StreamStory: Exploring Multivariate Time Series on Multiple Scales

Luka Stopar, Primoz Skraba, Marko Grobelnik and Dunja Mladenic, paper #302

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 summarizes the dataset in a qualitative manner using states and transitions. It shows a centralized structure with a large state in the middle, representing the researcher's home location, and many smaller satellite states representing various trips to several locations in Europe (e.g. Germany, Greece, Portugal, Denmark, 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 the NYC is highlighted, with several distinct short trips to NYC, and one longer stay shown in the middle (June-Oct. 2014), corresponding to a summer internship.

Abstract— In visualizing multivariate time series, it is difficult to simultaneously present both the dynamics and the structure of the data in an informative way. This paper presents an approach for the interactive visualization, exploration, and interpretation of multivariate time series. Our approach builds an abstract representation of the data based on a hierarchical, multiscale structure, where each scale is modeled as a continuous time Markov chain. All the aspects of the visualization are designed with the multiple scales in mind. Encoding visual cues consistently across scales enables intuitive exploration of the different scales, allowing a user to quickly find appropriate scales for their data. The construction uses a combination of machine learning methods so that it requires minimal user input. We also present a number of coordinated views and tools which help the user understand how structure in the abstract representation maps to the data. Some of these include attribute distribution and time histograms, a time series matrix - providing a cross-scale historical view of the data, and automatic labeling of the states. The visualization of the representation and the associated tools are integrated into an interactive, web-based tool called StreamStory. Using this tool, we show how this approach can be used to understand and find interesting long term and recurrent behavior in data using four different datasets, coming from weather data (i.e. rainfall and temperature), traffic monitoring sensors, GPS traces and wind velocity data. These represent a wide range of data and complexity but all show various interesting recurring patterns which we interpret with the help of StreamStory.

Index Terms—Time Series Analysis, Visualization Models, Visual Knowledge Discovery, Multiresolution Techniques, Coordinated and Multiple Views, Visual Knowledge Representation, Time-varying Data

- *Luka Stopar is with Jozef Stefan Institute and Jozef Stefan International Postgraduate School. E-mail: luka.stopar@ijs.si.*
- *Primoz Skraba is with Jozef Stefan Institute. E-mail: primoz.skraba@ijs.si.*
- *Marko Grobelnik is with Jozef Stefan Institute. E-mail: marko.grobelnik@ijs.si.*
- *Dunja Mladenic is with Jozef Stefan Institute and Jozef Stefan International Postgraduate School. E-mail: dunja.mladenic@ijs.si.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.

1 Introduction

Time series are one of the most common starting points for data analysis and 1D signals graphed over time is perhaps the most ubiquitous visualization today. We often capture multiple different measurements from time-varying systems giving rise to *multidimensional* or *multivariate time series*. Exploring this type of data is difficult as it is not obvious how to best visualize it. Common solutions such as common axis/plot or as some parallel view are unsatisfactory. They become difficult to interpret even for medium dimensional data. When dealing with complex and large data sets, users must identify, zoom into interesting time intervals and remove clutter manually, stealing their focus and hindering productivity.

Our goal is to visualize and explore multi-variate time series using a representation which highlights the intrinsic structure and longer-term dynamics *with minimal user input*. In this paper, we present a novel visual representation based on constructing a *hierarchy of Markov chains* and its implementation in a tool we call **StreamStory**. To construct the representation, we use a combination of machine learning techniques and complement the main graph-based visualization with a number of features and views to help the user interpret the data by interactively exploring the multi-scale representation.

StreamStory is especially adept at highlighting and investigating *recurrent behaviour* in time series data. Recurrence is easily recognizable as a *cycle* in the graph-based representation of the Markov chain. For example, the daily cycle can be represented as a cycle between "day" and "night" states. In Figure 1, there is a cycle present between the states labeled *NYC* (New York City) and *CA* (California), representing movement between the two locations in the United States. StreamStory helps the user understand abstract patterns in the representation with such features as automatic labels for states and an interactive timeline.

The visualization is based on the following pipeline: the structure of a multivariate time series is first captured by disregarding the temporal component and using clustering to partition the data. Each cluster captures a typical configuration of the data and is associated to a single operational state of a Markov chain. The dynamics are reintroduced to model the transition probabilities and finally, the states and transitions are aggregated to construct a hierarchy. An interactive web-based user interface (Figure 1) visualizes each Markov chain in the hierarchy as a graph (center panel) augmented by multiple other views. These include a historical overview of the data over the entire hierarchy (bottom panel) and information on the individual states (right panel).

The main contributions of this paper are:

1. A novel representation of multi-variate time series based on multi-scale Markov chains built from data. The Markov chain model enables longer-term temporal patterns to be visualized through graphical elements. The construction is hierarchical, providing different granularities at which the user can analyze the data.
2. A collection of tools which simplify exploration and interpretation of the qualitative representation. The tools provide multiple views on the data and using statistical techniques, suggest possible interpretations.
3. The approaches are integrated into a fully interactive web-based visualization tool, *StreamStory*. It requires minimal domain and tool-specific knowledge while allowing users to quickly identify and interpret different patterns in the data. The system is publicly available at `http://streamstory.ijs.si` with the examples in the paper as well as for experimentation.

2 Previous Work

We focus on 4 categories of previous work most closely related to StreamStory: multivariate data visualization, time series visualization, cluster visualization and multi-scale visualization. We place special attention on techniques which are in the intersection of two or more categories. In each category, we focus on techniques able to visualize large datasets. Despite each of these categories offering a wide array of tools and techniques, visualization of large, multivariate time series is still an understudied area.

Multivariate Data Visualization: Multivariate data visualization is a well established and mature research field. Keim and Kriegel [22] categorize multivariate data visualization techniques as *geometric projection* techniques such as *parallel coordinates* and *scatterplot matrices*, *icon-based* techniques like *Chernoff faces*, *pixel-based* techniques and *hierarchical and graph-based* techniques. Many of these have been refined and combined over the years. To alleviate the cluttering produced by large datasets, Fua et. al. [17] proposed *hierarhical parallel coordinates* which add hierarchical organization, statistical aggregation and brushing to interactively explore the dataset. Novotny [31] used clustering as a data abstraction tool to group similar data points, visualizing groups instead of individual data points. More recently, Geng et. al. [18] applied angular histograms to alleviate clutter. Angular histograms bin the data points along each axis and rotate the bins in the mean direction of the polylines going out of the bin. This allows users to quickly identify data frequencies along each axis and the relationships between the axis. While parallel coordinate type approaches can efficiently visualize a reasonable number of dimensions, they do not encode temporal information efficiently.

