

VISUALIZING QUALITATIVE PATTERNS IN MULTIVARIATE TIME SERIES

Luka Stopar

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

Supervisor: Prof. Dr. Dunja Mladenić, Jožef Stefan International Postgraduate School,
Ljubljana, Slovenia

Co-Supervisor: Dr. Primož Škraba, Senior Lecturer, Queen Mary, University of London
London, United Kingdom

Evaluation Board:

Doc. Dr. Bernard Ženko, Chair, Jožef Stefan International Postgraduate School, Ljubljana,
Slovenia

Assoc. Prof. Dr. Zoran Bosnić, Member, University of Ljubljana, Ljubljana, Slovenia

Assoc. Prof. Dr. Jasminka Dobša, Member, University of Zagreb, Varaždin, Croatia

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL



Luka Stopar

VISUALIZING QUALITATIVE PATTERNS
IN MULTIVARIATE TIME SERIES

Doctoral Dissertation

VIZUALIZACIJA KVALITATIVNIH VZORCEV
V VEČDIMENZIONALNIH ČASOVNIH VRSTAH

Doktorska disertacija

Supervisor: Prof. Dr. Dunja Mladenić

Co-Supervisor: Dr. Primož Škraba, Senior Lecturer

Ljubljana, Slovenia, December 2018

To Jasna

Acknowledgments

I would like to express gratitude to my PhD supervisor Prof. Dr. Dunja Mladenec for her guidance and advice throughout my studies. Special thanks goes to my co-supervisor Asst. Prof. Dr. Primož Škraba for his invaluable guidance and his technical and moral support.

I would like to thank Marko Grobelnik for suggesting several ideas and improvements to the tool developed as part of this thesis.

Assistance provided by my colleagues Blaž Fortuna, Jan Rupnik and Luka Bradeško is greatly appreciated.

I would also like to thank Luka Bradeško for providing the GPS dataset and Maruška Mole for providing the wind dataset.

I gratefully acknowledge the funding by EU projects ProaSense (FP7 612329), Optimum (H2020- MG-636160) and the Slovenian Research Agency (N1-0058).

Finally, I would like to thank my girlfriend Jasna and my parents Tanja and Stane for their moral support.

Abstract

This thesis presents an approach for the abstraction, exploration, interactive visualization and interpretation of large multivariate time series. Interesting patterns in such datasets often appear as periodic or recurrent behavior often caused by the interaction between the variables.

To identify such patterns, we summarize the data as conceptual states, modeling temporal dynamics as transition between the states. States are abstractions which summarize all the measurements in a particular region of the ambient space. Transitions, with the associated states, summarize the path made by the time series. This representation can visualize large datasets with potentially billions of examples.

We extend the representation to multiple spatial granularities, which we call scales, allowing the user to find patterns on several distinct scales. The spatial granularity of the states can highlight certain patterns while obscuring others. Therefore, we design a multi-scale representation, allowing patterns to emerge from within coarser scale states or as the interaction between different parts of several states.

The result is an interactive web-based visualization tool called StreamStory. StreamStory couples the abstract representation with several tools that map the abstractions back to domain-specific concepts using techniques from statistics, data mining 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 base the thesis on three main hypotheses: (i) combining techniques from machine learning, data mining and statistics into a tool for data abstraction and visualization can highlight the long-term behavior of a dataset and expose recurrent patterns while mitigating cluttering effects, (ii) long-term and recurrent patterns can emerge at different spatial granularities (i.e. scales) and traversing the scales interactively provides an effective way to identify where such patterns emerge and (iii) the visual abstractions that summarize the structure and dynamics of a dataset can be mapped back to the data and interpreted as domain-specific concepts with quantitative data-driven visual tools.

We use four real-world datasets to confirm these hypotheses. With all the datasets we provide an interpretation demonstrating how to map the abstractions back to domain-specific concepts. We use weather and traffic datasets to demonstrate how to find and interpret the long-term behavior of the data and a wind dataset to demonstrate how traversing the scales can reveal the structure in the data. We also present three patterns which are typical when exploring data using StreamStory with a GPS dataset. We find and interpret several known as well as previously unknown patterns in these datasets.

Povzetek

V tem doktorskem delu obravnavamo pristop preiskovanja ter interaktivne vizualizacije večdimenzionalnih časovnih vrst. V takšnih podatkih so zanimivi vzorci pogosto periodični in se pojavijo zaradi interakcije med spremenljivkami.

Takšne vzorce razkrijemo tako, da podatke povzamemo s konceptualnimi stanji, časovno dinamiko pa s prehodi med njimi. Stanja so abstrakcije, ki povzamejo meritve v določeni regiji prostora, prehodi pa povzamejo pot, ki jo podatki naredijo v času zbiranja. S takšno predstavitevjo lahko povzamemo velike količine podatkov, tudi več milijard meritev.

Predstavitev razširimo na več prostorskih granularnosti ter s tem omogočimo interaktivno preiskovanje na več stopnjah natančnosti. Granularnost prostora poudari nekatere vzorce, druge pa skrije. Posledično predstavimo večgranularno predstavitev, v kateri podrobni vzorci nastajajo iz posameznih grobih elementov ali pa z interakcijo njihovih delov.

Nadalje predstavimo interaktivno spletno vizualizacijo, poimenovano StreamStory. Poleg abstraktne predstavitve StreamStory nudi še statistična orodja ter orodja strojnega učenja in podatkovnega rudarjenja, ki pomagajo pri interpretaciji predstavitve, in sicer tako da abstraktne elemente preslikajo nazaj v domenske koncepte. StreamStory je narejen za laične uporabnike, ki niso specializirani za analizo podatkov, zato je obseg konfiguracije, potrebne za gradnjo in prikaz modela, majhen.

Delo temelji na treh hipotezah. Te so: (i) z združitvijo tehnik strojnega učenja, podatkovnega rudarjenja in statistike v orodje za abstrakcijo in prikaz podatkov lahko prikažemo njihove dolgoročne lastnosti ter ponavljajoče vzorce, poleg tega po omilimo nered, ki nastane pri prikazu večjih količin podatkov, (ii) dolgoročni in ponavljajoči se vzorci se lahko prikažejo na različnih prostorskih granularnostih, zaznamo jih lahko z interaktivno menjavo stopnje natančnosti, in (iii) vizualne abstrakcije, ki prikazujejo strukturo in dinamiko podatkov, lahko preslikamo nazaj v domenske koncepte ter interpretiramo s pomočjo kvantitativnih orodj.

Hipoteze potrdimo na štirih praktičnih primerih. Z razlago vsakega od njih pokažemo, kako se abstrakcije preslikajo v domenske koncepte. Na primeru vremenskih in prometnih podatkov prikažemo, kako lahko s StreamStory odkrijemo in interpretiramo dolgoročno obnašanje podatkov. S podatki o vetru pokažemo, kako lahko z menjavo prostorske granularnosti odkrijemo strukture v podatkih, na primeru lokacijskih podatkov pa pokažemo tri vzorce, ki se tipično pojavijo pri analizi podatkov s StreamStory. V podatkih razkrijemo precej znanih kakor tudi neznanih vzorcev, ki jih ustrezno interpretiramo.

Contents

List of Figures	xv
List of Tables	xxi
List of Algorithms	xxiii
Abbreviations	xxv
Symbols	xxvii
Glossary	xxix
1 Introduction	1
1.1 Motivation	1
1.2 Aims and Hypotheses	4
1.3 Contributions to Science	6
1.4 Overview of the Thesis	6
2 Related Work	9
2.1 General Approaches	9
2.2 Visualizations with a Linear Temporal Axis	9
2.3 Visualizations with a Curved Temporal Axis	10
2.4 Static Graph-Based Visualizations	11
2.5 Cluster Visualization	12
2.6 Markov Chains to Model Temporal Datasets	12
2.7 Multiscale Representations	12
2.8 Machine-Guided Visualizations	13
3 Methodology	15
3.1 Overview	15
3.2 Time Series Embedding	17
3.3 Constructing the States	17
3.4 Modeling Transitions	18
3.5 Constructing the Multiscale Hierarchy	20
4 Instantiating the Methodology	25
4.1 Overview	25
4.2 Time Series Embedding	25
4.3 Constructing the States	26
4.4 Modeling Transitions	28
4.5 Constructing the Multiscale Hierarchy	29
4.6 Highlighting Recurrent Behavior	30

4.6.1	Using Dynamical Context	30
4.6.2	Static Context Based on Time	30
5	Visual Representation	33
5.1	User Interface	33
5.2	Visual Elements	35
5.3	State Layout	38
5.4	Color Coding	39
5.5	Automatic State Labeling	40
6	Use Cases and Evaluation	43
6.1	Weather Data	43
6.2	Traffic Monitoring	46
6.3	GPS Tracking	48
6.4	Wind Monitoring	50
6.5	Expert Feedback	52
6.6	Comparison to Other Techniques	53
6.7	Performance Evaluation	55
7	Conclusions	57
	Appendix A Configuration and Parameters	59
A.1	Dataset and Attribute Selection	59
A.2	Attribute Configuration	59
A.3	Configuring the Data Abstraction Phase	62
A.4	Finishing Configuration	63
	References	65
	Bibliography	69
	Biography	71

List of Figures

Figure 1.1:	The excess of display items, or <i>clutter</i> , in the visualization hinders the exploration and analysis of a multivariate time series. To extract meaningful information, the analyst must zoom into interesting intervals, isolate informative variables and remove clutter manually. Each such step causes a context switch, stealing their focus and hindering their productivity.	2
Figure 1.2:	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 (i.e. Germany, Portugal, Sweden, and Slovenia) as well as the US, China, and India. The panel on the right shows the distribution of the attributes inside the selected state (blue border). Here the selected state is NYC (New York City - Latitude: 40.89° , Longitude: -73.92°). This is inferred using the geographical coordinates in the right panel. The bottom panel shows a cross-scale historical overview of the data and is linked with the central panel interactively and by color. Here 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 which lasted from June to October 2014 and corresponds to a summer internship.	3
Figure 2.1:	A visual representation of an individual’s music listening trends with a <i>Streamgraph</i> [8]. The x -axis represents time and each horizontal stripe represents an artist. The thickness of a stripe visualizes the number of times that their songs were played in a given week.	10
Figure 2.2:	A volumetric climate dataset visualized using <i>TransGraph</i> . Each node in the graph represents a spatio-temporal region of the visualized volume and is constructed by analyzing the distribution of values around its center. The edges represent transitions and indicate a change in the distribution. In this example the nodes corresponding to region (a) are highlighted in red and time evolves from the upper-right corner to the bottom-left.	11
Figure 2.3:	Luboschik et al. [32] use the heterogeneity of subsequent scales to indicate noteworthy information and highlight where drilling down into finer scales may be valuable. The approach stacks heterogeneity bands behind a line plot and uses saturated color to indicate changes between the data on subsequent scales. The green vertical line indicates the currently visualized scale.	13

Figure 3.1:	Overview of the proposed methodology. (a) The time series is embedded in the <i>embedding space</i> 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 <i>embedding space</i> . Each state represents a typical configuration of the temporal signal and summarizes the structure around it. (c) The temporal dimension is reintroduced as transitions between the states. Transitions are modeled by translating the partition into a Markov chain, with each state of the Markov chain representing a partition cell. (d) Finally, the Markov chain model is iteratively simplified by aggregating states into a hierarchy, creating a <i>hierarchical Markov chain</i> and giving a multiscale view of the data. . . .	16
Figure 3.2:	The evolution of a continuous-time Markov chain. A continuous time Markov chain models the temporal evolution of a random process that can assume a finite or countable set of states. Upon arriving into a state, the process remains constant for some amount of time, called the <i>holding time</i> , before transitioning to the next state. Continuous-time Markov chains assume exponentially distributed <i>holding times</i>	19
Figure 3.3:	When constructing the multiscale representation, we require that coarse-scale states be comprised of one or more distinct finer-scale states, creating a hierarchical partition of the <i>embedding space</i> and preventing the finer-scale states to be scattered across several coarser-scale states. . . .	21
Figure 5.1:	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 <i>icicle plot</i> . The two panels are interactively linked. States on the two panels are associated by color. The right panel visualizes properties of the currently selected state (blue border) and allows the user to rename the state and store a description.	34
Figure 5.2:	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.	35
Figure 5.3:	The <i>Time</i> 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 <i>Rainfall LOWEST</i> (dark purple) and the <i>Rainfall HIGHEST</i> (green) states of Figure 6.1 occur. We see the former typically occurs in the winter (a) and the latter in the summer (b).	37
Figure 5.4:	The decision tree describing the green <i>Rainfall HIGHEST</i> state from Figure 5.1 suggests that the state is characterized as having the average rainfall higher than 54.3mm. Each node in the decision tree visualizes the distribution of measurements belonging to the selected state versus all other states.	37

- Figure 5.5: 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 Figure 5.1 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. 38
- Figure 5.6: 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 that 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. Throughout the thesis we omit yellow 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. 39
- Figure 5.7: 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. 41
- Figure 6.1: The weather dataset. 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 on the bottom left of the figure. 44
- Figure 6.2: The *Attribute* histograms indicate that the purple *Rainfall LOWEST* and green *Rainfall HIGHEST* states in Figure 6.1(a) 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). 45
- Figure 6.3: The distribution of *Rainfall* (in mm) confirms that the states on the right of Figure 6.1(b) 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*). 46

- Figure 6.4: The traffic dataset. The traffic data exhibits cyclic patterns which suggest multiple periodic behaviors. The main states correspond to different times of day - the green states correspond to night, blue states are morning and evening and the others are daytime states. The middle scale (b) shows the daily dynamics which consists of three cyclic structures. The daily cycle starts and ends in the left green states which represent nighttime. In addition we find two smaller cycles. The first consists of the purple, magenta and red states in the center and represents morning rush hour. The second corresponds to afternoon rush hour and consists of the states on the right. 47
- Figure 6.5: The GPS dataset. The main states represent locations around the world. The coarse (a) and middle (b) scales summarize distant locations on different continents. At the finest scale (c), the dataset demonstrates several typical patterns which can be observed using StreamStory. The central state represents Slovenia, the researcher's 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. 49
- Figure 6.6: The 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 (i.e. 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. 51
- Figure 6.7: 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. 52
- Figure 6.8: A visual representation of the weather dataset generated using *Time-Curves*. The vertexes represent months while the edges indicate their temporal sequence. From the representation we clearly see a cyclic structure that we identified in Section 6.1. To a certain extent we can also compare the trajectories measured in separate years. However, we are unable to map the different parts of the cycle back to the yearly seasons. 53

- Figure 6.9: A visual representation of the traffic dataset generated using *Time-Curves*. Here the vertexes represent the hours of the day. This representation is more cluttered than the representation of the weather and it is difficult to interpret the temporal dynamics. On the other hand, we can quantitatively assess the density of the trajectories in different regions of the representation. For instance, we see that the upper-right region is much denser and much stabler than the bottom-left region. From our knowledge, we assume that the stable-most upper-right region represents the night hours while the bottom-left region represents the rush hour. 54
- Figure A.1: Screenshot of StreamStory’s initial configuration form. When starting configuration the user is required to upload a dataset in CSV format, select which attributes to use, select the attribute that represents the timestamp, select the temporal granularity (i.e. time unit) and decide whether to include attributes extracted from the timestamps of the measurements as presented in Section 4.6.2 into the feature vectors. Once this basic configuration is entered, the form is dynamic, rendering different content for different configuration options. 60
- Figure A.2: A sample of the weather dataset. The uploaded datasets are required to be in CSV format with labeled columns. One of the attributes is required to represent a numeric representation of time. This can be any millisecond time representation. 61
- Figure A.3: Attribute configuration. To configure the attributes, the user has to specify which attribute represents time, select the temporal granularity (i.e. time unit), indicate whether to include features extracted from timestamps in the embeddings, determine the type of the attributes and select for which attribute they would like to include derivative information into the feature vectors. For all options but the first we provide sensible defaults that can be used with most datasets. 62
- Figure A.4: Configuring the clustering algorithm. We support two choices of the clustering algorithm which computes the *initial states*. When choosing *k*-means (a), the user can select the number of *initial states* by specifying the parameter *k*, while specifying the maximal radius of the cluster when choosing DP-means (b). In case of DP-means, we also allow the user to enforce a lower and upper bound on the number of clusters. . . 63

List of Tables

Table 6.1: Measurements of the initialization time of a model indicate a strong dependence on the number of initial states.	55
---	----

List of Algorithms

Algorithm 4.1:	Pseudocode of the k -means algorithm	26
Algorithm 4.2:	Pseudocode of the DP-means algorithm	27
Algorithm 5.1:	Pseudocode of the attribute-based state labeling procedure	41
Algorithm 5.2:	Pseudocode to estimate the value of the cumulative distribution function from an attribute histogram	42

Abbreviations

MDS	... Multidimensional Scaling
HSL	... Hue Saturation Lightness
TF-IDF	... Term Frequency - Inverse Document Frequency
GPS	... Global Positioning System
CSV	... Comma Separated Values
JSI	... Jožef Stefan Institute
IPS	... International Postgraduate School
UK	... United Kingdom
US	... United States of America

Symbols

$\mathbb{R}^m$	... m -dimensional real-valued vector space
$\mathbb{R}^+$	... positive real numbers
$\mathbb{N}_m$	... the first m natural numbers
$\ \cdot\ $	... norm of a vector
$(X_n)_{n \geq 0}$	... an infinite sequence $X_0, X_1, \dots$
ϕ	... the embedding function
$O(\cdot)$	... the worst case asymptotic time complexity
$\Theta(\cdot)$	... the expected asymptotic time complexity
$P(X)$	... the probability of event X
Q	... the transition rate matrix of a continuous time Markov chain
Q_k	... the transition rate matrix of a continuous time Markov chain on scale k
q_{ij}	... the transition intensity between states i and j
q_i	... the negative rate of leaving state i of a Markov chain
π	... the stationary distribution of a Markov chain
Π	... a square matrix with the stationary distribution on the diagonal
S	... a set of states which a Markov chain can assume (i.e. its state space)
J_k	... a random variable which represents the times when a Markov chain switches states
H_k	... a random variable which represents the time between two jumps of a Markov chain
c_i	... the i -th centroid
$C(i)$	... the Voronoi cell around the i -th centroid

Glossary

Time series is a series of data points indexed in time order.

