

ChronoCorrelator

Citation for published version (APA):

van Dortmont, M. A. M. M., van den Elzen, S., & van Wijk, J. J. (2019). ChronoCorrelator: enriching events with time series. *Computer Graphics Forum*, 38(3), 387-399. <https://doi.org/10.1111/cgf.13697>

Document license:

TAVERNE

DOI:

[10.1111/cgf.13697](https://doi.org/10.1111/cgf.13697)

Document status and date:

Published: 10/07/2019

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

ChronoCorrelator: Enriching Events with Time Series


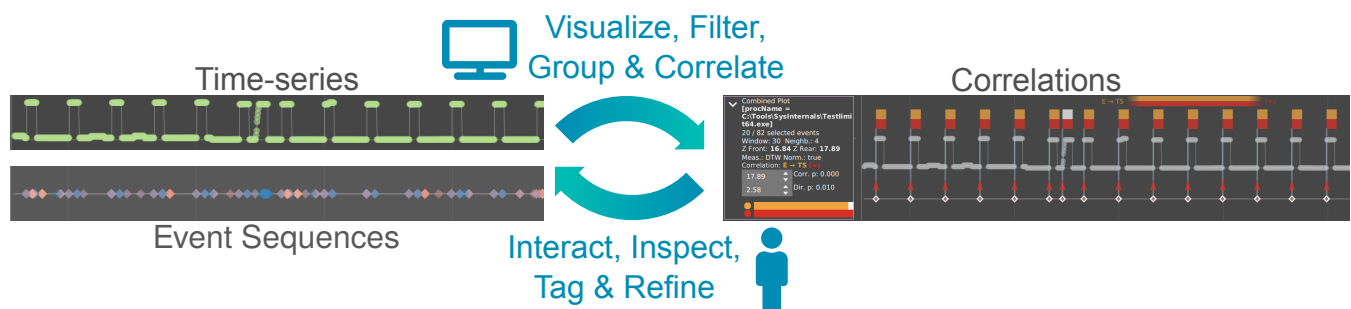
M.A.M.M. van Dortmont^{1,2} , S. van den Elzen² and J.J. van Wijk¹¹Eindhoven University of Technology²SynerScope B.V.

Figure 1: The concept of ChronoCorrelator: enable users to combine both event sequences and time series to discover correlations between the two using a visual analytics approach and enable them to filter, group and inspect events on the fly.

Abstract

Event sequences and time series are widely recorded in many application domains; examples are stock market prices, electronic health records, server operation and performance logs. Common goals for recording are monitoring, root cause analysis and predictive analytics. Current analysis methods generally focus on the exploration of either event sequences or time series. However, deeper insights are gained by combining both. We present a visual analytics approach where users can explore both time series and event data simultaneously, combining visualization, automated methods and human interaction. We enable users to iteratively refine the visualization. Correlations between event sequences and time series can be found by means of an interactive algorithm, which also computes the presence of monotonic effects. We illustrate the effectiveness of our method by applying it to real world and synthetic data sets.

CCS Concepts

• **Human-centered computing** → Visual analytics; Interaction design; • **Mathematics of computing** → Time series analysis;

1. Introduction

Event data is common across many domains, from websites logging the activity of their visitors and software tracking the interaction between its components, Internet of Things applications producing various sensor and event logs, to doctors keeping records about their patients treatments. The common theme is the desire to use this data to monitor or improve the processes involved or to find the source of a failure. Aside from their type and associated timestamp, events often carry additional information (e.g., the user logging in, the medication received by a patient and its dosage, the web page being visited), depending on the domain and type of event involved. In essence events often represent a change of state.

In many domains a second type of temporal data is being recorded in the form of time series. This sensor data is often not the direct

result of an event that occurred, but a recording, at regular intervals, of some variable belonging to the actors involved (e.g., the blood pressure of a patient in the ICU, a temperature reading of a critical component in a machine, the memory usage of a web server). Time series generally do not represent a clear change of state, due to their continuous nature, unlike event data. They can however provide more information about the overall state of an entity that may not be directly obtainable from event data alone. While some approaches might simply treat this data as another type of event, our approach is different. To enable simultaneous exploration of event and time series data our method facilitates a user-oriented approach where filtering, selections and attribute based partitioning are combined with a user-controllable projection of the time series data onto the event data, enabling them to enrich the event data with additional

context about the underlying activities and entities. The challenge here is that generally there is no data available to directly link patterns in the time series data to specific events, because of the way they are captured, separately and often for different purposes and possibly even from independent sources. Thus, a means of correlating events and time series is required. Simply augmenting the events with some aggregated attribute based on the time series, such as a mean, extremum or some other basic statistical measure within a window around the event would not be able to encode the complexity of patterns that may exist within the time series [Ans73]. Instead we chose an algorithmic approach to finding the linked patterns.

Our main contributions are:

- an exploration and analysis method centered around an algorithm that enables users to interactively find correlations between the event sequences and time series data by encoding the results in glyphs;
- simple local measures to find local deviations to the global correlation results found by the algorithm;
- interactive, linked views displaying both *event sequences* and *time series* data with a shared temporal axis, and;
- a prototype design (with best practices, design decisions and limitations) showing the interaction flow using real world and synthetic data sets.

Our work is based in part on discussions with security domain experts working in a Security Operations Center (SOC), looking to take a more exploratory threat hunting approach to cyber security, during the early stages of development.

1.1. Tasks

Based on these initial discussions with the security domain experts we came up with the following list of tasks to aid the users:

- **Determine a correlation between selected events and a time series.** The security experts mentioned doing this by querying around known alerts in various log files, looking for unusual patterns in other log files around the same time, trying to find patterns. The experts explained that linking event data to metrics such as memory usage is generally done manually and finding relations, if done at all, is done by hand.
- **Determine if, for a given set of events, there are subsets which show a stronger correlation when splitting the events into subsets based on some attribute;** for example in one of our usage scenarios we looked at CPU and memory usage time series and how they relate to logs containing information about application activity. By splitting events, for example per event type, user or application name, we investigate whether any unusual activity in the time series can be correlated to an application or user in the logs without splitting events into predetermined sequences a-priori. The domain experts mentioned that they preferred an approach that did not force them into a predefined projection of the data, as this was a major constraint of their existing workflow, and not ideally suited for the threat hunting scenario, i.e., dividing the events into sequences based on a "server" attribute might make looking at the data from a user perspective more difficult. Thus our approach does not force an a-priori split on the data.

- **Find events where the local pattern differs from the global pattern;** such deviations can indicate an outlier among the events.
- **Annotate findings, so that results can be used in later iterations of the exploration.** For example, if we find suspicious logins early on in our exploration, we can tag them for later use.

This leads us to the following user requirements:

- **explore** - a means to explore both events and time series;
- **correlate** - a method to find relationships between events and time series;
- **verify** - a way to inspect and verify the results;
- **detect** - a way to detect local deviations from the global pattern;
- **annotate** - a way to annotate the events.

To achieve these requirements we provide a system that

- enables users to iteratively discover events of interest by means of a visual exploration environment that shows both events and time series in a shared temporal space;
- connect those events to potentially relevant time series, through the use of an interactive, user-controlled, algorithm that can find potential correlations;
- visually inspect the results to determine their accuracy, and enable users to manually adjust threshold parameters;
- detect local deviations from the correlation results by means of visual comparison of the local pattern against the global one; and finally,
- annotate the events by means of a tagging operation, which feeds back into the exploration environment, allowing users to iterate on their findings.

The remainder of this paper is structured as follows: in [Section 2](#) we discuss related work. Next, we describe our approach to event sequence exploration and our prototype in [Section 3](#). In [Section 4](#) we discuss the design decisions with respect to interaction, data manipulation and visualization centered around the correlation algorithm used. Several usage scenarios [[IIC*13](#), [SMM12](#)] that guided our design are given in [Section 5](#). Limitations and observations are discussed in [Section 6](#). Finally, conclusions and future work are provided in [Section 7](#).

2. Related work

This section is divided into three separate parts, first discussing previous work that focuses specifically on event sequences, then providing an overview of existing work that focuses on time series data. Finally, we discuss existing work that combines these two types of temporal data. We also discuss some research outside the visualization literature that focuses on similar problems.

Event Sequences Event sequence exploration has been extensively studied within the visualization community [[DSP*17](#), [PMR*96](#), [AMST11](#)]. A common representation for events is to show them as linear sequences of glyphs, while providing interactive means to manipulate the sequences to perform tasks such as querying or exploration [[BM13](#), [Shn96](#)]. For example, Plaisant et al. [[PMR*96](#)] describe LifeLines, a system that gives a complete overview of a single patient's records, showing patient treatments and problems in a chronological overview. Wang et al. [[WWPS10](#)] build on this in LifeLines2 by adding the ability to explore and query multiple patient records at once and adding the ability to align

events across patients. Both systems represent events as glyphs on a linear axis, representing time. In the case of LifeLines2 the system can represent both absolute and relative time, the latter being used when aligning events to, for example, the first occurrence of a particular event type per patient. TimeSpan [LPK*16] focuses on the exploration and comparison of the steps involved in the treatment of stroke patients, with the goal of finding ways of minimizing the time between patients having a stroke and patients receiving proper treatment. In EventFlow [MLL*13] Monroe et al. provide mechanisms to iteratively simplify and filter complex event sequences to find meaningful patterns. Similan2 [WPTMS12] enables users to query event sequences to find patterns similar to the queried pattern, while giving them control over the parameters of the underlying similarity measure. Similarly, Chen et al. [CXR18] discuss a visual analytics system that enables users to match and summarize sequences using a visual query interface, that can find matches despite the presence of noise, such as slight variations in the event order. Krstajic et al. [KBK11] use a kernel density estimate based approach to improve the scaling of event series, showing moments in time with a higher event frequency as larger peaks. Cappers et al. [CvW18] discuss a system called EventPad to explore event sequences by use of multivariate regular expressions, focusing on querying, simplification and iterative refinement. Combining event sequences with other types of data is not new. For example, Krueger et al. [KTT17] present a system where event sequences are explored together with geo-spatial information to gain insight into movement sequences.

