029 rows and 12 fields) [46] and a Korean SAT dataset (14,098 rows and 12 fields). Both the movie and SAT datasets had six *C* fields and six *Q* fields. For all the datasets, we randomly shuffled the order of rows to limit the effect of potential bias.

Latency Condition. A body of research exists on how the latency of interactive systems can affect user behavior [2], [47]. To control the latency of ProReveal, we simulated the latency of each response by drawing a random number from a normal distribution with a mean of 3,000 ms and a standard deviation of 1,000 ms. The mean and standard deviation of latency were based on a benchmark of a scalable visualization system [19] and were longer than those used in previous studies (e.g., 600 ms and 1,200 ms [6], and 500 ms and 2,500 ms on average [8]). The first response of a visualization was provided faster (i.e., in 300 ms), which was also feasible in practice [19]. Each response covered 1% of data, so it took five minutes on average (i.e., 3,000 ms $\times$ 100 responses) to finish a visualization if it was the only one being computed in the system. In the experiment, progressive computation was done on a web browser, instead of a backend, to control the latency and avoid unexpected delay due to computation on distributed nodes.

Apparatus. Participants were seated in front of a desktop with two 24-inch monitors at a resolution of 1920 $\times$ 1080. The ProReveal interface was shown on the left monitor (hereafter, Interface), while a web app for the user study was presented on the right monitor (hereafter, Presenter). Presenter managed experimental sessions, such as playing

tutorial videos, presenting exploration recipes one by one, and receiving answers from the participants. Presenter remotely controlled Interface; the participants could begin exploring the given data on Interface with the data loaded when they were ready. Both Interface and Presenter logged all important interactions for analysis.

Procedure. After signing a consent form, participants had a tutorial session during which they watched four videos played on Presenter: one introductory video about progressive visual analytics, two videos about ProReveal and uncertain visualizations, and the last one about Guards. After watching each video, they took quick quizzes and practiced what they learned from the video on Interface. The entire tutorial took approximately 35 minutes. Then, the participants tried five exercise recipes (i.e., R1-R5), displayed one by one on Presenter. Since it took about five minutes for one visualization to finish, they were not able to see a complete result during the session. After finishing the exercise recipes, participants carried out Task 1 (i.e., R6-R10) without any support from the experimenter. The order of the recipes in Task 1 was randomized.

After an optional three-minute break, the participants were introduced to Task 2. The interface was configured to initially show one univariate visualization for each field in a dataset to provide a starting point for exploration. After the 10-minute exploration, participants reviewed their visualizations one by one with the experimenter. The experimenter transcribed 1) why they created each visualization and 2) what they found from the visualization. After completing both tasks, participants responded to an SUS (System Usability Scale) questionnaire [48] and described their experience with ProReveal. The entire session took about 80 minutes.

5.2 Results

We report the results of our user study in three parts: the accuracy and time in Task 1, the number of insights found in Task 2, and qualitative feedback from the participants.

Task 1. Out of 70 (5 recipes $\times$ 14 participants), participants chose correct answers except for only one case where the participant hastily chose one of two competing cells, overlooking their uncertainty. We also checked whether the participants correctly created Guards as in the recipes. Due to the limitation of drag and drop interactions in accuracy, we asked the participants to set a constant as closely as possible to a specific value or a certain range for `Value` and `Rank` Guards, and we permitted 5% margin for the constants of those Guards. Participants correctly created Guards in all 70 recipes: they spent on average 5.71 ($\sigma = 2.61$) seconds on creating the first visualization, 22.56 ($\sigma = 11.37$) seconds on creating a Guard, 16.75 ($\sigma = 5.67$) seconds on filtering and making the second visualization, and 85.43 ($\sigma = 30.75$) seconds on answering the question of recipes.

