Exploring Flow, Factors, and Outcomes of Temporal Event Sequences with the Outflow Visualization

Krist Wongsuphasawat and David Gotz

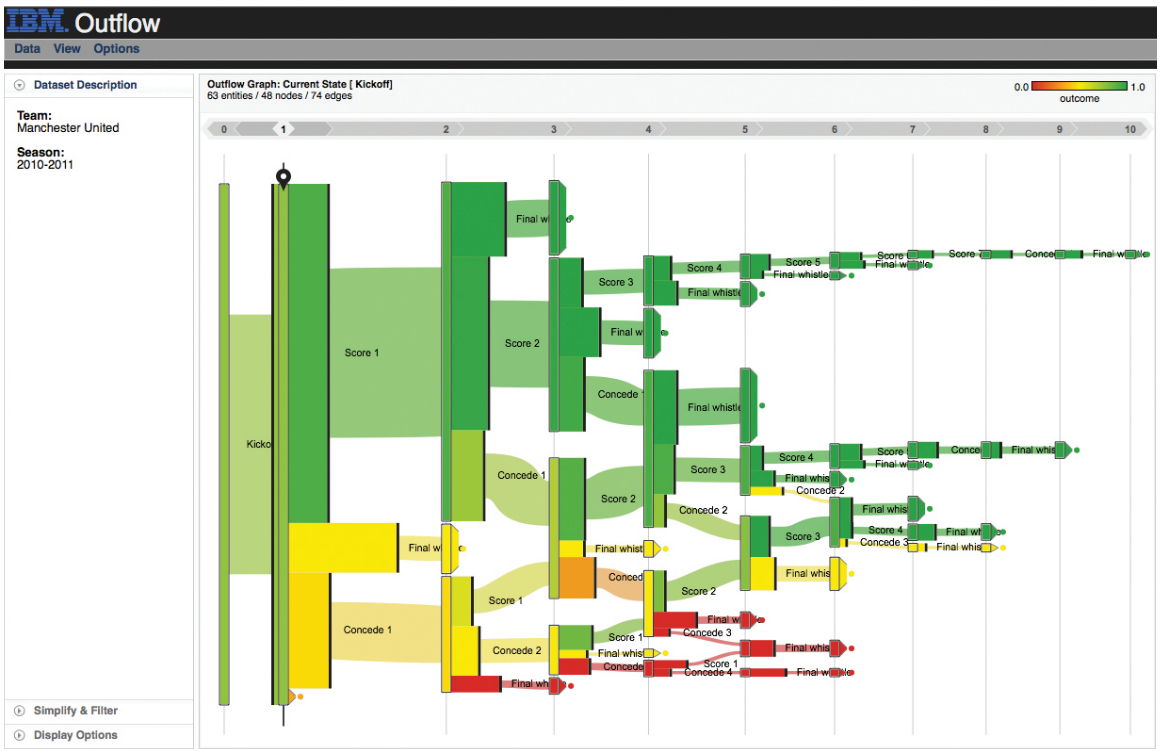


Fig. 1. Outflow processes temporal event data and visualizes aggregate event progression pathways together with associated statistics (e.g. outcome, duration, and cardinality). Users can interactively explore the paths via which entities arrive and depart various states. This screenshot shows a visualization of Manchester United's 2010-2011 soccer season. Green shows pathways with good outcomes (i.e., wins) while red shows pathways with bad outcomes (i.e., losses).

Abstract—Event sequence data is common in many domains, ranging from electronic medical records (EMRs) to sports events. Moreover, such sequences often result in measurable outcomes (e.g., life or death, win or loss). Collections of event sequences can be aggregated together to form event progression pathways. These pathways can then be connected with outcomes to model how alternative chains of events may lead to different results. This paper describes the *Outflow* visualization technique, designed to (1) aggregate multiple event sequences, (2) display the aggregate pathways through different event states with timing and cardinality, (3) summarize the pathways' corresponding outcomes, and (4) allow users to explore external factors that correlate with specific pathway state transitions. Results from a user study with twelve participants show that users were able to learn how to use Outflow easily with limited training and perform a range of tasks both accurately and rapidly.

Index Terms—Outflow, information visualization, temporal event sequences, state diagram, state transition.

1 Introduction

Life can often be described as a series of temporal *events*. These events contain rich information that, when put together into *event sequences*, can define history, reveal insightful facts, or lead to discoveries. Collections of event sequences are growing rapidly throughout many areas such as electronic medical records (EMRs), sports events, call centers, transportation incident logs, and student progress reports. In addition, many event sequences have associated *outcomes*. For example, outcomes for EMR data could be measured by cost, mortality or discharge rates. For sports, an outcome could be a win, loss or draw.

Analyzing common patterns, or *pathways*, in a set of event sequences can help people better understand aggregate event progression behavior. In addition, connecting these pathways to their associated outcomes can help data analysts discover how certain progression paths may lead to better or worse results.

For example, consider a medical dataset containing information

- *Krist Wongsuphasawat is with University of Maryland. This work is part of his internship at IBM T.J. Watson Research Center. E-mail: kristw@cs.umd.edu.*
- *David Gotz is with IBM T.J. Watson Research Center. E-mail: dgotz@us.ibm.com.*

Manuscript received 31 March 2012; accepted 1 August 2012; posted online 14 October 2012; mailed on 5 October 2012.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

about a set of patients with a dangerous disease. Each patient is described by an outcome measurement (e.g., if they survived the disease or not) and an event sequence containing the date that certain symptoms were first observed by a doctor. An analysis of pathways in such a dataset might lead to the discovery that patients with symptoms A, B and C were likely to die in the hospital while patients with symptoms A, B and D were more likely to recover.

Similarly, consider a dataset representing a set of soccer matches where goals are events and wins are considered a good outcome. An analysis of pathways in this dataset could help answer questions such as, "Does a team win more often when it comes from behind?"

While analyzing event sequence data as described above can help answer many questions, there are often external *factors*—beyond the set of event types that define an event sequence—that make an analysis even more complex. Such factors, such as the administration of a drug to a sick patient, or a soccer player receiving a red card (which leaves his/her team short-handed), can often change the course of subsequent events. These factors must be incorporated into an analysis to understand how they influence outcomes.

Finally, event collections can be massive. Major healthcare institutions have millions of medical records containing millions of event sequences, and these sequences can have many different event types. The scale and variability of this problem can lead to an extremely complex set of pathways for many scenarios. For example, even for a small data set with just five event types and where each event sequence has just five events, there are $3,125$ (5^5) possible pathways. This vast amount of information can be overwhelming and makes these datasets difficult to analyze.

To address these challenges, we have designed *Outflow*, an interactive visualization that combines multiple event sequences and their outcomes into a graph-based visual representation. Outflow can summarize collections of event sequences and display all pathways, time gaps between each step, and their associated outcomes. Users can interact with the visualization through direct manipulation techniques (e.g., selection and brushing) and a series of control widgets. The interactions allow users to explore the data in search of insights, including information about which factors correlate most strongly with specific pathways. An early prototype [32] of our approach received positive feedback and requests for extensions.

This paper describes the Outflow visualization in more detail by explaining key layout methods behind the visualization. We also introduce two important extensions—*simplification* and *factor analysis*—which were developed in response to preliminary feedback. We illustrate the generalizability of Outflow by discussing two applications (a medical use case and a sports statistics use case) and demonstrate its power via a formal user study. The key research contributions presented in this paper are as follows:

- An approach to aggregate and visualize multiple event sequences and their outcomes, called *Outflow*, that allows for the visual analysis of event progression pathways and their associated properties (including timing, cardinality, and outcomes).
- A multi-step layout process for Outflow graphs that reduces edge crossing and straightens unnecessarily curvy edges while preventing overlaps. We believe that a part of this process can be applied to directed acyclic graphs in general.
- A set of interaction techniques for exploring the Outflow visualization, including navigation, simplification, and correlated factor analysis.
- Results and discussion from a study evaluating user performance for a set of event sequence analysis tasks.

