Temporal Event Sequence Simplification

Megan Monroe, Rongjian Lan, Hanseung Lee, Catherine Plaisant, Ben Shneiderman

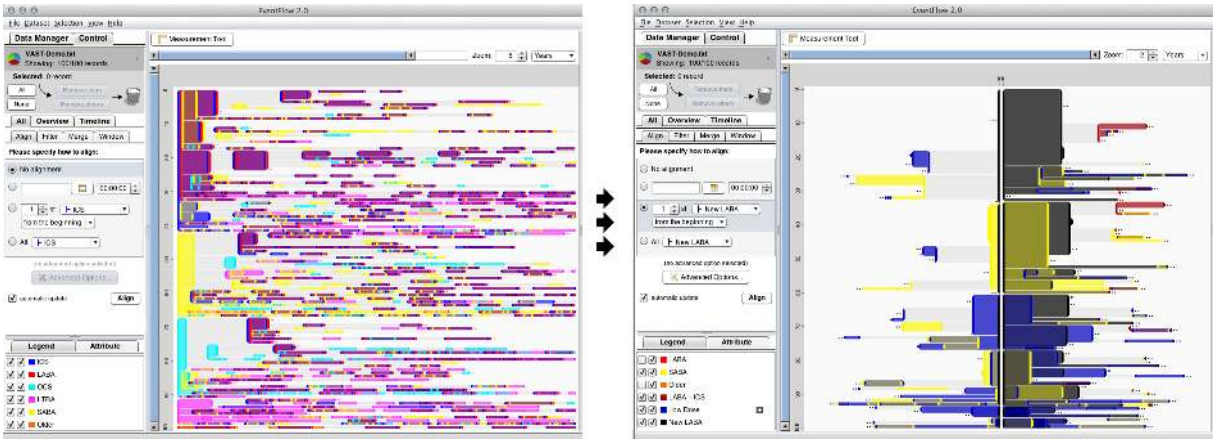

Fig. 1: In EventFlow, the original LABA dataset, consisting of over 2700 visual elements (left), was quickly pared down to the events most critical to the study. The simplified dataset (right) consists of only 492 visual elements, an 80% reduction in visual complexity. From this simplified figure, aligned by the patients' "new" LABA prescription, researchers were immediately able to notice the data sparsity on the left side of the alignment point, indicating that patients had not received other treatments in the months leading up to their LABA prescription (i.e. not following the recommended practices).

Abstract—Electronic Health Records (EHRs) have emerged as a cost-effective data source for conducting medical research. The difficulty in using EHRs for research purposes, however, is that both patient selection and record analysis must be conducted across very large, and typically very noisy datasets. Our previous work introduced EventFlow, a visualization tool that transforms an entire dataset of temporal event records into an aggregated display, allowing researchers to analyze population-level patterns and trends. As datasets become larger and more varied, however, it becomes increasingly difficult to provide a succinct, summarizing display. This paper presents a series of user-driven data simplifications that allow researchers to pare event records down to their core elements. Furthermore, we present a novel metric for measuring visual complexity, and a language for codifying disjoint strategies into an overarching simplification framework. These simplifications were used by real-world researchers to gain new and valuable insights from initially overwhelming datasets.

Index Terms—Event sequences, simplification, electronic heath records, temporal query.

1 Introduction

Temporal event data is at the forefront of today's Big Data boom. Across the world, sensors and forms and spreadsheets are capturing and storing every aspect of our existence as sequences of categorized, timestamped events. The motivating question is simple: how can we best leverage the things that have already happened to inform our future actions?

To this end, researchers approach these datasets with the goal of gaining understanding on two different levels:

- **Intra-Record Understanding:** Knowledge is gained about the sequence of events that comprises a single record (i.e. a patient record or a shipment record): Why was this patient transferred from the Emergency Room to the Intensive Care Unit (ICU)?
- **Inter-Record Understanding:** Knowledge is gained about population-level trends and patterns across an entire group of records: Of all the patients who were transferred to the ICU, what was the most common outcome?

- *Megan Monroe, Rongjian Lan, Hanseung Lee, Catherine Plaisant, and Ben Shneiderman are with the Department of Computer Science & Human-Computer Interaction Lab, University of Maryland.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 27 September 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

Two years ago, our effort to help researchers achieve these two levels of understanding resulted in the LifeFlow visualization tool [31]. LifeFlow was primarily used for analyzing point-based process log data. It combined the list-based display of its predecessor, LifeLines2 [28], with an aggregated display that summarized the entire dataset in a single view. The aggregated display (discussed in further detail in Section 3) was extremely effective at revealing inter-record trends, while the list-based display allowed users to explore the intra-record patterns.

LifeFlow caught the attention of researchers at the U.S. Army Pharmacovigilance Center (PVC). They were conducting a long-term study on asthma treatment, and the prescription of Long-Acting Beta Agonists (LABAs). This type of medication should only be prescribed when alternate treatments have proven ineffective. It is meant to be both preceded and followed by prescriptions for other, less-potent asthma medications [2]. The question researchers were trying to answer, given a sample dataset of 100 patients, was whether this recommended practice was actually being followed. Their request was that LifeFlow be given the capability to handle not only point events such as a heart attack or a surgery, but also interval events such as medication prescriptions and hospitals stays. This required an extensive remodeling of LifeFlow's data structure, display rendering, and the mechanisms for exploring and querying the data [17]. Happy to oblige, however, the application was restructured and redubbed as EventFlow [18](see Figure 2). Our work, it would seem, was finished.

Then the PVC researchers loaded their LABA dataset...

The resulting display, shown in Figure 1-left, was so visually complex that it was initially described as "confetti." EventFlow's aggregated display is generated by grouping records with the same event sequence, however, the patient records in the LABA dataset were so long and varied that the chances of two records having the same sequence were extremely small. In this case, EvenFlow cannot effectively group records, and the aggregated display defaults to rendering each record individually. The ability to leverage the visualization for inter-record understanding is lost.

This visual complexity became less of an isolated incident and more of a theme as we took on additional case studies in the medical domain. The complexity of raw EHR data, manifests itself on both the intra-record and inter-record fronts. For example, it is not always clear which symptoms motivate which treatments. Relationships between events have to be inferred from the raw data alone. A single patient record may contain simultaneous treatments for multiple conditions (intra-record complexity). Additionally, EHR data is collected under real-world conditions, where variables cannot be meticulously controlled. The strategy for boosting the signal through this noise then, is to use increasingly larger datasets (inter-record complexity).

Despite the size and heterogeneity of EHR datasets, however, the questions being asked of them typically only revolve around a subset of patients and events. Furthermore, due to the nuances of data entry and extraction, the event sequences in an initial dataset are frequently *more* complex than the real-world events that they represent. There is a huge opportunity to discard irrelevant data points, bearing in mind that relevance changes on a study-by-study or even a question-by-question basis. This paper reports on the design of a series of intuitive and effective controls that allow users to quickly simplify an event record dataset down to its most meaningful elements and most accurate representation. This process was structured around a design study framework as described in [21]. Not surprisingly, the effort focused on two different strategies for simplifying a dataset:

- **Intra-Record Simplification:** Altering the events within an event record.
- **Inter-Record Simplification:** Altering the subset of records in the dataset.

These simplifications were evaluated in multiple long-term case studies including:

- **LABA Study** with the U.S. Army Pharmacovigilance Center.
- **Opioid Misclassification** with the U.S. Army Pharmacovigilance Center.
- **The Diabetic Foot** with the University of Florida College of Medicine.
- **Attention Deficit Disorder Treatment** with the University of Maryland School of Medicine.
- **Pediatric Trauma Procedures** with MedStar Health Facilities.

The LABA study will serve as an overarching example throughout this paper, and the Opioid Misclassification study will be presented, in greater detail, in Section 7. We also describe the use of simplification on an experimental dataset of basketball statistics to demonstrate the generalizability of these approaches. Across all of these studies, we found that complex datasets could consistently be simplified by around 80%, and that extraneous records and events could be both removed and re-inserted in only a few clicks. The resulting visualizations allowed researchers to generate novel insights through both hypothesis generation and hypothesis testing, and communicate these results to their peers in clean, readable figures.