Seven participants voluntarily created additional Guards (26 out of 70 recipes) to confirm their answers even though they were not asked to do so. They mostly (in 20 recipes) created `Comparative` Guards to choose one between the top two, since we asked them to choose the answer with the most data items. Other Guards additionally used in Task 1 were `Rank` (8 recipes), `Value` (1), and `Power Law` (1).

Task 2. On average, the participants created 7.64 ($\sigma = 2.21$) visualizations except the initial univariate visualizations given by default. From those visualizations, they found 4.77 ($\sigma = 1.79$) insights by leaving 3.29 ($\sigma = 2.40$) Guards on the visualizations. They used the `Linear` Guard the most (22 times), followed by `Rank` (11 times), `Comparative` (9), `Range` (2), and `Normal` (2). `Value` and `Power Law` Guards were not used in Task 2. We did not find any significant difference between the datasets (i.e., movie and SAT) on the number of visualizations created, insights reported, and Guards created ($ps > .05$, ns).

Subjective Feedback and Interview. ProReveal received an average SUS score of 78.39 ($\sigma = 10.99$), which lies in between "Good" and "Excellent" adjective ratings [49]. Through an interview, we surveyed major strategies our participants developed to decide a correct answer. Observing a clear gap between confidence intervals was the most frequently used one (8 participants), and other responses were waiting until a specific amount of data was processed (7), using Guards (6), waiting as long as possible (2), and checking the stability of visualization over time (2).

In the interview, two participants also suggested new types of PVA-Guards that would be helpful for their exploration. P5 suggested it would be useful if he could leave a Guard on multiple visual elements at once (e.g., three bars), which calls for a new type of variable, that is, a *group* variable. With `Rank` Guards, such a group variable can be used to safeguard the knowledge that a group of categories are on the top (i.e., in the top 3). In addition, P7 suggested a Guard that tests the significance of linear regression would be helpful in his daily analysis, thus complementing `Linear` Guards.

6 Discussion

Our study results suggest that participants could understand the concept of Guards and incorporate Guards in their data exploration through the ProReveal interface. Participants could follow all visualization recipes and choose correct answers except for only one case after about 30-minute training, and seven participants voluntarily used Guards in Task 1. In addition, as the average SUS score suggests, they positively rated the ProReveal interface.

In this section, based on our results and observations as well as the participants' feedback, we reflect on how participants used Guards in progressive data exploration. We then discuss the limitations of our lab study and future research directions.

Benefits of PVA-Guards and ProReveal. In Task 1, six participants reported that using Guards was their major strategy to deal with uncertainty. P8 stated, "*The major benefit of Guards was the feedback on my intermediate findings from progressive visualization.*" In addition, P11 noted, "*Guards helped me to build trust on my hypotheses and proceed to subsequent analysis,*" which advocates the benefit of PVA-Guards.

The PVA-Guard list of ProReveal served as an overview of uncertain intermediate knowledge, which seems to be helpful in recalling the context of safeguarded knowledge. For example, P4 said "*I could be aware of the overall progress, because [the PVA-Guard list] persistently presents my Guards. I left Guards on interesting but uncertain knowledge, so that they*

are visible on the screen, and I continued to explore data." In addition, P11 stated, "*I can resume analysis on the visualizations (on which the Guards are left), because I captured them as Guards, which, I believe, can speed up the analysis,*" indicating she used Guards to recall the context of previous findings and resume the analysis.

Diversity in Uncertainty Interpretation. Our study revealed that participants employed different strategies to interpret the uncertainty of a visualization. Eight (out of 14) participants checked whether there was a clear gap in confidence intervals between two competing bars, but a "clear" gap is still arbitrary and subject to bias. Waiting for a specific amount of data being processed (seven out of 14) was the second most frequently used strategy, but it does not guarantee right decisions and can also misguide decisions. The amount of data participants thought enough to make decisions varied (e.g., 20%, 25%, or 33%). Considering the potential threats resulting from such heterogeneity in uncertainty interpretation, we believe PVA-Guards can be a systematic means for preventing people from making incorrect, hasty decisions and validating their correctness even after wrong decisions are made.

