

Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics

Charles D. Stolper, Adam Perer, and David Gotz

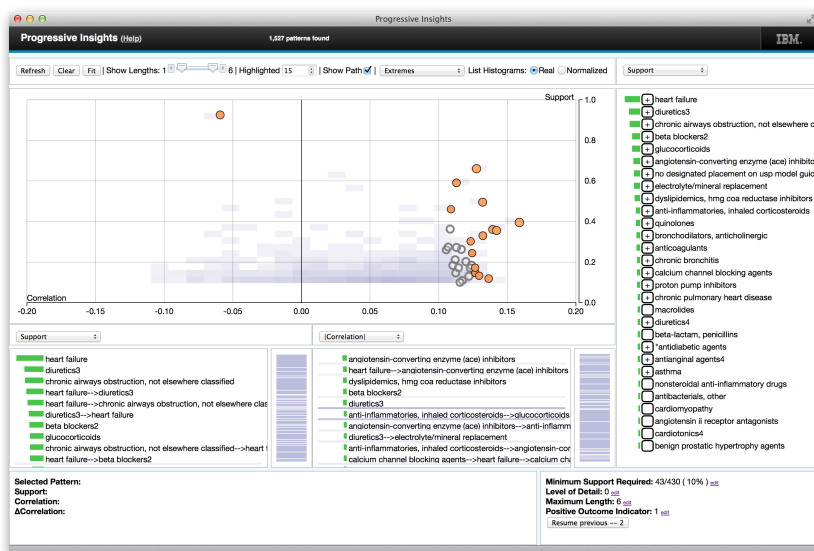


Fig. 1. The Progressive Insights system features progressive visual analytics, and supports user-driven exploration of in-progress analytics. Partial results from the progressive analytics enhance the scatterplot, list, and tree visualizations without interfering with users' cognitive workflow.

Abstract— As datasets grow and analytic algorithms become more complex, the typical workflow of analysts launching an analytic, waiting for it to complete, inspecting the results, and then re-launching the computation with adjusted parameters is not realistic for many real-world tasks. This paper presents an alternative workflow, *progressive visual analytics*, which enables an analyst to inspect partial results of an algorithm as they become available and interact with the algorithm to prioritize subspaces of interest. Progressive visual analytics depends on adapting analytical algorithms to produce meaningful partial results and enable analyst intervention without sacrificing computational speed. The paradigm also depends on adapting information visualization techniques to incorporate the constantly refining results without overwhelming analysts and provide interactions to support an analyst directing the analytic. The contributions of this paper include: a description of the progressive visual analytics paradigm; design goals for both the algorithms and visualizations in progressive visual analytics systems; an example progressive visual analytics system (Progressive Insights) for analyzing common patterns in a collection of event sequences; and an evaluation of Progressive Insights and the progressive visual analytics paradigm by clinical researchers analyzing electronic medical records.

Index Terms—Progressive visual analytics, information visualization, interactive machine learning, electronic medical records.

1 INTRODUCTION

The goal of visual analytics is to help users perform complex data exploration, analysis, and reasoning tasks. These systems typically work by combining the power of automated statistical computations with interactive visual interfaces, which can support analysts as they develop, test, and validate hypotheses from data.

- Charles D. Stolper is with the School of Interactive Computing, Georgia Institute of Technology. E-mail: chadstolper@gatech.edu.
- Adam Perer is with IBM T.J. Watson Research Center. E-mail: adam.perer@us.ibm.com.
- David Gotz is with the University of North Carolina at Chapel Hill. E-mail: gotz@unc.edu.

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014; date of publication xx xxx 2014; date of current version xx xxx 2014.

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

Interactive performance is at the core of these emerging visual analytics methods, allowing users to dynamically manipulate data via flexible exploration. However, for many applications this interactive capability is being threatened by two important trends: (1) a surge in “big data,” and (2) the development of more sophisticated and computationally expensive algorithms. Both of these trends can contribute to slower performance rates for automated analytics. As rates grow slower, delays in analytic processing can produce “speed bumps” in users’ exploratory workflows: periods of time between visual interactions where users are forced to wait for automated analytics algorithms to finish before continuing.

A range of techniques can be applied to overcome this challenge. One common approach is to *pre-compute* analytics on a large data set before allowing interactive visualization. This avoids delays due to analytic processing, but limits creative exploration by preventing new rounds of analytics with input data or the adjustment of parameters due to data-driven insights. An alternative approach is to *accelerate* analytic algorithms through massively parallel processing (e.g., MapRe-

duce [7, 8]). However, such methods can require significant hardware resources to accelerate long-running analytics over large data sets (which would normally require hours or perhaps days to complete) to interactive rates.

In this paper, we explore an alternative approach: *progressive visual analytics*. This method is based on the idea that analytic algorithms can be designed to produce semantically meaningful partial results during execution. These progressive results can then be integrated into interactive visualizations that allow users to immediately explore partial results, examine new results as soon as they are computed, and perform new rounds of exploratory analytics without waiting for previous analyses to complete. This approach is a sharp contrast to pre-computation and acceleration, both of which assume that analytics must run to completion before results are available for visual interaction. This approach also differs from techniques that deal with streaming data: in streaming data, new raw data arrives, while in progressive visual analytics, the raw data remains constant but new results derived from analytics on the data arrive.

A progressive approach promises many advantages for visual analytics. For example, the time that had been spent waiting for analytics to complete can now be spent evaluating early partial results. Based on those partial results, users can adjust input parameters, change data sets, or simply terminate mis-directed analytics that are producing unimportant results. In summary, progressive visual analytics promises to speed users' analyses by eliminating the periods of time between user interactions and the execution of computational analytics.

This vision, however, relies upon two requirements: (1) analytics must be designed to produce meaningful partial results during execution, and (2) the visual interface must be designed to properly manage the partial results as they are produced without interfering with the user's cognitive workflow. The research contributions we outline in this paper focus on these two issues. In particular, we describe:

- Design goals for supporting *progressive visual analytics*, a novel concept for user-driven visual exploration of in-progress analytics;
- An implementation of a progressive visual analytics system, Progressive Insights, for exploring frequent sequences of temporal patterns;
- A case study of domain experts using Progressive Insights to explore frequent patterns in electronic health records.

The remainder of this paper is organized as follows. We begin with a brief review of related work. We then motivate our work with a scenario for progressive visual analytics from the healthcare domain. Next, we outline a set of progressive visual analysis design requirements and describe a prototype system that we designed around these requirements. Finally, we present a case study that demonstrates the benefits of our approach to an analyst working in the medical domain.

2 RELATED WORK

There has recently been a variety of research into handling the challenges of visualization with "big data." Examples include Dix and Ellis's by chance [10] which uses statistical sampling to visualize very large data sets, Rosenbaum et al.'s progressive refinement techniques [29, 30] that visualize large data sets at progressively finer-grained detail, Liu et al.'s imMens [20] that uses similar techniques to supports visualization and interaction of binned aggregate data, and Lins et al.'s nanocubes [19] which provides infrastructure support for approximate data cubes. However, while these techniques focus on making the visualizations perform at interactive speeds, there has been relatively little work on making the analytics interactive with large scale data.

