Feature-Action Design Patterns for Storytelling Visualizations with Time Series Data

S. Khan[1], S. Jones[2], B. Bach[3], J. Cha[2], M. Chen[2], J. Meikle[2], J. C. Roberts[4], J. Thiyagalingam[2], J. Wood[5], P. D. Ritsos[4]
[1]Science and Technology Facilities Council (STFC), UK, [2]University of Oxford, UK
[3]University of Edinburgh, UK, [4]Bangor University, UK, [5]City University, UK

Abstract

We present a method to create storytelling visualization with time series data. Many personal decisions nowadays rely on access to dynamic data regularly, as we have seen during the COVID-19 pandemic. It is thus desirable to construct storytelling visualization for dynamic data that is selected by an individual for a specific context. Because of the need to tell data-dependent stories, predefined storyboards based on known data cannot accommodate dynamic data easily nor scale up to many different individuals and contexts. Motivated initially by the need to communicate time series data during the COVID-19 pandemic, we developed a novel computer-assisted method for meta-authoring of stories, which enables the design of storyboards that include feature-action patterns in anticipation of potential features that may appear in dynamically arrived or selected data. In addition to meta-storyboards involving COVID-19 data, we also present storyboards for telling stories about progress in a machine learning workflow. Our approach is complementary to traditional methods for authoring storytelling visualization, and provides an efficient means to construct data-dependent storyboards for different data-streams of similar contexts.

CCS Concepts

• ***Human-centered computing*** → ***Visualization systems and tools; Visualization techniques;*** • ***Computing methodologies*** → *Feature selection;*

1. Introduction

Visualization provides a powerful means for telling stories about data [KM13, GP01, TRB*18, LRIC15]. In most workflows, the authors and the developers of storytelling visualization are typically given a complete dataset that is not expected to change after the storyboard is constructed (e.g., [SXS*21, OBCT24]). Figure 1(a) illustrates such a workflow. With the rapid growth of data volume and context, it is increasingly important to deliver data-driven storytelling visualization to different audiences. Moreover, individuals will likely pay more attention to stories relevant to them, such as those about happenings close to their location, information affecting their decisions, or historical facts interesting to them.

Such requirements became strikingly noticeable during the COVID-19 pandemic. Institutions, such as governments and news outlets, were able to transfer data at the national level to stories effectively. Nevertheless, many members of the public found it difficult to access information of more immediate interest, such as finding out what happened in their location, checking recent data related their planned journeys, or comparing historical data between regions of interest. It is not scalable to author many different storyboards for every individual region or every pair of regions. Nor is it scalable to change storyboards manually, whenever a dynamic dataset is updated with newly-arrived data. Moreover, having multiple storyboards and changing them frequently can be tedious for developers who implement storytelling visualization software.

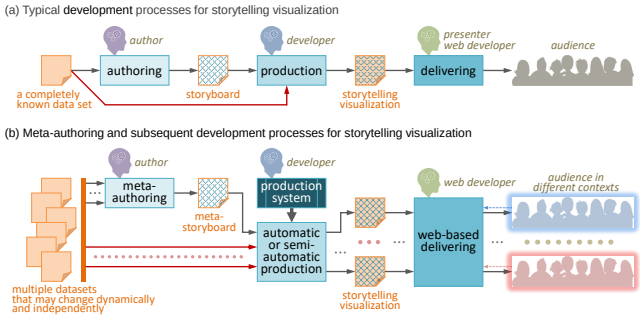


Figure 1: *(a) In a typical workflow for creating storytelling visualization, an author defines a storyboard for a known dataset, which is then developed as a web-based visualization, usually for a specific target audience. (b) With our approach, the author creates a meta-storyboard that works with multiple, dynamic, and often not-yet-inspected datasets. The storyboard is converted by a developer, following rules that facilitate the automatic or semi-automatic depiction of user-driven stories, for different target audiences.*

As illustrated in Figure 1(b), ideally, authors of storytelling vi-

sualization could construct a common storyboard, for multiple datasets that may change dynamically and independently. When the common storyboard is applied to multiple datasets at a particular moment (e.g., COVID-19 data streams for different regions in a given period), there could be an efficient mechanism to generate different storytelling visualization for audiences in different contexts (e.g., in different regions). We refer to such a common storyboard as a *meta-storyboard*, while we refer to the process of creating it as *meta-authoring* (outlined in Figure 2).

There are many technical challenges in enabling the development workflow in Figure 1(b), including devising software tools to support meta-authoring, mechanisms for mapping meta-storyboards to different datasets to generate different stories, and software platforms to deliver the resulting storytelling visualizations to audiences. Example outputs are frequently updated storytelling visualizations for different regions and/or periods, both selected by the users. In such cases, if a developer had to read a storyboard and manually transform it to a storytelling visualization for each data stream and whenever new data arrives, it would not be efficient or even feasible as the same process had to be repeated again and again. It is thus desirable to have an automatic or semi-automatic mechanism to combine a meta-storyboard with datasets that may be changing.

While such difficulties may be caused by the differences between datasets and time frames, many data patterns in these datasets are expected to be similar. A meta-storyboard is defined for a group of notionally-similar datasets, such as the time series of COVID-19 daily cases in different regions. Moreover, similar data patterns can be depicted using similar visual patterns. The key to developing an efficient mechanism for combining a meta-storyboard with individual datasets is to define, implement, and apply design patterns for storytelling visualization [BSB*18]. To avoid overusing the word "pattern", we use the term "data feature" in place of "data pattern", and "visualization action" in place of "visual pattern". Hence the design patterns in storytelling visualization are patterns of mapping feature to action, i.e., "feature-action" design patterns.

As part of the RAMPVIS project [CARA*22, DARA*22] for providing a variety of visualization capabilities to support epidemiological modelling, a small team focused on devising novel techniques for storytelling visualization, which enables meta-authoring, by providing a production workflow as shown in Figure 1(b). Our method is a novel addition to the emerging set of methods for automated storytelling (e.g., [WSZ*20, SKH*22]), and offers the first scalable solution to the meta-authoring of stories about multiple dynamic time series datasets. While our approach was developed in the context of the COVID-19 pandemic, it is generalisable beyond this context, e.g., for visualising stories on carbon emissions, plastic waste, personal financial spending, and so on. To demonstrate the feasibility, we repurposed the software to provide storytelling visualization of machine learning (ML) workflows.

Therefore, our contributions are: (i) meta-authoring as a new method for creating storyboards for dynamic multi-stream data, (ii) an algorithmic pipeline for using pre-defined feature-action patterns for realising meta-storyboards in response to unseen data (accompanied by supplementary material), and (iii) six storyboards of two case studies, as demonstration.

2. Related Work

Storytelling and visualization have a long history [TRB*18], helping to reveal information in ways that are intuitive and compelling [GP01]. Employed techniques are wide ranging from sketching [LKS13], slide shows [HDR*13], comics [WRC*21] to investigating specific tasks in storytelling, such as linking [ZOM19], collaboration [LRIC15], immersion [WFRR20] and learning methods [TLW*21].

2.1. Personalising Information Visualization

As many of our daily activities are mediated by some form of interactive technology, recording, sharing and utilising data and related information about said activities has become more prevalent. In particular, data-driven information that can be personal, i.e., concern us either as individuals or some identifiable collective, is already driving our decision making. For instance, Yousuf and Conlan explore personal visual-learning narratives [YC18]. During the COVID-19 pandemic, the degree of the technological mediation increased, either as we started working and socialising more online, or as we tried to follow the pandemic's progression through visualizations on public media. Looking at daily, national or local infection rates, became a daily activity which often determined our daily or monthly routine and choices.

Nevertheless, most of the visualizations that we had access to provided either none or limited functionality, when it came to enabling personalised points of view. For instance, dashboards such as the John Hopkins COVID-19 map [Cen22] provided only a world and national overview, whereas Governmental portals, such as the UK's Coronavirus dashboard [UK 22] allowed the selection of locations (via post code) but did not provide any form of contextualisation, or a story. This is also true for visualization interfaces that presents data on world events, which nonetheless affect our day to day lives (e.g., world crises and impact on food pricing) and often remain on national level averages.

Inevitably, providing more personalised storytelling requires: a) the availability of data for a more personal point of view, e.g., specific to a location, person, communities etc. and b) mechanisms to involve the users, and allow them to tailor their queries, explore the resulting visual depictions, and involve and engage them on a suitably personal level. Our work takes steps towards this direction.

2.2. Storytelling Concepts (especially Temporal Data)

Data-driven storytelling aims to provide a curated lens on evidence in data. It does so through contextual information, a curated set of messages, compelling narrative devices [SH14], and often a sequential ordering of information. It employs narrative patterns [BSB*18], such as *gradual reveal*, or *juxtaposition*, and communicates through a set of genres [SH10] such as videos [ARL*16], Data GIFs [SWT*20], slideshows [HDR*13, WLF*19], or data comics [BWF*18]. Visual narratives help to explain ideas [RBSN22], which are especially useful in education settings [YC18]. While studying data comics, Wang et al. [WWF*19] found that by breaking down the complexity of information into individual steps, and presenting them sequentially (i.e., comics),

readers could follow the story better, compared to infographics or illustrated texts. Feature-action data patterns create such sequences which a reader can navigate forwards and backwards.

