TimeSets: Timeline Visualization with Set Relations

Journal Title
XX(X):1–16

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Abstract

In this paper, we introduce a novel timeline visualization technique, TimeSets, that helps make sense of complex temporal datasets by showing the set relationships among individual events. TimeSets visually groups events that share a topic, such as a place or a person, while preserving their temporal order. It dynamically adjusts the level of detail for each event to suit the amount of information and display estate. Various design options were explored to address issues such as one event belonging to multiple topics. A controlled experiment was conducted to evaluate its effectiveness by comparing it to the KelpFusion method. The results showed significant advantage in accuracy and user preference.

Keywords

Timeline, time-series data visualization, set visualisation

Introduction

A timeline is a sequence of events and is typically visualized by plotting them along a time axis at the instant at or interval over which they occur[2]. Timelines are applied in many domains including personal biographies[2], analytic provenance[3], medical records[4], music[5], and historical events[6]. Events in a timeline are commonly categorized into groups or *sets*. For example, academic publications usually belong to one or more disciplines; similarly, news articles fall into different categories such as politics and sports. Back in 1765, one of the oldest documented timelines produced by Joseph Priestley – the Chart of Biography[7] – already used set visualization. The timeline includes two thousand famous persons from 1200 BC to 1800 AD, and Priestley classifies them into six categories based on their most well-known achievement[8]. The timeline is divided into six horizontal bands, one for each category, to visualize the set relations.

Timeline visualizations typically use icons to indicate time-point events[9] and horizontal bars for interval ones[2]. These are usually accompanied by a short line of text describing the event. To show set relations, existing methods either color-code the icons or use different shapes[10]. The layout algorithm of these methods usually focuses on avoiding event text overlap only[9;10]. As a result, events in the same set are not always placed close to each other. This makes it difficult to follow them chronologically or have an overview of the distribution of events in a set. Another common approach is to visually connect events in the same set[11]. Such a method can introduce extra edges and crossings, which hampers the readability of the timeline.

There has been considerable work on set relationship visualization, which commonly uses closed contours as in Venn or Euler diagrams. Texture and color can be used to depict more complex set relations[12]. However, these cannot be applied to the set relationships in timelines because the event position along the time axis is fixed. Recently there have been a number of papers on visualizing the set relationships of data items with fixed locations. To connect same-set elements, Bubble Sets[13] draws an iso-contour surrounding them, and LineSets[14] uses a Bézier curve passing through all the elements. KelpFusion employs both lines and areas to connect elements, and has been shown to have a significant advantage in readability tasks when compared to Bubble Sets and LineSets[15]. However, directly applying KelpFusion to timeline set visualization will introduce extra line segments and edge-text crossings that may reduce readability.

In this paper, we propose a novel timeline visualization, *TimeSets*, that facilitates making sense of set relations among events in a timeline. It provides an overview of set distribution, helps identify the trend of a set, and makes it possible to compare sets over time. For example, in Figure 1, there are many more events related to “White House” compared to other topics. Among “Judges, Courts” events, those are related to “White House” are more than

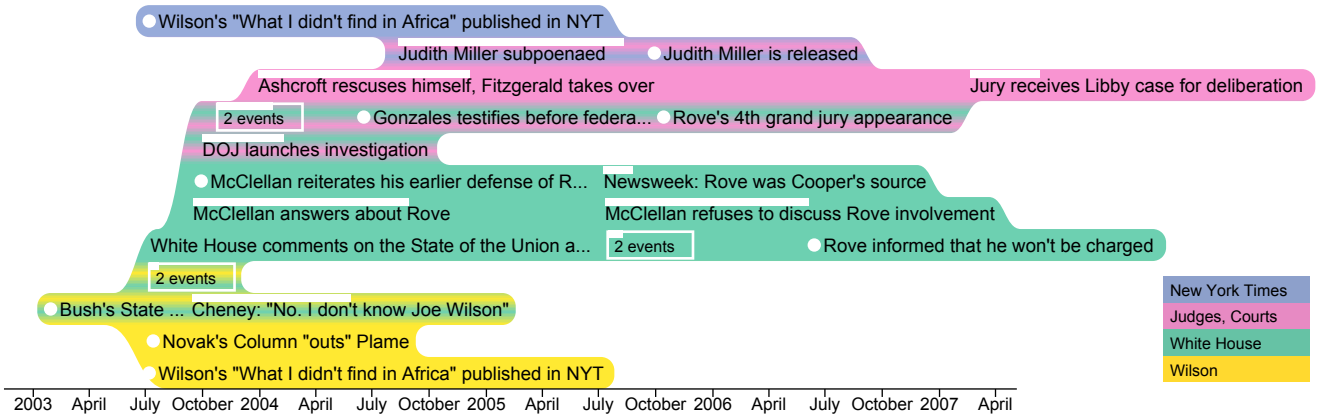


Figure 1. TimeSets visualization of the CIA leak case, in which the identity of CIA operative Valerie Plame was made public[1]. The timeline contains events that happened from 2002 to 2007, each has a timestamp or an interval, a label, and topics such as "White House". Events are positioned along the horizontal time axis based on timestamps, and vertically grouped by topics. A time-point event is shown with a white circle to its left, and an interval event with a horizontal bar on top showing its timespan. Each topic has a unique color (see the legend in the bottom right corner), and events shared by two topics have gradient backgrounds, transitioning between the colors of the two topics. The level of details of each event is adjusted dynamically based on the amount of information and available display space, resulting in three levels: full label, trimmed label (ending with "..."), and aggregated events (such as "2 events").

those related to "New York Times". Also, events about "Wilson" only appear at the beginning of the case, while "Judges, Courts" events appear later.

The design of TimeSets follows two Gestalt principles of grouping – *proximity* and *uniform connectedness*[16]. It places related events close together, and colors the set background to connect its events visually. More specifically, TimeSets:

- clearly shows the events within a set over time and their relationships with other sets;
- dynamically adjusts the level of details of each event to suit the amount of information and display estate;
- uses color gradient backgrounds for events belonging to multiple sets and curved set outlines to emphasize its grouping.

To show possible applications of TimeSets, we discuss a case study with publication data. Also, a controlled experiment was conducted to evaluate its effectiveness. The results showed that TimeSets was significantly more accurate than KelpFusion[15] – a state-of-the-art set visualization method), and was the preferred choice by the participants for aesthetics.

Related Work

Timeline Visualizations

The most common form of timeline visualization uses a horizontal axis to represent time progressing from left to right, with events positioned horizontally according to their timestamps. A well known example is LifeLines[4] – a visualization of personal medical records. LifeLines uses icons to indicate discrete events and thick horizontal lines for continuous ones. Timelines can be integrated into a tree format to represent changes in a hierarchy over time as in TimeTree[17]. Geographical information can also be embedded in timelines as in the classic visualization of Napoleon's March in Moscow in 1812–1813 by Charles Joseph Minard[18]. The book by Aigner et al.[19] provides a comprehensive review of timelines and other time-oriented data visualizations.

Techniques such as aggregation and interaction are commonly used when there are a large number of events. LifeLines[4] aggregates events to save display estate; for example, a series of similar prescriptions can be grouped together. ThemeRiver[20] or Streamgraph[21] uses a river metaphor to represent aggregated changes of themes over time in a large document collection. Each river is a theme, and its width at certain time points shows the number of documents in that theme. Common interaction techniques are often used in the visualization of large timelines to support their exploration, including overview+detail[5], filtering[2], and details-on-demand[22].

Set Relations in Timelines

According to the Gestalt principles of grouping, humans naturally perceive objects as a whole rather than as the sum of their parts[16]. Three of the principles are commonly used to show set relationships among events: similarity, proximity, and uniform connectedness.

The principle of *similarity* states that objects are perceptually grouped together if they are similar to each other[16]. This principle is extensively applied to show set

relations in timelines by using colors and shapes. Time indicators as icons (time-point events)[9] and bars (interval events)[23] are colored according to event set memberships. Different shapes for icons[10] and bars[4] are also used to distinguish set memberships. It is more challenging to represent multiple set memberships. LineSets[14] uses concentric circles for icons, where each circle is colored to represent one set.

According to the *proximity* principle, objects that are close together are perceived to be more related than objects that are spaced further apart[16]. In Chart of Biography[7], people within a category are placed in a horizontal band, away from people in other categories. LifeLines[4] splits medical records into different sets, such as *medication* or *diagnosis*, and places them into vertically stacks, which works well if no two sets overlap. Storyline visualizations[24;25] use curved lines to show interactions among characters within the movie timeline. Character lines converge to a bundle if they appear in the same interaction, and diverge when the it ends. Each line can be considered as a set passing through all of its members, and each interaction is a multi-set event. Thus, this method only works for interval events.