The remainder of this paper is organized as follows. Section 2 presents two motivating applications and is followed by a review of related work in Section 3. The Outflow design is discussed in Section 4. We then report the results of our evaluation in Section 5 and conclude in Section 6.

2 Motivation

Outflow provides a general solution for a class of event sequence analysis problems. This section describes two examples from different application domains which served as motivating problems for our work.

2.1 Congestive Heart Failure (CHF)

Outflow was originally inspired by a problem faced by a team of cardiologists. They were working to better understand disease evolution patterns using data from a cohort of patients at risk of developing *congestive heart failure (CHF)*. CHF is generally defined as the inability of the heart to supply sufficient blood flow to meet the needs of the body. CHF is a common, costly, and potentially deadly condition that afflicts roughly 2% of adults in developed countries with rates growing to 6-10% for those over 65 years of age [15]. The disease is difficult to manage and no system of diagnostic criteria has been universally accepted as the gold standard.

One commonly used system comes from the *Framingham study* [14]. This system requires the simultaneous presence of at least two major symptoms (e.g., S3 gallop, Acute pulmonary edema, Cardiomegaly) or one major symptom in conjunction with two minor symptoms (e.g., Nocturnal cough, Pleural effusion, Hepatomegaly). In total, 18 distinct Framingham symptoms have been defined.

While these symptoms are used regularly to diagnose CHF, our medical collaborators are interested in understanding how the various symptoms and their order of onset correlate with patient outcome. To examine this problem, we were given access to an anonymized dataset of 6,328 patient records. Each patient record includes timestamped entries for each time a patient was diagnosed with a Framingham symptom. For example:

Patient#1:(27 Jul 2009, Ankle edema), (14 Aug 2009, Pleural effusion), ...

Patient#2:(17 May 2002, S3 gallop), (1 Feb 2003, Cardiomegaly), ...

The dataset also contains information about medication orders and patient metadata. Available metadata includes date of birth, gender, date of CHF diagnosis, and (when applicable) date of death.

In line with the use of Framingham symptoms for diagnosis, we assume that once a symptom has been observed it applies perpetually. We therefore filter the event sequences for each patient to select only the first occurrence of a given symptom type. The filtered event sequences describe the *flow* for each patient through different disease states. For example, a filtered event sequence ***symptom A*** $\rightarrow$ ***symptom B*** indicates that the patient's flow is ***no symptom*** $\rightarrow$ ***symptom A*** $\rightarrow$ ***symptoms A and B***. We used the presence (or lack thereof) of a date of death as an outcome measure (dead or alive).

Our inspirational task was to examine aggregated statistics for the flows of many patients to find common disease progression paths. In addition, we wanted to discover any correlations between these paths and either (1) patient outcomes (i.e. mortality) or (2) external factors (i.e. medications).

2.2 Soccer Result Analysis

Although originally inspired by the medical application outlined above, Outflow itself is not domain specific and can generalize to other application areas. To demonstrate the broad applicability of our work, we have also used Outflow to analyze soccer match results. For example, Fig. 1 shows an Outflow visualization of the 2010-2011 season for Manchester United Football Club (Man U.), an English Premier League soccer club. Man U. has won the most trophies in English soccer, including a record 19 league titles and a record 11 FA Cups. It is one of the wealthiest and most widely supported teams in the world.

The 2010–2011 season was another successful one for Man U. in which they won multiple trophies. To better understand their route to success, we collected data from all 61 matches that Man U. played that season. For both Man U. and their opponents, we captured timestamped events for every kickoff, every goal scored, and every final whistle. We also recorded the outcome for every match (win, loss, or draw) along with timestamped records of every yellow and red cards.

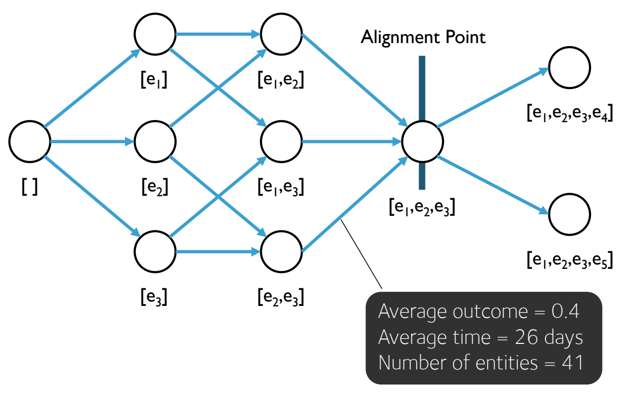


Fig. 2. Multiple temporal event sequences are aggregated into a representation called an *Outflow graph*. This structure is a directed acyclic graph (DAG) that captures the various event sequences that led to the alignment point and all the sequences that occurred after the alignment point. Aggregate statistics are then anchored to the graph to describe specific subsets of the data.

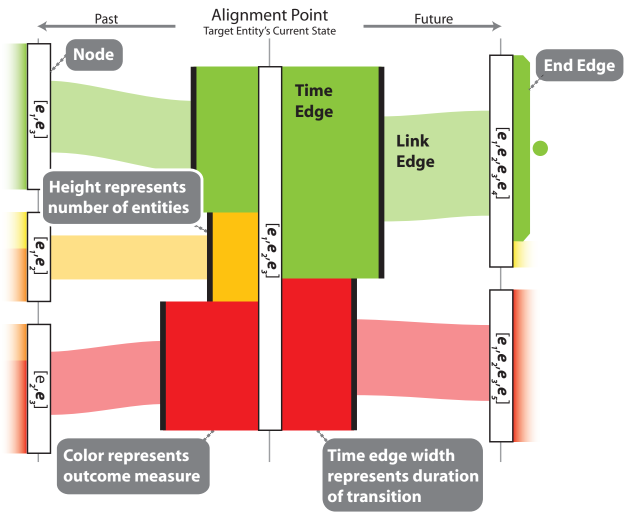

Past
Alignment Point
Target Entity's Current State
Future
Node
End Edge
Time Edge
Link Edge
Height represents number of entities
Color represents outcome measure
Time edge width represents duration of transition

Fig. 3. Outflow visually encodes nodes in the Outflow graph using vertical rectangles. Edges are represented using two distinct visual marks: time edges and link edges. Color is used to encode average outcome.

As in the healthcare case, events are cumulative. Each time a goal is scored, it is added to the scoreline with the goal tally increasing over the course of a match. Using Outflow, sports analysts are able to see the average outcome associated with each scoreline, find the average time to the next goal, predict what is likely to occur next, and understand how non-goal factors, such as red cards, can impact how a match may progress. This use case is the one tested in our evaluation where users were asked to perform many of these tasks. The study design and results are described in detail in Section 5.

3 Related Work

Temporal data takes many forms and a wide variety of research efforts have explored novel ways of tackling problems in this area. These include work on event sequence visualizations, state diagram visualizations, and flow visualizations.

3.1 Temporal Event Sequence Visualizations

A number of researchers have explored visualization techniques for temporal event sequences. In the early years, many systems focused on visualizing a single record [1, 2, 3, 7, 11, 12, 20]. The most common approach is to place events on a horizontal timeline according to the time that events occurred. Later, attention shifted towards visualizing multiple records in parallel. One popular technique is to stack instances of single-record visualizations and to provide additional functionality for searching [9, 28, 29, 30, 34, 35], filtering [30], and grouping [6, 18]. However, as the event sequences databases become larger, techniques that can provide abstractions of multiple event sequences are needed. More recently, a technique called *LifeFlow* [33] introduced a way to aggregate and provide an abstraction for multiple event sequences. LifeFlow's aggregation combines multiple event sequences into a tree. The Outflow technique described in this paper will combine multiple event sequences into a graph. The graph-based representation makes it easier to compare alternative paths to the same state. Outflow also integrates outcome statistics which are not part of LifeFlow's design.