Time Series Visualization of time series data has also been broadly studied [AMST11, MS03, SC00]. The distinction we focus on is the discrete and often multivariate nature of event data versus the often continuous nature of time series. With TimeSearcher [HS04] Hochheiser and Shneiderman introduce a system that enables the exploration of multiple similar time series by use of a visual selection and filtering mechanism. Balakrishnan et al. [BCZP11] present a visual analytics approach to exploring time series, where users can iteratively construct an analysis pipeline. Saito et al. [SMY*05] present a two-tone pseudo-coloring technique that reduces the vertical space required to visualize one-dimensional data. Similarly Heer et al. [HKA09] discuss a variation of the often-used line or area chart that folds charts back onto themselves to reduce the amount of space required to display them. Federico et al. [FHR*14] extend this to qualitative data. Hripcsak et al. [HAP11] designed an approach to correlate time series of lab results to each other and to concepts mapped onto binary time series. Köthür et al. present an approach [KWS*15] to correlate multiple time series with each other. Other systems exploit the cyclical nature of time to display data differently. Van Wijk and Van Selow [vWvS99] use a calendar-based approach to clustering of multiple time series, while Weber et al. [WAM01] map temporal data on a spiral or helix to enable simultaneous display of both linear and cyclical properties of the data. We however choose the linear approach, because this enables us to display multiple event sequences and time series in a juxtaposed fashion.

Combined Approach There is some previous work, mainly in the medical domain, that combines both event sequences and time series into a single framework. Rind et al. [RWA*13] provide an overview of tools and methods to explore and query electronic health records, including several that combine event data with time series

data for the same patient. For example Shahar et al. [SGBBT06] introduce KNAVE-II, a system that aids the analysis of patient records containing both event data and time series by integrating with a database of domain knowledge. It displays both treatment events and time series data on a single patient in a juxtaposed visualization. It supports analysis and classification of the time series using the database, but does not correlate events with the time series. Similarly VisuExplore [RAM*11] is a system showing both time series and event data side-by-side showing patient treatment information for a single patient. Wanner et al. [WSJ*14] discuss a system that enables users to link interesting intervals extracted from a time series with events extracted from textual data. Gschwandtner et al. [GAK*11] propose a system, CareCruiser, that enables medical experts to align treatment plans with data on patient condition to determine the effects of treatments on the patient, by means of color coded highlighting of the time series which can show the distance to some desired value, the progress over time or the slope of the time series. This encoding depends on medical domain knowledge to provide information about which values are acceptable and which are not. It does not however directly allow for correlations between events and time series. The way our approach differs from the systems mentioned here is that it does enable users to directly correlate between event sequences and time series, thus unlocking a more powerful way to explore the data.

Other Fields Other fields outside of visualization also seek to tackle similar problems. Xiao et al. [XYF*17] use Recurrent Neural Networks (RNNs) to analyze time series & event data and apply it to, amongst others, electronic health records to, for example, predict sequences of diagnoses for patients in the ICU. Dunkl et al. [DR-MGF14] use process mining techniques, such as Decision Point Analysis, to analyze event sequences with time series data. In this approach they attempt to bring the time series data into the event sequences by for example adding new events when the time series exceeds a given threshold or by adding additional attributes to existing events. Luo et al. [LLL*14] discuss an algorithm that calculates a global correlation between time series and event sequences. It is in fact this algorithm that we chose to build upon for this paper. By extending the algorithm with local measures and by using visualization and interaction, we show that deeper insights can be gained than by using the algorithm alone.

3. Approach

The disparate nature of event sequences and time series data makes it difficult to gain insight into the relation between these two data types, despite their shared temporal nature. Users are not only interested in the presence of patterns in time series data, but also how these relate to the presence of certain events in a related event sequence, and how this information might be used to gain insight into the cause of these events or perhaps to enable better prediction of future occurrences for similar events based on the signal provided by the time series. In order to do so, they need to be able to find which subsets of events relate to patterns present in the time series data.

As mentioned in Section 1.1, the initial requirements for our prototype came in part from discussions with security domain experts on a means of threat hunting: exploring various system logs and metrics to find new patterns.

To handle events and time series in a single application we chose

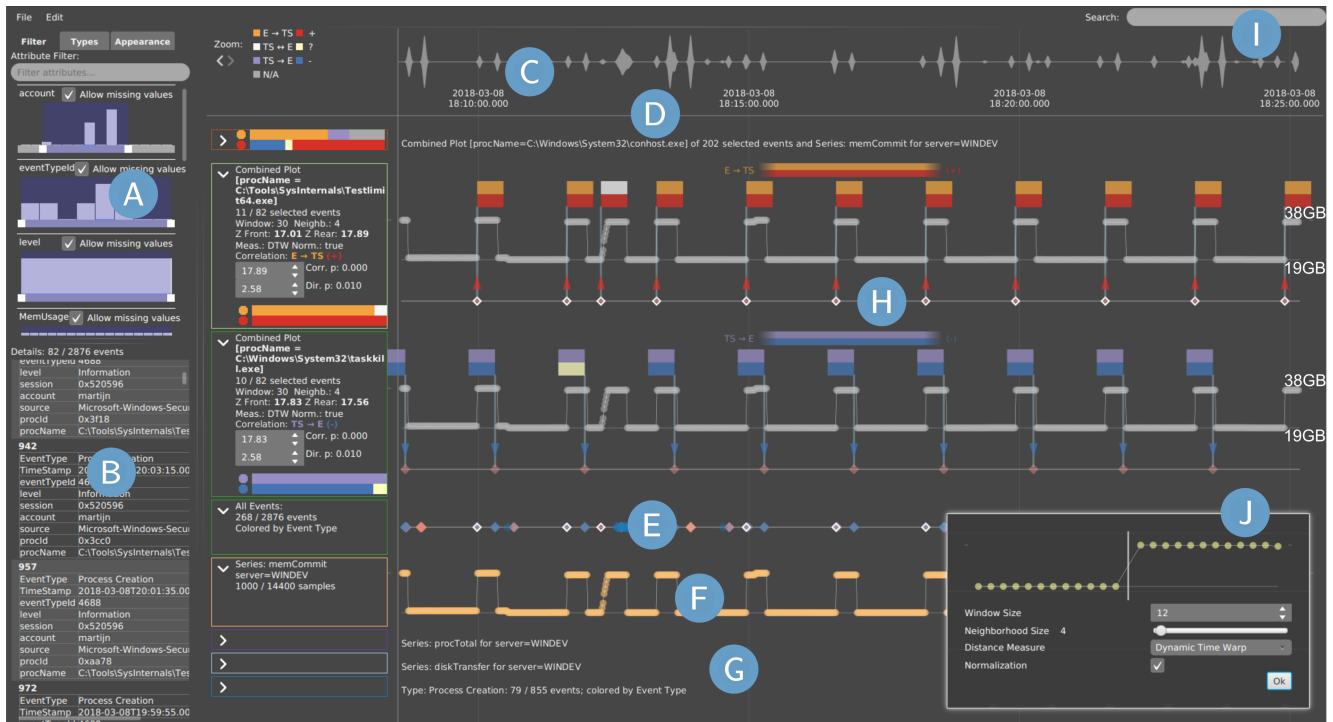


Figure 2: The main user interface of ChronoCorrelator: (A) Scented widgets to quickly filter out events; (B) Detail view; (C) A density plot showing the global density of events, as well as a legend for the correlation results; (D) Beneath the density plot: the shared time axis; (E) Event sequence; (F) Time series, represented by simple line charts; (G) Collapsed time series, event sequences; (H) A correlation result, showing events, time series and glyphs in a common chart. A summary is shown in the chart description to the left; (I) Search bar allows searching for events containing specific values; (J) Inset: the parameter GUI shown to select correlation settings.

for a representation where event sequences and time series are displayed juxtaposed vertically with a shared horizontal temporal axis, using proven visualization techniques. Event sequences are represented as a series of glyphs, sharing the horizontal time axis with the time series, which in turn are displayed as simple line-charts. When loading a dataset, users are initially presented with the events in a single event sequence and a chart per available time series. Operations on the data can introduce new sequences. For example, by grouping events by event type, users can create a separate sequence for each event type. By default, the event glyphs are colored by event type, but users can select attribute values to color events by and can even annotate events with custom attributes, by means of a text box, which can then be used to color the glyphs or create custom groupings.

