StreamStory: Exploring Multivariate Time Series on Multiple Scales

Luka Stopar, Primoz Skraba, Marko Grobelnik, and Dunja Mladenic

Abstract—This paper presents an approach for the interactive visualization, exploration and interpretation of large multivariate time series. Interesting patterns in such datasets usually appear as periodic or recurrent behavior often caused by the interaction between variables. To identify such patterns, we summarize the data as conceptual states, modeling temporal dynamics as transitions between the states. This representation can visualize large datasets with potentially billions of examples. We extend the representation to multiple spatial granularities allowing the user to find patterns on multiple scales. The result is an interactive web-based tool called StreamStory. StreamStory couples the abstraction with several tools that map the abstractions back to domain-specific concepts using techniques from statistics and machine learning. It is aimed at users who are not experts in data analytics, minimizing the number of parameters to configure out-of-the-box. We use three real-world datasets to demonstrate how StreamStory can be used to perform three main visual analytics tasks: identify the main states of a complex system and map them back to data-specific concepts, find high-level and long-term periodic behavior and traverse the scales to identify which scales exhibit interesting phenomena. We find and interpret several known, as well as previously unknown patterns in these datasets.

Index Terms—Time series analysis, visualization systems and software, data and knowledge visualization, markov processes, multivariate visualization, data mining

1 Introduction

In this paper, we examine the problem of visualization for the analysis and exploration of large *multivariate time series*. Common solutions to visualizing time series such as common axis/plot or parallel views are unsatisfactory. They fail to highlight the interactions between variables and become difficult to interpret even for data of medium dimensionality. When dealing with large, complex datasets, users must zoom into interesting intervals, isolate informative variables and remove clutter manually. Each step causes a context switch, stealing their focus and hindering productivity.

One general pattern which we are often interested in for time-varying systems are cyclical phenomena. Examples include natural patterns such as seasons of the year and the daily cycle in traffic, electrical usage, etc. It need not be periodic, but may occur irregularly, such as increased traffic for holidays or specific weather events (e.g., especially high temperatures). As these patterns are most often not known a priori, identifying them gives us important insights into the underlying system and is the driving motivation behind our approach.

To help highlight these patterns, we represent data as states and transitions. We define states as abstractions which summarize all of the measurements in a particular region of the visualized space (i.e., *ambient space*). Transitions (with the associated states) summarize the path made by the time series through the *ambient space*. This hides the local variation present in any complex system reducing clutter and allowing the visualization to summarize large datasets while helping the user identify conceptual modes. For example, we describe weather in terms of states like "summer", "fall", "winter" and "spring" which are characterized by temperature, humidity and other attributes. In this kind of representation, cyclic patterns appear naturally as repeating sequences of states (i.e., literally cycles).

The spatial granularity (i.e., *scale*) of the states can highlight certain patterns while hiding others. Hence, our approach is multiscale, allowing new patterns to emerge from within coarser scale states or as the interaction of different parts of several states. Following our previous example, states representing yearly seasons can be refined to represent finer grain characteristics. For instance, a "summer" state could be split into a sunny and rainy state both representing hot weather. The analyst could then identify seasonal patterns, patterns which range across many seasons as well as compare patterns from different seasons.

Our approach is realized as an interactive web-based visualization tool called StreamStory (see interface in Fig. 1). StreamStory is designed to help analyze and interpret aligned measurements coming from complex time varying systems. The system helps the user search for recurrent patterns by representing the data as a diagram of states and transitions, where recurrent patterns visually stand out. To construct its representation, StreamStory uses and adapts several *machine learning* techniques. To represent multiple spatial granularities, we merge finer-scale states

- *L. Stopar and D. Mladenic are with the Jozef Stefan Institute, Jozef Stefan International Postgraduate School, Ljubljana 1000, Slovenia. E-mail: {luka.stopar, dunja.mladenic}@ijs.si.*
- *P. Skraba and M. Grobelnik are with the Jozef Stefan Institute, Ljubljana 1000, Slovenia. E-mail: {primoz.skraba, marko.grobelnik}@ijs.si.*

Manuscript received 26 July 2017; revised 19 Mar. 2018; accepted 6 Apr. 2018. Date of publication 18 Apr. 2018; date of current version 27 Feb. 2019. (Corresponding author: Luka Stopar.)
Recommended for acceptance by J. Heer.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2018.2825424

Fig. 1. On the move: A multi-scale summary of GPS coordinates collected over the course of three and a half years using a smartphone. StreamStory qualitatively summarizes the dataset using states and transitions. It shows a centralized structure with a large state in the middle, representing the researcher's home location (Slovenia), and many smaller satellite states representing trips to several locations in Europe (e.g., Germany, Portugal, Sweden, and Slovenia) as well as the US, China, and India. The selected state (with a blue border) is NYC (New York City, see right panel - Latitude: $40.89°$, Longitude: $-73.92°$). The bottom panel shows a timeline where NYC is highlighted. It shows several distinct short trips to NYC and one longer stay shown in the middle (June-Oct. 2014), corresponding to a summer internship.

into increasingly coarse representations creating a hierarchical partition of the *ambient space*. StreamStory enables the user to interactively navigate this multiscale structure with multiple visual cues establishing cross-scale relationships. The tool supports several tasks:

1) based on the data, identify the main states of the observed system and map these states to data-specific concepts,
2) find high-level and long-term periodic and recurrent behaviour in the dataset and
3) explore the dataset at multiple scales to identify at which scales interesting phenomena occur.

To address these tasks, we present a methodology which is based on the following pipeline: (1) the temporal information is disregarded and data represented as a point cloud, (2) regions of the *ambient space* are associated with conceptual states, (3) the temporal component is reintroduced and the dynamics are modeled as transitions between the states and (4) states and transitions are aggregated to obtain a multiscale representation.

Based on this methodology, we construct an interface which visualizes the representation at a single scale as a graph (see the central panel of Fig. 1) augmented by multiple connected views which map the visual elements back to the data and help interpret the individual states and patterns. For example, in Fig. 1, the bottom panel visualizes a historical overview of the data over the entire multiscale structure and the information on one of the states can be seen in the right panel. Our main research contributions are:

1) A novel methodology for multivariate time series visualization, multiscale exploration and interpretation. The methodology is based on *hierarchical Markov chains* and captures long term behavior of the data through a simplified model of the dynamics as a graph structure.
2) A novel visualization approach which allows the user to explore the structure of the dataset. Based on the methodology, the approach uses visual cues, several connected views and auxiliary tools to help the user to identify patterns and map the abstract representation back to meaningful domain concepts.
3) A fully interactive, web-based visualization tool, *StreamStory*, which integrates our methodology and is publicly available at http://streamstory.ijs.si. The tool remains useable and interactive even for large datasets as the high level of abstraction prevents clutter and the use of Markov chains as the underlying model makes it computationally efficient (and so highly scalable).

StreamStory is designed for individuals who may not have expertise in data analysis or may be dealing with data they are unfamiliar with. Its abstractions are designed to help such users identify and investigate recurrent behavior in temporal data as recurrence is recognizable as a *cycle* in the diagram. The system also provides automatically generated suggestions of possible interpretations. Examples of the types of behavior captured include periodic yearly weather patterns as well as intermittent or irregular recurrences. For instance, in Fig. 1 there is a cycle present

between the states labeled NYC (New York City) and CA (California), representing an individual's travel between the two locations in the United States.

2 Previous Work

Beyond simple graphs, many of the techniques for visualizing time series are domain specific and can only be used for special purposes. For an overview, we refer the reader to [1], [2]. In a more generic approach, Shneiderman et al. [3] visualize multiple time series implemented in a tool called *TimeSearcher*. *TimeSearcher* visualizes multiple time series in a list, visualizing their distribution by rendering them on the same plot in a separate view. It offers an interactive mechanism called *TimeBoxes* to highlight and filter the time series of interest. To support multivariate time series *TimeSearcher* uses a tabbed view. While comprehensive, this tabbed view can overwhelm non-expert users and make it difficult to identify recurrences (particularly in noisy data).

The patterns of interest often lie in the interaction between the variables—relationships lost when visualizing each variable in a separate view. In their work, Havre et al. [4] visualize thematic variations in a text document collection over time. The approach uses a linear time axis and encodes the values of the attributes as a dynamic stacked graph. Later, Byron and Wattenberg extend the approach using a different color coding and a novel layout algorithm. A drawback of these techniques is that they can only visualize variates with a positive domain and it can be difficult to identify recurrent behaviour.

In a different approach, Peng et al. [5] discretize and color code the range of each attribute and visualize the high dimensional time series in a *time series matrix* together with the median and variation. While the transitions are clearly shown, it is difficult to understand the global structure of the data from the visualization.

Wang et al. [6] visualize discrete event data across multiple records using a linear timeline in a tool called *LifeLines*. *LifeLines* support rank, align and filter functionality to zoom in on records of interest and offer a temporal summary as bar charts showing the number of occurrences of a particular event. Burch et al. [7] propose *Timeline Trees* to visualize discrete transactions of hierarchically organized elements on a discrete timeline coupled with thumbnail views which allow users to detect dependencies between elements. While the approaches described thus far are effective at visualizing trends, when drilling down into the structure or comparing attributes/time series, they fail to highlight recurrent patterns in the data. Furthermore, they often suffer from clutter when used for high dimensional datasets or larger datasets (in terms of number of samples).

In a different approach, Haroz et al. present *connected scatterplots* [8] which visualize two attributes in a scatterplot indicating the temporal direction using arrows. In a more generic approach Bach et al. present *TimeCurves* [9] which warp the time axis so that similar points appear close to each other. By disregarding all but similarity information *TimeCurves* visualizes only temporal dynamics between the (potentially many) individual points. In relatively small datasets, this ensures that all the structure is shown. In larger datasets, clutter makes it difficult to interpret the main states of the visualized system. By staying at the sample level and not using any form of aggregation, both *connected scatterplots* and *TimeCurves* become cluttered when used with datasets containing even a few thousand examples. In contrast, by aggregating examples into states StreamStory can visualize large datasets with potentially billions of examples, highlighting long term behavior through visual elements using the properties of the Markov chain and suggesting possible interpretations.

