Interactive Visual Transformation for Symbolic Representation of Time-oriented Data

Tim Lammarsch[1], Wolfgang Aigner[1], Alessio Bertone[2], Markus Bögl[1], Theresia Gschwandtner[1], Silvia Miksch[1], and Alexander Rind[1]

[1] Institute of Software Technology and Interactive Systems, Vienna University of Technology, Austria
{lammarsch,aigner,boegl,gschwandtner,miksch,rind}@ifs.tuwien.ac.at
[2] Institute of Cartography, Dresden University of Technology, Germany
alessio.bertone@tu-dresden.de

Abstract Data Mining on time-oriented data has many real-world applications, like optimizing shift plans for shops or hospitals, or analyzing traffic or climate. As those data are often very large and multi-variate, several methods for symbolic representation of time-series have been proposed. Some of them are statistically robust, have a lower-bound distance measure, and are easy to configure, but do not consider temporal structures and domain knowledge of users. Other approaches, proposed as basis for Apriori pattern finding and similar algorithms, are strongly configurable, but the parametrization is hard to perform, resulting in ad-hoc decisions. Our contribution combines the strengths of both approaches: an interactive visual interface that helps defining event classes by applying statistical computations and domain knowledge at the same time. We are not focused on a particular application domain, but intend to make our approach useful for any kind of time-oriented data.

Keywords: Data Mining, KDD, Data Simplification, Visual Analytics

1 Introduction

In Knowledge Discovery in Databases (KDD) [1], pattern-based Data Mining methods [2] play an important role when the user looks for specific events instead of creating a general model. Such kind of Data Mining is useful in many different application domains, like optimizing shift plans for shops or hospitals, or analyzing traffic or climate. For most methods, (e.g., [3]), an event is a tuple consisting of an integer representing the type of the event and the point in time when the event happened. A pattern (or event sequence) is a triple of an array of events with a starting and an ending time. Less formal definitions for pattern with similar meaning are given as “a local feature of data” [2] or “a local structure in the data” [4]. As time series are often very large and multi-variate, several methods for simplification have been developed (see Section 2). This simplification allows more efficient algorithms for Data Mining tasks like classification, clustering, and pattern discovery [4]. Various publications about pattern discovery [3,5,6,7,8,9,10,11] require data simplified that way. Lin et al. [12] point out that based on the generic framework for temporal Data Mining [13], the first step for all data mining methods is "generating a simplification that fits into memory while retaining the essential features of interest". If the simplified representation of the data is symbolic (also considered discrete), methods from text processing and bioinformatics can also be used for time-oriented data [14,12]. We call the data we are dealing with time-oriented data instead of time series, as we focus on the fact that this kind of data can be multi-variate and the variables can be correlated [15]. In datasets, time usually acts as a reference domain, with each time reference pointing to one or more data values in the form $R \rightarrow C$, with R being a set of references and C a set of data values [16]. Time as a reference domain also comprises a number of important structural aspects [15], some of which are already dealt with in Data Mining [10,11]. Domain experts dealing with time-oriented data often consider the structure of time an aspect of utmost importance [17,18]. In this publication, we deal with simplification along the various data domains, while keeping the information along the reference domain (e.g., time) as unchanged as possible. Thus, other methods can be used to mine this information.

Current methods for temporal pattern discovery either assume that the data dimension already is symbolic [5,6,3,7,8,9] or include a very simple means of user-based simplification, without dealing with user interface issues [10], or doing that in a very limited manner [11]. Methods that deal with simplification of data [19,20,14,12], on the other hand, tend to have a simplistic view on time as reference domain, considering only a list of concurrent data tuples. Therefore, performing such methods can "spillage" information contained in the time domain. Moreover, they are focused on automated Data Mining methods and do not support user interaction regarding special important event classes. To bridge this gap, we present a novel visual interface that allows users to define and modify event classes either manually by giving restrictions in the data dimensions, or using a statistical method similar to [12]. Our interface gives immediate visual feedback on the effects of these modifications. Contrary to existing methods, our data simplification works on multi-variate data. It is statistically robust, and has a lower-bounding distance measure. Moreover, it preserves the (temporal) reference dimension for possible mining at later stages. Above all, our interactive user interface includes human users into the process, providing access to their domain knowledge. By doing so, we transform the existing approaches with more limited uses into a flexible tool that allows direct interaction between humans and a computer-based algorithm. The data structures and the code for calculating the event classes are already implemented. All the user interface (UI) elements have been designed and the code for calculations and visualizations has been implemented, but the connection as shown in Section 3 has to be implemented as well. So while we are showing mockups for the UI elements, the visualizations inside and the presented values are correct. Our UI is not focused on one single application domain, yet we provide an example from shift plan analysis for shops in Section 4.

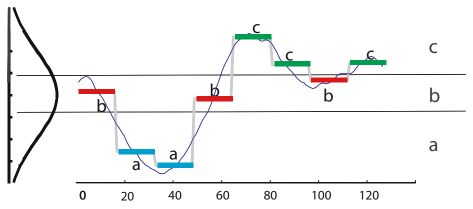


Figure 1. A time series is discretized in SAX by using statistically determined breakpoints to map the PAA approximation into event classes [12]

2 Related Work

In this section, we focus on three questions: (1) how is data simplification currently applied, (2) which algorithms that require simplified data could be improved by better simplifications methods, and (3) how are interactive visual interfaces already successfully applied in KDD.

Methods for Data Simplification. Dealing with data locally is a newer method than developing a global model [4,21]. The first such methods [20,19] therefore were compared to global methods, like Singular Value Decomposition, Discrete Fourier Transformation, and Discrete Wavelet Transformation. The Piecewise Aggregate Approximation (PAA) by Keogh et al. [20,19] can be considered as a kind of rasterization: time-oriented data with n points in time is converted to time-oriented data with w points in time. The authors call this "dimensionality reduction" because they treat time series as vectors in high-dimensional space. The amount of data is reduced along this dimension. However, the total number of variables in the dataset is unchanged. The only parameter that can be given by users for this transformation is the number of target references w. The PAA approximation can also be considered data simplification, but the resulting data is still value-based, while important information contained in the time reference might be hidden. PAA approximation is also part of SAX [14,12]. In SAX, a time series is first normalized to a mean value of 0 and a standard deviation of 1. The authors show that most time series have Gaussian distributed data values by analyzing various time series by means described by Larsen and Marx [22]. While this may not be true for all time series, we agree with Lin et al. [12] that this kind of time series is frequent enough that it deserves primary consideration. After applying the PAA, a resulting shortened time series is transformed to a symbolic (discrete) representation by allocating each of the w time windows to one of a number of event classes. The event classes are chosen on a statistical basis: Each class contains the same number of values (see Fig. 1). Lin et al. claim that the equiprobability of classes is important for several further analysis methods that can be performed after SAX, giving some examples [23,24]. For the same reason, they provide an Euclidean distance measure for their output. To preserve temporal information, we do not include the PAA-based rasterization in our work. The discretization step can be performed without rasterization. Another approach for data simplification is the rough set approach [25,26], used to include ontologies and/or domain knowledge. In the context of Data Mining, these comprise various methods to give event classes and their characteristics names and meaning.