The main user interface for the prototype is shown in Figure 2. The screen is divided into three sections. The top left shows a series of scented widgets [WHA07] enabling quick filtering of events being shown based on their attributes (Figure 2a). The bottom left shows detailed information on selected events, providing details-on-demand [Shn96] (2b). All available attributes and user added tags are listed. From here users can click on a tag to quickly select all events with that tag. The right side of the screen is used to show the event sequences (2e) and time series (2f). At the top a density plot shows the event density over time as a streamgraph [BW08] (2c). Beneath it, the event sequences and time-series are shown. The

main workflow of the prototype application, as shown in Figure 1, is centered around direct interaction with the sequences:

- Users can use the mouse to interact with the data to zoom & pan as well as select events by clicking or dragging the event sequences, enabling them to get details on the selected events. Alternatively, they can hover over a specific event to get details of the event in a tooltip.
- Users can collapse sequences, time series and correlation results that are not of interest (2g), and can trigger various operations from a context menu. From the context menu users can add or remove tags from selected events to annotate them, use a selected time series to filter event sequences by filtering out events that occur when the time series is within or alternatively outside a certain range and sort sequences. For example, this would enable a server administrator to filter out server events that occur during a period where the CPU usage is low.
- Transformations can be applied to time series, such as Z-normalization, resampling or a de-trending operation, which can help improve correlation results.
- Users can also run a correlation test based on an algorithm that is explained in depth in Section 4. Given a set of selected events and a time series, users are presented with a UI to configure the input parameters for the correlation algorithm, thus they are enabled to apply their domain knowledge for the configuration of the algorithm and it is not treated as a black box. The parameter UI (2j) shows a small section of aggregated time series, which

shows the average shape of the time-series before and after the selected events for the given window size, which helps the user to choose the right parameter values, and the tool will pre-seed the main parameters of the algorithm with reasonable values, based on the selected events and time-series. Once the algorithm has run, the results are shown in a combined event sequence and time series display (2h) that enables users to further interact with the parameters controlling statistical significance thresholds of the algorithm. By default the chosen values for the thresholds are set to give a reasonable level of confidence, but users can choose to either increase or decrease these thresholds. Given results found during correlation, users may wish to go back to refine their filters and selections or add tags to mark the selected events. The left area of a chart shows information about that item. In case of correlation results it also shows a summary view, which is visible even when the correlation result is collapsed.

- Instead of computing correlations on all events in a selection as a single sequence, users can also decide to perform a “Split & Correlate” operation, which splits the selection into separate sequences based on chosen event attributes and performs a correlation operation on the sequences separately. The results of this operation are ordered from highest to lowest confidence score.

An example of how ChronoCorrelator shows the results can be seen in Figure 2 where the results of a “Split & Correlate” operation are shown. The results of the correlation operation also enable users to compare the global result produced by the algorithm against the local patterns in the data, enabling them to find local deviations, as we discuss in Section 4.4.

The work of Du et al. [DSP*17] focuses mainly on event sequence analysis and exploration, but it enumerates strategies that can be used to outline the capabilities of ChronoCorrelator, to allow for easier comparison with existing tools. Our system supports extraction of records and categories (strategies S1 & S2), enables time windowing of the sequences by using time series as a filter (S5) and through the use of both predefined & custom categories it can be used to group events together (S9). The interactive discovery of potential correlations between events and time series, however, does not clearly fall in any of the defined categories, though in a way it can be considered as an instance of “aligning” (S4).

4. Correlation algorithm

Our approach to combining events and time series together is centered around the interactive application of the algorithm by Luo et al. [LLL*14] capable of detecting the presence, temporal order and direction of correlations between a set of events and a time series. By integrating this algorithm into a visual analytics workflow, users are enabled to quickly scan the possible parameter space and determine if correlations can be found. This enables them to answer questions like “Can events be predicted from fluctuations in a signal?” or “What effect do certain events have on a signal?”. We chose this algorithm because it is one of the few that focuses on the combination of time-series and event data, its relative simplicity and its ability to indicate not only the presence of correlation but also the direction of the relation (i.e., is it an event followed by a change in the time series or vice-versa) and the effect (i.e., are the events correlated with an increase or a decrease in the time-series). By enabling interaction between the users and the algorithm using

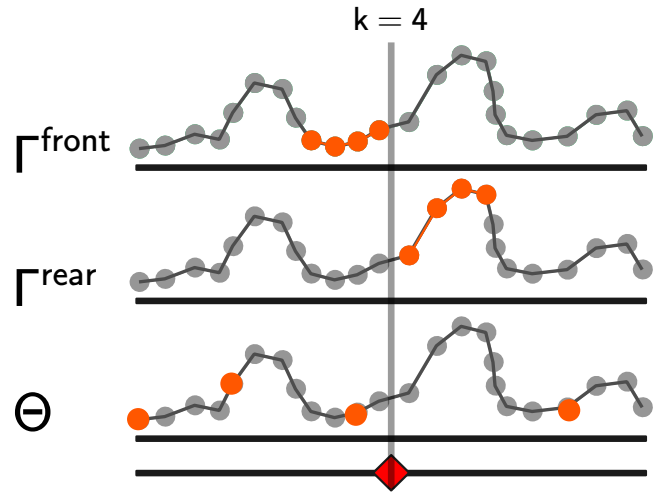


Figure 3: Sampling approach shown for a single event and window size $k = 4$. Γ^{front} takes the four preceding entries from the time series, Γ^{rear} takes the four entries following the event and for Θ entries are randomly sampled from the entire time series. This forms three samples of length k . One directly preceding the event, one directly following it and one bootstrapped from the entire time series.

visualization and interaction techniques we enable users to refine the parameters of the algorithm. Thus, we do not treat it as a black box, but choose to enable direct control of its parameters by the user, while supporting the user by providing good default settings based on the data and visual feedback on their choices. One of the limitations of the algorithm, however, is that it produces one global outcome for all selected events. To enable users to detect local deviations to the global pattern, we also introduce a pair of local measures, which we detail in Section 4.2. By visualizing the results side-by-side we enable users to detect local deviations from the global pattern.

4.1. Global algorithm

In this section we briefly describe the concept behind the algorithm for determining global correlations, both to introduce the terminology and symbols used later in the paper and to show how our local additions relate to the global results of the algorithm and to show where we deviate from the interpretation of these results. For a more in-depth and generalized discussion of the algorithm, including a discussion of its performance and accuracy, please refer to the work by Luo et al. [LLL*14]. The algorithm approaches the question of correlation as a two-sample problem. The input for the algorithm consists of: a set of events E , with $E = \{e_1, \dots, e_n\}$; a time series TS , with $TS = \{t_1, \dots, t_m\}$, where each t_i is a tuple of a timestamp and a value; a window size k and a neighborhood size r , where r is at most $n - 1$. The algorithm consists of several steps. In the first step it creates three sets of subsequences of length k each, all taken from the time series. Figure 3 shows the concept for window size $k = 4$. Each set will contain n subsequences, equal to the number of selected events. The first two, Γ^{front} and Γ^{rear} , consist of sampled subsequences of size k taken from the time series before and after the occurrence of each event in E , i.e., $\Gamma^{front} = \{l_k^{front}(TS, e_i), i = 1, \dots, n\}$, where $l_k^{front}(TS, e_i)$ is the subsequence of length k taken from series TS before the occurrence of event e_i . The same holds for Γ^{rear} for sub-

sequences that follow the events in E . For the edge case where there are not enough elements in the time series before or after an event, we chose to repeat the first or last element, respectively, to fill in the gaps. The third set, Θ , with $\Theta = \{\theta_0, \dots, \theta_n\}$ contains sampled subsequences θ_i , again of length k , but these subsequences are constructed by random sampling from TS . In short each of the three sets contains n samples of length k .

In the second step a nearest neighbor based method [Sch86] is applied for both Γ^{front} and Γ^{rear} . For this purpose ChronoCorrelator implements several distance metrics, including a simple Euclidean measure and the more robust DTW [BC94] metric.