Elements tend to be grouped together if they are visually connected[26]. Following this *uniform connectedness* principle, SchemaLine[27] draws a rectilinear path connecting events belonging to a same set together. Also, tmViewer[11] links related entities with line segments. Different line colors, thicknesses, and styles were used to distinguish set relations. This method can show events with multiple set memberships by connecting them with multiple edges. However, extra edges and crossings may negatively impact the readability of the timeline.

When similarity and proximity are applied together, the later principle dominates[12]. Moreover, uniform connectedness is stronger than proximity[26]. For example, objects with different colors and shapes but located close together are more likely to be perceived as a group, and distant objects but with a closed contour surrounding them also provide a strong sense of grouping. Applying these ideas to visualize set relations for timelines, methods relying on similarity such as colored icons[23] are less effective than spatial grouping methods such as LifeLines[2]. And those, in turn, are less effective than methods using line segments such as tmViewer[11].

Set Visualizations

Sets and their relationships can be visualized using Venn[28] or Euler[29] diagrams. Simonetto et al.[30] proposed a technique to automatically visualize sets that were previously not possible with Euler diagrams. However, the complex shapes it produces may reduce visualization readability. In their controlled study, Riche and Dwyer[31] showed that for complex set intersections, duplications of shared elements resulted in a better performance in readability tasks than a none-duplicated visualization with more complex shapes.

These methods assume the positions of set elements are not fixed, which reduces their applicability for geo-located or timeline events. Techniques without such constraints include Bubble Sets[13], LineSets[14], and KelpFusion[15]. These methods employ the connectedness principle of the Gestalt laws[26] by connecting set elements using extra visual elements. Bubble Sets draws an iso-contour surrounding elements within a set. This iso-contour is filled with a semi-transparent color so that the intersection between sets is shown as an area of blended color. Collins et al.[13] provided an example of applying Bubble Sets to a timeline, in which case a force-directed algorithm is used to adjust the vertical positions of elements while the horizontal position along the time axis is fixed.

LineSets applies a Bézier curve to connect data items. The curve follows the shortest path passing through all elements in the set. Its study showed that LineSets outperforms Bubble Sets in certain readability tasks [14]. KelpFusion, a hybrid technique, uses lines for data-sparse areas and surfaces for data-dense areas. The results of an evaluation on readability tasks[15] demonstrated that it outperforms Bubble Sets in both accuracy and completion time, and outperforms LineSets in completion time. There has been no reported attempt to apply LineSets or KelpFusion to timeline visualizations. It is expected that crossings between lines, areas and the event text may reduce the timeline readability.

TimeSets Visualization

Events

In TimeSets, an event is visualized as a line of text, showing its *label*. The visual indicator of event time is a circle (for a time-point event), or a horizontal bar (for an interval event). The *time circle* is shown to the left of the label, while the *time bar* is shown above the label. The time bar is semi-transparent for overlapping interval events, so the intersection part is visually different. To accommodate a large number of events, labels have three possible levels of detail:

- Complete: the entire event label is shown.
- Trimmed: only the first few words are shown, followed by three dots at the end.

- Aggregated: a few events are combined into a new one with its label indicating the number of events, such as "2 events".

The text border of an aggregated event is colored to make it visually different from non-aggregated events. Its time bar begins at the starting time of the earliest event, and ends at the finishing time of the latest event within the aggregate. Figure 2 shows a complete time-point, a trimmed time-point, an interval, an aggregate of two events, and two overlapping interval events.

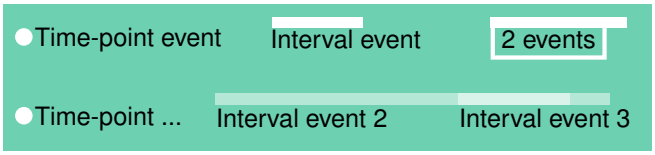


Figure 2. Visual representations of complete and trimmed time-point events, interval, aggregated, and overlapping events.

Sets

Design Overview As discussed in the related work, Gestalt principles of grouping are commonly used to show set relationships among events, most effectively uniform connectedness and proximity. Therefore, we also apply these two principles in our design. Proximity is achieved by moving same-set events close together, and coloring the set background makes the events visually connected.

Because the horizontal position of each event is decided by its timestamp, spatial grouping is achieved through vertical positioning. The sets are stacked vertically, and each set is further divided into a maximum of three horizontal *layers*: a top and a bottom layer for events shared with the set above and below respectively (if they exist), and a middle layer for other events in the set. There are maximal $2n-1$ layers in total for n sets of events. Figure 3 shows the layering for three sets.

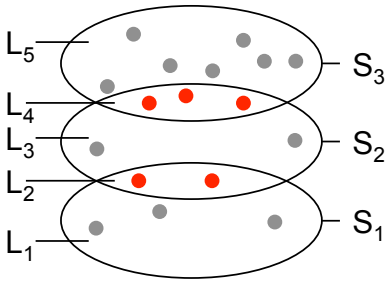
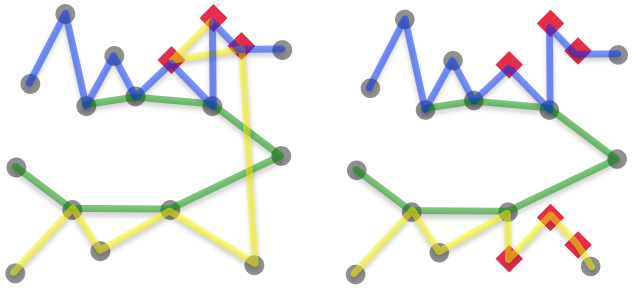


Figure 3. Layering for three sets S_1, S_2, and S_3. L_2 consists of events shared by S_1 and S_2, and L_4 consists of events shared by S_2 and S_3. Those shared events are red circles.

Shared events between two non-neighboring sets can reside in one set and connect to the other set using visual links such as curves [14] or areas [15]. Figure 4a shows a possible method of connecting the shared events (red squares) using edges and linking them to the yellow set to indicate that they also belong to that set. The alternative approach is to duplicate them in both sets. In Figure 4b, red squares are duplicated in both the blue and yellow sets. Duplication consumes more display space and could make viewers confuse when seeing same events multiple times. However, duplication allows all events of a same set being placed close together, which provides a compact visualization and easy comparison. Also, the study by Henry Riche and Dwyer [31] shows that complex set intersection shapes reduce readability compared to item duplication. Aiming for a clear visualization, we decided to duplicate events that belong to non-neighboring sets. Confused duplication and scalability will be addressed later using interaction and layout algorithm respectively.



(a) Use visual links to connect shared events from one set to another set.

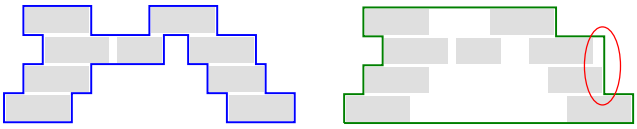
(b) Duplicate shared events in both sets.

Figure 4. Visualizing shared events (red squares) between two non-neighboring sets.

In subsequent sections, we discuss the detail of the set visualization algorithm, which consists of two main steps: the generation of set shapes, and then their coloring.

Shape Generation This algorithm takes as input a list of bounding-boxes of the set's events, and generates a closed-curve containing all these rectangles. The sizes and positions of the bounding boxes are decided by the layout algorithm described in the next section. A rectilinear shape can be generated using the scan-line algorithm [32], as shown in Figure 5a. The number of bends along the line is often used to assess the aesthetics and legibility of visualizations [24]. Even though the generated shape provides the minimal *data-ink* ratio [33], a large number of line bends may reduce its legibility.

To reduce the number of line bends, the top and the bottom sides of the set outline are flattened. The left and right sides are kept unchanged because they indicate the event timespan. On either side, the path can be "jagged" if two events start or end close to each other. Those close vertical segments



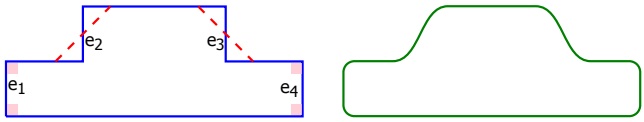
(a) The rectilinear shape, generated by a scan-line algorithm.

(b) The simplified shape by flattening and removing jags (red eclipse).

Figure 5. Rectilinear shape generation.

are combined to reduce line bends if their horizontal gap is smaller than a threshold. This trades off time accuracy for outline smoothness and can be controlled by the user. Figure 5b shows the result of this simplification.

To reduce the degree of line bends, the algorithm converts vertical segments to diagonal ones wherever possible, such as e_2 and e_3 in Figure 6a. Smoother lines are easier to follow[34], thus the algorithm further converts diagonal segments to Bézier curves and replaces right corners with quadrant arcs as in Figure 6b.



(a) Vertical segments e_2 and e_3 are converted to diagonal ones (dashed lines).