Recent years have seen the development of static graph-based techniques to visualize temporal data. Jänicke and Scheuermann use ϵ-machines to visualize a dynamic field in a static manner [10]. Similar to our approach, the field is modeled using states and transitions and visualized as a directed graph. In a similar approach, Gu and Wang [11] visualize temporal volumetric data using a graph-based visualization called *TransGraph*. Both approaches use coordinated views—one to visualize the graph and the other to show the original data in the ambient space. Visualizing the ambient space is not possible for higher-dimensional data and therefore we omit this view. Instead we employ MDS to layout the states to reflect their relative relationship in the ambient space and highlight different regions using color coding and automatic state labels. The main difference, however, comes at the conceptual level. While our approach defines states as compact, time-independent regions of the ambient space and uses transitions to model the trajectory of the data as it traverses the states, *TransGraph* and ϵ-machines define states as spatio-temporal regions, essentially modeling and visualizing change in the configuration of data at fixed locations.

By modeling the trajectory of the data through regions of the ambient space, Pylvänen et al. [12] describe an approach similar to our own. Applying *k-means* [13], they model the operational states of a multivariate time series as Voronoi cells in the ambient space, modeling temporal dynamics by counting transitions between the states. However, visualizing states only as a planar collection of points with no further information makes it difficult to interpret. Furthermore, using their technique, the scale must be fixed beforehand as the properties of *k-means* make it infeasible to reconstruct the visualization with a different number of states. StreamStory alleviates this issue by employing *hierarchical Markov chains* and using color coding to indicate hierarchical relationships allowing the user to interactively find appropriate scales to interpret the data.

There are several approaches for using Markov chains to model time series. In the univariate case, [14], [15], [16] this involves discretizing the range of the data before modeling transitions. For multivariate data Ching et al. [17], [18], [19] discretize the range of each attribute and model the dynamics using several linked Markov chains called a *multidimensional Markov chain*. Essentially, the approach models the trajectory of the data through a multidimensional grid. This can be prohibitive in high dimensional measurement space. Our approach partitions the ambient space into Voronoi cells and models the trajectory through the partitions. This captures additional structure, greatly reduces the number of parameters and simplifies the representation.

We visualize the states of the *Markov chain* by building upon ideas proposed in cluster visualization. We reuse several ideas such as projecting a representative of each cluster

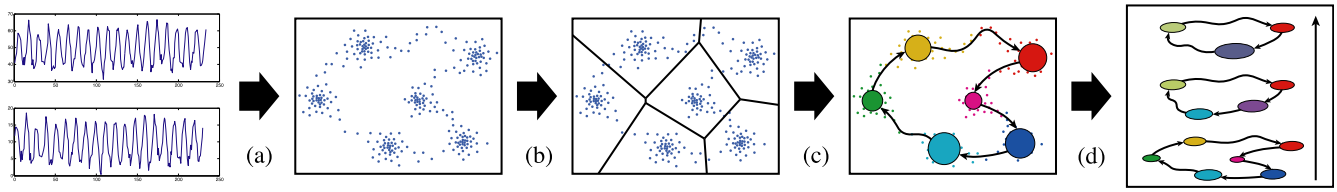

Fig. 2. Overview of the methodology. (a) The multivariate time series is first represented as a point cloud. As an example, we show two noisy approximately periodic signals mapped to points in 2D. (b) The states are constructed by partitioning the *ambient space* using a clustering algorithm. (c) Transitions are modeled by translating the partition into a Markov chain, with each state representing a partition cell. (d) Finally, the Markov chain model is simplified by iteratively aggregating states into a hierarchy, giving a multiscale view of the model.

onto a lower-dimensional space [20], [21], using multidimensional projection to approximate the structure of the clusters relative to each other, using visual attributes to describe the shape of a state [22] and using auxiliary widgets to help in the interpretation of the states [23], [24]. In addition, we generate qualitative descriptions of states in the form of automatic state labels and narrative descriptions shown in tooltips. Both help users quickly identify regions of interest as well as interpret the graph-based representation.

We create a *multiscale* representation to allow users to interactively select informative scales. The literature suggests other *multiscale* approaches. Woodring et al. [25] visualize temporal volumetric data in a *multiscale* manner by modeling each point in the volume as a time series, transforming it with wavelets and clustering by coefficients (i.e., frequency bands). The clusters are visualized in spreadsheet format with a separate view for details. The spreedsheet format can be overwhelming and is not appropriate for users who do not specialize in data analysis. Luboschik et al. [26] use the heterogeneity of subsequent scales as an indicator for noteworthy information and highlight where a drill-down to finer scales may be valuable. While we use a different indicator to determine interesting scales, this type of function can be incorporated into the system.

Finally, Auber et al. [27] visualize small world networks by hierarchically decomposing it into highly connected components and visualizing it in a two panel interface with an overview and details panel. The approach identifies and visualizes components of the graph, but it does not convey temporal information. Stolte et al. [28] describe a generic framework for visualizing multiscale datasets which replaces the linear zoom with a graph-based zoom path. In their framework, each node represents a specific visual and data abstraction. An analyst can independently zoom on either axis or independently change the level of detail in the data or visual abstractions. We use a linear zoom, as this provides the simplest interpretation for non-expert users.

3 Methodology

3.1 Overview

The proposed methodology enables interactive multiscale visualization, exploration and interpretation of large multivariate time series. Our goal is to identify recurrent behaviour in time series. This type of behaviour is characterized as paths in the *ambient space* which return to the same region several times. These regions represent the typical states of the time series. For instance, when describing traffic such states may include a "day" state and a "night" state. These high-level states can be decomposed onto finer-scale states like "morning", "afternoon", "evening" and "night" with the associated transitions following the daily cycle. Such characterizations provide a qualitative summary making it easier to describe the long-term behavior of the data.

Following this intuition, we propose a methodology which abstracts data into conceptual states and transitions. The methodology uses clustering to construct the states and represents temporal dynamics as transitions between the states using a Markov chain. The discretization of the system into states has several advantages over directly visualizing the data. First, long term patterns are represented as easily recognizable structures while preventing information overload/clutter making the tool useful for large datasets. Second, since the states and transitions are discrete, it can be simpler to interpret and map the abstract states back to the original measurements. This remains a difficult step and in Section 4 we discuss tools which automatically suggest possible interpretations of the representation. The main disadvantage of a discrete representation is that we lose some finer grain information about the data. Therefore, we provide tools which allow the user to investigate the individual states and transitions (Section 4). Further, in any discretization, the choice of scale affects which behaviours are emphasized, so we construct a hierarchy of representations allowing the user to find patterns at multiple scales.

The methodology is comprised of several steps (Fig. 2): (1) constructing a point cloud to represent the multivariate time series, (2) constructing the states by partitioning the *ambient space*, (3) modeling the transitions between the states and (4) aggregating the states and transitions into a hierarchy. We describe these steps in the following sections and conclude the section by presenting two techniques which help highlight recurrent behavior.

3.2 Constructing the Point Cloud

The first step of the methodology considers the data as a point cloud in multi-dimensional space by disregarding the temporal component (Fig. 2a). Point clouds are commonly used in machine learning, with each measurement (e.g., data associated to one timestamp) represented by a feature vector. This allows us to use a wide array of machine learning algorithms in the following steps of the methodology. We allow the user to select specific attributes which are not used in later steps of the methodology but can still be viewed in the final visualization. Superimposing attributes which are not used in building the representation is useful for studying correlations between the attributes.

3.3 Constructing the States

We next discretize the *ambient space* into non-overlapping regions which we associate with conceptual states (Fig. 2b).

Each state represents a typical configuration of the temporal signal and summarizes the structure around it. The vectors are partitioned based on a metric using a clustering[1] algorithm. For example, in Fig. 2, two noisy periodic signals are partitioned into six states. Any metric can be used, but we typically use euclidean distance, where by default, we normalize each dimension by the standard deviation. This helps mitigate incomparable scaling for the multivariate time series. We currently implement k-means and DP-means [29] to construct the partition, but any clustering technique can be used. The system is extensible and other clustering algorithms could easily be added. While other methods may be more appropriate for specific datasets, we chose these two specific algorithms due to the fact that they are computationally efficient and that they are known to be generally robust over a wide range of input. As our system is aimed at users who do not have expertise in machine learning, this step should use methods which are robust to parameter settings (to avoid parameter tuning).

Setting parameters, e.g., the number of centroids for k-means or the cluster radius for DP-means, must be done using domain knowledge. Since our method is multiscale, the parameter represents the finest computed scale and so the maximum number of states computed. For large datasets, the number of states is often the bottleneck in computation and so is limited. We have also found that a large number of states causes the visualization to become too cluttered at the finest scales. While the choice of parameter can and does effect the constructed states, we have found that interesting patterns are robust to the specific choice of parameters, e.g., in the experiments, we did not need to perform parameter tuning.

3.4 Modeling Transitions

Once the states are constructed, we model the dynamics as transitions in the lowest-scale Markov chain (Fig. 2c). Each conceptual state is associated with a state of a Markov chain. We refer to these as *initial states*. They form the lowest-scale representation of the data and are the basis for further computation.