These new simplifications required extensive software engineering efforts with novel data structures, but more importantly, they required fresh ways of thinking for both our users and ourselves. This involved defining a clear language of simplification, as well as metrics for evaluating its success. Our filter and transformation-based simplifications, and the open-ended possibilities of a universal, Find-and-Replace system of simplification dramatically expand the potential to extract signal from noise in large datasets.

This paper is organized as follows: Previous work on event sequence simplification is presented in Section 2. A discussion of visual complexity, and our novel metric for measuring it, follows in Section 3. The process of designing simplifications, and the evolution from filter-based simplifications to a universal, Find & Replace interface is described in Sections 4 through 6. The effectiveness of these methods and our work on two different datasets is presented in Sections 7 and 8. Finally, we discuss future work and conclude in Section 9.

2 Background

Solutions for dataset simplification have been explored for many different data types, including graphs and maps [19, 9, 6]. For event sequences, however, current solutions revolve around two primary techniques: filtering and clustering.

Filtering keeps control in the users' hands, allowing them to remove events and records based on preexisting attributes. These strategies are discussed in detail in Section 4. Filter options can be found in a wide range of tools for analyzing temporal event sequences. They are a principal component of the Align, Rank, and Filter framework for visualizing temporal data [29], employed by numerous applications [11, 27, 20]. Filtering is also relatively straightforward to perform using command-based tools [22, 10, 4]. However, there are no existing applications that extend filtering into an overarching framework of data simplification that gives users precise control over how their data is represented.

Clustering entrusts data mining algorithms to find meaningful patterns, despite noisy data, in order to group similar patients. However, this method brings about a new set of difficulties. First, the definition of what constitutes similarity between two patients can change on a study-by-study, or even question-by-question basis. In one study, the paramount metric of patient similarity may be the duration of time that a patient was taking a medication. For another study, it may be the number of emergency room visits. Because of this, the underlying clustering models must be constantly tuned and recalibrated. Even with supervised machine learning techniques [24, 23, 32], the extent to which rules can be effectively carried over between studies is unclear. Additionally, medical researchers are not likely to have expertise in data mining or clustering. This makes it difficult for these front-end users to evaluate the success of the clustering and provide meaningful feedback to the back-end developers. Finally, even when these methods achieve an effective grouping, the problem remains of generating a meaningful display from which inter-record hypotheses can be confirmed or refuted.

3 Complexity and Aggregation

It is impossible to address simplification without first defining some notion of complexity. Unfortunately, there is no single, universal metric for defining and measuring visual complexity. The topic has been extensively discussed in psychology [5, 26, 5] and, more recently, computer science [30, 15, 7]. The lack of a consensus suggests that the scale of visual complexity depends, to some extent, on the style and structure of the visualization itself.

To further complicate matters, the complexity of the visualization becomes irrelevant if the dataset has been simplified beyond use. This is a fine line to walk when the visualization is being used for both hypothesis testing and hypothesis generation. For hypothesis testing, there is a specific question to be answered, and users typically have a clear idea of what events and records are relevant to that question. In this case, the visualization can be simplified until only those elements remain. Hypothesis generation, however, is a more open-ended process. Users run the risk of oversimplifying their data, and missing potential insights. Thus, while complexity metrics are presented here and utilized in the following sections, we emphasize that these metrics must be tempered with some subjective notion of maintaining the integrity of the dataset.

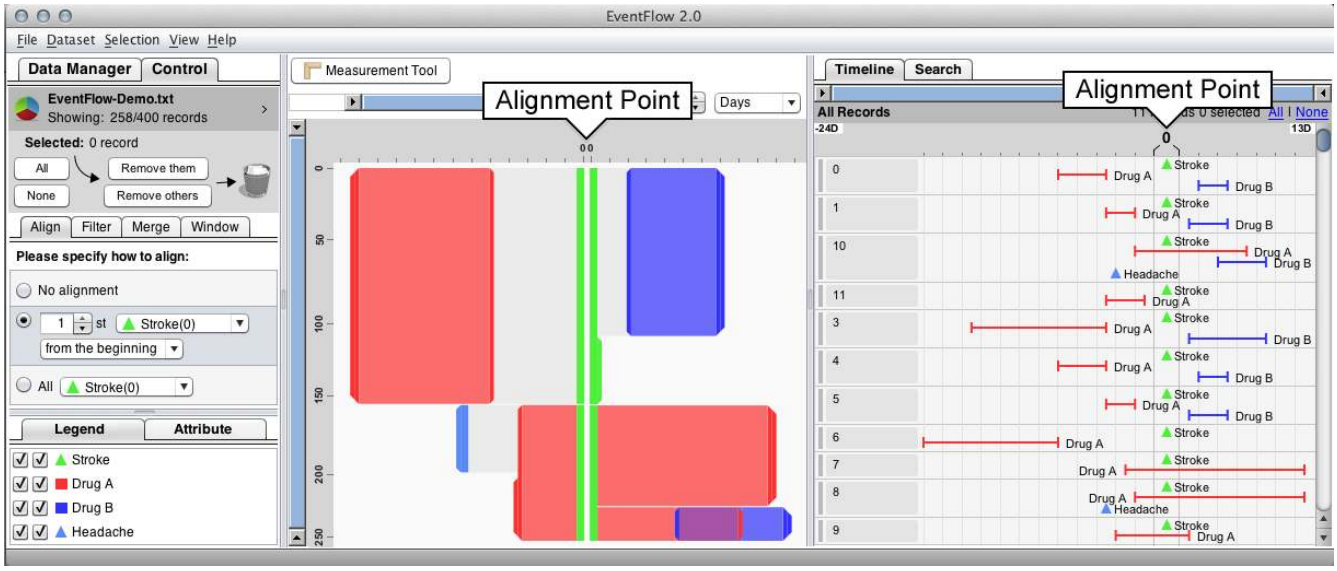

Fig. 2: EventFlow consists of three panels: the control panel (left), the aggregated record display (center), and the individual record display (right). Here, a sample dataset is aligned by the Stroke event in each patient record, creating mirrored alignments in both the individual and aggregated displays. In EventFlow, record filtering is done using the "Remove" buttons at the top of the control panel. Category filtering is done using the checkboxes in the legend. Time and Attribute filtering is done using the "Window" and "Attribute" tabs respectively.

As mentioned previously, EventFlow creates an aggregated view of a dataset by grouping records with the same event sequence. This grouping is done using a tree-structure that branches as the event sequences diverge from each other (see [31] for a detailed description this process). Events are represented using vertical bars, where the height of each bar is dictated by the number of records grouped at that branch. The default behavior is to root the aggregation at the beginning of each record, however users can also root the aggregation at any alignment point of interest. For example, in the LABA study, patients were frequently aligned by their first LABA prescription. In this case, two aggregations are done: one for the events that led up to this alignment point, and one for those that followed it (see Figure 1-right).

Because of this aggregation strategy, metrics such as the number of patient records, or even the number of total events in the dataset are not reliable predictors for the complexity of the resulting visualization. For example, Figure 2 depicts over 250 patient records in a clean and easily readable display. Ultimately, the complexity of EventFlow's visualization is dictated by how frequently, and the extent to which the records in the dataset share the same event sequence.

As such, visual complexity will be defined and measured by two different metrics: the number of visual elements in the display, and the average size of these elements. The first metric, proposed in [7], is simply a count of each vertical, colored bar in EventFlow's aggregated display. This metric gives a sense of how difficult it will be for users to weave the unique elements in the display into a holistic impression of their dataset. The number of visual elements should *decrease* as the dataset is simplified.

The second metric, defined more specifically as the average height of each vertical bar (as a percentage of the display height), provides insight into how well the individual records are being aggregated together into a summarizing display. Though this metric is specific to EventFlow, it draws on two of the oldest notions of visual complexity: separability and information density. Separability, introduced by Garner in 1974 [8], describes how larger items are easier to distinguish from one another, thus reducing the perceived complexity. Information density, proposed by Edward Tufte [25], relates to the amount of information conveyed by each visual element. In EventFlow, the height of each vertical bar represents the number of records aggregated at that branch. Thus, the average height across all these elements should *increase* as the dataset is simplified.