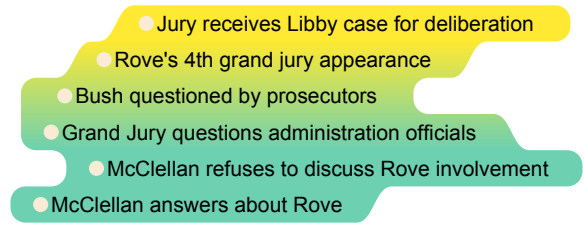
(b) Right corners are replaced by quadrants. e_2 and e_3 are smoothened by Bézier curves.

Figure 6. Shape smoothening by reducing the degree of line bends.

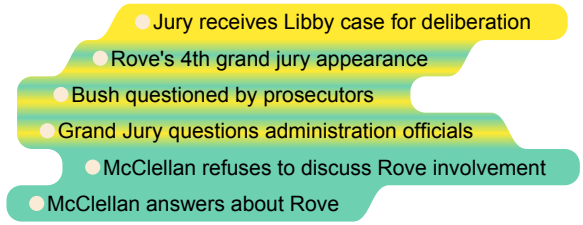
Coloring Each set is filled with a color selected from Qualitative Set 2 of ColorBrewer[35] to make them easily distinguishable. Two color filling options are considered: only the time circle or the entire event label. Our design follows uniform connectedness principle requiring visual connection among same-set events. When they are visually connected and only their time circles are filled, additional edges may reduce the readability. KelpFusion[15] follows this approach using lines and areas to connect time circles reducing its readability as in Figure 14b. In the second option, filling the entire label may produce a false impression about the event's time range. We choose this option and lessen the effect by coloring the gap between events as in Figure 14a. It also helps increase the sense of grouping compared to filling only the time circles.

The standard coloring method for set intersections is color blending as commonly used for Venn diagrams[12]. Color for each set is half-transparent, and alpha blending is applied to produce the new color for the intersection. However, the result may be irrelevant to the two inputs and confused as the color for a new set. For instance, in Figure 3, the yellow color of layer L_2 is not naturally considered as the common between light yellow of set S_1 and light green of set S_2.

To address this issue, we fill the intersection with a linear color gradient changing between the two set colors as in Figure 7a. While the gradient provides a smooth transition, it becomes difficult to recognize the two ends of the intersection. For example, it is not clear from Figure 7a that the background of the event *Rove's 4th grand jury appearance* (2nd row top down) is pure yellow or it has a mix of green as well. To solve this problem, multiple color transitions are used instead of a single transition. For instance, in Figure 7b, the color transitions between green and yellow are repeated multiple times so that both colors are clearly shown in every row of the intersection, and there is no significant difference in color perception among these rows.



(a) Intersection shown as a single color gradient.

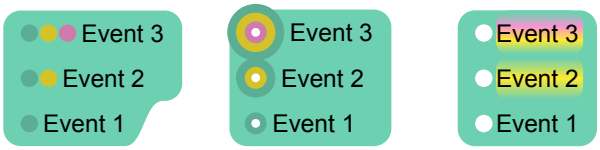


(b) Intersection shown as multiple color gradients.

Figure 7. Color gradient technique to encode set memberships. The gradient area shows three shared events between two sets.

Multiple-set Events Visualization With the vertical layering of sets as discussed above, three sets cannot be placed closed to each other; therefore, it is impossible to visualize intersections among three sets or more. This is also a challenging problem with many state-of-the-art methods[36]. To address this issue, similar to non-neighboring sets, we replicate events for each set they belong to so that all events in a same set stay close together providing a compact visualization and easy comparison. To provide full set memberships of events, one method is drawing edges to connect all replicates of the same event together. However, this may produce a cluttered visualization with many edge crossings. Another method is to color code the event according to its set memberships. One approach is to

color the event's time icon preceding its label using either multiple circles (Figure 8a) or concentric rings (Figure 8b). The former requires more horizontal space, while the latter needs more vertical space. Another approach is to color the background of the event's label. Color gradient is used for a smooth color transition as in Figure 8c. This visual encoding is consistent with the use of color gradient to show 2-set intersections. However, a timeline with many long-label events may produce a too colorful and distracted visualization. Also, limited label height may hamper the detection of color transition. To solve these problems, color is transitioned from left to right, and only run through a first few characters of the event label. Figure 13 shows this technique in a visualization with 200 events.



(a) Each circle represents a set.

(b) Each ring represents a set.

(c) Each color in the gradient background represents a set.

Figure 8. Multiple-set events visualization. Event 1 is single-set. Event 2 is double-set. Event 3 is triple-set.

For interval events, only the label background method can work because it does not have time circles, which can be added but at the cost of extra display space. Time bars can be used to show set memberships by dividing into multiple horizontal parts, each color for one set. However, this could be mis-interpreted as an event having different set membership in each part of its timespan.

Interaction

Interactive features are implemented to support timeline exploration. Details-on-demand provides related information without information overload. Mouse-over an event reveals its starting/ending time and complete label. When none of the multiple-set visualization techniques proposed above is used to statically display the full set memberships of an event, it is possible to use interaction to reveal that information. When an event is hovered, all of its replicates are highlighted, thus allows detecting all sets it belongs to. This method prevents adding an extra ink to the visualization; however, it requires users to discover the set information manually.

TimeSets provides interactive set filtering and time range navigation (zooming and panning). Clicking on a set in the legend (bottom-right corner in Figure 1) toggles its visibility. Time range zooming is performed using the mouse-wheel and the panning is controlled by dragging. Users can also interactively modify set ordering by changing the order in the legend through drag-and-drop. A smooth animated transition is provided for all the interactions to help users maintain their mental map[37]. A demonstration of these interactive features can be found in the supplemental video.

TimeSets Layout

The layout algorithm, determining the positions of sets and events within them, consists of four steps. First, the vertical ordering of sets is computed to ensure that two sets that share events are next to each other wherever possible. Then, sets are further divided into layers, and events are assigned to them according to their memberships. After that, the position and length of each event are computed, within the given display space. Finally, layers are compacted to remove any gaps between them, before layer sizes are adjusted to allow for a consistent level of detail across all sets.

Sets Ordering

The set ordering algorithm aims to maximize the number of events shared by neighboring sets. This can be mapped to a graph path problem. Given a list of sets $S = \{s_1, s_2, \ldots, s_n\}$, an undirected graph $G = (V, E)$ is created with each vertex v_i represents a set $s_i \in S$. Two vertices v_i and v_j are connected if s_i and s_j share an event. The weight of edge e_{ij} is the number of events shared by s_i and s_j. Finding a set order with the maximum number of events shared by neighboring sets is equivalent to finding a path with the maximum weight connecting all vertices in G. This longest path problem is known to be NP-hard, but the number of sets we plan to support is limited by the number of colors that human can easily distinguish when they are shown together. Therefore, we decided on a brute-force approach to find the optimal solution.

Layer Layout

This algorithm positions all the events within a layer. Its inputs are:

- The events belonging to the layer with their *label* and *time*.
- The maximum width and height of the layer.

The outputs are event locations within the constrained display area, optimizing for the following criteria:

Completeness, which measures how much event label is visible. More specifically, the *completeness ratio* is

defined as: $\theta = \frac{\alpha \cdot |E_c| + \beta \cdot |E_t|}{|E|}$, where $|E_c|$ is the number of complete events, $|E_t|$ is the number of trimmed events, and $|E|$ is the number of all events. α and β are the coefficients to indicate how strongly a complete event and a trimmed event contributes to the overall content richness of the layer. We practically set $\alpha = 1$ and $\beta = 0.5$.

Traceability, which measures how easy it is to follow the events chronologically. Events happened close in time should have small changes in their row levels to facilitate the tracing of them. More specifically, the *traceability ratio* is defined as: $\gamma = \frac{\sum_{i=1}^{|E|} (|l_{i+1} - l_i|)}{|E| - 1}$, where $|E|$ is the number of all events within the layer and l_i is the row level of event e_i.

The horizontal position of an event is fixed by its time. The layout algorithm decides on which row to position an event and the level of detail for its label.

Completeness Algorithm Starting with an empty layer, the algorithm inserts events chronologically. An event is placed in the lowest possible row where it does not overlap with any other events. If such a row does not exist, one of the earlier events is trimmed to make space for it. Among these events, the one with the least trimming is selected. However, if the label space is too short for a single word after trimming, the event will be combined with the current event to form a new aggregated event titled "2 events". Aggregated events cannot be trimmed, thus a new event overlapping with them should also be aggregated. For example, a new event that overlaps with a "2 events" aggregated event will be aggregated, resulting in a "3 events" aggregated event. The completeness algorithm maximizes the number of complete events $|E_c|$ and trimmed events $|E_t|$, thus yielding a maximum completeness. This algorithm is not optimized for traceability because an event is placed in the lowest possible row disregarding the row level of its preceding event.