The most relevant prior research on progressive visual analytics is the work by Angelini and Santucci [2] and Fisher et al. on incremental visualization [14]. Angelini and Santucci motivated this research by

describing how computation time might necessitate progressive visualization and model the errors that stem from doing so. Fisher et al. built a system, sampleAction, that supports an analyst exploring big data through statistical summaries of massive database tables by incrementally taking larger and larger samples from the table. For each sample, the authors use methods to determine statistical error bounds of extrapolating the sampled data to the full data and present this to the user. As each larger sample is processed, the view is updated to incorporate the more precise information. Our work builds on this research by generalizing past large database queries to data mining algorithms as well as providing the analyst with the ability to direct the course of the running analytic.

Progressive visual analytics is inherently dynamic, and there have been a number of commercial and research approaches to handling dynamics in visualization. Hetzler et al. [16] present design criteria for dynamic visualization that our design requirements for progressive visual analytics extend. Recent approaches, such as Tableau [35], Spotfire [37], Google Charts [15], Cubism.js [34], and Huron et al.'s Visual Sedimentation [17], update views at fixed intervals to visualize the most recent data. Analytic-backed dashboards, for example, Krstajic et al.'s Cloudlines [18] follow this same pattern of updating at fixed intervals, but rather than visualizing raw data, these systems visualize the results of an analytic on the dynamic data. Some work balances stability by giving users control of updates, for example, Hetzler et al. and Federico et al. [13] both use a slider to define how often updates occur.

In Section 5, we demonstrate an application of progressive visual analytics on data exploration of temporal patterns. There have been many previous systems designed to support this common task, and Aigner et al. provides a survey of time-series visualization techniques [1]. Within this larger field, researchers have more specifically studied using visualization to identify patterns in large event series collections. Systems such as LifeLines [27, 28], LifeLines2 [39, 40], PatternFinder [12], DataJewel [3], LifeFlow [43], OutFlow [42], and EventFlow [22, 23] all visualize event series in order to highlight common patterns so that they can be recognized and utilized by analysts. Wong et al. [41] have built a system that makes use of a similar analytic to the SPAM algorithm [4] that we adapt into a progressive analytic. Perer and Wang [25] introduced Frequence, a system that also integrates data mining and visualization for finding frequent patterns using an adaptation of SPAM to handle multiple levels-of-detail, temporal context, concurrency, and outcome analysis. In all of these systems, however, the event series data is pre-processed and pre-computed before visualization, and therefore does not satisfy the criteria or reap the benefits of progressive visual analytics.

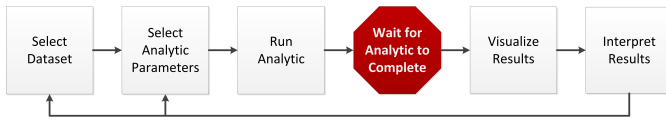
3 MOTIVATION

The challenges that motivate progressive visual analytics—extremely large data sets and complex, long-running analytics—are common to a wide variety of domains. In this section, we describe a specific scenario for progressive analytics motivated by challenges faced by some of our research collaborators in the healthcare industry.

Medical institutions, including both hospitals and outpatient facilities, are collecting increasingly detailed information about their patients. Traditional paper folders are being replaced with electronic health record (EHR) systems that combine information about diagnoses, procedures, medications, laboratory test results, and more. These records, which institutions collect for each of their thousands or millions of patients, serve primarily to help doctors capture and review the medical histories of individual patients. However, EHR data is also being used to conduct population-based studies through secondary use applications [31].

One example of a secondary use study is the identification of medical *patterns*—frequently appearing sequences of treatments, diagnoses, etc.—that are associated with unusually positive or negative outcomes [25]. In such applications, a medical analyst would generally apply a pattern mining algorithm to a very large set of EHR data. After supplying a set of input parameters, the algorithm might run for hours or days to produce a list of highly frequent event sequences.

Batch Visual Analytics Workflow



Progressive Visual Analytics Workflow

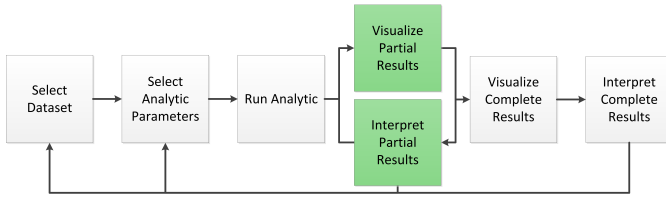


Fig. 2. A comparison between a traditional batch workflow and a workflow utilizing progressive visual analytics. The latter workflow avoids inefficiencies associated with ‘compute-wait-visualize’.

The long processing time is due in part to the complexity of medical data. It is common to find datasets with tens of thousands of distinct types of medical events¹, thousands or even millions of patients, and multiple years of medical data per patient. The long compute time is also due to the complexity of the mining algorithm itself as it traverses the complete search space to find frequent patterns.

Analysts working to understand a dataset must therefore initiate a round of analytics, wait for them to complete, and then visualize the results to understand what patterns exist, how frequently they appear, and how they associate with specific medical outcomes (e.g., mortality, cost, or hospital readmission rate). This process may be repeated many times as each analyst explores new hypotheses. Each change to any data mining parameters, or to the dataset itself (e.g., to focus on a specific set of patients who exhibit a pattern of interest) introduces a huge delay—a speed bump—to the analyst’s investigation.

The above medical scenario is used throughout the remainder of this paper to provide concrete examples of how the concepts we introduce can help support this type of investigation. Moreover, we have applied our prototype implementation, Progressive Insights, to this specific frequent pattern problem in the evaluation we present in Section 6.

4 PROGRESSIVE VISUAL ANALYTICS

Thomas and Cook define *visual analytics* as “the science of analytical reasoning facilitated by interactive visual interfaces” [36]. A common implementation of such systems, which we refer to as *batch visual analytics*, is when the core components (computational analysis and interactive visualization) are designed to operate in series. This may lead to an inefficient workflow of ‘compute-wait-visualize’, where computation is often performed first and then results are visualized once the computation has completed. Consider a typical batch visual analytics workflow such as the one illustrated at the top of Figure 2. Analysts cannot proceed with visual analysis while the analytic is running. As a result, each time the analysts inspect the visualized results and decide to filter the dataset or modify parameters, they hit a speed bump and must wait again for the analytic to fully complete execution.

The focus of our research is on the development of methods to avoid such speed bumps, where analysts can begin making decisions immediately after starting the analytic, as illustrated at the bottom of Figure 2. Visualizing partial results can provide early, meaningful clues that allow analysts to more quickly begin the next phase of exploration and revise their hypotheses, without waiting for long-running analytics to terminate. Analysts can also use partial results to quickly detect

¹Even ignoring procedures, medications, and lab tests, there are well over 10,000 distinct diagnosis codes in ICD-9-CM, a diagnosis coding system published by the United States Centers for Medicare and Medicaid Services (CMS). ICD-9-CM is widely used within the United States healthcare system.

and abort analytic processes that are unlikely to produce meaningful results, avoiding futile, time consuming executions. As the size of data and complexity of analytics grows, these timely and informed decisions can reduce both wasted analyst time and wasted computational resources. This strategy, which we refer to as *progressive visual analytics*, is in sharp contrast to the batch visual analytics paradigm of ‘compute-wait-visualize’.

However, in order for systems to provide progressive visual analytics capabilities, the analytics and visualization components that make up the system must be designed in specific ways. In the following sections, we describe the key design requirements and enumerate a set of concrete design goals.