A key point in storytelling is the relation between the story and the audience. In author-driven stories, narration, sequence, and content are defined by the author of the story, with little to no agency from the audience. In reader-driven storytelling, readers gain agency over the story and consequently can personalise it. Agency usually is achieved through interaction, e.g., by navigating within the story, data selection or free exploration [WRC*21]. However, one of the drawbacks of interactive storytelling is the discoverability of interaction affordances [Cox11, BEDF15], risking to impede personalised storytelling [MRL*17]. With our approach, we reduce the need for interaction and customization by creating prescribed stories for a given set of input data. As shown in Section 3, we keep interaction to a minimum and instead provide stories at different levels of granularity. While the sequence, narration, and contents is fixed, said granularity, which is informed by our algorithm (Section 4), defines how many messages the story has.

Storytelling of temporal data has a long tradition as demonstrated through two comprehensive surveys [RG13, BLB*17] which feature many historic examples of time-*lines*. Through their sequential nature, temporal data lends itself particularly well to sequential and author-driven storytelling [WaCP*15]. However, these bespoke examples make generalisation to other datasets hard. Commercial tools for creating visual timelines include TimelineJS [Nor18] and most recently Timeline Storyteller [BLB*17]. Likewise, Timeline Curator supports the creation of visual timelines from text, extracting data from the latter, and enabling the curation of the visual timeline [FBM15].

2.3. Automation in Storytelling

A good number of systems allow authors to create storytelling visualizations interactively, e.g., [OBCT24, MBS24]. Meanwhile automation for storytelling aims at increasing the access to information by lowering the burden for humans to create bespoke stories. Some systems aim more at supporting analysts gaining an overview over their data and personalising their interest through interaction [SXS*21], whereas other systems aim to support the creation of effective and compelling *communication* material [WSZ*20]. Other systems deal with fact extraction, as by Law et al. [LES20, SyzZW24]. Facts can be selected based on statistics and ordered based on a ranking score representing subjective relevance, interesting-ness, and importance [SXS*21, WSZ*20]. User interaction can enable dynamic data selection [WGH*24]. Our method works in a similar manner in that it provides a specific routine for determining specific features, while ranking is defined by a Gaussian function. To the best of our knowledge, all these systems explicitly focus on tabular data. In this paper, we consider storytelling visualization of temporal data.

Storytelling systems may group visual information into panels based on category [DHPP17], create dashboards [WWZ*21], or generate infographic-like fact sheets of visualizations around topics, featuring textual explanations [WSZ*20]. To organise data facts into sequences, graph-based approaches create a similarity graph from all facts and consequently select a sequence of visualizations (a path through the graph) based on minimising edge weight [YLRC10, KWHH17]. Data facts can also be ordered using a logic-oriented Monte Carlo tree search algorithm [SXS*21].

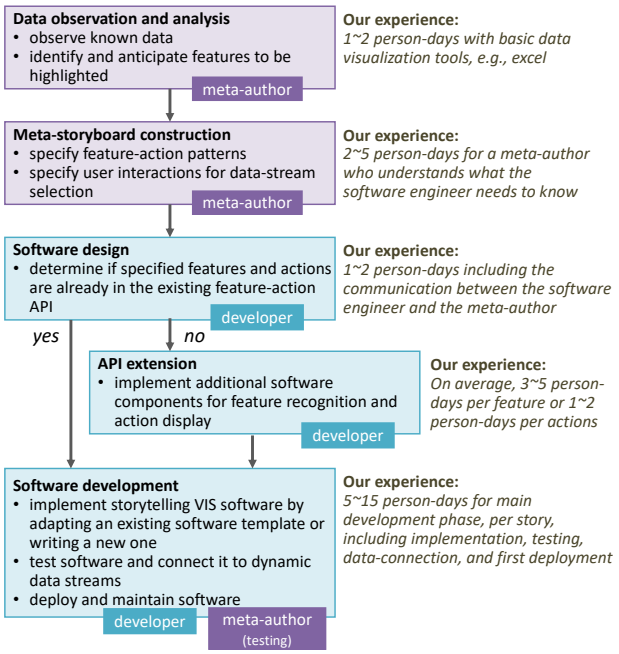


Figure 2: *Overview of the proposed meta-authoring process, involving a story meta-author and a developer. The resulting storytelling visualization software can be applied to many similar, dynamic, and often not-yet-inspected data streams.*

Closest to us, Parry et al. [PLC*11] presented an automated method to extract important frames from a video, for composing a sports game summary storyboard. They used a Gaussian mixture model to estimate the importance of individual frames. We adapted their technique for time series data. With our approach, data facts are extracted from the time series through analysis as well as complemented through public information on policy decisions. A Gaussian distribution defines which facts are shown for each given timeline and user-defined granularity. Consequently, timelines are build up step-by-step, progressing through identified breakpoints, while contextual information is displayed in textual labels. For our data, the narrative sequence follows the temporal order of breakpoints.

3. Meta-authoring

Our meta-authoring process enables story authors to define narratives that can be applied in many similar, dynamic, and often not-yet-inspected datas treams. As outlined in Figure 2, a meta-author needs to (1) explore the data; (2) turn specific story items into generalised data features F, and (3) map features to actions A. A developer then uses feature-action APIs to transform a meta-storyboard into a piece of generalised visualization software.

Firstly, a meta-author inspects some known data and ascertains key **features** F anticipated in the data. One important challenge that meta-authors face is to cope with the quantity, range of, and

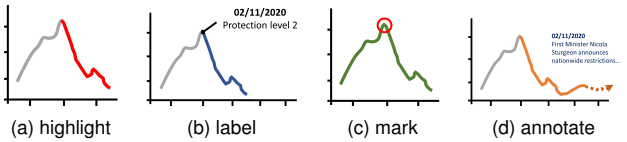


Figure 3: *Actions in our implementation with their parameters: (a) change the colour of a section of the time series line for highlighting. (b) draw a circle at a datapoint. (c) annotate the graph with a line attached at a particular point. (d) While animating a time series segment, place a text description on the opposite half of the graph. It is possible to only animate a time segment or animating a segment and annotating a point.*

precise values of the available data streams. Unlike in designing traditional storytelling visualization – where authors and developers would have immediate access to the data – in meta-authoring, a meta-author may not have (or do not need to have) seen all data in detail. Hence, it is critical for the meta-author to be able to anticipate the potential features that the data may or may not have, and create a story specification that can work either way.

Subsequently, the storyteller needs to express the **features** F in a descriptive and generalisable form, that allows the detection of said features without requiring knowledge of when (or whether) they occur in each data stream. An example would be "*find if the data in each month featuring a rising slope above 15°*".

Finally, the meta-author needs to consider what **actions** A occur for each encountered feature. Each action may highlight the feature visually and display a narrative (e.g., showing values and predefined text strings). A few examples of implemented actions can be seen in Figure 3. In the following sections, we outline three use cases from the RAMPVIS project, created with our meta-authoring process: (1) a story from single location with one time-series, (2) a comparison story, and (3) a scrollable time-line story.

3.1. Single location and time Series

The first story (Figure 4) focuses on one location, the quantity of positive COVID-19 cases and major events such as lockdowns, vaccination program progress, etc. The story involves two time series: a) COVID-19 case data which is continuous and spread over time, and b) related events, categorised in terms of importance.

In this case, the user can progress/rewind the story via buttons. An importance ranking value determines which events will be displayed, i.e., whether all or more significant events are displayed. The ranking of these events is decided by the author, whereas the ranking mechanism is explained in Section 4.2. As features are encountered and actions initiated, text descriptions are concatenated from author defined text strings, with location and data values from the data enriching the story text. This is shown in a labelled text-box linking the story text directly to the specified feature (Figure 4(c)).

This story's author used a spreadsheet to organise and map features and actions, which can be a suitable approach for several reasons. Exemplar features can be added to the spreadsheet and then edited into general ones. The order of the features can be readily re-arranged, whereas different priorities that help to identify the

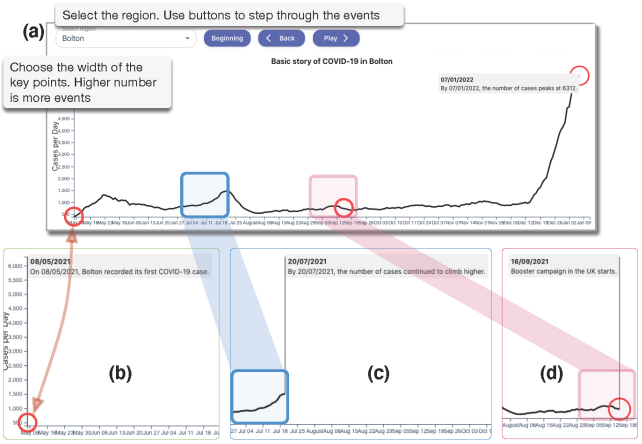


Figure 4: *Demonstration of the single-location story. Programmed features are located. When the Play button is pressed the delivery system actions to progresses to the next feature, the appropriate text is concatenated with real data, and placed in a suitable position (if there is space it will be shown to the right of the vertical line, otherwise to the left). (a) The full interface; (b) story start, (c) highest deaths per day, (d) booster vaccination program starts.*

importance and order of the story elements can also be added to the spreadsheet. In addition, it is simple to map features alongside actions, as different cells. Furthermore, code can be translated and aligned closely to the spreadsheet content, or even parse it.

3.2. Comparative Storyboard: Two Numerical Time Series

The comparison story (Figure 5) allows viewers to compare two different locations. In our example, a user selects Bedford (100,000 inhabitants, 50 miles from London) and Bradford (350,000 inhabitants, 150 miles North of Bedford), both in England, under the same COVID-19 restrictions and reasonably close; yet on different rail routes to London and distant enough to have separate ecosystems. Investigating them might indicate if the pandemic was moving towards or away from London.