Time-centered Algorithms for Pattern Finding. Lin et al. [14,12] focus on methods from text processing and bioinformatics as target of their work. When time is more than a simple counter of steps, however, different methods are capable of detecting more information. Most of those methods go back to the work by Agrawal et al. [5] who only consider patterns of events happening simultaneously (in their case, an event is the purchase of a product). They also consider time, but only as a method of separating different pattern candidates. However, they already perform planning towards further steps with more complex patterns that can overstretch time steps [6]. The events in those publications, like the purchase of a product, are inherently discrete data. Mannila et al. [3] focus further on sequences of events at different points in time. They also explicitly consider the time step at which one event occurs. They consider events to be determined by some external algorithm. Magnusson [7] is among the first to consider the time intervals between events, which is an important step to increase the consideration of time. His T-patterns are tree-shaped and therefore differ from the patterns in most other publications that are sequences. The work is focused on behavior analyses, so the events are found by some algorithm or even manually. Chen et al. [8] introduce the I-Apriori algorithm which extends the Apriori algorithm for pattern finding [4] by the consideration of intervals between events. Hu et al. [9] provide a similar approach where the focus is on patterns with events that do not need to be consecutive, as long as the time intervals are kept. Both approaches deal with events that either result from purchase and are inherently discrete, or assume that the data domain has been simplified to discrete events. Bertone et al. [10,11] provide a similar approach with user configurable time intervals that can have variable length and consider calendar aspects. They also mention the definition of events as multi-variate value combinations given by users. Therefore, Bertone et al. are also considering multi-variate data, but give limited explanation how users should deal with the complex event definition step. Summarizing those approaches, events are either considered as given, or the authors include a rather simple event finding process. Such simple event finding methods result in ad-hoc decisions that are not guaranteed to work, even if performed by very experienced users with domain knowledge. Therefore, we see a great need to find a better method of data simplification that results in more applicable event classes.

Samples for Application of Interactive Visual Interfaces in KDD. The Data Mining methods presented in the last subsection conduct pattern discovery as an automated task. Laxman et al. [4] discuss the advantages of this approach. However, interactive visual interfaces can greatly improve the applicability of the pattern finding process. Tominski [27] gives an overview how interactive visual interfaces are already used to find and display events according to the requirements and domain knowledge of users. He also provides an overview and formal model how events can be found with an interactive visual interface. Our interactive visual interface is developed with the guidelines from this publication in mind. The VISITORS system [28,29] has an interactive visual interface to explore and query patient data over time. It supports multi-variate data and combines both actual values and events. The events are determined by knowledge-based temporal abstraction methods. Similarly, Lifelines2 [30] is a system to explore events in time-oriented patient data. It provides specific interaction techniques such as alignment and temporal summaries. It expects data to be either of a categorical scale or simplified in advance. The simplified data resulting from application of our contribution are possible events that can be further analyzed by these systems. Activitree [31] provides a powerful interactive visual interface that users can employ to choose which patterns are important while performing algorithms, like those based on the Apriori approach [4]. As these examples show how interactive visual interfaces can improve various steps of Data Mining, we are heeding the call and present an interactive visual interface for the data simplification step of value discretization which has received insufficient attention in most existing work.

3 Interactive Visual Data Simplification

To present our interface, we need to (1) lay out the requirements, (2) describe the sample dataset we have used to design it, (3) show the interface itself, and (4) pay special attention to multi-variate data.

3.1 Requirements

We deal with multi-variate data as this kind of data is common among time-oriented data [15]. The number of variables can also be considered the number of dimensions in the multi-variate data space. The output should consist of a time reference and a single discrete data value. Each output element represents one single "event". Events can occur many times in any temporal order. However, the total number of different events is limited. All events that share the same characteristics and are grouped together by the user or by some kind of automated similarity computation are considered an "event class". In the current state of development, our user interface does not support giving events classes names or putting them into an ontology, however we are aware of the possible advantages of such expansions. We want users to be able to apply their domain knowledge and freely configure event classes. At the same time, we want to ensure equiprobable event classes, so that our approach can keep up with SAX [14,12] for Data Mining methods that require this kind of data discretization as input. Compared to SAX, we have to perform more complex statistical calculations in order to deal with multi-variate data. To combine the statistical and the manual approach, we need to provide three alternatives of defining the event borders between event classes:
Giving a target number of event classes. This results in classes with borders provided by the algorithm, similar to SAX, [14,12], but considerably more

complex when there is more than one data variable.
Completely free parametrization similar to the one proposed by Bertone et al. [10,11]. Event classes resulting from such kind of parametrization are not equiprobable. However, it is possible to have the set of event classes partitioned into groups, with each merged group being as probable as any other group.
Giving target value ranges that are supposed to be allocated into separate classes. The algorithm uses these ranges to calculate a number of classes (which cannot be chosen by the user in this case) with class borders at the given range borders. Below we show how these borders can be set.
All three alternatives should be performable at the same time. Making changes in one of those three alternatives has to be reflected immediately in the parameters for the other alternatives. At the same time, we need an interactive visualization for the data as well as the event classes.

3.2 Our Example Dataset

We used a synthetic dataset that is similar to the dataset analyzed by Bertone et al. [10,11]. The original dataset from that publication is not open to the public, therefore we use this alternative. The dataset covers data of a shop over one year. It is measured on a one-hour-raster a three data variables: *employees*, *customers*, and *turnover*. The dataset has the following characteristics:

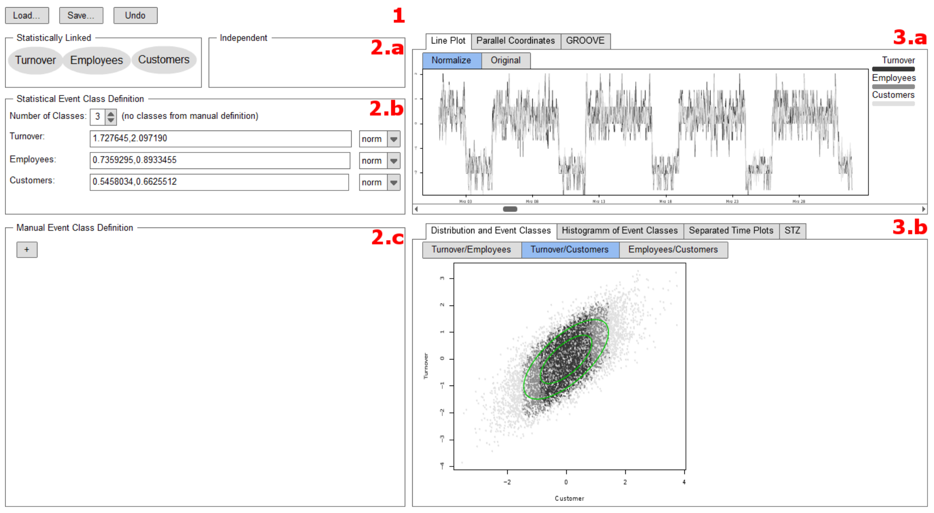
- All values are higher on weekdays and lower on weekends.
- When *customers* and *employees* are high at the same time, *turnover* is high.
- When *customers* or *employees* are high while the other of those two values is low, *turnover* is average.
- When *customers* and *employees* are low, *turnover* is also low.