Multivariate time series is a time series with multiple variables.

Embedding function is a function which embeds feature vectors into another space.

Partition is a grouping of a set or space into non-empty subsets in such a way that the subsets cover the entire set but do no overlap.

Voronoi cell is a region in a metric space constructed around a seed where all the points in the cell are closer to its seed than to any other seed.

Markov chain is a memoryless random process that can assume a finite number of states.

Transition rate matrix is an array of numbers that describes the rates at which a continuous time Markov chain moves between its states.

Initial distribution is a probability distribution over the set of states of a Markov chain which determines the likelihood of each state being the first in the sequence of the observed states.

Stationary distribution is a probability distribution to which the observed distribution of states of a Markov chain converges after enough time has passed.

Eigenvector is a non-zero vector that only changes by a scalar factor during a linear transformation.

Eigenvalue is the magnitude of change of the corresponding eigenvector during the linear transformation.

k-means clustering is a randomized, iterative, centroid-based grouping algorithm that groups objects into a predefined number of groups.

DP-means clustering is an iterative, centroid-based grouping algorithm that limits the radius of each group.

Multidimensional Scaling is an algorithm for projecting a high-dimensional dataset into a lower-dimensional space by preserving the pairwise distances between the points as well as possible.

Pseudocode is an informal operating description of a computer algorithm used to illustrate its operating principle.

Ambient space is a space that embeds a mathematical object.

Chapter 1

Introduction

In this thesis we discuss a novel approach for the abstraction and visualization of temporal datasets. Specifically, we target large *multivariate time series* where state-of-the-art methods produce clutter and/or fail to highlight any meaningful patterns. We recognize that temporal patterns can appear on different spatial granularities and design a multi-scale approach supported by a hierarchical abstraction. To construct the representation, we employ several techniques from *machine learning*, *data mining* and *statistics*. Once constructed, the representation is fully interactive and presented in a web-based visualization tool called StreamStory. StreamStory uses novel visual cues to highlight relationships between the abstract elements and data-driven techniques to map the abstract elements back to domain-specific concepts and suggest possible interpretations.

In the remainder of the chapter, we first motivate our approach. We then describe the aims and hypotheses that provide ground for the thesis and drive our design choices. Finally, we provide an overview of the thesis.

1.1 Motivation

We first motivate the need for abstraction and visualization of *multivariate time series*. Time series are abundant in many scientific and engineering applications, but common solutions to visualize such datasets like 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 such step causes a context switch, stealing their focus and hindering their productivity and with it the ability to find and interpret meaningful patterns.

We adopt the definition of visual clutter first used by Rosenholtz et al. [1] and define clutter as “the state in which excess items, or their representation or organization, lead to a degradation of performance at some task”. We focus on a specific effect of clutter called *lateral masking* where extra display items cause the occlusion of relevant information and degrade the performance of the analyst. This effect is illustrated in Figure 1.1 with a sample of a real-world multivariate time series where the excess of visual items hides temporal patterns and masks the behavior of the data.

One general pattern of interest in time-varying datasets are cyclical phenomena. Examples include natural patterns such as seasons of the year as well as societal patterns like the daily cycle in traffic, electrical usage, etc. It need not be periodic, but may occur irregularly, such as increased traffic for the holidays or specific weather events like high-speed winds or especially high temperatures. As these patterns are most often not known a priori, identifying them can give us important insight into the behavior of the underlying

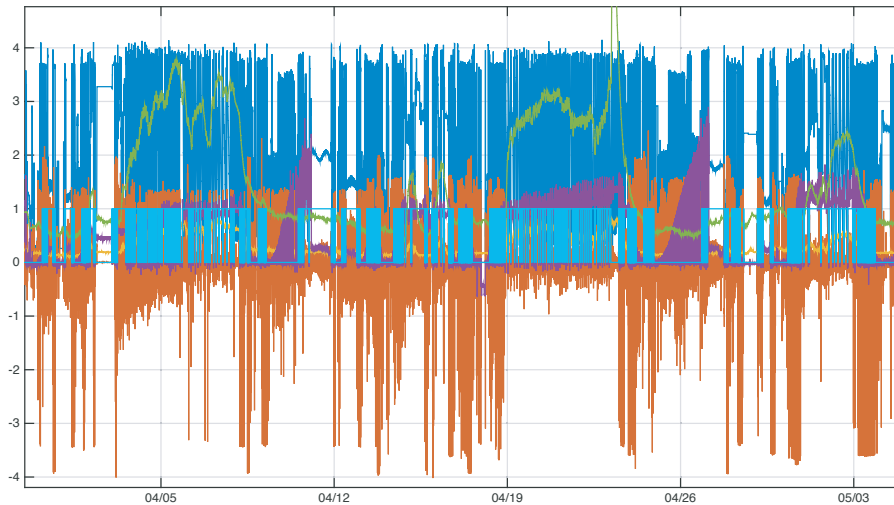


Figure 1.1: The excess of display items, or *clutter*, in the visualization hinders the exploration and analysis of a multivariate time series. To extract meaningful information, the analyst must zoom into interesting intervals, isolate informative variables and remove clutter manually. Each such step causes a context switch, stealing their focus and hindering their productivity.

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 the 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 characterized by and 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 [2], [3] (see interface in Figure 1.2). 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. It constructs the representation automatically by adapting and applying several *machine learning* techniques. To represent multiple spatial granularities, it merges finer-scale states into increasingly coarse representations creating a hierarchical partition of the *ambient space*. StreamStory en-



Figure 1.2: 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 (i.e. Germany, Portugal, Sweden, and Slovenia) as well as the US, China, and India. The panel on the right shows the distribution of the attributes inside the selected state (blue border). Here the selected state is NYC (New York City - Latitude: 40.89° , Longitude: -73.92°). This is inferred using the geographical coordinates in the right panel. The bottom panel shows a cross-scale historical overview of the data and is linked with the central panel interactively and by color. Here 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 which lasted from June to October 2014 and corresponds to a summer internship.

ables the user to interactively navigate this multiscale structure with multiple visual cues establishing cross-scale relationships. It supports several visual analytics tasks:

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

We address these tasks by designing a methodology based on the following pipeline: (1) the temporal information is disregarded and the data represented with a static embedding, (2) connected regions of the *embedding 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 a user interface which visualizes the representation at a single scale as a graph (see the central panel of Figure 1.2) 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 Figure 1.2, 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.

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 captured behavior include periodic yearly weather patterns as well as intermittent or irregular recurrences. For instance, in Figure 1.2 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.

The research presented in this thesis has the potential to make an impact on the global scale. As a general purpose tool for visualizing multivariate time series that focuses on non-experts, StreamStory can be used by researchers and engineers who analyze temporal patterns in time series from various disciplines. Some examples include meteorologists who analyze recurrent patterns in weather phenomena, engineers monitoring the production line, who analyze the temporal behavior of the factory’s equipment, for instance in relation to scrap, drilling engineers who analyze recurrent behavior in vibrations present during a drilling operation, etc.

1.2 Aims and Hypotheses

Now we present the main aims of this thesis and list the hypotheses that drive our development and influence various design choices we make when developing the proposed methodology. The main aims of the thesis are:

1. To propose a new approach for the abstraction of multivariate time series. The approach helps the user summarize the dataset qualitatively, capturing the structure of the data and retaining the long-term behavior of the data while discarding short-term variations. We recognize that datasets usually exhibit patterns at different spatial granularities (i.e. *scales*). We therefore develop a multiscale approach that retains and abstracts several distinct scales.
2. To develop and present a methodology to implement this approach. When developing our methodology, we make several design choices which enable abstracting large datasets with potentially billions of examples. The methodology first creates a hierarchical partition, effectively discretizing the *ambient space* while preserving cross-scale relationships. On top of the hierarchical partition, we construct a *hierarchical Markov chain* representation. This makes the representation compact while enabling us to compute the long-term properties of the dataset with linear algebra operations that are instantaneous given the compact representation. In contrast to most related work, we use continuous time Markov chains instead of their more popular discrete-time counterparts. This design choice enables us to remove transitions from a state back to itself in the final representation and eliminates a resampling step from the pre-processing phase. The hierarchical partition creates parent-child relationships in the representation, effectively linking states from different scales to common domain-specific concepts.
3. To instantiate the proposed methodology with efficient techniques which allow the visualization of large datasets, produce stable results and minimize parameter tuning

making the resulting system suitable for non-expert users. To support large datasets, our approach uses techniques from *statistics*, *machine learning* and *data mining*. It uses selected techniques like k -means clustering which have been tested and shown to give stable results across a wide range of datasets while still being computationally efficient and appropriate for use with large datasets. We select techniques that do not require many parameters and where unavoidable we provide sensible defaults, minimizing the required parameter tuning, making the approach appropriate for users who do not specialize in fields related to data analytics.

4. To develop and present a visual representation of the resulting abstract representation that effectively conveys qualitative information and highlights long-term, especially recurrent, behavior. The visual representation exploits the properties of the underlying Markov chain to highlight the structure and dynamics of the data in the *ambient space*, highlighting properties of the states and transitions with various visual attributes like size, thickness, color, etc. It enables the interactive traversal of the multiscale structure, remaining consistent during the interactions and using visual cues to highlight the hierarchical relationships. Using auxiliary views it helps the user understand and navigate the abstract representation and uses data-driven approaches to suggest possible interpretations of the elements of the multiscale structure. In addition to effectively conveying information, we design the visual representation to be aesthetic to attract non-expert users.
5. To rigorously evaluate the resulting system in real-world scenarios and present results in an unbiased manner. We demonstrate the results on four real-world datasets of varying complexity, showing that the system confirms the three main hypotheses of this thesis. The datasets are selected from various domains and include two datasets describing weather, a dataset of collected GPS traces and a dataset recording traffic. We evaluate our system on several research projects and gather and present feedback from three categories of users ranging from experts in data analytics, domain specialists and non-experts (e.g. workers on a factory floor). We explicitly compare our system to existing systems to demonstrate its benefits and shortcomings with respect to the state-of-the-art in visual analytics. Finally, we demonstrate the performance of the developed system. Specifically, we focus on the most time consuming, initial step when constructing the abstract representation, since once constructed, the system is fully interactive.

Our main hypotheses are:

H1: Combining techniques from *machine learning* (e.g. k -means, agglomerative clustering, etc.), *data mining* (e.g. time-delay embedding, multidimensional scaling, etc.) and *statistics* (e.g. Markov chains, maximum likelihood estimators, etc.) into a tool for data abstraction and visualization can highlight the long-term behavior of a dataset and expose recurrent patterns while mitigating cluttering effects typical when visualizing large datasets.

H2: Long-term and recurrent patterns can emerge on different spatial granularities (i.e. *scales*). Traversing the scales interactively provides an effective way to identify scales where such patterns emerge.

H2: The visual abstractions that summarize the structure and dynamics of the dataset can be mapped back to the data and interpreted as domain-specific concepts by interacting with quantitative data-driven visual tools.

1.3 Contributions to Science

Most *state-of-the-art* visualization techniques visualize multivariate time series as collections of points either on a linear or spiral temporal axis. The variates are typically visualized as several univariate time series on the same axis, using different vertical axes or in parallel views that can be organized by the analyst. However, these approaches cause clutter when visualizing larger datasets and become difficult to interpret when visualizing data with more than one variate. Some approaches eliminate the dimensionality problem by representing the data as a collection of points organized by similarity and indicating the temporal sequence as arrows between the points. Others abstract and aggregate the examples which enables them to visualize large datasets. But none of these approaches eliminate both problems mentioned above. As you will see in Chapter 2, some techniques use *machine-learning* to guide the exploration of the dataset and suggest regions of the space that may be worth inspecting. However, these are domain specific and cannot be used to investigate multivariate time series.

We design a methodology and interactive visualization tool for the abstraction and exploration of qualitative patterns in multivariate time series. Our approach does not represent time with an axis like the traditional approaches, but rather abstracts the data as qualitative elements and indicates the temporal behavior as transitions between the abstract elements creating a graph-like structure. It allows the user to interactively change the visualized scale. The tool constructs the representation automatically by applying techniques from *statistics*, *data mining* and *machine learning* and adapts several visualization techniques to help map the abstract representation back to domain specific concepts. Our main research contributions are threefold:

1. A novel methodology for multivariate time series abstraction, visualization, multiscale exploration and interpretation. The methodology is based on *hierarchical Markov chains* and captures the long-term behavior of the data through a simplified model of the dynamics as a graph structure. We augment and instantiate the methodology with specific techniques from statistics, data mining and machine learning.
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 usable 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 thus highly scalable.

1.4 Overview of the Thesis

The thesis is structured as follows. Chapter 1 motivates and introduces the proposed approach. It lists the hypotheses that drive the development of this approach and describes the main aims of the thesis. Chapter 2 presents the literature related to the proposed approach. It presents the state-of-the-art from several fields related to various aspects of our approach, mainly focusing on visualization and visual analytics systems and approaches. It provides a rigorous comparison to other state-of-the-art techniques.

Next Chapter 3 presents the proposed methodology designed to construct the abstraction of multivariate time series. The methodology consists of four steps and abstracts large datasets in a computationally efficient manner. Chapter 4 instantiates the methodology, populating the steps with specific techniques from machine learning and data mining. We use efficient techniques making the system capable of abstracting large datasets. We select the algorithms that minimize the configuration required out-of-the-box while remaining efficient and stable across various inputs.

Then, Chapter 5 presents the visual representation developed on top of the proposed methodology. This system is novel in itself. It combines several visual cues to highlight various aspects of the abstractions while remaining visually pleasing (i.e. aesthetic).

We then evaluate the main hypotheses of this thesis in Chapter 6. We use real-world datasets and provide feedback from users in two stages of development. We compare our approach to related state-of-the-art techniques explicitly in Section 6.6 and evaluate the computational aspects of the resulting system in Section 6.7. Finally, we conclude the thesis with a summary and ideas for further work in Chapter 7.

Chapter 2

Related Work

In this chapter, we discuss other approaches that are related to StreamStory, some of them were leveraged to develop StreamStory. Since StreamStory is primarily a tool for multiscale visualization of multivariate time series, we focus on tools and techniques for visualization. We discuss techniques from several sub-fields of visualization including general approaches to visualizing time series, approaches for visualizing multivariate time series with a temporal axis, static approaches for visualizing time series and cluster visualization. We also discuss modeling approaches related to StreamStory including several approaches to model data using Markov chains and approaches to create multiscale representations of data. These techniques are discussed in the following subsections.

2.1 General Approaches

The literature suggests many approaches for visualizing time series. Examples include *point plots*, *line plots*, *parallel bar charts*, and *sparklines* which use a linear time axis and draw the values of the variates on the same or parallel views. Other techniques like *spiral graphs* and *spiral displays* emphasize periodic behavior by curving the time axis into a spiral and mapping the data onto this spiral, visualizing quantitative data with color, line thickness or texture. For an overview, we refer the reader to [4], [5]. These techniques, however, become cluttered when visualizing large datasets and do not highlight the interactions between the variables.

Shneiderman et al. [6] visualize multiple time series 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.

2.2 Visualizations with a Linear Temporal Axis

The patterns of interest often lie in the interaction between the variables and the relationships are lost when visualizing each variable in a separate view. In their work, Havre et al. [7] visualize thematic variations in a text document collection over time as a dynamic *Stacked Graph*. The approach uses a linear time axis and encodes the values of the attributes as in a dynamic stack that changes as time evolves. Later, Byron and Wattenberg [8] extend the approach using a different color coding and a novel layout algorithm

in an approach called the *Streamgraph* that is shown in Figure 2.1. A drawback of these techniques is that they can only visualize variates with a positive domain and it can be difficult to identify recurrent behavior.

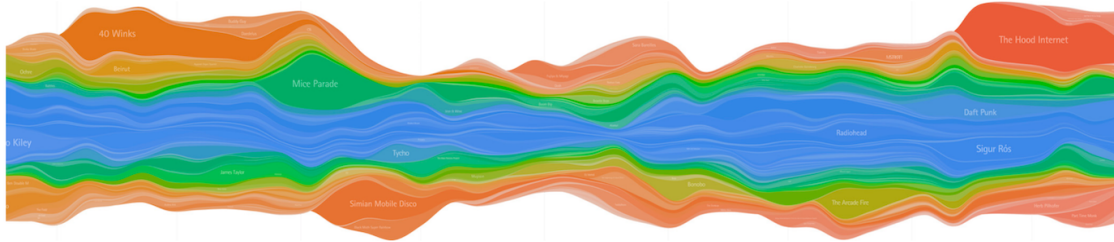


Figure 2.1: A visual representation of an individual’s music listening trends with a *Streamgraph* [8]. The x -axis represents time and each horizontal stripe represents an artist. The thickness of a stripe visualizes the number of times that their songs were played in a given week.

In a different approach, Peng et al. [9] 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. [10] 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. [11] 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 the number of examples.