4.1 Progressive and User-Driven Analytics

Progressive analytics are statistical algorithms that produce semantically meaningful partial results that improve in quality over time. While many algorithms produce partial results during execution, only a subset produce meaningful partial results that are reflective of an overall dataset and appropriate for inspection and interpretation by analysts. In other words, the partial results of progressive algorithms are of the same format as the final results of those algorithms. For example, both K-means [21] and EM [9] clustering algorithms are examples of progressive analytics, as their intermediate results are interpretable estimates of the final result. In the case of K-means, each iteration of the algorithm assigns each data point to a cluster. In the case of EM, after each M-step, each data point has an expected probability distribution over possible clusters. Analysts may inspect the current state of these algorithms at any time, aware that the clusters or distributions may change as the analytic progresses.

User-driven analytics are statistical algorithms that can adapt to input by analysts. An example user-driven analytic would be an EM clustering algorithm that allows the analyst to pre-set class weights for data points. As opposed to non-user-driven analytics, this class of algorithms enables analysts to incorporate their domain knowledge or data-driven insights into the analysis.

In some cases, existing analytics can be adapted to be progressive and/or user-driven. For example, the K-means algorithm can be adapted to store partial results in a database and to allow an analyst to manually classify certain points.

4.2 Progressive and Interactive Visualizations

Just as progressive analytics produce more complete results over time, a *progressive visualization* is able to incorporate partial results into the visual display as they are produced by an analytic algorithm. However, as changes of the retinal properties of visual elements (such as position, size, and color) often distract users, a design conflict exists between (a) showing the most up-to-date information and (b) keeping analysts from being distracted by constant updates.

The challenge of progressive visualization is in balancing the trade-offs between these two imperatives. A compromise suggests that progressive visualizations should inform analysts of new information through subtle, ambient cues. Furthermore, analysts should determine when to incorporate changes using refresh-on-demand interaction to choose when to replace or update the information. This solution minimizes distractions, informs analysts of the current visualization’s accuracy, and provides analysts with control over the analytic process.

Progressive visualizations by their nature act as interfaces from the underlying computational algorithms to the analysts using the system, communicating what the statistical analytics have uncovered. Since interactive analytics are written to adapt to analysts’ input, however, the visualization must also function as an effective interface from the analyst to the computation. A visualization should provide the analyst with apparent affordances for adjusting and directing the analytic, in addition to effectively communicating the results of the analysis to the analyst.

4.3 Design Goals for Progressive Visual Analytics

In summary, we enumerate seven specific design goals for systems that support progressive visual analytics. These goals are an extension of

Hetzler et al.’s [16] design guidelines for dynamic visualization. First, analytics components should be designed to:

1. Provide increasingly meaningful partial results as the algorithm executes
2. Allow users to focus the algorithm to subspaces of interest
3. Allow users to ignore irrelevant subspaces

Visualizations should be designed to:

4. Minimize distractions by not changing views excessively
5. Provide cues to indicate where new results have been found by analytics
6. Support an on-demand refresh when analysts are ready to explore the latest results
7. Provide an interface for users to specify where analytics should focus, as well as the portions of the problem space that should be ignored

When followed, these design goals can help ensure that a system’s components provide the capabilities required to support a progressive visual analysis workflow. To demonstrate the benefits of this approach, the next section describes Progressive Insights, a prototype implementation designed around these goals.

5 PROGRESSIVE INSIGHTS

In order to demonstrate the benefits of progressive visual analytics, we have built a prototype system called Progressive Insights. Progressive Insights is designed to support an analyst searching for common patterns of events as in the motivating healthcare example outlined in Section 3. Progressive Insights provides a user interface with multiple coordinated views (see Figure 1) that integrate closely with an event-pattern mining module. The pattern analysis component employs a modified implementation of Ayers et al.’s SPAM algorithm [4], which we redesigned to meet the progressive and user-driven requirements outlined in the previous section. The interface includes several progressive and interactive visualizations, including two List Views, a Scatterplot View, and a Tree View. Adhering to our design goals for progressive and interactive visualizations, each visualization provides subtle visual cues when new information is available and supports user-driven direction of the underlying pattern mining process. In this section we provide a brief description of our adaptations to the original SPAM algorithm and describe Progressive Insight’s three primary view types. We conclude with a discussion of the overall system, capturing how the views and analytics work together to provide a progressive visual analytic experience.

5.1 The SPAM Algorithm

The Sequential PAttern Mining (SPAM) algorithm, proposed by Ayres et al. [4], is designed to efficiently—in terms of time required to process an entire dataset—identify the most common sequences occurring in a large collection of event series. To briefly demonstrate how the algorithm works, we present a toy example of the algorithm’s execution.

Consider an event series collection with an event library containing four distinct event types: $\{W, X, Y, Z\}$ and four individual event sequences:

- $X \rightarrow Y \rightarrow Z$
- $X \rightarrow Z \rightarrow Y$
- $X \rightarrow Y \rightarrow X \rightarrow X$
- $Z \rightarrow X \rightarrow Y \rightarrow Y$

SPAM is designed to identify subsequences within this data that occur frequently. One key parameter, therefore, is the *minimum support* threshold which specifies the smallest fraction of event series in which a pattern must be found to be reported as “frequent” by the algorithm. For this example, consider a minimum support level of 50%, which

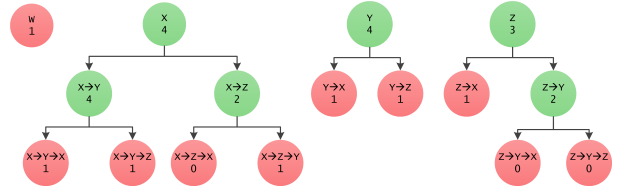


Fig. 3. An example of an execution tree to illustrate how the SPAM algorithm mines patterns. In this example, SPAM was configured to have a minimum support threshold of 2.

would require that a pattern be present in at least two of the four patterns.

SPAM begins with the creation of bitmap representations of each event type in the event library. It uses a highly-scalable vertical bitmap representation that allows for the efficient extension of patterns and checking of support counts, even for very large data sets. We refer the reader to [4] for more details about the bitmap representation and its efficiencies, as well as its support for handling patterns with concurrent events.

Once these bitmaps have been built, the algorithm identifies all of the singleton events that pass the minimum support threshold as an initial set of patterns. In our example, X , Y , and Z all occur at least twice while W only occurs once. Since W itself does not occur often enough to pass the minimum support threshold, it is clear that no patterns involving W can pass the threshold. This means that any pattern involving W can safely be ignored as infrequent.

The algorithm recursively extends each pattern with each of the other frequent singleton patterns until reaching a user-set maximum length. If a pattern’s support is found to be below the support threshold, the pattern is ignored as infrequent and is no longer extended. Effectively, SPAM performs a depth-first traversal over a tree representing all possible frequent patterns, pruning the tree wherever the minimum support is not reached. Thus, it begins by extending the X pattern with Y . Since $X \rightarrow Y$ appears in all four series, the algorithm outputs the pattern as frequent. It then extends $X \rightarrow Y$ with X , forming $X \rightarrow Y \rightarrow X$, which only appears in one series. It is therefore ignored and the algorithm returns to $X \rightarrow Y$. This is extended with Z , forming $X \rightarrow Y \rightarrow Z$, which again appears in only one sequence. This process continues, producing the tree shown in Figure 3.