We will present our user interface using this dataset as exemplary input-data.

3.3 The User Interface

The user interface consists of three main parts which in turn contain different parts themselves (see Fig. 2, the parts described below are marked by red text):

1. At the top, some basic commands are presented. We include an undo button that reverts the last action.
2. At the left, we have an area where users can edit the parameters of event classes: (a) At the top, the user can specify if given variables are statistically linked or independent from each other. The user does so by clicking on the variable, shifting it between the "statistically linked" and "independent" area. We will explain the meaning of this classification below. (b) In the middle, event classes can be configured using statistical properties. (c) At the bottom, manual event classes can be defined using specific values.
3. At the right, the raw data and events are visualized. The interactive visualization allows for modification by users. (a) At the top, the dataset is shown (a line plot in Fig. 2). (b) At the bottom, the event classes are shown (a scatter plot in Fig. 2).



Number of Classes: 3 (no classes from manual definition)

Turnover: 1.727645,2.097190 norm

Employees: 0.7359295,0.8933455 norm

Customers: 0.5458034,0.6625512 norm

Figure 2. Our user interface with (1) the basic commands (2.a/b) automated statistical calculation of three event classes, (3.a) a line plot, and (3.b) a value and class distribution scatter plot

Event classes are calculated in two stages. The first stage is based on statistical calculations. Fig. 1 already showed how equiprobable event classes are calculated in SAX. We extended this method to n value dimensions. Due to correlation effects, the classes cannot be related to low or high values for multiple dimensions. Instead, we use distance from the mean as criterion. Fig. 3.a) shows *customers* and *turnover* of our example dataset in a 2D scatter plot with three classes. Fig. 3.c) shows all variables of our example dataset (i.e., *customers*, *employees*, and *turnover* data) in a 3D scatter plot with three classes. Interpreting the 3D view is not always straightforward and easy (see below). Therefore, we project the 3D scatter plot visualization on a plane spanned by two variables that can be switched by the user. In our user interface, we support two variants of configuring equiprobable event classes: (1) A target number can be given by the user. The system calculates equiprobable classes as done by SAX, but it works for any number of variables instead of only one. (2) Class border values can be given for any variable. The data value space can be set in normalized form (with mean of zero and standard deviation of one), or the absolute values can be used. With respect to the correlation between variables, changing the borders along one variable changes the number of classes as well as the borders along the other variables. The borders are forming hyperellipsoids (ellipsoids for three variables, ellipses for two variables). For normalized values, the border values are given on the coordinate axis of one variable, with the other variables having values of zero. Otherwise, the respective mean values are used. For each value a hyperellipsoid representing a class border is formed, with its actual shape given by the correlation of the variables. Additional borders are added to the given

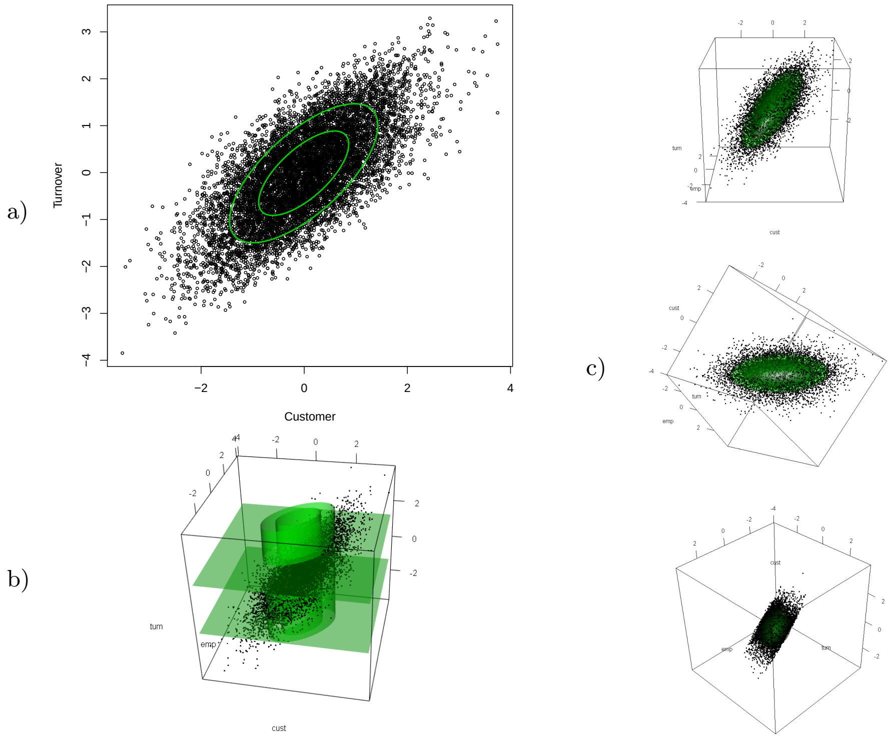
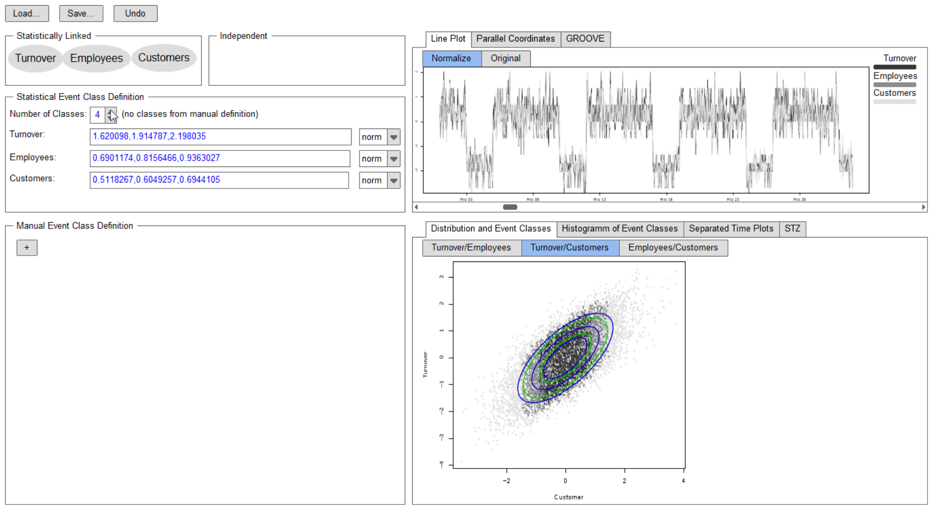


Figure 3. a) The *customers* and *turnover* part of or example data in a 2D scatterplot. The green ellipses are the class borders. b) A combination of manual and calculated event classes in a 3D-view. c) Our example data in a 3D scatterplot from several directions. The green ellipsoids are the class borders.

borders in order to keep the number of events per class equiprobable, so it is likely that the algorithm will add further border position values. The various settings can all be modified; the system automatically updates the dependent values accordingly.