Traceability Algorithm To improve traceability, this algorithm inserts a new event at the same row as its previous event. If there is an overlap, the previous event is trimmed to make space. The *trim ratio* of an event is defined as the ratio of the remaining text length to its original length. An event can only be trimmed if the resulting trim ratio is greater than a minimum threshold t_{min}, where $0 \leq t_{min} \leq 1$. This value determines how much completeness can be traded for traceability. If the resulting trim ratio is smaller than t_{min}, the event moves up or down to find a suitable row, up to r_{max} rows on both sides. This value decides how far an event can be, in terms of row level difference, from the previous event, which essentially trades traceability for completeness. If no suitable row can be found within the +/- r_{max} rows, the new event comes back to the level of its proceeding event, which is then trimmed or aggregated with the new event as in the completeness algorithm. Figure 9 shows an example of these two algorithms used to layout seven events.

(a) Completeness algorithm: $\theta = 1, \gamma = 5/3$.

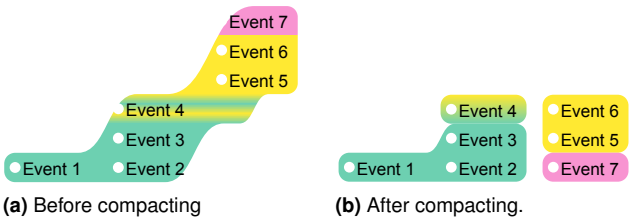
(b) Traceability algorithm ($t_{min} = 0.5, r_{max} = 1$): $\theta = 6/7, \gamma = 2/3$.

Figure 9. Two layout algorithms. Each rectangle is the bounding box of an event with its label.

Both layer layout algorithms run in linear time in terms of the number of events, because during the event insertion, the completeness algorithm checks up to a constant value – the layer height, and the traceability algorithm checks at most $(2 \times r_{max} + 1)$ rows.

Compacting

The aforementioned layer layout needs a maximum number of rows it can use as an input. Initially, it is assigned proportionally to the number of events within each layer. After the layout of each layer is independently computed, some may be moved to fill the new space that appears between layers. This includes moving two layers closer if there is a gap in between, or moving a layer into a newly created space if its set does not share events with any other sets. Figure 10 shows an example of compacting. The freed space is assigned to the layer with the lowest completeness ratio θ. Then, layouts of all layers are recomputed and compacted again. The process repeats until no more space can be saved.



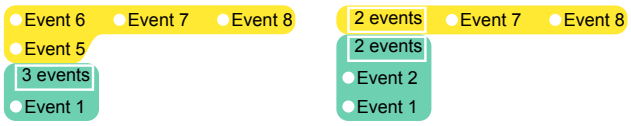
(a) Before compacting

(b) After compacting.

Figure 10. Layer compacting example.

Balancing

This last step ensures that all the layers have similar levels of detail, i.e., it minimizes the variance of completeness, $\frac{\sum_{i=1}^{n}(\theta_i-\bar{\theta})^2}{n}$, where n is the number of layers and $\bar{\theta}$ is the mean of completeness ratio θ. A brute force approach tests all possible combinations of layer height h_i such that $\sum_{i=1}^{n} h_i = H$ for a minimum variance. However, the number of combinations is an exponential of n. Instead, we used a heuristic algorithm that relies on the fact that completeness ratio increases with layer height because there is more space to display labels. The algorithm reduces the completeness ratio variance by iteratively taking a row from the layer with the largest ratio and giving it to the layer with the smallest one, until the variance no longer decreases. Figure 11 shows an example of balancing.



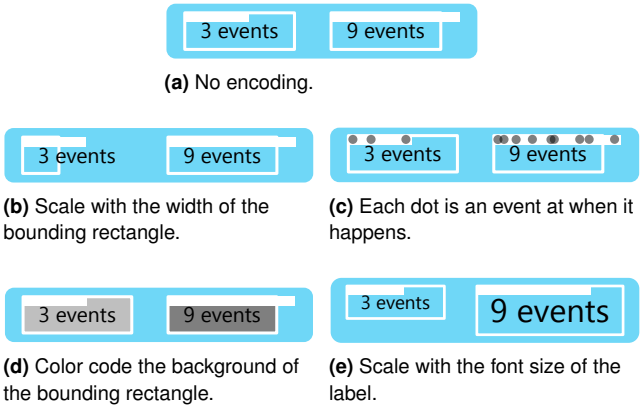
(a) Before balancing: $\theta_{green} = 0.25, \theta_{yellow} = 1.$

(b) After balancing: $\theta_{green} = \theta_{yellow} = 0.5.$

Figure 11. Layer balancing example.

Scalability

Aggregation allows TimeSets to handle a large number of events. However, the visual encoding of aggregated events is imperfect: "2 events" and "100 events" are visually represented in the same way with rectangular border. Four options are considered to address this issue and illustrated in Figure 12. First, the width of the bounding rectangle can be scaled to indicate the number of events (Figure 12b). However, the visual difference could be subtle and difficult to attract attention from an overview. The second option is to plot each individual event as a dot at when it happens (Figure 12c). This also provides a temporal distribution of events rather than just the quantity. When events happen at close or exactly the same time, dots are overlapped, and it is more difficult to see the pattern. Another approach is to color code the background of the bounding rectangle using luminance or intensity (Figure 12d). However, when many aggregated events are displayed, their backgrounds could interfere with the set colors. Last is to scale the font size of the label according to the number of events (Figure 12e). Currently, each event is completely placed in one single row with uniform height. Scaling the height of aggregated events needs to revise the layout algorithm. All these methods have their strength and limitation, thus deciding the best one is challenging and out of scope for this paper. We leave the implementation and evaluation of these methods as future work.



(a) No encoding.

(b) Scale with the width of the bounding rectangle.

(c) Each dot is an event at when it happens.

(d) Color code the background of the bounding rectangle.

(e) Scale with the font size of the label.

Figure 12. Different visual representations of the number of events in an aggregate.

The existing layout is suitable for a small timeline with a few hundreds of events or a detailed view where individual events are of high importance. Figure 13 shows 225 publications within 15 years. TimeSets relies on color to distinguish sets, therefore it is constrained by the number of colors that human can differentiate at the same time, which is about 12, according to Munzner's book [38].

Case Study

TimeSets can be applied to domains requiring the understanding of temporal events including history, movies, publications, etc. Figure 1 shows a timeline of the CIA leak case [1] covering both time-point and interval events happening from 2002 to 2007. In this section, we choose another domain, academic publications, to demonstrate TimeSets. A subset of 200 publications with the most citations is extracted from the IEEE InfoVis articles. Each publication is assigned one or many *concepts* such as *network* or *evaluation*. We use concept as the set attribute to group publications. Figure 13 shows the visualization of this dataset. No aggregation is needed when producing the layout, only complete and trimmed labels are displayed in the visualization.

TimeSets can show distribution of categorical data over time as in ThemeRiver [20]. A quick glance at the visualization brings us a surprise. There is much void space on the left as opposed to a very dense area on the right indicating that there are many more highly cited papers published in the last ten years of the dataset than in the first ten years. This trend also holds for individual concepts. Each colored band starts with a single row and increases its height towards the end of the timeline. This observation is in contrast with the common thought: papers published in a longer time would

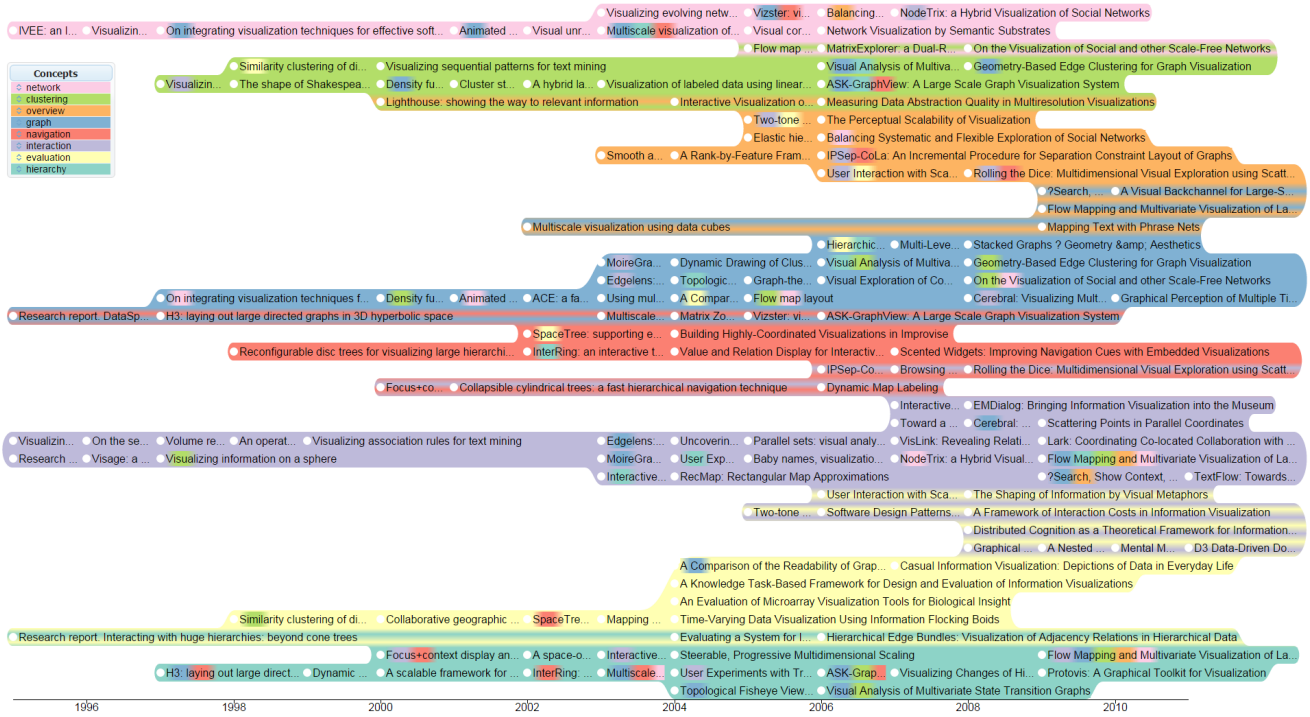


Figure 13. TimeSets visualization of 200 publications with the most citations in the IEEE InfoVis conference from 1995 to 2013. Concepts are used to group publications and only eight concepts appearing most in those publications are shown (see the legend in the top left corner).