Similar to the previous example, each line-graph corresponds to a location. Because there are two time series, the comparison between them is the key narrative element. The meta-author defined the same set of features to be detected for both locations. When the detected features were visualized, it appeared to alternate between features in the two line-graphs. Such features are relevant to the characteristics of COVID-19 waves; e.g., cases increased in two weeks. The meta-author defined actions according to features. Messaging for this story has two main goals: (a) to highlight features on each time series, in a similar manner to the previous story, and (b) to highlight comparisons between the time series [GAW*11].

This example explicates several key facets. First, the story starts with the region that has the first significant event; in this case, it starts with Bradford as shown in Figure 5(b). Second the animation automatically alternates between regions and significant events. The meta-author designed the story of each city around its major COVID-19 waves, which are identified by matching peak-related features in the data. The meta-author instructed that events for both

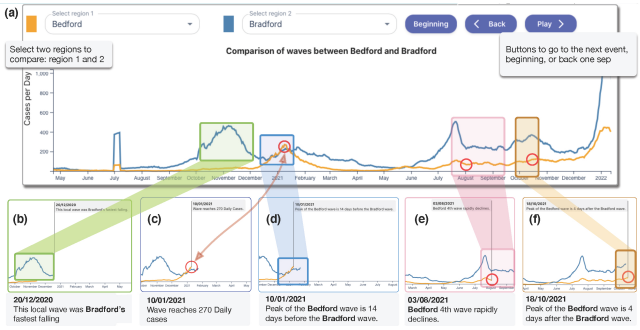


Figure 5: *Comparison story demonstration, where (a) depicts the final frame. The story is shown in stages, moving key 'features', and alternating 'actions' between region 1 and 2. The insets (b-f) depict several key event features, which are incrementally shown as the story progresses; (b) a single feature and action about Bradford (region 2); (c) story action focusing on peaks, with data specific to the local site; (d and f) comparison feature showing differences in terms of days; (e) feature comparison based on calculated data.*

Figure 6: *The country story (in this example focusing on England). (a) Shows is a screenshot near the end of the story. Insets (b-d) highlight three features. As the reader plays the story, so the line-graph animates to reveal the next part of the story, the text narrative animates (sliding to reveal the next text). The text narrative shows the current text bracketed by the previous and next story text.*

cities were to be grouped around the same wave and to be played in the order of waves. With such grouping, comparisons can be made between the two cities. For example, in Figure 5(b) the story focuses on Bradford, and then in Figure 5(c) the other region (Bedford) catches up. Later, the developer implemented this instruction by processing of features in two time series concurrently. The meta-author also wished that the highest ranked features were shown and highlighted. Our example depicts a few events from one region, before moving to events from the other. Regions with higher ranking, or higher data values, will play first. Third, we compare events; whether they are different, same or larger. We look at significant rises, declines and peaks; see Figure 5 (d), (e) and (f), respectively. For instance, Figure 5(d) shows how the text explicitly compares two values (saying that Bedford was 14 days before Bradford). The point of interest is indicated with red circle and a vertical text label (generated from the data).

3.3. Scrollable Storyboard: Two Numerical Time Series

Our third demonstration provides a holistic scrolling story of the events in England and includes three additional facets. First we display data as a line-graph, which provides an overview of all events in the storyline. Second, we present text descriptions for the current event and elided versions for the previous and the forthcoming event. The text scrolls as the audience plays the story. In this manner, the current story text is shown in the context of what has happened, plus a hint of what is to come (Figure 6).

The scrollable story was designed with two abstractions in mind: (a) the detail of individual, and (b) the overview of all, features (cases and deaths time series). The design was captured using the FDS [RHR16] method. Features were determined in relation to either cases, deaths, or both, and an overview is depicted in the event-line. Each highly-ranked feature is highlighted by a circle, with the current one darker. Current, previous and next event text descriptions are updated for each event.

Important features, such as "first recorded case", number of deaths or hospitalisation cases in tens of thousands, rise in cases by 20% and so on, can be automatically detected respectively. By defining features with notable pre-defined values it is possible to signify interesting events from the time-series data as well. For example, people will be interested in the first case, when cases (or deaths) reach 100, 1000 and so on. Finally, events such as lockdown start/end dates, vaccination start dates, quotes from parliamentarians etc., can be ranked as in the previous examples.

4. Algorithmic Pipeline

In this section, we describe the underlying algorithmic pipeline of our method. Figure 7 illustrates an overview of our method's workflow. The orange boxes in the top row show the human inputs, of the story viewer (data selection by virtue of story and location selection, fetched from the data repository) and the storyteller (remaining inputs). The light green boxes along the thick brown line is the data interface that translate the viewer's and storyteller's inputs to inputs algorithms can process. The remaining boxes, below the light brown line, illustrate the data flow and algorithmic process in the flow. We consider two categories of time series data:

Numerical time series (NTS) – This category includes commonly-encountered time series during the COVID-19 pandemic, such as daily, accumulated, normalised, and k-day moving average data, with semantics such as number of cases, hospitalisations, fatalities, vaccinations, etc. Figure 8(a) shows an example of a time series of the infection rate in Aberdeen City between March 2020 and October 2021. A feature detection algorithm that searches for peaks is applied to the data. Each detected peak is segmented and highlighted by a different colour and its apex indicated by a black dot.

Categorical time series (CTS) – This category extends the notion of NTS by considering each data point at time t can have a categorical or nominal value. Assuming that "null" or "no value" is a valid

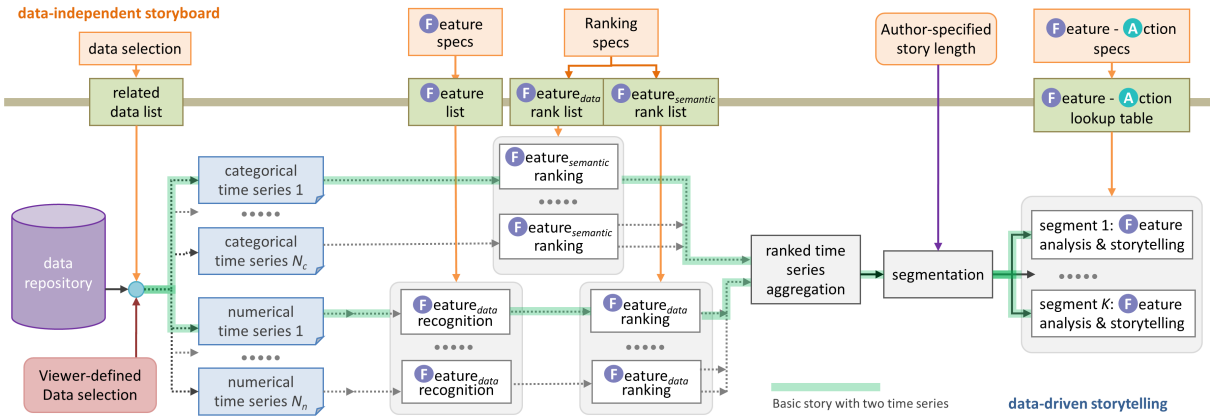


Figure 7: *Overview of our system's workflow. The author creates a story by selecting a data which require a specific dataset from the repository. They then define features, determine the ranking (importance) of these features and map detected features (F) to specific actions (A). They also define the effective length of the story (□). These are converted by the developer in lookup tables and lists used by the algorithm (□). The lists are then used to process the different categorical and numerical time series (□), by detecting and ranking the defined features (▭ stages). The time-series are then aggregated, and segmented based on the author-specified story length. Once the user selects the location(s) of the story (□) the story commences, and actions are invoked when any corresponding features are detected. The path for the simple story, described in Section 3, is highlighted in green. Further technical details can be found in Appendices A and B in the supplementary materials.*

categorical value, every data point in the time series has a value in the same way as NTS. Fig. 8(b) shows an example of a categorical time series, represented as a timeline featuring major healthcare policy changes, such as "lockdown". The example details events between the period of March 2020 to October 2021. Each blue dot on the timeline represents a different semantic event.