When the user is about to change a value, this change is reflected in the whole user interface: In case the user modifies the parameters in the text boxes on the left, the text is entered in blue as long as *Enter* has not been pressed. The newly calculated class-borders are indicated by blue ellipses in addition to the current class-borders which are shown in green (see Fig. 4). Two situations result in the system showing “possible future” values: Hovering the mouse over an arrow (see Fig. 4) or editing a value in a text box. By pressing the *Enter* key, the change is permanent (unless Undo is used). By pressing *Esc*, the text box and all values are resetted.

Variables that are not included in statistical event class definition (because the user has shifted them to manual event class definition) are ignored at this



Number of Classes: 4 (no classes from manual definition)

Turnover: 1.620098,1.914787,2.198035 norm

Employees: 0.6901174,0.8156466,0.9363027 norm

Customers: 0.5118267,0.6049257,0.6944105 norm

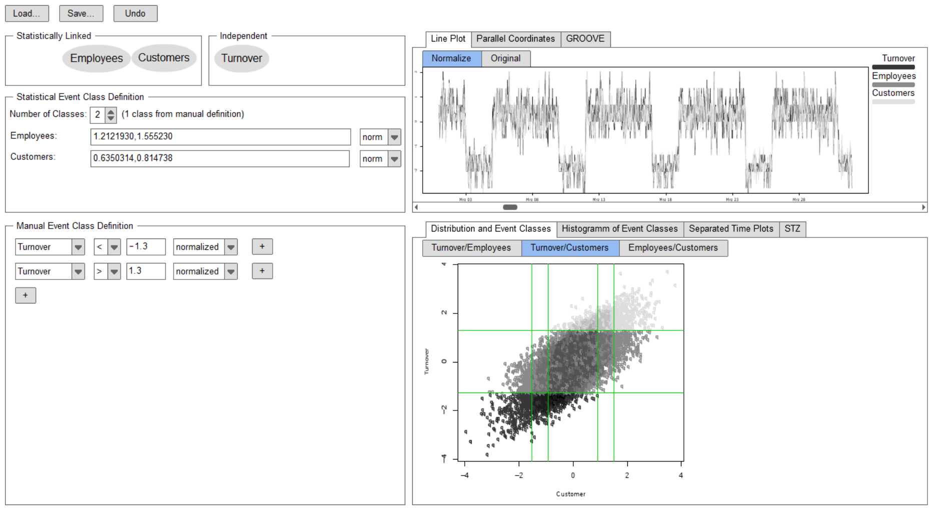
Figure 4. Our user interface while the user hovers the mouse over the button for increasing the number of classes. The values that would become actual by clicking are shown in blue, as are the class borders that would result from this user interaction.

stage. A data element is placed in a certain class no matter what data values are given for ignored variables. The number of data elements for the equiprobability calculation considers several data elements as identical, even if they differ along the ignored variables.

For variables that are set to be independent, manual event class definition can be performed. This is done in a second stage after statistical calculations. The manual event class definition works as described by Bertone et al. [10,11]. The vertical "+" button adds new event classes. The horizontal "+" button to the right of an existing event class adds new constraints for this class (see Fig. 5). All data elements that do not fit to the constraints are placed in an "other" event class. The data value space can be set to normalized values or as absolute values. The class borders resulting from this stage are added to the class borders from the first stage. If classes from the first stage are further intersected in the second stage, equiprobability for the resulting classes cannot be imparted. However, the combinations of classes that emerge from the second stage are still equiprobable. If no manual event class definition is performed, only the classes from the first stage exist. When changes are about to be made to manual event class definition, they are reflected at all parts of the user interface in blue color. This works the same way as described above for statistical event class definition.

Fig. 2, 4, 5, and 8 show how the dataset is visualized (upper right). We provide three different visualizations of the dataset:

Line Plot is a widely known method for visualizing time-oriented data [32] (see Fig. 2 and 4). Time is plotted over the horizontal axis, while the value of a variable determines the position at the vertical axis. Multiple variables are



Statistically Linked: Employees, Customers
Independent: Turnover

Statistical Event Class Definition
Number of Classes: 2 (1 class from manual definition)
Employees: 1.2121930,1.555230 norm
Customers: 0.6350314,0.814738 norm

Manual Event Class Definition
Turnover < −1.3 normalized
Turnover > 1.3 normalized

Figure 5. Our user interface can define event classes by a combination of statistical equiprobability present our user interface using this dataset as exemplary input-data and manual borders. The resulting class-borders are indicated by green lines in the projected scatter plot on the right.

shown in various shades of grey. Blue and green shades are not used because they are needed for other user interface elements. This visualization can be actived when users need to have a better view on data values.

Parallel Coordinates (see Fig. 6) by Inselberg and Dimsdale [33] do not show the flow of time. Instead, each of the axes represents one variable. The position is determined by the data values. The lines connecting the axes show the correlations between values. Therefore, this visualization can be activated when users need to investigate these correlations.

GROOVE (see Fig. 7) is a pixel-based visualization technique developed specifically for time-oriented data [34]. It is based on recursive patterns [35]. Both axes are used for time, while the data value is mapped to the color of pixels. Instead of adapting GROOVE to show multiple data variables at once, we use small multiples [36]. This visualization can be activated when users need to understand temporal structures.

Fig. 2, 4, 5, and 8 also show how event classes are visualized (lower right). Four kinds of visualization are possible:

Data Distribution and Event Classes are similar to the scatter plot [33] that has already been shown in Fig. 3.a). The points are data elements projected on a plane which can be selected by a tab bar over the visualization. The borders of event classes are shown as green lines. These borders are also projected. They represent the maximum circumference of the actual border frames. Elliptical borders result from statistical event class definition. Intersecting straight lines (as seen in Fig. 5) result either from manual event

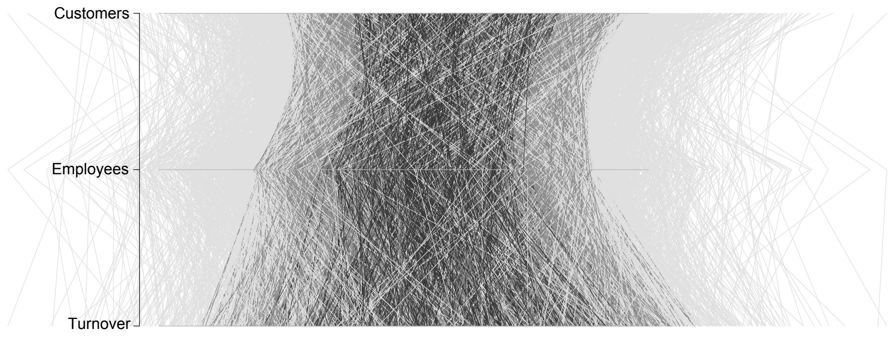


Figure 6. In the Parallel Coordinates [33] view, each data variable is represented on one axis. A data element is represented by a polyline, connecting the corresponding attribute values on the parallel axes. Depending on the available display area, the value range, and the number of variables, the axes may be vertical like in the reference, or horizontal like in this example.