Transitions between these states model the trajectory of the temporal signal as it traverses the corresponding discrete regions of the *ambient space* (i.e., states). We assume that the sequence of states is generated by a *continuous-time random process*. Besides the sequence itself, such a process is characterized by the time the process spends in a state upon arrival (i.e., *stay time*). We assume exponentially distributed *stay times* and model the process as a *continuous-time Markov chain* [30], extracting the transition probabilities from the *jump chain*. The *jump chain* models the sequence of jumps and their associated probabilities, discarding the *stay times* [30]. This design choice has two main consequences. First, by using continuous time Markov chains, we do not require input data to be equally sampled in time, eliminating the need for an additional preprocessing step. Furthermore, time series are often sampled from continuous processes making the assumption of continuous time more natural. Second, by using the *jump chain* to visualize dynamics, we reduce clutter by removing transitions of the form $i \to i$ from the visualization.

1. We refer to clusters as conceptual states.

A continuous time Markov chain is completely defined by a transition rate matrix denoted by Q. Its off-diagonal elements q_{ij} represent the rates of transitioning from state i to state j (in $\#jumps/timeunit$). The rows of Q sum to zero, therefore the (negative) diagonal elements $q_i = -q_{ii}$ represent the rate of leaving state i. To learn the parameters from the data, we use a maximum likelihood estimator, which in the case of Markov chains is simply given by Equation (1) for each non-diagonal entry of the matrix.

$$\tilde{q}_{ij} = \frac{N_{ij}}{t_i}, \tag{1}$$

where N_{ij} represents the number of transitions $i \to j$ observed in the training set, while t_i represents the total time spent in state i. The diagonal elements are then extracted as the negative sum of the rows:

$$\tilde{q}_{ii} = -\sum_{k=1}^{n} \tilde{q}_{ik}.$$

The key advantage of using Markov chains is that while their representation is compact, they still allow us to compute several long-term properties efficiently, making the system scalable and responsive. They provide a summary of longer term behaviour which can convey the corresponding patterns effectively. It would be possible to use more complex models, e.g., more complex transition and stay probabilities, but it would significantly increase the computational cost of constructing and interacting with the models. The benefit, however, would be negligible, since the system provides several tools which allow the user to drill down to investigate short-term behavior (see Section 4).

3.5 Constructing the Hierarchy

The final step constructs the multiscale representation by aggregating states and transitions into a hierarchy. We construct a hierarchy of Markov chains (Fig. 2d) and associate each level with a scale used in the final visualization. This allows the user to interactively change the visualized scale and find interesting patterns without the need to reconstruct the representation. The construction is a two step process: (i) constructing the topology (i.e., deciding which states are merged) and (ii) aggregating the states of the lowest-scale Markov chain to obtain higher scale representations.

StreamStory implements two approaches to construct the hierarchical topology. The first is a bottom-up distance-based approach which uses *mean linkage* agglomerative clustering (UPGMA) [31] to merge pairs of states with minimal distance. As in the clustering step, we use the euclidean metric to measure the pairwise distances. The intuition is that states that lie closer in euclidean space are more similar and should be merged before more distant states. The second approach is a top-down transition-based approach. The method used is a modification of the algorithm proposed in [32] to continuous-time Markov chains. It begins by treating the whole Markov chain as a single high-level super-state. It then iteratively performs binary splitting on a state of the Markov chain. State splitting is performed by considering the Markov sub-chain induced by the sub-states of the currently chosen state. We then consider the *min-cut* problem

on the weighted graph induced by the sub-chain, taking transition intensities as weights. To find an approximate solution, we use a standard technique from spectral graph theory, i.e., considering the sign of the eigenvector corresponding to the second largest eigenvalue of the symmetrized transition rate matrix [33], [34], [35]. The splitting continues until all such sub-chains contain only a single state and no split can be made—that is, the procedure reaches the original Markov chain.

Once the hierarchical relationships are known, we compute higher-scale Markov chains by averaging transition rates. As input to step (ii), we are given a set of merge points (or scales) s_k and, for each scale s_k a set of aggregation rules, describing which *initial states* should be merged. These are encoded in the matrix P_k, where $(P_k)_{ij} = 1$ if and only if the *initial state* i should be aggregated into state j. To represent the Markov chain on scale s_k, we calculate its transition rate matrix Q_k using the following formula:

$$Q_k = (P_k^T \Pi P_k)^{-1} P_k^T \Pi Q P_k, \tag{2}$$

where Q represents the transition rate matrix between the *initial states*, P_k is the aggregation matrix and $\Pi = diag(\pi)$ is a matrix with the stationary distribution [30] of Q on the diagonal. The formula calculates the transition rate between two high-scale states as the weighted average of transition rates between the associated *initial states*. To preserve the *ergodic* properties, the entries of the stationary distribution of the source states are used as weights.

Finally, we reduce the number of scales in the hierarchy to highlight changes in the structure. The goal is to choose scales which are qualitatively different. We choose the scales by associating each scale with a vector of eigenvalues of the transition matrix at that scale. Then we perform k-means clustering on the vectors with the representative closest to the centroid of each cluster taken as one of the chosen scales. The number of clusters is chosen as the minimum of half of the number of initial states and 10. This is to ensure that the resulting visualization is not overly cluttered and provides useful information. The adaptive choice of scales is important to avoid imposing an additional parameter on the user, while ensuring the structure present at different scales is displayed.[2] We then post-process the hierarchical model for visualization - computing the layout and enriching the structure with other attributes such as automatic labels of the states.

3.6 Highlighting Recurrent Behavior

We provide the user with two options which help highlight and reveal recurrent behavior. The options influence the structure of the model at construction time by altering the feature vectors used to identify the *initial states* in the clustering step.

Using Dynamical Context. Based on *time-delay embedding* [36] the first approach adds derivative information to the feature vectors. It does so by appending values from the previous time steps to each feature vector in the dataset, effectively doubling the dimension of the signal. For instance, for a time-delay of 1,

$$x(t) \mapsto \begin{pmatrix} x(t) \\ x(t-1) \end{pmatrix},$$

it appends the previous time step to the current time step. This helps reduce ambiguities in recurrences by effectively considering the derivative of the time series. The intuition is that for two points to be similar, their paths (for some number of time steps) must also be similar. Following the weather example in Section 1, the feature vectors are extended using values of the previous month, separating spring and autumn points. This approach was originally developed for recovering the dynamics from chaotic systems [36].

Static Context Based on Time. The second approach extends the feature vectors with qualitative temporal information extracted from the timestamps of individual measurements. The feature vector is extended with a categorical feature which describes when in time the measurement occurred on the specified temporal granularity. The categorical feature is represented by a binary vector,[3] hence the dimension increases by the number of possible categorical values. It requires the user to select a temporal granularity when constructing the model. For instance, when the time unit selected is "day," the feature vector will be extended with a categorical feature indicating which day of the week the timestamp represents, e.g., Monday, Tuesday, etc. increasing the dimension by 7. Other options include: (a) *second* with six categories representing ten second intervals in a minute, (b) *minute* analogous to second, (c) *hour* with 24 categories representing each hour in a day, (d) *day* and (e) *month* with 12 categories representing the months of a year.

The approach emphasizes periodicities. The additional categorical feature naturally brings points which occur in that category closer together, e.g., with the *day* feature, all points on Monday are brought closer, while points occurring on different days are moved further away. This feature is useful if the domain expert would like to identify patterns at a specific temporal granularity. Each feature represents a granularity and highlights a typical period, e.g., the *hour*, *day*, and *month* features highlight the daily, weekly and yearly cycles respectively. Additionally, it allows the user to identify which states occur during a specific time period (e.g., which states occur most often on Monday).

4 Visual Representation

We begin this section with the specific requirements which guide the design of our visualization, followed by a description of the visual elements and how they satisfy the given requirements. The visualization should provide a high level conceptual summary of the data which prevents visual clutter when dealing with large datasets. Temporal patterns, especially periodic and recurrent behavior, should be apparent. Users should be able to zoom into specific parts of the summary to find these patterns. Finally and most importantly, the visualization must help the user interpret patterns by linking elements of the summary back to the data.

2. For an advanced user, the automatic choice could be overridden to a user-specified choice.

3. A binary vector for a data point is 1 for its categorical value and 0 for all other categorical values.

Fig. 3. The visualization consists of 3 panels. The central panel visualizes the data through a Markov chain at the current scale. The bottom panel shows a cross-scale historical overview of the dataset using an *icicle plot*. States on the two panels are associated by color. The right panel visualizes properties of the currently selected state (blue border).

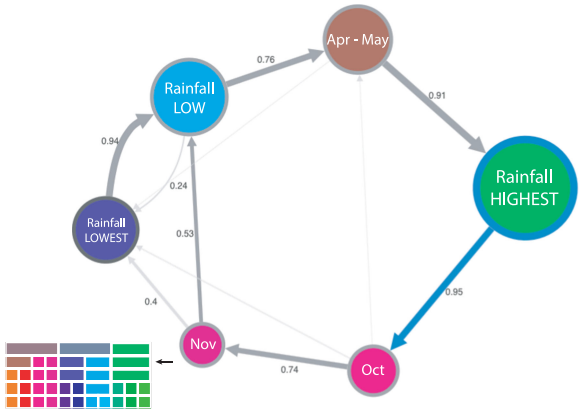

Fig. 4. Visual attributes encode different properties of the states: (a) The area of each circle encodes the proportion of time the system spent in the state in the training dataset. (b) The position reflects the states' position in the ambient space relative to other states. (c) The color encodes the position of the state in the hierarchical topology of the visualized model. The hierarchy is shown by the colored square on the lower left, with the arrow indicating the current scale. (d) The label highlights attribute and time-based properties which are typical for the state when compared to other states. (e) A bold blue border highlights the selected state.

The hierarchical Markov chain provides the multiscale summary which simplifies the data into discrete states, avoiding clutter while still capturing the dynamics. It is visualized using a graph-based representation augmented by several visual cues to help the user understand the global structure of the data. The user can interactively switch the scale at which the structure is visualized. Perhaps the most important aspect of the visualization are the auxiliary tools which provide context for the structure. They allow the user to drill-down into the individual states and transitions, visualize their properties, and automatically suggest possible interpretations using data-driven approaches. The result is a more intuitive visual representation.