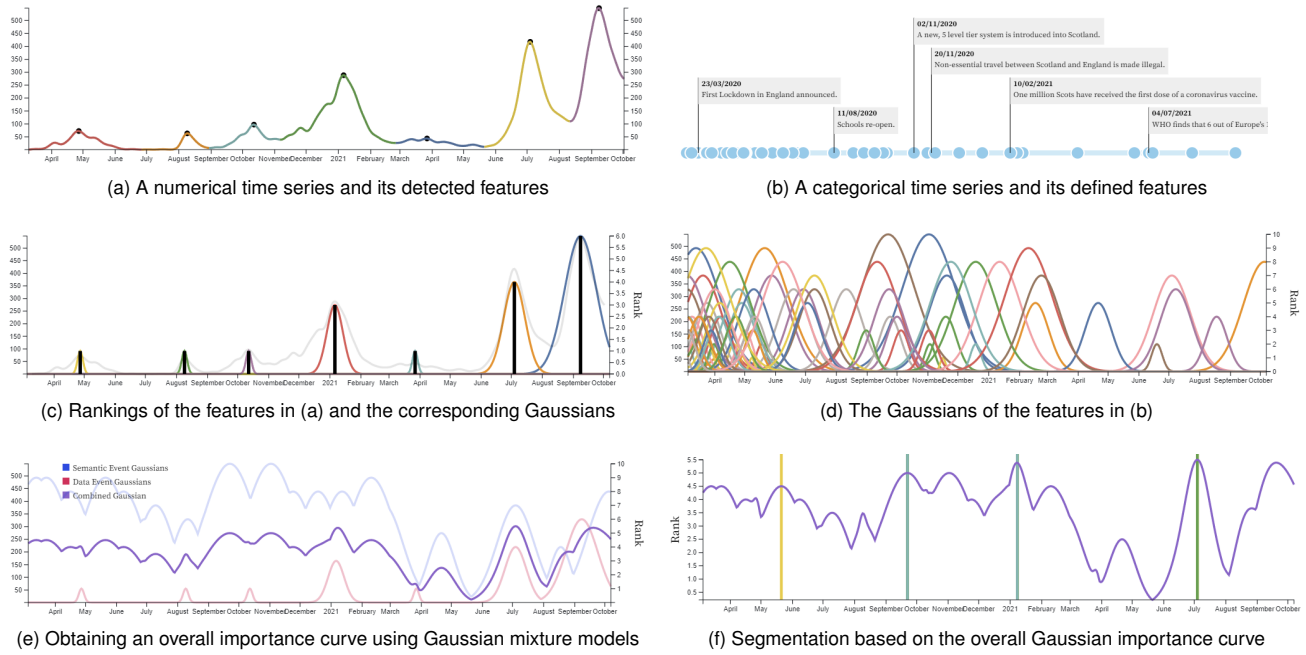
4.1. Feature Detection

Feature detection is a widely-used data analysis process in many applications, such as signal, speech, and image processing. In a storyboard about time series, there are some common features, such as "peaks", "valleys", "steep climb/fall", data milestones, smoothness of data, etc. Nevertheless, the design of a feature detection algorithm may not always be straightforward. For example, a "peak' may be defined a local maximum point in a time series – there are usually a lot of them. Alternatively, it may be defined a global maximum point – there is often just one such point, or if there are a few, they are unlikely distributed sparsely in different periods. On the other hand, a story author typically requires a more complex feature definition, e.g., a local maximum that is of some distance from other peaks. Furthermore, the way a human may identify a peak is completely different to how a computer mathematically defines one. In our case, the feature detection algorithms had to go through several rounds of adjustments to match the desires of the story designers. An example of the results of our peak detection algorithm can be seen in Figure 8(a). The algorithm selects possible peaks from the set of all maximal points in the data. In decreasing height order, we traverse down the slopes to the left and right of each maxima until a minimum point is reached. In doing so, the bounds of the peak are found and any maxima along the way, considered part of the same peak, are removed from the selection pool. The data is segmented into different peak regions, as can be seen in Figure 8(a), with different colours representing different peaks. In many ways, algorithms for detecting different features in an NTS assign categorical values to the data points at different time points. These categorical values collectively define a CTS.

4.2. Importance Ranking

Given a CTS with many different categorical values, a story designer will almost always wish to place different levels of emphasis on different data points according to their categorical values. For example, a "full lockdown" event may be considered more important than a minor policy change. Similarly, given an NTS, following feature detection processes, the detected features can also be associated with different levels of importance. As shown in Figure 7, we conveniently refer to the importance ranking for a CTS as Feature$_{semantic}$ ranking, and that for an NTS as Feature$_{data}$ ranking. In our system, importance ranking processes allows real-value ranking with a maximum value ($r_{\max} > 0$) that must be consistently defined within each story. For all our case studies, we defined $r_{\max} = 10$. The black bars in Figure 8(c) indicate the importance ranking values assigned to each peak feature detected in Figure 8(a), and ranking takes into account the height of the peak.

Because the importance ranking is based on categorical values of events and features, the ranking is somehow discrete and the interactions between different raking values (e.g., the black bars in Figure 8(c)) are difficult to compute numerically. We thus assign a *Gaussian component distribution* (commonly referred to as a *Gaussian* for short) to each ranking value as Gaussians indicate an interaction between two features/events and the level of interaction can be numerically computed. In many cases, there could be many interactions, such as Figure 8(d), where each Gaussian corresponds to each categorical values (except null) in Figure 8(b).



(a) A numerical time series and its detected features

(b) A categorical time series and its defined features

(c) Rankings of the features in (a) and the corresponding Gaussians

(d) The Gaussians of the features in (b)

(e) Obtaining an overall importance curve using Gaussian mixture models

(f) Segmentation based on the overall Gaussian importance curve

Figure 8: *Graphical depiction of the algorithmic pipeline for automated storytelling. (a) Peaks, and rising and falling segments of the time series are detected. (b) Event features in the categorical time series are typically predefined and ranked. (c) Each detected data feature is given a rank (illustrated as black bar). Each ranking value is converted to a Gaussian curve. (d) Likewise, ranked semantic events are converted individually to Gaussian curves. (e) Gaussian curves are combined using Gaussian mixture models. Here a max-model is used within a time series and a mean-model is used between time series. (f) The story is divided in segments according to the combined importance curve. The number of segments is defined by a meta-author. In this example, the three green lines show the results of the segmentation algorithm, and four yellow lines indicate the segmentation results if five segments are required (three yellow lines happen to coincide with the green lines).*

In storytelling visualization, Gaussian mixture models were previously applied to video data [PLC*11]. As Gaussians are continuous curves in $[-\infty,\infty]$, we can use Gaussian mixture models to obtain a continuous importance curve by combining all Gaussians within a time series and an *overall importance curve* by combining the importance curves of all NTS and CTS related to a storyboard. For example, in Figure 8(e), the light red time series is the mixed importance curve of the Gaussians in Figure 8(c), while the light blue time series is that of the Gaussians in Figure 8(d). A max() Gaussian mixture function is used in both cases. These two mixed importance curves are then mixed using a mean() Gaussian mixture function, yielding the purple overall importance curve that encodes the importance features in Figure 8(a) and events in Figure 8(b).

4.3. Timeline Segmentation

Once we have obtained the overall importance curve for a storyboard, we can start the process of generating a storytelling visualization. Similar to telling a story in speech or writing, we would like to select relatively more important features to be "mentioned visually" in a story according to the time or other resources available to the story. In this work, we consider two types of resources, the *number of button pressing actions* in interactive story progression, and the *total animation time* in fully automated story progression. If there were too many button pressing actions or the animation were too long, the viewers would become weary or impatient. If there were too few interactions or the animation were too short, the viewers may find the information provided inadequate. Technically, we would like to divide the timeline of a storyboard into k sections. In interactive story progression, k sections require k button pressing actions (including the start). In automated story progression, k determines the total animation time as $k\times$"unit section time". Figure 8(f) shows two possible segmentation results with $k=3$ and $k=5$ respectively.

We developed an algorithm to segment the timeline of a storyboard by considering the peaks in the overall importance curve of the storyboard. The algorithm selects the top $k-1$ peaks with a maximal gap $\Delta_{\max}$ between neighbouring peaks. In this way, we ensure that the most important temporal points are selected and located at the section boundaries, where we can insert more visualization actions. With interactive story progression, users can spend more time to view the current visualization before pressing the button for progression. In automated story progression, we can slow the animation at each boundary between the two sections.

4.4. Feature-Action Profiling

As aforementioned, the output of the meta-authoring process is a feature-action specifications expressed by the story author in different ways (e.g., textual descriptions, pseudocode and sketches). A developer translates the specifications to feature-action data patterns in a lookup table, where *features* are categorical labels of all features that might be detected in the NTS and CTS select by story

viewers, and *actions* are calls to appropriate functions of the software components, for displaying various visual artefacts (Figure 3) as well as exhibit the desired software behaviours in terms of interaction, graphics, and animation.

The story's length is controlled by how many segments the Gaussian curve is split into and how many features there are in the lookup table. A story designer can extend the time a story takes to complete, by increasing the number of features in the table to pick up on. This timing can also be controlled using the feature rankings. For instance, to decrease the time to complete a story one could restrict the algorithm to showing only the most important events, say the top three events in a segment or maybe just those with a ranking of 8 or greater.

The algorithm for feature-action profiling visits each segment one-by-one. Within segments each feature is visited in chronological order. Each time, it refers to the feature-action lookup table to check if the current feature matches any entry. If so the visual action is initiated.

4.5. Generation of Storytelling Visualization

After all sections have been profiled, the "play" button becomes available to users. In the mode of interactive story progression, a viewer presses the button to activate the storytelling visualization of a section. In response, the underlying algorithm visits the NTS and CTS of user-selected region in the order of time. For any feature (or event) that is marked as being included in the story, the algorithm invokes the corresponding action. When the end of the section is encountered, the algorithm waits until the viewer presses the "play" button, to proceed to the next section. In the mode of automated story progression, instead of waiting for a viewer's interaction, the algorithm waits for a short time span, and continues to the succeeding section automatically. To make all stories have a consistent look and feel, this mode is not currently deployed.

5. Evaluation and Reflection

To evaluate our approach, we grouped stakeholders of this work according three perspectives: (i) story authors, (ii) software developers, and (iii) public members and one public engagement expert. The first two roles are the intended users of our approach, where as the latter were chosen to provide insight on the viewer-centric nature of our method's outcomes. We requested their written feedback, using a shared online document and collected their comments as evaluation and reflection, similar to a retrospective verbal protocol analysis. While we provided five questions for each group as prompts, we gave explicitly the direction "please feel free to add any comment." The full text of the prompts and feedback is given in Appendix C in the supplementary materials. The main feedback points are summarised below, with direct extractions from stakeholders' input in *italics*.