5.2 Progressive and Interactive SPAM

The SPAM algorithm is natively progressive in that it reports patterns as it finds them. This partially satisfies Design Goal #1. However, the depth-first nature of the algorithm—both as originally proposed and in the publicly available implementations of the algorithm²—means that the partial results reported early in the process are not broadly representative of the overall dataset. In addition, the original SPAM algorithm was not designed to allow user control over how and when it traverses the search space as required by Design Goals #2 and #3.

To better support these design goals, we modified the SPAM algorithm to utilize a breadth-first traversal of patterns. This change provides two significant benefits. First, the revised breadth-first implementation of the algorithm reports patterns from shortest to longest, giving an overview of short but frequent patterns before moving on to find those that are longer and more specific. This change more fully supports Design Goal #1. Second, the queue-based implementation employed during breadth-first traversal can be used to support priority-based steering of the analytic algorithm. For example, driven by user input, specific patterns of interest can be moved to the front of the processing queue. In addition to prioritization, the queue can be used to support analyst-driven pruning. Patterns explicitly marked as irrelevant by a user can be removed from the queue to immediately prune the execution tree. These interactive capabilities directly support Design Goals #2 and #3. Moreover, this example demonstrates

²e.g., the SPMF Framework: <http://www.philippe-fournier-viger.com/spmf/>

Ranking Function	Description
Support	Number of event series containing pattern
Correlation	Pearson correlation of the pattern's presence with an outcome measure
Correlation	Absolute value of Correlation
Δ Correlation	Difference between pattern's prefix's Correlation to the pattern's Correlation
Δ Sibling Correlation	Difference between this pattern's Δ Correlation and the mean of it and its siblings' Δ Correlations
Extremes	The pattern's distance from the origin in the scatterplot visualization. (E.g. the normalized sum of Support and Correlation).

Table 1. A list of the ranking functions implemented in Progressive Insights

that rather minor changes to the design of an analytic algorithm can make dramatic differences with respect to how it supports progressive visual analytics requirements.

5.3 List View

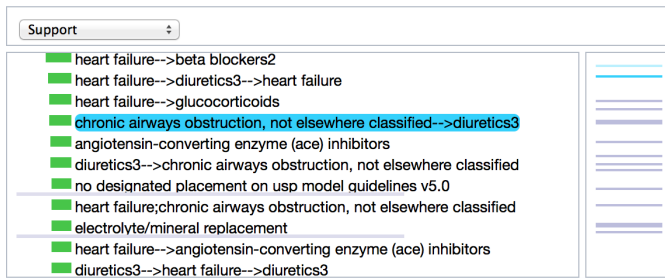


Fig. 4. Progressive Insight's list view consists of a) on the left, the ranked list featuring histograms of the ranking metric, and b) on the right, a mini-map of the complete list. In this example, the selected pattern is highlighted in blue in the list and in the mini-map. The prefixes of the pattern are highlighted in light blue. New patterns from the progressive results are indicated in the list by purple lines, and in the mini-map by purple regions.

The list view is the first of three visualization views included within the Progressive Insights system. Designed following the guidelines outlined in Section 4.3, this view provides a simple but progressive visualization of a dynamic top- n list. The list view is used in Progressive Insights to display the list of patterns detected by the SPAM algorithm sorted by one of several ranking measures. Two instances of the list view are provided in the default configuration of the interface, allowing analysts to compare two different ranking methods simultaneously. By switching between different ranking metrics, the view allows analysts to rapidly browse patterns that the algorithm determines have the highest level of support, correlation, or any of the other ranking functions listed in Table 1.

Figure 4 depicts an instance of a list view. Each list view consists of three sub-views: (a) the ranked list; (b) a histogram paired with the items in the list; and (c) a dynamically updating Seesoft-esque [11] mini-map of the complete pattern list.

The mini-map provides a clickable overview of all patterns identified by the analytics, while the ranked list provides details of each pattern. Each pattern is augmented with a histogram bar whose length is proportional to the magnitude of the selected ranking feature. If the selected ranking is signed (e.g., correlation can be positive or negative), the histogram's color indicates whether the pattern correlates with a positive (green) or negative (red) outcome.

The list view is used to display a continuously changing list of patterns, which grows dynamically as new patterns are detected by the

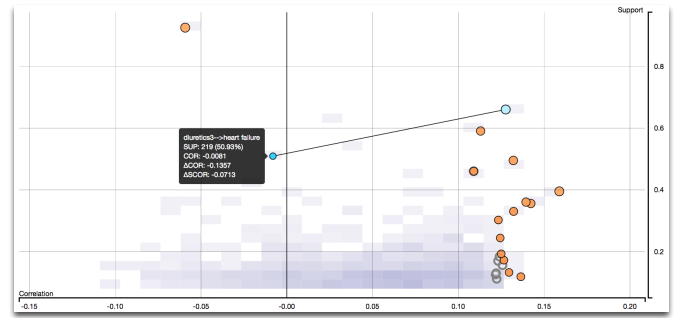


Fig. 6. Progressive Insight's scatterplot view. The top 10 patterns according to the selected metric are shown in orange. All frequent patterns found by the system are represented in aggregate by the purple heatmap. Patterns that were formerly in the top 10 but have since fallen out are shown as empty circles. The selected pattern is highlighted in blue and its prefixes are highlighted in light blue.

progressive SPAM algorithm. One beneficial property of a ranked list view for dynamic data is that the items already in the list will not change positions relative to each other; i.e. if a pattern is ranked higher than another pattern, that pattern will always be ranked higher. However, given the progressive nature of our pattern mining algorithm, new patterns are often discovered that are ranked between previously found patterns. Therefore, a critical challenge for the list view is the process by which these new insertions are added to the display.

In order to minimize movement (Design Goal #4), the list views do not immediately insert discovered patterns into their place in the list. Instead, while the items in the list remain unchanged, the list container provides visual cues as to where in the list new patterns would be inserted if the view were to be fully refreshed. The cues are rendered using purple, horizontal marks that appear between entries in the list. The presence of these marks quickly conveys that the current list is not up-to-date (Design Goal #5), and their locations show where the new results will appear when fully integrated into the view. The color saturation of the marks conveys how many patterns will be inserted at a given position in the list - a darker mark implies the existence of more patterns waiting to be added to the view.

Following Design Goal #6, analysts are given explicit control over when to refresh the views with the latest results. Analysts can use their judgment to determine when, or if, it is most valuable to perform such a refresh. For example, if the top five patterns are already displayed, and a number of lower-ranked patterns have since been found but not yet loaded, the analyst may choose to not refresh the view, avoiding an unproductive yet major visual change.

In addition to the local cues provided by the horizontal marks in the list, the mini-map provides a global picture of where in the list new unloaded patterns are waiting to be visually integrated. In the mini-map, purple regions represent the number and positions of patterns that have not yet been loaded, while white regions show portions of the list for which no new patterns are pending. In combination with the local cues embedded within the list itself, the mini-map design provides analysts with a clear indication regarding where new results have been reported by a progressively running analytic (Design Goal #5).