Cao et. al. [7] visualize multivariate data by encoding properties of attributes into the visual attributes of clusters, positioning the clusters to reflect the inter cluster relationships.

The techniques mentioned above are suitable for visualizing low to medium dimensional data. When the number of dimensions becomes large (such as text documents) techniques like *multidimensional projection* [8, 21, 26, 29] can be used to project data points onto a lower-dimensional space (usually a plane). Patterns can then be visualized by highlighting the structure of the density distribution [15, 19]. But while techniques developed for visualizing multivariate data are excellent at revealing the structure of the data, they fail to encode the temporal component efficiently, making it infeasible to find temporal patterns.

Time Series Visualization: Time series visualization is perhaps the most ubiquitous visualization today. Beyond simple graphs, many of the existing techniques are domain specific and can only be used for specialized purposes. For an overview, we refer the reader to [2, 3].

Generic time series visualization techniques developed in recent years include *connected scatterplots* which display two time series in a scatterplot, connecting two consecutive points with a line and indicating the temporal direction with arrows. While able to highlight interesting patterns, connected scatterplots can only handle two-variate time series and suffer from clutter when visualizing large datasets.

Van Goethem et. al. [38] use trend detection to visualize time series data. They create *trends* by grouping time series that behave similarly over time. Trend detection is performed at different granularities, bearing some resemblance to our multi-scale approach. Their approach solves the problem of highlighting trends in short-term behavior in similar time series, while our approach focuses more on identifying qualitative patterns in long-term behavior.

Peng et. al. [32] visualize high dimensional time series by discretizing the values of each attribute into quantiles and visualizing the result in a time series matrix together with the median and variation of each attribute and the average of all time series, used to identify trends. Although the approach is effective when identifying trends and differences between time series it does not highlight recurrent behaviour.

Recently, Bach et. al. [4] presented a generic technique to visualize temporal data called *Time Curves*. *Time Curves* are based only on similarity between the data points and do not require an explicit representation of each data point. They warp the time axis so that similar points appear close to each other and encode the temporal component as curves between sequential data points. The authors show that *time curves* can be used to identify interesting qualitative patterns in the data in a multi-scale manner. However, *time curves* suffer from a high computational cost incurred by using MDS [10] on the whole dataset. This makes their use infeasible for visualizing large datasets.

Cluster Visualization: There is no single best visualization to visualize the results of a clustering algorithm. The choice of the visualization depends on the aspect of the clusters the designer would like to empha-

Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

size. If the goal is to highlight the behaviour of the variates across the clusters, one might use *parallel coordinates* [5,20,24,41] to visualize a representative of each cluster. On the other hand, projecting their representatives onto a lower-dimensional space (usually the plane) [11,27] and using visual attributes to describe the shape of the cluster can quickly highlight the structure of the dataset. Cao et. al. [7] provide an example of this type of visualization, encoding a cluster's data attributes and statistical information as *treemap*-like [35] icons.

If the data is assumed to have a hierarchical or multi-scale structure, visualizing hierarhical clusters is often a more appropriate choice [13, 34]. StreamStory combines the three approaches described above into a single interactive visualization tool, using automatic labeling to highlight differences in the ambient space and adds a temporal component making it more adept at visualizing time series.

In their work, Pylvanen et. al. [33] describe an approach similar to our own. By applying k-means [39], they visualize the operational states, including the order of transitions, in temporal multivariate data. The primary application of their tool is to find abnormal behavior, so their tool only shows states as collections of points with no additional information except the projected coordinates, making it difficult to interpret a state solely based on the properties of the associated cluster. Furthermore, their approach requires users to define the number of states in the visualization before the representation is actually constructed. Reconstructing the visualization on coarser or finer scales requires recomputation and so hinders interactive data exploration.

Multi-Scale Visualization: Multi-scale visualizations identify patterns in data at several levels of detail. This is especially important when the relevant level of detail is not known a priori. Several visualization tools provide some form of zooming or multi-scale interface [36], many of them domain specific [23]. *Semantic zooming* is a general technique to develop multi-scale interactive visualizations [40]. It includes representing objects metaphorically on a 2D canvas. The user can then manipulate the canvas viewport to view different parts of the canvas. When changing the size of the viewport (elevation) the representation of the objects is changed. In such approaches, data can be abstracted in several ways. Two examples include *hierarchical clusters* and *data cubes* [36].

In our work, we leverage multi-scale techniques in combination with other techniques to produce an interactive visualization tool capable of highlighting temporal patterns in multivariate time series data.

3 METHODOLOGY

The goal of StreamStory is to construct an abstraction which captures the qualitative structure of a dataset to help understand long-term dynamics with minimal user input. We base the abstraction on Markov chains [30], using the states and transitions to represent the structure and dynamics respectively. A key feature of our representation is to extend this abstraction over multiple scales allowing users to find suitable scales to interpret their data through real-time interactive exploration. This minimizes context switching and allows the user to focus on interacting with the representation. There are 4 steps in constructing our representation, shown in Fig. 2.

We first consider the time series as a point cloud in multidimensional space (Fig. 2(a)), disregarding the time component. The next step identifies typical operational states in the data (Fig. 2(b)). The points are partitioned using a clustering algorithm based on a metric between the data points. For example, in Figure 2, two noisy periodic signals are partitioned roughly according to the phase. 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 [25] to construct the partition, but any clustering technique can be used. The number of centroids (in the case of k-means) or the cluster radius (in the case of DP-means) should be chosen based on domain knowledge and the finest scale the users require. For large datasets, we often limit the number of states since a large number of states increases initial construction time and the visualization becomes too cluttered at the finest levels.

Each cluster is then associated with a state of a Markov chain (Fig. 2(c)). We refer to these as *initial states*. They form the lowest-scale representation of the data and are the basis for further computation. The dynamics are modeled through the transition probabilities between the states, extracted from the *jump chain*. This design choice has two main consequences. First, by using continuous time Markov chains over their more popular discrete-time counterparts, 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 visual clutter by removing transitions of the form $i \to i$ from the visualization.

The data needed to represent a continuous time Markov chain is stored in a transition rate matrix, denoted by Q. Its non-diagonal elements q_{ij} represent the rates of transitioning from state i to state j (in $\#jumps/time\,unit$). 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 of the Markov chain from data, we compute 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 last step is to construct a hierarchy of such Markov chains by aggregating the *initial* states into *coarser*, higher level states (Fig. 2(d)) and associating each level of the hierarchy with a scale used in the final visualization. The construction of the hierarchy is a two step process: (i) constructing the topology (e.g. deciding which states are merged) and (ii) aggregating the states of the lowest-scale Markov chain to obtain higher scale representations.