5.1. From the Perspective of Story Authors

Three story authors (who created the stories in Section 3) provided feedback of more than 1400 words. The main points are summarised below:

- Meta-authoring can be more challenging than telling a story about a single dataset. Authors *need to think about the story holistically*, and how different data, say from different regions, may *support key messages*. *It forces authors to consider the underlying data in detail,* which is *a good practice in storytelling*. *Generalizing the story is definitely a gain from using this approach. Creating a draft storyline helped think about how to generalise and abstract the information into the meta-instructions*.
- Creating feature-action design patterns can be *somewhat challenging*. *Certainly some of the key features were obvious, such as a peak, rise and fall. But not all were easy* to define. It seems to make an author consider *the progression of the story in a more "mathematical" way*. *Information needs to be abstracted or generalised into instructions* for software developers.
- *The workflow relies on the software engineer to both interpret the design and create code*. It is appropriate at the moment with the state of the art technology for storytelling visualization, but *there are opportunities for further automation*. The approach of having *software engineer in-the-loop* may work for *large organisations*, but independent meta-authors or those in *small organisations* can benefit from meta-authoring tools.
- In the future, *an advanced interface could be developed* for meta-authors *to create these feature-action* design patterns. *One could envision an interface where storytellers can pick and match story segments and combine them* sequentially or hierarchically. Feature-action *abstraction can be exploited to create a more modular approach for the end users* (i.e., meta-authors).

5.2. From the Perspective of the Developer

Two research software developers were directly involved in developing the pipeline in Figure 7, various feature detection algorithms, and visualization components. They, together with a third stakeholder who helped design the algorithmic pipeline, provided feedback totalling more than 1200 words. The main points include:

- It is not difficult for a technical developer to support the meta-authors in general. Given a storyboard and visualization guidance from the meta-authors, *a programmer can create the storytelling visualization easily*. *The notion of feature-action design patterns was accepted by both meta-authors and technical developers.* The technical developers were able to *translate* meta-authors' *qualitative specification of design patterns* to the actual implementation in a *look-up table relatively easily.*
- *An implication of creating stories for the public is that features must understandable and simple. Developing such feature detection algorithms varied from taking a few hours to a few days.*
- Regarding software components that implement actions, while it is not hard to implement them individually, *it can take up to a week to* put them together for a story. Among our stories, *the longest development, for an entire story took roughly four days.*
- *Observable is a very good "drafting" software. It would take less than a day to implement a story in Observable if the story requires no new feature or action.* However, it was not feasible for us to place many dynamic data streams on Observable. We had to develop and deploy storytelling visualization on an infrastructure that keeps the dynamic data streams.
- *the visualization software components (e.g., multi-line time se-*

ries plots, textual annotations, timelines, animation, etc.) are implemented using D3.js [BOH11]. With the existing software code and programming experience, *developing new software components using D3.js had been quick and easy.* These software components *were implemented as reusable classes and functions.* Gradually, the time for implementing new software components for a new story *can be reduced* significantly.

- In the future, it will be desirable and feasible to develop software systems to support the workflow from meta-authoring to story deployment. Developers and meta-authors have suggested a number of options, including (i) *visual programming environment*, (ii) *object-oriented/template-based programming environment*, (iv) *scripting language*, (v) *markup language (e.g., XML)*, (vi) *declarative data interchange format (e.g., JSON, YAML)*, and (vii) *form-like webpage*. *the software components developed in this project can provide a good basis for future developments.*

5.3. From the Perspective of Public Engagement

We were interested in hearing from those stakeholders who had expertise in public engagement as well as who are the potential viewers of storytelling visualization. One expert of public engagement provided feedback directly, and two members of the public provided verbal feedback, which were transcribed by a co-author. Together the three stakeholders provided feedback totalling more than 900 words. The main points in the feedback include:

- *There has been a surfeit of data relating to COVID-19 in the public domain, which may well seem like information overload to members of the public. Without context, a time series may not convey much meaning. The story telling both provides context and divides the data into digestible segments, allowing the viewer to look at the situation leading up to an event and following after it, and to understand/reflect on each segment before moving on to the next. ... Because the stories are told linearly and from a local area perspective, the user can situate themselves both in place and time in relation to the events, which adds to the human interest and relatability of the data.*
- *Because it takes only one selection button for a region and then the play button to start, it is actually slightly easier than many search interfaces. If it were available some time in 2020 (and perhaps the early part of 2021), it would get used a lot.*
- In the stories, *the event descriptions are simple and effective, giving clear information about key landmarks. The speed of the animation provides momentum, and the event pauses give the user control over how fast to move on with the story.*
- *The stories are likely to appeal to members of the public who are regular news viewers, who listen to scientific podcasts, and who take an interest in the local or regional context of the pandemic and how policies or actions affect its progress.*
- *This form of location-dependent, data-driven storytelling would be useful and of interest to members of the public*, and could be used to visualise *housing, energy and fuel prices*, *school performance*, *crime rate*, and *transport statistics*. *Comparison of different cities, areas, or regions would be useful.*
- *Would it be possible for an individual to create such a story to share with friends? Some people are quite keen to share data, such as their daily walking step counts.*

6. Further Application and Evaluation

While the aforementioned meta-authoring method for creating storytelling visualization was developed during the COVID-19 pandemic, we have always been keen to apply the method to other domains, and demonstrate its generalisability. An opportunity arose when one of our visualization (VIS hereafter) researcher joined a ML team in the Science and Technology Facilities Council (STFC) in the UK. The ML team has been providing physics, nuclear, space, and astronomy scientists with ML models for processing a variety of data captured using different sensory modalities (e.g., [HBJT20]). One R&D strand of the team has been to develop and analyse ML benchmarks as the basis for providing guidelines and best practices in AI for Science [HPT*21, TSFH22].

Requirements Analysis. Two VIS researchers met the ML team leader and identified general requirements for using automated visualization tools in ML workflows, as well as several specific requirements for using storytelling visualization. As the team always has a few ongoing ML workflows concurrently and each may last up to 18 months, they need to observe their progress and provenance frequently, preferably though dynamically-updated stories. Following this meeting, one VIS researcher collaborated closely with an ML researcher working on ML benchmarks. The VIS researchers designed and implemented two storyboards initially as shown in Figure 9(a,c). During one evaluation meeting, we identified the need for an analytical dashboard with storytelling on demand. This became a new storyboard, shown in Figure 9(b).

Meta-Storyboards. The STFC ML team has an established process for storing log data for ML training and testing. When a model is tested, each logged event consists of data such as date, time, hyperparameters, and accuracy measures. To re-purpose the meta-authoring software developed in the context of COVID-19, we focused on log data featuring time series.

A **Provenance Story** (Figure 9(a)) was designed around the provenance and progress of an ML workflow. A viewer can select a story to be told from the perspective of a specific hyperparameter. The animated visualization shows the logged chronological events focuses on those related to the selected hyperparameter. Interesting data features include, for instance, when the accuracy has a significant improvement or reaches a peak point. Visual actions (e.g., text message, colour highlighting) are triggered by such features.

An **Analytical Dashboard & Story** (Figure 9(b)) enables individual ML researchers to visualise logged events in both static and storytelling manner. Whenever a model-testing process is completed, the ML researcher concerned needs to observe the new logged event. To prevent the storytelling mechanism from slowing down the routine VIS tasks, the storyboard makes static visualization as default, with a visual design that focuses on the ordered values of a selected hyperparameter rather than event date and time. Drawing our experience of working on COVID-19 data [KNAR*22, BFAR*23, KNA*22], we introduced elements of dashboard design to improve the efficiency of the routine VIS tasks. The storytelling mechanism can be activated when the ML researcher wishes to be reminded of the provenience, or other team members wish to be told stories about individual hyperparameters.

Finally, a **Multivariate Story** (Figure 9(c)) tells stories about

Figure 9: *Screenshots of three stories implemented for supporting ML workflows developed using our approach.*

multiple hyperparameters through a parallel coordinates plot (PCP). Viewers can select a specific hyperparameter to inform the importance ranking of features (see Section 4.2). The animated visualization shows the logged events chronologically, but unlike Figure 9(a), viewers can see all hyperparameters in the same visualization. As the visualization is more complex, the feature-action patterns involve more frequent pause-and-continue actions.

Evaluation and Improvement. In the STFC ML team, the VIS researcher was in contact with ML colleagues regularly, discussing requirements and design options while conducting frequent bite-sized evaluations. In addition, there were two formal evaluation meetings. In the first meeting, ML researchers were shown a first-draft working version of the Provenance and Analytics Dashboard stories (Figure 9(a,c)), which prompted them to make a number of revision suggestions. We also noticed that different team members suggested a revision on the requirements of the Provenance Story. Moreover, the team wanted to observe the progress and provenance of all ML workflows in a single storytelling visualization. Finally, individual ML developers wanted to observe new results quickly whenever a trained model has been tested and observe the provenance data on demand. We decided to create a separate storyboard for these, i.e., the Analytical Dashboard & Story (Figure 9(b)).

Although that meeting was the first time when the ML researchers worked with a PCP, without much explanation by VIS researchers, they found the Multivariate Story particularly useful for observing the combined impact caused by different hyperparameters. This was somewhat unexpected, since many visualization students and users often encounter difficulties in learning to work with PCPs. We hypothesised that because storytelling shows data-events line-by-line, this may make the concept of PCPs more intuitive. Animation might also reduce the confusion caused by cluttering, while enabling viewers to pay attention to the interesting data-events without much visual searching effort. In the second evaluation meeting, we focused on the Analytical Dashboard & Story (Figure 9(b)). The ML experts were enthusiastic about the dashboard and discussed the ideal data to be shown in the information boxes, and the interesting features in the plot, so that the storytelling mechanism highlights them automatically.