The initial LABA dataset began with over 2700 visual elements, each averaging only 1.14% of the total display height (Figure 1-left). From this display, the PVC researchers found it virtually impossible to visually parse out any meaningful trends. In the following three sections, we will describe the simplifications that these researchers used to ultimately gain novel insight from this dataset using EventFlow.

4 Filter-Based Simplifications

Filter-based simplifications allow users to remove events and records from the dataset based on the fundamental features of the data type. They have been implemented, in some form, across a wide range of visualization and command-based tools and are an essential component of visual analytics. We discuss them here for the sake of completeness, but more importantly, to demonstrate that in real-world data analysis, filter-based simplifications are most effective when they can be interleaved with more advanced simplifications. In EventFlow, filter-based simplifications are designed to be readily accessible and easily reversible (see Figure 2).

The use of EventFlow's filter-based simplifications can be demonstrated by an initial question that the PVC researchers had while reviewing the LABA dataset. The dataset was set up such that one LABA prescription in each patient record had been flagged with an attribute to indicate that this was the patient's "first" LABA prescription. It was around this prescription that they planned to look for the intended pattern of less-potent asthma medications.

To make this attribute visible in EventFlow's aggregated display, the researchers converted it into a marker event that was inserted into each record (see the full description of marker events in Section 5). When the dataset was then aligned by this marker, they noticed that, for many patients, their so-called "first" LABA prescription was actually preceded by other LABA prescriptions.

The researchers determined that this was occurring because the criteria used for identifying the "first" LABA prescription was only contingent on there being no other LABA prescriptions in the preceding 3 months. However, the original data extraction was done by selecting all prescriptions for the entire year preceding the "first" LABA, resulting in the inclusion of LABA prescriptions that were greater than 3 but less than 12 months prior. Their question, was whether this inconsistency would be eliminated if they updated the extraction script to require that the "first" LABA be preceded by 6 months without another LABA prescription.

By Record

Filtering by record allows users to remove a subset of records, either through query (described in further detail in Section 6), or by simply clicking any element in the individual or aggregated display.

Removing a large subset of uninteresting records can free up a substantial amount of screen space to render the interesting elements at a larger scale. For example, out of the entire LABA dataset, the question of whether or not it would be necessary to do a new data extraction could be answered by examining the relationship between only two event points in a subset of the patient records. To isolate the critical records, the researchers used EventFlow's query system [17] to identify patients whose "first" LABA prescription was preceded by another LABA prescription. Records not matching this search were filtered from the dataset, narrowing 100 records down to 26 (see Figure 3, Steps 1 $\rightarrow$ 2).

Step 1: 2724 elements, 1.14% of display height per element
Step 2: 993 elements, 4.69% of display height per element

By Category

EventFlow's most fundamental form of intra-record simplification, is the ability to remove event categories from the display using the category check boxes in EventFlow's legend. Reducing the number of event categories can significantly increase the chances that two records will have the same event sequence and thus aggregate into fewer and larger visual elements. Returning to the question of whether to perform a new data extraction, the PVC researchers narrowed the LABA dataset down further by filtering out all of the event categories except LABA and "first" LABA events (see Step 3 of Figure 3). Some of these filtered categories, in fact, remained excluded from the dataset for the duration of the study.

Step 3: 99 elements, 15.01% of display height per element

By Time

Filtering by time allows users to limit EventFlow's aggregated display to a certain window of time around the current alignment point. Since the chances of two records having the same event sequence decreases as you get further and further away from the alignment point, the visual elements at the far ends of the display will be the smallest. The time filter allows users to limit the display to only the largest, most information dense elements.

The final step in simplifying the LABA dataset to answer the "first" LABA question was to filter out all events more than 6 months prior to the alignment point. In doing this, researchers could clearly see that only one patient had their *actual* first LABA prescription within 6 months of their "first" LABA event. All of the other patients had LABA prescriptions prior to this cutoff point. This meant that only one patient would have been excluded from the dataset if the extraction were changed to require that the "first" LABA be preceded by 6 months without another LABA prescription. Based on this information, the PVC researchers decided that it was not worth doing a new data extraction. The "first" LABA prescription, from this point on, was referred to as a "new" LABA prescription. This simplification process can be seen in its entirety in Figure 3.

Step 4: 35 elements, 22.75% of display height per element

By Attribute

EventFlow allows attributes to be assigned to both records and individual events. For example, a patient record might have the attribute "gender," and a prescription event might have the attribute "dosage." Users can perform both intra-record simplification and inter-record simplification by removing events or records based on their attributes. Again, this serves to either increase the chances of aggregation, or free up screen space for the records of interest. While this filtering technique was not used as part of the LABA study, it was used extensively on many of the other datasets.

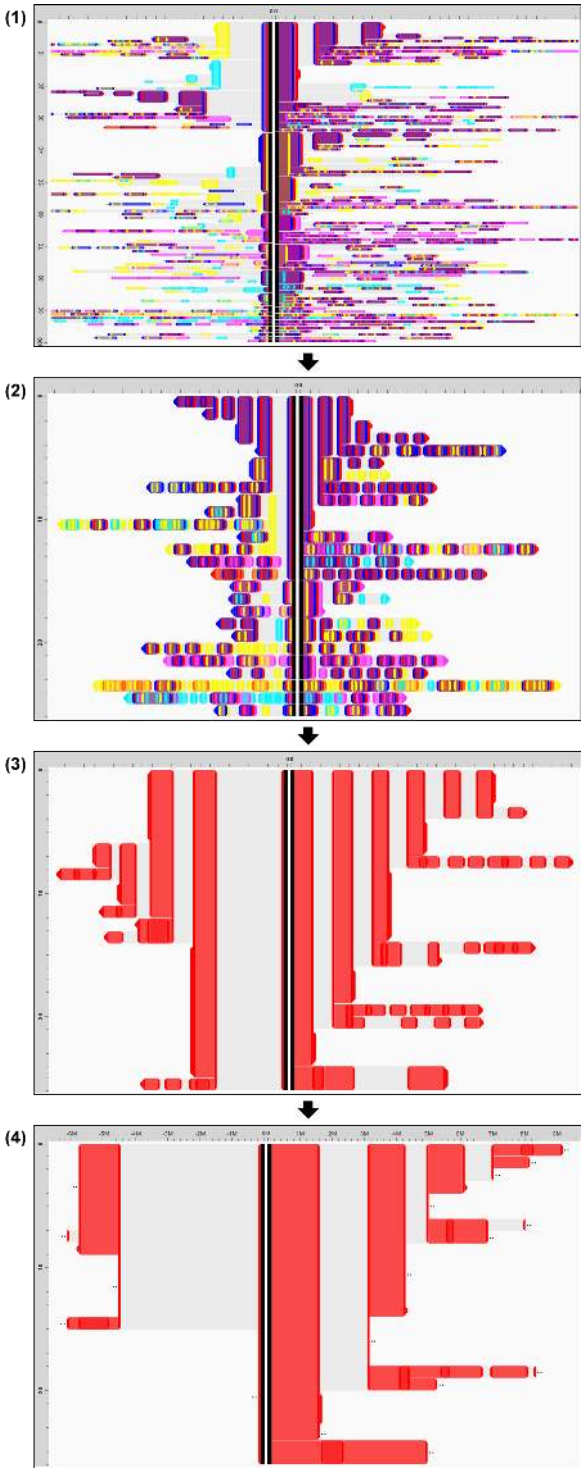

Fig. 3: Should a new data extraction be done to exclude patients who had LABA prescriptions prior to their "first" LABA prescription? - **Step 1:** The original LABA dataset is aligned by the "first" LABA marker event (in black). **Step 2:** Patients without a LABA prescription before their "first" LABA are filtered out. **Step 3:** Categories other than LABA prescriptions (in red) and the "first" marker are filtered out. **Step 4:** Data is filtered to 6 months around the alignment point.

5 Transformation-Based Simplifications

While the filter-based simplifications in the previous section can significantly reduce large datasets, they are designed only to *remove* information. However, real-world datasets are not only messy with respect to volume, but also with respect to the logical translation between the event data, and the real-world events that transpired. Users frequently need to manipulate their data, not just by removing events, but by transforming the way it's represented, a process typically known as data wrangling [12, 13].