3.2 State Diagram Visualizations

Our approach aggregates event sequences into an *Outflow graph* which is analogous to a state diagram [5] or state transition graph. State diagrams are used in computer science and related fields to represent a system of states and state changes. State diagrams are generally displayed as simple node-link diagrams where each state is depicted as a node and transitions are drawn as links [4]. Many visualizations of state diagrams have been developed [4, 21, 22, 27, 31]. These typically focus on multivariate graphs where a number of attributes are associated with every node. Variants on traditional state diagrams have also been explored, such as *Petri nets* (also known as a *place/transition* net or *P/T net*) [17], which offer a graphical notation for stepwise processes that include choice, iteration, and concurrent execution. However, to the best of our knowledge, these approaches do not display or allow easy comparison of the transition time, which is one of Outflow's design goals.

3.3 Flow & Parallel Coordinates Visualizations

Another group of visualizations called *Sankey Diagrams* [10, 23] was designed to visualize flow quantities in process systems. van den Elzen and van Wijk [26] also apply a technique similar to Sankey Diagrams for visualizing decision trees. However, they focus on displaying the proportion of the flow that splits in different ways, without showing temporal information about each transition. The visual display of Outflow also looks similar to parallel sets [13], but the underlying data types are different. Parallel coordinates are used for multidimensional data while Outflow was designed for temporal event sequences.

During the development of Outflow, Google introduced "Flow Visualization" [16] as part of Google Analytics to show the sequences of pages that visitors flow through website. This visualization, however, does not include time gaps between each step in the visual display and events are not accumulated.

4 Description of the Visualization

Outflow's design consists of four key elements: data aggregation, visual encoding, graph layout and drawing methods, and user interaction. This core design is then expanded to support two important extensions: simplification and factors.

4.1 Data Aggregation

The first step in creating an Outflow visualization is data aggregation. An *entity* E (e.g., a patient record) contains a series of timestamped events (e_j). Our technique can aggregate multiple entities under the following assumptions: (1) Events are persistent. (2) The order of events does not matter when determining if two or more entities should be aggregated. For example, both use cases in Section 2 exhibit these properties.

To begin aggregation, we treat each entity as a progression *pathway* through different states (S_i) with transitions between states S_m and S_n denoted as $T_{m \rightarrow n}$. Each state is defined as a set of zero or more events that an entity has experienced at or before time t_i,

$$\begin{aligned} E &= (S_0,t_0) \rightarrow (S_1,t_1) \rightarrow (S_2,t_2) \rightarrow (S_3,t_3) \rightarrow \ldots \rightarrow (S_n,t_n) \\ S_i &= [e_1, e_2, e_3, \ldots, e_i] \end{aligned}$$

Given a collection of entities, $\{E\}$, we choose one state (which must be experienced by all $E \in \{E\}$) as an *alignment point*. For example, we can align a set of medical records around a state where all patients have the same three symptoms (and no other symptoms). After choosing an alignment point[1], we aggregate all entities that pass through the alignment point into a data structure called an *Outflow graph* (Fig. 2).

An Outflow graph is a state diagram expressed as a directed acyclic graph (DAG). A node is added to the graph for each unique state observed in $\{E\}$ (e.g., a node for each unique combination of co-occurring symptoms). Edges represent each state transitions observed in $\{E\}$, and they are annotated with various statistics: the number of entities that make the corresponding transition, the average outcome for these entities, and the average time gap between the states.

Therefore, an Outflow graph captures all event paths in $\{E\}$ that lead to the alignment point and all event paths that occur after the alignment point. In the medical analysis example, users can select a target patient from the database and use the target patient's current state as the alignment point. This approach allows for the analysis of historical data when considering the possible future progression of symptoms for the selected target patient. In the soccer analysis, users can align by a state with a specific score (e.g., 2-1), which would include only matches that, at some point in time during the game, reached the specified state (e.g., two "Score" and one "Concede" events). This could be useful for prediction from historical data.

This approach to aggregation produces a graph whose size is independent of the numbers of records. Instead, the size of the graph is dependent on the number of states in the dataset. The representation can therefore scale well to handle large numbers of records assuming a manageable state space. For example, one dataset that has nine records and another dataset that has nine million records may have the same number of states and could therefore be displayed using a comparable amount of screen space.

4.2 Visual Encoding

Based on the information contained in the Outflow graph, we have designed a rich visual encoding that displays (1) the time gap for each state change, (2) the cardinality of entities in each state and state transition, and (3) the average outcome for each state and transition. Drawing in part on prior work from FlowMap [19] and LifeFlow [33], we developed the visual encoding shown in Fig. 3.

Node (State): Each state is represented by a rectangle whose height is proportional to the number of entities.

Layer: We slice the graph vertically into layers. Layer i contains all states with i events. The layers are sorted from left to right, showing information from the past to the future. For example, in Fig. 1, the first layer (layer 0) contains only one node, which represents all records before any event. The next layer (layer 1) also has one node because all games begin with a "Kick off" event. Layer 2, however, has three nodes because each game evolved in one of three different ways: "Score", "Concede", or "Final whistle".

Edge (Transition): Each state transition is displayed using two visual marks: a *time edge* and a *link edge*. Time edges are rectangles whose width is proportional to the average time gap of the transition and height is proportional to the number of entities. Link edges connect nodes and time edges to convey sequentiality.

End Edge: Not all entities end in the same state. We use a trapezoid followed by a circle to mark these endpoints. Like transition edges, the height of the trapezoid is proportional to the number of entities that end at the corresponding state. The circles are included to ensure that small end edges remain visible.

[1] The system uses S_0 (the state where no events have occurred) as the default if no other alignment point is specified.

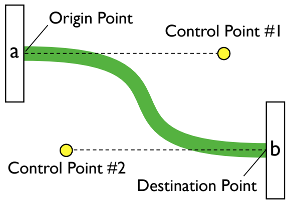


Fig. 4. Link edges are drawn as cubic Bézier curves. Control points are selected to ensure horizontal starting and ending edge slopes.

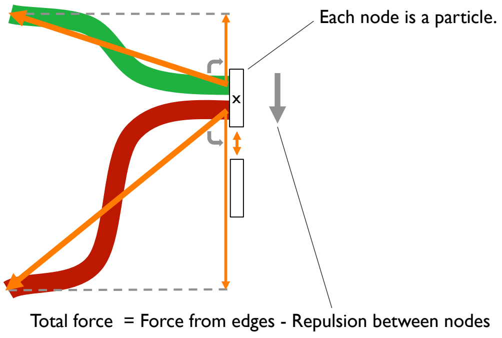


Fig. 5. A spring-based optimization algorithm is used to obtain straighter (and easier to read) edges. Nodes and edges are simulated as particles and springs, respectively. Spring repulsions are inserted between nodes. During the optimization, nodes are gradually moved along a vertical axis to reduce the spring tension in their edges.

Color-coding: Colors assigned to edges are used to encode the average outcome for the corresponding set of entities. The outcome values are normalized to the [0,1] range with higher values representing better outcomes. The normalized values are then mapped to a color scale. In our prototype, the color-coding scales linearly from red (outcome of 0) to yellow (0.5) to green (1). The color scale can be adjusted to accommodate users with color vision deficiency.

4.3 Graph Layout and Drawing Methods

Graphs with many nodes and edges can be difficult to visualize due to possible edge crossings and overlapping visual marks. We apply several techniques to emphasize connectivity and reduce clutter in the visualization.

4.3.1 Bézier Curve

Each link edge is drawn as a cubic Bézier curve to emphasize the connectivity and flow of the paths in the visual display. We make the control line from the origin point to the first control point perpendicular to the origin node, and the control line from the destination point to the second control point perpendicular to the destination node. As shown in Fig. 4, this ensures that the edges are horizontal at both the start and end.

4.3.2 Sugiyama's Heuristics

Outflow initially sorts nodes and edges in each layer according to their outcomes. However, this often leads to unnecessarily complex visualizations because of edge crossings. Therefore, we apply Sugiyama's heuristics [25], a well-known graph layout algorithm for DAGs, to reorder the elements in each layer to reduce edge crossings. Fig. 6a and 6b show example layouts of the same data before and after applying Sugiyama's heuristics, respectively.