have more citations. One possible explanation is that the IEEE InfoVis conference accepts more papers over time – in the dataset, there are 18 articles in 1995 while 37 articles in 2013. Another reason could be that publications in the last ten years are really of high quality.

TimeSets cannot show all intersections among sets; however, its layout maximizes the number of shared elements between two neighboring sets. As a result, the visible intersections usually have the most elements among all intersection. In the visualization, the most notable gradient area is the intersection between yellow and purple sets implying that there are many excellent papers focusing on both *evaluation* and *interaction*. Another observation at the top of the visualization with three concepts: *networking*, *clustering* and *overview* with clustering is in between the other two. This is expected because clustering techniques are important in visualizing large networks or getting the overview of a large dataset.

In this figure, TimeSets uses the color gradient method to show full memberships of multi-set elements. For instance, inside the pink band, there are quite a few small blue gradients for *network* papers has many blue small gradients for *graph* papers. This is sensible because the closeness between these two concepts and they may often appear together in a paper. Another interesting observation is at the bottom band – *hierarchy*. The last paper "Flow Mapping and Multivariate Visualization..." includes five concepts: *hierarchy* (blue background) and *interaction*, *graph*, *overview* and *network* (small gradients).

Evaluation

Method

We considered timeline visualizations that apply the two most powerful Gestalt principles of grouping to include in the evaluation. For *proximity* principle, as discussed in the related work, LifeLines[2] cannot show multi-set events, and storyline visualizations[24;25] only work for interval multi-set events. For *uniform connectedness* principle, methods that connect all events belonging to the same set together without using a designated layout to reduce edge crossings such as tmViewer[11] produce a cluttered visualization. To the best of our knowledge, there is no existing visualization that is designed to show multi-set relations and temporal information of events together. Therefore, rather than evaluating both the layout and the set visualization technique of TimeSets, we decided to focus only on the second one. We compared TimeSets with a set visualization technique that can apply on top of an existing timeline. We chose KelpFusion because among similar techniques, it has been shown to have the best performance in readability tasks in both accuracy and completion time[15]. We acknowledged

that KelpFusion was not specifically designed to work with timelines. However, KelpFusion can work with any given layouts, and it is the best choice for this evaluation. We conducted a controlled experiment to compare TimeSets and KelpFusion. It followed a within-subject design; and accuracy, time and user preference were collected.

Datasets We used generated data for the experiment to remove the possibility that participants might be distracted by their existing knowledge of scenarios. Only time-point events are used because KelpFusion needs a set of points as its input. The complexity of dataset was controlled by two parameters: the number of sets and the average number of events per set. Overall, half of the events were part of more than one set; this is the same ratio as in the CIA Leak case dataset. The details of the four levels of complexity used in the experiment are shown in Table 1.

Table 1. Data Set Statistics

Complexity	# sets	# events	# intersections
Level 1	3	30	15
Level 2	3	45	23
Level 3	5	50	25
Level 4	5	75	38

We introduce three approaches to visualize intersections between more than two sets; however, evaluating all of them would triple the number of trials and make the experiment too long. We plan a separate experiment to study which method is the most effective as our future work. In this experiment, we only tested two-sets intersections and simple white circles are used for events' time indicators.

Images of this dataset using the KelpFusion method were generously provided by the method's author. To avoid bias, our method also used static images instead of interactive visualizations. Colors for both methods were Qualitative Set 2 of ColorBrewer[35]. KelpFusion does not have its own layout; therefore, our layout algorithm was used for both settings. Only one algorithm is used to prevent adding another factor to the experiment, which doubles the number of trials for participants. The traceability algorithm was chosen because reading comprehension is not required for the tasks. Figure 14 shows example images used in the experiment.

Tasks We followed the task design in the KelpFusion technique evaluation[15], including estimation and precise comparison of set sizes, and counting the number of elements in a set. Two time-related tasks were added to evaluate the temporal aspect of the visualization. Therefore, there are five tasks in total. We considered three categories of set readability tasks, relating to: the *set* itself, the *intersection* of two sets, and the *difference* between two sets. However, it was impractical to include all 5×3 task types in the experiment. Therefore, we decided to use two tasks for the set only, two tasks for the intersection, and one task for the difference. Tasks together with examples are listed in Table 2. Each participant would complete a total of 40 questions.

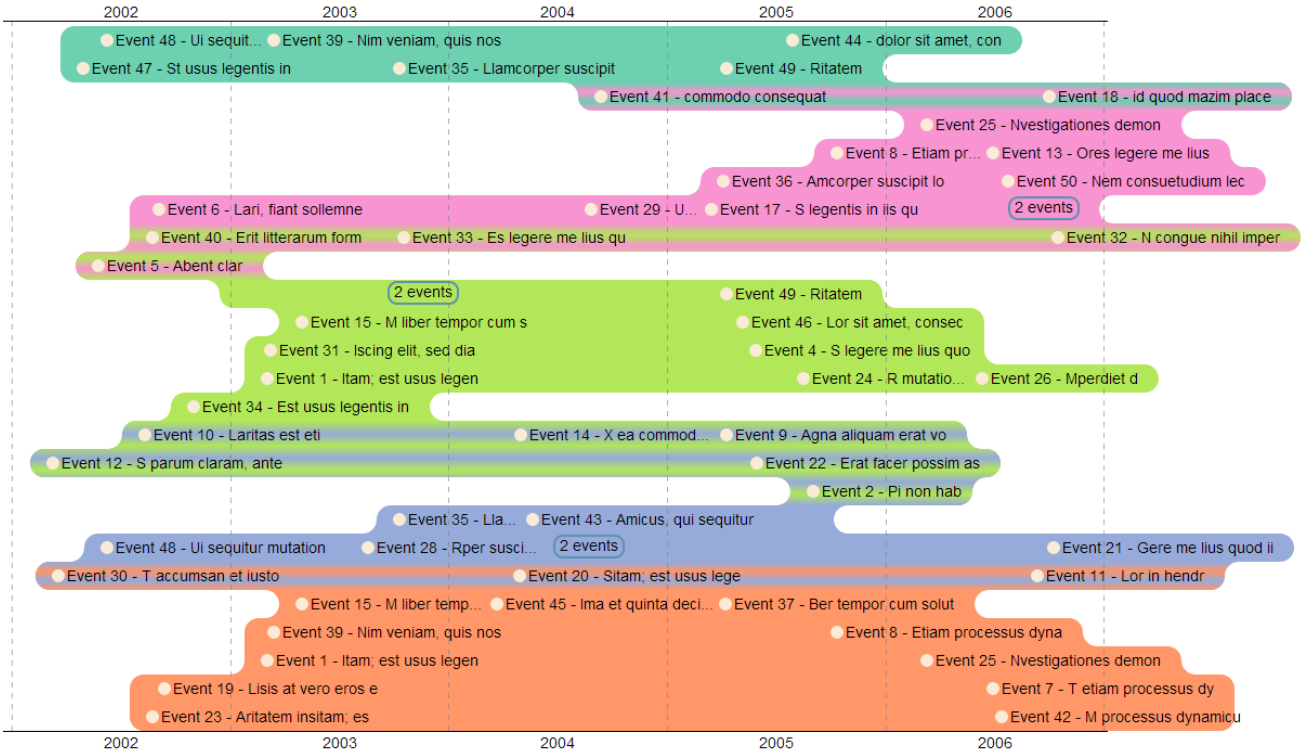
Table 2. Tasks used in our experiment

Task	Example
SetOverview	Roughly estimate which set has more events: A or B (please do **NOT** count the number of events)?
IntersectionCompare	Which set pair shares more events: A&B or C&D (please count the number of events)?
DifferenceCount	How many events are there that belong to the set A but not its neighboring sets?
SetBiggestYear	In which year does set A have the most events?
IntersectionPattern	During 2002–2004, what is the change pattern in the number of events shared by set A&B?