To discover the types of data transformations that were needed be-

yond filtering, we spent extensive time with all of our collaborators as they explored their datasets. Based on their difficulties and feedback, we designed a series of transformation-based simplifications that met universal demands across all five research groups. These simplifications were deployed, one by one, in order to gain an initial understanding of whether they were effective at reducing complexity.

Interval Event Merging

In many cases, a given dataset is not only unnecessarily complex, but it is more complex than the events that actually transpired. For example, in the LABA dataset, prescription intervals were calculated based on the dispensing date and the intended duration of the prescription. The start and end date of each prescription then, was dictated substantially by the behavior of the patient: Perhaps it was convenient for the patient to refill their asthma medication a few days before the previous one ran out. In the raw event data, this presents as a series of overlapping and disjoint medication intervals.

When researchers at the PVC analyze medication use, however, they are primarily concerned with exposure. That is, the time during which the patient was actually taking a medication and exposed to its effects. Exposure can only be roughly inferred from a series of prescriptions, but it can generally be assumed that if a patient receives a series of thirty-day prescriptions, with each prescription starting a few days before or after the previous one ends, that this indicates a continuous, steady exposure. Short gaps and overlaps between successive prescriptions do not necessarily imply that the patient stopped taking the medication or took a double dose during those few days. It is important to distinguish this scenario, which results from an obvious and common behavior, from situations where a combined dose of a single medication is a critical and interesting phenomenon.

In order to better represent exposure, and not to clutter the display with clinically uninteresting information about the exact dates of prescription refills, EventFlow provides a simple interface that allows for multiple interval events of the same category to be merged into a single interval event (see Figure 4). This can be done in two ways: eliminating gaps of a certain duration, and eliminating overlaps of a certain duration. This feature was designed to serve as a shortcut to two subsets of Allen's 13 interval relationships [1]: disjoint intervals and overlapping intervals. Much like Morchen's hierarchical interval model [16], interval merging focuses on duration and coincidence, rather than the exact sequence of interval end-points.

For the LABA study, any prescriptions within 14 days of each other (a standard allowance) were merged into a single exposure interval. This simplification was performed for nearly every medication category in the dataset. Interval merging can be easily undone, or saved permanently as a new dataset.

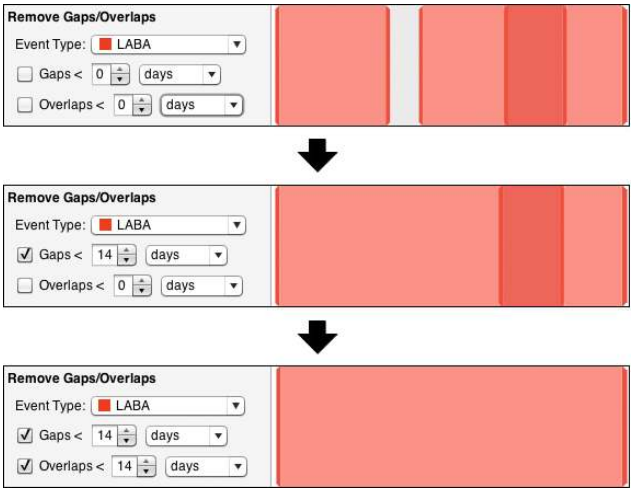

Fig. 4: A series of disjoint and overlapping intervals are merged into a single interval.

Category Merging

Category merging allows for multiple event categories to be combined into a single meta-category. This feature is particularly useful when individual medication types can be merged into an overarching drug class, or when drugs are prescribed in tandem. In the LABA study, three different medications were considered to be similar, low-dose treatment strategies that could serve as a valid precursor to a LABA prescription. These medications were merged into a single event category entitled "Low-Dose."

Unlike category filtering, category merging does not reduce the number of events being displayed. However, these meta-categories increase the chances that two records will have the same event sequence, and thus be merged into larger bars in the aggregated display (see Figure 5).

Marker Event Insertion

In many datasets, certain event types repeat over and over again throughout a single record. However, for a particular question, only one of these occurrences may be of paramount interest. Marker events allow users to insert a new event at any alignment point within the data. For example, in the LABA study, the PVC researchers aligned their dataset by the LABA event flagged with the "first" attribute, and inserted a marker event at this point in each record that was (eventually) called "New LABA" (see Step 1 of Figure 3). For questions pertaining only to the "new" LABA prescription then, the entire category of LABA prescriptions could be filtered from the dataset, leaving only the single "New LABA" marker. Swapping an entire category for one event decreases the number of visual elements in the display and increases the chances of larger, more information dense elements.

6 A Universal Simplification System

EventFlow's filter and transformation-based simplifications proved to be extremely effective in simplifying datasets, both by removing unnecessary records, and by re-representing the events in the remaining records. So effective, in fact, that their deployment was immediately followed by a barrage of requests from our users for increasingly customized simplifications. In some cases, the feature being requested was specific to a single study, or even a single research question. It did not make sense to continue to implement these features on an ad hoc basis, as separate interfaces within EventFlow. The result would have been an extremely cluttered and complex application. Upon further reflection, however, it became apparent that all of these simplifications had the same general objective: users wanted to find a certain event of pattern of events, and replace it with a more meaningful representation.

In addition to EventFlow's simplification features, development had just been completed on a graphic-based, advanced query system. The advanced query system allowed users to draw out their query using the same icons that are used to represent events in the individual record display. The goal of this query system was to give users more precise control over inter-record simplification, offering access to the full range of Allen's interval relationships [1], as well as a host of other temporal features. Users could select either the records that matched their query, or the records that did not match their query, and then remove the selected records from the dataset. The system had been extensively tested both in lab settings and with users to evaluate accuracy and usability, including the clarity of query specification language and the handling of false positives/negatives[17]. The question then, was whether we could leverage this existing system to also perform intra-record simplification.

Find & Replace

To perform intra-record simplification, EventFlow's advanced query system was augmented to include a replace feature [14]. This allows users to not only find a sequence of events, but to replace it with an event sequence of their choosing. The replacement sequence is specified using the same graphical language that is used to specify the search sequence (see Figure 6), and can include existing event categories or new event categories. Find & Replace capitalizes on the

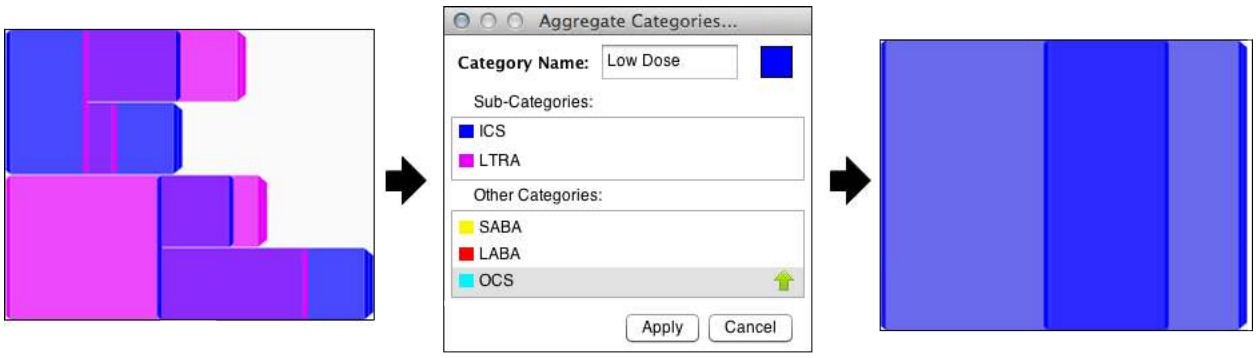

Fig. 5: Four different event permutations (left) involving an inhaled corticosteroid (ICS - in blue) and a leukotriene-receptor antagonist (LTRA - in pink) are aggregated into a single grouping (right) when these two categories are merged together.

fact that users are typically very familiar with this functionality based on their experience with a wide range of other applications including word processing and spreadsheet manipulation.