To construct the topology, we begin with a partition and the Markov chain induced by this partition. StreamStory implements two approaches to construct the hierarchy. The first is a bottom-up distance-based approach which uses *mean linkage* agglomerative clustering (UPGMA) [14] 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 [12] 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 [1,6,9]. 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.

For step (ii), we compute the final representation of the hierarchical model by aggregating the states. As input, 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 at which scale. 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:

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

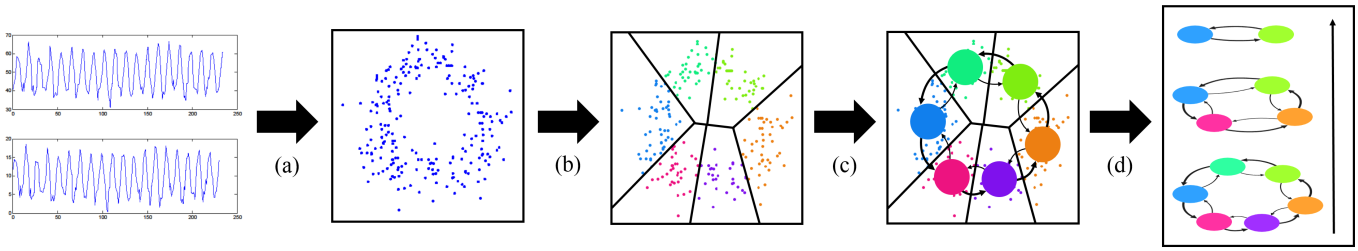

Fig. 2: Overview of the pipeline. (a) The multivariate time series is represented as a point cloud. To illustrate, we show two noisy approximately periodic signals, mapping to points in 2D. (b) The space is partitioned and the typical states of the system are found using a clustering algorithm. Here the regions correspond to intervals of the phase of the signal. (c) The partition is translated into a Markov chain model, with each state representing a partition cell. (d) The Markov chain model is simplified by aggregating states, giving a multi-scale view of the model.

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.

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. To choose the scales, we associate each scale to 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 chosen scales. The number of clusters is chosen as the minimum of half of the number of initial states and 10. 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.

3.1 Highlighting Recurrent Behavior

We provide the user two options which help highlight and reveal recurrent behaviour in some datasets. 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: The first approach is based on a *time-delay embedding* [37]. It adds values from previous time steps to the feature vector. For example, 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 effectively doubling the dimension of the signal. This can help reduce ambiguities in recurrences by effectively considering the derivative of the individual time series, since for two points to be similar, their paths (for some number of time steps) must also be similar. This approach was originally developed for recovering the dynamics from chaotic systems [37].

Static Context Based on Time: The second approach uses qualitative temporal information extracted from the timestamps of individual measurements. It requires the user to select a time granularity when constructing the model. The feature vector is extended with a categorical feature which describes when in time the measurement occurred on the specified time granularity. The categorical feature is represented by a binary vector[1], hence the dimension increases by the number of possible categorical values. 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.

This emphasizes certain 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. 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.

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

4 Visual Representation

With the underlying hierarchical model constructed, it is presented in a web-based, interactive user interface shown in Figure 3. It consists of three main parts: (i) the central panel showing the Markov chain on the current scale, (ii) the bottom panel showing a cross-scale historical overview and (iii) the right panel showing details about selected states.

Fig. 3: The visualization consists of 3 panels. The central panel shows the Markov chain model at the current scale. The bottom panel shows a cross-scale historical overview using a time series matrix. States on the two panels are associated by color. The right panel shows the distribution of attributes in the currently selected states (blue border).

The central panel is the main panel in our visualization. It visualizes a single scale of the hierarchical Markov chain model. This avoids overwhelming the user with parallel views or the occlusion problems associated with a 3D encoding. We visualize the Markov chain associated with the chosen scale as a diagram with circles and arrows, representing states and transitions respectively. To switch scales, users can use the scroll function or slide a scrollbar on the left of the panel.

When rendering a state, we use five visual attributes: (a) radius, (b) position, (c) color, (e) label and (f) border to encode different properties of the state (see Fig. 4). 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 **multi-scale 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 chosen so that

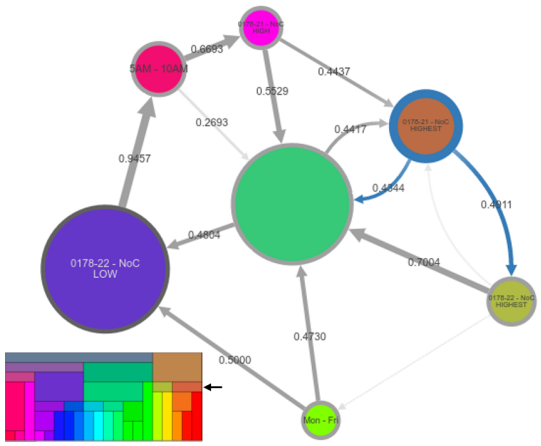

Fig. 4: Different visual attributes encode different properties of the states: (a) The area of each circle encodes the proportion of time the system spent in the state in the training dataset. (b) The position reflects the states' position in the ambient space relative to other states. (c) The color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

the area of the circle is linearly proportional to the states' entry in the stationary distribution of the Markov chain [30]. The 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 4.1 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 4.2 for details). Throughout the paper, we use a color square (Fig. 4) to show assignment of colors.

When the model is constructed, StreamStory automatically assigns labels to states using the method presented in Section 4.3. 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. 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.

In our representation, we model dynamics as transitions between states, 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. To further reduce clutter, we group transitions into three categories: (a) high, (b) medium and (c) low probability transitions. We define transitions with probability greater than α as high, between α and β as medium, and below β as low. By default, we set $\alpha = 0.4$ and $\beta = 0.2$. We highlight the arrows for group (a) with a darker color and blur those in group (c) by drawing them using a dotted line and omitting a label (see Fig. 4). To further highlight dominant transitions, we also enable the user to interactively reduce their number using a horizontal scrollbar at the bottom of the main panel.

At each scale, the sequence of states occurring in the data represents a discretization of the path of the time series though the ambient space. We show the resulting sequences for all (chosen) scales on the bottom panel of Fig. 3 as a *time series matrix*, giving a historical overview of the data. Each row of the matrix is associated with a scale in the hierarchical representation. The scale is labeled left of the row. The time series matrix shares its color coding with the central panel, allowing users to quickly identify states.

The time series matrix is interactively integrated with the representation on the central panel. The label of current scale is highlighted and the selected state in both representations is highlighted. This allows users to select a state by clicking it on either representation. When selected through the time series matrix, the appropriate scale is automatically shown in the central representation.