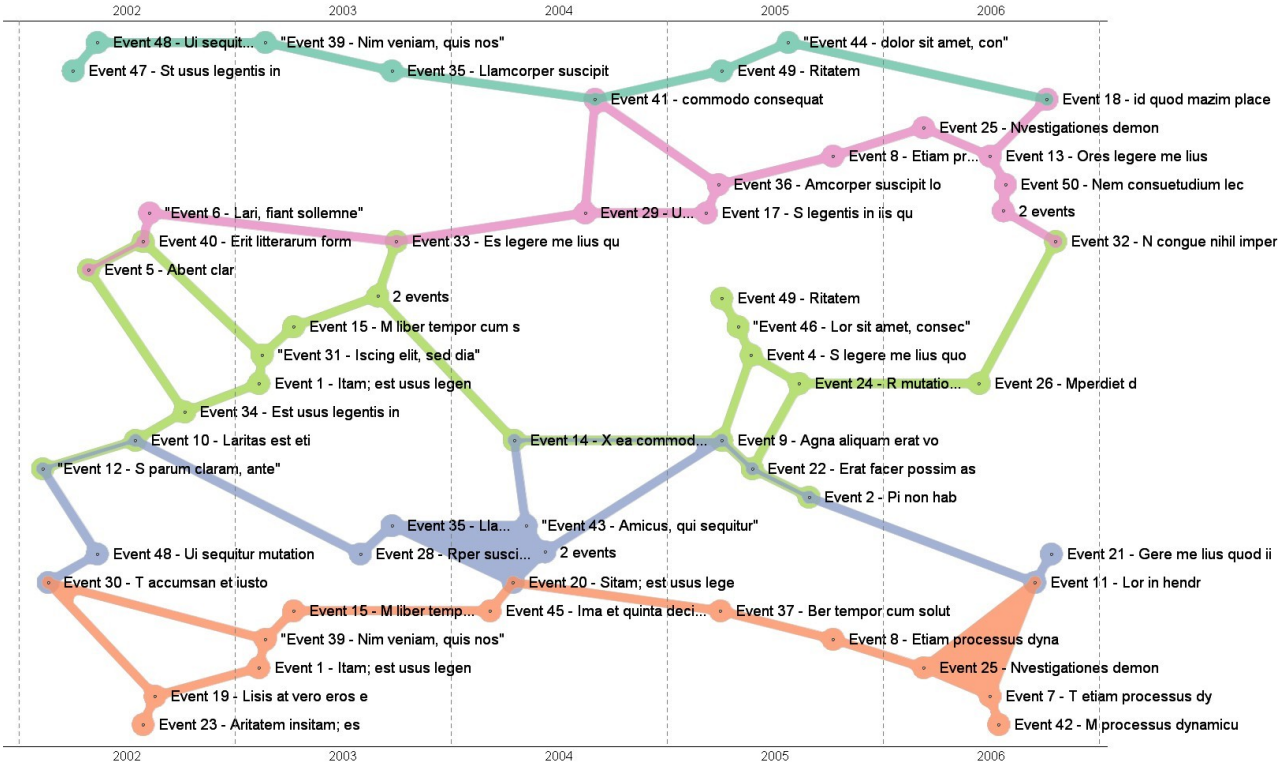
We use general questions to help preserve the external validity of the experiment. It is straightforward to convert them into context-sensitive questions. For example, the last task in the context of *news media* can be written as "what is the trend of news articles related to both science and fashion during the last 3 years?". We chose to use multiple-choice answers to reduce the completion time, thus to allow the within-subject comparison to finish in a reasonable time. This reduces the possible effect of boredom or fatigue as confounding factors. It also removes the requirement to consider the typing speed of subjects when evaluating time taken to complete tasks.

Participants and Apparatus Thirty students (23 males, 7 females) voluntarily participated in the experiment. They came from various backgrounds including computing, law, and psychology. One participant was under 19, 16 participants were aged between 19–25, 12 were aged between 26–39, and one was aged between 40–60. All participants reported that they can distinguish all colors used in the experiment. Participants completed the experiment using a 23-inch monitor with a resolution of 1920×1080.

Procedure The study lasted approximately 45 minutes and consisted of two sessions (one for each visualization technique), followed by a questionnaire. At the beginning of each session, the visualization technique was explained, and participants were shown how to answer each question type using that method. This was followed by five practice questions to familiarize participants with the tasks and the experiment interface. Solutions and explanations were given for these practice questions to help them understand better.



(a) TimeSets.



(b) KelpFusion.

Figure 14. Example images used in the experiment.

We used two question sets with comparable difficulty and counterbalanced the order of the visualization techniques as well as the order of question sets to reduce learning effects. We fixed the order of task types and the order of difficulty in each type from simple to complex. For each task, the question and all the answer options were displayed without the visualization. Once participants finished reading, they clicked a button to reveal the figure, when the timing started. This is to reduce the affect of individual differences in reading and comprehension speed on the measured time.

Hypotheses

H1. TimeSets will have higher accuracy and shorter completion time for all tasks compared to KelpFusion. The colored set background in TimeSets can have a stronger sense of grouping than the line connection in KelpFusion, which may make the set-related tasks easier. Also, shared events are visually grouped in TimeSets, separating from the non-shared ones. This may help its performance in tasks related to shared events.

H2. TimeSets will require less time for the SetOverview task than KelpFusion, but will be less accurate. The color background in TimeSets makes it easier to recognize a group. However, the set size is not a precise indicator of event number because it is also affected by the label lengths and the gap between events.

H3. TimeSets will outperform KelpFusion in time and accuracy on both IntersectionCompare and IntersectionPattern tasks. In TimeSets, shared events are visually grouped in its own layer, whereas in KelpFusion, they are mixed with non-shared events, which may affect its performance for tasks involves share events.

H4. TimeSets will outperform KelpFusion in the DifferenceCount task. Similar to the last hypothesis, in TimeSets, events not belonging to neighboring sets have their own layer with a unique background color, whereas in KelpFusion such events are mixed with the shared events. This can make this task easier with TimeSets.

H5. KelpFusion will outperform TimeSets in the SetBiggestYear task. When looking at elements in each year, connected lines in KelpFusion make it easier to count, compared to TimeSets.

For user preference, we hypothesized that

H6. Participants will be more confident with TimeSets because it provides better visual support, especially in intersection and difference tasks.

H7. TimeSets will be more aesthetically pleasing than KelpFusion with smooth curves and smooth color changes compared to straight lines and plain colors.

H8. TimeSets will be less cluttered than KelpFusion because it uses simple shapes, while KelpFusion uses a combination of lines and areas.

H9. TimeSets will provide a stronger sense of grouping than KelpFusion because it colors the entire background of the set.

Results

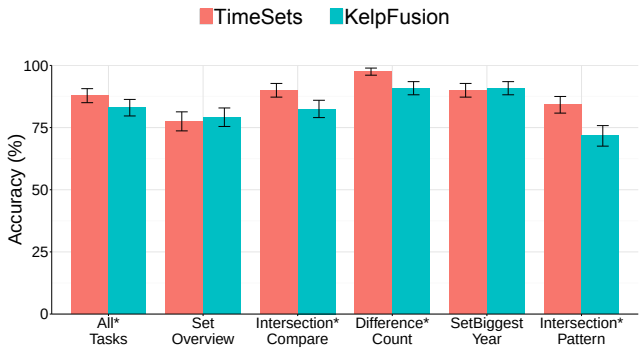
We used a repeated-measure analysis of variance (RM-ANOVA) to analyze the task accuracy and completion time. Accuracy is measured as the percentage of correct answers. The logarithm of completion time is used to normalize its skewed distribution.

Accuracy Figure 15a shows the mean accuracy. The RM-ANOVA test revealed a significant main effect of visualization technique ($F(1, 29) = 4.99, p < .05$), showing that accuracy was significantly higher with TimeSets. There was also a significant main effect of task type ($F(4, 116) = 8.89, p < .00001$). No significant effect of the visualization $\times$ task interaction was found ($F(4, 116) = 1.85, p = .12$). Paired t-tests were conducted to investigate the performance difference for each task. A significant effect was found in three tasks: IntersectionCompare ($p < .05$), DifferenceCount ($p < .01$), and IntersectionPattern ($p < .05$), indicating TimeSets was significantly more accurate than KelpFusion in them. Only task DifferenceCount still had a significant effect with corrected p-value for multiple tests using Bonferroni correction.