4.3.3 Force-directed Layout

Once the order of nodes in a layer has been determined, the next step is to calculate the actual vertical position for each node. Positions are initially assigned by distributing the nodes equally along a layer's vertical axis. However, this method often results in unnecessarily curvy

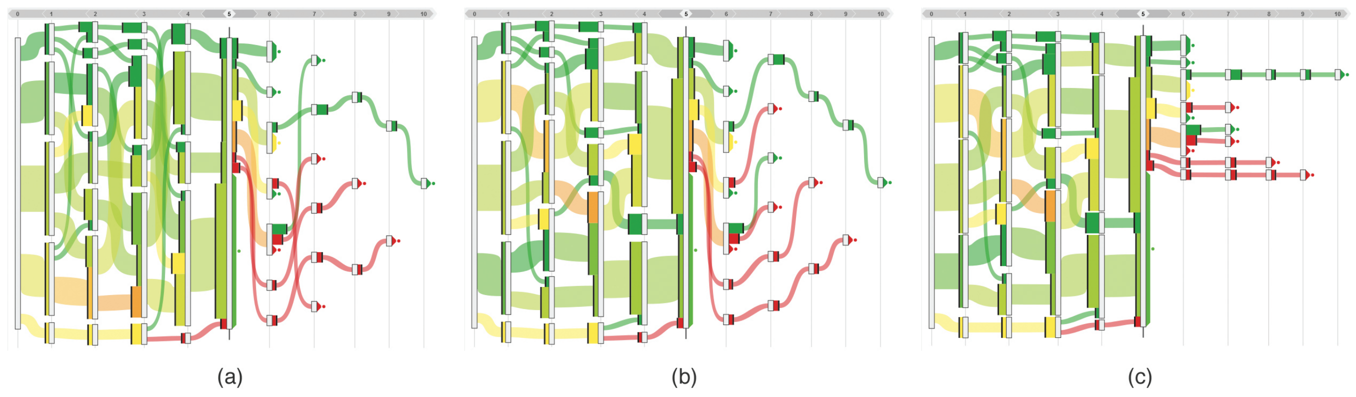


Fig. 6. A multi-stage layout process improves legibility. (a) Initial layout after sorting edges by outcome. (b) After applying Sugiyama's heuristics to reduce crossings. (c) The final visualization after both Sugiyama's heuristics and Outflow's force-directed algorithm to obtain straighter edges.

paths (Fig. 6b) that make the visualization more difficult to follow. To produce straighter paths, we apply a spring-based force-directed optimization algorithm [8] that reduces edge curvature. As illustrated in Fig. 5, nodes and edges are simulated as particles and springs, respectively. To prevent nodes from getting too close to each other, spring repulsions are also inserted between nodes. Through a series of iterations, nodes are gradually moved along the layer's vertical axis to reduce the spring tensions. The optimization stops either when the entire spring system is stable or when a maximum number of iterations has been reached. Fig. 6c shows the an improved visualization with straightened edges obtained by applying this force-directed algorithm.

4.3.4 Edge Routing

When large time differences exist between two alternative paths, time edges and link edges can overlap (Fig. 7a). This can make it more difficult for users to trace paths through the visualization during an analysis. We resolve this issue by routing link edges through an intermediate point that avoids crossings (Fig. 7b). The intermediate point is calculated by (1) finding the longest time edge from n neighbor edges in the direction traveled by a link edge (up/down), and (2) moving the origin of the link edge horizontally beyond the longest time edge's x position. This method does not guarantee avoidance for neighbors further than n. However, in practice, a low value of n (e.g., $n = 3$) provides effective reduction in overlaps without excessive edge routing.

4.4 Basic Interactions

To allow interactive data exploration, we further designed Outflow to support the following user interaction capabilities.

Panning & Zooming: Users can pan and zoom to uncover detailed structure.

Filtering: Users can filter both nodes and edges based on the number of associated entities to remove small subgroups.

Event Type Selection: Users can select which event types are used to construct the Outflow graph. This allows, for instance, for the omission of events that users deem uninteresting. For example, users in our medical use case can include/exclude a symptom (e.g., "Ankle Edema") if they deem it relevant/irrelevant to an analysis. In response, the visualization will be recomputed dynamically.

Brushing: Hovering the mouse over a node or an edge will highlight all paths traveled by all entities passing through the corresponding point in the Outflow graph (Fig. 8).

Tooltips: Hovering also triggers the display of tooltips which provide information about individual nodes and edges. Tooltips show all events associated with the corresponding node/edge, the average outcome, and the total number of entities in the subgroup (Fig. 8).

Pinning: Users can "pin" a node or edge to freeze the brushed selection. This interaction is performed by clicking on the element to be pinned. Users can then move the mouse pointer to display tooltips for brushed subsets. This allows the quick retrieval of information about

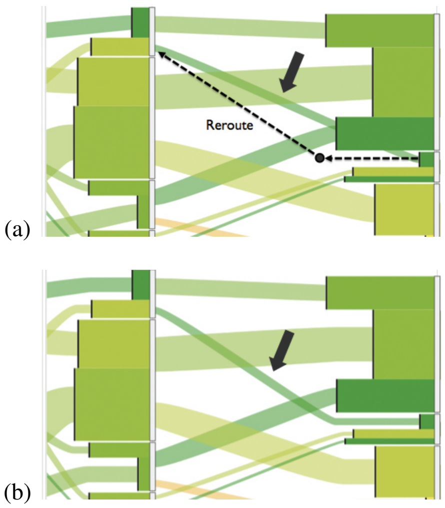


Fig. 7. Edge routing prevents overlaps between time and link edges. (a) A link edge is seen passing "behind" the time edge above it. Outflow's edge routing algorithm extends the link edge horizontally beyond the occluding time edge. (b) The new route avoids the overlap and makes the time edge fully visible.

subsets that satisfy two constraints. For example, a user in our soccer use case can pin soccer games that reached a 2-2 score before moving the mouse pointer to hover over the 1-0 state to see detailed information about the set of matches that pass through both nodes.

4.5 Simplification

The methods outlined earlier in this section can significantly reduce visual clutter and make the visualization more legible. However, there are still situations when visual complexity arises due to inherent complexity in the underlying data. To enable analyses of these more challenging datasets, Outflow includes a simplification algorithm that actively simplifies the underlying graph structure used to construct the visualization.

We apply a hierarchical clustering method that reduces the number of states in an Outflow graph by merging similar states within the same layer. States with a similarity distance less than a user-specified threshold are grouped together, while states that do not satisfy the threshold remain distinct. The user controls the threshold via a slider on the user interface. Our prototype defines the similarity distance between two states as the difference in average outcomes (Equation 1). $node_i.outcome$ denotes an average outcome within $node_i$. Alternative measures could be easily substituted (e.g., a similarity-based metric