Using Γ^{front} as an example, a combined sample pool $Z = \Gamma \cup \Theta$ is constructed. Each sample only occurs in either Γ or Θ , making the sets effectively mutually exclusive, though samples with identical values could potentially result from the random sampling. This does not violate the definition of mutually exclusive used here as long as a specific sample is clearly marked as being a member of *only one* of Γ^{front} or Θ , so that for each sample in Z we know whether it comes from Θ or Γ^{front} . For each sample in Z we wish to determine the set of r nearest neighbors. Now using an indicator function:

$$I_r(x, \Gamma^{front}, \Theta) = \begin{cases} 1, & \text{if } x \in \Theta \wedge NN_r(x, Z) \in \Theta \\ 1, & \text{if } x \in \Gamma^{front} \wedge NN_r(x, Z) \in \Gamma^{front} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

where x is a sample in Z and $NN_r(x, Z)$ indicates the r^{th} nearest neighbor of x in $Z \setminus x$, we can define a measure:

$$T_{r,p} = \frac{1}{pr} \sum_{i=1}^p \sum_{j=1}^r I_j(x_i, \Gamma^{front}, \Theta). \quad (2)$$

Here $p = 2n$ in our situation, which is the total size of the combined sample pool Z (see the work by Luo [LLL*14] for a more in-depth explanation why). This measure is the proportion, in the set of r nearest neighbors, for each subsequence taken from the same set (Γ or Θ) as the given subsequence. This ratio should be low if the samples are mixed well (i.e., we cannot distinguish between samples taken from Γ or Θ) and high when a large proportion of nearest neighbors comes from the same set, thus implying the underlying distributions of Γ and Θ are different. Then given a large enough sample pool size p ,

$$C = \sqrt{pr} \frac{(T_{r,p} - \mu_r)}{\sigma_r} \quad (3)$$

has a standard Gaussian distribution with

$$\mu_r = (\lambda_1)^2 + (\lambda_2)^2 \quad (4)$$

and

$$\sigma_r = \lambda_1 \lambda_2 + 4\lambda_1^2 \lambda_2^2 \quad (5)$$

where, for our case $\lambda_1 = \lambda_2 = n/p$. Now it is a simple application of the Gaussian distribution test, $C > \alpha$ with, for example, $\alpha = 2.58$ for $P = 0.01$, to determine if Γ^{front} and Θ are taken from different distributions. The same test is also applied to Γ^{rear} . If either set is found to be significantly different, we can state that E and TS are correlated ($E \sim TS$). If Γ^{front} and Θ are statistically different, we can state that the events E often follow significant changes in TS , which we can denote as $TS \rightarrow E$. If Γ^{rear} and Θ are statistically

different, we can state that the events E often precede significant changes in TS , which we can denote as $E \rightarrow TS$. Thus aside from being able to determine the presence of a correlation, the algorithm also detects the apparent temporal ordering of events and changes in the time series. Note that this implies that if an event occurs during a statistically different interval, where both the preceding and following samples in the time series statistically differ from Θ , this would be detected as a $TS \rightarrow E$ case, since the event is preceded by changes in the time series. However, this is where we diverge from the algorithm described by Luo et al. [LLL*14] by treating this case as a separate outcome $TS \leftrightarrow E$.

Finally, the algorithm can detect, if present, the effect type of the correlation, i.e., whether there is a noticeable monotonic effect on TS related to the occurrence of E and whether it is positive or negative, as defined in Luo et al. [LLL*14]. It does this by calculating a two-sample t-test, which, for our case, can be defined as:

$$t_{score} = \frac{\mu_{\Gamma^{front}} - \mu_{\Gamma^{rear}}}{\sqrt{\frac{\sigma_{\Gamma^{front}}^2 + \sigma_{\Gamma^{rear}}^2}{n}}} \quad (6)$$

This t_{score} can then be tested against a value α_{effect} (where for example $\alpha_{effect} = 2.58$ implies $P = 0.01$). If $t_{score} > \alpha_{effect}$ then there is a negative monotonic effect ($E \rightarrow TS$, $TS \rightarrow E$ or $TS \leftrightarrow E$) and if $t_{score} < -\alpha_{effect}$ there is a positive monotonic effect ($E \leftarrow TS$, $TS \leftarrow E$ or $TS \leftrightarrow E$). For values where $|t_{score}| < \alpha_{effect}$ no clear effect can be detected. Note that the algorithm provides us with a single result for the entire set E .

The runtime complexity of the algorithm is $O(n^2)$, mainly because of the distance measure that needs to be calculated for each of the subsequences in Z between each other.

Note that the use of the Gaussian distribution test requires that the underlying distribution should be normal, which is not something we can always guarantee for real world data sets. However, because of the central limit theorem, if the number of events is high enough, this should not be a problem. A common threshold for this is 30, however depending on the dataset, a smaller number may be sufficient. The prototype warns when a smaller number of events is used to test the correlation. We also choose a default p-value of 0.01 that is stricter than the more common 0.05, although the user is free to alter these values by adjusting the α parameter.

The algorithm is designed to operate on regularly sampled time series, however, the time series may be irregularly sampled. To compensate for this, ChronoCorrelator can resample the time series, although we found that, at least in our test cases, even with irregularly sampled time series the results were, in general, acceptable even without resampling, since the correlation scores differed little from the results when resampling.

4.2. Local measures

As mentioned earlier, we extend the global results produced by the algorithm with some local measures that provide an indication of local deviations. First, we determine the μ_{ts} and σ_{ts} of the entire time series. Then, for each event e_i , we take the corresponding subsequences from Γ^{front} and Γ^{rear} and determine if the means, μ_i^{front} and μ_i^{rear} , are significantly different from μ_{ts} :

$$\delta_i^{front} = |\mu_i^{front} - \mu_{ts}| > \sigma_{ts} \quad (7)$$

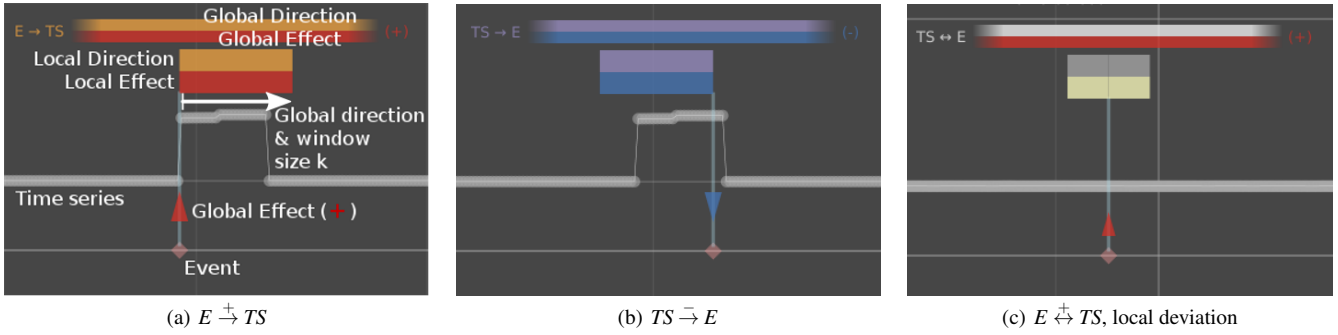


Figure 4: Glyphs used to convey correlation, direction and effect. The results are displayed over a combined display of the time series, in gray, and the event sequence (colored diamonds) beneath it. Results are encoded into flag-like glyphs. Images: (a) shows a correlation where an event is followed by a positive increase in the time series. The local correlation is in agreement with the global correlation. (b) An event following a drop in the time series. Again, both local and global scores agree. (c) An event where both Γ^{front} and Γ^{rear} differ significantly globally from the rest of the time series, however for the event shown we are unable to detect a local correlation, hence the gray color.

and

$$\delta_i^{rear} = |\mu_i^{rear} - \mu_{ts}| > \sigma_{ts}. \quad (8)$$

This produces one of 4 outcomes: if neither are true, we find no local correlation; if both are true, we find that $TS \leftrightarrow E$, if only δ_i^{front} , $TS \rightarrow E$ and if only δ_i^{rear} , $E \rightarrow TS$. This gives us a local indicator of both the correlation and its direction. For the local effect we use the following equation:

$$f_i = \frac{\mu_i^{rear} - \mu_i^{front}}{\sigma_{ts}}. \quad (9)$$

If $f_i \geq 1$ we consider it a local positive effect (+), if $f_i \leq -1$ we consider it a local negative effect (-) and when $-1 < f_i < 1$ we consider it undecided (?).

4.3. Robustness to noise

There are two main ways noise can be a factor for the global algorithm. Firstly, noise can be introduced on the event side, by running the algorithm on a set of events, where some of the selected events are different from the others and thus likely uncorrelated to whatever time series we are trying to match them against, or by leaving out some events that should have been included, though the later case is generally less problematic. From tests on both synthetic and real world data, we found that as long as the size of our event selection was large enough, having a few missing or extra events did not have that much of an impact on the end result. An important factor in this regard is the *neighborhood size* r . Since this does not have to be overly large [LLL*14], the algorithm is in general fairly robust against this type of noise.

The second type of noise is noise in the time series. To test the robustness of the algorithm, we tested this with a synthetic data set where we have multiple noisy time series, where we introduce the same, correlated, signal in each, but with varying signal strengths: we start with a signal that is about twice as strong as the strongest background noise and reduce that in steps to 75%, 50%, 40%, 30%, 25% and 10%. Each reduction in signal strength had a noticeable effect on the Z-scores, however we determined that even at 40% the algorithm was able to determine a correlation, even though at this level it was not always clearly visible to the observer anymore. At even lower levels we needed to adjust the confidence threshold α ,

though below 25% this meant that the correlation was no longer statistically relevant. We also noted that for the lower signal strengths, the relatively simple measures that were used to determine the default values for the window size k and neighborhood size r were no longer producing optimal values, however for the test we kept these at a fixed level. Another factor in this is the chosen distance measure used to determine the nearest neighbors. The default, DTW, seemed to produce generally more satisfactory correlations on our real data sets, however we noted that during our synthetic noise tests the simpler Euclidean measure was slightly more robust to this type of noise as the signal got weaker, though the difference was small.

4.4. Visual Representation of Correlation

To communicate the results of the correlation to the user and to enable the comparison of local results to the global ones, the following aspects should be shown:

1. presence and direction of the global correlation;
2. presence and type (positive or negative) of any global monotonic effect;
3. presence and direction of any local correlation;
4. presence and type of any local effect;
5. the relation of the above with the events and time series used for the correlation.