Unexpected Use of the PVA-Guard Panel. We observed that participants sometimes used the PVA-Guard panel (Figure 1d) to bolster confidence on their hypotheses by just checking uncertainty measures (e.g., p values) from a preview of a Guard without actually creating it. Participants opened the PVA-Guard panel (by choosing a Guard type on the PVA-Guard menu) 4.93 times ($\sigma = 3.05$) on average in Task 2, but only 3.29 ($\sigma = 2.40$) Guards were actually created. This means that the participants closed the panel without creating a Guard in approximately 33% of cases. P3 said, "*I used the panel just to check if the popularity field is linearly related to the score field, since I wanted to know how popularity is calculated,*" which indicates that he appropriated the PVA-Guard panel to understand the data.

Concerns on the Use of p Values. We used p values as validity measures, but this should be done with care in future designs. The benefits of p values would be their interpretability and familiarity, but we found that they can give an illusion of certainty. P2 stated, "*I was surprised that p values change more than I expected. I am not sure I can absolutely trust p values.*" Similarly, P6 noted, "*At first, I tried to choose an answer as fast as possible, but after I saw fluctuation in p values, I became more cautious.*" Although the interface would be more complicated, providing control over multiple hypothesis testing can alleviate this problem [7].

Limitations and Future Work. In our study, participants explored data that they have not seen before, although we used datasets that they were familiar with (i.e., movie and SAT). This might be a reason why participants preferred Guards with relative values (i.e., `Linear`, `Rank`, and `Comparative`) in Task 2 to Guards with absolute values (i.e., `Value` and `Range`) that require deeper understanding of the data domain.

We aimed to evaluate the usability of ProReveal in a lab study. Therefore, our latency conditions such as the latency of a visualization (i.e., 3 seconds on average) and the time taken to complete a visualization (i.e., 5 minutes on average) were shorter than ones common in practice. We are interested in deploying our system and investigating how Guards can be used to validate the conclusions from the exploration of real-world data where visualizations take a few hours or a day to finish. In this case, a session can span a few days, so it would be important to allow people to recall and continue the previous analysis where PVA-Guards can be possibly used. Furthermore, it would be also interesting to explore a different combination of design choices from ones we made. For example, we may want to employ more active approaches (e.g., alert an analyst) for the exploration of large-scale data since one can be offline when a violation happens due to the length of the session.

Guards can be logically combined to express higher-level knowledge. However, since the ten low-level tasks [28] mainly focus on analytic activities on multidimensional tabular datasets, our Guards are not complete enough to represent general knowledge gained from data exploration. However, if we have a solid task taxonomy for a certain type of datasets, for example, a task taxonomy for graph visualization [50], we can apply a similar approach to ours to build a new set of Guards. Finally, we designed the interactions for Guards on two visualization types (i.e., gradient plots and heatmaps). We chose the two visualizations because they can visualize univariate or bivariate results with a proven capability of showing uncertainty. In the future, it would be interesting to extend Guards to a wider range of visualizations such as progressive parallel coordinates [51].

7 Conclusion

Despite the benefits of progressive visual analytics (PVA), managing the trustworthiness of intermediate outcomes has been regarded as a core concern when applying PVA to a wider range of scenarios. To tackle this problem, we present a novel concept of Progressive Visual Analytics with Safeguards and a proof-of-concept system, ProReveal, which incorporates seven types of PVA-Guards. ProReveal allows people to present their uncertain intermediate knowledge as PVA-Guards on visualizations during progressive data exploration. Then, ProReveal validates PVA-Guards online, providing their validity measures as an estimate of uncertainty. The results of our user study were promising; we found that people can utilize this concept to gain trust in the intermediate knowledge and steer their exploration. We also found a potential benefit of PVA-Guards—a means of achieving consistency in progressive data exploration to alleviate the heterogeneity in uncertainty interpretation. We believe our concept can be extended to a broader range of tasks, datasets, and visualizations in the near future, serving as an effective means of achieving trustworthiness in PVA.