In the LABA study, Find & Replace was used when the researchers observed that LABAs are typically prescribed in tandem with an Inhaled Cortical Steroid (ICS). The result, in the aggregated display, is a sequence of four event points that essentially represent one treatment. This pattern was simplified by replacing it with a single interval, representing the combined prescription of LABA + ICS (see Figure 6). Furthermore, as described previously, each record contained a point event that marked the start of a "new" LABA prescription. This marker and the interval it tagged were replaced, by representing it as simply a "New LABA" interval. In total, the following sequence of simplifications were performed on the LABA dataset:

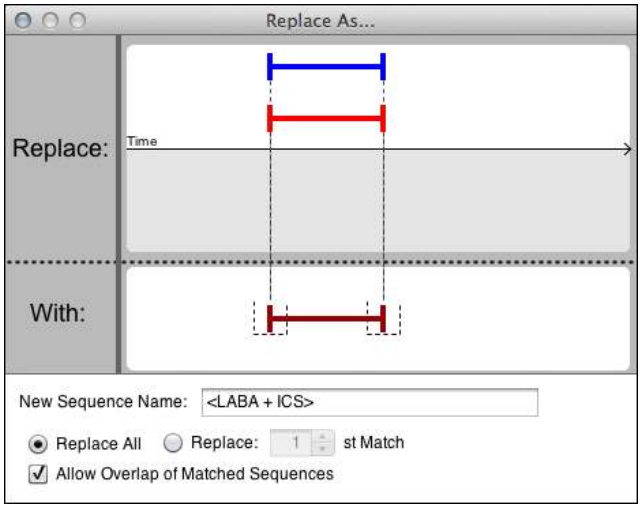

Fig. 6: Concurrent LABA (in bright red) and ICS (in blue) prescriptions are replaced by a single interval representing the combined treatment of LABA + ICS (in dark red).

1. Add marker event for "new" LABA prescriptions.
   2724 elements, **1.14%** of display height per element
2. Filter out extraneous categories.
   2694 elements, **1.15%** of display height per element
3. Replace concurrent LABA and ICS prescription with single LABA + ICS interval.
   2061 elements, **1.16%** of display height per element
4. Replace LABA prescription marked as "new" with a single New LABA interval.
   1993 elements, **1.15%** of display height per element
5. Merge intervals to reflect exposure.
   1450 elements, **1.19%** of display height per element
6. Aggregate similar medications into single category.
   1332 elements, **1.25%** of display height per element
7. Align by point of interest.
   1123 elements, **1.56%** of display height per element
8. Filter display to the six months on either side of the alignment.
   491 elements, **2.08%** of display height per element

The resulting, aggregated display is shown in Figure 1-right. This simplified display consists of only 492 visual elements, each averaging 2.08% of the display height per element. This constitutes an 81% reduction in visual elements, and an 82% increase in the average size of each element. It becomes possible to visually extract meaningful information from the display. For example, the PVC researchers immediately observed that a significant number of patients had not received any other treatment in the six months prior to their LABA prescription, which is not the recommended practice. This was a new and valuable insight that generated countless other questions.

It is critical to point out that this series of simplifications to the LABA dataset was only an *initial* pass to narrow the data down to the general theme of the research objective. As questions get more specific, the dataset can be further simplified. Find & Replace can also be used to insert new events (without removing existing events), which allows users to see exactly which events are being selected by the "find" component of this feature. The system can also be used to delete events (by replacing a search sequence with an empty replace sequence). Modified records can be saved as a new dataset, and all replacement sequences are catalogued, allowing users to undo them if need be.

Most importantly, Find & Replace can be used to replicate the functionality of both filter and transformation-based simplifications in a single interface. As a result, the overall application can be trimmed down to a much simpler set of interfaces and controls without compromising any functionality. Alternatively, the more specialized simplification interfaces could remain in the application to serve as introductory training for the more complex, Find & Replace interface.

7 Opioid Misclassification

In addition to the LABA study, the researchers at the PVC were investigating opioid use as part of a second long-term study. Opioids, a class of medications that are typically prescribed for pain management, can be habit-forming when taken improperly and are blamed for an increasing number of drug related deaths [3]. In the most recent phase of this work, researchers were trying to determine within a sample dataset of 1000 records if patients that had been previously classified as acute users were actually misclassified chronic users. These classifications are determined by reviewing a series of prescriptions, which could be for either high or low dose opioids, and consolidating them into "episodes" that are then classified as acute, intermediate or chronic. The question was whether patients with repeated acute episodes should be classified as chronic users.

The first step of the simplification process was for the researchers to take the raw prescription data (1064 visual elements, with each element averaging only .49% of the total display height), and restructure it into acute, intermediate, and chronic episodes. An episode was defined as any consecutive prescriptions within 14 days of each other. To account for this, they first merged the high and low dose prescriptions into a single event category, and then used the Find & Replace system to mark the first prescription in each episode (see Figure 7).

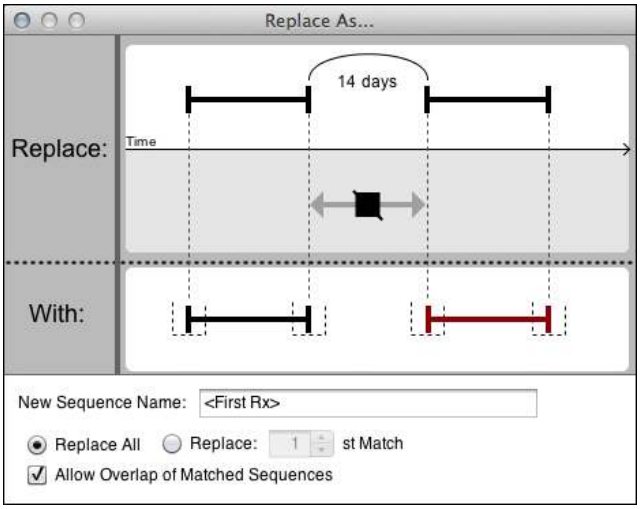

Fig. 7: The first prescription in each episode, defined as any prescription preceded by 14 days without another prescription, is replaced with a marker interval (in red). A second replacement was also done to replace the first prescription of each record, which would also be considered the start of an episode, with the same marker interval.

From here, they replaced episodes with acute, intermediate, and chronic events based on the duration of the episode and the number of prescriptions. For example, acute episodes were any episodes under 60 days and consisting of two or fewer prescriptions. These were identified by first replacing one and two prescription intervals with a placeholder interval. Then, placeholder intervals lasting less than 60 days were replaced by a new event representing an acute interval. Similar strategies were use to identify chronic episodes, and any remaining episodes were replaced as intermediate episodes. The resulting dataset consisted of only 180 visual elements, each averaging 1.96% of the total display height, an 83% reduction in visual elements and almost a 300% increase in the average element height (see Figure 8 - Steps 1 $\rightarrow$ 2).

Step 1: 1064 elements, .49% of display height per element
Step 2: 180 elements, 1.96% of display height per element

Using this simplified dataset, it was immediately apparent from the aggregated display that the majority of acute users had been classified correctly. These patients had only one, acute episode, and were easily distinguishable from the rest of the population that consisted of more varied episodes. Using selection, these patients were removed from the dataset, reducing the number of patients by over 60% in a single click (see Figure 8 - Steps 3). This filtering did not significantly reduce the number of visual elements, but increased the average height of the remaining elements to 3.04% of the total display height.

Step 3: 180 elements, 3.04% of display height per element

The researchers then narrowed the dataset down further, using query, to include only the patients who had consecutive acute episodes. From here, they could scroll over the time lapse between each pair of acute episodes to see the distribution of when each patient started their second episode in relation to ending their first. If these patients were exhibiting chronic behavior, the distribution would have been heavily front-weighted. That is, patients would be starting their second acute episode very soon after their first.