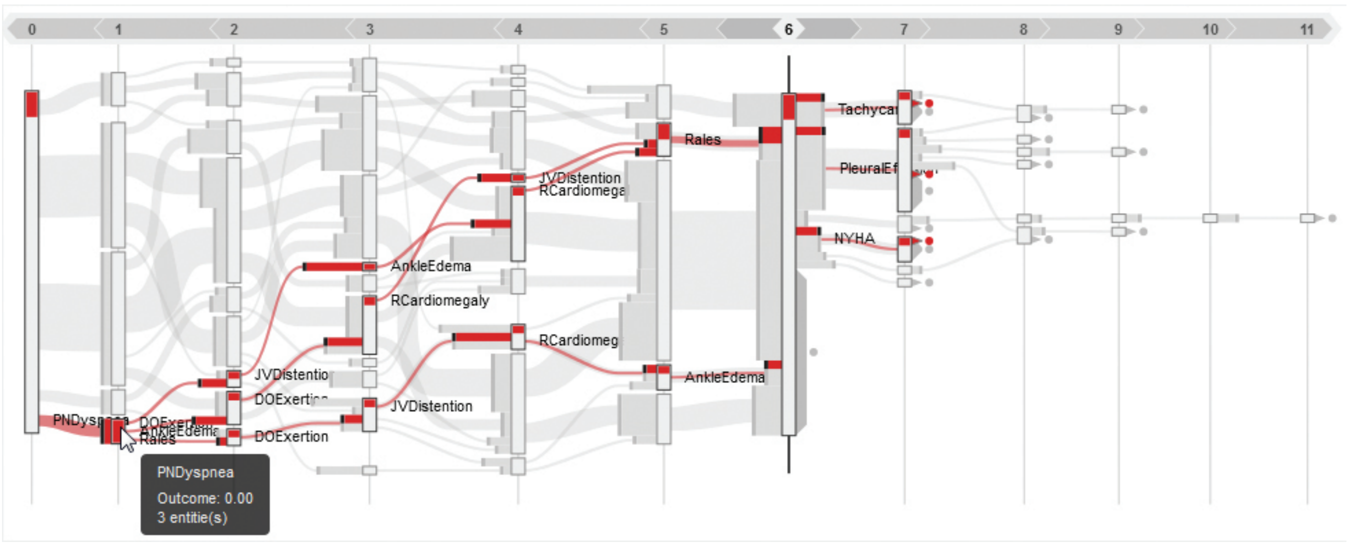


Fig. 8. Interactive brushing allows users to highlight paths emanating from specific nodes or edges in the visualization. This allows users to quickly see alternative progression paths taken by entities passing through a given state.

using application-specific properties of the underlying entities).

$$d(node_A, node_B) = |node_A.outcome - node_B.outcome| \quad (1)$$

The hierarchical process begins as follows. First, each state in a layer is assigned to its own cluster for which it is the only member. Then, distances between all pairs of clusters are computed. The distance between two clusters, defined in Equation 2, is determined by the average of the distances between all nodes in the first cluster to all nodes in the second cluster.

$$d(cluster_X, cluster_Y) = \frac{\sum_{m \in cluster_X \& n \in cluster_Y} d(m,n)}{size(cluster_X) * size(cluster_Y)} \quad (2)$$

Once the distances have been computed for all pairs of clusters in a given layer, clusters are merged in a greedy fashion. The most similar pair of clusters is merged, and the cluster distances are updated to reflect the new clustering. The greedy process repeats until either (1) only one cluster remains, or (2) the most similar pair of remaining clusters has a distance above the threshold specified by the user.

After the simplification process completes, clusters containing multiple states are rendered as a single node filled with gray in the visualization. To preserve state transition information, the edges are not merged even though their origin nodes or destination nodes may have been simplified (Fig. 9). Nodes that are the alone in their cluster are rendered using the normal Outflow visual encoding.

The simplification method can group similar states into clusters and therefore reduce the number of nodes and visual complexity. However, there is a limitation when the nodes cannot be clearly separated into distinct groups; the final clusters may not be semantically meaningful and may result in a less informative graph.

4.6 Factors

As described so far, Outflow provides an interactive visualization of event pathways and their associated outcomes. However, it does not yet incorporate external factors that can often influence how the events progress. For example, while goals determine the pathways in our soccer use case, yellow and red cards can have a major impact on how a game unfolds. Similarly, CHF patients' symptoms may be strongly influenced by the medications they are prescribed. In cases where they can be controlled, factors can be important clues to analysts in understanding how they can influence patients' progress.

Given the importance of factor analysis, we extend the basic Outflow data model to associate a set of timestamped factors (f_i, t_i) with each entity. Because of the timestamps, each occurrence of a factor can be placed within the sequence of events associated with the corresponding entity. The Outflow graph for a set of entities is constructed as before using only the entities' events. For each node and edge in the graph, we then compute additional statistics to identify correlated factors and suggest them to users via the user interface.

For our prototype, we have derived two metrics to detect factors that occur unusually often (or rarely) before a given state or transition. These metrics could be easily replaced with more sophisticated metrics that are more suitable for specific scenarios. In fact, we envision having a collection of metrics that users could choose from to measure various types of associations.

Our baseline metrics are inspired by the **term frequency-inverse document frequency** ($tf{\star}idf$) measure [24] used widely in information retrieval: *presence correlation* and *absence correlation*.

1. *Presence correlation* detects factors that are unusually frequent for a given state or transition. For states, if a factor f_i often occurs before reaching state S_j while f_i rarely occurs elsewhere in the dataset, then f_i will be given a high presence correlation value for the corresponding state. We measure this correlation using a **presence correlation** (C_p) score, which we define as follows.

$$R_p = \frac{\text{number of entities with } f_i \text{ before } S_j}{\text{number of entities reaching } S_j} \quad (3)$$

$$R_{sp}^{-1} = log\left(\frac{\text{number of states}}{1 + \text{number of states preceded by } f_i}\right) \quad (4)$$

$$C_p = R_p \cdot R_{sp}^{-1}$$

A similar calculation is made for transitions by substituting S_j with $T_{m \to n}$ in Equation 3 and replacing number of states with number of transitions in Equation 4. For R_{sp}^{-1}, we only count states (or transitions) for the current layer or earlier.

2. *Absence correlation* detects factors that are unusually rare for a given state or transition. For states, if a factor f_i rarely occurs before reaching state S_j while f_i occurs commonly elsewhere in the dataset, then f_i will be given a high absence correlation value for the corresponding state. This correlation is formulated as the **absence correlation** (C_a) score defined below.

$$R_a = \frac{\text{number of entities without } f_i \text{ before } S_j}{\text{number of entities reaching } S_j} \quad (5)$$

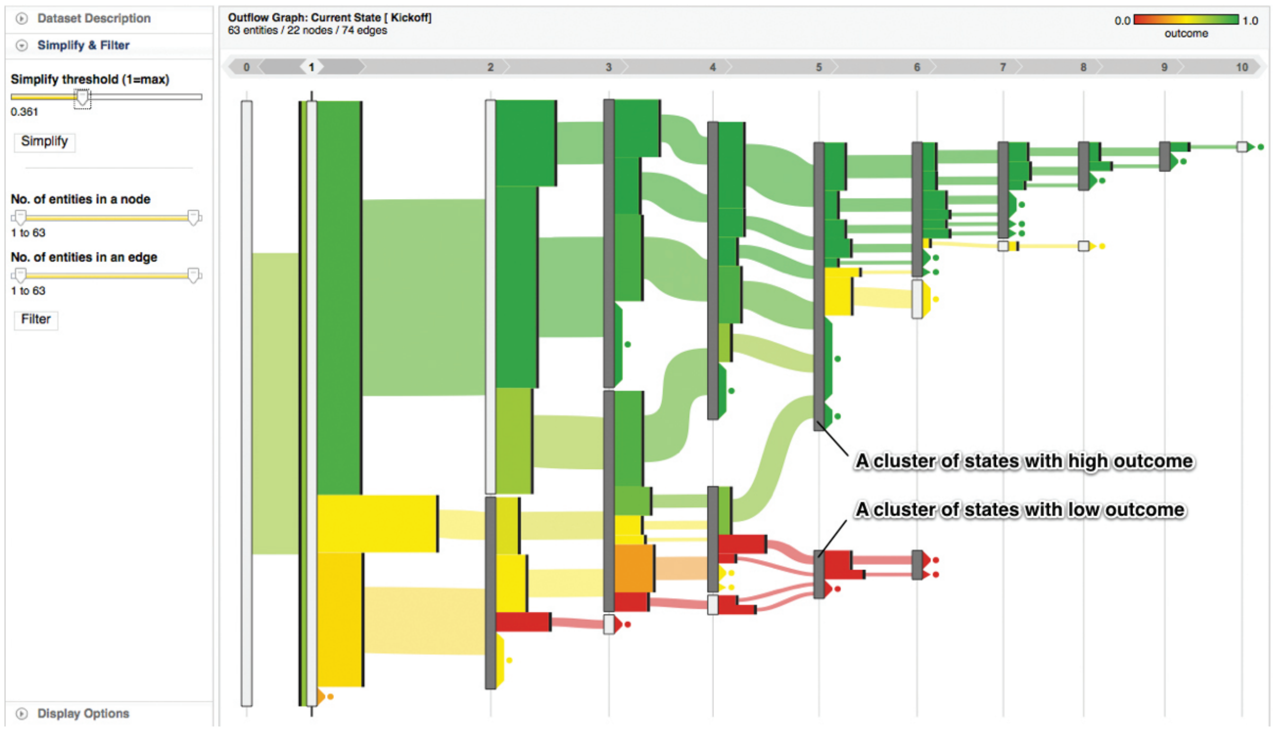

Dataset Description
Simplify & Filter
Simplify threshold (1=max)
0.361
Simplify
No. of entities in a node
1 to 63
No. of entities in an edge
1 to 63
Filter
Display Options
Outflow Graph: Current State [Kickoff]
63 entities / 22 nodes / 74 edges
0.0
1.0
outcome
0
1
2
3
4
5
6
7
8
9
10
A cluster of states with high outcome
A cluster of states with low outcome