To help interpret temporal patterns, we add a second time-based view on the bottom panel. The view shows when the selected state occurred on different time granularities using histograms. Fig. 5 shows the daily histogram of the selected state from Fig. 4. The histogram shows that

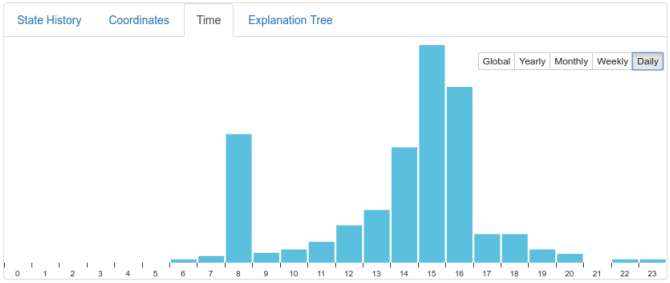

Fig. 5: A daily histogram showing when in time a state typically occurs. The state corresponds to dense traffic, which the histogram shows that it typically occurs in the morning and afternoon (see Section 5.1).

the state usually occurs in the morning and afternoon. The bottom panel offers two other views for the selected state: parallel coordinates and an explanation/decision tree. Parallel coordinates are a standard tool while the decision tree is can be used to explain high dimensional clusters [16]. These tools are aimed primarily to specialists, so we defer the explanation to the supplementary material.

The right panel (Fig. 6) visualizes the mean and distribution of each attribute inside the selected state. 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.

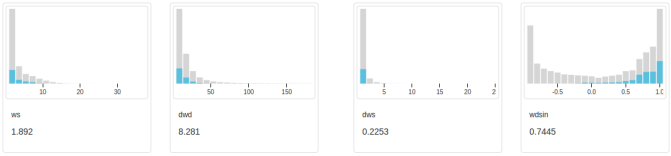

Fig. 6: The panel showing the distribution of 4 attributes of a state compared to the global distributions - the data are measurements of wind speed and direction (Section 5.2). The attributes are wind speed (ws), change in wind speed (dws), change in direction (dwd), and the sine of the direction (wdsin), where the 0° is north and 90° is east. The state is characterised by a primarily easterly wind direction.

Finally, users can compare states through a menu in the corner of the central panel. This view shows the mean values of a single attribute across states (see Fig. 12). States are colored orange for high values of the selected attribute and blue for low values.

4.1 State Layout

There are 3 phases to compute the state 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

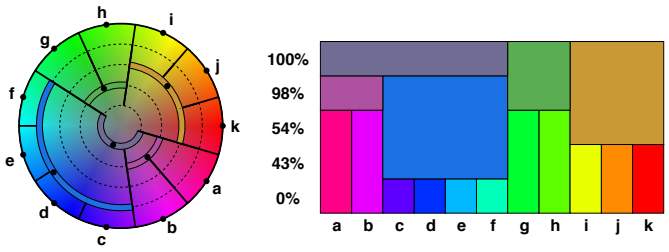

Fig. 7: An illustration of the color coding procedure. The initial states are distributed equally along the edge of the HSL color wheel. Each concentric circle of decreasing saturation represents a subsequently coarser scale. When two states merge, the new hue value is computed as the weighted average of the two corresponding hues (not all merges are shown). On the right is the corresponding color assignment across scales. Each rectangle corresponds to a dot on the left. A state that does not merge across scales maintains its color.

the states aggregated into state s. The second phase then traverses the scales iteratively and computes the coordinates of each new state as:

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

where $p(i)$ represents the position of state i and $\pi(i)$ its entry in the stationary distribution. By weighting by 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, which ensures that the states do not overlap in the final representation.

4.2 Color Coding

We have two goals when assigning color to the states. We would like (a) similar states to have a more similar color than very different states and (b) to differentiate between coarser scale states and finer scale states. We use two attributes of the HSL color scale to encode two properties of each state in the hierarchical representation (see Fig. 7).

The first of these properties is the finest scale at which the state first appears (i.e. the states' scale), encoded using saturation. Intuitively, the finer scale states have a more saturated color than the coarser scale states. We discretize the saturation into equal intervals from $s_{min} > 0$ to $s_{max} = 1$, numbering the scales 1 to n and assigning saturation $1 - \frac{k(s_{max} - s_{min})}{n}$ to all states with scale k.

While saturation reflects the scale where a state first appears, we use hue to encode the relationship between states at the same scale. We use a recursive procedure to calculate the hue of each node. The first phase traverses the aggregation tree (representing how states are merged) to determine an ordering of the initial states so that merged states are always adjacent in the ordering. The second phase then assigns hues to each of the states bottom-up. Using the ordering described above, the hue range $(0, 2\pi]$ is distributed equally between the initial states, as shown on the boundary of the color wheel in Fig. 7 (left). As for the layout, a parent state's hue is the weighted average of the hues of its children, with the weights are based on the stationary distribution of the Markov chain at the appropriate scale. This makes the color of more dominant states dominate their subtree, see Fig. 1 and 3.

4.3 Automatic State Labeling

Our initial experiments showed that a representation with abstract states and transitions often overwhelms a first-time user and makes it difficult to interpret the data. To address this, the system provides automatic, data-driven state labeling providing a concise mapping from the states to the attributes. The labels are computed during model construction and, for each state, provides 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).

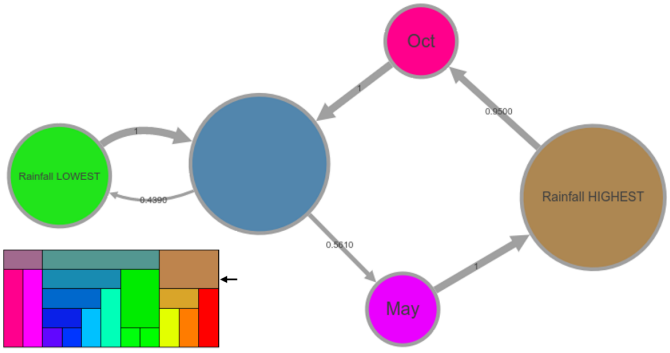

Fig. 8: Automatic labels for two states which typically occur in October and May as well as two states characterized by high and low rainfall respectively. If no typical description is found, the label is left blank (e.g. the blue state).

Labels are generated using a two phase procedure. 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 state distribution is compared to the γ-th and δ-th percentile of the global distribution. If the mean is below P_γ, we assign the label *LOWEST* to the attribute, 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. The second phase scans the histograms at each granularity (e.g. daily, weekly, etc.) searching for cyclically continuous peaks. We consider the k-th 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 at more than ζ mass (with default $\zeta = 0.7$) is found, the peaks' time range is considered as 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 *Jul-Sep*.