7. Conclusion

In this work, we presented a new method for creating storytelling visualizations, based on a workflow for meta-authoring with feature-action design patterns. We have demonstrated the feasibility of this new method through three storyboards developed in the COVID-19 visualization context, followed by three storyboards describing ML workflows. Our work addresses the challenge of creating generic storyboards that can be applied to different yet similar, often partially unknown data streams and can respond to different (data) features automatically using different (visualization) actions. As summarised in Section 5 and Section 6, the method has been welcomed by different stakeholders, who have already looked ahead to the potential applications in the future as well as more advanced technologies that can provide system-level support to meta-authoring workflows. Building on this work, we hope to continue improve techniques for storytelling visualization.

Acknowledgments

The authors would like to express their gratitude to the SCRC management team for its leadership, all SCRC members (including VIS volunteers) for their selfless efforts, and the UK Science

and Technology Facilities Council (STFC) for providing and managing the RAMPVIS hardware platform. In particular, we thank all other VIS volunteers for their contributions to various RAMPVIS activities since May 2020. This work was supported in part by the UKRI/EPSRC grants EP/V054236/1 and EP/X029557/1.

References

[ARL*16] AMINI F., RICHE N. H., LEE B., MONROY-HERNANDEZ A., IRANI P.: Authoring data-driven videos with dataclips. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2016), 501–510. doi:10.1109/TVCG.2016.2598647. 2

[BEDF15] BOY J., EVEILLARD L., DETIENNE F., FEKETE J.-D.: Suggested interactivity: Seeking perceived affordances for information visualization. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2015), 639–648. doi:10.1109/TVCG.2015.2467201. 3

[BFAR*23] BACH B., FREEMAN E., ABDUL-RAHMAN A., TURKAY C., KHAN S., FAN Y., CHEN M.: Dashboard design patterns. *IEEE Transactions on Visualization and Computer Graphics 29*, 1 (2023), 342–352. doi:10.1109/TVCG.2022.3209448. 9

[BLB*17] BREHMER M., LEE B., BACH B., RICHE N. H., MUNZNER T.: Timelines Revisited: A Design Space and Considerations for Expressive Storytelling. *IEEE Transactions on Visualization and Computer Graphics 23*, 9 (2017), 2151–2164. doi:10.1109/TVCG.2016.2614803. 3

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2301–2309. doi:10.1109/TVCG.2011.185. 9

[BSB*18] BACH B., STEFANER M., BOY J., DRUCKER S., BARTRAM L., WOOD J., CIUCCARELLI P., ENGELHARDT Y., KOEPPEN U., TVERSKY B.: Narrative design patterns for data-driven storytelling. In *Data-driven storytelling*. AK Peters/CRC Press, 2018, pp. 107–133. 2

[BWF*18] BACH B., WANG Z., FARINELLA M., MURRAY-RUST D., HENRY RICHE N.: Design Patterns for Data Comics. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (New York, NY, USA, 2018), CHI '18, ACM, p. 1–12. doi:10.1145/3173574.3173612. 2

[CARA*22] CHEN M., ABDUL-RAHMAN A., ARCHAMBAULT D., DYKES J., SLINGSBY A., RITSOS P. D., TORSNEY-WEIR T., TURKAY C., BACH B., BRETT A., FANG H., JIANU R., KHAN S., LARAMEE R. S., MATTHEWS L., NGUYEN P. H., REEVE R., ROBERTS J. C., VIDAL F., WANG Q., WOOD J., XU K.: RAMPVIS: Answering the challenges of building visualisation capabilities for large-scale emergency responses. *Epidemics 39* (2022), 100569. doi:10.1016/j.epidem.2022.100569. 2

[Cen22] CENTER FOR SYSTEMS SCIENCE AND ENGINEERING (CSSE) AT JOHNS HOPKINS UNIVERSITY (JHU): *COVID-19 Dashboard*. [Online]. Available https://coronavirus.jhu.edu/map.html, 2022. Accessed: 25/03/2022. 2

[Cox11] COX A.: How editing and design changes news graphics. In *IEEE Conference on Visual Analytics Science and Technology (VAST)* (2011), pp. xiii–xiii. doi:10.1109/VAST.2011.6102432. 3

[DARA*22] DYKES J., ABDUL-RAHMAN A., ARCHAMBAULT D., BACH B., BORGO R., CHEN M., ENRIGHT J., FANG H., FIRAT E. E., FREEMAN E., GÖNEN T., HARRIS C., JIANU R., JOHN N. W., KHAN S., LAHIFF A., LARAMEE R. S., MATTHEWS L., MOHR S., NGUYEN P. H., RAHAT A. A. M., REEVE R., RITSOS P. D., ROBERTS J. C., SLINGSBY A., SWALLOW B., TORSNEY-WEIR T., TURKAY C., TURNER R., VIDAL F. P., WANG Q., WOOD J., XU K.: Visualization for epidemiological modelling: Challenges, solutions, reflections & recommendations. *Philosophical Transactions of the Royal Society A 380* (2022), 2233. doi:10.1098/rsta.2021.0299. 2

[DHPP17] DEMIRALP Ç., HAAS P. J., PARTHASARATHY S., PEDAPATI T.: Foresight. In *Proceedings of the VLDB Endowment* (Aug. 2017), vol. 10, VLDB Endowment, pp. 1937–1940. URL: https://doi.org/10.14778/3137765.3137813, doi:10.14778/3137765.3137813. 3

[FBM15] FULDA J., BREHMER M., MUNZNER T.: TimeLineCurator: Interactive authoring of visual timelines from unstructured text. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2015), 300–309. doi:10.1109/TVCG.2015.2467531. 3

[GAW*11] GLEICHER M., ALBERS D., WALKER R., JUSUFI I., HANSEN C. D., ROBERTS J. C.: Visual comparison for information visualization. *Information Visualization 10*, 4 (2011), 289–309. doi:10.1177/1473871611416549. 4

[GP01] GERSHON N., PAGE W.: What Storytelling Can Do for Information Visualization. *Communication of the ACM 44*, 8 (aug 2001), 31–37. doi:10.1145/381641.381653. 1, 2

[HBJT20] HEY T., BUTLER K., JACKSON S., THIYAGALINGAM J.: Machine learning and big scientific data. *Philosophical Transactions of the Royal Society A 378* (2020), 20190054. 9

[HDR*13] HULLMAN J., DRUCKER S., RICHE N. H., LEE B., FISHER D., ADAR E.: A deeper understanding of sequence in narrative visualization. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2406–2415. doi:10.1109/TVCG.2013.119. 2

[HPT*21] HENGHES B., PETTITT C., THIYAGALINGAM J., HEY T., LAHAV O.: Benchmarking and scalability of machine learning methods for photometric redshift estimation. *Monthly Notices of Royal Astronomical Society 505*, 4 (2021), 4847–4856. 9

[KM13] KOSARA R., MACKINLAY J.: Storytelling: The next step for visualization. *Computer 46*, 5 (2013), 44–50. doi:10.1109/MC.2013.36. 1

[KNA*22] KHAN S., NGUYEN P., ABDUL-RAHMAN A., FREEMAN E., TURKAY C., CHEN M.: Rapid Development of a Data Visualization Service in an Emergency Response. *IEEE Transactions on Services Computing 15*, 3 (2022), 1251 – 1264. doi:10.1109/TSC.2022.3164146. 9

[KNAR*22] KHAN S., NGUYEN P. H., ABDUL-RAHMAN A., BACH B., CHEN M., FREEMAN E., TURKAY C.: Propagating Visual Designs to Numerous Plots and Dashboards. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (2022), 86–95. doi:10.1109/TVCG.2021.3114828. 9

[KWHH17] KIM Y., WONGSUPHASAWAT K., HULLMAN J., HEER J.: GraphScape: A Model for Automated Reasoning about Visualization Similarity and Sequencing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (New York, NY, USA, May 2017), ACM, pp. 2628–2638. URL: https://doi.org/10.1145/3025453.3025866, doi:10.1145/3025453.3025866. 3

[LES20] LAW P., ENDERT A., STASKO J. T.: Characterizing automated data insights. In *Proceedings of the IEEE Visualization Conference - Short Papers* (2020), IEEE, pp. 171–175. URL: https://doi.org/gk8brf, doi:10.1109/VIS47514.2020.00041. 3

[LKS13] LEE B., KAZI R. H., SMITH G.: SketchStory: Telling More Engaging Stories with Data through Freeform Sketching. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2416–2425. doi:10.1109/TVCG.2013.191. 2

[LRIC15] LEE B., RICHE N. H., ISENBERG P., CARPENDALE S.: More Than Telling a Story: Transforming Data into Visually Shared Stories. *IEEE Computer Graphics & Applications 35*, 5 (2015), 84–90. doi:10.1109/MCG.2015.99. 1, 2

[MBS24] MOERTH E., BRUCKNER S., SMIT N. N.: Scrollyvis: Interactive visual authoring of guided dynamic narratives for scientific scrollytelling. *IEEE Transactions on Visualization and Computer Graphics 30*, 1 (2024). 3

[MRL*17] MCKENNA S., RICHE N. H., LEE B., BOY J., MEYER M.: Visual Narrative Flow: Exploring Factors Shaping Data Visualization Story Reading Experiences. *Computer Graphics Forum 36*, 3 (2017), 377–387. doi:10.1111/cgf.13195. 3