Fig. 9. Using the same dataset as illustrated in Fig. 1, a user has adjusted the simplification slider to group states with similar outcomes. Clustered states are represented with gray nodes. This simplified view shows that as more events occur (i.e., as more goals are scored), the paths diverge into two distinct sets of clustered states. The simplified states more clearly separate winning scorelines from losing scorelines. As more goals are scored, the probable outcomes of the games become more obvious.

$$R_{sa}^{-1} = log\left(\frac{\text{number of states}}{1+\text{number of states not preceded by } f_i}\right) \quad (6)$$

$$C_a = R_a \cdot R_{sa}^{-1}$$

A similar calculation is made for transitions by substituting S_j with $T_{m\to n}$ in Equation 5 and replacing number of states with number of transitions in Equation 6. For R_{sa}^{-1}, we only count states (or transitions) for the current layer or earlier.

A new sidebar panel is added to the Outflow user interface to display these correlation scores. When users hover any time edge, the panel is updated to display the most highly correlated present and absent factors for the corresponding transition. The panel can be seen on the right side of Fig. 10. Factors are listed alphabetically and displayed with a histogram to convey the strength of the correlation.

5 User Study

We conducted a user study to evaluate Outflow's ability to support a variety of event sequence analysis tasks. We first describe the study's design which asked users to answer questions about Man U.'s 2010-2011 soccer season using a visualization of the taste described in Section 2.2. We then report the study's results and discuss its findings.

5.1 Design

We asked 12 users (eight males and four females) to participate in our study. All users were adult professionals who are comfortable with computers. None of the users would consider themselves "soccer experts", but all have a basic understanding of the game. None of the users had any prior experience using Outflow.

5.1.1 Procedure

Each user participated in a single 60-minute session during which they were observed by a study moderator. Each session started with a brief orientation in which the moderator explained Outflow's design and interactions. Participants were then allowed to experiment with the visualization to gain some experience working with the system.

After roughly 15 minutes, the formal section of the study began. Participants were asked to perform a list of tasks. While the tasks were performed, the moderator recorded both accuracy and time to completion for each task. Users were then asked to freely explore the data and describe any interesting findings. After that, users were given a written questionnaire to gather subjective feedback. Finally, we debriefed the participants to learn about their experience and any comments or suggestions they might have.

5.1.2 Tasks and Questionnaire

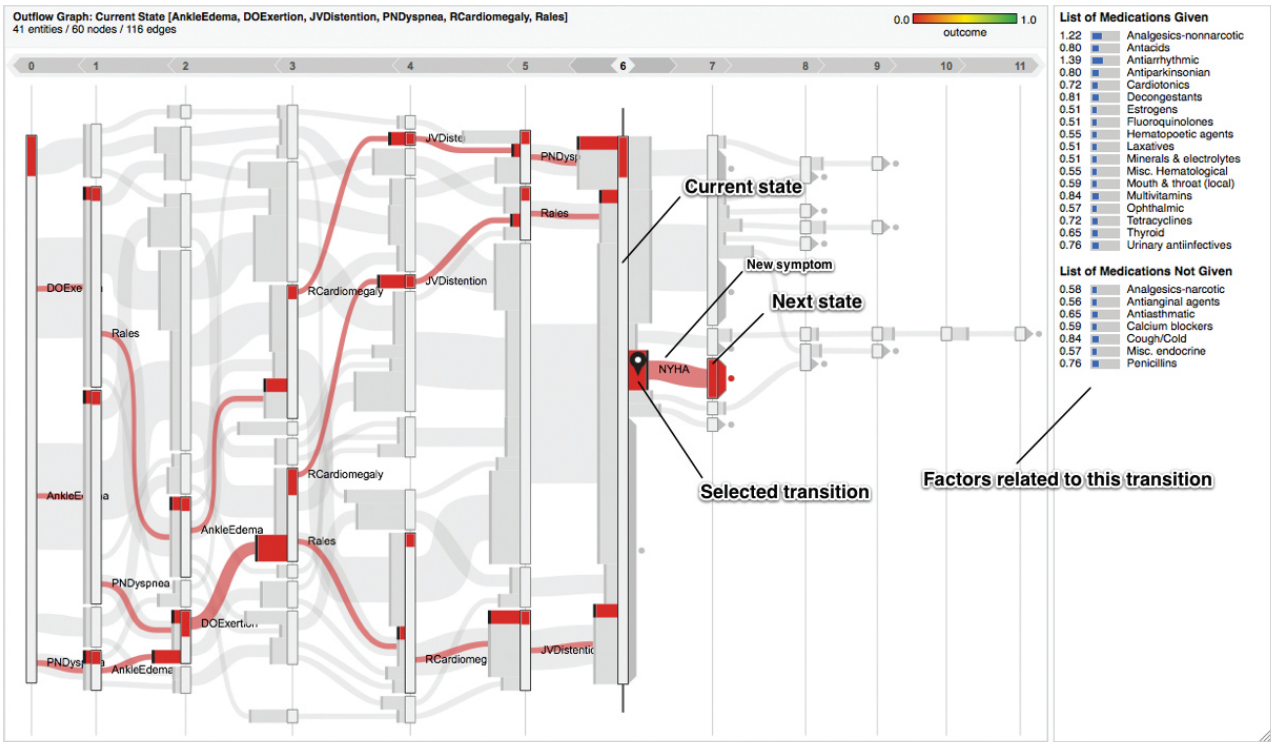
Each user was given a list of 16 tasks to perform. The tasks were designed to evaluate people's ability to understand proportion and cardinality of states and transitions, transition time, outcome of states and transitions, and factors associated with transitions.

The first nine tasks were designed to measure the participant's ability to interpret Outflow's visual representation. These tasks were further divided into two sets: practice tasks and test tasks. The first four tasks, unbeknownst to the participants, were used only to ensure that participants fully explored Outflow's visual design. Timing/accuracy data for these tasks was not included in our analysis. The five test tasks (T1–T5) asked questions similar to the preceding practice tasks, but on different aspects of the dataset.

The next seven tasks were designed to measure performance when using Outflow's interactive capabilities. Once again, these tasks were split into two groups: practice tasks and test tasks. The first three were designed to give participants practice using Outflow's interactive features and results were not included in our analysis. The four remaining tasks (T6–T9) asked similar questions about other aspects of the dataset. The test tasks used in our study are as follows:

T1. "Can you find the state where Man U. conceded the first goal?" *Objective*: Traverse graph using labels.

T2. "What happened most rarely after Man U. conceded the first goal?" *Objective*: Interpret proportion from height of time edge.



List of Medications Given

1.22	Analgesics-nonnarcotic
0.80	Antacids
1.39	Antiarrhythmic
0.80	Antiparkinsonian
0.72	Cardiotonics
0.81	Decongestants
0.51	Estrogens
0.51	Fluoroquinolones
0.55	Hematopoetic agents
0.51	Laxatives
0.51	Minerals & electrolytes
0.55	Misc. Hematological
0.59	Mouth & throat (local)
0.84	Multivitamins
0.57	Ophthalmic
0.72	Tetracyclines
0.65	Thyroid
0.76	Urinary antiinfectives

List of Medications Not Given

0.58	Analgesics-narcotic
0.56	Antianginal agents
0.65	Antiasthmatic
0.59	Calcium blockers
0.84	Cough/Cold
0.57	Misc. endocrine
0.76	Penicillins

Fig. 10. Outflow highlights factors that are strongly correlated with specific event pathways. In this screenshot, a physician has focused on a group of patients transitioning from the current state due to the onset of the "NYHA" symptom. This transition seems to be deadly, as seen from the color-coding in red. The right sidebar displays medications (factors) with high correlations to this transition. The factor with the highest correlation in this example is prescribing antiarrhythmic agents. This correlation, which may or may not be causal, can help clinicians generate hypotheses about how best to treat a patient.