5 Experiments

We provide some examples of usage and insights which can be gained by using the StreamStory system. We use 4 different datasets to highlight all the relevant aspects of our visualization. In each case, we point out how the features of the system aided in the analysis. We also show the dependence of the construction time on dataset size and number of initial states.

5.1 Weather & Traffic Data

We first illustrate how StreamStory can help find periodic and cyclic behavior as well as highlight variations in the cycles. We consider 2 different datasets: the first is a weather dataset which consists of average monthly rainfall and temperature readings collected at Nottingham Castle, UK over 20 years between 1920 and 1940; the second dataset consists of two traffic counters at different points on the bypass around Ljubljana, Slovenia over a period of one year in 2014. Both datasets exhibit cyclic behaviour with similar patterns but at different time scales - weather is cyclic on a yearly scale and traffic on a daily scale. It is worth noting that these scales were found directly in the data with no parameter tuning and with no prior knowledge of the data.

We begin with the simplest dataset. The model for the weather data, was constructed using 12 initial states. The resulting structure is clearest when the number of states coincides with the period of the behaviour (12 months aligns with the yearly weather cycle). However,

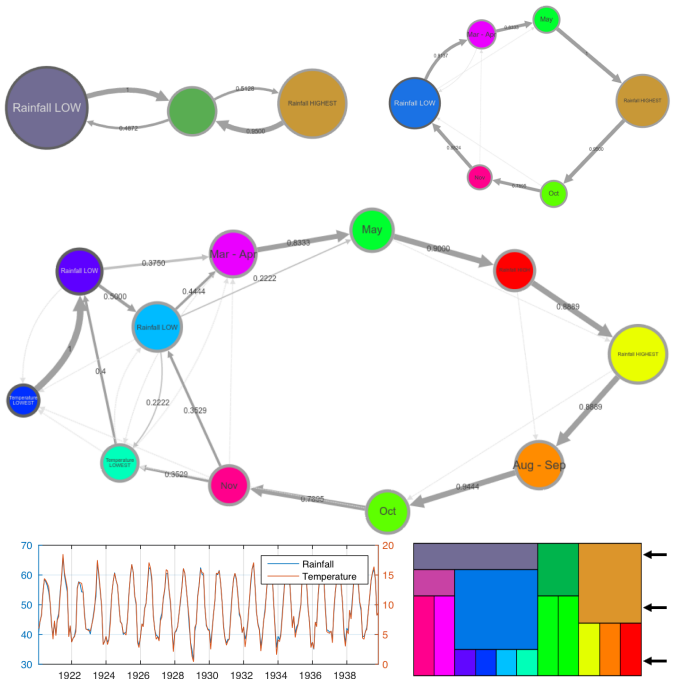

Fig. 9: The model for rainfall and temperature readings along with a parallel plot of the data. The yearly periodicity of the signals appears as a cycle at the middle and fine scales.

the structure can be seen with any larger number of states (including non-multiples of 12). In addition to each month's average temperature and rainfall, we also include the previous month's values (as described in Section 3.1) raising the dimension to 4. Figure 9 shows the model at 3 different scales and the original data.

At a coarse scale, there are 3 states - two large states with a smaller transition state between them. The labels indicate the right state represents rainy weather while the left state represents low rainfall. Checking the individual states (Figure 10), we see that the left state also has a lower temperature distribution as opposed to the right state. This suggests the right state represents summer while the left state represents winter. We confirm this using the yearly histograms (the left state lasts from November to April and the right state from June to September).

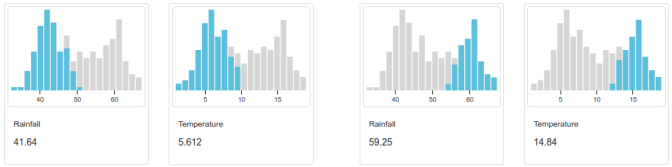

Fig. 10: Attribute distributions of the Rainfall LOW (left) and Rainfall HIGHEST (right) from Fig. 9. Rainfall LOW also has low temperatures while the opposite is true for both attributes for the right state.

Moving to a finer scale with 6 states, the right state remains fixed, while the other two states split and the middle state splits into two disconnected states. The left purple state splits into 3 connected states. Here we begin to see the typical cyclic shape, suggested by the two periodic signals in Fig. 9. Furthermore, the automatic labels indicate that the two middle states represent May and October respectively. Using the *Time* tab (Fig. 11), we see that *Rainfall LOW* occurs between December and March while *Rainfall HIGHEST* occurs between June and September, confirming that we see the yearly cycle with winter on the left. In this case, the labels suggest where we may look for additional structure. The summer season is labeled with high rainfall and the winter with low rainfall, suggesting that the summers are significantly rainier than the winters in the UK. Viewing the values of both the attributes across the states, we visualize each of the attributes

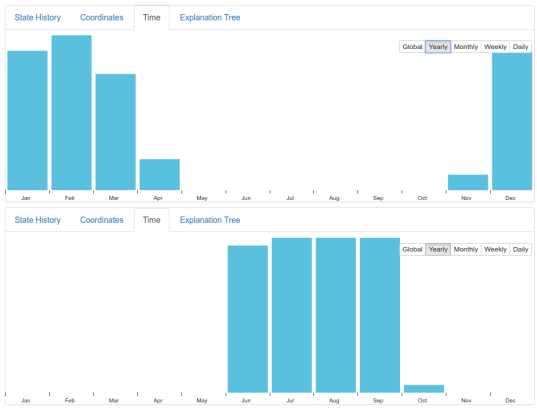

Fig. 11: Yearly histograms of two states from Fig. 9. The histograms suggest the corresponding states typically occur in winter (top) and summer (bottom) months respectively.

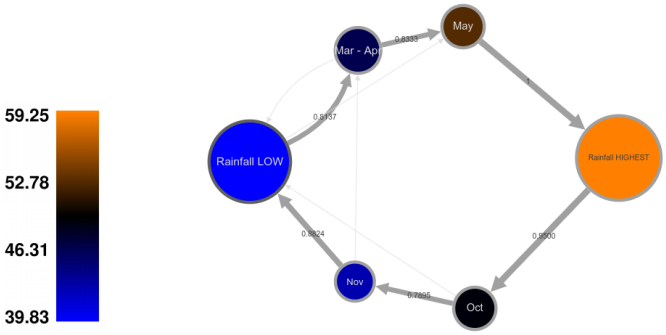

Fig. 12: The distribution of rainfall (in *mm*) confirm that the states on the right are rainier than the states on the left. The same view for temperature is almost identical suggesting a high correlation between the attributes.

across all the states rather than individually. Recall, states with low values are shown in blue while high values are indicated by orange. Fig. 12 shows the values for *Rainfall*. The view for *Temperature* looks identical, suggesting a large correlation between the two, inline with the plot in Fig. 9.