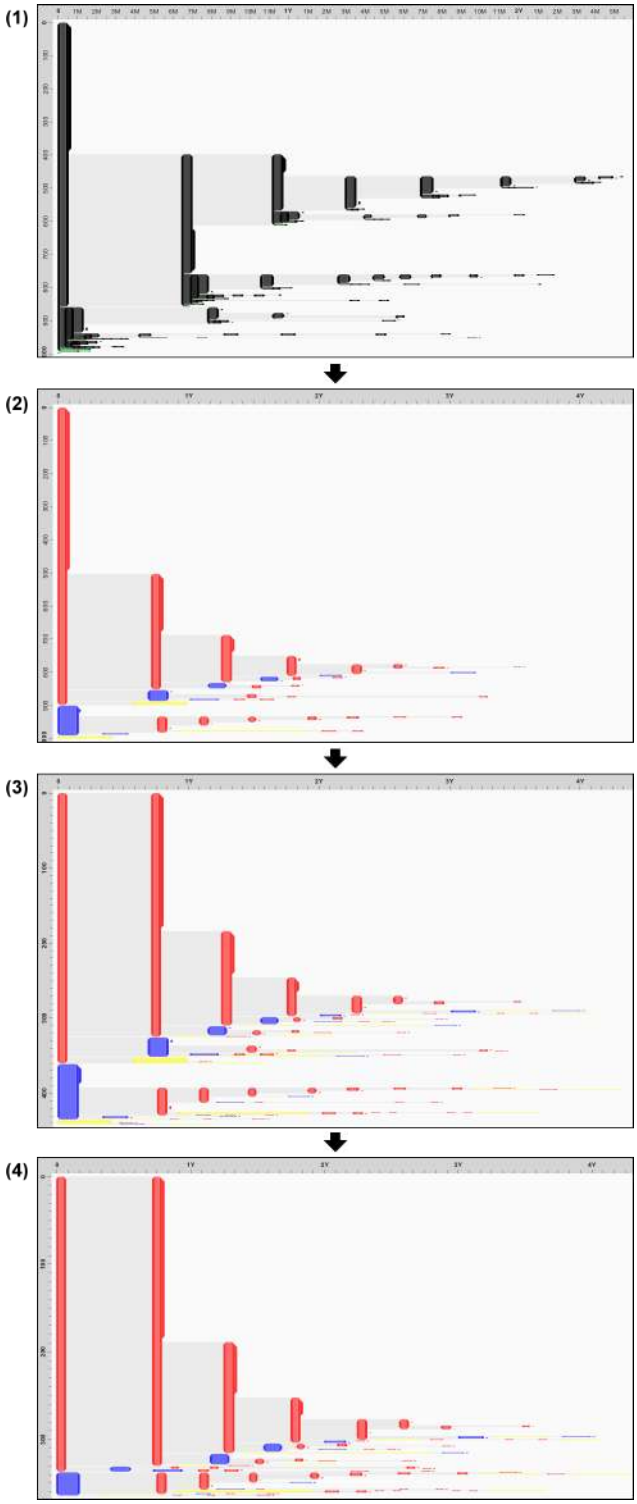

Fig. 8: Were chronic opioid users misclassified as acute users? - **Step 1:** The original Opioid dataset consists only of raw prescription data. **Step 2:** The dataset is restructured into acute (red), intermediate (in blue), and chronic episodes (in red). **Step 3:** Patients that only had one acute episode are removed. **Step 4:** Patients without consecutive acute episodes are removed.

Step 4: 156 elements, 3.79% of display height per element

However, this was not the case. The lapse of time between when patients started their second episode, relative to the end of their first episode, was uniformly distributed. This indicated that there was no

obvious reason to believe that a significant portion of these patients had been misclassified. The final consensus was that patients had been accurately classified as either acute or chronic users. Figure 8 summarizes the opioid simplifications.

8 Basketball Play-By-Play Breakdown

To demonstrate that EventFlow's simplification capabilities can be applied to datasets outside of the medical domain, we created a dataset (or, more accurately, two datasets) based on the play-by-play statistics from the University of Maryland men's basketball team. Play-by-play data consists of the events that took place over the course of a game (shots, rebounds, steals), timestamped against the game clock. The play-by-play breakdown of every college and professional basketball game is freely available online. For this study, we looked at the March 16th, 2013 game against the University of North Carolina (UNC), a game that Maryland ultimately lost by three points. The goal of this experimental analysis was to try to determine what went wrong.

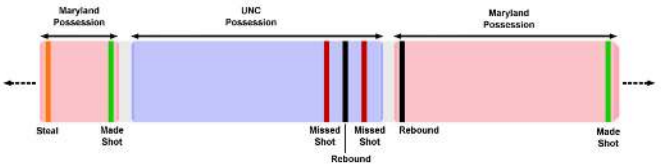

Fig. 9: Moving through time, left to right, the game can viewed as a series of alternating possessions (Maryland in red, UNC in blue.

A basketball game can be thought of as a series of disjoint, alternating possessions between the two teams (see Figure 9). The two phenomenon that we can look at, using EventFlow, is how events affect each other within possessions, and how events affect each other across possessions. The latter of these objectives comes in the form of two questions:

- How well does Maryland transition from offense to defense?
- How well does Maryland transition from defense to offense?

These questions were addressed by splitting the play-by-play data into two overlapping datasets. The first dataset takes the entire game of possession changes, and segments it into two-possession increments of a Maryland possession followed by a UNC possession. The second dataset does the same thing, but in increments of a UNC possession followed by a Maryland possession. In both datasets, each two-possession increment is treated as a record. Thus, the three-possession sequence shown above would be segmented such that the first and second possessions constituted one record in the Offense→Defense dataset, and the second and third possessions constituted one record in the Defense→Offense dataset.

From a simplification standpoint, the primary challenge of the play-by-play data was the granularity of the event categories. Nearly every category had sub-categories that could be useful for certain questions, but irrelevant for others. Shots could be 3-point jumpers, 2-point jumpers, or layups. Rebounds could be offensive or defensive. Timeouts could be called by either team or the officials. Without some degree of category merging, there would be too many categories, resulting in too many colors in the display for users to visually discern between.

Our strategy in analyzing this dataset then, was to begin by merging the event categories as much as possible, to produce the least visually complex display. The Offense→Defense dataset, loaded into EventFlow's aggregated display and aligned by the possession change, is shown in Figure 10. At this level of aggregation, it is easy to pick out the notable dynamics of the game, such as the events that caused the possession change, and the various scoring attempts. Essentially, the initial simplification strategy was to oversimplify.

From there, complexity could be reinserted only when it was needed for a particular question. This could be done by simply un-merging the meta-category, however, it frequently proved more effective to

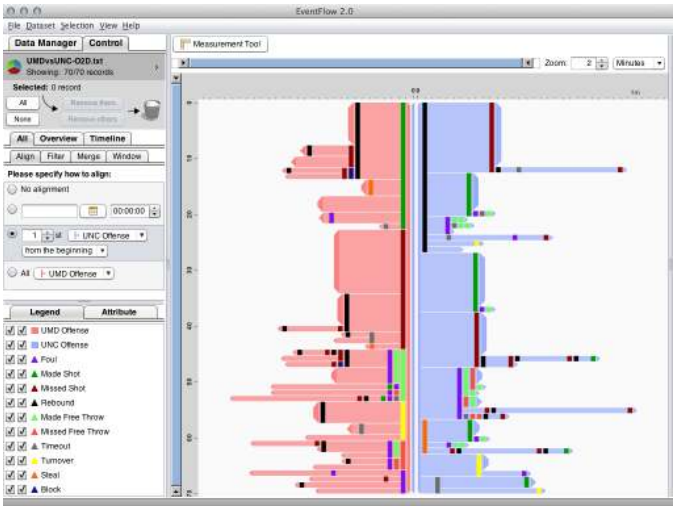

Fig. 10: The initial aggregated display of the Offense→Defense dataset. The data is aligned by the possession change, with the Maryland possession to the left of the alignment and the UNC possession to the right.

reinsert complexity using the Find & Replace interface, as it offered more fine-tuned control over how much complexity was being added. For example, offensive and defensive rebounds were merged into the meta-category, "Rebound." One of the questions we had of this dataset was, "Did UNC capitalize on their offensive rebounds?" This question could be answered by un-merging all of the rebound events across both possessions. However, since a defensive rebound results in an immediate possession change, we could use Find & Replace to isolate only the rebounds in which UNC had possession and maintained possession, and replace them as "Offensive Rebound" (see Figure 11).