4.1 User Interface

The user interface consists of three connected panels shown in Fig. 3: (i) the central panel visualizes the data as a static graph based on the Markov chain at the current scale, (ii) the bottom panel shows a cross-scale historical overview of the dataset and (iii) the right panel shows details about selected states.

The central panel is the main panel in our visualization and guides the exploration of the dataset. It visualizes the long-term behavior of the data using the Markov chain associated with the current scale using circles and arrows as states and transitions respectively. Interacting with the panel changes the information displayed in the other two panels which act as auxiliary views and provide detailed information. This avoids overwhelming the user with parallel views and the occlusion problems associated with a three dimensional encoding. Users can traverse scales by using the scroll function on the mouse or by sliding a scrollbar on the left of the panel.

We visualize states as circles and use five visual attributes to encode their properties (see Fig. 4). These are: (a) radius, (b) position, (c) color, (d) label and (e) border. The choice is designed to provide as complete picture as possible without overwhelming the user. The use of these attributes in the representation of Markov chains is fairly standard. Our main contribution is to assign these values in a way which respects the multiscale hierarchy. This helps the user understand how the structure at each scale relates to finer and coarser scales.

The radius encodes the proportion of time the modeled system spent in the associated state in the training dataset. It is set so that the area of the circle is linearly proportional to the states' entry in the stationary distribution of the Markov chain [30]. Position reflects the states' relative position in the ambient space. It is computed using multidimensional scaling (MDS) to preserve pairwise distances between the centroids of the original partitions. Additionally, we follow with a cross-scale repulsive step to avoid state overlap. The initial layout is constructed automatically, but users can rearrange the states to fit their understanding of the data (see Section 4.2 for details). Color encodes the states' positions in the hierarchical topology. We use saturation to encode the states' scale (i.e., the finest scale on which the state appears) and hue to encode the distance between states on the same scale (see Section 4.3 for details). Throughout the paper, we use a color square (Fig. 4) to show assignment of colors.

To help interpretation, StreamStory automatically assigns labels to states using the method presented in Section 4.4. A label highlights the properties which are typical for the state when compared to other states. This includes the configuration of attributes and the typical times when the state occurs. We use a tooltip as an extension for the label - providing a textual description of the state. The user can interactively change the label to reflect their own interpretation of the state. Finally, when a state is selected, we highlight it using a blue border and all the transitions leaving the selected state using the same color.

Temporal dynamics are modeled as transitions between states and visualized as arrows between the associated circles. We encode the likelihood of a transition as the thickness of the corresponding arrow. Instead of the traditional transition probabilities, we show the transitions of the *jump chain* associated with the Markov chain [30]. This reduces clutter by eliminating transitions from a state back to itself.

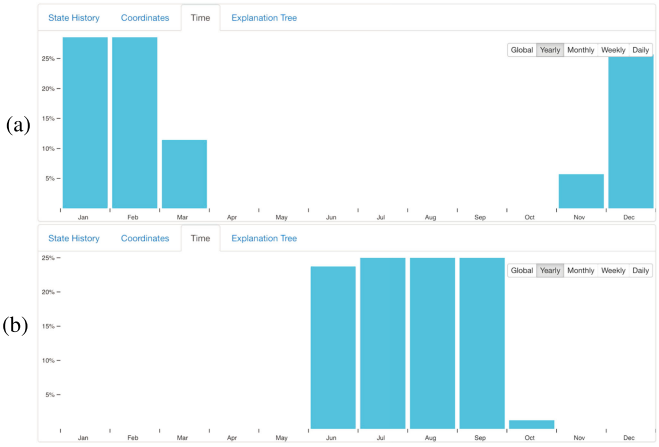

Fig. 5. The *Time* histogram explicitly shows when a state or transition occurs. It supports several temporal granularities (e.g., a daily histogram showing the hours of the day) which can be used to identify periodic behavior. The example above shows the months when the *Rainfall LOWEST* (dark purple) and the *Rainfall HIGHEST* (green) states of Fig. 9 occur. We see the former typically occurs in the winter (a) and the latter in the summer (a).

To further reduce clutter, we group transitions into high, medium and low probability groups. We highlight arrows in the high probability group with a darker color and blur arrows in the low probability group by using a dotted line and omitting a label (see Fig. 4). By default we regard transitions with probability over 0.4 as high and those with probability below 0.2 as low probability transitions. To further highlight dominant transitions, we allow the user to interactively reduce their number using a horizontal scrollbar at the bottom of the main panel.

The bottom panel has several views. The view on the bottom panel is selected via tabs as can be seen in Figs. 1 and 3. The first view is an *icicle plot* which visualizes the dataset as a cross-scale historical overview. It shows the sequence of states which occur in the data for all scales simultaneously (Fig. 3). It can be used to isolate and investigate known historic events and identify the scales where events are visible. For instance, suppose the user is interested in a particularly cold winter. They can use the *icicle plot* to zoom into the desired year and identify the (sequence of) states which occurred in the winter months. For each row, the associated scale is labeled on the left-hand side. The label of the current scale is highlighted. Color coding is shared with the central panel, allowing users to quickly identify states. The *icicle plot* is also interactively integrated with the central panel. This allows users to select a state by clicking it on either representation. When selected, the state is highlighted in both representations. When selected through the *icicle plot*, the appropriate scale is automatically shown in the central panel.

A second time-based view on the bottom panel can help interpret temporal patterns by showing when (e.g., which day of the week) a state or transition typically occurs. The main panel allows users to select either states or transitions. When one is selected, the time-based view on the bottom panel shows when it occurred at different temporal granularities using histograms, mapping states and transitions to temporal concepts. The available granularities are: daily, weekly, monthly and yearly. Fig. 5 shows the bottom panel with the yearly histogram of two states from Fig. 3.

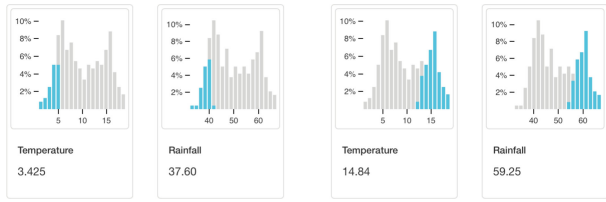

Fig. 6. The *Attribute* histograms show the distribution of attributes in the selected state with the global distribution of the attribute as context in the background. Here we show the distribution of two attributes in the purple *Rainfall LOWEST* and green *Rainfall HIGHEST* states from Fig. 3 respectively. The histograms indicate a lower distribution of temperature and rainfall in the *Rainfall LOWEST* state (a) and a high distribution of temperature and rainfall in the *Rainfall HIGHEST* state (b) suggesting a correlation between the two attributes.

The histograms show that one state usually occurs in the winter months (Fig. 5a) while the other occurs in the summer months (Fig. 5b). The bottom panel offers two other views for the selected state: parallel coordinates and an explanation/decision tree. Parallel coordinates [37] are a standard tool while the decision tree can be used to explain high dimensional clusters [38]. These tools associate states to data specific concepts. They are aimed primarily at specialists, so we defer the views and further explanation to the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2018.2825424.

The right panel visualizes properties of the selected state. This includes the mean and distribution of all the attributes (Fig. 6). For context, we show the global distribution for each attribute in the background. Attribute distributions help users identify the meaning of states. For instance, when visualizing weather, a state with a high temperature distribution immediately suggests a summer state. For transitions, the distributions of the attributes immediately before and after the transition are shown.

Finally, by selecting an attribute in the menu in the upper right corner of the central panel, users can compare the states with respect to that attribute. Once selected, states in the central panel are recolored by the mean value of the attribute in each state (see Fig. 11). This allows the user to correlate the structure of the dataset with the value of the attribute. States with high average values are colored orange while those with low values are colored blue.

4.2 State Layout

The state layout reflects the positions of states in the *ambient space* consistently across the scales while avoiding overlap. It is designed to highlight the structure of the dataset. There are three phases to compute the layout. The first phase projects the centroids of the *initial states* onto the plane, using multidimensional scaling (MDS) to preserve the pairwise distances between the centroids as well as possible. Next, we iteratively compute the coordinates of states on coarser scales as a weighted average of their children. Traversing consecutive scales from finer to coarser, let $s_1, s_2, \ldots, s_k$ be the states aggregated into state s. The second phase then traverses the scales iteratively and computes the coordinates of each new state as:

$$p(s) = \frac{\sum_{i=1}^{k} p(s_i)\pi(s_i)}{\sum_{i=1}^{k} \pi(s_i)}. \quad (3)$$

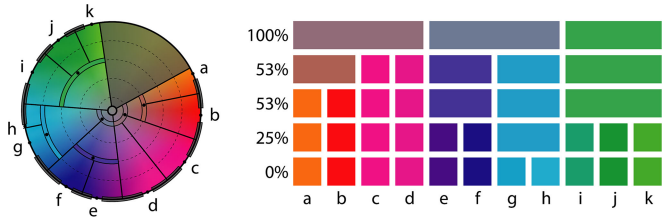

Fig. 7. The color coding procedure. The *initial states* are distributed along the edge of the HSL color wheel. Each state is given a range proportional to its entry in the stationary distribution. Each concentric circle of decreasing saturation represents a subsequently coarser scale. When states merge (note not all merges are shown), the new hue is the weighted average of the corresponding angles. Additionally, we add a "buffer" which acts as an unused color interval between the sub-trees and remove a range of colors from the visualization (yellow was omitted for aesthetic reasons). On the right is the corresponding color assignment across the scales. Each rectangle on the bottom corresponds to a dot on the edge of the color wheel. The upper rectangles visualize how states merge across the scales with each merge represented by a dot on the left. A state which does not merge across the scales maintains its color.