2.3 Visualizations with a Curved Temporal Axis

In a different approach, Haroz et al. present *connected scatterplots* [12]. *Connected scatterplots* visualize two attributes in a scatterplot indicating the temporal direction using arrows. In a more generic approach, Bach et al. present *TimeCurves* [13] which warp the temporal axis so that similar points appear close to each other. For an illustration of *TimeCurves* see Figure 6.8 and Figure 6.9. By disregarding all but similarity information *TimeCurves* visualize 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.

2.4 Static Graph-Based Visualizations

Recent years have seen the development of static graph-based approaches to visualize temporal data. Jänicke and Scheuermann use ϵ -machines to visualize a dynamic field in a static manner [14]. 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 [15] visualize temporal volumetric data using a graph-based visualization called *TransGraph* shown in Figure 2.2. 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 Multidimensional Scaling (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 the data at fixed locations.

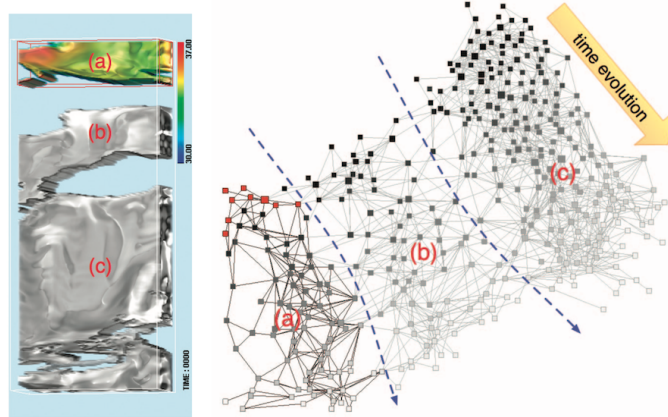


Figure 2.2: A volumetric climate dataset visualized using *TransGraph*. Each node in the graph represents a spatio-temporal region of the visualized volume and is constructed by analyzing the distribution of values around its center. The edges represent transitions and indicate a change in the distribution. In this example the nodes corresponding to region (a) are highlighted in red and time evolves from the upper-right corner to the bottom-left.

By modeling the trajectory of the data through regions of the *ambient space*, Pylvänen et al. [16] describe an approach similar to our own. Applying *k*-means [17], 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.

2.5 Cluster Visualization

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 onto a lower-dimensional space [18], [19], using multidimensional projection to approximate the structure of the clusters relative to each other, using visual attributes to describe the shape of a state [20] and using auxiliary widgets to help in the interpretation of the states [21], [22]. 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 the regions of interest as well as interpret the graph-based representation.

Trampuš et al. [23], [24] highlight the diversity in public news by clustering textual documents, drawing them on a plane and allowing the user to interactively explore the corpus by re-ranking a list of documents based on the position of a focus point relative to the clusters. Like StreamStory, their approach clusters the items and projects them onto a plane using MDS. To visualize the groups, they extract their most characteristic keywords by comparing their TF-IDF weight and represent the group as a tag cloud. Similarly, StreamStory uses quantitative techniques to generate state labels. StreamStory extends the above approach in two directions. First, by modeling the data using a Markov chain, it adds the temporal dimension and visualizes the dynamics alongside the structure. Second, it constructs a multi-scale representation to allow the user to interactively find patterns that appear on different spatial granularities.

2.6 Markov Chains to Model Temporal Datasets

There are several approaches for using Markov chains to model time series. In the univariate case [25]–[27] this involves discretizing the range of the data before modeling transitions. For multivariate data, Ching et al. [28]–[30] 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 a high-dimensional *ambient 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.

The main advantage of using Markov models is their simplicity, ease to compute and store. When modeling data with a Markov chain, the long-term (i.e. *ergodic*) properties of the model can be computed with linear algebra operations. Their main shortcoming is that they are memoryless, i.e. given the current state, the model has no knowledge of its history and cannot use it to predict future developments. Furthermore, simplifying the model into a finite set of states can discard information about the modeled system.

2.7 Multiscale Representations

We create a *multiscale* representation to allow users to interactively select informative scales. The literature suggests other *multiscale* approaches. Woodring et al. [31] 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, effectively grouping time series with similar frequency bands. The clusters are visualized in spreadsheet format with a separate view for details. The spreadsheet format can be overwhelming and is not appropriate for users who do not specialize in data analysis.

Luboschik et al. [32] use the heterogeneity of subsequent scales as an indicator of noteworthy information and highlight where a drill-down to finer scales may be valuable. Their approach is shown in Figure 2.3. While we use a different indicator to determine interesting scales, this type of function can be incorporated into the system.

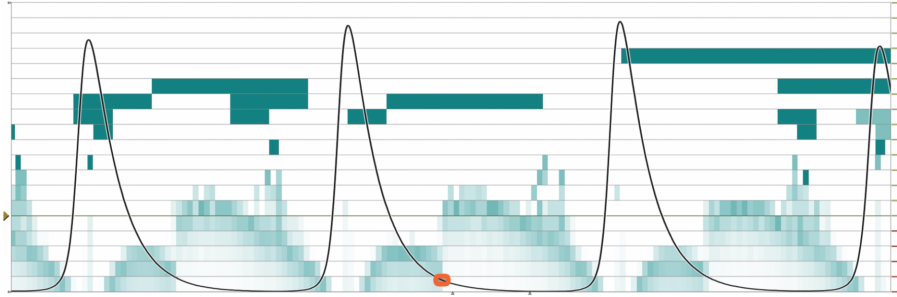


Figure 2.3: Luboschik et al. [32] use the heterogeneity of subsequent scales to indicate noteworthy information and highlight where drilling down into finer scales may be valuable. The approach stacks heterogeneity bands behind a line plot and uses saturated color to indicate changes between the data on subsequent scales. The green vertical line indicates the currently visualized scale.

Auber et al. [33] 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. [34] describe a generic framework for visualizing multiscale datasets which replaces the linear zoom with a graph-based zoom path. In their framework, each node in the graph represents a specific visual and data abstraction. An analyst can independently zoom on either axis to independently change the level of detail in the data abstractions or visual abstractions. We use a linear zoom, as this provides the simplest interpretation for non-expert users.

2.8 Machine-Guided Visualizations

Recent decades have seen the development of machine-guided visualization techniques where the computer helps the user to explore the data by guiding them through the data space and highlighting interesting patterns and regions.

In their work Leban et al. [35] use machine learning to find projections that best separate data with different class values on a plane and are thus visually informative. By selecting pairs of attributes used as the basis for the projection, the method generates a list of most informative projections as computed numerically by evaluating the performance of a classifier trained on the projected data. Doing so, the method saves the analyst from searching through $\binom{d}{2}$ pairs of attributes when visualizing a dataset. However, the method only works on data with a class attribute and does not visualize any temporal information.

Fortuna et al. [36] present a technique which uses techniques from machine learning and text mining to guide the user in creating an ontology from a textual corpus. The technique visualizes the ontology as a tree structure augmented by several views. It suggests possible new concepts by employing machine learning and allows the user to adjust the properties of the ontology interactively. However, the technique is designed to model textual corpora and is not suitable for visualizing multivariate time series.

Overall, we re-use and build upon several ideas from the existing literature when developing StreamStory. We visualize the structure of the dataset by clustering the data,

projecting a representative of each cluster onto a plane and using visual cues to visualize the properties of the clusters, extending the existing literature by generating qualitative text-based descriptions of the clusters in the form of automatic state labels. We use a multi-scale representation to allow the user to investigate the patterns on several spatial granularities interactively. We use Markov chains to model the dynamics of the data and, similarly to *TimeCurves*, use a graph-based representation, indicating the temporal dimension using edges between the associated nodes. In contrast to *TimeCurves*, in our representation the nodes represent the aggregation of several distinct measurements making recurrences appear as cycles in the diagram and enabling us to visualize large datasets. We describe these techniques in more detail in subsequent chapters.

Chapter 3

Methodology

In this chapter we define the proposed theoretical methodology to abstract multivariate time series. We define the required steps, their input and output as well as the theoretical requirements for each step. We begin with an overview of the methodology and continue with a detailed description of each individual step.

3.1 Overview

The proposed methodology enables interactive multiscale visualization, exploration and interpretation of large multivariate time series. Our goal is to identify recurrent patterns in time series. This type of behavior 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 into finer-scale states like “morning”, “afternoon”, “evening” and “night” with the associated transitions following the daily cycle. Such characterizations abstract the data, hiding local variations and providing 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 partitioning 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, however, remains a difficult step and in Chapter 5 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 that allow the user to investigate the individual states and transitions (Chapter 5). Further, in any discretization, the choice of scales affects which behaviors 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 shown in Figure 3.1. These are: (a) embedding the measurements of the time series in the *embedding space*, representing the data as a point cloud, (b) constructing the states by partitioning the *embedding space*, (c) modeling the temporal dynamics as transitions between the states and (d) aggregating states and transition into a hierarchy to obtain a multiscale representation. We describe these steps in the following sections.

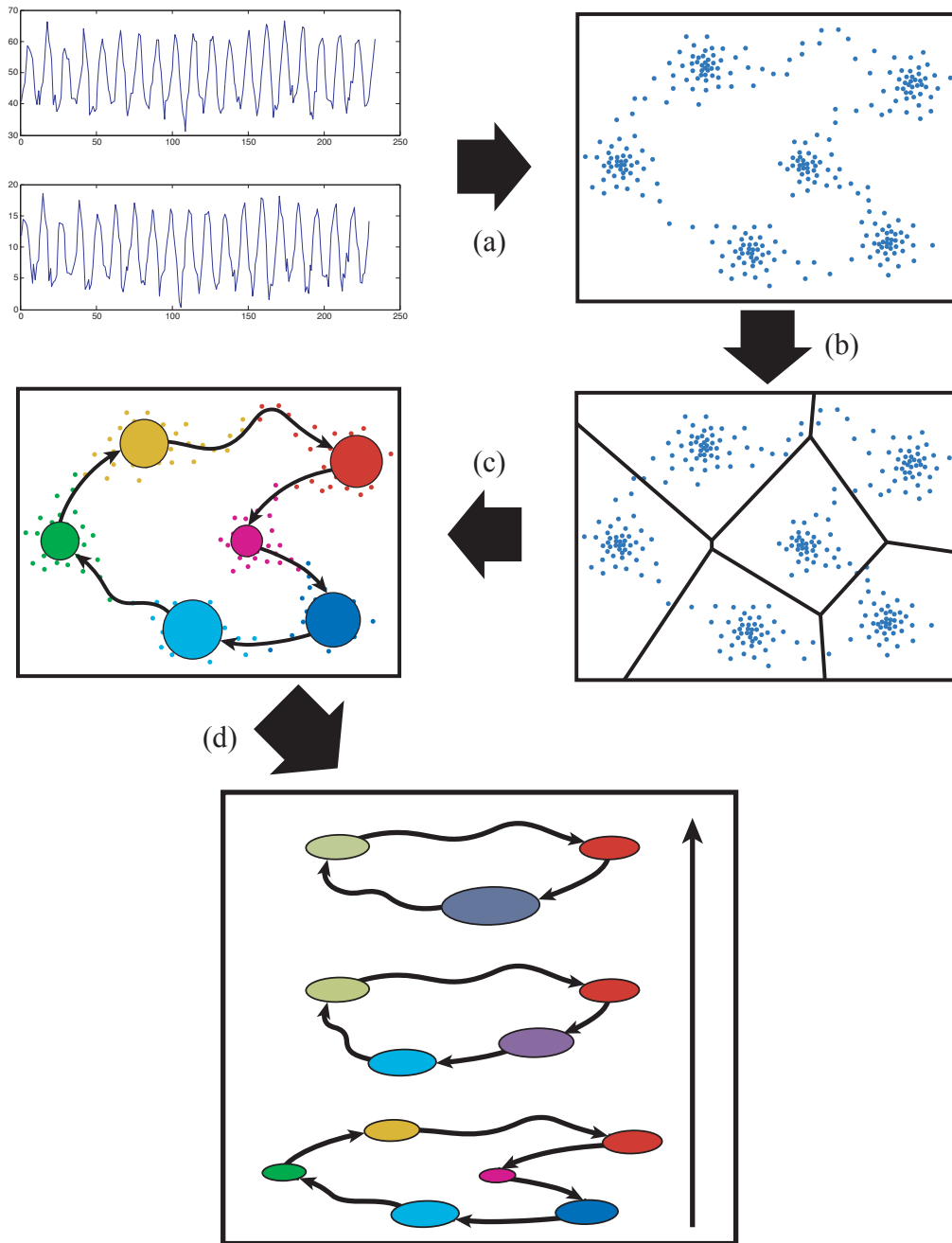


Figure 3.1: Overview of the proposed methodology. (a) The time series is embedded in the *embedding space* as a point cloud. As an example, we show two noisy approximately periodic signals mapped to points in 2D. (b) The states are constructed by partitioning the *embedding space*. Each state represents a typical configuration of the temporal signal and summarizes the structure around it. (c) The temporal dimension is reintroduced as transitions between the states. Transitions are modeled by translating the partition into a Markov chain, with each state of the Markov chain representing a partition cell. (d) Finally, the Markov chain model is iteratively simplified by aggregating states into a hierarchy, creating a *hierarchical Markov chain* and giving a multiscale view of the data.

3.2 Time Series Embedding

The first step of the methodology captures the spatial structure of the data by embedding the time series in an m -dimensional *embedding space* as a point cloud. We define a time series as a sequence of tuples $(t_0, x_0), (t_1, x_1), \dots, (t_i, x_i), \dots$ where $t_i \in \mathbb{R}^+$ is a timestamp and $x_i \in \mathbb{R}^d$ represents the measurement taken at time t_i . A point cloud is a set of data points in space. They are commonly used in machine learning, with each measurement (i.e. data associated to one timestamp) represented by a feature vector.

The *embedding space* is a metric space $(\mathbb{R}^m, d)$ where $d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^+$ is a metric. We define the embedding function $f : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^{m_1} \times \mathbb{R}^{m_2} \times \mathbb{R}^{m_3}$ as:

$$f(t_i, x_i) = (g(x_i), \tau(t_i), h(x_1, x_2, \dots, x_{i-1})), \quad (3.1)$$

where t_i is the timestamp of the i -th measurement, x_i is the measurement associated with the timestamp and $m = m_1 + m_2 + m_3$ is the dimension of the *embedding space*. The functions g , τ and h are used to add information into the feature vector. The static embedding function g adds information about the measurement, function τ adds temporal information and function h adds information about the past trajectory of the time series.

In our framework, we require that the static embedding function g is invertible and preserves local neighborhoods. This ensures that similar points are not scattered to distant regions of the *embedding space* and that the abstractions can be mapped back to the data and interpreted as domain-specific concepts. More formally, we require $g^{-1}(y)$ be defined for all $y \in \mathbb{R}^m$ and for each $\epsilon \geq 0$ that there exists $\delta \geq 0$ such that for all $x, y \in \mathbb{R}^d$ such that $\|x - y\| < \epsilon$ it follows that $\|g(x) - g(y)\| < \delta$. We discuss two approaches to construct the embedding function in Section 4.6.

The output of the first step is a set of points $\{y_1, y_2, \dots, y_n\}$ where $y_i = f(t_i, x_i)$, embedded in the *embedding space*.

3.3 Constructing the States

We next discretize the *embedding space* into a set of non-overlapping regions which we associate with conceptual states (Figure 3.1(b)). Each state represents a typical configuration of the temporal signal and summarizes the structure around it. This is essential for visualizing large datasets as this abstraction reduces the local variation, removing clutter from the final representation.

To construct the state-set, the points in the *embedding space* should be partitioned based on some metric $d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^+$. Partitioning divides the embedding space into one or more disjoint subsets or regions. A partition should contain points that lie close to each other based on d . Each partition is associated with a conceptual state. For example in Figure 3.1, two noisy periodic signals are partitioned into six states. States are connected, non-empty subsets of the embedding space (see Definition 3.1). We do not enforce any specific metric in this step, but note that to aid interpretation and avoid confusing and overwhelming non-expert users, the same metric should be used in all steps of the methodology.

Definition 3.1. Let V be a vector space and let $d : V \times V \rightarrow \mathbb{R}^+$ be a metric defined on the elements of V . A set $S \subset V$ is a state in V if it satisfies the following conditions:

1. S is non-empty
2. S is connected with respect to d .

A state-set $\mathbb{S}$ is a partition $\mathbb{S} = \{S_i\}_{i \in \mathbb{N}_n}$ where each element S_i is a state.

The output of the second step is a matrix of centroids $C \in \mathbb{R}^{m \times k}$, where each column c_i represents a centroid in the *embedding space*. Each centroid serves as a representative of a state and summarizes all the measurements in the state. Every point y_i in the *embedding space* is assigned to the state j whose centroid c_j minimizes the formula below.

$$j = \arg \min_{j \in \mathbb{N}_k} d(y_i, c_j) \quad (3.2)$$

As our system is aimed at users who do not have expertise in data analytics, this step should use methods which are robust to parameter settings to avoid parameter tuning. Since this is the most time-consuming step of the methodology, efficient partitioning algorithms should be used.

3.4 Modeling Transitions

Once the states are constructed, we model the dynamics as transitions between them (Figure 3.1(c)). We refer to these as the *initial states* and denote them by S . 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*.

We assume that the sequence of states $(X_n)_{n \geq 0}$ 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, called the *holding time*. We assume exponentially distributed *holding times* and model the process as a *continuous-time Markov chain* [37]. 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 [38]. Furthermore, time series are often sampled from continuous processes making the assumption of continuous time more natural.