class definition or are projected elliptical cylinders. Fig. 3.b) explains this fact and shows how the classes could be represented in three dimensions. When users are about to make changes to the event class configuration, the new class borders that would result are shown as blue lines in addition to the green lines showing the current borders. It is also possible to directly interact with the class borders: They can be grabbed with the mouse and dragged. This interaction results in the creation of new borders. This border is placed directly at the mouse position (assuming a value of zero for the variables that are not part of the projection plain). When the dragged border was resulting from statistical calculation, other borders are calculated in a way so that the classes remain equiprobable (see Fig. 8). For manually defined classes, the definition of the class is simply changed. When pressing the *Esc* key before releasing the mouse button, the dragging is canceled.

Histogram of Event Classes is a visualization that shows how many data elements fall in the various event classes [37]. If only statistical event class definition is used, the classes are all the same size (and the histogram is not needed). If manual event class definition is used, the histogram shows how the probability of these classes are distributed (we aim for equal probabilities). The current state is shown in black, blue lines show predicted changes when users are currently making changes to classes. In Fig. 9, we show an example histogram for the classes defined in Fig. 5. As some of the class borders are defined in an equiprobable way, and some are defined manually, summing up these bars in the right combination would result in three equal bars.

Separated Time Plots with class borders (see Fig. 10) are a mixture between the distribution, as the points are colored according to classes, and the line plots, as the horizontal axis shows time. They are focused on displaying the

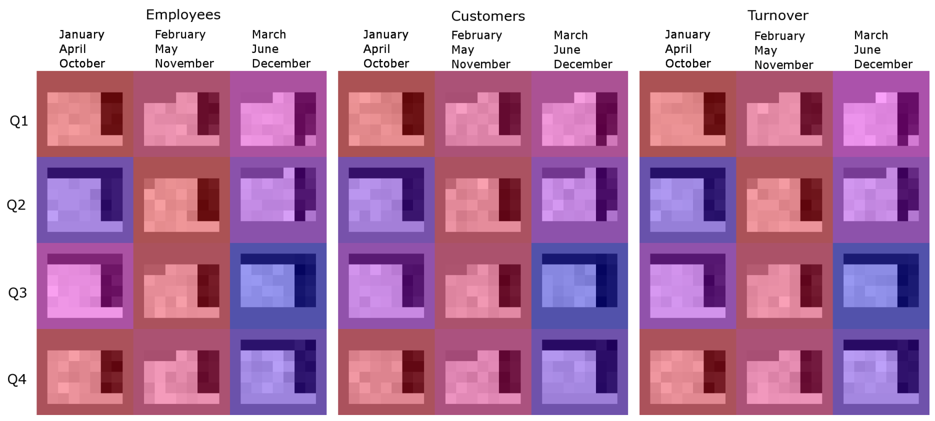


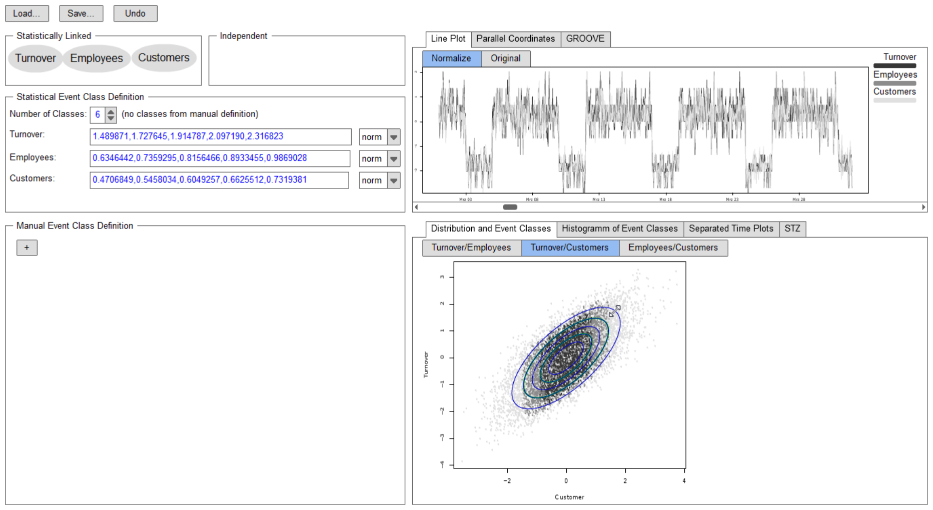
Figure 7. GROOVE visualizations [34] of the three variables. Each part with uniform hue represents one month, red means high average, blue low average. Each pixel represents one day, bright means high value, dark means low value.

event borders. Therefore, this visualization is similar to the visualization used to explain the event borders of SAX [12] (see Fig. 1).

STZ is a visualization method for qualitative abstractions and the associated quantitative time-oriented data. This interactive visualization technique, which is referred to as SemanticTimeZoom (STZ), is adopted from the Midgaard system [38,39]. While, in previous publications the qualitative abstractions were defined on basis of a single variable, typically based on severity ranges from domain knowledge, we can apply here the event data from our classification as common abstraction for all variables. Thus, the sequence of events is shown in combination with the development of numerical raw data along the time axis (see Fig. 11). STZ allows the user to dynamically switch between different levels of detail depending on vertical space. At a low detail level it shows only events, which are represented by colored boxes. Fig. 11 demonstrates the medium level, where raw data is shown as line plot and the area is colored by event class. At high detail level the raw data is again represented in a line plot with marks at the time points when event classes change.

3.4 Dealing with more than three Variables

Our user interface can deal with as many variables as the computer performance and the screen space permit. For visualizations that place the variables along the dimensions, a projection on two dimensions is necessary. While a projection on a pseudo-3D-view is possible, we focus on a full 2D projection because in a pseudo 3D-view (1) the occlusion is too severe (2) the actual position of points is too hard to grasp (3) user interactions are too complicated. An example for these problems can be perceived in Fig. 3.b). For a straight projection on two



Number of Classes: 6 (no classes from manual definition)

Turnover: 1.489871,1.727645,1.914787,2.097190,2.316823 norm

Employees: 0.6346442,0.7359295,0.8156466,0.8933455,0.9869028 norm

Customers: 0.4706849,0.5458034,0.6049257,0.6625512,0.7319381 norm

Figure 8. Our user interface while the user drags the class borders at the right with the mouse. The potential new class borders are shown in blue. On the left side, the numerical values that represent these borders are shown, releasing the mouse sets these values, which makes them become black.

dimensions, the values of the projected dimensions can be just ignored. This results in exact projections for the points, for class borders the maximum values have to be taken. So for three dimensions, the silhouette is shown. For four dimensions, when mentally picturing an animation over three dimensions, the ellipse is growing and shrinking again. Here, the projection shows the maximum circumference. For five or more dimensions, there is no real imaginable model, but the rule can still be applied. For values normalized to a mean of zero, the maximum circumference also exists where the other dimensions have values of zero, so dragging the class borders results in a clearly defined user interaction. Due to the fact that the class borders are showing the maximum circumference, several values that are outside one of the borders are projected inside the border. Here, the different gray levels of the data points can help to some degree.