[Nor18] NORTHWESTERN UNIVERSITY KNIGHT LAB: *Timeline.js*. [Online]. Available https://timeline.knightlab.com/, 2018. Accessed: 30/03/2022. 3

[OBCT24] OFFENWANGER A., BREHMER M., CHEVALIER F., TSANDILAS T.: Timesplines: Sketch-based authoring of flexible and idiosyncratic timelines. *IEEE Transactions on Visualization and Computer Graphics 30*, 1 (2024). 1, 3

[PLC*11] PARRY M. L., LEGG P. A., CHUNG D. H., GRIFFITHS I. W., CHEN M.: Hierarchical Event Selection for Video Storyboards with a Case Study on Snooker Video Visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1747–1756. doi:10.1109/TVCG.2011.208. 3, 7

[RBSN22] ROBERTS J. C., BUTCHER P., SHERLOCK A., NASON S.: Explanatory Journeys: Visualising to Understand and Explain Administrative Justice Paths of Redress. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (2022), 518–528. doi:10.1109/TVCG.2021.3114818. 2

[RG13] ROSENBERG D., GRAFTON A.: *Cartographies of time: A history of the timeline*. Princeton Architectural Press, 2013. 3

[RHR16] ROBERTS J. C., HEADLEAND C., RITSOS P. D.: Sketching Designs Using the Five Design-Sheet Methodology. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 419–428. doi:10.1109/TVCG.2015.2467271. 5

[SH10] SEGEL E., HEER J.: Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1139–1148. doi:10.1109/TVCG.2010.179. 2

[SH14] SATYANARAYAN A., HEER J.: Authoring narrative visualizations with Ellipsis. *Computer Graphics Forum 33*, 3 (2014), 361–370. doi:https://doi.org/10.1111/cgf.12392. 2

[SKH*22] SHIN M., KIM J., HAN Y., XIE L., WHITELAW M., KWON B. C., KO S., ELMQVIST N.: Roslingifier: Semi-automated storytelling for animated scatterplots. *IEEE Transactions on Visualization and Computer Graphics* (2022), 1–1. doi:10.1109/TVCG.2022.3146329. 2

[SWT*20] SHU X., WU A., TANG J., BACH B., WU Y., QU H.: What makes a Data-GIF Understandable? *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2020), 1492–1502. doi:10.1109/TVCG.2020.3030396. 2

[SXS*21] SHI D., XU X., SUN F., SHI Y., CAO N.: Calliope: Automatic Visual Data Story Generation from a Spreadsheet. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2021), 453–463. URL: https://doi.org/10.1109/TVCG.2020.3030403, doi:10.1109/TVCG.2020.3030403. 1, 3

[SyzZW24] SHEN L., YIZHI ZHANG, ZHANG H., WANG Y.: Data player: Automatic generation of data videos with narration-animation interplay. *IEEE Transactions on Visualization and Computer Graphics 30*, 1 (2024). 3

[TLW*21] TANG T., LI R., WU X., LIU S., KNITTEL J., KOCH S., ERTL T., YU L., REN P., WU Y.: PlotThread: Creating Expressive Storyline Visualizations using Reinforcement Learning. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2021), 294–303. doi:10.1109/TVCG.2020.3030467. 2

[TRB*18] TONG C., ROBERTS R. C., BORGO R., WALTON S. P., LARAMEE R. S., WEGBA K., LU A., WANG Y., QU H., LUO Q., MA X.: Storytelling and Visualization: An Extended Survey. *Information 9* (2018), 65. doi:10.3390/info9030065. 1, 2

[TSFH22] THIYAGALINGAM J., SHANKAR M., FOX G., HEY T.: Scientific machine learning benchmarks. *Nature Reviews Physics 4* (2022), 413–420. 9

[UK 22] UK HEALTH SECURITY AGENCY: *GOV.UK Coronavirus (COVID-19) in the UK*. [Online]. Available https://coronavirus.data.gov.uk/, 2022. Accessed: 25/03/2022. 2

[WaCP*15] WALKER R., AP CENYDD L., POP S., MILES H. C., HUGHES C. J., TEAHAN W. J., ROBERTS J. C.: Storyboarding for visual analytics. *Information Visualization 14*, 1 (2015), 27–50. doi:10.1177/1473871613487089. 3

[WFRR20] WILLIAMS R. L., FARMER D., ROBERTS J. C., RITSOS P. D.: Immersive visualisation of COVID-19 UK travel and US happiness data. In *Posters of the IEEE Conference on Visualization* (Oct. 2020). 2

[WGH*24] WU G., GUO S., HOFFSWELL J., CHAN G. Y.-Y., ROSSI R., KOH E.: Socrates: Data story generation via adaptive machine-guided elicitation of user feedback. *IEEE Transactions on Visualization and Computer Graphics 30*, 1 (2024). 3

[WLF*19] WANG Q., LI Z., FU S., CUI W., QU H.: Narvis: Authoring Narrative Slideshows for Introducing Data Visualization Designs. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (2019), 779–788. doi:10.1109/TVCG.2018.2865232. 2

[WRC*21] WANG Z., ROMAT H., CHEVALIER F., RICHE N. H., MURRAY-RUST D., BACH B.: Interactive Data Comics. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (2021), 944–954. doi:10.1109/TVCG.2021.3114849. 2, 3

[WSZ*20] WANG Y., SUN Z., ZHANG H., CUI W., XU K., MA X., ZHANG D.: DataShot: Automatic Generation of Fact Sheets from Tabular Data. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 895–905. URL: https://doi.org/10.1109/TVCG.2019.2934398, doi:10.1109/TVCG.2019.2934398. 2, 3

[WWF*19] WANG Z., WANG S., FARINELLA M., MURRAY-RUST D., HENRY RICHE N., BACH B.: Comparing effectiveness and engagement of data comics and infographics. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (New York, NY, USA, 2019), ACM, p. 1–12. doi:10.1145/3290605.3300483. 2

[WWZ*21] WU A., WANG Y., ZHOU M., HE X., ZHANG H., QU H., ZHANG D.: MultiVision: Designing Analytical Dashboards with Deep Learning Based Recommendation. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (2021), 162–172. doi:10.1109/TVCG.2021.3114826. 3

[YC18] YOUSUF B., CONLAN O.: Supporting Student Engagement Through Explorable Visual Narratives. *IEEE Transactions on Learning Technologies 11*, 3 (2018), 307–320. doi:10.1109/TLT.2017.2722416. 2

[YLRC10] YU L., LU A., RIBARSKY W., CHEN W.: Automatic Animation for Time-Varying Data Visualization. *Computer Graphics Forum 29*, 7 (2010), 2271–2280. doi:10.1111/j.1467-8659.2010.01816.x. 3

[ZOM19] ZHI Q., OTTLEY A., METOYER R.: Linking and layout: Exploring the integration of text and visualization in storytelling. *Computer Graphics Forum 38*, 3 (2019), 675–685. doi:10.1111/cgf.13719. 2

Supplementary Materials

Feature-Action Design Patterns for Storytelling Visualizations with Time Series Data

S. Khan, S. Jones, B. Bach, J. Cha, M. Chen, J. Meikle,
J. C. Roberts, J. Thiyagalingam, J. Wood, P. D. Ritsos

Introduction to Appendices

These set of appendices provide further details about the technical aspects of the method that is described in Section 4 and illustrated in Figure 7, including:

- Appendix A: The general structure of a feature-action table illustrated in the top-right of Figure 7.
- Appendix B: The general structure of a meta-storyboard program for the algorithmic workflow illustrated in the lower-half of Figure 7.

Appendix A: Feature Action Tables

As shown in Figure 2 (the 2nd box) and Figure 7 (top-right), a meta-author formulates the specification of a set of feature-action patterns, and then discusses the specification with a developer, who defines a feature-action table that can be read by a program. Normally, the program encodes a meta-storyboard and is used to create storytelling visualization.

The meta-author's specification can be documented in many forms, such as word files, spreadsheets, hand-written and hand-drawn notes, and so on, while the **feature-action table**, defined by the developer, must be in a format that the meta-storyboard program can read. We currently use CSV files for feature-action tables. Each feature-action table has the following columns:

1. **TimeSeriesId**: `<id>` of a time series in the storyboard.
2. **Feature**: `<name>` of a feature extraction function in the Meta-Storyboard Features API (MSB Features).
3. **FeatureParams**: a list of parameters for the feature concerned in the form of `<parameter name>:<parameter value>, <parameter name>:<parameter value>, ....`
4. **Rank**: a rank value $\in [1, 10]$ assigned to the action for the detected feature.
5. **Action**: `<name>` of an action function in the Meta-Storyboard Actions API (MSB Actions).
6. **ActionParams**: a list of parameters for the action concerned in the form of `<parameter name>:<parameter value>, <parameter name>:<parameter value>, ....`
7. **Text**: a text section in the form of `<text>` defined as string template. Note that the list of parameters for the action is in the column **ActionParams**.
8. **Comments**: optional comments by the meta-author or the developer. These are not designed to be understood by computer programs.

A meta-authoring storyboard may need to deal with multiple time series, e.g., a categorical time series for dates and events, two numerical time series for the number of cases in two different regions, and the "difference time series" derived from the numerical time series. Features can be defined for each time series independently. The current version of the software assumes that complex features for characterizing interactions among two or more time series are detected in two ways: (a) through a derived time series resulting from a subroutine that processes two or more time series, such as a difference function, (b) through the Gaussian mixture process following the feature extraction step.