A final interaction supported by the list view is the ability to select a pattern by clicking on its entry in the list. This action highlights the pattern by changing the color of the pattern in the list to orange. At the same time, all patterns that are the prefix for the selected list item are highlighted in blue. The same highlighting occurs in the mini-map, providing a clear overview of the relative rankings between a selection and related patterns. As discussed in Section 5.6, the system brushes this selection in all other views which provides a coordinated selection capability across the entire user interface.

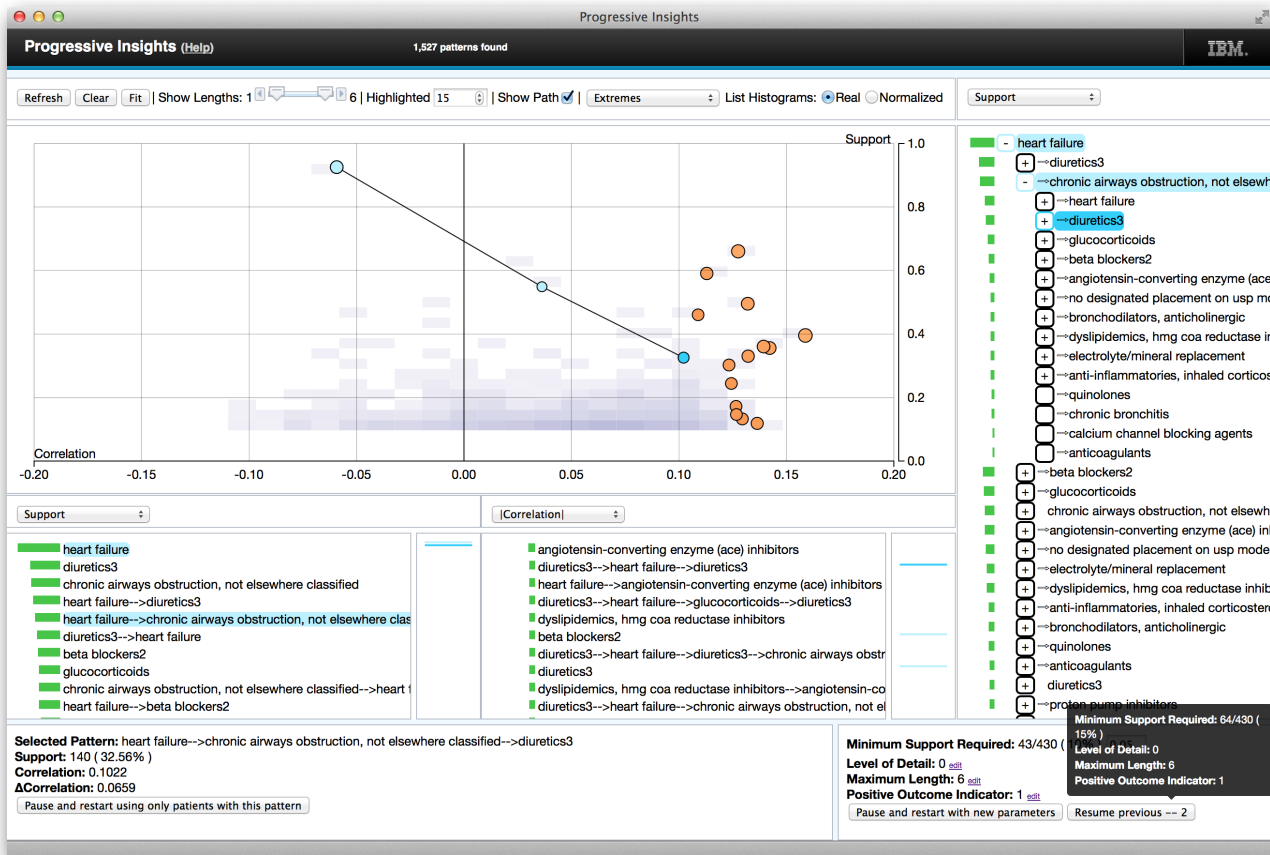


Fig. 5. In Figure 1, the analytics were still in-progress. This figure shows Progressive Insights after the analytics have completed, and after the user has performed a refresh to clear any partial results that are no longer highly ranked.

5.4 Scatterplot View

The second view in the Progressive Insights system is the scatterplot view. Like the list view, the scatterplot view provides analysts with information on the highly ranked patterns. However, the scatterplot provides enhanced interactive capabilities that let an analyst use his or her knowledge to explore the information in a more user-directed fashion.

The scatterplot displays patterns using two metrics at a time, providing analysts with a sense of how a given pair of dimensions relate to each other within the data set. Unlike the list view, which primarily conveys rank, the scatter plot provides analysts with a more precise visualization for comparing the magnitude of the differences in score between patterns.

We represent patterns with orange circles located within the two dimensional scatter plot. The size of the circles can be mapped to a third, user-selected metric. Moreover, to avoid over-plotting and to focus users' visual attention to the most highly ranked events, the system displays only the top n patterns according to the selected metric. The analyst can dynamically select and experiment with various metrics and values of n as part of an interactive exploratory analysis. Analysts can choose from the same list of metrics used in the list view (see Table 1).

Analysts can also choose the two dimensions that control the axes of the scatterplot for the data domain. For example, in the electronic medical records use case that we describe in Section 6, the two selected axes for the scatterplot view are (a) the fraction of patients in which the pattern was found (support) and (b) the correlation that the pattern had with the patient avoiding hospitalization.

Like the list view, selecting a pattern by clicking on its mark in the scatterplot highlights the pattern in orange and the prefixes of the

pattern in blue. However, the scatterplot view also provides a more direct visual representation of this pattern-to-prefix relationship. In particular, an analyst can choose to connect the selected pattern to its parent, grandparent, and all other prefix patterns with a black line. As shown in Figure 5, the line provides a connected path from the selected pattern all the way to the first singleton event that started the pattern. This allows the user to visually see the development of a pattern from the initial event through to the full sequence. Such a capability allows for the identification of patterns that increase or decrease correlation with the positive outcome.

While we have described the basic functionality, a number of more advanced features have been developed to address the unique challenges of progressive analytics. First, the progressive nature of the pattern mining algorithm means that the list of top n patterns—the group of patterns selected for explicit display within the scatterplot—can change dynamically as new patterns are detected.

Similar to the list view, the scatterplot view is designed to defer changes to the visual display even as patterns may be pushed out of the top n by the arrival of new results. This prevents the problem of “disappearing” patterns: the unexpected removal of a pattern that is under active investigation by an analyst. Such an abrupt transition would violate the tenet of avoiding unexpected changes to the display (Design Goal #4).

Therefore, rather than immediately removing a demoted pattern's circle from the view, we subtly transition the fill of the circle to be transparent and shrink its size to a small radius circle that represents metric values below the top n threshold.

Over time, this can result in a proliferation of small clear circles as new results continuously push patterns out of the top n category. Therefore, a *Clear* button is provided to let an analyst remove the out-

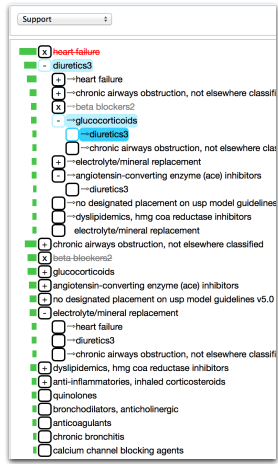


Fig. 7. Progressive Insight’s tree view shows a hierarchical representation of each pattern. Each pattern’s last event has a histogram bar of the ranking metric. User-pruned subtrees have a red strike and user-hidden patterns are grayed out. Selected patterns are highlighted in blue and prefixes are highlighted in light blue.

of-date patterns from the display when he or she is ready, following Design Goal #6.