Before describing continuous time Markov chains, we describe their discrete time counterparts. A discrete-time Markov chain $(X_n)_{n \geq 0}$ is a sequence of random variables $X_0, X_1, \dots$ which can assume a finite or countable set of states S . In each time step, the process either jumps from its current state $i \in S$ to state $j \in S$ with probability p_{ij} or remains in state i with probability $p_i = p_{ii}$. Formally, we say that the discrete-time random process $(X_n)_{n \geq 0}$ is a discrete-time Markov chain with state space S and initial distribution λ if and only if the following conditions are met [37]:

1. The initial state X_0 has distribution λ .
2. Conditional on $X_n = i$, the next state X_{n+1} has distribution $(p_{ij})_{j \in S}$ and is independent of $X_0, X_1, \dots, X_{n-1}$.

The second condition is called the *Markov property*. Intuitively, it states that given the current state, the sequence of states that the process has visited has no influence over where it will go in the future.

The main difference between continuous-time Markov chains and their discrete-time counterparts is that continuous-time Markov chains do not make transitions at discrete time steps, but can transition at any time, possibly making an infinite number of transitions in a finite time interval, a phenomena called an explosion. A continuous time Markov chain $(X_t)_{t \geq 0}$ with values in S is a family of random variables $X_t : \Omega \rightarrow S$. Every path $t \mapsto X_t(\omega)$ must remain constant for a while in each new state. We call the times $J_0, J_1, \dots$ when $(X_t)_{t \geq 0}$ transitions from one state to another the *jump times*. The time that the process spends in a state upon arrival is called the *holding time* and is denoted by $H_k = J_{k+1} - J_k$.

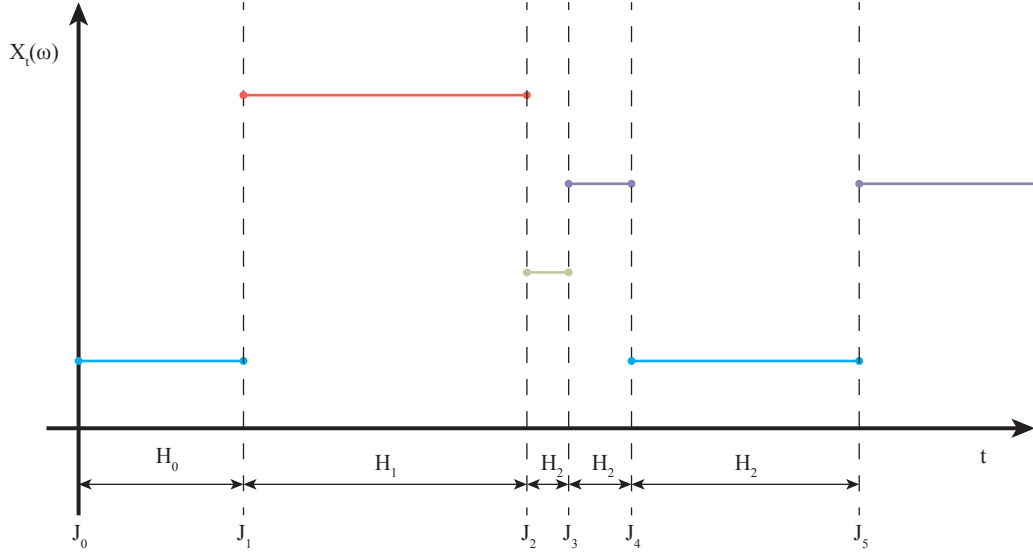


Figure 3.2: The evolution of a continuous-time Markov chain. A continuous time Markov chain models the temporal evolution of a random process that can assume a finite or countable set of states. Upon arriving into a state, the process remains constant for some amount of time, called the *holding time*, before transitioning to the next state. Continuous-time Markov chains assume exponentially distributed *holding times*.

The evolution of a continuous time Markov chain along with the *jump times* and *holding times* is illustrated in Figure 3.2.

A continuous time Markov chain is represented by two structures: (i) the transition rate matrix $Q \in \mathbb{R}^{m \times m}$ and (ii) the initial distribution $\lambda \in \mathbb{R}^m$. In both cases, m represents the number of states in S which in our case is the number of *initial states*. The non-diagonal elements of Q , q_{ij} describe the rate of transitioning from state i to state j in the number of jumps per time unit, while the negative diagonal elements $q_i = -q_{ii}$ describe the rate of leaving state i . Every continuous-time Markov chain $(X_t)_{t \geq 0}$ has an associated *jump chain* $(Y_n)_{n \geq 0}$ which is a discrete-time Markov chain that models only the sequence of jumps, disregarding the *holding times*.

In this thesis we define a continuous time Markov chain in terms of its *jump chain* and *holding times*. A continuous time random process is a Markov chain if its *jump chain* is a discrete time Markov chain and its *holding times* are exponentially distributed. For a formal definition see Definition 3.2.

Definition 3.2. Let $(X_t)_{t \geq 0}$ be a continuous time random process that assumes a countable set of states S and let λ be a probability distribution on S . We say that $(X_t)_{t \geq 0}$ is a *continuous-time Markov chain* if and only if it satisfies the following conditions:

1. its *jump chain* $(Y_n)_{n \geq 0}$ is a discrete time Markov chain with initial distribution λ and
2. conditional on $Y_0 = i_0, Y_1 = i_1, \dots, Y_{n-1} = i_{n-1}$, its holding times $H_0, H_1, \dots, H_{n-1}$ are independent exponentially distributed random variables with parameters $q_{i_0}, q_{i_1}, \dots, q_{i_{n-1}}$.

In this step of the methodology, we make two assumptions on the computed Markov chain $(X_t)_{t \geq 0}$. First, we assume that $(X_t)_{t \geq 0}$ is *non-explosive*. This means that there can never be a situation where the chain will make an infinite number of jumps in a finite time interval. This assumption is always satisfied in real-world datasets. Second, we assume that $(X_t)_{t \geq 0}$ is *recurrent*. The second condition states that there must exist a path with

positive probability between every pair of states in S . While the second assumption can be violated, our experience shows it does not pose problems in practice. In fact the second assumption can only be violated with extraordinary starting conditions, where the process starts in a state to which it never returns again or ends in an anomalous state. These conditions can easily be avoided in a pre-processing step by ignoring the first and/or last part of the dataset.

One property we are often interested in when discussing Markov chains is the stationary distribution which we denote with π . The stationary distribution is strongly linked to the ergodic properties of the Markov chain. For each state i , its corresponding element in the stationary distribution π_i describes the proportion of time that the process spends in the state when running indefinitely. That is for each state i , the probability of the process being in state i at time t converges to π_i :

$$P(X_t = i | X_0 = j) \xrightarrow{t \rightarrow \infty} \pi_i, \quad (3.3)$$

where X_t is the state which the process assumes at time t and X_0 is the starting state. This probability converges to the associated entry in the stationary distribution independently of the starting state. In case of continuous time Markov chains, the stationary distribution is computed as the left eigenvector associated with eigenvalue 0 as:

$$\pi Q = 0 \quad (3.4)$$

where π is the non-trivial solution to the above equation.

The output of this step is the matrix of centroids computed in Section 3.3 and the transition rate matrix $Q \in \mathbb{R}^m$ of a continuous time Markov chain. We omit the *initial distribution* needed to completely define the Markov chain since we are only interested in its ergodic properties which in the case of *recurrent* and *non-explosive* Markov chains are independent of the starting state.

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 behavior which can convey the corresponding patterns effectively. It would be possible to use more complex models, e.g. more complex transition models and holding times, but it would significantly increase the computational cost of constructing and interacting with the model and/or its memory footprint. 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 Chapter 5).

3.5 Constructing the Multiscale Hierarchy

The final step constructs the multiscale representation by aggregating the states and transitions into a hierarchy. In this step, we construct a hierarchy of Markov chains (Figure 3.1(d)) and associate each level of the hierarchy with a scale in the final visualization.

First we construct a hierarchical partition of the *embedding space*. We require that coarser-scale states be comprised of one or more distinct finer scale states. Equivalently, the lower-scale states should be partitions of coarser-scale states resulting in a *hierarchical partition*. This ensures consistency when traversing the scales as it prevents an *initial state* from being scattered into several coarser-scale states. We illustrate the idea in Figure 3.3. In the figure, we see two *initial states* (i.e. magenta and blue) merged into a state on the middle scale. The cyan state skips the middle scale and is only merged on the coarsest scale.

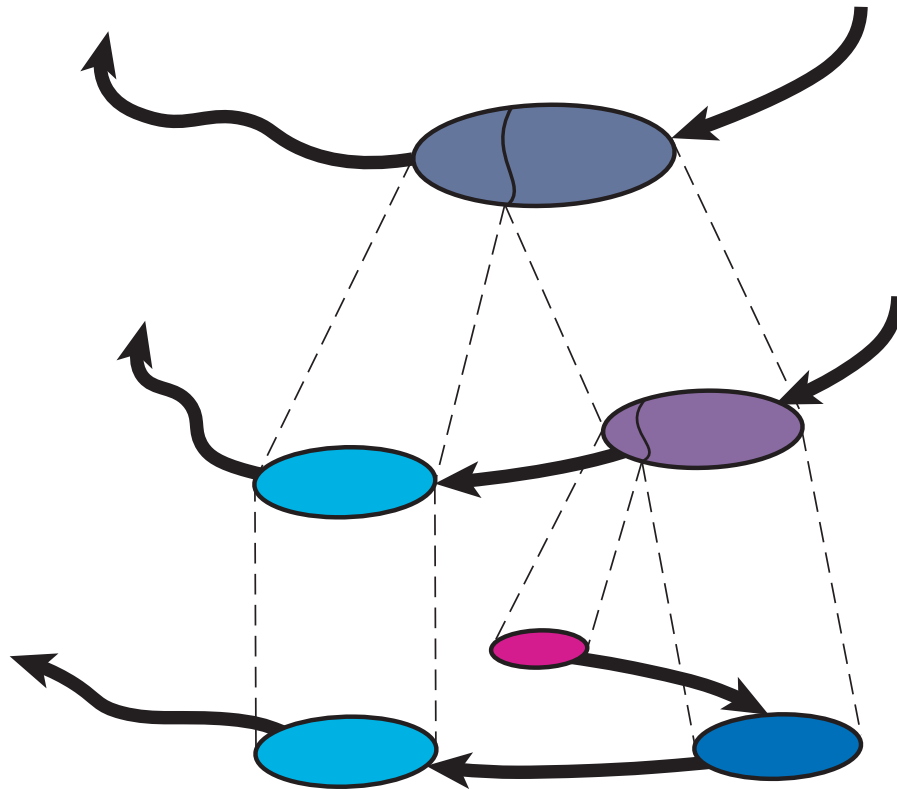


Figure 3.3: When constructing the multiscale representation, we require that coarse-scale states be comprised of one or more distinct finer-scale states, creating a hierarchical partition of the *embedding space* and preventing the finer-scale states to be scattered across several coarser-scale states.

We then model the transition rates on coarser scales by summing the weighted transition rates between the *initial states*, preserving the ergodic properties of the original Markov chain by using entries from its stationary distribution π as weights. In Theorems 3.1 and 3.2 we derive an explicit formula for calculating the transition rate between two coarse scale states. Intuitively, the formula calculates the transition rate as the weighted average of transition rates between the associated *initial states*, using entries of the stationary distribution as weights.

Theorem 3.1. *Let $(X_n)_{n \geq 0}$ be a discrete-time Markov chain with transition matrix P that can assume a countable set of states S and let $p_{ij} = (P)_{ij}$ be the probability of a transition from state i to state j . Let A and B be non-overlapping subsets of S . Then the probability of $(X_n)_{n \geq 0}$ transitioning from the states in A to the states in B converges to p_{AB} as $n \rightarrow \infty$, where p_{AB} is defined as:*

$$p_{AB} = \frac{\sum_{i \in A} \pi_i \sum_{j \in B} p_{ij}}{\sum_{i \in A} \pi_i}.$$

Proof. Let A and B be non-overlapping subsets of S . Then the probability of $(X_n)_{n \geq 0}$ transitioning from the states in A to the states in B is:

$$\begin{aligned} p_{AB} &= P(X_{n+1} \in B | X_n \in A) \\ &= \sum_{j \in B} P(X_{n+1} = j | X_n \in A) \\ &= \sum_{j \in B} \frac{P(X_n \in A | X_{n+1} = j) P(X_{n+1} = j)}{P(X_n \in A)} \\ &= \sum_{j \in B} \frac{\sum_{i \in A} P(X_n = i | X_{n+1} = j) P(X_{n+1} = j)}{\sum_{i \in A} P(X_n = i)} \\ &= \sum_{j \in B} \frac{\sum_{i \in A} P(X_n = i) p_{ij}}{\sum_{i \in A} P(X_n = i)} \xrightarrow{n \rightarrow \infty} \frac{\sum_{i \in A} \pi_i \sum_{j \in B} p_{ij}}{\sum_{i \in A} \pi_i} \end{aligned}$$

□

Theorem 3.2. *Let $(X_t)_{t \geq 0}$ be a continuous-time Markov chain with transition rate matrix Q that assumes states in a countable set S . Let q_{ij} be the transition rate from state i to state j and let A and B be two non-overlapping subsets of S . Then the transition rate between the sets A and B converges to q_{AB} as $t \rightarrow \infty$, where q_{AB} is defined as:*

$$q_{AB} = \frac{\sum_{i \in A} \pi_i \sum_{j \in B} q_{ij}}{\sum_{i \in A} \pi_i}. \quad (3.5)$$

Proof. Let $(Z_n^{(h)})_{n \geq 0}$ be a discrete-time Markov chain, with a transition rate matrix $P(h)$, obtained by sampling $(X_t)_{t \geq 0}$ at times $0, h, 2h, \dots$. By definition, we know that $P(h) = e^{hQ}$ and that $Q = \lim_{h \searrow 0} \frac{\partial}{\partial h} P(h)$. Let A and B be non-overlapping subsets of S . Then we have

$$\begin{aligned} q_{AB} &= \lim_{h \searrow 0} \frac{\partial}{\partial h} p_{AB}(h) \\ &= \lim_{h \searrow 0} \frac{\sum_{i \in A} \pi_i \sum_{j \in B} \frac{\partial}{\partial h} p_{ij}(h)}{\sum_{i \in A} \pi_i}. \end{aligned}$$

As $h \searrow 0$ the transition probability p_{ij} becomes $p_{ij} \approx \delta_{ij} + q_{ij}h + o(h^2)$. Thus we have

$$\begin{aligned} q_{AB} &= \lim_{h \searrow 0} \frac{\sum_{i \in A} \pi_i \sum_{j \in B} \frac{\partial}{\partial h} (\delta_{ij} + q_{ij}h + o(h^2))}{\sum_{i \in A} \pi_i} \\ &= \lim_{h \searrow 0} \frac{\sum_{i \in A} \pi_i \sum_{j \in B} (q_{ij} + o(h))}{\sum_{i \in A} \pi_i} \\ &= \frac{\sum_{i \in A} \pi_i \sum_{j \in B} q_{ij}}{\sum_{i \in A} \pi_i} \end{aligned}$$

□

To write Equation 3.5 in matrix form we first encode the state aggregation rules into a matrix $P_k \in \{0, 1\}^{m \times m_k}$. Specifically, the *initial state* j should be merged into state i on scale k if and only if the (j, i) -th entry of P_k equals 1. To compute the Markov chain on scale k , we then calculate its transition rate matrix Q_k as:

$$Q_k = (P_k^T \Pi P_k)^{-1} P_k^T \Pi Q P_k. \quad (3.6)$$

In the formula above, Q represents the transition rate matrix between the *initial states*, P_k is the aggregation matrix for scale k and $\Pi = \text{diag}(\pi)$ is a matrix with the stationary distribution of Q on the diagonal. Similarly, one could compute the initial distribution as:

$$\lambda_k = \lambda P_k, \quad (3.7)$$

summing the entries of the states which are merged.

The centroid of a coarse scale state is computed as the weighted average of the centroids of the associated initial states. Like above, we use entries in the stationary distribution as weights and compute the centroid matrix C_k for scale k as:

$$C_k = C \Pi P_k (P^T \Pi P)^{-1}. \quad (3.8)$$

This ensures that states where the process spends more time move less when traversing the scales.

This completes the description of the theoretical methodology that constructs our abstract data representation. We continue to describe the specific algorithms and design choices made in the implementation of StreamStory in Chapter 4 and move on to describe its visual representation in Chapter 5.

Chapter 4

Instantiating the Methodology

4.1 Overview

In this chapter, we instantiate the methodology described in Chapter 3 by populating it with specific algorithms and describing the specific design choices and implementation details. We place special emphasis on designing a scalable solution, capable of abstracting large datasets. At the same time, we minimize the expected effort spent on parameter tuning as this would discourage non-expert users from getting familiar with the tool.

We follow the structure of Chapter 3, presenting the instantiation of each of the four steps individually. We conclude the chapter by presenting two techniques from *data mining* which help highlight recurrent behavior.

4.2 Time Series Embedding

The first step of the methodology represents data statically by embedding the measurements in a high-dimensional *embedding space* as a point cloud. The measurements are embedded with an embedding function f which consists of three components: (i) the static embedding function $g(x)$, (ii) the time embedding function $\tau(t)$ and (iii) the history embedding function h . Recall from Section 3.2 that to support mapping the abstractions back to domain-specific concepts, we enforce two requirements on the static embedding function $g : \mathbb{R}^d \rightarrow \mathbb{R}^m$: (a) we require that g be invertible over its codomain and (b) we require g to preserve local neighborhoods. The latter prevents similar measurements from being dispersed during the transformation, preserving the topology of the states. In our prototype implementation, we support three embedding functions.