Acknowledgments

This work has supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C2089062). The authors would like to thank the reviewers of our initial submission to InfoVis 2019 for their feedback and suggestions, which helped us improve our paper.

REFERENCES

[1] C. D. Stolper, A. Perer, and D. Gotz, "Progressive visual analytics: User-driven visual exploration of in-progress analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1653–1662, Dec 2014.

[2] J. Nielsen, "Response times: The 3 important limits," https://www.nngroup.com/articles/response-times-3-important-limits/, Mar. 2016.

[3] R. B. Miller, "Response time in man-computer conversational transactions," in *Proc. of the Fall Joint Computer Conference, Part I*. ACM, 1968, pp. 267–277.

[4] B. Shneiderman, "Response time and display rate in human performance with computers," *ACM Computing Surveys*, vol. 16, no. 3, pp. 265–285, Sep. 1984.

[5] D. Fisher, I. Popov, S. Drucker, and M. Schraefel, "Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster," in *Proceedings of the 2012 CHI Conference on Human Factors in Computing Systems*, 2012, pp. 1673–1682.

[6] E. Zgraggen, A. Galakatos, A. Crotty, J.-D. Fekete, and T. Kraska, "How progressive visualizations affect exploratory analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 8, pp. 1977–1987, 2017.

[7] Z. Zhao, L. De Stefani, E. Zgraggen, C. Binnig, E. Upfal, and T. Kraska, "Controlling false discoveries during interactive data exploration," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: ACM, 2017, pp. 527–540. [Online]. Available: http://doi.acm.org/10.1145/3035918.3064019

[8] Y. Wu, L. Xu, R. Chang, J. M. Hellerstein, and E. Wu, "Making sense of asynchrony in interactive data visualizations," *CoRR*, vol. abs/1806.01499, 2018. [Online]. Available: http://arxiv.org/abs/1806.01499

[9] D. Moritz, D. Fisher, B. Ding, and C. Wang, "Trust, but verify: Optimistic visualizations of approximate queries for exploring big data," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: ACM, 2017, pp. 2904–2915. [Online]. Available: http://doi.acm.org/10.1145/3025453.3025456

[10] J.-D. Fekete and R. Primet, "Progressive analytics: A computation paradigm for exploratory data analysis," *ArXiv e-prints*, July 2016. [Online]. Available: http://arxiv.org/abs/1607.05162

[11] J. M. Hellerstein, P. J. Haas, and H. J. Wang, "Online aggregation," in *Proc. of the 1997 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '97. ACM, 1997, pp. 171–182.

[12] H.-J. Schulz, M. Angelini, G. Santucci, and H. Schumann, "An enhanced visualization process model for incremental visualization," *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 7, pp. 1830–1842, July 2016.

[13] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit, "Opening the black box: Strategies for increased user involvement in existing algorithm implementations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1643–1652, Dec 2014.

[14] J.-D. Fekete, D. Fisher, A. Nandi, and M. Sedlmair, "Progressive Data Analysis and Visualization (Dagstuhl Seminar 18411)," *Dagstuhl Reports*, vol. 8, no. 10, pp. 1–40, 2019. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2019/10346

[15] M. Angelini, G. Santucci, H. Schumann, and H.-J. Schulz, "A review and characterization of progressive visual analytics," *Informatics*, vol. 5, p. 31, 2018.

[16] L. Micallef, H.-J. Schulz, M. Angelini, M. Aupetit, R. Chang, J. Kohlhammer, A. Perer, and G. Santucci, "The Human User in Progressive Visual Analytics," in *EuroVis 2019 - Short Papers*, J. Johansson, F. Sadlo, and G. E. Marai, Eds. The Eurographics Association, 2019.