At the finest scale with 12 states, we see both the summer and winter states splitting up into many smaller states. We use the structure we found at the coarser scales to help us interpret the additional states. The visualization immediately suggests that a typical summer and (especially) winter are more unpredictable than spring and autumn. This can be observed directly from the visualization, since the number of states on the left and right sides is greater than the top and bottom. Since the visualization draws the states consistently across scales, we know when in the year each region corresponds to. This is an example of something which cannot be seen by plotting the time series (Fig. 9).

The second example comes from two traffic counters positioned on the highway ring around Ljubljana, Slovenia measuring the rate of cars passing each counter, sampled every hour. The model is shown in Fig. 13 along with a plot of a one month sample of the data. At a coarse scale, we see 3 states. The purple state on the left represents a nighttime state with a low car rate. The brown state on the right represents heavy rush hours traffic in the morning from 7 AM to 8 AM and in the afternoon between 3 PM and 5 PM. The bottom green state represents moderate traffic and occurs at noon and in the evening hours. This was found by examining the daily histogram for each state.

Switching to a middle scale, we find a wheel-like structure consisting of several cycles. The green state with moderate traffic now forms the center of the diagram, while the purple state on the left is nighttime. Time flow travels in the clockwise direction with the red and pink states representing morning and the green and yellow states on the

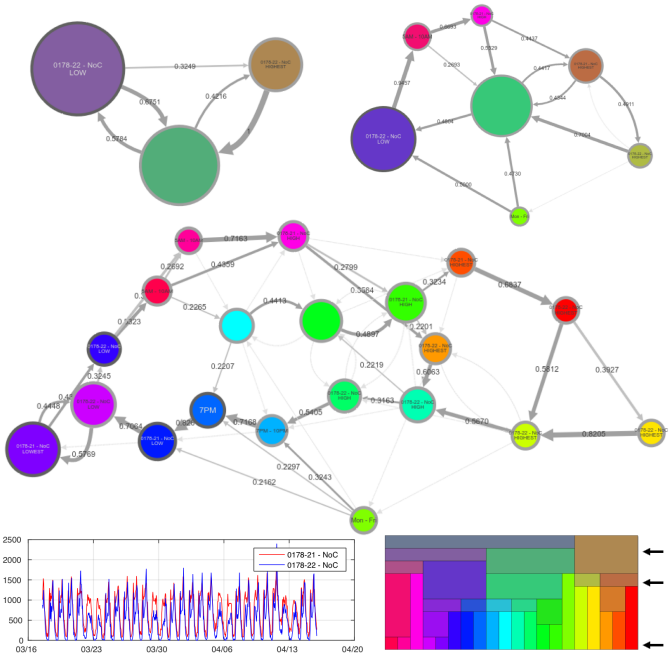

Fig. 13: Data from two traffic counters positioned on the bypass around Ljubljana, Slovenia. At the middle scale (upper-right), there is a wheel-like structure with the green state in the center representing moderate traffic. The finest scale (bottom) reveals two cycles: the nightly cycle (through the blue and purple states on the left), and the afternoon cycle (through the red and yellow states on the right).

bottom-right representing evening. The brown state represents rush hour, occurring between 7-8AM and 3-5PM. At this scale, all states have a high probability transition into the moderate traffic state. This can be expected since we only use the amount of traffic rather than any temporal information when constructing the model, so a moderate rate is always the transition state between high and low traffic intensitys.

There are two main cycles. The first is the night cycle, beginning with moderate evening traffic in the central green state and transitioning to the purple state (very low traffic) representing nighttime. The cycle continues through the red and pink states, representing morning rush hour, and finishes back the center state, which now represents moderate traffic around noon. The second cycle again leaves the center state, through the brown and yellow states, representing afternoon rush hour, and again finishing back in the center state. This highlights a difference between the weather dataset and the traffic dataset. In the weather dataset, spring and autumn were separated states, whereas in the traffic dataset, the midday and afternoon moderate traffic gets merged into one state. This is due to the use of previous values as features (Section 3.1). In the weather data set, while May and September have similar distributions, the previous month (April and August respectively) are sufficiently different to separate the states. In the traffic case, the sampling rate of one hour is sufficiently high that the difference in previous samples is not sufficient to separate the moderate traffic state.

At the finest scale, we find more structure in these cyclic structures. As at the coarser scale, the purple state on the left represents night and time flows in the clockwise direction, with the blue, red and pink representing morning states. The green states in the middle represent midday states. These are highly dynamic showing the unpredictability of traffic at the two locations during the day, i.e. the variation in traffic intensity is quite high. The red, orange and yellow states on the right of the diagram represent afternoon rush hour. From the time histograms, we see that these mostly occur on weekdays. The exception is the right-most yellow state which represents Friday rush hour. It has the highest car rate on both counters, due to extra traffic coinciding with the beginning of the weekend.

Finally, the cyan and blue states on the bottom are evening states. These are characterized by a moderate car rate. The bottom-most

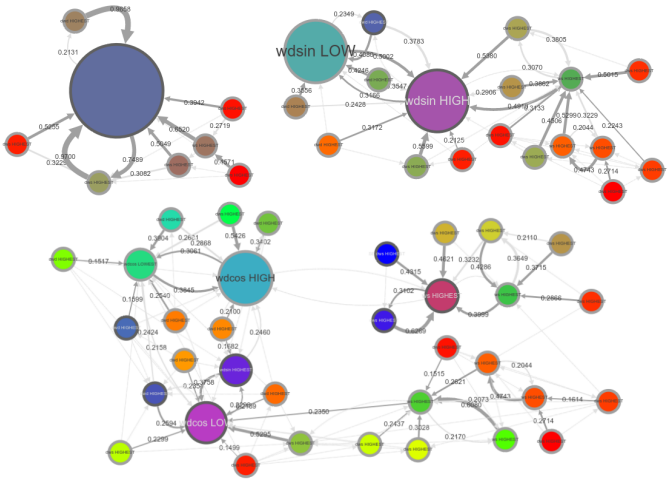

Fig. 14: The model of the wind dataset. The coarse scale (top-left) has a single dominant state with several satellite states. At a medium scale (top-right), three main groups appear - the groups around the two large states represent different directions of calm winds while the right represents the typical gusty, high-speed Bora winds. At the finest scale (bottom), the left group corresponds to calm weather, while the two small groups on the right represent two types of Bora winds, differentiated by the dominant wind direction.

state typically occurs intermittently on weekday evenings. This is an anomalous state where the traffic intensity is low but above average.