4 Usage Scenario

We perform our usage scenario on the synthetic dataset as described in Section 3. A user wants to find important patterns in the data from her shop. This is a realistic scenario based on real-world application [10,17,18]. There are several important tasks to solve like "how many employees should be in the shop at a given time to maximize profit?", with profit being driven by *turnover*, but diminished by labor costs. By viewing the data in our user interface, it is likely that specific characteristics of the dataset given in Section 3 can already be seen. For example, the difference between weekdays and weekends becomes obvious

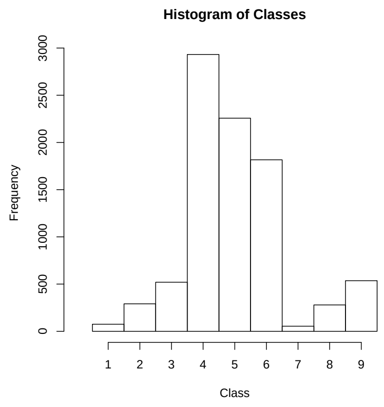


Figure 9. A histogram [37] showing the distribution of events classes in the resulting event data (in this example, the classes are taken from Fig. 5).

in the GROOVE visualizations (see Fig. 7). In our example, the GROOVE visualizations are very similar among the variables. This is due to the fact that the correlation is rather high. This is an advantage though, as a high correlation simplifies finding suitable event classes. In the example case, it means that the shift plan is already not bad. The user tries several numbers of classes (see Fig. 2, 4, and 8). In the end, she finds that for her business, it is important to separate the cases of high *turnover* and low *turnover* from average *turnover*. The reason is that she needs to find the definite causes for the high *turnover* (hoping to reproduce them), and for low *turnover* (hoping to prevent them). Sacrificing equiprobability regarding *turnover* might be necessary. Based on her domain-knowledge, she deems a high *turnover* of about one third higher than the standard deviation and a low *turnover* of about one third lower then the standard deviation as the most important cases. Therefore, she sets manual event classes for *turnover* (see Fig. 5). This results in the categorization of nine different types of events. These classes are represented by the different gray levels of points in the lower right of Fig. 5. Of course, this view is not the solution for the whole task, but a crucial step that enables the application of Data Mining methods relying on those classes. In those future steps, sample questions that could be answered are, which situations can lead to low turnover even if there are average customers, and how high turnover can be achieved even with average customers. The classes for these situations are provided.

5 Expert Review

We have placed our user interface under review by two different experts who both have been working in a university environment with close ties to industry. They received a preliminary version of the paper and information about the focus of the review by E-Mail and replied the same way. We analyzed the textual reports and

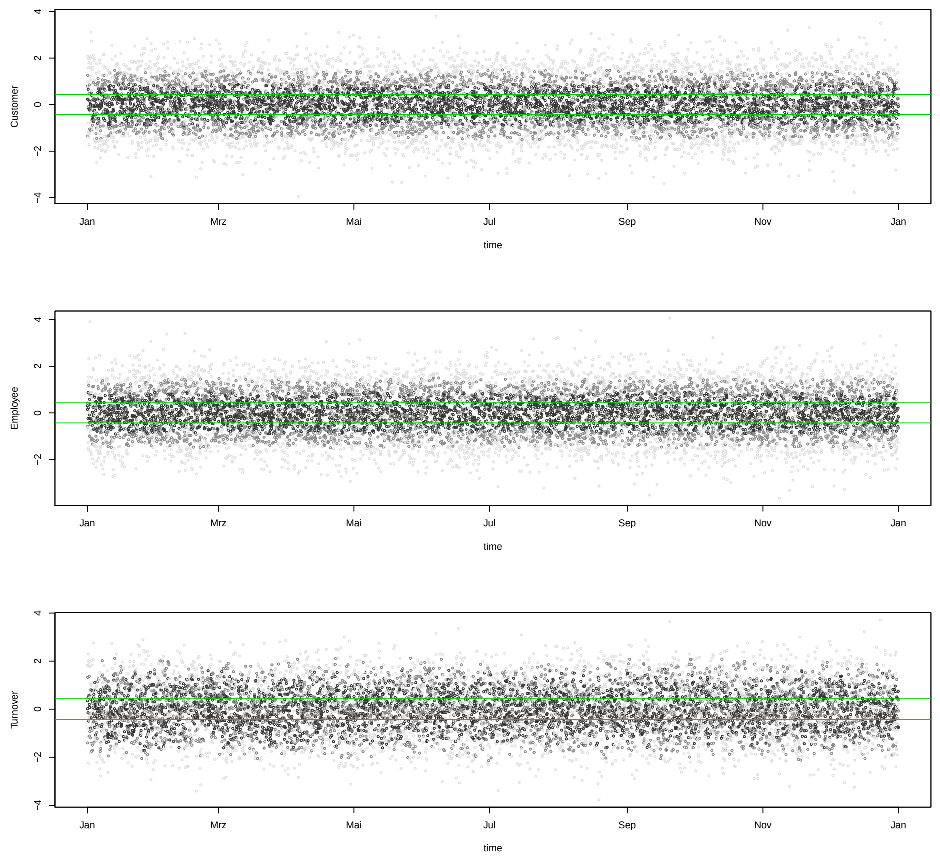


Figure 10. Separated Time Plots: the vertical axes gives the normalized data value, the horizontal axis shows time. The points are shown in three different gray levels: each level means affiliation with one specific event class. The projected class borders are shown in green. As more than one data dimension influences the borders, the affiliation is not fully conferred by these borders.

implemented as much as possible in the actual paper, while scheduling the rest for future work. As there were only two expert reviews, more complex analysis was not necessary. Goal of the reviews was to get insights about the general applicability of the approach and of possible user interaction pitfalls prior to further implementation work.