[17] M. Angelini, T. May, G. Santucci, and H.-J. Schulz, "On Quality Indicators for Progressive Visual Analytics," in *EuroVis Workshop on Visual Analytics (EuroVA)*, T. v. Landesberger and C. Turkay, Eds. The Eurographics Association, 2019.

[18] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. USENIX Association, 2010, pp. 10–10.

[19] J. Jo, W. Kim, S. Yoo, B. Kim, and J. Seo, "Swifttuna: Responsive and incremental visual exploration of large-scale multidimensional data," in *2017 IEEE Pacific Visualization Symposium (PacificVis)*, April 2017, pp. 131–140.

[20] N. Pezzotti, T. Hllt, J. Van Gemert, B. Lelieveldt, E. Eisemann, and A. Vilanova, "Deepeyes: Progressive visual analytics for designing deep neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, 01 2018.

[21] F. Sperrle, J. Bernard, M. Sedlmair, D. Keim, and M. El-Assady, "Speculative execution for guided visual analytics," *arXiv preprint arXiv:1908.02627*, 2019.

[22] S. K. Badam, N. Elmqvist, and J.-D. Fekete, "Steering the craft: Ui elements and visualizations for supporting progressive visual analytics," *Comput. Graph. Forum*, vol. 36, no. 3, pp. 491–502, Jun. 2017. [Online]. Available: https://doi.org/10.1111/cgf.13205

[23] N. Pezzotti, B. P. F. Lelieveldt, L. van der Maaten, T. Höllt, E. Eisemann, and A. Vilanova, "Approximated and user steerable tsne for progressive visual analytics," *CoRR*, vol. abs/1512.01655, 2015.

[24] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[25] J. Jo, J. Seo, and J. Fekete, "Panene: A progressive algorithm for indexing and querying approximate k-nearest neighbors," *IEEE Transactions on Visualization and Computer Graphics*, p. preprint, 2018.

[26] C. Turkay, E. Kaya, S. Balcisoy, and H. Hauser, "Designing Progressive and Interactive Analytics Processes for High-Dimensional Data Analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 131–140, Jan 2017.

[27] S. Rahman, M. Aliakbarpour, H. K. Kong, E. Blais, K. Karahalios, A. Parameswaran, and R. Rubinfield, "I've seen "enough": Incrementally improving visualizations to support rapid decision making," *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1262–1273, Aug. 2017. [Online]. Available: https://doi.org/10.14778/3137628.3137637

[28] R. Amar, J. Eagan, and J. Stasko, "Low-level components of analytic activity in information visualization," in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, Oct 2005, pp. 111–117.

[29] N. Smirnov *et al.*, "Table for estimating the goodness of fit of empirical distributions," *Annals of Mathematical Statistics*, vol. 19, no. 2, pp. 279–281, 1948.

[30] Y.-M. Jang, R. Mallipeddi, S. Lee, H.-W. Kwak, and M. Lee, "Human intention recognition based on eyeball movement pattern and pupil size variation," *Neurocomputing*, vol. 128, pp. 421 – 432, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231213008795

[31] A. Endert, P. Fiaux, and C. North, "Semantic interaction for visual text analytics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 473–482. [Online]. Available: http://doi.acm.org/10.1145/2207676.2207741

[32] K. Abdullah, C. P. Lee, G. J. Conti, J. A. Copeland, and J. T. Stasko, "Ids rainstorm: Visualizing ids alarms." in *VizSEC*. Citeseer, 2005, p. 1.

[33] M. Correll, D. Moritz, and J. Heer, "Value-suppressing uncertainty palettes," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: ACM, 2018, pp. 642:1–642:11. [Online]. Available: http://doi.acm.org/10.1145/3173574.3174216

[34] M. Correll and M. Gleicher, "Error bars considered harmful: Exploring alternate encodings for mean and error," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2142–2151, Dec 2014.