By default, StreamStory embeds the measurements by disregarding the temporal and historic information. Specifically, we use the identity $g(x) = x$ static embedding function and use $\tau(t) = \emptyset$ and $h(x_1, x_2, \dots, x_{i-1}) = \emptyset$. The identity function satisfies both conditions defined in Section 3.3. First, it is invertible for every $x \in \mathbb{R}^m$, and second, it preserves local neighborhoods. While there are several other options, we choose the identity function because it provides the most intuitive mapping between the abstract states back to domain-specific concepts. Indeed, to interpret the states, no transformation is required.

When configuring a dataset (see Appendix A), we also allow the user to select specific attributes which are not used in the representation and omitted in further steps of the methodology, but can still be viewed in the final representation. Superimposing attributes which are not used in building the representation is useful for studying correlations between the attributes.

The system supports two other embedding functions discussed in Section 4.6, both designed to highlight recurrent behavior. One adds information about the local dynamics,

appending derivative information to the feature vector by defining function h . The other appends static temporal information by defining τ , thus highlighting certain periodicities.

4.3 Constructing the States

The next step imposes a metric on the *embedding space* and partitions it, associating the partitions with conceptual states. Each partition is represented by a centroid which summarizes the structure of the data around it.

While any metric can be used, we typically use the Euclidean distance as we find it generally works well with data of medium dimensionality and is intuitive for most engineers. Before clustering, we normalize each dimension by its standard deviation. This helps mitigate incomparable scaling for the multivariate time series.

To construct the partition, we use a clustering algorithm. At the time of writing the system supports k -means and DP-means [39] to compute the centroids and construct the partition, but any clustering technique can be used. Both algorithms construct the partition as a set of Voronoi cells $\{C(i)\}_{i \in \mathbb{N}_k}$ around a set of centroids $\{c_1, c_2, \dots, c_k\}$. A Voronoi cell around a centroid c_i contains all the points which are closer to c_i than any other centroid:

$$C(i) = \{x \in \mathbb{R}^m \mid \|x, c_i\| \leq \|x, c_j\| \ \forall j \in \mathbb{N}_k\}. \quad (4.1)$$

Both algorithms are based on an iterative approach which partitions a set of examples $x_1, x_2, \dots, x_n$ into a set of clusters. The clusters are represented with centroid vectors $c_j \in \mathbb{R}^m$ which are computed as the mean of all the measurements in the corresponding cell. For k -means, the number of clusters k is a parameter and is fixed. The algorithm is presented in Algorithm 4.1.

Algorithm 4.1: Pseudocode of the k -means algorithm

Data: a set of measurements $X = \{x_1, x_2, \dots, x_n\}$

a metric $d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^+$

Result: a set of centroids C

Set initial k centroids $C = \{c_1, c_2, \dots, c_k\}$

repeat

 Assign each example $x_i \in X$ to cluster C_j that minimizes $\arg \min_{j \in \mathbb{N}_k} d(x_i, c_j)$

 Update the centroids $c_j = \frac{1}{\|C_j\|} \sum_{x \in C_j} x$

until the assignment of examples to centroids has not changed

The k -means algorithm first selects a set of centroids C . In each iteration it then assigns each measurement x_i to the cluster or cell C_j with the nearest centroid c_j (i.e. the cluster that minimizes $d(x_i, c_j)$ for the selected metric). The procedure runs until the assignment of measurements to clusters remains the same in two consecutive iterations. There are various approaches to select the initial set of centroids and initialize the algorithm. The simplest approach is to select the initial set of centroids by randomly sampling the measurements. However, such an approach can result in slow convergence or convergence to local optima. The literature suggests several approaches and StreamStory uses the approach called k -means++ proposed in [40]. k -means++ samples the initial centroids from the measurements but adjusts the probability of a measurement being selected proportional to the distance to the other selected centroids. Measurements closer to the other centroids thus have a lower probability of being selected resulting in the initial centroids being more dispersed on average.

A drawback of k -means is that the number of centroids has to be specified beforehand, requiring the user to assume the granularity of the data. To alleviate this issue, we support DP-means [39] as an alternative. The algorithm is presented in Algorithm 4.2.

Algorithm 4.2: Pseudocode of the DP-means algorithm

Data: a set of measurements $X = \{x_1, x_2, \dots, x_n\}$

a metric $d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^+$

Result: a set of centroids C

Set initial centroid $C = \{c_1\}$

repeat

 For each example $x_i \in X$ calculate $d_i = \min_{j \in C} d(x_i, c_j)$

if $\exists i$ such that $d_i > \lambda$ **then**

 Add x_i to C

else

 Assign each example x_i to the cluster C_j that minimizes $\arg \min_{j \in \mathbb{N}_k} d(x_i, c_j)$

 Update the centroids $c_j = \frac{1}{\|C_j\|} \sum_{x \in C_j} x$

end

until the assignment of examples to centroids has not changed

Similar to k -means, DP-means is an iterative procedure. The main difference is the way the algorithm constructs the centroids C . Instead of selecting the centroids at initialization, DP-means starts with a single centroid. Like k -means it then assigns the measurements to the cluster with the nearest centroid in each iteration. However, in contrast to k -means, if the distance between a measurement x_i and its assigned centroid is larger than a design time parameter λ , the measurement x_i is used as the centroid of a newly formed cluster. Thus $\sqrt{\lambda}$ represents the maximal radius of a cluster, controlling the number of clusters indirectly.

The main advantage of these two algorithms is that they are computationally efficient. The running time of the k -means algorithm is determined by its initialization step, assignment and update steps and the number of iterations. The time complexity of the k -means++ initialization is $\Theta(kn)$. Likewise, the time complexity of the assignment step is $\Theta(kn)$ and the time complexity of the update step is $\Theta(n)$ resulting in time complexity $\Theta(kn)$ per iteration [41]. When used with StreamStory, the number of clusters k is usually small compared to n making the time complexity of a single iteration linear with respect to the input. The number of iterations has been shown to be super-polynomial in the worst case. However, in practice the algorithm is known for its fast convergence. In our experiments, the algorithm only rarely required more than a thousand iterations to converge.

Like k -means, the time complexity of an iteration of DP-means is $\Theta(kn)$ [42]. However, unlike k -means, the number of centroids can differ in each iteration. Although in the worst case the number of clusters is linear in n , we find that in practice this number is bounded by the volume spanned by the dataset and the choice of the cluster radius λ . Like k -means the worst case number of iterations is super-polynomial but in practice the algorithm has fast convergence.

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.

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, these parameters represent the finest computed scale and thus control the maximum number of states that is 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 overly cluttered at the finest scales. While the choice of parameter can and does affect the constructed states, we have found that many interesting patterns are robust to the specific choice of parameters. For instance, in the experiments in Chapter 6 we did not need to perform parameter tuning.

4.4 Modeling Transitions

We now describe how we instantiate the continuous time Markov chain used to model temporal dynamics between the *initial states* on specific datasets. A continuous time Markov chain with m states is completely defined by a transition rate matrix $Q \in \mathbb{R}^{m \times m}$ and the *initial distribution* $\lambda \in \mathbb{R}^n$. The non-diagonal elements of Q , denoted by q_{ij} represent the rates of transitioning from state i to state j .

We estimate the non-diagonal elements of Q using the maximum likelihood estimator, which in the case of Markov chains is simply given by Equation 4.2 for each non-diagonal entry of the matrix.

$$\tilde{q}_{ij} = \frac{N_{ij}}{t_i} \quad (4.2)$$

In the equation above, N_{ij} represents the number of jumps from state i to state j while t_i represents the total time spent in state i as observed in the training data. The (negative) diagonal elements of Q , denoted by $q_i = -q_{ii}$, represent the rate of leaving state i . They are computed as the negative sum of the row (i.e. the rows of Q sum to zero):

$$\tilde{q}_i = - \sum_{\substack{i=1 \\ j \neq i}}^n \tilde{q}_{ij}. \quad (4.3)$$

We initialize the Markov chain in the first state that was observed in the training data. Assuming this state is i , this is equivalent to defining the initial distribution as the i -th base vector $\tilde{\lambda} = e_i$. We note that under the assumptions of Section 3.4, the initial distribution does not impact the ergodic properties of the Markov chain [37] and therefore does not influence its long-term behavior. This makes our representation independent of the specific choice of the initial distribution and therefore, we omit it from the representation.

In the final visualization, we use the transition probabilities from the *jump chain* to visualize temporal dynamics. The *jump chain* $(Y_n)_{n \geq 0}$ associated with a continuous time Markov chain $(X_t)_{t \geq 0}$ with transition rate matrix Q is a discrete time Markov chain that follows the same sequence of jumps as $(X_t)_{t \geq 0}$, but disregards the *holding times* [37] (i.e. the time the process stays in a state upon arrival). It is defined by a probability matrix P with elements p_{ij} which represent the transition probabilities. These are estimated as:

$$\tilde{p}_{ij} = \begin{cases} \frac{\tilde{q}_{ij}}{\tilde{q}_i}, & \text{if } i \neq j \\ 0, & \text{otherwise} \end{cases} \quad (4.4)$$

This design choice reduces clutter by removing transitions of the form $i \rightarrow i$ from the visual representation.

4.5 Constructing the Multiscale Hierarchy

The final step constructs a hierarchical partition of the *embedding space* that supports the *hierarchical Markov chain* representation and defines the topology of the states. In this representation the states of the *hierarchical Markov chain* are associated with appropriate partitions on each level of the hierarchy. The transitions between these states are computed as specified in Section 3.5.

To construct the hierarchical partition, StreamStory supports two approaches. The first is a bottom-up distance-based approach which uses *mean linkage* agglomerative clustering (UPGMA) [43] to iteratively merge pairs of states with minimal distance. In this step, the same metric is used as in Section 4.3 to measure the pairwise distances between the states. We typically use Euclidean distance as we find it performs well with data of medium dimensionality. 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 [44] 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 selected 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 rates as weights. To find an approximate solution, we use a standard technique from spectral graph theory. We consider the generalized eigenvalue problem and assign states based on the sign structure of the eigenvector corresponding to the second largest eigenvalue of the symmetrized transition rate matrix [45]–[47]. Specifically, we solve the following generalized eigenvalue problem:

$$\hat{Q}v = \lambda_2 \Pi v \quad (4.5)$$

where $\Pi = \text{diag}(\pi)$ is a matrix with the stationary distribution of $(X_t)_{t \geq 0}$ on the diagonal, λ_2 is the second largest eigenvalue, v is its corresponding eigenvector and $\hat{Q}$ is the symmetrized transition rate matrix defined as:

$$\hat{Q} = \frac{1}{2} (\Pi Q + Q^T \Pi). \quad (4.6)$$

We then use the sign structure of the eigenvector v to assign the states to one of the two new partitions.

Finally, we reduce the number of scales in the hierarchy by extracting the most representative scales that 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 extracted from the transition rate matrix at that scale. Then we perform k -means clustering on these vectors with the representative closest to the centroid of each cluster taken as one of the chosen scales. Intuitively, this final step forces scales with similar *ergodic* and convergence properties into the same group since the two largest eigenvalues (by absolute value) are associated with these properties. The number of clusters is chosen as the minimum of half of the number of *initial states* and ten. This number was chosen experimentally. It ensures 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. For an advanced user this automatic choice can be overridden to a manual one. We then post-process the hierarchical model for visualization, computing the layout of the states and enriching the structure by computing various statistics about states and transitions which are used to aid interpretation of the model. For details, please see Chapter 5.

4.6 Highlighting Recurrent Behavior

We provide the user with two options which help highlight recurrent behavior. The options influence the structure of the model by defining the *embedding function* when embedding the points in the point cloud in the initial step of the methodology presented in Section 3.2. The first defines the history embedding function to include information about the derivative into the feature vectors, while the second defines the time embedding function to emphasize periodicities by including information about the time that the measurement was taken. Both approaches effectively embed the points in a higher dimensional space which highlights certain temporal behavior by moving points with different characteristics further apart.

4.6.1 Using Dynamical Context

Based on the *time-delay embedding* [48] 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 one,

$$x(t) \mapsto \begin{pmatrix} x(t) \\ x(t-1) \end{pmatrix} \quad (4.7)$$

it appends the previous time step to the current time step. This helps reduce ambiguities by effectively considering the derivative of the time series at the sampled point $x(t)$. Intuitively, this embedding separates points with different local dynamics. It assumes that for two points to be similar, their path through the *ambient space* must also be similar for some number of time steps. Following the weather example from Chapter 1, the feature vectors are extended using values from the previous month separating points from spring and autumn months. The approach was originally developed for recovering the dynamics from chaotic systems [48].

4.6.2 Static Context Based on Time

The second approach separates points with dissimilar temporal properties by extending 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 which is one in the dimension that represents its categorical value and zero in all other dimensions. Hence the dimension increases by the number of possible categorical values. The approach requires the user to select a temporal granularity (i.e. time unit) before constructing the model. For instance, when the time unit selected is “day,” each feature vector is extended with a categorical feature indicating in which day of the week the associated measurement was taken (e.g. Monday, Tuesday, etc.), which increases the dimension by seven. All the supported options include:

- (a) *second* with six categories representing ten second intervals in a minute,
- (b) *minute* analogous to *second*,
- (c) *hour* with 24 categories indicating the hour of the day,
- (d) *day* with seven categories representing the days of the week and
- (e) *month* with twelve categories representing the months of the year.

The approach emphasizes periodicities. The additional categorical feature moves points which occur in different categories further apart while preserving the distance between points in the same category. For instance with the *day* feature, the distance between the points that occurred on Monday is preserved. On the other hand, the distance between a point that occurred on Monday and one that occurred on Tuesday and between the same point and one that occurred on Friday is increased, moving them further apart. This feature is useful if the domain expert would like to identify patterns at a specific temporal granularity. Each option represents a granularity and highlights a particular period. For instance, the *hour*, *day* and *month* features highlight the daily, weekly and yearly cycles respectively. Additionally, it allows the user to identify which states occurred during a specific time period (e.g. which states occur most often on Monday).

Chapter 5

Visual Representation

We begin this chapter with the specific requirements that guide the design of our visualization. We follow with the detailed description of the visual elements, their interactions and how they satisfy the given requirements. First, the visualization should provide a high level conceptual summary of the data which prevents visual clutter when visualizing large datasets. Temporal patterns, especially periodic and recurrent behavior, should be apparent. Second, users should be able to change the level of detail in the summary to find these patterns. This process should minimize context switching. Finally and most importantly, the visualization must help the user to interpret these patterns by linking elements of the visual summary back to the data.

The hierarchical Markov chain provides the multiscale summary that simplifies the data into discrete states, avoiding clutter while still capturing and highlighting its dynamics and long-term behavior. 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, highlight their differences, and automatically suggest possible interpretations using data-driven approaches. The result is a more intuitive visual representation.

5.1 User Interface

The user interface consists of three connected panels shown in Figure 5.1: (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 the 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. In this representation, recurrent behavior can be identified as a cycle in the diagram. 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.

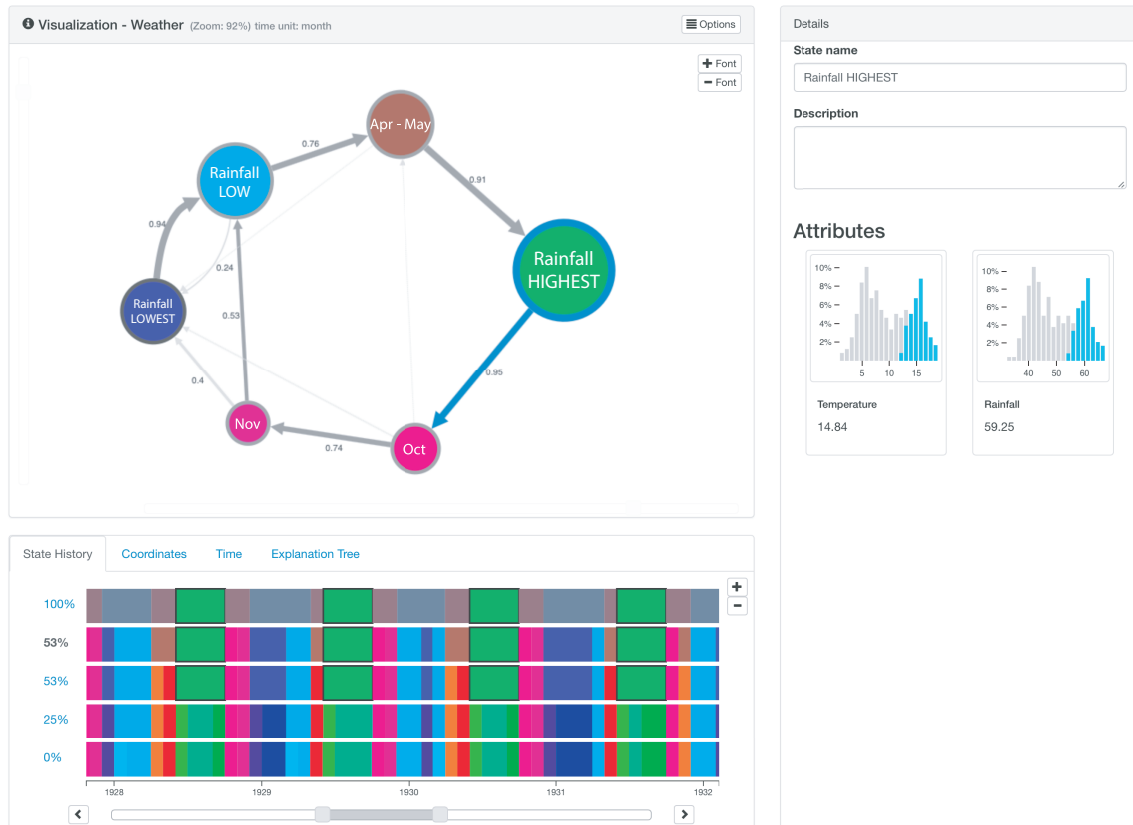


Figure 5.1: The visualization consists of 3 panels. The central panel visualizes the data through a Markov chain at the current scale. The bottom panel shows a cross-scale historical overview of the dataset using an *icicle plot*. The two panels are interactively linked. States on the two panels are associated by color. The right panel visualizes properties of the currently selected state (blue border) and allows the user to rename the state and store a description.