T3. "Was it faster to concede a goal or to score a goal from the state in T1?" *Objective*: Interpret time from time edge width.

T4. "Was it more likely for Man U. to win or lose after the state in T1?" *Objective*: Interpret state outcome.

T5. "What is the most common score after two goals are scored (2-0, 1-1 or 0-2)?" *Objective*: Traverse graph and interpret proportion from state node height.

T6. "Can you find the state where Man U. led 2-1?" *Objective*: Traverse graph using labels.

T7. "Which transition from the state in T6 led to the lowest percentage of winning? How many percent?" *Objective*: Use tooltip.

T8. "Which factor(s) are highly correlated with the transition in T6?" *Objective*: Understand factors.

T9. "For games that reached 2-2, which situation resulted in better outcomes? (a) Scoring to move from down 1-2 into a 2-2 tie or (b) conceding to move from a 2-1 lead to a 2-2 tie?" *Objective*: Use tooltip. The actual data shows a 0.73 outcome for (a) and a 0.67 outcome for (b). While the similar outcome values make the difference in color-coding hard to see, the tooltip provides access to the numerical outcome statistics.

At the end of their study session, participants completed a questionnaire. It contained eight questions (Q1–Q8) to which users responded on a 7-point Likert scale (1=very easy, 7=very hard). The questionnaire also included free response questions to gather more subjective feedback. The eight Likert-scale questions included the following:

Q1. Is it easy or hard to learn how to use?

Q2. Is it easy or hard to interpret proportion of states?

Q3. Is it easy or hard to interpret proportion of transitions?

Q4. Is it easy or hard to interpret transition time?

Q5. Is it easy or hard to interpret outcome of states?

Q6. Is it easy or hard to interpret outcome of transitions?

Q7. Is it easy or hard to understand factors correlated with transitions?

Q8. Is it easy or hard to find a particular state in the graph?

5.2 Results

5.2.1 Accuracy

Overall, participants were able to complete the tasks accurately with only three mistakes observed out of 108 total tasks (97.2% accuracy).

Two users erred on T4. These participants were able to identify the node that was the focus of the question. However, neither responded based on the color of the identified node. One user looked at all of the future paths and mentally aggregated/averaged the values (incorrectly) to guess at the eventual outcome. This approach is subject to bias because long paths cover more pixels even if their outcome is not representative. When told of their mistake, the user said that looking at the color of the node "would have been trivial. I just forgot." The second user who answered T4 incorrectly responded based on the color of the largest outbound path. While that path corresponds to the most often occurring next event, it does not by itself represent the overall average outcome for the identified node. We hypothesize that both errors were due in large part to the users' lack of experience.

The only other error occurred on T9. However, during the questionnaire portion of the study session, it was determined that the user misunderstood the task. He actually used Outflow correctly to answer what he thought the question was asking.

5.2.2 Speed

Participants were able to finish the tasks rapidly with average completion times ranging between 5.33 to 64.22 seconds.[2] The wide range

[2]Task completion times were as follows: (T1) Mean=5.33 ± SD=4.18 s; (T2) 8.79 ± 7.09 s; (T3) 8.16 ± 7.64 s; (T4) 26.26 ± 14.39 s; (T5) 49.53 ±

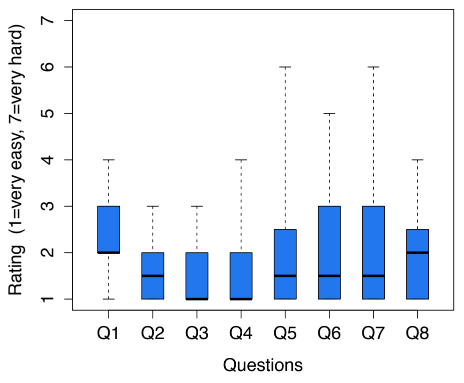


Fig. 11. Questionnaire results for each of the eight questions (Q1–Q8) answered on a 7-point scale by our study participants.

in timings reflects in part variations in task complexity. However, the standard deviations between timings for individual tasks were fairly large (up to 30.47 s). In general, some users quickly mapped tasks to a feature of the visualization while others spent more time thinking about what exactly the question was asking before settling on an answer. Overall, the times are quite fast and we believe that allowing novice users to precisely answer a complex task like T9 in as little as one minute highlights Outflow's utility.

5.2.3 Unrestricted Exploration

After completing the study tasks, users were given time to freely explore the visualization. Participants were able to quickly identify several interesting aspects of the dataset. For example, many users quickly found the highest scoring game which had a total of eight goals. Participants were also drawn to states that had multiple "arriving" edges and compared outcomes for the alternative paths. This is similar to what they were asked to do in T9. One user discovered a general trend that Man U. rarely received yellow cards when they lost. This could be interpreted as a sign of lack of intensity in those games.

Users also enjoyed "the ability to investigate the progression of a large number of games on one screen." Users observed a number of strong global trends, including: (1) Man U. wins a large majority of their games, (2) Man U. wins most high scoring games, (3) Man U. often comes back after falling behind, and (4) Man U. rarely loses a game when ahead.

5.2.4 Questionnaire and Debriefing

Results from the questionnaire are shown in Fig. 11. A pairwise t-test between all questions shows no significant difference between the ratings of each question. Average ratings from all questions are between 1.50–2.33, suggesting that the participants generally found Outflow easy to learn and easy to use.[3]

However, not all participants responded in the same way. As shown in Fig. 11, there were a small number of higher ratings for questions Q5-Q7 that indicate some frustration. In fact, all of the high scores (> 4) came from a single participant who had difficulty understanding the difference between the outcome of a *state* and the outcome of a *transition*. He repeatedly asked for clarification and commented in our questionnaire that tasks on these topics required "some effort to parse the wording." The moderator explained that while a state (e.g., a tied score of 1-1) has one overage outcome, there can be multiple transitions to that state (e.g., 1-0 $\rightarrow$ concede $\rightarrow$ 1-1 vs. 0-1 $\rightarrow$ score $\rightarrow$ 1-1). The average outcomes for these transitions can be different (e.g., 1-0 $\rightarrow$ concede $\rightarrow$ 1-1 $\approx$ 67% win vs. 0-1 $\rightarrow$ score $\rightarrow$ 1-1 $\approx$ 73% win). The user understood for a moment, but then quickly became confused again. This led to his high responses and slower task completion times.

Based on free response questions and interviews with the moderator, participants felt overall that Outflow was "pretty", "looks cool", and that the colors were "very meaningful". They said that it provides a "good high level view" that encodes a lot of information into a single "simple to follow" visualization. In addition, users felt that the ability to see outcome associated with alternative paths out of (or into) a given state, and not just the state itself, is a powerful feature. "I like the difference between states and transitions [which allowed me] to compare two paths to the same state and to understand differences." Another participant commented that "highlighting of paths was very helpful."

When asked about learning to use Outflow, some participants expressed that the tool is unique, and therefore required some training to get used to it. However, those participants also felt strongly that by the end of the study they were proficient in using the tool. One remarked that they would have done much better on the study tasks "with a few more minutes of training at the start" suggesting a short learning curve.

Some limitations of the current design were also identified during the study. One participant pointed out that users tend to view width as time, but that the widest sequences don't necessarily take the longest (in fact, soccer games all take roughly the same amount of time). This is indeed a limitation of the technique. Outflow handles graphs with multiple incoming edges to a node. Because these different paths to a node can have different durations, time can not be represented horizontally using an absolute time axis. While making comparisons between alternative paths easier, this design choice can make temporal comparisons across multiple steps somewhat harder.