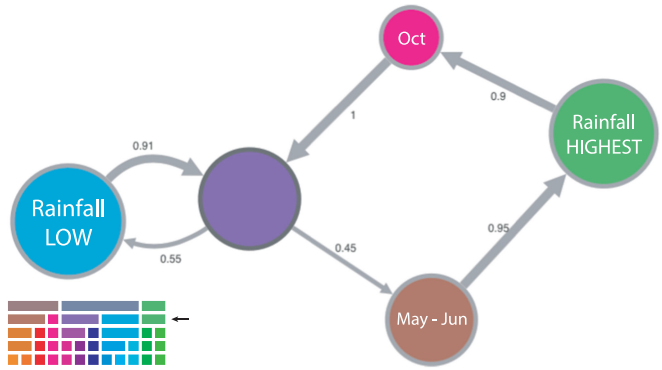

Fig. 8. Automatic labels characterize a state based either on the typical configuration of attributes inside the state or time when the state typically occurs. Here we show two states, which were characterized based on the *Rainfall* attribute (blue and green) and two states characterized by typically occurring in October and between May and June (pink and brown respectively). The central purple state is left blank, indicating that no typical characterization was found.

where $p(i)$ represents the position of state i and $\pi(i)$ is its entry in the stationary distribution. By weighting using the stationary distribution, the states where the process spends more time move less as we change scales. Finally, we apply a cross-scale repulsive scheme by implicitly constructing bounding circles around states and iteratively moving the centers of states with overlapping bounding circles away from each other. This ensures that the states do not overlap in the final representation.

4.3 Color Coding

Color coding has been shown to be effective for visualizing qualitative relationships in the data. It gives the designer three degrees of freedom which they can use to highlight aspects of the visualized structure. If used properly it can enhance the overall aesthetics of the visualization tool.

On a single scale the state layout, sizes and attribute-based labels give the user insight into the structure of the ambient space. However, the cross-scale relationships are still obscured by the single scale visualization.

StreamStory uses color coding to highlight cross-scale relationships and enable the user to traverse the scales efficiently. It uses *proximity based coloring* [39] to visualize the proximity of states in the hierarchical structure and extends it to encode the states' scale (i.e., the finest scale at which the state first appears) using saturation. Intuitively, states in the same sub-tree are considered more similar than states in adjacent sub-trees and are assigned more similar hue. Likewise finer scale states are more saturated than coarser states allowing the user to quickly distinguish more conceptual, high-level states from their lower scale counterparts and allowing them to quickly identify states which persist through the hierarchy. The idea is shown in Fig. 7.

The procedure first lays out the initial states around the external edge of the HSL color wheel. Each state is given a range proportional to its entry in the stationary distribution of the *Markov chain* reflecting the proportion of time the process spends in the state. We add an additional buffer between states from different sub-trees of the hierarchy. It acts as an unused color interval and helps visually separate states from different parts of the hierarchy. The size of the interval is proportional to the scale where the sub-trees split (i.e., states separated at coarser scales are separated by larger intervals). At design time, we allow developers to specify the range of hue used in the visualization. In all examples, we use a range which omits yellow (see Fig. 7).

Hue is assigned to the coarser states as the weighted average of the hue of their children (i.e., the weighted average of the angles) with lower saturation. The weights are determined by the stationary distribution of the Markov chain at the appropriate scale. This makes the color of the larger states dominate their sub-tree (see Figs. 1 and 3). We discretize the saturation into equal intervals from $s_{min} > 0$ to $s_{max} = 1$, numbering the scales 1 to n and assigning saturation $1 - k(s_{max} - s_{min})/n$ to all states with scale k.

4.4 Automatic State Labeling

To help interpretation, StreamStory automatically generates descriptions of states, used as initial labels. Our initial experiments showed that a representation with abstract states and transitions often overwhelms a first-time user, making it difficult to conceptually map the states of the model back to the data. To address this, the system uses an automatic, data-driven state labeling procedure which generates a short textual description of a state, providing a concise mapping from the states to the attributes. The labels are computed during model construction and, for each state, provide a label based on two factors: (a) the values of attributes inside the state compared to other states and (b) the times when the state typically occurs. The label is shown as the default text on the state (Fig. 8). The states can be easily renamed while interacting with the system by selecting the state and entering a new name in the right panel under "State name"— see NYC in Fig 1.

Labels are generated in two phases. The first phase computes labels based on attributes. If this fails, the second phase tries to generate time-based labels. If both phases fail, the state is left blank.

Attribute-based labels are computed by comparing the distribution of each attribute inside the state to the attribute's global distribution. More specifically, the mean value of the states' distribution is compared to the γth and δth percentile of the global distribution. If the mean is below the γth percentile, we assign the label *LOWEST* to the attribute,

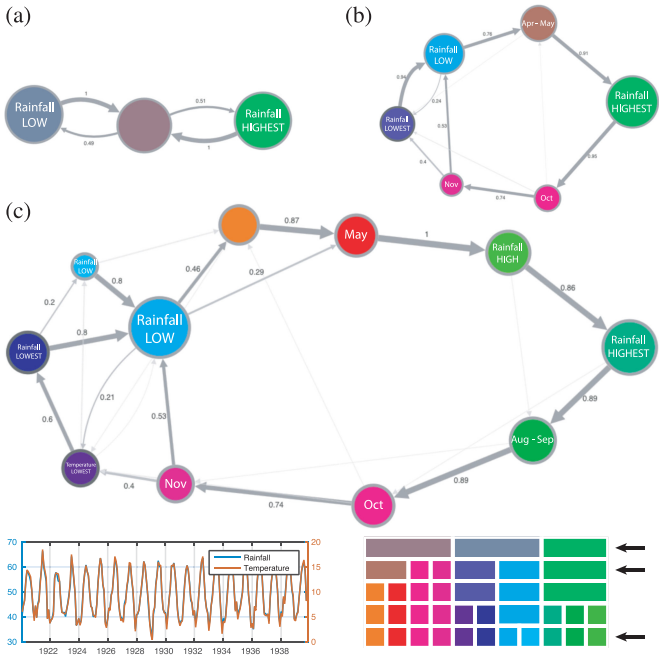

Fig. 9. In the weather dataset we observe a cyclical pattern, indicating periodic long-term behavior with the main states corresponding to different parts of the year. The yearly periodicity appears at the middle (b) and fine (c) scales. In this case, this can be confirmed directly by the parallel plot (bottom-left).

while assigning label *LOW* if the mean is between both percentiles. Analogous rules are used for *HIGH* and *HIGHEST*. As rule of thumb, we use default values $\gamma = 12$ and $\delta = 25$. Among all the attribute labels, we use the one in the lowest/highest percentile as the final state label. If no such label exists, we attempt to generate time-based labels.

The second phase generates labels based on time by making use of the *Time* histograms presented earlier. It scans the histograms at each granularity (e.g., daily, weekly, etc.) searching for cyclically continuous peaks. We consider the kth bin a peak, if it contains more mass than the average bin: $b_k > \sum b_j/n$, where n represents the total number of bins in the histogram and b_k the value of bin k. Two consecutive peaks are considered a single peak. If a single peak with more than ζ mass (with default $\zeta = 0.7$) is found, the peaks' time range is considered as a candidate label. Among all the candidate time-based labels, we select the one with its peak containing the most mass. For example, from the yearly histogram this would produce a label of the form *May - Jun* like the brown state in Fig. 8.

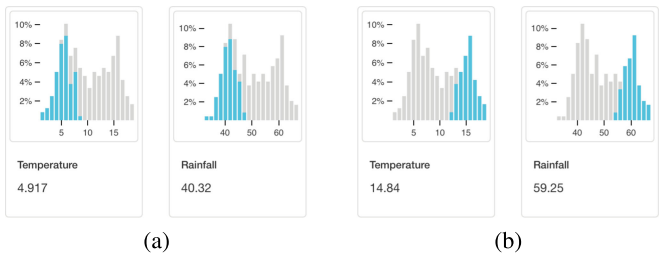

Fig. 10. The *Attribute* histograms indicate that the purple *Rainfall LOWEST* and green *Rainfall HIGHEST* states in Fig. 9a correspond to winter and summer months respectively. This is indicated by a low distribution of temperature and rainfall in the *Rainfall LOWEST* state (a) and a high distribution of temperature and rainfall in the *Rainfall HIGHEST* state (b).

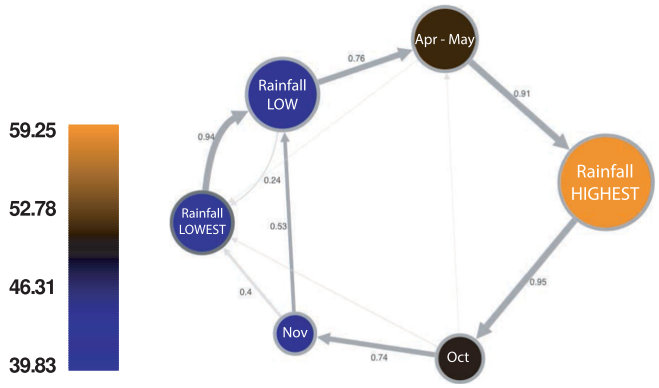

Fig. 11. The distribution of *Rainfall* (in mm) confirms that the states on the right of Fig. 9b are rainier than the states on the left. The same view for *Temperature* is almost identical suggesting a high correlation between the attributes (we only show the view for *Rainfall*).