5.2 Visual Elements

We now describe the visual elements used to develop the visualization and how they visualize and highlight aspects of the abstract representation. We design the elements to highlight the long-term properties of the data, using properties of the underlying Markov chain used as the abstraction. We place special emphasis on avoiding information overload, designing the elements to be interactive, showing information upon request instead of presenting all the information at once.

We visualize states as circles and use five visual attributes to encode their properties (see Figure 5.2). 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 scientific contribution is to assign these values in a way which respects and highlights properties of the multiscale hierarchy. This helps the user understand how the structure at each scale relates to finer and coarser scales.

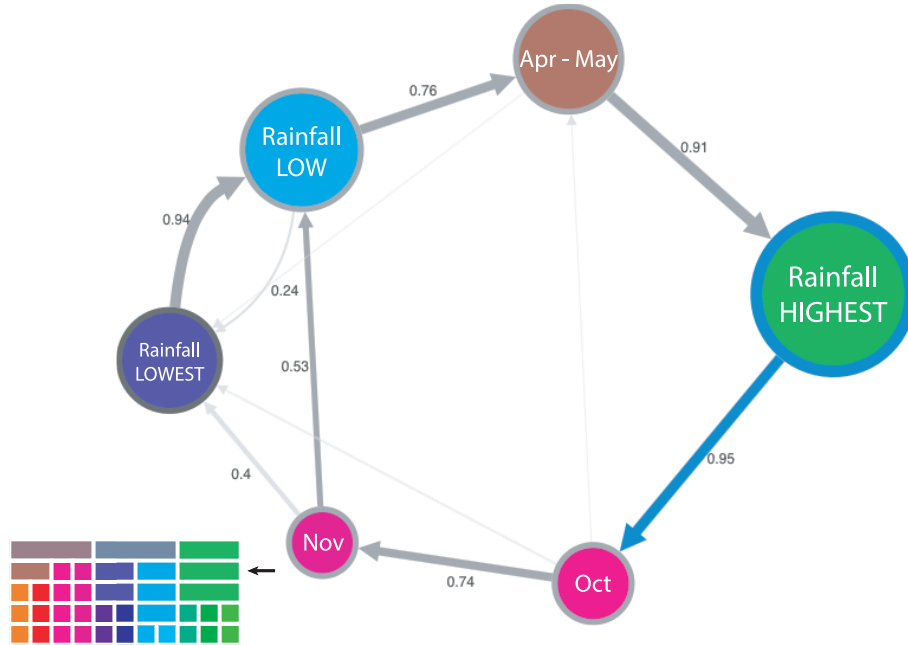


Figure 5.2: Visual attributes encode different properties of the states: (a) The area of each circle encodes the proportion of time the system spent in the state in the training dataset. (b) The position reflects the states’ position in the ambient space relative to other states. (c) The color encodes 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 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 π [37]. Position reflects the states’ position in the ambient space relative to other states. 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 5.3 for details). Color encodes the

states' position 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 the states on the same scale. For details see Section 5.4. Throughout the thesis, we use a color square like the one in Figure 5.2 to show the assignment of colors across the scales.

To help interpretation, StreamStory automatically assigns labels to states using the method presented in Section 5.5. A label highlights the properties that are typical for a 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 data. When selected the user can interactively change the states' label to reflect their own interpretation of the state through a form on the right panel. 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 the 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 [37]. This reduces clutter by eliminating transitions from a state back to itself. 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 Figure 5.2). 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 Figure 1.2 and Figure 5.1. The first view is an *icicle plot* which summarizes the dataset as a cross-scale historical overview. It shows the sequence of states which occur in the data for all scales simultaneously (Figure 5.1). 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 scale 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. Figure 5.3 shows the bottom panel with the yearly histogram of two states from Figure 5.1. The histograms show that one state usually occurs in the winter months (Figure 5.3(a)) while the other occurs in the summer months (Figure 5.3(b)).

The bottom panel offers two other views for the selected state. A description of the selected state is generated through decision trees [17] shown on the bottom panel. Decision trees are classification models often used in domains such as medicine for their explanatory power. When a decision tree is induced, a splitting attribute and cut value are chosen re-

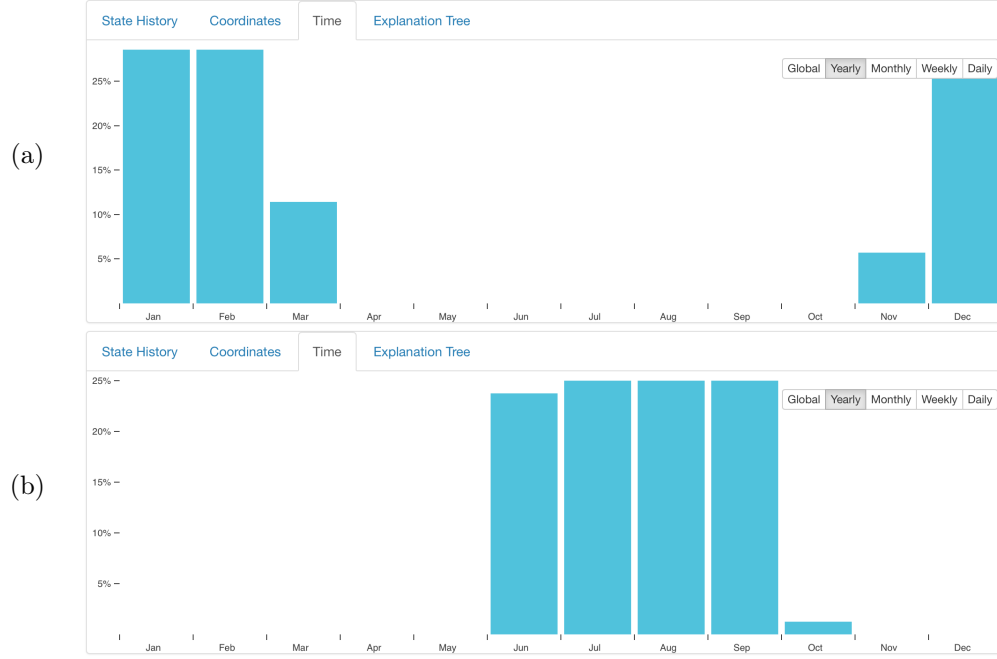


Figure 5.3: The *Time* histogram explicitly shows when a state or transition occurs. It supports several temporal granularities (e.g. a daily histogram showing the hours of the day) which can be used to identify periodic behavior. The example above shows the months when the *Rainfall LOWEST* (dark purple) and the *Rainfall HIGHEST* (green) states of Figure 6.1 occur. We see the former typically occurs in the winter (a) and the latter in the summer (b).

cursively by a design time criterion. The user can then interpret the tree by traversing the path from the root to one of the leafs. An example showing a decision tree for the selected state in Figure 5.1 is shown in Figure 5.4. In the figure, we can see that the state labeled *Rainfall HIGHEST* is primarily characterized by having rainfall higher than 54.3mm. Decision trees are provided as another tool for explaining the states. We compute one tree for each state by classifying all the measurements of the state against the measurements of all other states, obtaining a quantitative description of the states'. Each node in the decision tree shows the distribution of measurements belonging to the selected state versus all other states, indicating which nodes provide reliable interpretations. We provide a short summary of the state by extracting rules from the decision trees. The extracted rules are in the form of $A_i > a_i \cup A_j \in (a_{start}, a_{end}]$ where A_k represents the k -th attribute and a_{start} and a_{end} represent the thresholds computed by the decision trees. The rules are shown in the tooltip when hovering over a state.

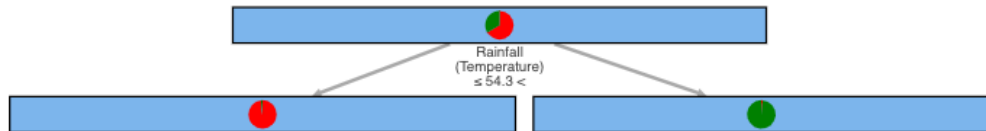


Figure 5.4: The decision tree describing the green *Rainfall HIGHEST* state from Figure 5.1 suggests that the state is characterized as having the average rainfall higher than 54.3mm. Each node in the decision tree visualizes the distribution of measurements belonging to the selected state versus all other states.

The right panel visualizes properties of the selected state. This includes the mean and distribution of all the attributes (Figure 5.5). For context, we show the global distribution of 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.

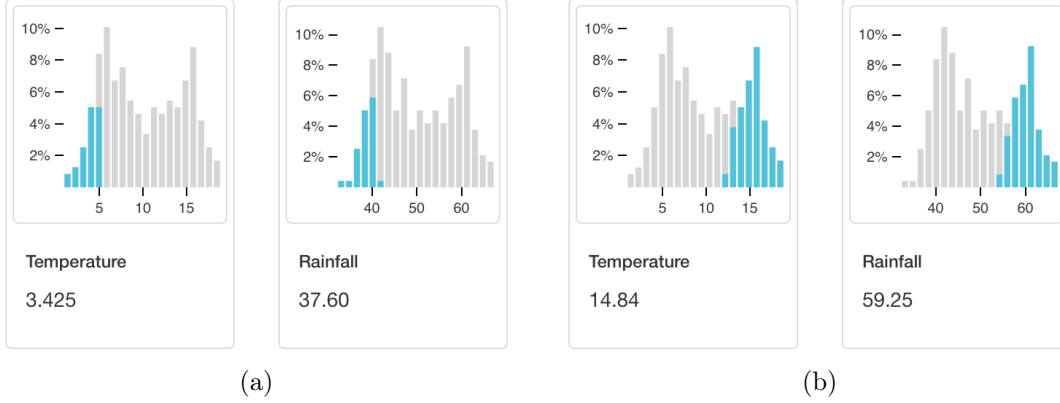


Figure 5.5: The *Attribute* histograms show the distribution of attributes in the selected state with the global distribution of the attribute as context in the background. Here we show the distribution of two attributes in the purple *Rainfall LOWEST* and green *Rainfall HIGHEST* states from Figure 5.1 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.

Finally, by selecting an attribute through a menu in the upper right corner of the central panel, users can compare the states with respect to that attribute. Once selected, the states in the central panel are recolored by the mean value of the attribute in each state (see Figure 6.3). 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.

5.3 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 the weighted average of their children. Traversing consecutive scales from finer to coarser, let $s_1, s_2, \dots, s_k$ be the states aggregated into state s . The second phase then traverses the scales iteratively and computes the coordinates of each new state as:

$$p(s) = \frac{\sum_{i=1}^k p(s_i) \pi(s_i)}{\sum_{i=1}^k \pi(s_i)} \quad (5.1)$$

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 the 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.

5.4 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 graph-based visualization.

StreamStory uses color coding to highlight cross-scale relationships and enable the user to traverse the scales efficiently. It uses *proximity based coloring* [49] 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-trees 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 Figure 5.6.

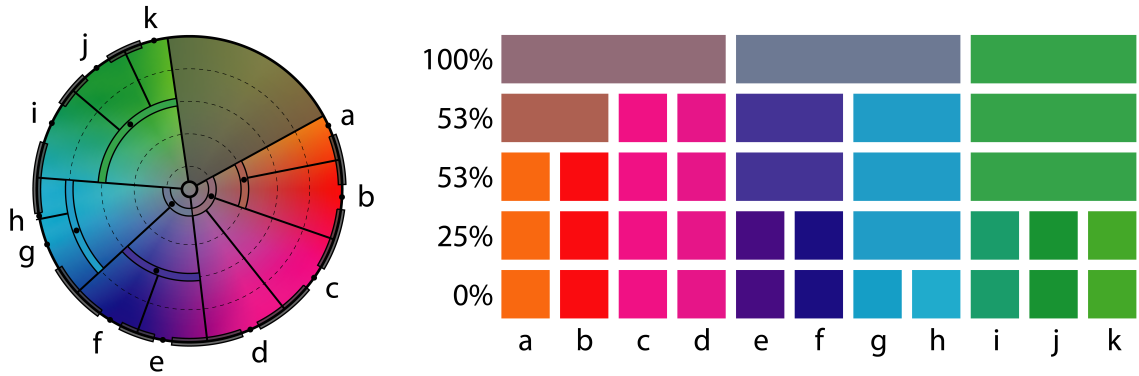


Figure 5.6: The color coding procedure. The *initial states* are distributed along the edge of the HSL color wheel. Each state is given a range proportional to its entry in the stationary distribution. Each concentric circle of decreasing saturation represents a subsequently coarser scale. When states merge (note that 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. Throughout the thesis we omit yellow 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.

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. This

gives states where the process spends more time more distinct color. 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 Figure 5.6).

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. Like with several other design choices we extract weights from the stationary distribution of the Markov chain at the appropriate scale. This makes the color of the larger states dominate their sub-tree (see Figures 1.2 and 5.1). We discretize the saturation into equal intervals from $s_{min} > 0$ to $s_{max} = 1$, numbering the scales 1 to n and assigning saturation s to all states with scale k according to the following formula:

$$s = 1 - \frac{s_k}{s_{max} - s_{min}}. \quad (5.2)$$

5.5 Automatic State Labeling

To help interpretation, StreamStory automatically generates descriptions of states, used as their 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 structure 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 one of two factors: (a) the values of the attributes inside the state when compared to values in other states and (b) the times when the state typically occurs. The label is shown as the default text on the state (Figure 5.7). 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 Figure 1.2.

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 attributes’ global distribution. More specifically, the mean value of the states’ distribution is compared to the γ -th and δ -th percentile of the global distribution. If the mean is below the γ -th percentile, we assign the label *LOWEST* to the attribute, while assigning label *LOW* if the mean is between both percentiles. We use analogous rules to assign labels *HIGH* and *HIGHEST* for high percentiles. We use default values of $\gamma = 12$ and $\delta = 25$ which perform well empirically. 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 procedure is shown in Algorithm 5.1. As input the procedure takes four values. The two histograms L and G with n bins each encode the local and global distribution of the attribute respectively, while the values of γ and δ represent the target quantiles. Each bin H_i is represented as a tuple $\langle \min(H_i), \text{count}(H_i), \max(H_i) \rangle$ where $\min(H_i)$ and $\max(H_i)$ encode the lower and upper bound of the corresponding interval while $\text{count}(H_i)$ encodes the number of measurements that were observed in the interval.

The algorithm first computes the total number of measurements taken in the target

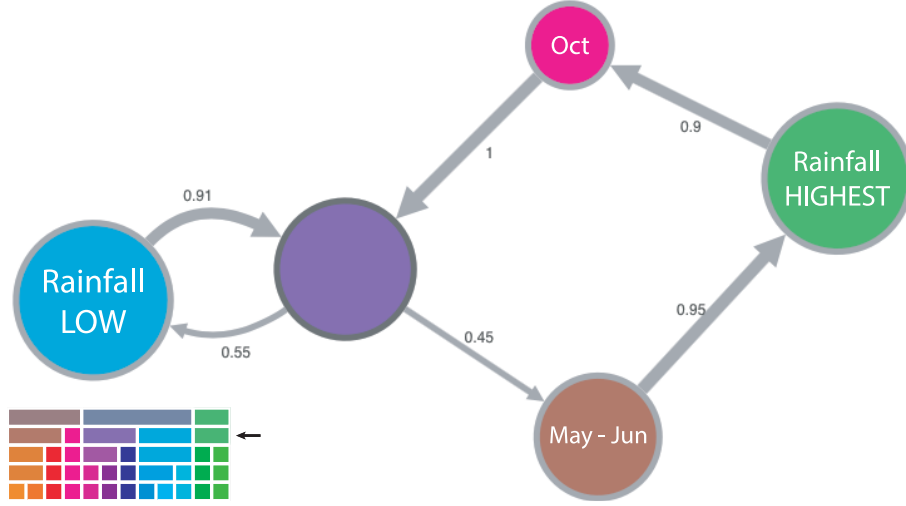


Figure 5.7: Automatic labels characterize a state based either on the typical configuration of attributes inside the state or time when the state typically occurs. Here we show two states, which were characterized based on the *Rainfall* attribute (blue and green) and two states characterized by typically 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.