While showing the top n patterns is important, this may not provide the analyst with an effective overview of the entire set of patterns identified by the analytic. To resolve this deficiency, we augment the scatterplot with a background heatmap showing the density of all identified patterns within the scatterplot’s two dimensional space. The heatmap is rendered using a k by k grid over the viewport of the scatterplot. Zooming and panning the scatterplot does not move the grid. Instead, the heatmap is dynamically recomputed with a new grid as users navigate the scatterplot. This provides dynamically scalable resolution to the heatmap as users explore the scatterplot’s space. Analysts can adjust the granularity of the heatmap (the k that defines the k by k grid) interactively. Each cell is colored using log-scale color mapping indicating the number of known patterns in the corresponding region of the scatterplot. The heatmap is updated at regular intervals using a subtle color change animation. Regions becoming darker (representing denser) act as ambient cues to the analyst that these regions may be worth refreshing and investigating, an application of Design Goal #5.

Users can zoom and pan the view in order to see the heatmap in more or less detail. Users can also hover over a cell of the heatmap to see detailed information about how many patterns are represented by the cell. Moreover, users can explicitly add the corresponding patterns to the view by double-clicking the cell. In this way, the analyst is not limited to viewing the top n patterns of a metric and instead can also view those patterns in regions of interest (Design Goal #2). Finally, as in the list view, pattern selections are linked with other views to support multi-view brushing.

5.5 Tree View

The third view supported by Progressive Insights is the tree view, designed to provide a familiar hierarchical view for users to direct the progressive analytic to subspaces of interest. This view satisfies a critical requirement for effective progressive visual analysis: the ability for the analyst to incorporate their domain knowledge to steer the underlying progressive algorithmic analysis process.

The tree view displays all discovered patterns according to the tree hierarchy of prefix relationships inherent in the patterns. The analyst can drill down into pattern extensions that are semantically interesting. This is in contrast to the metric-driven exploration supported by list and scatterplot views.

Within each level of the tree hierarchy, patterns are sorted according to one of the available ranking metrics in Table 1. As with the list view,

we provide histograms to go along with each pattern, sized and colored according to the value of the selected metric. This sorting allows the analyst to quickly make comparisons between sibling patterns: those patterns that have the same prefix but a different suffix.

Each pattern in the tree is accompanied by an icon (see Figure 7). There are three primary icons used within the tree: +, -, or *empty*. These icons represent, respectively, (a) a pattern with an expanded list of one or more children (i.e. extended patterns that share the corresponding pattern as a common prefix), (b) a pattern with a collapsed list of one or more children, and (c) a pattern with no children that meet the support threshold for the pattern mining analytic.

These symbols are commonly used in representing vertically-aligned trees. However, the progressive nature of the algorithm producing the items in the tree means that there is uncertainty as to whether or not a given pattern has any frequent extensions as children. Moreover, the breadth-first nature of the progressive SPAM algorithm described in Section 5.2 means that users may expand the tree to reach a pattern deep enough in the hierarchy that it has not yet been processed by a long running pattern mining algorithm. As a result, the system does not know if there are extensions that meet the support criteria. Therefore, we introduce a fourth, secondary icon that uses a ? symbol. For these unexplored subspaces, users can click on patterns with this icon to prioritize the corresponding pattern in the mining algorithm’s processing queue. This intuitive mechanism allows users to seamlessly communicate their task requirements directly the underlying analytics. This minimizes lag between analyst actions and the system’s analytic response, an example of Design Goal #2.

Similarly, analysts can also use the tree view to direct the pattern mining analytic to ignore certain branches as per Design Goal #3. First, an analyst can *prune* the execution tree at a certain pattern by right-clicking the node. By doing so, the analyst directs the analytic not to extend the pruned pattern any further, even if common extensions exist. Furthermore, the system recursively marks any otherwise acceptable patterns stemming from the pruned pattern as unacceptable. Second, the analyst may *hide* a pattern by alt-clicking the node. This command marks all patterns that have the hidden pattern anywhere in the pattern as unacceptable. For example, if the analyst hides “ $X \rightarrow Y$ ”, then the system would mark “ $Y \rightarrow X \rightarrow Y$ ” as unacceptable, even though it is in a different branch of the execution tree. Both of these actions can be undone by the user. If a *prune* or *hide* action is undone, each pattern marked as unacceptable as a result of the operation is added to the execution queue to check if it passes the support threshold or has any frequent extensions.

Finally, the analyst can take advantage of the hierarchical structure of patterns to select specific patterns of interest. As in the other views, clicking a pattern highlights the pattern in orange, the pattern’s prefixes in blue, and propagates this selection to brush the corresponding patterns in other views.

5.6 UI and Interaction

As we mentioned at the start of this section, all of the views in Progressive Insights are linked: when a pattern is selected in one view, that selection is propagated and the pattern is selected in every view. In addition to highlighting the selected pattern in dark blue in each view, we also highlight the prefixes of the selected pattern in light blue.

Selecting a pattern does more than just highlight, however. Following Shneiderman’s “overview first, zoom and filter, details on demand” mantra [32], we include two methods for accessing details on demand. First, hovering over any representation of a pattern, whether a circle in the scatterplot view or a pattern string in the list or tree views, brings up a tooltip with the full pattern string and the pattern’s metric values. Second, Progressive Insights displays the full pattern string of any selected pattern and all available metric values in a small panel at the bottom of the interface.

Progressive visual analytics depends on the ability to make decisions while the analysis is still computing. One such capability (mentioned in Section 4) is the ability to adjust the parameters for a given analytic algorithm without waiting for a potentially long-running analytic process to complete. Users should be able to view early, initial

results, decide what new analytic parameters are required, and immediately restart the analytic with adjusted parameters. With this requirement in mind, and applying Design Goal #7, we dedicate the lower-right corner of the Progressive Insights user interface to widgets that can control the various parameters of the SPAM algorithm: the minimum support required, the level of detail, and the maximum pattern length. The analyst can adjust these parameters at any time and restart the analysis with new settings. This action pauses the current run and places the new settings at the top of a *run stack*. With the run stack, analysts can experiment with various parameter settings and harmlessly revert back to previous settings without throwing away possibly time-consuming results by pressing a “Resume previous” button, as shown in the bottom of Figure 5. A tooltip on the “Resume previous” button shows parameters of the previous run to help users keep track of paused runs. In addition, the number of paused runs on the run stack is displayed in the text of the button itself.

Similarly, another decision that analysts can make (as highlighted in Section 4) is to change the input dataset being sent to the underlying analytic algorithm. For example, a medical analyst might wish to only compare patterns amongst a subset of patients known to have a specific important pattern found earlier in an analysis. To support this task, Progressive Insights allows analysts to restart analytic on arbitrary subsets of patients as selected in any of the provided views.