These two examples exhibit typical cyclic behaviour that can be readily interpreted using the visualization. While the cyclic structure is not surprising, the visualization helps identify additional structure such as variations in the cycles as well as non-obvious states such as Friday afternoon rush hour (as opposed to other days of the week).

5.2 Wind Data

Next, we show an example of less structured data. 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 [28], strong gusts of wind which can reach over 150 km/h. This particular wind pattern has remained largely unexplored. We find several known characterizations as well as several new characterizations of the wind patterns. Importantly, the dataset comes with an expert's annotation indicating the presence of Bora at a resolution of 10 minutes.

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. 14.

At a coarse scale, the data exhibits a centralized structure, with a dominant central state and several satellite states. The satellite states mainly represent extreme values, so the bottom right states correspond to Bora winds. Moving to a middle scale, we see three main groups appear. The two groups centered on the large states (center and left) differ mainly by the change in wind direction but are still predominantly calm winds. The group on the right, consisting of smaller states arranged in a star shape are characterized by high wind speeds. These represent the high speed gusts characteristic of Bora winds.

At a finer scale, we see two small groups appear on the right. This represents the most relevant scale for understanding this phenomena. The two small groups represent two dominant directions of Bora winds. The first (bottom right) 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 the gusty nature of the

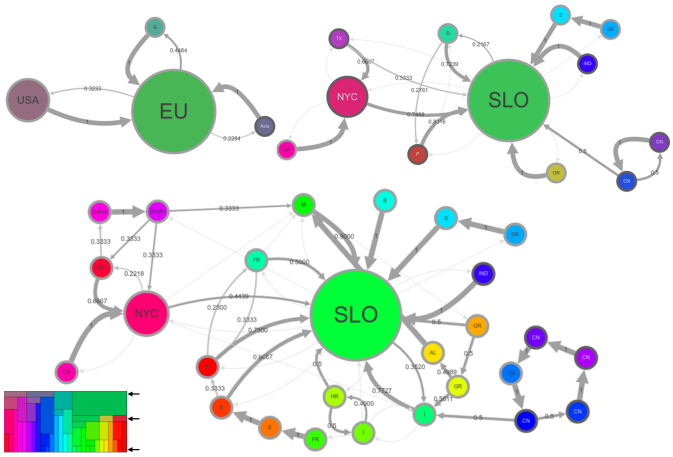

Fig. 15: The model of a person's movement over 3.5 years (GPS coordinates). At the finest scale, there are several interesting cycles including one corresponding to a vacation in China (blue states on the right - CN) and a longer cycle (bottom states) corresponding to a road trip through southwest Europe (Slovenia (SLO) - Italy (I) - France (FR) - Spain (E) - E - Portugal (P) - SLO).

wind along with the high wind speeds. The second small group (top right) is a second 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 [28]. This second type of Bora was first observed in this dataset.

At this scale, other relevant phenomena were observed. In addition to high wind speeds, the two Bora groups have large changes of wind speed and lower changes in wind direction. 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. This lack of structure in the non-Bora group is to be expected as it is wind whose speed and direction are highly random. Since the wind speed is low, the direction may change often.

We also see some strong cycles appear in the two Bora groups. For example, in the weak Bora group (top right), there is a cycle between the high variation in wind speed in a southwest direction and high speed in the southwest direction (*dws HIGHEST* and *ws HIGHEST* respectively). This is expected given 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 strong Bora subgroup with the difference that the dominant direction is northwest. Moving to finer scales, we do not see any other interesting structure. We only find the random variations of the wind rather than any interesting patterns.

5.3 GPS Data

Our final example is GPS data based on personal measurements collected by a researcher using Google's Location History tool over a period of 3.5 years between July 2012 and January 2016. The data consists of timestamped GPS coordinates with significant gaps (i.e. missing data) and a large variation in locations. The data includes several trips around Europe as well as to the United States and Asia. Here we manually labeled the states by checking the centroid coordinates of the states on a map. The model is shown in Fig. 15.

At a coarse scale, the dataset exhibits a similar structure as the wind example - a large central state with satellites. Checking the coordinates of the centroids, we find that the central green state corresponds to Europe, which corresponds to "home," while the left purple state corresponds to the United States. The two other states are Scandinavia on top and Asia on the right.

At a middle scale, the Unites States splits into into New York, California and Texas. New York is the largest of the 3 states, reflecting a summer internship in 2014. Europe splits apart into Slovenia (SLO - large green), where the individual lives, Germany (D - small green), Portugal (P - brown) and Greece (GR - yellow) and Scandinavia becomes Sweden and Denmark (S and DK - both light blue). Asia is split into three states. The bottom-right two states reflect a trip to China (CN), while the purple state near the Scandinavian countries is India (IND). This roughly captures various trips the individual has taken.

When viewing the visualization at the finest scale, we see several interesting patterns emerge. In the bottom-right corner, the trip to China becomes a cycle with five states, corresponding to different stops during the trip. On the bottom-left, we see a road trip through Europe, starting from Italy through France (I and FR - both green states on the bottom), then Spain (E -two orange states on the bottom-left) and finishing in Portugal (P - red). Unfortunately, the return trip was not recorded. In the US, New York splits into New York City and upstate New York and Texas splits into Dallas and Austin. The coordinates of the centroids make it straightforward to check the locations.

5.4 User Feedback

During development, several experts and non-experts were asked for feedback on useability. When evaluated by non-experts, the automatic labels and time series matrix were most helpful for interpretation. Indeed, before the implementation of the first two features, non-expert users had trouble understanding the abstract representation of the data. The experts' feedback also included the utility of the histogram views and cross-state coloring by attribute for understanding the structure of the model. Overall, machine learning specific views such as the decision trees were only useful to users who were familiar with the technique, e.g. data analysts, computer scientists, etc.

5.5 Performance Evaluation

Once the model is constructed, the system is fully interactive. The construction time varies depending on the configuration choices and size of the dataset. In the experiments above, the weather (6K) and GPS data (400k) were nearly instantaneous (1s) while the traffic data (12.5 MB) took about 10s. The wind data was much larger (145 MB), and so took approximately 11 min. We also performed a simple 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 ($155MB$), while the second (B) contains $\approx 3M$ measurements ($500MB$). For each dataset, we selected 7 attributes randomly and configured 10, 20 and 40 initial states respectively. The results are summarized in Table 1 and we see that the main dependence is on the number of initial states.

Table 1: Measurements of the initialization time of a model.

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

6 Conclusion and Future Work

We presented a novel abstraction for multivariate time series based on multi-scale continuous time Markov chains extracted from data using machine learning techniques. Based on this, we developed StreamStory, an interactive visualization tool for exploring multivariate time series. The data is shown in qualitative manner - via a hierarchy of directed graphs - allowing users to find suitable scales to interpret the data. The system requires minimal user input and in addition to automatically constructing the representation, it provides numerous auxiliary tools to aid in interpretation. Directions for future work include:

Interactive Feedback: Once constructed, the structure of a model is fixed. We intend to investigate how to allow the user to merge and split states interactively, modifying the structure of a model.