Algorithm 5.1: Pseudocode of the attribute-based state labeling procedure

Data:

$G[n]$, a histogram with n bins encoding the global distribution of the attribute

$L[n]$, a histogram with n bins encoding the local distribution of the attribute

γ , the p -value corresponding to the lowest quantile (i.e. first threshold)

δ , the p -value corresponding to the low quantile (i.e. second threshold)

Result: A tuple of the form $(p, label)$ that encodes the attributes score p and label or $\emptyset$ is no label is found. The attribute with the lowest score is chosen as the final label.

$$t_L \leftarrow \sum_{i=1}^n \text{count}(L_i)$$

$$\mu_L \leftarrow \frac{1}{2t_L} \sum_{i=1}^n \text{count}(L_i) (\min(L_i) + \max(l_i))$$

$$F_G(\mu_L) \leftarrow \text{cdf}(G, \mu_L)$$

```

if  $F_G(\mu_L) < \gamma$  then
  | return  $(F_G(\mu_L), \text{LOWEST})$ 
end
if  $F_G(\mu_L) < \delta$  then
  | return  $(F_G(\mu_L), \text{LOW})$ 
end
if  $F_G(\mu_L) > 1 - \gamma$  then
  | return  $(1 - F_G(\mu_L), \text{HIGHEST})$ 
end
if  $F_G(\mu_L) > 1 - \delta$  then
  | return  $(1 - F_G(\mu_L), \text{HIGH})$ 
end

```

return $\emptyset$

state t_L and then estimates the mean value of the attribute in that state μ_L . Next, it estimates the value of the *cumulative distribution function* $F_G(\mu_L)$ of the global distribution of the attribute at μ_L via call to Algorithm 5.2. Finally, $F_G(\mu_L)$ is compared to γ , δ , $1 - \gamma$

Algorithm 5.2: Pseudocode to estimate the value of the cumulative distribution function from an attribute histogram

Data:

$H[n]$, a histogram encoding the distribution of the attribute

v , the value of the attribute

Result: p , the estimated value of the cumulative distribution function at v

Function $\text{cdf}(h, v)$

```

    if  $v < \min(h_1)$  then
        return 0
    end
     $t \leftarrow \sum_{i=1}^n \text{count}(H_i)$ 
    for  $i = 1, \dots, n$  do
        if  $\max(H_i) \geq v$  then
            return  $\frac{1}{t} \left( s + \frac{\text{count}(H_i)}{2} \right)$ 
        end
         $s \leftarrow s + \text{count}(H_i)$ 
    end
    return 1
end
```

and $1 - \delta$ and the label is output along with its score or $\emptyset$ if no threshold is reached. In this phase, the attribute with the lowest score is chosen as the final label of the state.

The second phase generates labels based on time by making use of *Time* histograms presented earlier. It 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. That is, if $b_k > \sum_{j=1}^n b_j / n$ where n represents the total number of bins in the histogram and b_j represents the value of bin j . Two consecutive peaks are considered a single peak. If a single peak with more than ψ mass (with default $\psi = 0.7$) is found, the peaks' time range is considered as a candidate label. Similar to phase one, we select the time range with its peak containing the most mass as the candidate label. For example, from the yearly histogram this would produce a label of the form *May - Jun* like the brown state in Figure 5.7.

Chapter 6

Use Cases and Evaluation

We use four datasets of varying complexity to confirm the three main hypotheses of this thesis. 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. We support this with a traffic dataset which exhibits similar, although more complex and dynamic, behavior. In the latter, we are able to identify two main drivers of periodic behavior which exhibit periodicities on different temporal scales. We then demonstrate how traversing the scales can reveal structure in the data with a wind dataset. 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 should thus not be directly compared. Finally, we discuss feedback gathered from experts during the development of the tool and compare StreamStory to related techniques.

6.1 Weather Data

We begin with a weather dataset used as an illustrative example throughout the thesis. The dataset consists of average monthly temperature and rainfall readings collected at Nottingham Castle, United Kingdom, over twenty 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 visual abstractions back to domain-specific concepts (i.e. months, seasons, etc.).

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 4.6.1, raising the dimension of the *ambient space* to four. Figure 6.1 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 Figure 6.1(a) 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 (Figure 6.2(a)) as opposed to the right state (Figure 6.2(b)). This suggests the right state represents the summer season while the left state represents the winter season. We confirm this by checking the yearly histograms which show that the left state lasts from November to April while the right state lasts from June to September. While in this case the interpretation is straightforward, the automatic labeling presents a clear suggestion which can easily be verified by the user using the supporting views.

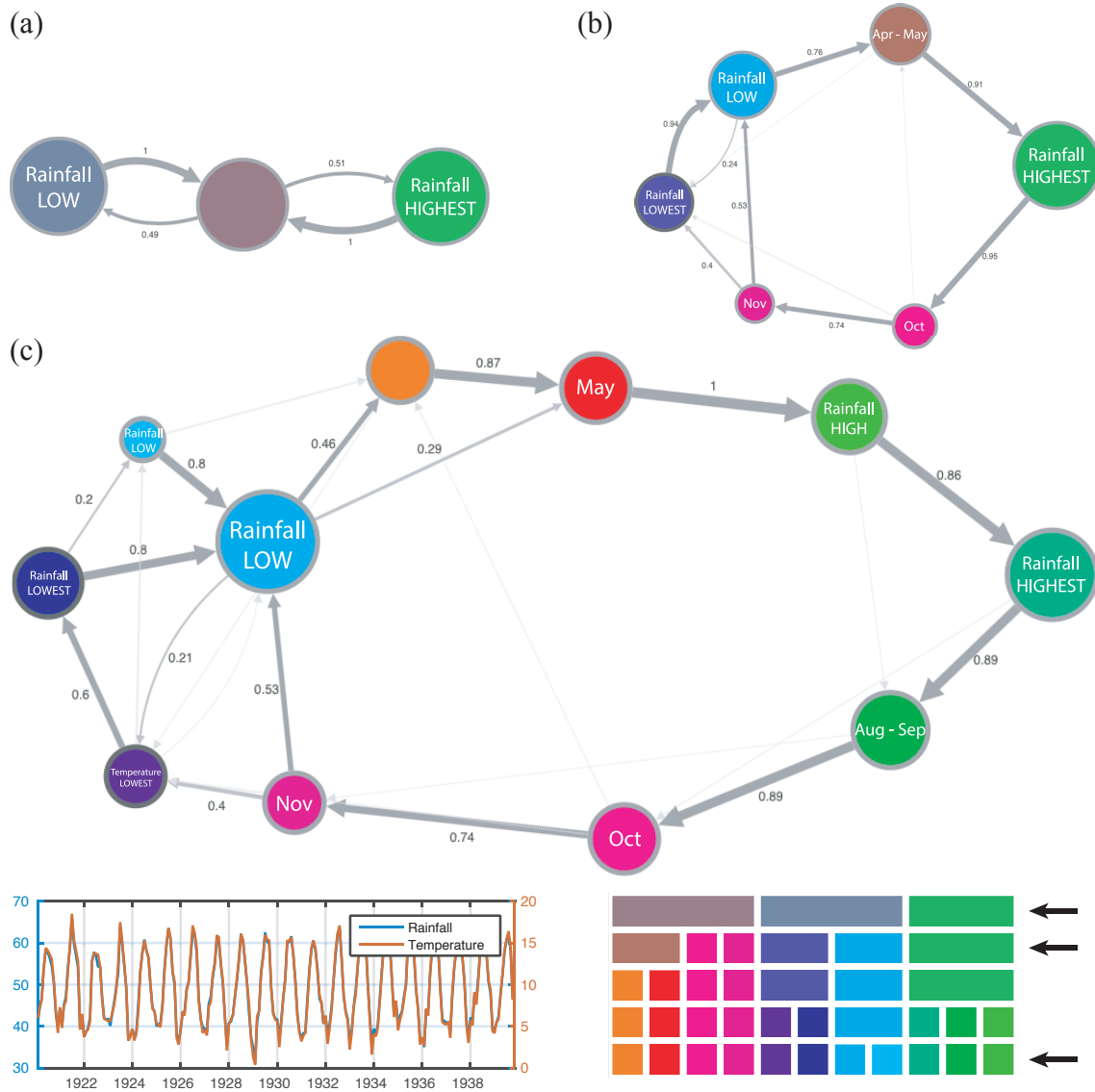


Figure 6.1: The weather dataset. In the weather dataset we observe a cyclical pattern, indicating periodic long-term behavior with the main states corresponding to different parts of the year. The yearly periodicity appears at the middle (b) and fine (c) scales. In this case, this can be confirmed directly by the parallel plot on the bottom left of the figure.

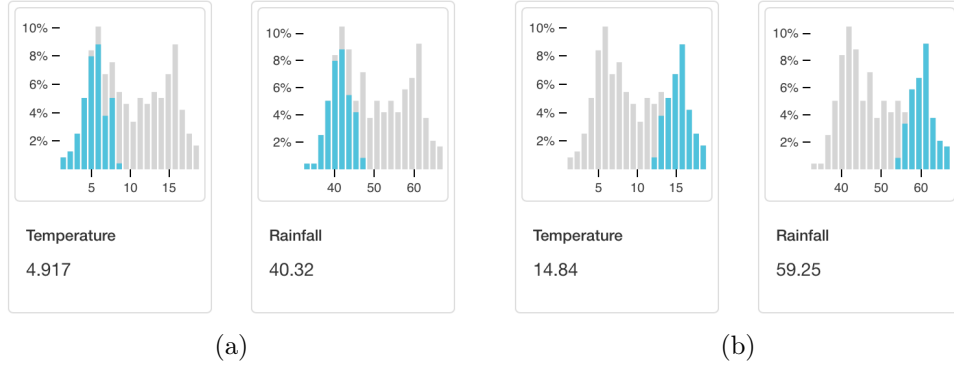


Figure 6.2: The *Attribute* histograms indicate that the purple *Rainfall LOWEST* and green *Rainfall HIGHEST* states in Figure 6.1(a) correspond to winter and summer months respectively. This is indicated by a low distribution of temperature and rainfall in the *Rainfall LOWEST* state (a) and a high distribution of temperature and rainfall in the *Rainfall HIGHEST* state (b).

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 (Figure 6.1(b)), 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 at the bottom of Figure 6.1. Furthermore, the automatic labels indicate that the top- and bottom-most states represent spring and autumn months respectively. Using the *Time* tab (Figure 5.3), we see that *Rainfall LOWEST* (dark purple) occurs between December and March (Figure 5.3(a)) while *Rainfall HIGHEST* (green) occurs between June and September (Figure 5.3(b)), 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. This suggests that the summers are significantly rainier than the winters in the UK. Through a menu in the corner of the central panel we visualize the distribution of each of the attributes by showing the mean value of the attribute across the states. Figure 6.3 shows the values for *Rainfall*. Recall from Section 5.2 that 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 attributes, inline with the plot in Figure 6.1.

At the finest scale (Figure 6.1(c)), we see a very dynamic winter compared to the relatively stable summer. With twelve 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 scale, we know when in the year each region corresponds to. The number of states and uniform transitions probabilities in the winter region suggests 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 (for instance 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 users findings. 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 examples illustrate

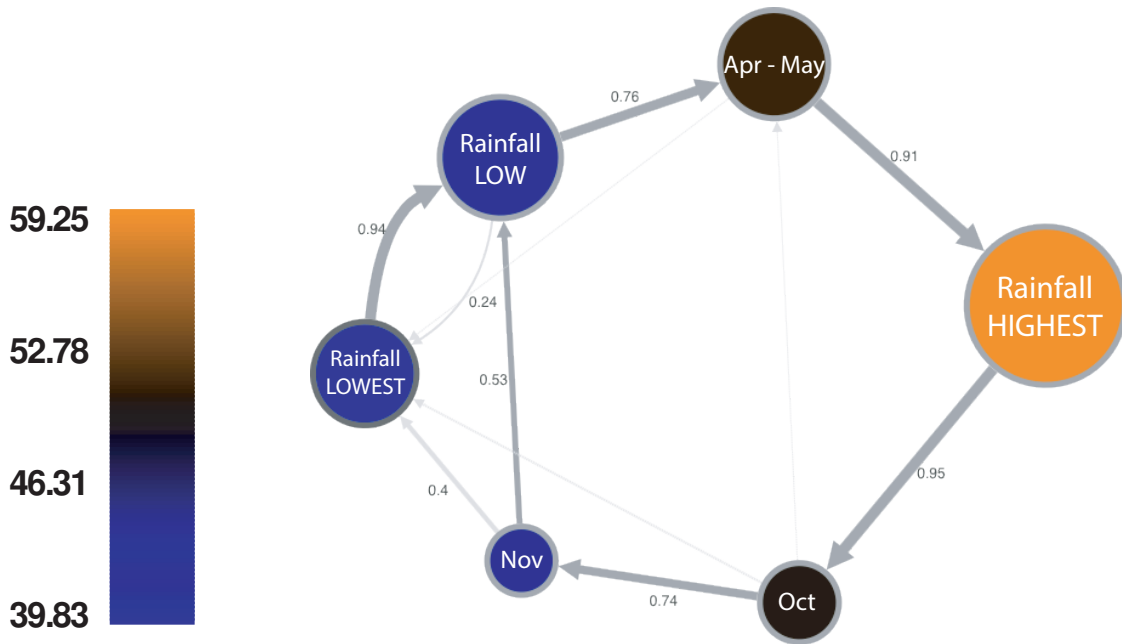


Figure 6.3: The distribution of *Rainfall* (in mm) confirms that the states on the right of Figure 6.1(b) are rainier than the states on the left. The same view for *Temperature* is almost identical suggesting a high correlation between the attributes (we only show the view for *Rainfall*).

the versatility of StreamStory.

6.2 Traffic Monitoring

We now demonstrate how StreamStory can be used to identify and interpret long-term and recurrent behavior using a dataset with a more complex cyclical structure. The dataset was collected from two traffic counters positioned on the highway ring around Ljubljana, Slovenia, measuring the rate of cars passing each counter, sampled on an hourly basis. The visualization is shown in Figure 6.4 along with a plot of a one-month sample of the data. Although much more unpredictable than (average) weather, at finer scales the visualization shows cyclic behavior on a daily basis.

At a coarse scale in Figure 6.4(a), we identify four main states. Checking the labels and daily histograms (Section 5.2), we identify the left green state as a nighttime state with low car rates. The pink state represents heavy traffic during the main commute times in the morning, afternoon and evening. The purple state represents moderate traffic and typically occurs in the early morning and around noon. The brown state on the right represents extremely heavy traffic and typically occurs on weekdays, especially Thursdays and Fridays, between 4 PM and 6 PM. It can be interpreted as increased traffic out of the city.

The middle scale exhibits interesting phenomena in the form of three daily cycles that are arranged in a figure eight pattern. In Figure 6.4(b), time flows clockwise, starting in the green states on the left representing nighttime. It then transitions to the large purple state which is characterized by moderate traffic. Checking the *Time* histograms we find a small cycle which occurs every morning. From the purple state, the system transitions through the magenta and red states, both characterized by relatively heavy traffic in the middle of the figure before returning back to the purple state at around 10 AM. This cycle represents

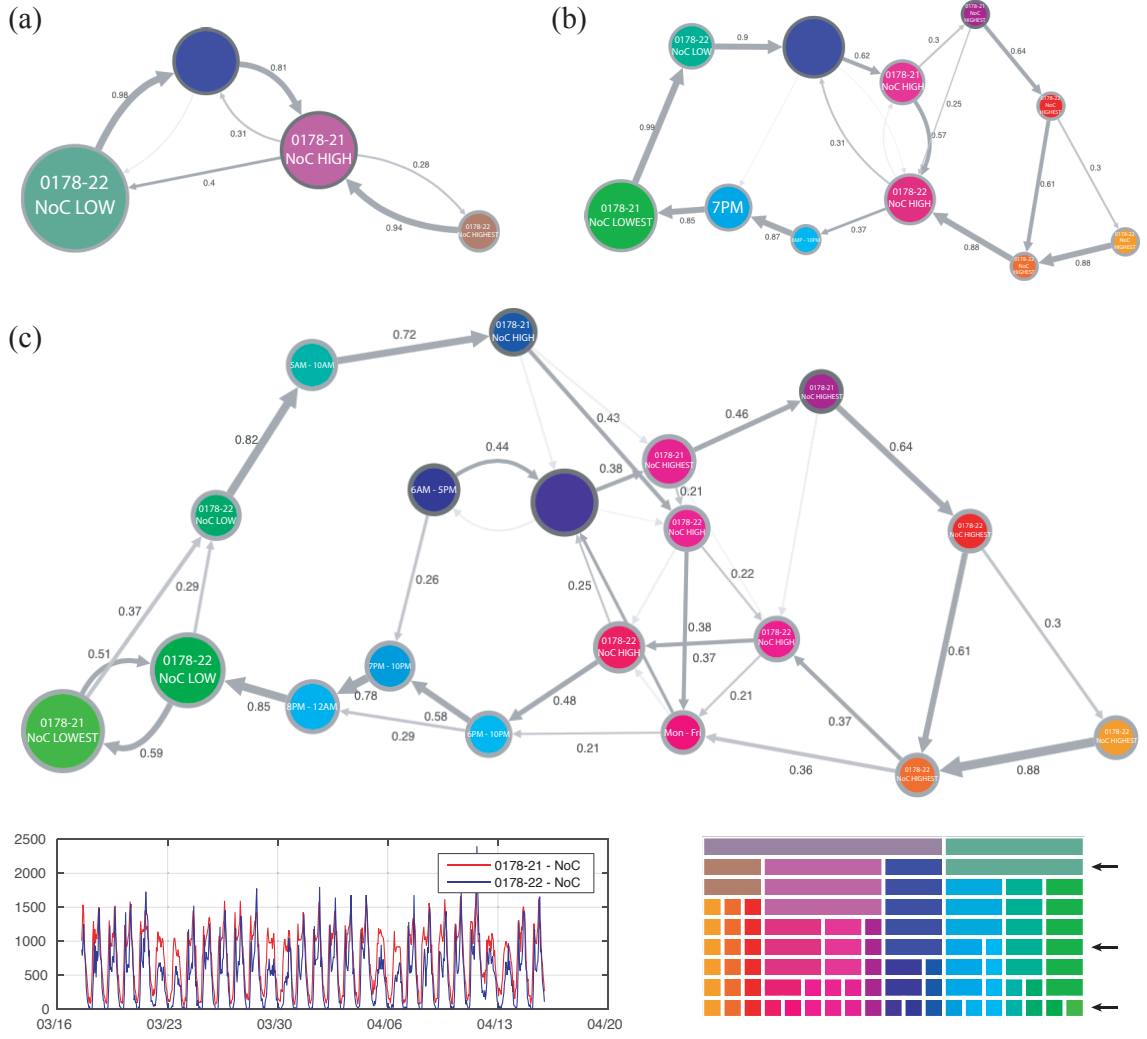


Figure 6.4: The traffic dataset. The traffic data exhibits cyclic patterns which suggest multiple periodic behaviors. The main states correspond to different times of day - the green states correspond to night, blue states are morning and evening and the others are daytime states. The middle scale (b) shows the daily dynamics which consists of three cyclic structures. The daily cycle starts and ends in the left green states which represent nighttime. In addition we find two smaller cycles. The first consists of the purple, magenta and red states in the center and represents morning rush hour. The second corresponds to afternoon rush hour and consists of the states on the right.

the dynamics of the morning rush hour. The second cycle passes through the center and then takes a different path. On weekdays, the system goes from the large purple state through the magenta state to the small purple state on the top right of the figure. It then continues through the red and dark orange states on the right and returns to the center. These three states are characterized by very high traffic and occur around 3 PM, 4 PM and 5 PM respectively, hence this second cycle represents the afternoon rush hour. The light orange state on the far right of the figure represents the most extreme traffic and typically only occurs on Fridays between 4 PM and 6 PM. It can be interpreted as increased traffic out of the city for the weekend. From the center the system then transitions to the cyan and blue states on the bottom left, which represent relatively low evening traffic, before returning to the large green nighttime state.