5 Use Cases

We use three datasets of varying complexity to demonstrate how StreamStory can be used to perform the analytics tasks listed at the beginning of the paper. With all the examples, we provide an interpretation demonstrating how to map elements of the abstract representation back to domain-specific concepts. We use a weather dataset to illustrate how to find and interpret the long-term behavior of the data and a wind dataset to demonstrate how finding the appropriate scales can reveal structure in the data. We present three patterns which are typical when exploring data using StreamStory using a GPS dataset. To illustrate the multiscale structure, we show three different scales for each example. We note that the sub-figures were scaled independently and so should not be directly compared. Finally, we discuss feedback gathered from experts during the development of the tool and compare StreamStory to related techniques.

5.1 Weather Data

We begin with a weather dataset used as an illustrative example throughout the paper. The dataset consists of average monthly temperature and rainfall readings collected at Nottingham Castle, UK over 20 years between 1920 and 1940. We demonstrate how StreamStory can be used to identify the main states of the dataset, find long-term recurrent behavior and map the abstractions back to domain-specific concepts.

We construct a representation with 12 *initial states*. In addition to the rainfall and temperature, we include the previous months' values in the feature vectors as explained in Section 3.6, raising the dimension to 4. Fig. 9 shows the model at three different scales along with the original data.

At a coarse scale, we identify three main high-level states corresponding to different parts of the year. In Fig. 9a we see two large states with a transition state between them. The labels indicate the right state represents rainy weather while the left state represents low rainfall. Checking the individual states, we observe that the left state also has a lower temperature distribution (Fig. 10a) as opposed to the right state (Fig. 10b).

This suggests the right state represents summer while the left state represents winter. We confirm this using the yearly histograms showing the left state lasts from November to

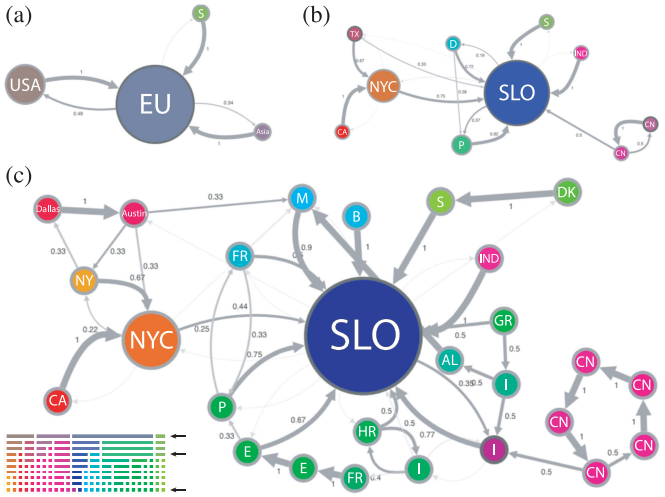

Fig. 12. The GPS dataset. The main states represent locations around the world. At the finest scale (c), the dataset demonstrates several typical patterns which can be observed using StreamStory. The central state represents Slovenia, the researchers home location. Several trips to various locations in Europe form a wheel-like pattern around the central state. On the bottom we see a path corresponding to a road trip through southwest Europe (Italy (I) - France (FR) - Spain (E) - E - Portugal (P) - SLO). Finally on the bottom right, we see a cycle corresponding to a vacation in China.

April and the right state from June to September. While in this case the interpretation is straightforward, the automatic labelling presents a clear suggestion which can easily be verified by the user using the supporting views.

On the middle scale, we identify long-term recurrent behavior in the form of a yearly cycle. Moving to a finer scale with six states (Fig. 9b), the right state remains fixed, while the other two states split. The blue winter state splits into two, while the middle red state splits into three disconnected states. The cyclic shape suggests periodic long-term behavior, supported by the original data on the bottom of Fig. 9. Furthermore, the automatic labels indicate that the top- and bottom-most states represent spring and autumn months respectively. Using the *Time* tab (Fig. 5), we see that *Rainfall LOWEST* (dark purple) occurs between December and March (Fig. 5a) while *Rainfall HIGHEST* (green) occurs between June and September (Fig. 5b), confirming the yearly cycle with winter on the left.

The labels also suggest where we may look for additional structure. The summer season is labeled with high rainfall while states in the winter are labeled with low rainfall, suggesting that the summers are significantly rainier than the winters in the UK. Through a menu in the corner of the panel we visualize the distribution of each of the attributes by showing the mean value of the attribute across the states. Fig. 11 shows the values for *Rainfall*. Recall, states with low values are shown in blue, while states with high values are indicated by orange. The view for *Temperature* looks identical, suggesting a high correlation between the two, inline with the plot in Fig. 9.

At the finest scale (Fig. 9c), we see a very dynamic winter compared to the relatively stable summer. With 12 states, we see both summer and winter states splitting up into many smaller states. We can use the structure we found at the coarser scales to help us interpret the additional states. Since the visualization draws the states consistently across scales, we know when in the year each region corresponds to. The number of states and non-uniform transition probabilities in the winter region suggest that winters are much more unpredictable than other seasons.

This demonstrates how we are able to identify and interpret the recurrent behavior, some expected (such as yearly cycles), others unexpected (e.g., more variance in winter weather). By inspecting the labels, *Attribute* histograms and *Time* histograms, the conceptual meaning of each state and the transitions become transparent to the user. Importantly, the system allows for quick verification of the interpretation by the user. This is a simple illustrative example which verifies that known patterns can easily be identified. Although different techniques could be used to analyze this dataset, the remaining two examples illustrate the versatility of StreamStory.

5.2 GPS Data

Our next example demonstrates several different types of patterns the user can observe using StreamStory. The dataset contains personal GPS measurements (latitude, longitude and timestamp) collected by a researcher using Google's Location History tool over a period of 3.5 years between July 2012 and January 2016. It includes several trips around Europe as well as to the United States and Asia. In this example, we manually labeled the states by checking the coordinates of their centroids on a map. Note that this could be automated in the case of geographic data.

At a coarse scale (Fig. 12a), the dataset exhibits a large central state with three smaller satellites. Checking the coordinates of the centroids, we find that the central blue state corresponds to Europe, which corresponds to "home," while the left orange state corresponds to the United States. The other states are Asia on the right and Scandinavia on top respectively. The latter was separated from Europe due to its relatively large distance to the other European countries visited.

At the middle scale, additional structure appears. In Fig. 12b we see the United States state split into New York, California and Texas. New York is the largest of the 3 states, reflecting a summer internship in 2014. Europe splits into Slovenia (SLO - large blue), where the individual lives, Germany (D - small cyan) and Portugal (P - bottom green). Asia is split into three purple states. The bottom-right two states reflect a trip to China (CN), while the purple state near the Scandinavia is India (IND). This captures various trips the individual has taken.

The finest scale (Fig. 12c) demonstrates three of the most typical patterns which can be observed using StreamStory. Europe becomes a wheel-like pattern with Slovenia in the center. In the bottom-right corner, the trip to China becomes a cycle with five states, corresponding to different stops during the trip. The cycle indicates that the trip started and ended in the same city. On the bottom, we see a path representing a road trip through Europe. The path starts with Italy through France (I and FR—both green states on the bottom), then Spain (E) and finishing in Portugal (P—leftmost green state). Unfortunately, the return trip was not recorded. In the US, New York splits into New York City and upstate New York while Texas splits into Dallas and Austin. These states do not exhibit any special structure.

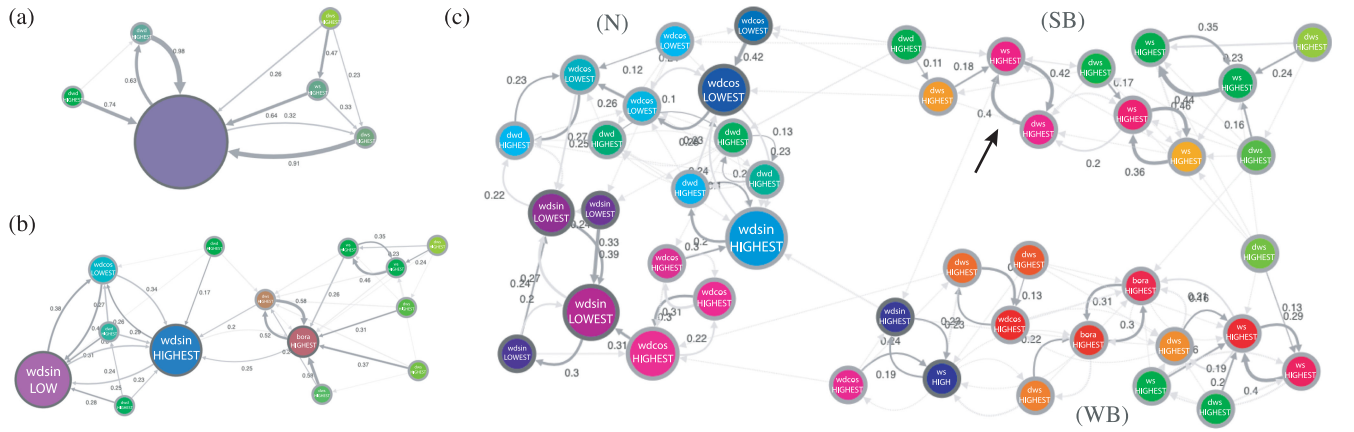

Fig. 13. Wind dataset. We identify several groups of states representing different wind behavior, including two groups of Bora winds with different dominant directions. (a) At a coarse scale, there is a single dominant state with several satellites representing extreme behavior (e.g., high wind speed). (b) At a medium scale, two main groups appear - the group with two large states on the left represents calm winds while the star-like group on the right represents the typical gusty, high-speed Bora winds. (c) At a fine scale, we identify two groups of Bora on the right with different intensities and dominant directions (SB) and (WB). (WB) is due to a split in the topography of the valley, which causes weaker gusts. By inspecting the additional views, we find that the weaker Bora group (WB) changes direction faster than the stronger Bora group (SB). The group on the left (N) represents calm winds. A strong cycle between high wind speed and a large speed variation representing gusty wind is shown by the arrow. Inspecting the data at a finer scale shows no additional structure.