6 CASE STUDY

In order to evaluate the utility of progressive visual analytics, we chose to conduct a case study of a real analyst using his own real-world datasets to demonstrate its effectiveness at reaching insights in practice. There is a growing belief in the visualization community that traditional evaluation metrics (e.g. measuring task time completion or number of errors) are often insufficient to evaluate visualization systems [5, 26, 33]. Using the evaluation methodology developed by Perer and Shneiderman [24], we conducted a 2-month long-term case study with Dr. Stein Olav Skrovseth, a medical informatician at the Norwegian Centre for Integrated Care and Telemedicine at the University Hospital of North Norway (UNN).

Dr. Skrovseth was interested in using visual analytics to determine if certain sequences of surgical events correlate with health outcomes of patients. If such correlations exist, this analysis could guide revisions or enhancements to surgical guidelines at his institution. To do his analysis, he needed to mine frequent patterns of over 87,000 patients in UNN’s surgical department. Each patient has a series of time-stamped procedures and diagnoses, which comprise the relevant temporal events to mine. Without a tool such as Progressive Insights, Dr. Skrovseth would typically have to rely on a team of technologists to re-run the data mining algorithms, who wait for the lengthy mining algorithms to complete, then wait for different technologists to create reports on the results, and then deliver the results so he can analyze them. Throughout the analysis, he expected the need to test and modify multiple cohorts, outcome measures, and algorithm parameters, which made him an ideal candidate for our evaluation, as each of these modifications in his traditional workflow generate many computation and human delay speed bumps.

6.1 Early Use

Initially, Dr. Skrovseth wished to understand any patterns among all 87,000 patients in his surgical department’s database. As hospital readmissions are often used to monitor quality of patient care in many health care organizations [38], Skrovseth selected this as the outcome variable. This divided his patient population into approximately 17,000 cases (with re-admission) and 70,000 controls (no readmissions).

During his analysis, Skrovseth would often need to restart the analytics by changing parameters such as the support threshold parameter. Without having to wait for the algorithm to finish, which could take up to 30 minutes depending on the specified support, he was immediately able to select nodes of interest (e.g. Cancer diagnoses) from the progressive results and examine the patterns occurring after such a diagnosis. By prioritizing an event, less time was wasted in conversations

with his colleagues about the dataset. The progressive nature of the tool afforded him and his team flexibility to investigate new hypotheses interactively.

Nonetheless, in this overall cohort, few strong correlations were found. While many of the strong negative correlations (e.g. emergency procedures, infections) were sensible, Skrovseth’s team hypothesized this small amount may be due to a non-balanced population with many more controls than cases.

After balancing the population cohorts, the correlations became stronger and more meaningful. It was clear certain surgeries led to fewer readmissions (Laparoscopic) whereas others were highly correlated with readmission (Gall bladder surgery). When Dr. Skrovseth presented this to a team of surgeons at a leading hospital, they agreed these results made clinical sense. One of the surgeons commented, “The advantage of visualization and Progressive Insights is that you can insert a very large number of patient variables in the model, and identify the ones that are most associated with a specific outcome.”

Interesting, the surgeons wondered if there were also some inherent biases in the data. For instance, for patients who travel very far from home, they might get readmitted in their local hospitals and this readmission data would not be captured within the dataset.

It was interesting to note that the mined event sequences turned out to be very short. While not initially expected by Skrovseth, when he analyzed the data closely, this made sense as most of the events happen concurrently with the surgery. The ability for Progressive Insights to handle concurrent events turned out to be critical.

6.2 Mature Use

After using the tool extensively and exploring multiple hypotheses, Skrovseth and his colleagues found promise in investigating a subset of surgical patients: those having an episode of gastrointestinal surgery, which left them with a cohort of 35,131 patients. However, using Progressive Insights, it became clear that only 20% (7011) of these patients admitted to the surgery department ended up having an operation, so the cohort was further refined to include only patients with operations.

After several progressive analysis sessions, where various parameters were examined, Skrovseth and his colleagues became interested in patients that were readmitted with a hospitalization within 30 days after index surgery. However, initial results viewed in Progressive Insights raised questions if there was a location bias where patients may follow-up at their local hospital instead of UNN. The dataset was refined once again to filter out patients whose home address was outside the region where UNN is the local hospital, as those patients may have been admitted to a different hospital. As Skrovseth became interested in how certain patterns correlate with gender, age, and length of hospital visit, these variables were also augmented to the dataset. Furthermore, as patterns were explored, they became curious of the correlation of patterns with admissions into the Intensive Care Unit, a separate department in UNN. This was the final augmentation of the dataset.

Skrovseth noted that the ability to quickly interpret progressive results of the cohorts allowed them to reach preliminary insights about the dataset, and afforded them the flexibility to iteratively improve the cohort without constraints. In fact, many of the insights emerged while only partial results were available. An example of this exploration is captured in Figure 8. Ultimately, this led to a variety of interesting findings relevant to understanding correlations with hospital readmissions. For example, patterns containing surgical events (Laparoscopic Surgery, Appendectomy) are strongly associated with patients who are not re-admitted, whereas others (Endoscopic insertion of stent into bile duct, Loop enterostomy) strongly correlate with re-admitted patients. Not all of the correlations were expected by the clinical research team, and they are currently tabulating their findings into a medical publication to bring this to the attention of their peers.

6.3 Outcome

Using Progressive Insights, Dr. Skrovseth and his colleagues were able to understand the temporal patterns among their surgical patients



Fig. 8. A screenshot of one of Dr. Skrovseth's sessions using Progressive Insights to discover insights in his surgical database.

and understand how certain medical events correlate with hospital readmissions. They noted that the ability to receive progressive feedback during their analysis allowed them to reach decisions faster and afforded them freedom to do more explorations of additional datasets than they otherwise would have using traditional tools. The ability to restart analytics and then select event types of interest allowed the researchers to direct the algorithm to investigate their hypotheses, and this allowed them to get fast real-time feedback, and minimize the speed bumps while interacting with time-sensitive medical experts.

The resulting analysis has led Skrovseth and his peers to a variety of findings that are currently being reported in a manuscript for a leading medical informatics journal. Skrovseth believes the potential impact of these findings is quite large, as this knowledge can inform his community about important clinical practices. Skrovseth believes these findings can lead to clues to reduce hospital re-admissions which, even if by a small percentage, can be hugely significant to hospital budgets and surgical department operations. Without progressive visual analytics, Skrovseth admits that some of their findings may have never been discovered.

7 CONCLUSION AND FUTURE WORK

In this paper, we introduce progressive visual analytics, a paradigm that gives users access to semantically meaningful partial results during the execution of analytics, and allows exploration of these partial results in integrated, interactive visualizations. We then provide design goals for how analytics can be designed to produce meaningful partial results during execution, and design goals for how visualizations can be designed to properly manage partial results without interfering with users' cognitive workflow. We also provide details of an implementation of a progressive visual analytics system, Progressive Insights, for exploring frequent sequences of temporal patterns. We then provide concrete evidence of the promise of progressive visual analytics by showcasing a long-term case study of clinical researchers utilizing progressive results while exploring frequent patterns in electronic health records.

However, there remain many aspects of progressive visual analytics to be explored. While we have presented generalizable design goals,

we have only provided a concrete implementation and evaluation for a single algorithm and task: frequent sequence mining analytics. We plan to apply the progressive visual analytics paradigm to other analytical algorithms and tasks and more complex data cases such as analyzing heterogeneous results of multiple progressive algorithms. The system we presented also utilizes a linear search strategy, and we hope to explore the design constraints for alternative search strategies such as a provenance-driven nonlinear approach [6]. Another unexplored aspect of progressive visual analytics is that progressive analytics may produce partial results at very different frequencies. We plan to enhance our guidelines for progressive visualization designers anticipating varying rates of updates. Furthermore, we plan to evaluate progressive visual analytics with additional case studies in other domains besides medicine.