Participants also suggested ideas for new features including (1) the ability to pin multiple states at once, and (2) moving the display of correlated factors into the main visualization space (instead of the separate sidebar used in the current design). We agree, and we are planning to explore design alternatives for these features in future work.

6 Conclusions and Future Work

This paper presented *Outflow*, a visualization that enables interactive analysis of event sequence collections. Outflow aggregates multiple event sequences and displays aggregate event pathway information including timing and cardinality. It uses color-coding to depict the outcomes associated with each pathway. Outflow also allows users to explore how external factors correlate with specific states and transitions. Two simple metrics for factor recommendation were presented. We provided a detailed description of the visualization's design including a multi-step layout process designed to reduce visual complexity. A portion of this process—the combination of Sugiyama's algorithm to reduce edge crossings and force-directed layout to straighten unnecessarily curvy edges—is also applicable to a more general class of DAG layout problems. Results from a user study with twelve participants were provided which demonstrated that users were able to learn how to use Outflow easily within fifteen minutes of training, and were able to accurately and quickly perform a broad range of analysis tasks.

While the initial results from our study are promising, there are many possible topics to explore in future work. This includes the integration of Outflow with forecasting/prediction algorithms, the use of additional layout techniques to reduce visual clutter, simplifying the graph by aggregating similar paths, more advanced factor analysis and expanded interaction capabilities. Furthermore, we understand the limitations of the user study in this paper, which aims to test the usability and learnability of Outflow. Additional evaluation studies, such as deeper evaluations with domain experts and formal comparisons to alternatives are needed.

Acknowledgments

The authors wish to thank Charalambos (Harry) Stavropoulos, Robert Sorrentino and Jimeng Sun for their help and suggestions. We also appreciate comments from Sureyya Tarkan, Kostas Pantazos, Cody Dunne, John Alexis Guerra Gómez, Megan Monroe, Catherine Plaisant and Ben Shneiderman.

30.47 s; (T6) 10.19$\pm$ 3.01 s; (T7) 16.98 $\pm$ 10.81 s; (T8) 7.98 $\pm$ 4.01 s; (T9) 64.22$\pm$26.92 s

[3]Question ratings were as follows: (Q1) Mean=2.33$\pm$ SD=0.78; (Q2) 1.58$\pm$0.67; (Q3) 1.5$\pm$0.67; (Q4) 1.58$\pm$0.90; (Q5) 2.08$\pm$1.56; (Q6) 2.17$\pm$1.53; (Q7) 2.25$\pm$1.71; and (Q8) 2$\pm$0.953

References

[1] W. Aigner, S. Miksch, B. Thurnher, and S. Biffl. PlanningLines: Novel Glyphs for Representing Temporal Uncertainties and Their Evaluation. In *Proceedings of the International Conference on Information Visualization (IV)*, pages 457–463. IEEE, 2005.

[2] P. André, M. L. Wilson, A. Russell, D. A. Smith, A. Owens, and M. Schraefel. Continuum: designing timelines for hierarchies, relationships and scale. *Proceedings of the Annual ACM Symposium on User Interface Software and Technology (UIST)*, page 101, 2007.

[3] R. Bade, S. Schlechtweg, and S. Miksch. Connecting time-oriented data and information to a coherent interactive visualization. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 105–112. ACM, 2004.

[4] J. Blaas, C. P. Botha, E. Grundy, M. W. Jones, R. S. Laramee, and F. H. Post. Smooth graphs for visual exploration of higher-order state transitions. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):969–76, 2009.

[5] T. Booth. *Sequential Machines and Automata Theory*. John Wiley and Sons, New York, NY, USA, 1967.

[6] M. Burch, F. Beck, and S. Diehl. Timeline trees: visualizing sequences of transactions in information hierarchies. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 75–82, Napoli, Italy, 2008. ACM.

[7] S. B. Cousins and M. G. Kahn. The visual display of temporal information. *Artificial Intelligence in Medicine*, 3(6):341–357, 1991.

[8] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, Oct. 1994.

[9] J. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A Visual Interface for Multivariate Temporal Data: Finding Patterns of Events across Multiple Histories. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, volume 0, pages 167–174. IEEE, Oct. 2006.

[10] M. Friendly. Visions and re-visions of Charles Joseph Minard. *Journal of Educational and Behavioral Statistics*, 27(1):31–51, 2002.

[11] B. L. Harrison, R. Owen, and R. M. Baecker. Timelines: an interactive system for the collection and visualization of temporal data. In *Proceedings of Graphics Interface (GI)*, pages 141–141. Citeseer, 1994.

[12] G. M. Karam. Visualization using timelines. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 125–137. ACM, 1994.

[13] R. Kosara, F. Bendix, and H. Hauser. Parallel sets: interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–68, 2006.

[14] P. A. McKee, W. P. Castelli, P. M. McNamara, and W. B. Kannel. The natural history of congestive heart failure: the Framingham study. *The New England journal of medicine*, 285(26):1441–6, Dec. 1971.

[15] J. J. McMurray and M. A. Pfeffer. Heart failure. *Lancet*, 365(9474):1877–1889, 2005.

[16] P. Mui. Introducing Flow Visualization: visualizing visitor flow, 2011.

[17] C. A. Petri. Communication with automata. Technical report, DTIC Research Report, 1966.

[18] D. Phan, A. Paepcke, and T. Winograd. Progressive multiples for communication-minded visualization. In *Proceedings of Graphics Interface (GI)*, page 225, New York, New York, USA, 2007. ACM.

[19] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 219–224. IEEE, 2005.

[20] C. Plaisant, R. Mushlin, A. Snyder, J. Li, D. Heller, and B. Shneiderman. LifeLines: using visualization to enhance navigation and analysis of patient records. In *Proceedings of the AMIA Annual Symposium*, pages 76–80. American Medical Informatics Association, 1998.

[21] A. J. Pretorius and J. J. van Wijk. Visual analysis of multivariate state transition graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):685–92, 2006.

[22] A. J. Pretorius and J. J. van Wijk. Visual Inspection of Multivariate Graphs. *Computer Graphics Forum*, 27(3):967–974, May 2008.

[23] P. Riehmann, M. Hanfler, and B. Froehlich. Interactive Sankey diagrams. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 233–240. IEEE, 2005.

[24] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[25] K. Sugiyama, S. Tagawa, and M. Toda. Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[26] S. van den Elzen and J. J. van Wijk. BaobabView: Interactive construction and analysis of decision trees. In *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 151–160. Ieee, Oct. 2011.

[27] F. van Ham, H. van De Wetering, and J. J. van Wijk. Interactive visualization of state transition systems. *IEEE Transactions on Visualization and Computer Graphics*, 8(4):319–329, Oct. 2002.

[28] K. Vrotsou, K. Ellegard, and M. Cooper. Everyday Life Discoveries: Mining and Visualizing Activity Patterns in Social Science Diary Data. In *Proceedings of the International Conference on Information Visualization (IV)*, pages 130–138. IEEE, July 2007.

[29] K. Vrotsou, J. Johansson, and M. Cooper. ActiviTree: interactive visual exploration of sequences in event-based data using graph similarity. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):945–52, 2009.

[30] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 457–466. ACM, 2008.

[31] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI)*, page 811. ACM, 2006.

[32] K. Wongsuphasawat and D. Gotz. Outflow: Visualizing patient flow by symptoms and outcome. In *IEEE VisWeek Workshop on Visual Analytics in Healthcare*, pages 25–28, 2011.

[33] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. LifeFlow: Visualizing an Overview of Event Sequences. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 1747–1756. ACM, 2011.

[34] K. Wongsuphasawat, C. Plaisant, M. Taieb-Maimon, and B. Shneiderman. Querying Event Sequences by Exact Match or Similarity Search: Design and Empirical Evaluation. *Interacting with computers*, 24(2):55–68, Mar. 2012.

[35] K. Wongsuphasawat and B. Shneiderman. Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 27–34. IEEE, Oct. 2009.