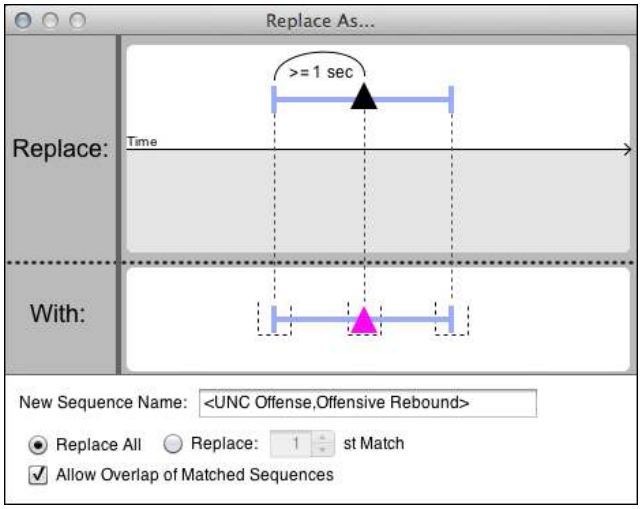

Fig. 11: Rebounds (in black) in which UNC had possession and maintained possession are replaced as "Offensive Rebounds" (in pink).

The big question remained then: "What could Maryland have done in order to win?" It's an interesting question, given that the final score differential was only one basket. In fact, using various different filtering, merging and un-merging, and Find & Replace, the two datasets produced almost identical aggregations of how each team performed. However, there was one glaring difference:

We can think of an offensive possession as starting either actively (via a steal or a rebound) or passively (via an opponent's score or a dead ball). In both datasets, we can isolate the passive possession changes by removing the possessions that began with either a steal or a rebound. The remaining possessions can be further simplified by removing dead ball events, such as timeouts or jump balls. Finally,

since the datasets are already filtered by the type of possession change, we can remove the preceding possession entirely (using Find & Replace). This process, being done to the Offense→Defense dataset, is shown in Figure 12.

Step 1: 183 elements, 5.32% of display height per element
Step 2: 106 elements, 9.26% of display height per element
Step 3: 92 elements, 10.46% of display height per element
Step 4: 39 elements, 9.98% of display height per element

Figure 13 shows both of the resulting simplifications. Both teams had roughly the same number of offensive possessions coming out of passive possession changes (Maryland - 38, UNC - 37). However, UNC was clearly executing their offense more efficiently. Their most common outcome was a score which, on average, occurred much faster than the first event in any of the Maryland possessions. In passive possession change scenarios, UNC scored on twice as many possessions as Maryland did. It would be reasonable then, to conjecture that the game was ultimately decided by UNC's ability to quickly execute their half court sets. Defensively, Maryland could have improved their chances by scouting these plays, and practicing to defend them at speed.

9 Conclusions and Future work

Event sequence simplification is critical to obtaining population-level overviews and more accurate representations of real-world events. This paper describes the design process for targeted simplifications that allow users to precisely and iteratively pare down complex temporal event datasets to the key visual elements that reveal meaningful patterns. Our work draws on techniques from both temporal event query and data mining, as well as countless hours with domain experts, working to understand how temporal relationships can be accessed and transformed within complex datasets.

We believe that these innovative simplifications could benefit many developers of temporal analysis systems as well as the researchers who use them. Working with 5 real-world user teams, these simplifications could reduce the visual complexity of initially overwhelming datasets by over 80%. This reduction allowed researchers to quickly and successfully generate and test hypotheses, as well as produce comprehensible figures for communicating their results. While validation with other datasets and in other domains is needed to further support this work, these simplifications appear to be a powerful and generalizable approach for solving problems with temporal datasets.

While EventFlow has been successful thus far in allowing users to eliminate complexity, it is important to remember that these capabilities make it equally possible for users to remove or obscure important features of their datasets. This can occur either accidentally or as a deliberate attempt to mislead others. To prevent this from occurring, we are working to better couple EventFlow's logging system with the data input files, so that new datasets cannot be generated without including a complete history of the modifications performed.

As mentioned previously, EventFlow currently allows users to save modified records as a new dataset. We are currently working to extend this capability by allowing users to save their progress on a given dataset, and then apply that progress to a new dataset. That is, EventFlow will allow users to save a series of simplifications as a stored procedure that can be applied to a new dataset before it is loaded into the visualization. Users can skip over the "confetti" stage of visual complexity and arrive immediately at a visualization from which they can glean meaningful insight. These stored procedures could be saved and shared, allowing the individual progress on one dataset to be applied to new datasets quickly and easily.

An additional benefit that will stem from our on-going work on stored procedure simplification, is how it can affect EventFlow's ability to scale to increasingly larger datasets. When a dataset is reduced by 80%, then intuitively, application should be able to load a simplified dataset that is 5 times larger. This could have a significant impact on the extremely large datasets that put a strain on memory usage and rendering times, and will be the focus of EventFlow's development moving forward.

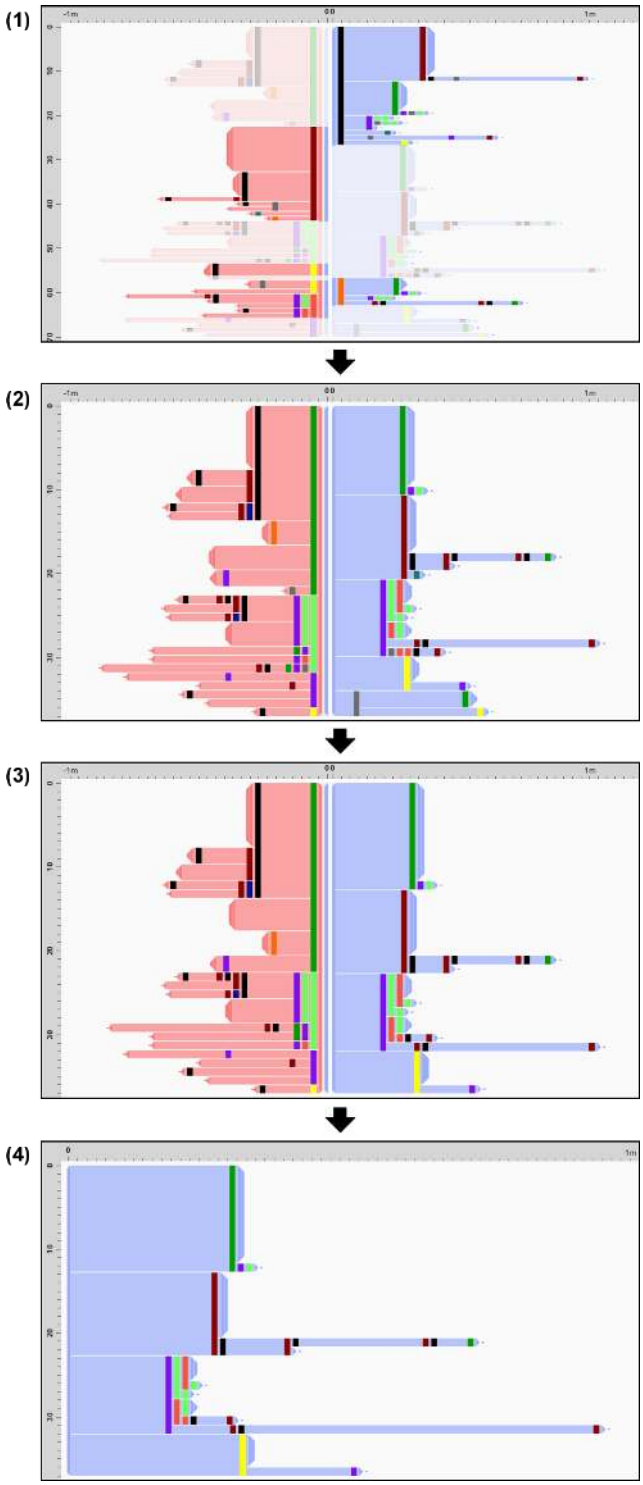

Fig. 12: How did the UNC offense perform off of passive transitions? - **Step 1:** Active possession changes are selected from the original Offense→Defense dataset based on rebounds (in black) and steals (in orange). **Step 2:** These possessions are removed. **Step 3:** Dead ball events are filtered out. **Step 4:** The Maryland possession is removed.