While we deliberately opted not to run a controlled study so we could hear insight-based stories of success from domain experts, future work demands to concretely measure the advantages and disadvantages of interfaces that do and do not support progressive analytics. Such analysis will enhance the discussion of the benefits and trade-offs of our approach.

While much future work remains in the area of progressive visual analytics, we conclude by reiterating that in the age of Big Data with users requiring results from complex analytics, the need for interactive, user-directed analytics is clear. We view our work as a first step in highlighting the promise of integrating continuous partial results into visual analytics tools.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Stein Skrovseth and his colleagues for sharing their usage of Progressive Insights in our case study. In addition, we wish to thank Fei Wang for sharing his expertise in frequent sequence mining. Finally, we wish to thank the VAST reviewers for their valuable feedback. The work presented in this paper was performed while all authors were affiliated with the IBM T.J. Watson Research Center.

REFERENCES

- [1] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Human-Computer Interaction Series. Springer-Verlag, London, 2011. DOI: 10.1007/978-0-85729-079-3.
- [2] M. Angelini and G. Santucci. Modeling incremental visualizations. In *Eurovis Workshop on Visual Analytics*, pages 13–17, 2013.
- [3] M. Ankerst, A. Kao, R. Tjoelker, and C. Wang. DataJewel: integrating visualization with temporal data mining. In S. J. Simoff, M. H. Bhlen, and A. Mazeika, editors, *Visual Data Mining*, number 4404 in LNCS, pages 312–330. Springer Berlin Heidelberg, Jan. 2008.
- [4] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential Pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '02*, pages 429–435, New York, NY, USA, 2002. ACM.
- [5] E. Bertini, H. Lam, and A. Perer. Summaries: a special issue on evaluation for information visualization. *Information Visualization*, 10(3), 2011.
- [6] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. VisTrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD '06*, pages 745–747, New York, NY, USA, 2006. ACM.
- [7] C. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19:281–288, 2007.
- [8] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
- [9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, Jan. 1977.
- [10] A. Dix and G. Ellis. by chance enhancing interaction with large data sets through statistical sampling. In *Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '02*, pages 167–176, New York, NY, USA, 2002. ACM.
- [11] S. Eick, J. Steffen, and J. Sumner, E.E. Seesoft—a tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering*, 18(11):957–968, 1992.
- [12] J. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *Visual Analytics Science And Technology, 2006 IEEE Symposium On*, pages 167–174, 2006.
- [13] P. Federico, W. Aigner, S. Miksch, F. Windhager, and L. Zenk. A visual analytics approach to dynamic social networks. In *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies, i-KNOW '11*, pages 47:1–47:8, New York, NY, USA, 2011. ACM.
- [14] 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 SIGCHI Conference on Human Factors in Computing Systems, CHI '12*, pages 1673–1682, New York, NY, USA, 2012. ACM.
- [15] Google Inc. Google charts.
- [16] E. Hetzler, V. Crow, D. Payne, and A. Turner. Turning the bucket of text into a pipe. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*, pages 89–94, Oct. 2005.
- [17] S. Huron, R. Vuillemot, and J.-D. Fekete. Visual sedimentation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2446–2455, 2013.
- [18] M. Krstajic, E. Bertini, and D. Keim. CloudLines: compact display of event episodes in multiple time-series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432–2439, 2011.
- [19] L. Lins, J. Klosowski, and C. Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, 2013.
- [20] Z. Liu, B. Jiang, and J. Heer. imMens: real-time visual querying of big data. *Computer Graphics Forum (Proc. EuroVis)*, 32(3), 2013.
- [21] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. The Regents of the University of California, 1967.
- [22] M. Monroe, R. Lan, H. Lee, C. Plaisant, and B. Shneiderman. Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2227–2236, 2013.
- [23] M. Monroe, R. Lan, J. Morales del Olmo, B. Shneiderman, C. Plaisant, and J. Millstein. The challenges of specifying intervals and absences in temporal queries: A graphical language approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '13*, pages 2349–2358, New York, NY, USA, 2013. ACM.
- [24] A. Perer and B. Shneiderman. Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 265–274, 2008.
- [25] A. Perer and F. Wang. Frequency: Interactive mining and visualization of temporal frequent event sequences. In *Proceedings of the 19th International Conference on Intelligent User Interfaces, IUI '14*, pages 153–162, New York, NY, USA, 2014. ACM.
- [26] C. Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, pages 109–116. ACM, 2004.
- [27] C. Plaisant, D. Heller, J. Li, B. Shneiderman, R. Mushlin, and J. Karat. Visualizing medical records with LifeLines. In *CHI '98 Conference Summary on Human Factors in Computing Systems, CHI '98*, pages 28–29, New York, NY, USA, 1998. ACM.
- [28] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. LifeLines: visualizing personal histories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '96*, pages 221–227, New York, NY, USA, 1996. ACM.
- [29] R. Rosenbaum and H. Schumann. Progressive refinement: more than a means to overcome limited bandwidth. volume 7243, pages 724301–724301–12, 2009.
- [30] R. Rosenbaum, J. Zhi, and B. Hamann. Progressive parallel coordinates. In *2012 IEEE Pacific Visualization Symposium*, pages 25–32, Feb. 2012.
- [31] C. Safran, M. Bloomrosen, W. E. Hammond, S. Labkoff, S. Markel-Fox, P. C. Tang, and D. E. Detmer. Toward a national framework for the secondary use of health data: An american medical informatics association white paper. *J Am Med Inform Assoc*, 14(1):1–9, 2007.
- [32] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages, 1996. Proceedings*, pages 336–343, Sept. 1996.
- [33] B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Proceedings of the 2006 BELIV Workshop*, pages 1–7, 2006.
- [34] Square Inc. Cubism.js.
- [35] Tableau Software. Tableau.
- [36] J. J. Thomas and K. A. Cook, editors. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Center, 2005.
- [37] Tibco Software. Spotfire.
- [38] C. van Walraven, C. Bennett, A. Jennings, P. Austin, and A. Forster. Proportion of hospital readmissions deemed avoidable: a systematic review. *Canadian Medical Association Journal*, 183(7):391–402, 2011.
- [39] T. Wang, C. Plaisant, B. Shneiderman, N. Spring, D. Roseman, G. Marchand, V. Mukherjee, and M. Smith. Temporal summaries: Supporting temporal categorical searching, aggregation and comparison. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1049–1056, 2009.
- [40] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events: Discovering patterns in electronic health records. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08*, pages 457–466, New York, NY, USA, 2008. ACM.
- [41] P. C. Wong, W. Cowley, H. Foote, E. Jurrus, and J. Thomas. Visualizing sequential patterns for text mining. In *IEEE Symposium on Information Visualization, 2000. InfoVis 2000*, pages 105–111, 2000.
- [42] K. Wongsuphasawat and D. Gotz. Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2659–2668, 2012.
- [43] K. Wongsuphasawat, J. A. Guerra Gmez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. LifeFlow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11*, pages 1747–1756, New York, NY, USA, 2011. ACM.