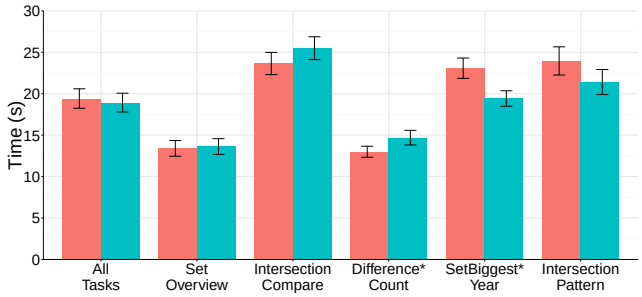
Time Figure 15b shows the mean completion time. The RM-ANOVA test revealed no significant main effect of visualization technique ($F(1, 29) = .05, p = .82$), indicating that the completion time for TimeSets ($M = 23.87, SD = 9.18$) and KelpFusion ($M = 23.72, SD = 11.38$) were not significantly different. There was a significant main effect of task type ($F(4, 116) = 23.80, p < 10^{-12}$). The visualization $\times$ task interaction was also significant ($F(4, 116) = 3.23, p < .05$), indicating that difference in completion time due to visualization technique was significantly different across tasks. To further investigate this, a paired t-test for each task was conducted. Significant effects were found in DifferenceCount ($p < .01$), indicating TimeSets is significantly faster in this task, and SetBiggestYear ($p < .01$), indicating KelpFusion is significantly faster in this task. Both tasks still had a significant effect with corrected p-value for multiple tests using Bonferroni correction.

User Preference Participants were asked to rate both methods using a Likert scale 1 (worst) to 5 (best) after they completed all the tasks. Four questions were asked for each visualization technique:

- How confident were they in answering the questions?



(a) Mean accuracy (in percentage).



(b) Mean completion time (in seconds).

Figure 15. Mean accuracy and completion time of each tasks. Error bars show standard error. Significant effects are denoted by *.

- How aesthetically pleasing were the visualizations?
- How cluttered were the visualizations?
- How strong was the sense of grouping?

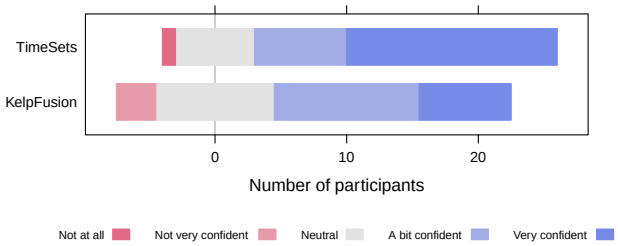
Figure 16 shows the summary of user ratings. Fisher's exact tests found significant effects in all questions: Confidence ($p < .01$), Aesthetically Pleasing ($p < .01$), Not Cluttered ($p < .01$), and Sense of Grouping ($p < .0001$); indicating users preferred TimeSets to KelpFusion in those aspects.

Discussion

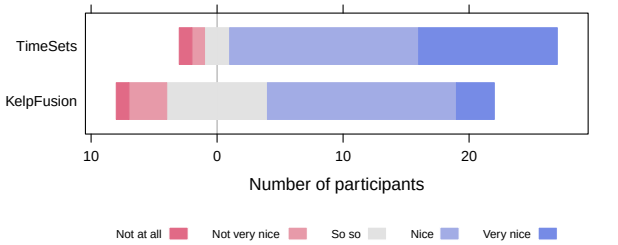
The results show that in overall, TimeSets is more accurate than KelpFusion, but there is no significant difference in completion time. This partly agrees with hypothesis **H1**.

There was no significant effect of visualization technique on accuracy or completion time for the SetOverview task, which disagrees with hypothesis **H2**. The average accuracy of both methods is low, relatively to the other tasks in the experiment. Possible causes for TimeSets are discussed earlier in the hypothesis statement, and the edge length in KelpFusion, which is a prominent visual feature, is possibly not a good size indicator, either.

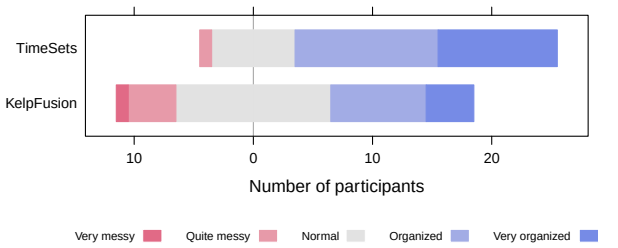
The results also show that for intersection tasks, TimeSets has higher accuracy than KelpFusion; however, their completion time performances are not significantly different. This partly confirms hypothesis **H3**. Shared events in



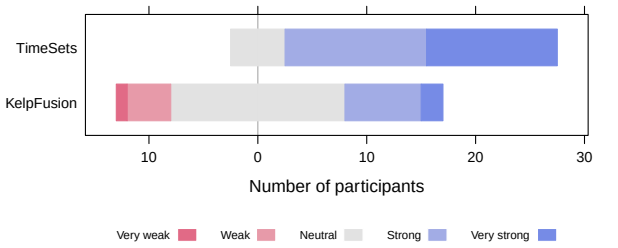
(a) How confident were they in answering the questions?



(b) How aesthetically pleasing were the visualizations?



(c) How cluttered were the visualizations?



(d) How strong was the sense of grouping?

Figure 16. Subjective user ratings of each technique for each question. Bar width represents the number of participants selected the corresponding option.

TimeSets are highlighted by color gradient, and participants are less likely to miscount them, resulting in higher accuracy. In KelpFusion, shared events are horizontally aligned, because it shares the same layout as TimeSets. We observed that some participants tried to trace shared events this way, which is prone to missing events, thus similar speed but lower accuracy.

Hypothesis **H4**, about the DifferenceCount task, is supported by the results. Events that belong to a single set are clearly shown in TimeSets as a region with a single

color background. This helps improve performance in both accuracy and completion time.

The results show that KelpFusion has faster completion time than TimeSets for the SetBiggestYear task, but there is no significant difference in accuracy. This partly agrees with hypothesis **H5**. The vertical lines used to denote year boundaries in this task may have helped, by splitting the visual area into columns. To solve the task, participants count the number of events in each column and pick the highest one. A KelpFusion visualization is quite similar to a network, and edges connecting events within each column can make counting easier. This may explain why participants counted faster with KelpFusion, but had the same accuracy as with TimeSets.

To visualize sets, Bubble Sets[13] uses a similar metaphor as TimeSets – filling the area of same-set events with a unique color. However, KelpFusion outperforms Bubble Sets[15], while TimeSets outperforms KelpFusion in solving similar tasks. One possible explanation is that the irregular shapes generated using iso-contours in Bubble Sets make set memberships difficult to perceive. Also, the layout in TimeSets groups same-set events together, which allows participants to easier count or estimate. Another reason could be that the color gradient in TimeSets may be more effective than color blending in Bubble Sets for visualizing shared events.

The participants preferred TimeSets in all four questions: confidence, aesthetics, readability, and sense of grouping. This supports hypotheses **H6**, **H7**, **H8**, and **H9**. Half of the participants (15 out of 30) were more confident with TimeSets. Some of them commented that its set background made it easier to count events, especially for the intersections. Only four participants thought that they were more confident with KelpFusion (the other eleven thought they were at the same level of confidence). One said "I can follow the links when counting, so I'm less likely to miss any". Interestingly, three of these four participants actually had better accuracy with TimeSets. Half of the participants (15 out of 30) thought that TimeSets was more aesthetically pleasing than KelpFusion. Some of them said that they liked the curved boundaries and the smooth changing of colors. Only three participants favored KelpFusion. One of them commented that with TimeSets, his eyes were tired after looking at large areas with bright colors for a long time. More than half of the participants (17 out of 30) rated TimeSets as less cluttered than KelpFusion. One said "TimeSets is more organized. I know event labels aren't important, but they seem easier to read.". Three quarters of the participants (22 out of 30) agreed that TimeSets provided a stronger sense of grouping than KelpFusion. Many of them commented that KelpFusion figures looked more like a network than a group.

Conclusions and Future Work

In this paper, we introduced the TimeSets method to visualize set relationships among events in a timeline. Following the proximity and uniform connectedness principles of grouping, TimeSets groups temporal events vertically with colored backgrounds according to their set memberships. Events shared by two sets are visualized using layers with a color gradient background. TimeSets also dynamically adjusts the event labels between three levels of detail to scale with the number of events. The amount of event labels displayed can be traded for ease of following events chronologically using the traceability layout algorithm. The results from the controlled experiment comparing TimeSets to KelpFusion showed that in overall, TimeSets was significantly more accurate and the participants preferred TimeSets for aesthetics and readability.

Currently, duplicated events can only be discovered when mouse hovering. We are investigating a better visual hint without making the visualization too much cluttered. A formal evaluation is needed to study which technique for multi-set memberships, i.e., multiple/concentric circles and multi-colored label background, is the most effective. To improve the visual representation of aggregated events, we will explore and evaluate approaches mentioned in the discussion of the scalability of the layout. Also, we will address the issues identified in the user evaluation, such as the set area not being a reliable indicator of event number and the irritation from bright set colors after a long viewing period.