A.1 Meta-Storyboard Features API

Conceptually, features are defined as functions in the MSB Features API, and the function names and their parameter names are used as predefined constants in feature-action tables. Note that developers may include undefined feature names or parameter names in a feature-action table during their communication with meta-authors with the knowledge that they will add such features and/or parameters into the MSB Features API. Meanwhile, all our storyboard programs are able to ignore undefined features and parameters in a way similar to web browsers ignoring unknown HTML tags.

In order to implement and manage similar feature functions in an object-oriented manner, we group similar feature functions using a high-level feature construct. Table 1 shows a list of high-level feature constructs. Each construct is defined in conjunction with a list of parameters, such as (`GTE:1000, LTE:2000`) for a value range [1000, 2000]. This allows high-level constructs to be decomposed into low-level constructs. For example, the PEAK feature has low-level constructs for detecting, e.g., the k-th peak or a peak with additional slope criterion. For each construct, a pre-defined function, `detectFeatures()`, analyzes the given time series according to the feature parameters in the feature-action table.

Table 1: *The main features implemented in the MSB Features API for specifying time series features.*

Feature Name	Description
	numerical features
FIRST	encountering the first data point
CURRENT	setting action location at the current data point
SEARCH	setting a search range (default: to the end)
LAST	encountering the last data point
MIN	encountering the MIN value
MAX	encountering the MAX value
VALUE	encountering a segment with specified value range
STDEV	encountering a segment with specified STDEV range
PEAK	encountering a peak (not necessarily MAX)
VALLEY	encountering a valley (not necessarily MIN)
RISE	encountering a rising segment
FALL	encountering a falling segment
SLOPE	encountering a segment with specified slope range
	categorical features
EVENT	encountering an event with a specific event label

A.2 Meta-Storyboard Actions API

Similar to features, actions are conceptually defined as functions in the MSB Actions API, and the function names and their parameter names are used as predefined constants in feature-action tables.

Similarly, all our storyboard programs are able to ignore undefined actions and parameters. Similar action functions are also grouped together using a high-level feature construct. Table 2 shows a list of high-level action constructs. Action parameters facilitate the decomposition of a high-level construct into low-level constructs.

When a feature is detected, the corresponding action is not activated immediately in order for all feature-action patterns to be further processed (e.g., Gaussian mixture, segmentation, and rank-based event selection). Hence when a feature is detected, a function `registerActions()` registers an action at the data point where the feature is encountered.

Table 2: *The main actions implemented in the MSB Actions API for specifying visualization actions.*

Action Name	Description
DRAW_DATA	display a set of data points
DRAW_AXIS	display an axis or all axes
TEXT_BOX	display a text string in a predefined message box
TEXT_POS	draw or remove a text string on the plotting canvas
LINE	draw or remove a highlighting line
CIRCLE	draw or remove a highlighting circle
RECTANGLE	draw or remove a highlighting rectangle
ARROW	draw or remove a highlighting arrow
NTS	highlight or dehighlight a numerical TS segment
CTS	highlight or dehighlight a categorical TS segment
NODE	highlight or dehighlight a graph node
CONNECTOR	highlight or dehighlight a graph edge (connector)
AXIS	highlight, or dehighlight a section of an axis
PAUSE	pause the animation for a specific amount of time

A.3 Example Feature-Action Tables

We present two example feature-action tables, Tables 3 and 4. They are for illustration and they are simpler than the meta-storyboards that were developed in our case studies.

Appendix B: Meta-Storyboard Programs

B.1 Generic Meta-Storyboard Program

Programs for meta-storyboards have very similar structures. A new program can be implemented easily by adapting an existing program. The structure of a meta-storyboard program typically has the following six steps:

STEP 1: Data Selection and Preprocessing

- Based on a user's input, the program selects the data. For example, with COVID-19 data, a user may select a region, the program assigns the time series for the selected region to TS1, which is the ID used in the feature-action table.
- For a more complicated storyboard, this step may include processes for creating derived data, e.g., computing the difference between two time series selected by a user.

STEP 2: Processing Feature-Action Table

- For each row in the feature-action table, the program
  - detects the specified feature,
  - registers the specified action against the feature, and
  - assigns the specified rank to the action.
- In addition to the parameters for features and actions, there is a data buffer shared by the feature detection function `detectFeatures()` and the action registration function `registerActions()`. The buffer allows `registerActions()` to access information such as the current data point, the previous data point before the feature detection (excluding the dummy feature CURRENT), and so on.
- When there are multiple time series specified in the feature-action table, this step processes all time series concurrently.

STEP 3: Integrated Multiple Time Series

- If there are multiple time series, the program creates an integrated time series with combined ranks and all registered actions. The ranks are combined using a Gaussian Mixture Model (GMM).

STEP 4: Segmentation and Action Selection

- The program determines the number of time segments according to the length of the animation to be created. The default number is 1 segment, i.e., no segmentation is required.
- The program applies a segmentation algorithm to divide the integrated time series according to the specified number of segments.
- For each time series, the program selects the important feature-action pairs (with higher ranks) according to the time allowed for the animation of this segment.

STEP 5: Create Animated Visualization

- The program displays the selected actions as an animated sequence.

B.2 Software Prototyping, Adaption, and Generalisation

We implemented six different meta-storyboards, with different settings in terms of instructions from the meta-authors, application data, target users, feature-action tables, and visual representations. Three meta-authors were involved in creating the three storyboards for COVID-19. One developer wrote programs for prototyping these meta-storyboards initially, and a second developer ported the programs to the RAMPVIS server with some adaptation. The second developer subsequently implemented three ML storyboards, which were created by two meta-authors. The generic program described in Appendix B.1 was formulated by the developers after bringing all these programs and the associated experience together. We are continuing this development as part of an ML infrastructure.

Table 3: *An example of a feature action table used to create a single-region meta-storyboard with COVID-19 data. The visual results are similar to the screens shown in Figures 4 and 6. Empty cells are interpreted as null values.*

ID	Feature	Feature Parameters	Rank	Action	Action Parameters	Text	Comment
TS1	FIRST		10	DRAW_AXIS			
TS1	VALUE	GT:0	5	DRAW_DATA			> 0
TS1	CURRENT		5	TEXT_BOX	BOX:1	{REGION} recorded its first COVID-19 case.	
TS1	SEARCH	UPTO:28	10	DRAW_DATA			up to 28 days
TS1	RISE	SLOPE_GTE:15	5	TEXT_BOX	BOX:1	The number of cases grew.	
TS1	SEARCH	UPTO:28	10	DRAW_DATA			up to 28 days
TS1	SLOPE	GTE:10	6	TEXT_BOX		By {DATE}, the number of cases continued to climb higher. Let us all make a great effort to help bring the number down. Be safe and support the NHS.	case A
TS1	SLOPE	LTE:-10	6	TEXT_BOX		By {DATE}, the number of cases dropped noticeably. Excellent effort. Be safe and support the NHS.	case B
TS1	SLOPE	GT:-10, LT:10	3	TEXT_BOX		By {DATE}, the number of cases remained low. We should continue to be vigilant.	case C
TS1	PEAK		10	DRAW_DATA			default parameters
TS1	CURRENT		10	TEXT_BOX		By {DATE}, the number of peaks at {HEIGHT}.	
TS1	CURRENT		10	CIRCLE	SIZE:10, STROKE_WIDTH:3, COLOR:#E84A5F, OPACITY:0.6		
TS1	CURRENT		5	PAUSE	TIME:10		10 sec.
TS1	SEARCH	UPTO:28	5	DRAW_DATA			up to 28 days
TS1	FALL	SLOPE_LTE:-15	5	TEXT_BOX		By {DATE}, the number of cases came down noticeably. We should continue to be vigilant.	
...	...	...	...	...	...	...	...

Table 4: *An example feature action table for the ML provenance story as shown in Figure 9(a).*

ID	Feature	Feature Parameters	Rank	Action	Action Parameters	Text	Comment
TS1	FIRST		10	DRAW_DATA			
TS1	CURRENT		10	TEXT_BOX	BOX:1	A newly-trained model achieved testing accuracy of {TEST}% and training accuracy of {TRAIN}%, denoted {TEST}% [{TRAIN}%].	
TS1	CURRENT		10	CIRCLE	SIZE:10, OPACITY:0.6, STROKE_WIDTH:3, COLOR:#FFA500		
TS1	CURRENT		10	NODE	SIZE:5, COLOR:#FFA500, OPACITY:0.6		
TS1	CURRENT		10	TEXT_POS	X:10, Y:180, COLOR_TEXT:#EC5800, COLOR_BG:#808080, FONT_SIZE:13	Accuracy: {TEST}%	
TS1	CURRENT		10	NODE	SIZE:5, COLOR:#FFA500, OPACITY:0.6		
TS1	MAX		10	DRAW_DATA			
TS1	CURRENT		10	TEXT_BOX	BOX:1	On {DATE}, a model achieved the best testing accuracy {TEST}% [{TRAIN}%].	
TS1	CURRENT		10	CIRCLE	SIZE:10, COLOR:#008000, OPACITY:0.6, STROKE_WIDTH:3, VISIBLE:TRUE		
TS1	CURRENT		10	NODE	SIZE:5, COLOR:#008000, OPACITY:0.6, VISIBLE:TRUE		
TS1	CURRENT		5	PAUSE	TIME:15		15 sec.
TS1	LAST		10	DRAW_DATA			