This example also illustrates a drawback of the color coding. At the finest scale, the US and China have similar colors despite being spatially distant. This happens because at the coarsest scale their relatively small subtrees (when compared to Europe) are assigned adjacent, relatively small color ranges. A possible mitigation is to increase the spacing between subtrees. However, this would narrow their color ranges and reduce differentiation within the subtrees. In general it is not possible for the assignments at the coarser levels to reflect the distance between the subtrees since it would require projecting the ambient space into a one dimensional color space.

5.3 Wind Data

In our final example, we illustrate the utility of traversing the scales to identify unexpected patterns and characteristics in higher-dimensional, less structured data than the examples above. The data consists of measurements from a weather station in Ajdovščina, Slovenia—second-level measurements taken of wind speed and direction during March 2016. The station is located in the Vipava valley which experiences the Bora wind phenomenon [40], strong gusts of wind which can reach over 150 km/h. This particular wind pattern has remained largely unexplored and there are no *expected* patterns. The dataset comes with an expert's annotation indicating the presence of Bora at a resolution of 10 minutes.

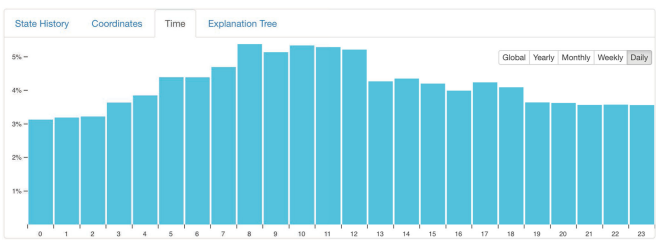

Fig. 14. The daily histogram of a state predominantly characterized by Bora does not show any dominant daily patterns. With a small bias toward morning hours, we see that the state occurs at all times of day.

For constructing the model, we do not use the annotations, rather visualizing them after the model is constructed to help identify correlations and structure in the data. The dataset has 6 attributes, including wind speed, direction (represented as the sine and cosine of the angle) and the change in each of the values, sampled every second for one month. In this case, a plot of the raw data is uninformative but the resulting model at three scales can be seen in Fig. 13.

At a coarse scale in Fig. 13a, the dataset exhibits a similar structure as our previous example—a large central state with satellites. The satellite states mainly represent extreme values with the states on the right corresponding to the Bora winds. Moving to a middle scale in Fig. 13b, we identify the main states of interest. The group around the two large states (left) is characterized by low wind speeds and represents predominantly calm winds. The three largest states in the group (blue, purple and cyan) differ mainly by the direction of the wind while the small green states represent the changes in direction. The group on the right, consisting of smaller states, is characterized by high wind speeds. These represent the high speed gusts characteristic of Bora winds. The histogram in Fig. 14 shows the unstructured nature of Bora winds.

At a finer scale, in Fig. 13c we see two groups appear on the right. This represents the most relevant scale for understanding this phenomena. The two groups on the right, (SB) and (WB), represent two dominant directions of Bora winds. The group (SB) represents wind which is perpendicular to the orographic barrier (a ridge running along the valley). This has been the main meteorological understanding of the phenomenon - cold air spilling over the ridge and travelling down the face typically gives rise to high wind speeds and the gusty nature of the wind.

The other group, (WB), represents a *second, previously unidentified type of Bora*, which is much weaker and travels more along the direction of the valley. This is due to a split in the valley which causes weaker gusts but is nonetheless consistent with Bora winds [40]. This type of Bora was first

TABLE 1
Measurements of the Initialization Time of a Model Indicate a Strong Dependence on the Number of Initial States

Initial states	10	20	40
Dataset A	$1m41s$	$1m50s$	$2m31s$
Dataset B	$4m11s$	$6m10s$	$14m54s$

observed in this dataset, due to the difficulty in manually isolating a suitable subset of data where this phenomenon occurs. StreamStory highlights the phenomenon through strong transitions among the states in the group.

At this scale, there are other relevant phenomena. In addition to high wind speeds, as indicated by the state labels, the two Bora groups have large changes of wind speed and smaller changes in wind direction. Inspecting the *Attribute* histograms, we note however that the weaker group changes wind direction faster than the stronger group. The direction of the wind is most spread out and changes the fastest in the non-Bora group (N). This is primarily indicated by the state labels and uniform transition probabilities. This lack of structure in the non-Bora group is to be expected, since low wind speeds allow for quick changes in direction.

We also see some strong cycles appear in the two Bora groups. For example, in the strong Bora group (SB), there is a cycle between high variation in wind speed and high wind speed in the southwest direction (*dws HIGHEST* and *ws HIGHEST*), pointed out in Fig. 13c. This cyclic pattern reflects the gusty nature of the wind—the two states capture the gusts (large change in wind speed) and the peak speeds respectively. A similar pattern, but less pronounced, can be seen in the weak Bora subgroup with the difference that the dominant direction is northwest.

Moving to even finer scales, we do not see any other interesting structure. We only find the random variations of the wind rather than any interesting patterns. This illustrates how choosing the correct scale is often critical to finding meaningful patterns in data.

5.4 Expert Feedback

During the development of the tool, user feedback was solicited on usability. This includes domain experts, data analysis experts as well as non-experts. Initially, the system was tested on industrial processes with the non-expert users. Two test iterations were conducted at different stages of development. Each included five experts who deal directly with the industrial process (e.g., foreman on a factory floor). Each expert was asked to interact with the system for an hour before giving feedback. The first iteration showed that the users had significant trouble mapping the states to conceptual interpretations. The addition of the automatic labels and *icicle plot* improved interpretability greatly.

The second group of users were domain experts for the data (e.g., a meteorologist for the weather data). The experts' feedback also confirmed the importance of automatic labels in interpreting the structure of the model as well as highlighting the utility of the histogram views and cross-state coloring by attribute for exploring a data set. Several views, especially machine learning specific views such as the decision trees were only useful to users who were familiar with the technique, e.g., data analysis experts.

An important evolution during the development of the system is the increase in automatic choices of parameters. While this does reduce the manual control of the system, it greatly improves the system's "out-of-the-box" usability. Just as in the case of state labels, the default parameters suggest a coherent big picture of the data allowing the user to explore the data without concerning themselves with parameter tuning (which can be done after a user is comfortable with the system). A typical example of this is the automatic choice of scales to visualize based on a measure of change in the representation. Once the user is comfortable with the system, setting different parameters manually can be exposed.

5.5 Comparison to Other Techniques

Though there are many different techniques for visualizing time series (as discussed in Section 2), we could not identify any directly comparable system. In particular, the target of most of the other approaches are not non-expert users and often not larger datasets. In the supplementary material, available online, we show the result of visualizing one of our examples (the weather example of Section 5.1) with *TimeCurves* [9]. Comparing the results, we found the main drawbacks of using *TimeCurves* over StreamStory are: (a) the inability to map elements of the visual representation back to the data and (b) the clutter produced by large datasets. On the other hand, by discretizing data into states, StreamStory removes information from the visual representation which is especially noticeable on small datasets. Overall, we did not find a direct comparison especially meaningful since the techniques are aimed at different use cases. For instance, we do not expect StreamStory to perform well on the Wikipedia dataset[4] from [9]. StreamStory works best for large datasets of medium dimension, where many iterations of a cycle are present. In the case of very high dimension and sparse data, different approaches are needed since StreamStory removes many of the important details in the data. However, as the examples show, in many cases, the system allows for intuitive exploration of data highlighting interesting patterns.

5.6 Performance Evaluation

In this section, we demonstrate how the time needed to construct a StreamStory model varies depending on the size of the dataset and choice of parameters. We note that once the model is constructed, the visualization is fully interactive. In the experiments above, the weather (6K) and GPS data (400k) were nearly instantaneous (1s). The wind data was much larger (145 MB), and so took approximately 11 minutes. We performed an experiment, testing two datasets of different sizes, fixing the dimension and varying the number of *initial states* in the model. The first dataset (A) contains $285k$ measurements (155 MB), while the second (B) contains ≈ 3 M measurements (500 MB). For each dataset, we selected 7 attributes randomly and configured 10, 20 and 40 initial states respectively. We ran the experiment on a laptop with a

4. We could not test this since we do not have access to the preprocessed dataset.

quad core 2.5 GHz CPU with 8 MB cache and 16 GB of RAM. The results are summarized in Table 1. The table indicates a strong dependence on the number of initial states.

6 Conclusion and Future Work

In this paper, we presented a novel methodology and tool for visualizing large multivariate time series called StreamStory. StreamStory abstracts the data and visualizes its long-term behavior using hierarchical Markov chains, encoding the properties of the Markov chain into a graph-based visual representation. The abstraction is complemented with several tools, driven by statistical and *machine learning* methods, which map the visual abstractions back to domain-specific concepts and suggest possible interpretations. To find interesting patterns, StreamStory allows users to interactively change the scale with the visual cues remaining consistent through the transitions. To construct the abstraction requires minimal tool specific knowledge and the system can help explore datasets without extensive domain knowledge.

Overall, it allows users to interactively identify and interpret the main states of interest in the dataset, highlighting long-term recurrent behavior. We demonstrated the usefulness of the tool on three real-world datasets of varying complexity. Using StreamStory, we identified several previously known and unknown patterns, presented feedback gathered from experts and illustrated the scalability of the system. In the future, we plan to extend the system in several different directions, which include:

Interactive Feedback. Rather than fixing the structure of a model during construction, we will investigate how the user can interactively merge and split states.

Encoding Additional Information Into States. Additional information could be included in the representations of the states by constructing a *TreeMap*-like encoding to convey cross-scale information about their sub-trees or splitting their borders into sections to show the distribution of the attributes inside the state. However, it is not clear whether this would introduce clutter.