Review 1 focused on the applicability of our results for KDD. This review was done by an expert for Visual Data Mining, Temporal Data Mining, Information Visualization and Visual Analytics who took about two hours to get familiar with the topic and three hours for the evaluation:

The expert considers the method "definitely of high interest, as the selection of the most suitable event classes for a given task is usually not obvious and a bad choice may negatively influence the further steps of the analysis". The description is rated as

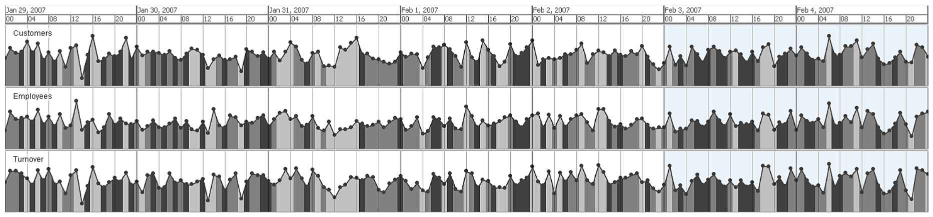


Figure 11. The SemanticTimeZoom (STZ) adopted from Midgaard [38,39] can interactively combine numerical raw data with qualitative events.

"clear and well supported by references" and the expert sees "a clear advantage for the user performing the first stages of his/her analysis, or refining not satisfactory results". Two problems are mentioned:

1. The expert is concerned about the number of attributes that can be accounted for in the visual interface and demands a better explanation for such situations. Furthermore, he proposes a Scatterplot Matrix instead of a single projection. *We improved our explanation on how to deal with more than three variables as well as our explanation of user interactions in the data distribution views. Furthermore, we included a discussion of the limitations of two dimensions as well as possible future developments in Section 6.*
2. The expert considers the choice of the event classes "already a part of the visual analysis". Therefore, in his opinion, our technique provides a bit more than only data simplification. *We pick up the topics of visual analysis, time, and steps of the KDD process in Section 6.*

In sum, it can be said that the expert was very positive about the applicability of the method for KDD while seeing some user interface issues that we intend to deal with in the future.

Review 2 focused on the usability of our user interactions. This review was done by an expert for Interactive Visual Interfaces, Visual Decision Support, Temporal Representations, HCI, Information Visualization and Visual Analytics who took one hour to get familiar with the topic and one hour for the evaluation. We provided a list of tasks. The expert solved these tasks by hypothetically applying the user interface. Based on the answers, we made some changes to the user interface. The labels of several visualizations had to be clarified. In total, the expert had no problem dealing with the interface, but the statistical background needed to fully employ the interface seems to be rather high. Still, we think that a decent class definition result can be achieved by domain experts. The result becomes better with more statistics skills and domain knowledge—which both are even harder requirements for other methods. After answering the questions, the reviewer gave an assessment of the usability:

1. Inconsistent: when I edit textboxes or drag ellipses, the new borders are shown in blue. For the +/− buttons, this already happens when I hover them with the mouse. It would be better if that happened when the buttons are pressed.
   There actually is a small inconsistency here. However, in the current state of development, textbox content is not actually changed before pressing Enter*. The same is true for the dragging of class borders: They are not changed till releasing the*

mouse. When changing the number of classes with the arrow button, the click already makes the change fixed, so there is no room for a blue preview after the hovering. We think that this has to be reevaluated on an actual implementation.

2. If I change the borders manually using some rules (with the lower left view), the new borders are shown immediately as green lines. These lines should be blue first and only become green after pressing *Enter*, removing the blue ones (see Fig. 5).
   This is a good input that will be included in the implementation of the interface.
3. GROOVE uses blue. This can result in misunderstandings as blue is the result of interactions here.
   At the same time, green is always used for class borders. To solve this issue, we will first evaluate whether GROOVE will be kept as part of the user interface. If it stays, we will evaluate whether it can be changed to an overlay of saturation and lightness. If this works, we will evaluate making GROOVE fully blue and interactions red.
4. Histogram: if the distributions resulting from an interaction are shown in blue again, the current condition should be green again. However, I am not sure if this visual metaphor should be used here, as no borders are shown, but the number of elements in the particular classes.
   We will analyze several color schemes using the working prototype.
5. There are too many possibilities to show the data, resulting in a high learning curve and confusion while switching between two different visualizations. Why are that many visualizations necessary? What is the advantage of GROOVE and simple line charts over separated time plots? Why the parallel coordinates? The correlations are hard to see in parallel coordinates and there are scatterplots anyway?
   A (smaller) choice of visualizations has to be made. However, we first have to test various visualizations and find out which ones are best, hoping that this is not too dependent on different datasets.
6. There is much overplotting in the line charts and the legend is hard to read. Why are the boxes not filled? They look like checkboxes.
   Currently, we cannot reduce the overplotting. We will look into solutions for line plot overplotting for the working prototype. The legend has already improved in the new version of the mockups shown in this paper.
7. The various time visualizations show different time spans. The time spans should be the same for all visualizations.
   The visualizations are intended to be zoomed and panned. As the mockups are static, they show states that we consider most illustrative.

6 Conclusion, Limitations, and Future Work

We have presented an interactive visual user interface that enables users to define event classes among a set of time-oriented data. These event classes can be (1) equiprobable and work similar to SAX [12], (2) freely configurable as described by Bertone et al. [10], or a combination of automated and user-based methods. The visual interface enables users to interactively develop suitable event classes for their needs. Our approach is in particular oriented to support multi-variate data. The data visualization area, and, to some degree, the event class visualization area, help users to gain an understanding of the data by means of interactive visualization. This understanding goes beyond the data variables—it also extends into understanding time-oriented effects. We consider this understanding advantageous for performing the event class configuration and, therefore, have included it into the user interface even though it surpasses the topic of data simplification. Furthermore, we can conceive a more extensive user interface that also includes other steps off KDD. In such an interface, the visualizations would be absolutely necessary. They would also help possible expansions of the user interface that let users provide names for event classes and their characteristics or even place them into an ontology. For many data variables, the two-dimensional view is a strong abstraction of the actual distribution. As a 2D-display and traditional user input devices make showing more dimensions hard to impossible, this limitation cannot be overcome directly. However, the application of linked views like a Scatterplot Matrix that show projections on different dimensions might help. Still, the number of variables that our approach can deal with in a meaningful way might be limited. To explore that actual limits of the method, a working prototype will be used. We are in the planning stage of such a prototype that will also include the proposed changes explained in Section 5. We can currently apply statistical classification under the assumption that the data has a Gaussian distribution and the Euclidian distance is an adequate measurement. Lin et al. show that this is sufficient for many important use cases [12] (see Section 2). On the other hand, Vlachos et al. [40] show that there are different tasks which require other classification and distance methods. Our approach can be extended to incorporate such methods. Furthermore, our approach is developed with time-oriented data in mind, as time is a complex data reference [15] and its complex structure is important for users [17,18]. However, other reference dimensions, like space, are also very important and show similar structures. As we focus on leaving time untouched, for other methods to deal with its structure, it is easy to extend our method to other reference domains. Many methods, like SAX, apply rasterization of the time dimension prior to further methods, because time-oriented data can be huge. However, rasterization can hide important information. Therefore, finding an optimal raster size is an important issue that can be dealt with by applying interactive visual interfaces. Others problems that are closely related to rasterization are dealing with missing values and outliers. Time-oriented data and the information contained in it is heavily influenced by certain aspects of social life and other phenomena [15]. To perform KDD on time-oriented data, special methods have been developed [10,11], but many more are needed to fully deal with all those aspects. Inventing such methods seems to be the next important step after successfully performing data simplification. Approaches like Activitree [31] show that interactive visual interfaces are an outstanding way to make such methods more accessible and effective.