These aspects are shown in several ways in ChronoCorrelator. We display a textual representation of the correlation and the parameters used, next to the result sequence. This covers points 1 through 4, but does not communicate the relation between event sequence and time series effectively. To this end, we encode the individual event results as glyphs.

4.4.1. Glyph design

We show the correlation by means of flag-like glyphs projected on top of the event sequence and time series. The glyphs are positioned in such a way, as to enable per event verification of the global results, both against the local results and the actual time series in question.

The glyphs consist of three main parts, as shown in Figure 4. The first part is a vertical line or “flag post” starting in the event belonging to the glyph to the top of the time series. This enables a quick comparison of the event position in relation to the related

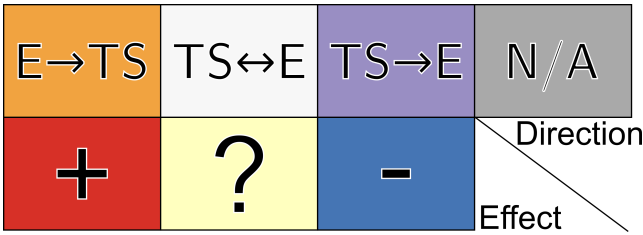


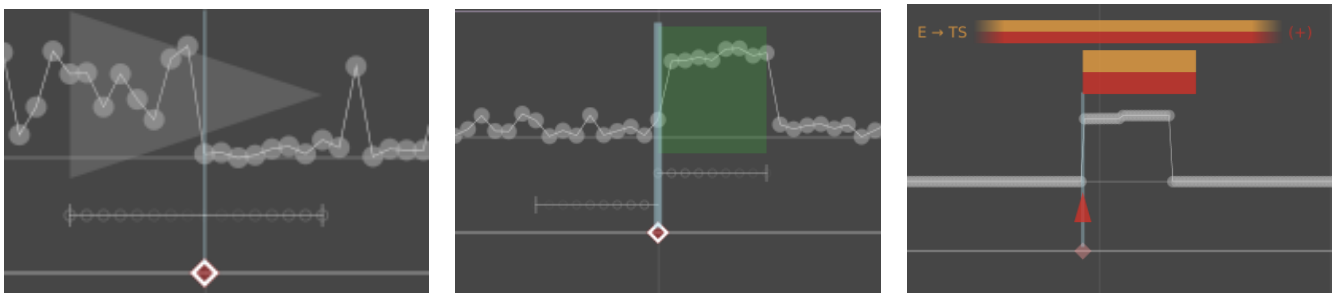
Figure 5: Legend for the color scale used in the application.

pattern in the time series. The second part is a marker, shown on top of the flag pole between the time series and the event sequence: An upward pointing arrow when there is a global positive effect, a downward pointing arrow when there is a global negative effect and a circle when no effect could be determined. The global effect is also encoded into the color of this marker. The colors are chosen from the ColorBrewer2 site [Bre13]. We chose a color-blind friendly diverging sequence (Figure 5), using a red-yellow-blue scale, where red encodes a positive and blue a negative effect, as shown in Figure 4. We chose the arrow representation, because it is a representation most users will be familiar with for encoding changing values, as seen, for example, in a stock ticker. The third part is the “flag” itself, which is placed above the time-series. This encodes three aspects:

- the direction of the global correlation; encoded into the direction of the flag: right for $E \rightarrow TS$, left for $TS \rightarrow E$ and both ways for $TS \leftrightarrow E$. This enables users to verify the correctness of the global direction of the correlation against the patterns in the time series. If they align there should be a deviation in the time series beneath the flag itself;
- the window size k ; the flag extends k time steps out from the flag post in the detected direction, allowing for easy inspection of the correlation in relation to the time-series. By encoding k into the flag size users can quickly see if the chosen value for k was correct or whether a different value may be warranted;
- the local direction and effect. The top half encodes the local correlation and direction in an orange-white-purple color scale, also chosen from the ColorBrewer2 site (Figure 5), and the bottom half encodes the local effect in a the same red-yellow-blue scale

used for the markers. A legend at the top of the screen explains the color-coding used.

We settled on this flag-like glyph after trying several other representations, shown in Figure 6. For example, in an earlier iteration (Figure 6(a)) we used arrow-head shapes, pointing left or right, instead of the simpler rectangular shapes we settled on, rendered on top of the time-series. However, these had several disadvantages. Firstly, it was not intuitively clear to users which temporal direction was being encoded by the different arrow directions. Secondly the arrow-heads suffered from poor readability when neighboring glyphs start overlapping, in particular, when zoomed out. This problem also exists for the chosen glyphs of course, however, they suffer less from the readability issue, in question. To deal with the overlap, we also implemented a hovering interaction which hides any overlapping flags near the mouse cursor and we highlight the section of the time-series connected to the highlighted glyph, so inspecting individual results remains possible. To further deal with the issue of overlap obscuring some of the glyph details, we also encode the global correlation, direction and effect into the color of two horizontal lines at the top of the chart, above the flags, as shown in Figure 4. The colors are the same diverging scales used for the local effect encoding in the individual glyphs: the top line encodes the direction in an orange-white-purple scale and the bottom line encodes the effect in a red-yellow-blue scale. This representation was added to enable quick comparison between the global and local results, encoded in glyphs, in particular, when zoomed out, or when glyphs of neighboring events overlap, partially obscuring the individual glyphs. Examples of this can be seen in Figures 4(a) and 4(b), where the local results seem to agree with the global results and Figure 4(c) where they do not. It should be noted that while the exact shape of the glyphs may be obscured in some cases, the flag colors, encoding the local results remain clearly visible, as can be seen in Figure 9, for example. Note that when this happens the glyphs start forming two horizontal lines, not unlike the ones at the top of the screen. Thus, even when the window size k is larger than the distance between two neighboring events, causing overlap and even when zooming out might cause some of the details of the glyphs to become difficult to distinguish, users are still able to quickly detect any local deviations to the globally detected results by simply comparing the two lines at the top of the chart with the lines formed by the glyphs.



(a) Arrow design, projected on top of time series, using whiskers to encode the global effect. The arrow covered both the window before and after the event and its direction indicated the correlation direction. (b) Flag design, showing only global information, using a red-green color scheme to encode the effect, which was also encoded in the whiskers beneath the flag. (c) Final flag design, showing local and global information, flag above the time series to avoid occlusion, added global color encoding at the top of the chart, using arrowhead instead of whiskers to encode effect.

Figure 6: Evolution of the glyph design.



Figure 7: Summary view example, global (circles) and local (stacked bars) results are not in agreement.

Furthermore, we also show a summary of the global and local results in the control area of the chart, even when it is collapsed, as can be seen in Figure 7. This summary view uses the same color coding as the flags to show the global correlation in the form of two colored circles and the local distribution of local correlations in the form of stacked horizontal bar charts, again using the same diverging color scales. The top circle encodes the direction of the global correlation: orange for $E \rightarrow TS$, purple for $TS \rightarrow E$, white for $TS \leftrightarrow E$ and finally gray if no correlation was detected. The top bar encode the distribution of the local results using the same color scheme. The bottom circle indicates whether the global effect was positive (red), negative (blue) or unclear (yellow). The bar next to it, in turn, encodes the distribution of the local effect results. This summary enables quick inspection of multiple correlations at both the global and local levels, even when the chart itself is collapsed, as well as enabling a quick comparison of the global and local results. In the case of Figure 7, we can see for example that the local and global results are not in agreement. The local scores seem to indicate it may be necessary to split the events further, since the results seem to largely fall into 2 separate groups, which was not clear from the global score alone.

5. Usage scenarios

We have applied our approach to both synthetic and real-world datasets. As discussed earlier we mainly used the synthetic data sets to verify the functionality and to test the robustness against noise. One dataset was created by monitoring memory usage, CPU usage and disk usage on a server to create the time-series, sampling every second. To generate the events we monitored user login, logout, process creation and termination over a period of 5 hours. During this test we ran some applications on top of any that were already running on the server in question. One generated a heavy CPU load, while another, designed to generate a considerable memory load, was repeatedly started and stopped both via a script and manually. A second set is based on an open data set containing information about the treatment of patients in an Intensive Care Unit (ICU) over a period of years, called MIMIC-III [GAG*00, JPS*16]. It contains treatment information, as well as information about admission, dismissal and, if applicable, death of the patients. It also has data on various monitored values which we can map to a time series per patient.

5.1. Server Administration

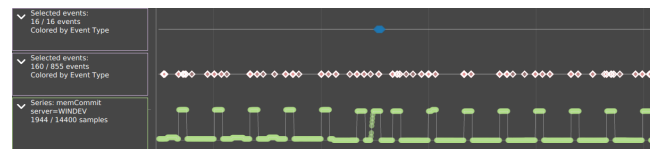
Server administrators attempt to keep servers operating in proper working order, ensuring that all authorized users can run their tasks. This can be particularly challenging in environments where workloads are dynamic, varying in type of task and resource requirements. To monitor the health of such systems, the administrators may set up basic alerting to monitor for situations where resource usage is high for a prolonged period or where critical tasks take more than an acceptable amount of time. It then falls to the administrator to determine the cause of the alert and, if necessary, remedy the situation.