Comparing Multiple Datasets. The current system is designed to investigate one dataset at a time. An interesting question is how to use this abstraction to compare two or more datasets (e.g., wind measurements taken at different times of the year).

Alternative Representations. Finally, how can we combine our approach with other techniques discussed in Section 2, to obtain similar conceptual abstractions for a wider class of data.

Acknowledgments

The authors thank Luka Bradeško for providing the GPS dataset and Maruška Mole for providing the wind dataset and expertise. This work was supported by the Slovenian Research Agency (N1-0058) and by the EC projects ProaSense (FP7 612329) and OPTIMUM (H2020-MG-636160).

References

[1] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale, "A review of temporal data visualizations based on space-time cube operations," in *Proc. Eurographics Conf. Vis.*, 2014, pp. 23–41.

[2] W. Aigner, S. Miksch, H. Schumann, and C. Tominski, *Visualization of Time-Oriented Data*, 1st ed., Berlin, Germany: Springer, 2011.

[3] H. Hochheiser and B. Shneiderman, "Dynamic query tools for time series data sets: Timebox widgets for interactive exploration," *Inf. Vis.*, vol. 3, no. 1, pp. 1–18, Mar. 2004.

[4] S. Havre, B. Hetzler, and L. Nowell, "Themeriver: Visualizing theme changes over time," in *Proc. IEEE Symp Inf. Vis.*, 2000, pp. 115–123.

[5] R. Peng, "A method for visualizing multivariate time series data," *J. Statistical Softw. Code Snippets*, vol. 25, no. 1, pp. 1–17, Feb. 2008.

[6] T. D. Wang, C. Plaisant, B. Shneiderman, N. Spring, D. Roseman, G. Marchand, V. Mukherjee, and M. Smith, "Temporal summaries: Supporting temporal categorical searching, aggregation and comparison," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 6, pp. 1049–1056, Nov. 2009.

[7] M. Burch, F. Beck, and S. Diehl, "Timeline trees: Visualizing sequences of transactions in information hierarchies," in *Proc. Work. Conf. Adv. Vis. Interfaces*, 2008, pp. 75–82.

[8] S. Haroz, R. Kosara, and S. L. Franconeri, "The connected scatterplot for presenting paired time series," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 9, pp. 2174–2186, Sep. 2016.

[9] B. Bach, C. Shi, N. Heulot, T. Madhyastha, T. Grabowski, and P. Dragicevic, "Time curves: Folding time to visualize patterns of temporal evolution in data," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 1, pp. 559–568, Jan. 2016.

[10] H. Jänicke and G. Scheuermann, "Steady visualization of the dynamics in fluids using epsilon-machines," *Comput. Graph.*, vol. 33, no. 5, pp. 597–606, 2009.

[11] Y. Gu and C. Wang, "Transgraph: Hierarchical exploration of transition relationships in time-varying volumetric data," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2015–2024, Dec. 2011.

[12] M. Pylvänen, S. Äyrämö, and T. Kärkkäinen, "Visualizing time series state changes with prototype based clustering," in *Proc. Conf. Adaptive Natural Comp. Algorithms*, 2009, pp. 619–628.

[13] O. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook*. New York, NY, USA: Springer, 2005.

[14] T. Liu, "Application of markov chains to analyze and predict the time series," *Modern Appl. Sci.*, vol. 4, no. 5, 2010, Art. no. 162.

[15] V. Soloviev, V. Saptsin, and D. Chabanenko, "Markov chains application to the financial-economic time series prediction," *ArXiv e-prints*, http://adsabs.harvard.edu/abs/2011arXiv1111.5254S, 2011.

[16] S. Vasanthi, M. Subha, and S. T. Nambi, "An empirical study on stock index trend prediction using markov chain analysis," *J. Banking Financial Serv. Insurance Res.*, vol. 1, no. 1, pp. 72–91, 2011.

[17] W. Ching, S. Zhang, and M. Ng, "On multi-dimensional markov chain models," *Pacific J. Optimization*, vol. 3, pp. 235–243, 2007.

[18] C. Wang, T.-Z. Huang, and W.-K. Ching, "A new multivariate markov chain model for adding a new categorical data sequence," *Math. Problems Eng.*, vol. 2014, 2014, Art. no. 502808.

[19] D. Zhu and W. Ching, "Multivariate markov chain models for production planning," *Int. J. Intell. Eng. Inform.*, vol. 1, no. 2, pp. 156–173, 2011.

[20] I. Davidson, "Visualizing clustering results," in *Proc. SIAM Int. Conf. Data Mining*, 2002, pp. 3–18.

[21] M. Grobelnik, L. Stopar, and B. Fortuna, "Newssearch: Search and dynamic re-ranking over news corpora," in *Proc. Conf. Data Mining Data Warehouses*, 2012, pp. 209–212.

[22] N. Cao, D. Gotz, J. Sun, and H. Qu, "Dicon: Interactive visual analysis of multidimensional clusters," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2581–2590, Dec. 2011.

[23] J. Johansson, P. Ljung, M. Jern, and M. Cooper, "Revealing structure within clustered parallel coordinates displays," in *Proc. IEEE Symp. Inf. Vis.*, Oct. 2005, pp. 125–132.

[24] Y. Xiang, D. Fuhry, R. Jin, Y. Zhao, and K. Huang, "Visualizing clusters in parallel coordinates for visual knowledge discovery," in *Proc. 16th Pacific-Asia Conf. Advances Knowl. Discovery Data Mining*, 2012, pp. 505–516.

[25] J. Woodring and H.-W. Shen, "Multiscale time activity data exploration via temporal clustering visualization spreadsheet," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 1, pp. 123–137, Jan. 2009.

[26] M. Luboschik, C. Maus, H. J. Schulz, H. Schumann, and A. Uhrmacher, "Heterogeneity-based guidance for exploring multiscale data in systems biology," in *Proc. IEEE Symp. Biol. Data Vis.*, Oct. 2012, pp. 33–40.

[27] D. Auber, Y. Chiricota, F. Jourdan, and G. Melançon, "Multiscale visualization of small world networks," in *Proc. 9th Annu. IEEE Conf. Inf. Vis.*, 2003, pp. 75–81.
[28] C. Stolte, D. Tang, and P. Hanrahan, "Multiscale visualization using data cubes," *IEEE Trans. Vis. Comput. Graph.*, vol. 9, no. 2, pp. 176–187, Apr. 2003.
[29] B. Kulis and M. I. Jordan, "Revisiting k-means: New algorithms via Bayesian nonparametrics," in *Proc. 29th Int. Cnf. Int. Conf. Machine Learning*, Edinburgh, Scotland, 2012, pp. 1131–1138.
[30] J. Norris, *Markov Chains*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[31] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*. Hoboken, NJ, USA: Wiley, 2009.
[32] K. Deng, P. G. Mehta, and S. P. Meyn, "Optimal kullback-leibler aggregation via spectral theory of markov chains," *IEEE Trans. Autom. Control*, vol. 56, no. 12, pp. 2793–2808, Dec. 2011.
[33] R. Agaev and P. Chebotarev, "On the spectra of nonsymmetric laplacian matrices," *Linear Algebra App.*, vol. 399, pp. 157–168, 2005.
[34] F. R. K. Chung, *Spectral Graph Theory*. Boston, MA, USA: AMS, 1997.
[35] D. Boley, G. Ranjan, and Z.-L. Zhang, "Commute times for a directed graph using an asymmetric laplacian," *Linear Algebra Appl.*, vol. 435, no. 2, pp. 224–242, 2011.
[36] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*. Berlin, Germany: Springer, 1981, pp. 366–381.
[37] A. Inselberg and B. Dimsdale, "Parallel coordinates: A tool for visualizing multi-dimensional geometry," in *Proc. 1st Conf. Vis.*, 1990, pp. 361–378.
[38] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*. Berlin, Germany: Springer, 2001, vol. 1.
[39] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner, "Hierarchical parallel coordinates for exploration of large datasets," in *Proc. Vis.*, Oct. 1999, pp. 43–508.
[40] M. Mole, "Study of the properties of air flow over orographic barrier," Ph.D. dissertation, Univ. Nova Gorica, Nova Gorica, Slovenia, 2017.

Luka Stopar received the BS degree in computer science and mathematics from the University of Ljubljana, Ljubljana, Slovenia, in 2013. He works as a researcher with J. Stefan Institute, and he is working toward the PhD degree at the Jozef Stefan International Postgraduate School. His research intrests include developing and applying techniques for analyzing high-volume multivariate data streams, including visualization, exploration, summarization, prediction and anomaly detection.

Primoz Skraba received the PhD degree in electrical engineering from Stanford University, in 2009. He is currently a Senior Researcher with the Jozef Stefan Institute, Slovenia. His main research interests are applications of topology to computer science including data analysis, machine learning, sensor networks, and visualization.

Marko Grobelnik is currently employed with the AI Lab at Jozef Stefan Institute. Previously he was in the Computer Science Department at the University of Ljubljana and Department of intelligent systems and Department for knowledge technologies at the Jozef Stefan Institute. His primary focus of research and applications is intelligent data analysis which deals with unconventional scenarios going beyond classical statistical approaches and solving problems including unstructured or semi structured data.

Dunja Mladenic works as a researcher and a project leader at J. Stefan Institute, leading Artificial Intelligence Laboratory and teaching at J. Stefan International Postgraduate School. She has extensive research experience in study and development of Machine Learning, Data/Text Mining, Semantic Technology techniques and their application on real-world problems. She was also a visiting researcher with the School of Computer Science, Carnegie Mellon University for several years. She has numerous published papers in refereed journals and conferences, edited and authored several books, served on program committees of international conferences and organized international events.