Acknowledgements. This work was supported by the FWF Austrian Science Fund via the HypoVis project (#P22883).

References

1. Piateski, G., Frawley, W.: Knowledge discovery in databases. MIT press (1991)

2. Hand, D., Mannila, H., Smyth, P.: Principles of data mining. MIT press (2001)
3. Mannila, H., Toivonen, H., Inkeri Verkamo, A.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery **1**(3) (1997) 259–289
4. Laxman, S., Sastry, P.: A Survey of Temporal Data Mining. Sadhana **31**(2) (2006) 173–198
5. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. ACM SIGMOD Record **22**(2) (1993) 207–216
6. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. Advances in Database TechnologyEDBT'96 (1996) 1–17
7. Magnusson, M.: Discovering hidden time patterns in behavior: T-patterns and their detection. Behavior Research Methods **32**(1) (2000) 93–110
8. Chen, Y., Chiang, M., Ko, M.: Discovering time-interval sequential patterns in sequence databases. Expert Systems with Applications **25**(3) (2003) 343–354
9. Hu, Y., Huang, T., Yang, H., Chen, Y.: On mining multi-time-interval sequential patterns. Data & Knowledge Engineering **68**(10) (2009) 1112–1127
10. Bertone, A., Lammarsch, T., Turic, T., Aigner, W., Miksch, S., Gaertner, J.: MuTIny: a multi-time interval pattern discovery approach to preserve the temporal information in between. In: Proc. of ECDM10. (2010) 101–106
11. Bertone, A., Lammarsch, T., Turic, T., Aigner, W., Miksch, S.: Does Jason Bourne need Visual Analytics to catch the Jackal? In Kohlhammer, J., Keim, D., eds.: Proc. First International Symposium on Visual Analytics Science and Technology held in Europe (EuroVAST 2010), Eurographics (2010) 61–67
12. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. Data Mining and Knowledge Discovery **15**(2) (2007) 107–144
13. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. ACM SIGMOD Record **23**(2) (May 1994) 419–429
14. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proc. 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, ACM (2003) 2–11
15. Aigner, W., Miksch, S., Schumann, H., Tominski, C.: Visualization of Time-Oriented Data. Springer (2011)
16. Andrienko, N., Andrienko, G.: Exploratory analysis of spatial and temporal data: a systematic approach. Springer Verlag (2006)
17. Smuc, M., Mayr, E., Lammarsch, T., Bertone, A., Aigner, W., Risku, H., Miksch, S.: Visualizations at First Sight: Do Insights Require Traininig? In: Proc. of USAB08, Springer (2008) 261–280
18. Smuc, M., Mayr, E., Lammarsch, T., Aigner, W., Miksch, S., Gärtner, J.: To Score or Not to Score? Tripling Insights for Participatory Design. IEEE Computer Graphics and Applications **29**(3) (2009) 29–38
19. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowledge and information Systems **3**(3) (2001) 263–286
20. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Locally adaptive dimensionality reduction for indexing large time series databases. ACM SIGMOD Record **30**(2) (2001) 151–162
21. Yule, G.: On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character **226** (1927) 267–298

22. Marx, M., Larsen, R.: Introduction to mathematical statistics and its applications. Pearson/Prentice Hall (2006)
23. Apostolico, A., Bock, M., Lonardi, S.: Monotony of surprise and large-scale quest for unusual words. Journal of Computational Biology **10**(3-4) (2003) 283–311
24. Lonardi, S.: Global detectors of unusual words: design, implementation, and applications to pattern discovery in biosequences. PhD thesis, Purdue University (2001)
25. Nguyen, T.T., Skowron, A.: Rough set approach to domain knowledge approximation. In Wang, G., Liu, Q., Yao, Y., Skowron, A., eds.: Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, Berlin, Springer (2003) 221–228
26. Chen, H., Lv, S.: Study on ontology model based on rough set. In: Int. Symp. Intelligent Information Technology and Security Informatics, IEEE (2010) 105–108
27. Tominski, C.: Event-based concepts for user-driven visualization. Information Visualization **10**(1) (2011) 65–81
28. Klimov, D., Shahar, Y., Taieb-Maimon, M.: Intelligent selection and retrieval of multiple time-oriented records. Journal of Intelligent Information Systems **35**(2) (2010) 261–300
29. Klimov, D., Shahar, Y., Taieb-Maimon, M.: Intelligent visualization and exploration of time-oriented data of multiple patients. Artificial Intelligence in Medicine **49**(1) (2010) 11–31
30. Wang, T., Plaisant, C., Shneiderman, B., Spring, N., Roseman, D., Marchand, G., Mukherjee, V., Smith, M.: Temporal summaries: Supporting temporal categorical searching, aggregation and comparison. IEEE Trans. Visualization and Computer Graphics **15**(6) (2009) 1049–1056
31. Vrotsou, K., Johansson, J., Cooper, M.: Activitree: interactive visual exploration of sequences in event-based data using graph similarity. IEEE Trans. Visualization and Computer Graphics **15** (2009) 945–952
32. Funkhouser, H.: A note on a tenth century graph. Osiris (1936) 260–262
33. Inselberg, A., Dimsdale, B.: Parallel coordinates: A tool for visualizing multidimensional geometry. In: Proc. 1st Conf. Visualization '90, IEEE (1990) 361–378
34. Lammarsch, T., Aigner, W., Bertone, A., Gärtner, J., Mayr, E., Miksch, S., Smuc, M.: Hierarchical Temporal Patterns and Interactive Aggregated Views for Pixel-based Visualizations. In: Proc. of IV09, IEEE (2009) 44–49
35. Keim, D., Kriegel, H.P., Ankerst, M.: Recursive pattern: A technique for visualizing very large amounts of data. In: Proc. IEEE Visualization (Vis95). (1995) 279–286
36. Tufte, E.R.: The Visual Display of Quantitative Information. Graphics Press, Cheshire, CT (1983)
37. Pearson, K.: Contributions to the mathematical theory of evolution. ii. skew variation in homogeneous material. Philosophical Transactions of the Royal Society of London. A **186** (1895) 343–414
38. Bade, R., Schlechtweg, S., Miksch, S.: Connecting time-oriented data and information to a coherent interactive visualization. In: Proc. SIGCHI Conf. Human Factors in Computing Systems, Vienna, Austria, ACM (2004) 105–112
39. Aigner, W., Rind, A., Hoffmann, S.: Comparative evaluation of an interactive time-series visualization that combines quantitative data with qualitative abstractions. Computer Graphics Forum **31**(3) (2012) 995–1004
40. Vlachos, M., Kollios, G., Gunopulos, D.: Discovering similar multidimensional trajectories. In: Proc. 18th Int. Conf. Data Engineering, IEEE (2002) 673–684