The challenge here is that there is no direct link in the data from the alert events to the actual cause of the resource usage. Performance data is generally logged separately and in a different format from event logs. The alerting system itself may also be separate from the event logging system, though alerts are often also logged as events.

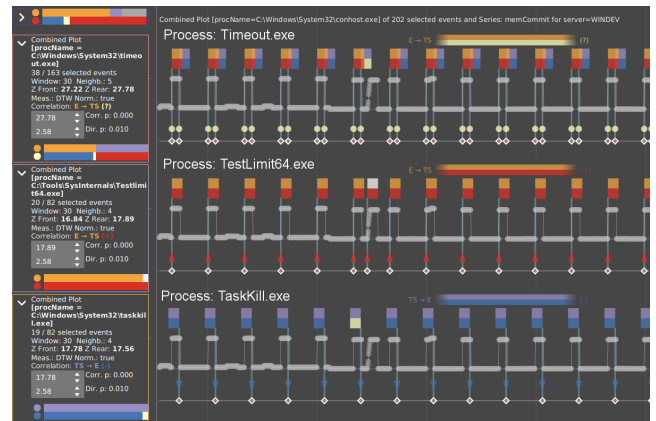
The task of finding the cause of the alerts is a type of root cause analysis, where the administrator is tasked with establishing a timeline leading up to the alerts with the stated goal of taking away the cause of the problems and, if possible, preventing any future occurrence. Traditionally, administrators use a variety of tools to analyze these cases, such as grep [BK09], LogStash [Tur13] or Nagios [Bar08]. They use these tools to first extract the relevant intervals from the performance logs based on the timestamp information provided by the alert. From there they have to query the event logs to extract the likely event sequences involved from the logs,



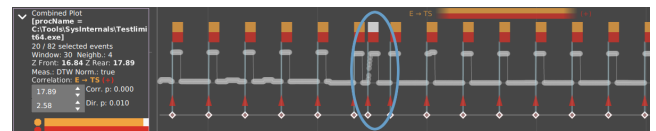
(a) Moving average of memory usage was above threshold (30GB), triggering alerts. The alert events are shown in the top sequence.



(b) Filtered and selected 855 Process Creation events.



(c) Split & Correlate operation applied on the *procName* attribute, creating separate sequences for each separate process name.



(d) Increasing the correlation thresholds improves direction detection and shows only one process correlated with all peaks. Circled correlation is different from others and is not correlated with *TaskKill* or *TimeOut*.

Figure 8: Server administration example, showing several steps in the exploration.

enabling them to manually inspect the sequences further to find a likely source of the alerts. By applying our visual analytics approach instead, we enable the administrators to simultaneously explore the event and performance data and by enabling them to quickly correlate events, such as the alerts to the time series, they can instantly see how the two data sources are related. In our example our administrator has received alerts about anomalous memory usage on one of the servers after hours. A quick look at the data shows that there is indeed increased memory usage, shown in Figure 8(a). The alerts were triggered because the average memory usage during a minute was higher than 30GB. To find a likely cause we filter and select the *Process Creation* events, which are logged to track the starting of an application (Figure 8(b)). To find correlations with memory usage we then apply the “*Split & Correlate*” operation on the selected events and the memory usage time series. This operation, as its name implies splits the event sequence on a given attribute (or attributes). In our case we chose to split on the *procName* attribute, which is the name of the executable being run. It then applies a correlation operation with suitable default settings for the window size k and the α threshold and orders the results based on how well they correlate. After inspecting the results we removed several sequences, which had no correlation or involved normal system processes, and are left with the correlations as seen in Figure 8(c).

We can now begin to form a hypothesis using our domain knowledge. The processes involved are three system utilities *TaskKill*, *TimeOut* and *ConHost* and one other application *TestLimit64*. We know that *ConHost* is involved in handling console processes. We also know that the process itself usually does not heavily use CPU or memory. It does however indicate that whatever is going on was most likely being performed from a console window, since there is a strong correlation. The other two system processes, *TaskKill* and *TimeOut* also do not have a heavy CPU or memory load. Their functions are respectively to kill a running task by process id or name and to introduce a delay in scripted environments respectively. We can see that the occurrences of *TaskKill* seem to correlate with most of the decreases in memory usage in the time window being investigated, which suggests that *TaskKill* may have been used to terminate whatever process was causing the memory usage. It is also clear that there are two instances of *TimeOut* correlated with almost each peak in the memory usage. As seen in Figure 8(d), there is only one process correlated with the start of each peak in the memory usage time series, including the circled peak: *TestLimit64*. We also note that this peak shows a different local pattern, which is one of the things that first drew our attention to it. A quick internet search tells us that this utility is used to simulate memory loads from the command line.

Given the above we can reconstruct the sequence of events that lead to the alerts. A script running from a command prompt (correlation with *ConHost*) is set up to run *TestLimit64* for a short period of time before killing the test utility using *TaskKill*. To perform the wait between the start of the utility and the termination, as well as the wait between ending one instance of the test utility and starting a new cycle *TimeOut* was used. However, the circled peak is not correlated to any instance of *TaskKill* or *TimeOut*. It is correlated to a *ConHost* and a *TestLimit64* instance. This suggests that it was a separate run of the test utility, most likely from a separate console. It is also this manual instance, together with the surrounding scripted

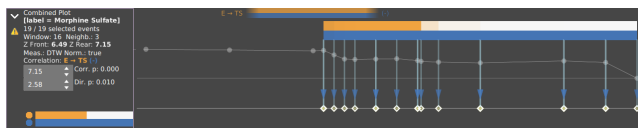
instances that seems to have triggered the alerts, which use a moving average to detect heightened memory usage.

As we, the authors, are the culprits in this particular example case, we can confirm that this is essentially what happened. While this particular case may have simply been a test for our prototype, the techniques combining visualization, interaction, algorithmic support and human domain knowledge described here can be used in general. This scenario is based on the threat hunting scenario laid out by the security experts we talked to during the initial design phases. While working on this usage scenario, we found and subsequently implemented several operations that make using the tool more efficient, for example, we found that the ability to manually sort the sequences and time series to be useful in keeping an overview. Also, the introduction of the “*Split & Correlate*” operation was the result of using our own prototype. It makes quickly correlating various partitionings of the events against a given time series simple. The combination of partitioning and further application of other operations is one that makes iteratively exploring data easier and can be seen in other tools as well [CvW18]. By presenting the results from the operation in an order where the strongest correlations are found at the top, ChronoCorrelator helps in sifting through the results. The use of visualization and our local measures helps in finding deviations in the global pattern that the main algorithm is not able to detect, such as the local deviations in Figure 8(c).

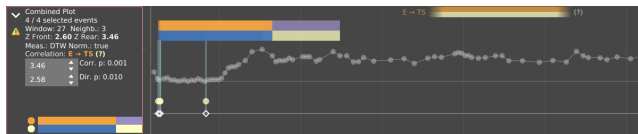
5.2. MIMIC-III

Our second usage scenario, based on the *MIMIC-III* set [GAG*00, JPS*16], describing the condition and treatment of patients in an ICU, contained real, but anonymized, records for some 40,000 patients, spanning over a decade. By applying ChronoCorrelator to this data set, we illustrate the generic nature of the method. We picked several patients from the data set, based on availability of data for the patients and investigated whether we could find any correlation between the treatment used and the patient vital signs, as recorded in the set. For the time series we chose several vital signs that are recorded for most, if not all, patients, such as: *Heart rate*, *Mean blood pressure*, *O₂ Saturation level* and *Respiratory rate*. This is one of the areas where this data set is different from the first usage scenario: the time series are generally only applicable to the events relating to one patient. Thus we investigated the patients separately. Also, the time-series in this data set are sparser than in the first scenario. In this scenario we used the notes by the health care professionals included in the data set as a ground truth for the patient status and treatments chosen. For our events we used the administering of medications, the admission or discharge from the ICU during a hospital stay, which in some cases could happen multiple times, and various procedures performed on the patients. Here we filtered out certain events, such as the administering of standard saline solution, as this is done for most patients, continuously, over the course of their stay in the ICU and correlating interval events, especially longer interval events, is difficult with the current approach, as we discuss in Section 6.

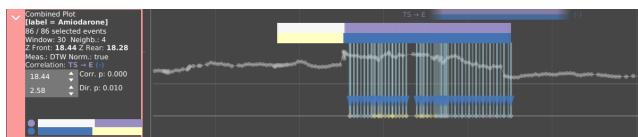
In one case, for example, where a patient was brought into hospital because of cardiac arrest we found a strong correlation between the administering of *Morphine Sulfate*, several hours before the eventual death of the patient, and both the *O₂* saturation and the heart rate. This was achieved by first splitting the events on their type and



(a) $TS \vec{\rightarrow} E$ correlation between O₂ Saturation and Morphine Sulfate being administered. Matched against notes indicating *Comfort Measure Only* order, shortly before patient death.



(b) Correlation between body temperature and notes mentioning *Arctic Sun*, a device meant to cool down patients. Note that due to the small number of events, evidence for effect was not strong enough.