Acknowledgments

We appreciate the partial support of the Oracle Corporation and NIH-National Cancer Institute grant RC1-CA147489, Interactive Exploration of Temporal Patterns in Electronic Health Records. We would like to thank Jeff Millstein from Oracle and Seth Powsner from Yale

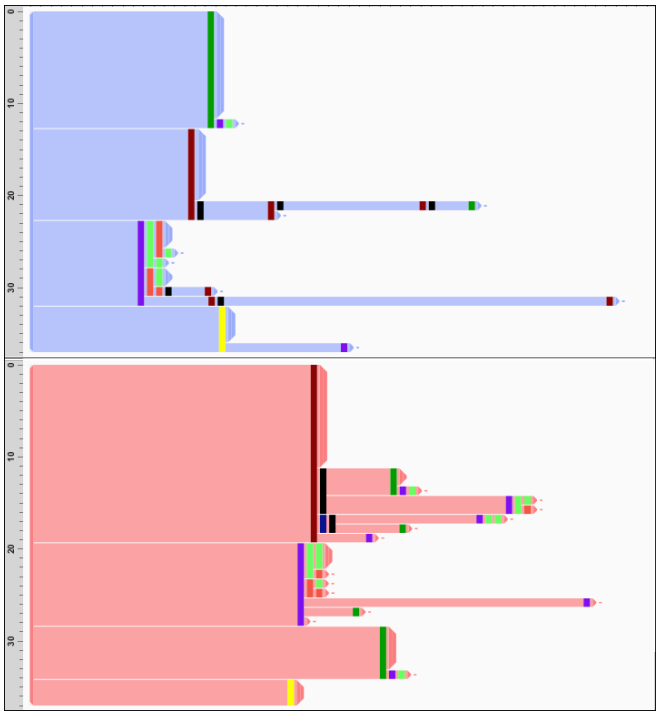

Fig. 13: The UNC offense, coming off of passive possession changes (top) vs. the Maryland offense, coming off of passive possession changes (bottom). UNC executed their offense visibly faster than Maryland (the horizontal axis is time), converting most of these possession into points (in green).

School of Medicine for their continued input and feedback. We appreciate the collaboration of researchers at the US Army Pharmacovigilance Center and the Univ. of Maryland Human-Computer Interaction Lab.

REFERENCES

[1] J. F. Allen and G. Ferguson. Actions and events in interval temporal logic. In *Journal of Logic and Computation*, volume 4, pages 531–579, 1994.

[2] BestPractice. *Asthma in adults*. http://bestpractice.bmj.com/best-practice/monograph/44.html, January 2013.

[3] CDC. *QuickStats: Number of Deaths From Poisoning, Drug Poisoning, and Drug Poisoning Involving Opioid Analgesics*. http://www.cdc.gov/mmwr/preview/mmwrhtml/mm6212a7.htm, March 2013.

[4] A. K. Das and M. A. Musen. A temporal query system for protocol-directed decision support. In *Methods of information in medicine*, volume 33, pages 358–70, 1994.

[5] D. C. Donderi. Visual complexity: A review. In *Psychological Bulletin*, volume 132, pages 73–97, 2006.

[6] C. Dunne and B. Shneiderman. Motif simplication: Improving network visualization readability with fan and parallel glyphs. In *Proceedings of the 2013 Annual Conference on Human Factors in Computing Systems (CHI'13)*, page to appear, 2013.

[7] S. G. Eick and A. F. Karr. Visual scalability. Technical Report 106, National Institute of Statistical Sciences, Research Triangle Park, NC, 2002.

[8] W. R. Garner. *The processing of information and structure.* Wiley, New York, NY, 1974.

[9] F. Haag, S. Lohmann, and T. Ertl. Simplifying filter/flow graphs by subgraph substitution. In *Visual Languages and Human-Centric Computing (VL/HCC)*, volume 20, pages 145–148. IEEE, 2012.

[10] C. S. Jensen, J. Cliord, and S. K. G. et al. The tsql benchmark. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, pages QQ1–QQ28, 1993.

[11] J. Jin and P. Szekely. Interactive querying of temporal data using a comic strip metaphor. In *2010 IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 163–170. IEEE, 2010.

[12] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck, and P. Buono. Research directions in data wrangling: visuatizations and transformations for usable and credible data. In *Information Visualization - Special issue on State of the Field and New Research Directions*, volume 10, pages 271–288, 2011.

[13] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11)*, pages 3363–3372. ACM, 2011.

[14] R. Lan, H. Lee, A. Fong, M. Monroe, C. Plaisant, and B. Shneiderman. Temporal search and replace: An interactive tool for the analysis of temporal event sequences. Technical Report HCIL-2013-TBD, HCIL, University of Maryland, College Park, Maryland, 2013.

[15] M. Lima. *Visual Complexity: Mapping Patterns of Information*. Princeton Architectural Press, 2011.

[16] F. Moerchen. Algorithms for time series knowledge mining. In *KDD '06 Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 668–673, 2006.

[17] M. Monroe, R. Lan, J. M. del Olmo, B. Shneiderman, C. Plaisant, and J. Millstein. The challenges of specifying intervals and absences in temporal queries: A graphical language approach. In *Proc. of ACM Conference on Human-Computer Interaction*, page TBD, 2013.

[18] M. Monroe, K. Wongsuphasawat, C. Plaisant, B. Shneiderman, J. Millstein, and S. Gold. Exploring point and interval event patterns: Display methods and interactive visual query. Technical Report HCIL-2012-06, HCIL, University of Maryland, College Park, Maryland, 2012.

[19] N. Mustafa, S. Krishnan, G. Varadhan, and S. Venkatasubramanian. Dynamic simplication and visualization of large maps. In *International Journal of Geographic Information Systems*, volume 20, page 273302, 2006.

[20] S. J. Nordb. Information visualisation and the electronic health record. In *Masters Thesis: Norwegian University of Science and Technology*, 2006.

[21] M. Sedlmair, M. D. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. In *IEEE Transactions on Visualization and Computer Graphics*, volume 18, pages 2431–2440, 2012.

[22] R. Snodgrass. The temporal query language tquel. In *ACM Transactions on Database Systems (TODS)*, volume 12, pages 247–298. ACM, 1987.

[23] J. Sun, F. Wang, J. Hu, and S. Edabollahi. Visual cluster analysis in support of clinical decision intelligence. In *AMIA Annual Symposium Proceedings*, page 481490, 2011.

[24] J. Sun, F. Wang, J. Hu, and S. Edabollahi. Supervised patient similarity measure of heterogeneous patient records. In *SIGKDD ACM Special Interest Group on Knowledge Discovery in Data*, volume 14, pages 16–24. ACM, 2012.

[25] E. Tufte. *The Visual Display of Quantitative Information*. Graphic Press, 1983.

[26] P. C. Vitz. Preference for different amounts of visual complexity. In *Behavioral Science*, volume 11, page 105114, 1966.

[27] K. Vrotsou, J. Johansson, and M. Cooper. Activitree: Interactive visual exploration of sequences in event-based data using graph similarity. In *IEEE Transactions on Visualization and Computer Graphics*, volume 15, pages 945–952, 2000.

[28] T. D. Wang. Interactive visualization techniques for searching temporal categorical data. In *Ph.D. Dissertation from the Department of Computer Science, University of Maryland*, 2010.

[29] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, B. Shneiderman, and S. Murphy. Aligning temporal data by sentinel events: Discovering patterns in electronic health records. In *Proc. of ACM Conference on Human Factors in Computing Systems*, pages 457–466, 2008.

[30] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2004.

[31] K. Wongsuphasawat, J. A. G. Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. Lifeflow: Visualizing an overview of event sequences. In *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems (CHI'11)*, pages 1747–1756. ACM, 2011.

[32] H. Wu, B. Salzberg, G. C. Sharp, S. B. Jiang, H. Shirato, and D. Kaeli. Subsequence matching on structured time series data. In *SIGMOD '05 Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 682–693. ACM, 2005.