[35] "mpl colormaps," https://bids.github.io/colormap/, accessed: 2019-03-31.

[36] T. Munzner, *Visualization analysis and design*. AK Peters/CRC Press, 2014.

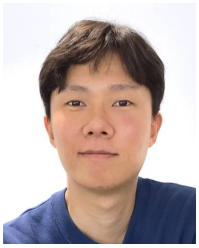
[37] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, Oct. 2016. [Online]. Available: http://doi.acm.org/10.1145/2934664

[38] T. Prusti, J. De Bruijne, A. G. Brown, A. Vallenari, C. Babusiaux, C. Bailer-Jones, U. Bastian, M. Biermann, D. W. Evans, L. Eyer *et al.*, "The gaia mission," *Astronomy & Astrophysics*, vol. 595, p. A1, 2016.

[39] A. Brown, A. Vallenari, T. Prusti, J. De Bruijne, C. Babusiaux, C. Bailer-Jones, M. Biermann, D. W. Evans, L. Eyer, F. Jansen

et al., "Gaia data release 2-summary of the contents and survey properties," *Astronomy & astrophysics*, vol. 616, p. A1, 2018.
[40] "Typescript: Javascript that scales." https://www.typescriptlang.org/, accessed: 2019-03-31.
[41] M. Bostock, V. Ogievetsky, and J. Heer, "D3 Data-Driven Documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, Dec. 2011.
[42] "Angular," https://angular.io/, accessed: 2019-03-31.
[43] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer, "Voyager: Exploratory analysis via faceted browsing of visualization recommendations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 649–658, Jan 2016.
[44] J. Jo, S. L'Yi, B. Lee, and J. Seo, "Touchpivot: Blending wimp & post-wimp interfaces for data exploration on tablet devices," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: ACM, 2017, pp. 2660–2671. [Online]. Available: http://doi.acm.org/10.1145/3025453.3025752
[45] "Vega datasets," https://github.com/vega/vega-datasets, accessed: 2019-03-31.
[46] "The movies dataset," https://www.kaggle.com/rounakbanik/the-movies-dataset#movies_metadata.csv, accessed: 2019-03-31.
[47] J. Nielsen, *Usability engineering*. Elsevier, 1994.
[48] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.
[49] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Intl. Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.
[50] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry, "Task taxonomy for graph visualization," in *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*. ACM, 2006, pp. 1–5.
[51] R. Rosenbaum, J. Zhi, and B. Hamann, "Progressive parallel coordinates," in *2012 IEEE Pacific Visualization Symposium*, Feb 2012, pp. 25–32.

Jaemin Jo is a PhD student at Human-Computer Interaction Laboratory of Seoul National University, Korea. His research interests include Human-Computer Interaction and large-scale data visualization. He is especially interested in progressive visualization systems that facilitate the responsive exploration of large-scale data. He received a B.S. degree in computer science and engineering from Seoul National University in 2014.



Sehi L'Yi is a PhD student at Human-Computer Interaction Laboratory of Seoul National University, Korea. His research interests include Human-Computer Interaction and Information Visualization. He is currently focusing on developing systems and techniques for visualization recommendation to lower the barrier for non-experts to explore their data. He received a B.S. degree in computer science and engineering from Chungbuk National University in 2013.

Bongshin Lee is a sr. principal researcher at Microsoft Research. Her research interests include Human-Computer Interaction, Information Visualization, and User Interfaces and Interaction Techniques. She is currently focusing on developing innovative ways for people to create visualizations, interact with their data, and share data-driven stories visually. She received her PhD in Computer Science from the University of Maryland at College Park in 2006.

Jinwook Seo is a professor in the Department of Computer Science and Engineering, Seoul National University, where he is also the Director of the Human-Computer Interaction Laboratory. His research interests include Human-Computer Interaction, Information Visualization, and Biomedical Informatics. He received his PhD in Computer Science from the University of Maryland at College Park in 2005.