(c) Splitting on medication shows a correlation between a drug called *Amiodarone* and heart rate.

Figure 9: Result images for the MIMIC-III data set.

then applying the *Split & Correlate* operation on the *INPUT* events, which describe the various substances being administered to the patients. This leads us to believe that Morphine, an analgesic, was used as a comfort measure to reduce discomfort toward the end. Validating this with the notes provided, suggests that the patient was extubated by order of the family around the same time and that a *CMO order* was given (Comfort Measures Only) by the family, as shown in Figure 9(a). We noted that the correlations where the local measures were in strong agreement with the global correlation score, like this one, often indicated a clear correlation.

In another patient we isolated *NOTES* events mentioning “Arctic Sun” and correlated them to the patient body temperature. There was a correlation between the two as can be seen in Figure 9(b). An internet search revealed that “Arctic Sun” refers to a medical device used to rapidly lower the body temperature of patients to reduce the risk of brain damage [Wik]. Note that due to the relatively low number of events involved evidence for the presence of a global monotonic effect was not strong enough to be statistically significant, which is one of the limitations of the algorithm, as we also discuss in Section 6.

Similarly, on the data for a final patient, suffering from sepsis, we applied the *Split & Correlate* operation to find a correlation between the administering of a drug called Amiodarone and the heart rate, which is an antiarrhythmic medication, which suggests it might have an effect on heart rate as seen in Figure 9(c). The drug is used to lower the heart rate which is in line with the fact that according to the correlation, the drug was first administered after the heart rate increased and the effect of the drug events is that the heart rate is lowered, which is reflected in the apparent decrease in the time series after the event ($TS \vec{\rightarrow} E$).

This scenario is different in that in most scenarios it would not make sense to explore the data of multiple patients together, as the

events relating to one patient would likely not be correlated to any time series recorded from another patient.

6. Discussion and Limitations

The usage scenarios in Section 5 show that combining data from event sequences and time series can improve insights into the behavior of the underlying entities involved.

By combining visualization and interactive exploration of the data with algorithmic support for correlation detection, we provide users with the ability to find patterns that extend beyond what they could find by only looking at events or time series in isolation, while also giving them some statistical evidence supporting the correlation.

Furthermore, by extending the global results produced by the algorithm by Luo et al. [LLL*14] with a set of local measures and visualizations, users are enabled to detect local deviations in the global correlation pattern. By applying the prototype to various usage scenarios we discovered the need for several features, such as the *Split & Correlate* feature, time series re-sampling capabilities and sequence sorting options.

However, the usage scenarios also show that the global algorithm has limitations. It is fairly sensitive to variations in the input parameters. The algorithm works best with a larger number of events, and because it determines the presence of correlation by determining the difference between samples taken before and after the events with random samples taken from the entire time series, it may have difficulty detecting patterns where events are correlated with prolonged changes in the time series. In essence, the algorithm can be overwhelmed by an abundance of signal in the noise. This is however not necessarily a problem for ChronoCorrelator, as we do not use the algorithm as a black box. Instead we give users full control of the input parameters of the algorithm, thus being able to take advantage of their domain knowledge in the process, we do however aid the user by calculating reasonable default values for the parameters. Also, it is not necessary to rely on this particular algorithm, as it could be replaced by another.

The correlation is based on a user controlled window parameter k , however, there currently is no mechanism in place to extract the actual (likely) length of the pattern that is being sampled, thus the correlation may be linked to only a part of a larger pattern. Adding such a capability could aid in better understanding the patterns found. The default value for k is however chosen by extracting the first peak in the autocorrelation of the time series, as suggested by Luo et al. [LLL*14] and this seems to produce a reasonable estimate for the cases we looked at. We also provide the user with an averaged time-series of size $2 * k$ centered around the events, which can help in choosing a sensible value for k . Finally, the glyphs themselves encode k in the flag size, so users are enabled to compare the window size against the time-series, thus enabling them to visually inspect the correctness of this setting. For the second important parameter, the number of neighbors r , we chose $\ln(p)$, which is suggested as a good choice in literature [Sch86, LLL*14].

Scalability may be an issue when the amount of events and time series measurements increases, both in terms of visualization and in terms of the performance of the correlation algorithm. To mitigate the scaling issues in the visual representation it may be necessary to use some form of aggregation to keep the visualization both read-

able and responsive. For example, integrating a technique such as discussed in Krstajic et al. [KBK11], would help keep the event sequences readable even when dealing with large numbers of events. However, as mentioned in Section 4.4.1, users are still able to interpret the glyphs, even when there is overlap, since the local results, encoded in the flag color remains visible even when there is a large degree of overlap and the global results are doubly encoded in a pair of colored lines at the top of the chart (Figure 4(a)). On the algorithm side, the best way of improving performance is reducing the number of events used for correlation. Using sampling when event counts become too large could keep the algorithm usable in an interactive fashion, as long as users still have the option to apply the algorithm to the full set, with the knowledge that it could take longer. The current implementation has acceptable interactive performance for event selections of up to roughly 10,000 events and time-series with 300,000 steps. Any more, and the calculation time increases to beyond interactive levels, as also observed by Luo et al. [LLL*14], so a dual approach might be warranted, where interactivity is maintained when working with large sets by operating on a sample by default, while still allowing the system to run the algorithm on the entire set to get the most accurate results.

Our prototype currently has limited querying capabilities. Because our focus was on the correlation algorithm, we only implemented some simple querying capabilities into our system. To deliver a single environment, that can be broadly used, it should also implement querying capabilities such as those found in the work discussed in Section 2.

7. Conclusion

We present a novel visual analytics approach to gain insight into event data in combination with time series data.

We show that an approach combining these two disparate, but related, data types helps users to gain insight into the relation between patterns in time series data and the underlying events that are linked to these patterns. By providing users with the ability to correlate events with selected time series, via algorithmic support, they can determine if there is a relationship between the two, and do so in a manner which gives statistical support for their findings. By extending the global algorithm with local measures we enable users to also find deviations from the global pattern. Through our usage scenarios we have shown that our visual analytics approach aids in the analysis of event and time series data by providing the means to build and test hypotheses that are backed by statistical evidence. Since the suggested approach does not make any assumptions on the underlying data, it is generic and can be used in a wide range of domains, such as, but not limited to, system administration, electronic health record analysis or finance.

7.1. Future Work

Currently the application of the correlation algorithm and the time-series-as-filter operation are the primary means of interaction between event sequences and time series. Adding more operations could increase the applications further. One could imagine operations, such as the ability to filter time series based on event intervals, the generation of new time series based on event sequence data or conversely the extraction of new events from a time series, as possible candidates.

Another possible extension would be the addition of correlation capabilities between pairs of time series or pairs of event sequences, complementing the capabilities of the current algorithm that focuses on correlations between events and time series alone. The correlation algorithm currently only supports point events, not interval events, so perhaps a rule-based approach to simplify event sequences [CvW18] could enable users to link complex event subsequences to patterns within time series instead of the current approach where single events are used for correlations.

References

- [AMST11] AIGNER W., MIKSCH S., SCHUMANN H., TOMINSKI C.: *Visualization of Time-Oriented Data*. Springer Publishing Company, Incorporated, 2011. 2, 3
- [Ans73] ANSCOMBE F. J.: Graphs in statistical analysis. *The American Statistician* 27, 1 (1973), 17–21. 2
- [Bar08] BARTH W.: *Nagios: System and Network Monitoring*, 2nd ed. No Starch Press, San Francisco, CA, USA, 2008. 9
- [BC94] BERNDT D. J., CLIFFORD J.: Using dynamic time warping to find patterns in time series. In *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining* (1994), AAAIWS'94, AAAI Press, pp. 359–370. 6
- [BCZP11] BALAKRISHNAN R., CHEVALIER F., ZHAO J., PIETRIGA E.: Exploratory analysis of time-series with chronolenses. *IEEE Trans. Vis. Comput. Graphics* 17 (09 2011), 2422–2431. 3
- [BK09] BAMBENEK J., KLUS A.: *grep Pocket Reference: A Quick Pocket Reference for a Utility Every Unix User Needs*. Pocket Reference (O'Reilly), O'Reilly Media, 2009. 9
- [BM13] BREHMER M., MUNZNER T.: A multi-level typology of abstract visualization tasks. *IEEE Trans. Vis. Comput. Graphics* 19, 12 (Dec 2013), 2376–2385. 2
- [Bre13] BREWER C. A.: Colorbrewer2. <http://colorbrewer2.org/>, 2013. [Online; accessed 2018-08-09]. 8
- [BW08] BYRON L., WATTENBERG M.: Stacked graphs - geometry & aesthetics. *IEEE Trans. Vis. Comput. Graphics* 14, 6 (Nov 2008), 1245–1252. 4
- [CvW18] CAPPERS B. C. M., VAN WIJK J. J.: Exploring multivariate event sequences using rules, aggregations, and selections. *IEEE Trans. Vis. Comput. Graphics* 24, 1 (2018), 532–541. 3, 10, 12
- [CX18] CHEN Y., XU P., REN L.: Sequence synopsis: Optimize visual summary of temporal event data. *IEEE Trans. Vis. Comput. Graphics* 24, 1 (Jan. 2018), 45–55. 3
- [DRMGF14] DUNKL R., RINDERLE-MA S., GROSSMANN W., FRÖSCHL K. A.: Decision point analysis of time series data in process-aware information systems. In *CAISE Forum 2014* (June 2014), ceur-ws.org/Vol-1164/, pp. 33–40. 3
- [DSP*17] DU F., SHNEIDERMAN B., PLAISANT C., MALIK S., PERER A.: Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus. *IEEE Trans. Vis. Comput. Graphics* 23, 6 (June 2017), 1636–1649. 2, 5
- [FHR*14] FEDERICO P., HOFFMANN S., RIND A., AIGNER W., MIKSCH S.: Qualizon graphs: Space-efficient time-series visualization with qualitative abstractions. In *Proc. 2014 Int. Working Conf. on Advanced Visual Interfaces* (New York, NY, USA, 2014), AVI '14, ACM, pp. 273–280. 3
- [GAG*00] GOLDBERGER A. L., AMARAL L. A. N., GLASS L., HAUSDORFF J. M., IVANOV P. C., MARK R. G., MIETUS J. E., MOODY G. B., PENG C.-K., STANLEY H. E.: PhysioBank, PhysioToolkit, and PhysioNet. *Circulation* 101, 23 (2000), e215–e220. 9, 10
- [GAK*11] GSCHWANDTNER T., AIGNER W., KAISER K., MIKSCH S., SEYFANG A.: CareCruiser: Exploring and visualizing plans, events, and effects interactively. In *2011 IEEE Pacific Visualization Symposium* (March 2011), pp. 43–50. 3