Automatic Parameter Selection: Currently, there are a number of parameters needed for model construction. Another possible direction is to explore mechanisms for automatically selecting and tuning of these parameters.

Online Modeling: Finally, we plan to explore how to perform online updates to the model so the system can deal with streaming data.

REFERENCES

[1] R. Agaev and P. Chebotarev. On the spectra of nonsymmetric laplacian matrices. *Linear Algebra and its Applications*, 399:157 – 168, 2005. Special Issue devoted to papers presented at the International Meeting on Matrix Analysis and Applications, Ft. Lauderdale, FL, 14-16 December 2003.

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

[3] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale. A Review of Temporal Data Visualizations Based on Space-Time Cube Operations. In R. Borgo, R. Maciejewski, and I. Viola, eds., *EuroVis - STARs*. The Eurographics Association, 2014.

[4] 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 Transactions on Visualization & Computer Graphics*, 22(1):559–568, 2016.

[5] M. R. Berthold and L. O. Hall. Visualizing fuzzy points in parallel coordinates. Technical report, Berkeley, CA, USA, 1999.

[6] D. Boley, G. Ranjan, and Z.-L. Zhang. Commute times for a directed graph using an asymmetric laplacian. *Linear Algebra and its Applications*, 435(2):224 – 242, 2011.

[7] N. Cao, D. Gotz, J. Sun, and H. Qu. Dicon: Interactive visual analysis of multidimensional clusters. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2581–2590, Dec 2011.

[8] Y. Chen, L. Wang, M. Dong, and J. Hua. Exemplar-based visualization of large document corpus (infovis2009-1115). *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1161–1168, Nov. 2009.

[9] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[10] T. F. Cox and M. Cox. *Multidimensional Scaling, Second Edition*. Chapman and Hall/CRC, 2 ed., 2000.

[11] I. Davidson. Visualizing clustering results. In *SIAM International Conference on Data Mining*, 2002.

[12] K. Deng, P. G. Mehta, and S. P. Meyn. Optimal kullback-leibler aggregation via spectral theory of markov chains. *IEEE Transactions on Automatic Control*, 56(12):2793–2808, Dec 2011.

[13] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[14] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. Wiley Publishing, 4th ed., 2009.

[15] B. Fortuna, M. Grobelnik, and D. Mladenić. Visualization of text document corpus. *Informatica*, pp. 497–502, 2005.

[16] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, vol. 1. Springer series in statistics Springer, Berlin, 2001.

[17] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *Visualization '99. Proceedings*, pp. 43–508, Oct 1999.

[18] Z. Geng, Z. Peng, R. S. Laramee, R. Walker, and J. C. Roberts. Angular histograms: Frequency-based visualizations for large, high dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, pp. 2572–2580.

[19] G. Heyer, P. Oesterling, H. Jänicke, C. Heine, and G. Scheuermann. Visualization of high-dimensional point clouds using their density distribution's topology. *IEEE Transactions on Visualization & Computer Graphics*, 17:1547–1559, 2011.

[20] J. Johansson, P. Ljung, M. Jern, and M. Cooper. Revealing structure within clustered parallel coordinates displays. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pp. 125–132, Oct 2005.

[21] P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato. Local affine multidimensional projection. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2563–2571, Dec. 2011. doi: 10.1109/TVCG.2011.220

[22] D. A. Keim and H. P. Kriegel. Visualization techniques for mining large databases: a comparison. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):923–938, Dec 1996.

[23] T. Klein, M. van der Zwan, and A. Telea. Dynamic multiscale visualization of flight data. In *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, vol. 1, pp. 104–114, Jan 2014.

[24] R. Kosara, F. Bendix, and H. Hauser. Parallel sets: interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, July 2006.

[25] B. Kulis and M. I. Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. *CoRR*, abs/1111.0352, 2011.

[26] J. H. Lee, K. T. McDonnell, A. Zelenyuk, D. Imre, and K. Mueller. A structure-based distance metric for high-dimensional space exploration with multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):351–364, March 2014.

[27] M. G. Luka Stopar, Blaz Fortuna. Newssearch: Search and dynamic re-ranking over news corpora. In *Conference on Data Mining and Data Warehouses*, 2012.

[28] M. Mole. *Study of the properties of air flow over orographic barrier*. PhD thesis, University of Nova Gorica, 2017.

[29] L. G. Nonato, C. T. Silva, J. D. II, and E. W. Anderson. Interactive vector field feature identification. *IEEE Transactions on Visualization & Computer Graphics*, 16:1560–1568, 2010.

[30] J. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[31] M. Novotny. Visually effective information visualization of large data. In *In 8th Central European Seminar on Computer Graphics (CESCG 2004*, pp. 41–48. CRC Press, 2004.

[32] R. Peng. A method for visualizing multivariate time series data. *Journal of Statistical Software, Code Snippets*, 25(1):1–17, 2 2008.

[33] M. Pylvänen, S. Äyrämö, and T. Kärkkäinen. Visualizing time series state changes with prototype based clustering. In *Proceedings of the 9th International Conference on Adaptive and Natural Computing Algorithms*, ICANNGA'09, pp. 619–628. Springer-Verlag, Berlin, Heidelberg, 2009.

[34] J. Seo and B. Shneiderman. Interactively exploring hierarchical clustering results [gene identification]. *Computer*, 35(7):80–86, July 2002.

[35] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, Jan. 1992.

[36] C. Stolte, D. Tang, and P. Hanrahan. Multiscale visualization using data cubes. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):176–187, April 2003.

[37] F. Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pp. 366–381. Springer, 1981.

[38] G. A. Van, F. Staals, M. Löffler, J. Dykes, and B. Speckmann. Multi-granular trend detection for time-series analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):661–670, Jan 2017.

[39] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[40] A. Woodruff, C. Olston, A. Aiken, M. Chu, V. Ercegovac, M. Lin, M. Spalding, and M. Stonebraker. Datasplash: A direct manipulation environment for programming semantic zoom visualizations of tabular data. *Journal of Visual Languages & Computing*, 12(5):551 – 571, 2001.

[41] Y. Xiang, D. Fuhry, R. Jin, Y. Zhao, and K. Huang. Visualizing clusters in parallel coordinates for visual knowledge discovery. In *Proceedings of the 16th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part I*, PAKDD'12, pp. 505–516. Springer-Verlag, Berlin, Heidelberg, 2012.