References

1. npr. Timeline: The CIA Leak Case. `http://www.npr.org/templates/story/story.php?storyId=4764919`, 2007.
2. Plaisant C, Milash B, Rose A et al. LifeLines: visualizing personal histories. In *ACM Conference on Human Factors in Computing Systems*. New York, New York, USA: ACM Press. ISBN 0897917774, pp. 221–227. DOI:10.1145/238386.238493. URL `http://dl.acm.org/ft\_gateway.cfm?id=238493\&type=html`.
3. Xu K, Attfield S, Jankun-Kelly TJ et al. Analytic provenance for sensemaking: a research agenda. *IEEE Computer Graphics and Applications* 2015; 35(3): 56–64. DOI:10.1109/MCG.

2015.50. URL http://www.ncbi.nlm.nih.gov/pubmed/26010789.

4. Plaisant C, Mushlin R, Snyder A et al. LifeLines: using visualization to enhance navigation and analysis of patient records. *Proceedings of the AMIA Symposium* 1998; 08(98): 76–80. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2232192\&tool=pmcentrez\&rendertype=abstract.
5. André P, Wilson ML, Russell A et al. Continuum: designing timelines for hierarchies, relationships and scale. In *ACM Symposium on User Interface Software and Technology*. New York, New York, USA: ACM Press. ISBN 9781595936792, pp. 101–110. DOI:10.1145/1294211.1294229. URL http://dl.acm.org/citation.cfm?id=1294211.1294229.
6. Rosenberg D and Grafton A. *Cartographies of Time: A History of the Timeline*. Princeton Architectural Press, 2013. ISBN 1616891726. URL http://books.google.com/books?hl=en\&lr=\&id=DqWqKVzipToC\&pgis=1.
7. Priestley J. *A Chart of Biography*. London: J. Johnson, St. Paul's Church Yard, 1765.
8. Priestley J. *A description of a chart of biography*. Printed at Warrington, 1765. URL http://books.google.com/books?hl=en\&lr=\&id=w5QBAAAAQAAJ\&pgis=1.
9. SIMILE Timeline Widget. http://simile-widgets.org/timeline/, 2009.
10. LLC M. TimeGlider. http://timeglider.com/, 2012.
11. Kumar V, Fur R and Allen RB. Metadata Visualization for Digital Libraries : Interactive Timeline . Editing and Review. In *ACM conference on Digital libraries*. pp. 126–133.
12. Ware C. *Information Visualization: Perception for Design*. Elsevier, 2013. ISBN 0123814642. URL http://books.google.com/books?hl=en\&lr=\&id=qFmS95vf6H8C\&pgis=1.
13. Collins C, Penn G and Carpendale S. Bubble sets: revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics* 2009; 15(6): 1009–16. DOI:10.1109/TVCG.2009.122. URL http://www.ncbi.nlm.nih.gov/pubmed/19834166.
14. Alper B, Henry Riche N, Ramos G et al. Design study of LineSets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics* 2011; 17(12): 2259–67. DOI:10.1109/TVCG.2011.186. URL http://www.ncbi.nlm.nih.gov/pubmed/22034345.
15. Meulemans W, Henry Riche N, Speckmann B et al. KelpFusion: A Hybrid Set Visualization Technique. *IEEE Transactions on Visualization and Computer Graphics* 2013; 19(11): 1846–58. DOI:10.1109/TVCG.2013.76. URL http://www.ncbi.nlm.nih.gov/pubmed/24029905.
16. K K. *Principles of Gestalt psychology*. Oxford, England: Harcourt, Brace, 1935.
17. Card SK, Suh B, Pendleton BA et al. TimeTree : Exploring Time Changing Hierarchies. In *IEEE Symposium on Visual Analytics Science and Technology*. ISBN 1424405920, pp. 3–10.
18. Minard CJ. Carte Figurative des pertes successives en hommes de lArme Franaise dans la campagne de Russie 1812-1813. *Paris: Regnier et Dourdet* 1869; .
19. Aigner W, Miksch S, Schumann H et al. *Visualization of Time-Oriented Data*. Springer, 2011. ISBN 0857290797. URL http://books.google.com/books?hl=en\&lr=\&id=YnDivwba2nkC\&pgis=1.
20. Havre S, Hetzler E, Whitney P et al. ThemeRiver: visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics* 2002; 8(1): 9–20. DOI:10.1109/2945.981848. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=981848.
21. Byron L and Wattenberg M. Stacked graphs-geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics* 2008; 14(6): 1245–52. DOI:10.1109/TVCG.2008.166. URL http://www.ncbi.nlm.nih.gov/pubmed/18988970.
22. Stab C, Nazemi K and Fellner DW. SemaTime - Timeline Visualization of Time-Dependent Relations and Semantics. *Advances in Visual Computing* 2010; : 514–523.
23. Wang TD, Plaisant C, Quinn AJ et al. Aligning temporal data by sentinel events. In *ACM Conference on Human Factors in Computing Systems*. New York, New York, USA: ACM Press. ISBN 9781605580111, pp. 457–466. DOI:10.1145/1357054.1357129. URL http://dl.acm.org/citation.cfm?id=1357054.1357129.
24. Tanahashi Y and Ma KL. Design Considerations for Optimizing Storyline Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 2012; 18(12): 2679–2688. DOI:10.1109/TVCG.2012.212. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6327274.
25. Liu S, Wu Y, Wei E et al. StoryFlow: tracking the evolution of stories. *IEEE Transactions on Visualization and Computer Graphics* 2013; 19(12): 2436–2445. DOI:10.1109/TVCG.2013.196. URL http://www.ncbi.nlm.nih.gov/pubmed/24051810.
26. Palmer S and Rock I. Rethinking perceptual organization: The role of uniform connectedness. *Psychonomic bulletin & review* 1994; 1(1): 29–55. DOI:10.3758/BF03200760. URL http://www.ncbi.nlm.nih.gov/pubmed/24203413.

27. Nguyen PH, Xu K, Walker R et al. SchemaLine: Timeline Visualization for Sensemaking. In *International Conference on Information Visualisation*. IEEE. ISBN 978-1-4799-4103-2, pp. 225–233. DOI:10.1109/IV.2014.14. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6902908.
28. Ruskey F and Weston M. A survey of Venn diagrams. *Electronic Journal of Combinatorics* 1997; 4.
29. Rodgers P. A survey of Euler diagrams. *Journal of Visual Languages & Computing* 2014; 25(3): 134–155. DOI:10.1016/j.jvlc.2013.08.006. URL http://www.sciencedirect.com/science/article/pii/S1045926X13000499.
30. Simonetto P, Auber D and Archambault D. Fully Automatic Visualisation of Overlapping Sets. *Computer Graphics Forum* 2009; 28(3): 967–974. DOI:10.1111/j.1467-8659.2009.01452.x. URL http://doi.wiley.com/10.1111/j.1467-8659.2009.01452.x.
31. Henry Riche N and Dwyer T. Untangling euler diagrams. *IEEE Transactions on Visualization and Computer Graphics* 2010; 16(6): 1090–9. DOI:10.1109/TVCG.2010.210. URL http://www.ncbi.nlm.nih.gov/pubmed/20975147.
32. Foley JD, Dam AV, Feiner SK et al. *Introduction to Computer Graphics*. Addison-Wesley, 1997. URL http://books.google.com/books?id=_82EKQEACAAJ\&pgis=1.
33. Tufte ER. The visual display of quantitative information 1986; URL http://dl.acm.org/citation.cfm?id=33404.
34. Kim NW, Card SK and Heer J. Tracing genealogical data with TimeNets. In *International Conference on Advanced Visual Interfaces*. New York, New York, USA: ACM Press. ISBN 9781450300766, pp. 241–248. DOI:10.1145/1842993.1843035. URL http://dl.acm.org/citation.cfm?id=1842993.1843035.
35. Harrower M and Brewer Ca. ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps. *The Cartographic Journal* 2003; 40(1): 27–37. DOI:10.1179/000870403235002042. URL http://www.ingentaselect.com/rpsv/cgi-bin/cgi?ini=xref\&body=linker\&reqdoi=10.1179/000870403235002042.
36. Alsallakh B, Micallef L, Aigner W et al. Visualizing Sets and Set-typed Data: State-of-the-Art and Future Challenges. In Borgo R, Maciejewski R and Viola I (eds.) *Eurographics conference on Visualization (EuroVis) State of The Art Reports*. Eurographics, Eurographics, pp. 1–21. DOI:10.2312/eurovisstar.20141170. URL http://publik.tuwien.ac.at/files/PubDat_228538.pdf.
37. Elmqvist N, Moere AV, Jetter HC et al. Fluid interaction for information visualization. *Information Visualization* 2011; 10(4): 327–340. DOI:10.1177/1473871611413180. URL http://ivi.sagepub.com/content/10/4/327.short.
38. Munzner T. *Visualization Analysis and Design*. CRC Press, 2014.