- [HAP11] HRIPCSAK G., ALBERS D. J., PEROTTE A. J.: Exploiting time in electronic health record correlations. *JAMIA* 18, Supplement (2011), 109–115. 3
- [HKA09] HEER J., KONG N., AGRAWALA M.: Sizing the horizon: The effects of chart size and layering on the graphical perception of time series visualizations. In *Proc. SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2009), CHI '09, ACM, pp. 1303–1312. 3
- [HS04] HOCHHEISER H., SHNEIDERMAN B.: Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Information Visualization* 3, 1 (2004), 1–18. 3
- [IIC*13] ISENBERG T., ISENBERG P., CHEN J., SEDLMAIR M., MÄÜLLER T.: A systematic review on the practice of evaluating visualization. *IEEE Trans. Vis. Comput. Graphics* 19, 12 (Dec 2013), 2818–2827. 2
- [JPS*16] JOHNSON A. E. W., POLLARD T. J., SHEN L., LEHMAN L.-W. H., FENG M., GHASSEMI M., MOODY B., SZOLOVITS P., ANTHONY CELI L., MARK R. G.: MIMIC-III, a freely accessible critical care database. *Scientific Data* 3 (May 2016), 160035 EP –. Data Descriptor. 9, 10
- [KKB11] KRSTAJIC M., BERTINI E., KEIM D. A.: CloudLines: Compact display of event episodes in multiple time-series. *IEEE Trans. Vis. Comput. Graphics* 17, 12 (2011), 2432–2439. 3, 12
- [KTT17] KRUEGER R., TREMEL T., THOM D.: VESPa 2.0: Data-driven behavior models for visual analytics of movement sequences. In *2017 Int. Symposium on Big Data Visual Analytics (BDVA)* (Nov 2017), pp. 1–8. 3
- [KWS*15] KÖTHUR P., WITT C., SIPS M., MARWAN N., SCHINKEL S., DRANSCH D.: Visual analytics for correlation-based comparison of time series ensembles. *Comput. Graph. Forum* 34, 3 (June 2015), 411–420. 3
- [LLL*14] LUO C., LOU J.-G., LIN Q., FU Q., DING R., ZHANG D., WANG Z.: Correlating events with time series for incident diagnosis. In *Proc. 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (New York, NY, USA, 2014), KDD '14, ACM, pp. 1583–1592. 3, 5, 6, 7, 11, 12
- [LPK*16] LOORAK M. H., PERIN C., KAMAL N., HILL M., CARPENDALE S.: TimeSpan: Using visualization to explore temporal multi-dimensional data of stroke patients. *IEEE Trans. Vis. Comput. Graphics* 22, 1 (Jan 2016), 409–418. 3
- [MLL*13] MONROE M., LAN R., LEE H., PLAISANT C., SHNEIDERMAN B.: Temporal event sequence simplification. *IEEE Trans. Vis. Comput. Graphics* 19, 12 (Dec 2013), 2227–2236. 3
- [MS03] MULLER W., SCHUMANN H.: Visualization methods for time-dependent data - an overview. In *Proc. Winter Simulation Conference*. (Dec 2003), vol. 1, pp. 737–745. 3
- [PMR*96] PLAISANT C., MILASH B., ROSE A., WIDOFF S., SHNEIDERMAN B.: Lifelines: Visualizing personal histories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1996), CHI '96, ACM, pp. 221–227. 2
- [RAM*11] RIND A., AIGNER W., MIKSCH S., WILTNER S., POHL M., TURIC T., DREXLER F.: Visual exploration of time-oriented patient data for chronic diseases: Design study and evaluation. In *Proc. 7th Conf. on Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society: Information Quality in e-Health* (Berlin, Heidelberg, 2011), USAB'11, Springer-Verlag, pp. 301–320. 3
- [RWA*13] RIND A., WANG T. D., AIGNER W., MIKSCH S., WONGSUPHASAWAT K., PLAISANT C., SHNEIDERMAN B.: Interactive information visualization to explore and query electronic health records. *Foundations and Trends in Human-Computer Interaction* 5, 3 (Feb. 2013), 207–298. 3
- [SC00] SILVA S. F., CATARCI T.: Visualization of linear time-oriented data: a survey. In *Proc. First Int. Conf. on Web Information Systems Engineering* (2000), vol. 1, pp. 310–319. 3
- [Sch86] SCHILLING M. F.: Multivariate two-sample tests based on nearest neighbors. *Journal of the American Statistical Association* 81, 395 (1986), 799–806. 6, 11
- [SGBBT06] SHAHAR Y., GOREN-BAR D., BOAZ D., TAHAN G.: Distributed, intelligent, interactive visualization and exploration of time-oriented clinical data and their abstractions. *Artificial Intelligence in Medicine* 38, 2 (2006), 115 – 135. Temporal Representation and Reasoning in Medicine. 3
- [Shn96] SHNEIDERMAN B.: The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. IEEE Symp. on Visual Languages* (Washington, DC, USA, 1996), VL '96, IEEE Computer Society, pp. 336–. 2, 4
- [SMM12] SEDLMAIR M., MEYER M., MUNZNER T.: Design study methodology: Reflections from the trenches and the stacks. *IEEE Trans. Vis. Comput. Graphics* 18, 12 (Dec 2012), 2431–2440. 2
- [SMY*05] SAITO T., MIYAMURA H. N., YAMAMOTO M., SAITO H., HOSHIYA Y., KASEDA T.: Two-tone pseudo coloring: compact visualization for one-dimensional data. In *IEEE Symp. Inf. Vis., INFOVIS 2005*. (Oct 2005), pp. 173–180. 3
- [Tur13] TURNBULL J.: *The Logstash Book*. James Turnbull, 2013. 9
- [vWvS99] VAN WIJK J. J., VAN SELOW E. R.: Cluster and calendar based visualization of time series data. In *IEEE Symp. Inf. Vis., INFOVIS 1999, San Francisco, California, USA*. (Oct 24–29 1999), IEEE Computer Society, pp. 4–9. 3
- [WAM01] WEBER M., ALEXA M., MÜLLER W.: Visualizing time-series on spirals. In *Proc. IEEE Symp. on Information Visualization* (Washington, DC, USA, 2001), INFOVIS 2001, IEEE Computer Society, pp. 7–. 3
- [WHA07] WILLETT W., HEER J., AGRAWALA M.: Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Trans. Vis. Comput. Graphics* 13, 6 (Nov. 2007), 1129–1136. 4
- [Wik] WIKIPEDIA: Arctic Sun medical device; [online, accessed 2018-03-30]. 11
- [WPTMS12] WONGSUPHASAWAT K., PLAISANT C., TAIEB-MAIMON M., SHNEIDERMAN B.: Querying event sequences by exact match or similarity search: Design and empirical evaluation. *Interacting with Computers* 24, 2 (2012), 55 – 68. 3
- [WSJ*14] WANNER F., SCHRECK T., JENTNER W., SHARALIEVA L., KEIM D. A.: Relating interesting quantitative time series patterns with text events and text features. In *Visualization and Data Analysis, San Francisco, CA, USA, Feb 3–5, 2014* (2014), p. 90170G. 3
- [WWPS10] WANG T. D., WONGSUPHASAWAT K., PLAISANT C., SHNEIDERMAN B.: Visual information seeking in multiple electronic health records: Design recommendations and a process model. In *Proc. 1st ACM Int. Health Informatics Symposium* (New York, NY, USA, 2010), IHI '10, ACM, pp. 46–55. 2
- [XYF*17] XIAO S., YAN J., FARAJTABAR M., SONG L., YANG X., ZHA H.: Joint modeling of event sequence and time series with attentional twin recurrent neural networks. *CoRR abs/1703.08524* (2017). 3