The finest scale in Figure 6.4(c) does not provide any new insights but merely shows the patterns described above in greater detail. The two green states now split into four nighttime states which differ primarily by the amount of traffic. Similarly, the central states split into many states. The afternoon cycle on the right is stable across the lower scales and remains constant. This data was collected and analyzed in conjunction with traffic experts¹. Initially, it was not clear that the data exhibited any such patterns (beyond the general increase in traffic during the commute times). Using the system, the individual patterns become obvious and can be readily interpreted.

By identifying the three cycles, this example demonstrates how StreamStory can be used to identify and interpret more complex cyclic structures which appear only locally (i.e. in specific parts of the *ambient space*) and how using the *Time* histogram such patterns can be mapped back to domain-specific concepts.

6.3 GPS Tracking

Our next example demonstrates several different types of patterns the user can observe when using StreamStory. The dataset, shown in Figure 6.5, contains personal GPS measurements, consisting of a timestamp, latitude and longitude, 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. We note that this process could be automated in the case of geographic data.

At a coarse scale (Figure 6.5(a)), 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 the person’s home location. The left orange state corresponds to the United States and 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 Figure 6.5(b) we see the United States state split into New York, California and Texas. New York is the largest of the three 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 Scandinavia is India (IND). This captures the various trips the individual has taken.

The finest scale (Figure 6.5(c)) demonstrates three of the most typical patterns which can be observed when 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 - also green) and finishing in Portugal (P - leftmost green state). Unfortunately, the return trip was not recorded. In the United States, New York splits into New York City (NYC) and upstate New York (NY) while Texas splits into Dallas and Austin. These states do not exhibit any special 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

¹The traffic dataset was collected and analyzed as part of the EU FP7 project Mobis, Grant Agreement No 318452.

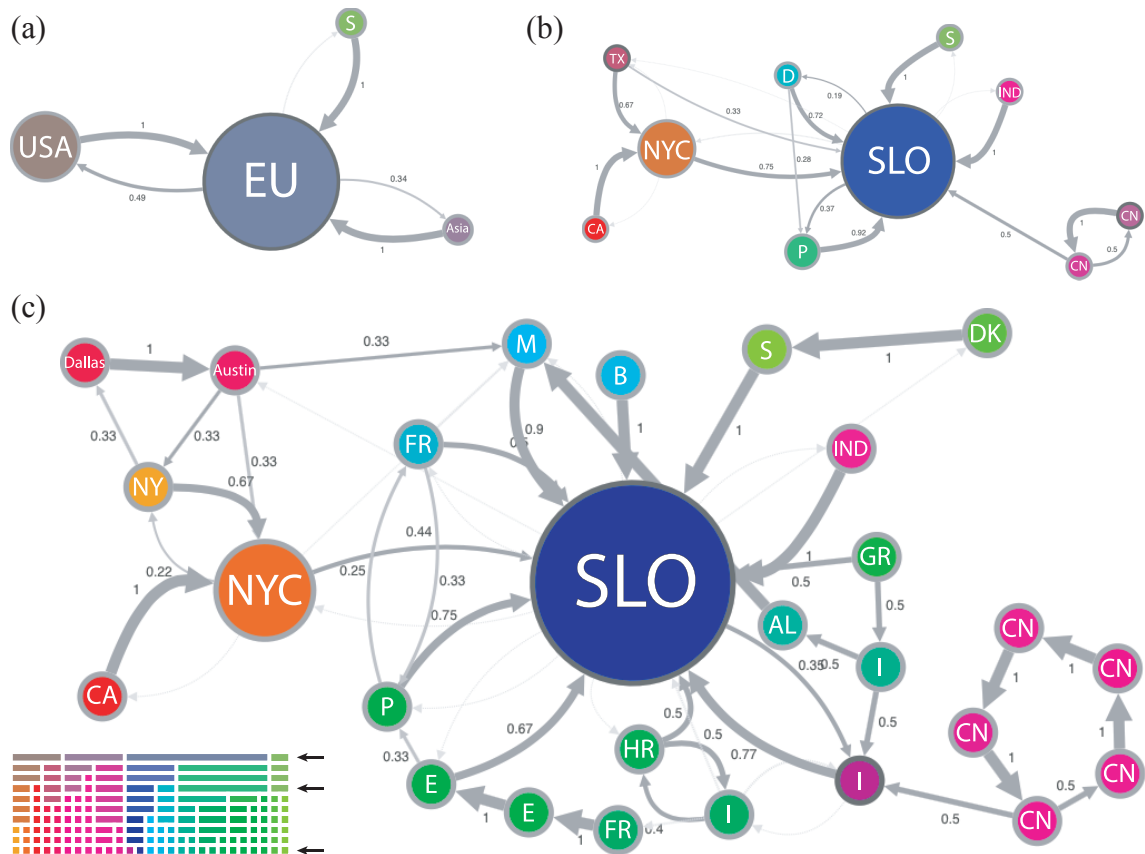


Figure 6.5: The GPS dataset. The main states represent locations around the world. The coarse (a) and middle (b) scales summarize distant locations on different continents. At the finest scale (c), the dataset demonstrates several typical patterns which can be observed using StreamStory. The central state represents Slovenia, the researcher's 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.

the coarsest scale their relatively small sub-trees (when compared to Europe) are assigned adjacent, relatively small color ranges. A possible mitigation is to increase the spacing between the sub-trees. However, this would narrow their color ranges and reduce differentiation within the sub-trees. In general it is not possible for the assignments at the coarser scales to reflect the distance between the sub-trees since it would require projecting the *ambient space* into a one dimensional color space.

6.4 Wind Monitoring

In our final example, we demonstrate 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. It contains 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 [50], strong gusts of wind which can reach over 150km/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.

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 six attributes. This includes the wind speed, direction that is represented as the sine and cosine of the angle and the change in each of these values. The values were sampled every second for a month. In this case, a plot of the raw data is uninformative but the resulting model at three scales can be seen in Figure 6.6.

At a coarse scale in Figure 6.6(a), 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. This can be seen by inspecting the *Attribute* histograms which indicate high wind speeds for the states on the right and confirmed by checking the *Attribute* histogram of the Bora indicator.

Moving to the middle scale in Figure 6.6(b), we identify the main states of interest. The group around the two large states on the 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 Figure 6.7 shows the unstructured nature of Bora winds. Indeed, with a minor bias toward morning hours, we see the state occurs at all times of the day, exhibiting no dominant daily patterns.

At a finer scale, in Figure 6.6(c) 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 Bora phenomenon - cold air spilling over the ridge and traveling down the face of the valley 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 [50]. This type of Bora was first observed in this dataset, due to the difficulty in manually isolating a suitable subset of data where this phenomenon occurs. StreamStory highlights

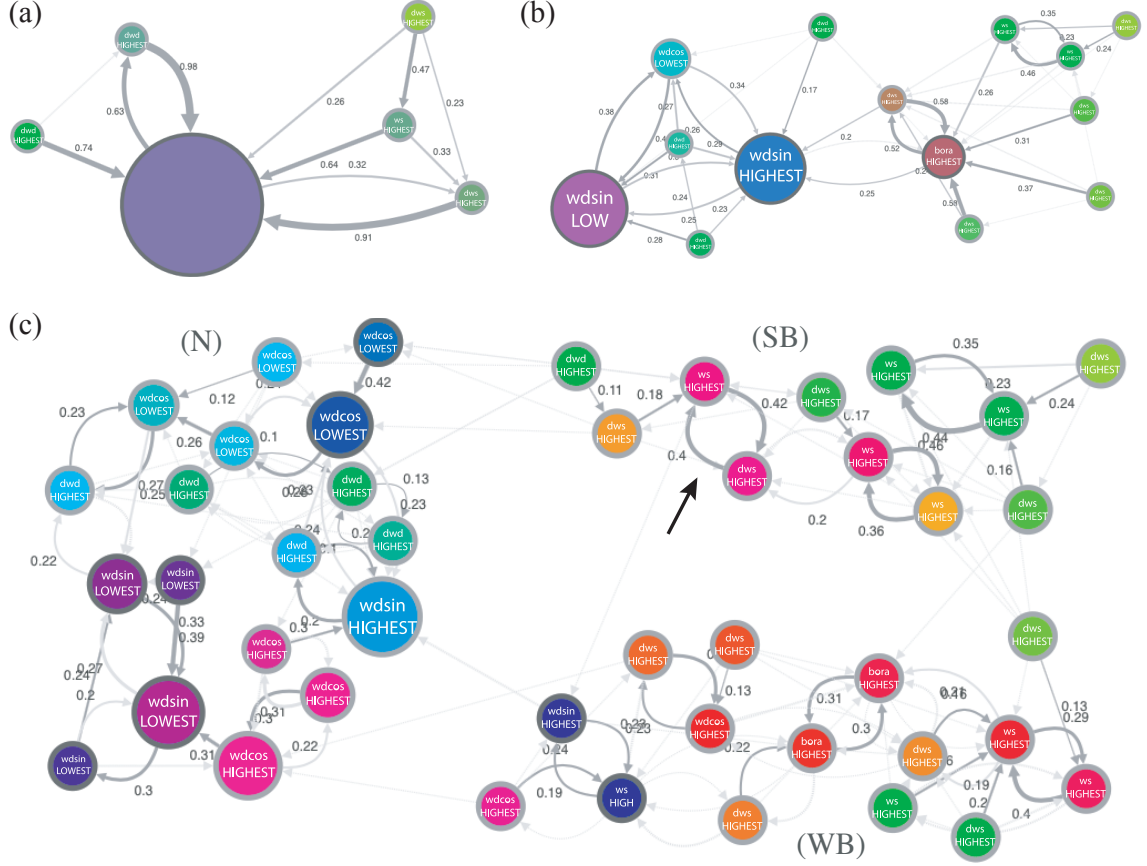


Figure 6.6: The wind dataset. We identify several groups of states representing different wind behavior, including two groups of Bora winds with different dominant directions. (a) At a coarse scale, there is a single dominant state with several satellites representing extreme behavior (i.e. high wind 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.

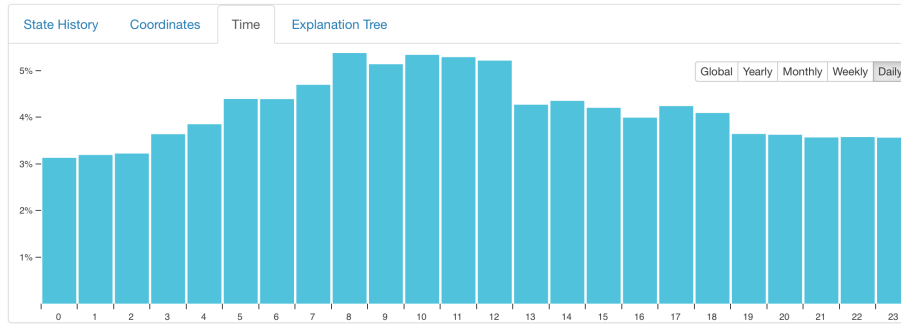


Figure 6.7: The daily histogram of a state predominantly characterized by Bora does not show any dominant daily patterns. With a small bias toward morning hours, we see that the state occurs at all times of day.

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 Bora 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 appearing in the two Bora groups. For instance, in the strong Bora group (SB), there is a cycle between the states representing high variation in wind speed and high wind speed in the southwest direction (*dws HIGHEST* and *ws HIGHEST*), pointed out in Figure 6.6(c). This cyclic pattern reflects the gusty nature of the wind - the two states capture the gusts (i.e. large changes 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 the data.

6.5 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 (i.e. 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, tooltips and the *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 dataset. 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 who used the tool in research

projects.

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 they are familiar 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.

6.6 Comparison to Other Techniques

In this section, we evaluate the effectiveness of StreamStory by comparing its results to the results of a comparable, and publicly available, system. Though there are many techniques for visualizing time series (as discussed in Chapter 2), we could not identify any directly comparable system. Similar approaches do not target non-expert users and are usually not designed specifically for visualizing large datasets. After an extensive analysis, we found the most comparable system to be *TimeCurves* [13]. *TimeCurves* represent each data point as a vertex and indicate the temporal sequence with edges between the vertexes. The edges are warped (or curved) so that similar points appear close to each other while dissimilar points appear further apart.



Figure 6.8: A visual representation of the weather dataset generated using *TimeCurves*. The vertexes represent months while the edges indicate their temporal sequence. From the representation we clearly see a cyclic structure that we identified in Section 6.1. To a certain extent we can also compare the trajectories measured in separate years. However, we are unable to map the different parts of the cycle back to the yearly seasons.

We demonstrate the key differences using two datasets from this thesis: (a) the weather dataset presented in Section 6.1 of the thesis and (b) the traffic dataset presented in Section 6.2. When visualizing both datasets we use the same preprocessing techniques as

were used when constructing the associated StreamStory representation. We note, however, that for the traffic dataset, only a small sample (i.e. one month) of data could be used since larger samples were rejected by the online demo.

Analyzing the representation, we find that the main drawbacks of using *TimeCurves* over StreamStory are: (i) the inability to map elements of the visual representation back to the data and (ii) large datasets tend to produce clutter. On the other hand, by discretizing data into states, StreamStory removes information from the visual representation which is especially noticeable for small datasets. The representations of the two datasets are shown in Figure 6.8 and Figure 6.9.

Comparing the results of the weather dataset (Figure 6.8), we find the main difference of this representation is the inability to map visual elements back to the original data to interpret domain-specific concepts. Though the figure shows a prominent cyclic pattern, the visualization does not allow for easily interpreting this cyclic pattern. From our knowledge of the dataset, we assume this represents the yearly cycle which we discuss in Section 6.1, but we cannot distinguish the different seasons.

The traffic dataset, shown in Figure 6.9, exhibits a mussel-like shape but the clutter conceals the cyclic structure we identified in Section 6.2. This dataset is more cluttered than

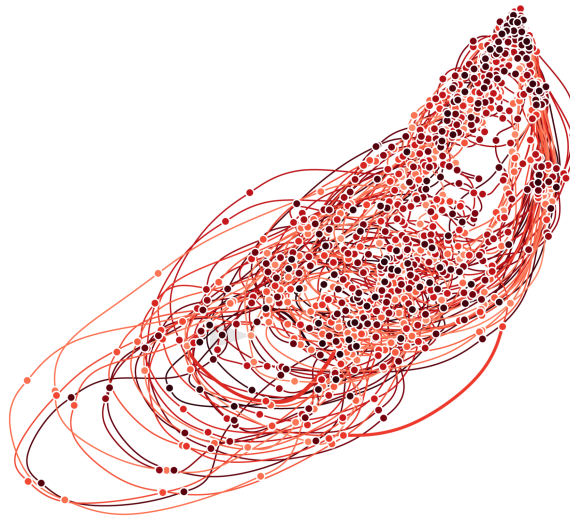


Figure 6.9: A visual representation of the traffic dataset generated using *TimeCurves*. Here the vertexes represent the hours of the day. This representation is more cluttered than the representation of the weather and it is difficult to interpret the temporal dynamics. On the other hand, we can quantitatively assess the density of the trajectories in different regions of the representation. For instance, we see that the upper-right region is much denser and much stabler than the bottom-left region. From our knowledge, we assume that the stable-most upper-right region represents the night hours while the bottom-left region represents the rush hour.

the weather dataset. The points in the upper-right corner appear most stable. Based on our knowledge, we assume these represent nighttime measurements discussed in Section 6.2. Likewise, we assume that the points in the lower-left corner represent the extreme traffic typical for Friday rush hour. There appears to be some activity in the center of the figure,

but due to clutter, we are unable to recognize any structured patterns.

Overall, we do 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 from [13]. Unfortunately, we could not test this hypothesis since we do not have access to the preprocessed dataset. StreamStory works best for large datasets of medium dimension, where many iterations of a cycle are present. In the case of very high dimensional and sparse data, different approaches are needed since StreamStory removes many of the important details in the data. However, as our examples show, in many cases, the system allows for intuitive exploration of the data highlighting interesting patterns.

6.7 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, only a summary of the data is stored and the visualization is fully interactive. In the experiments presented in Sections 6.1 to 6.4, the weather dataset which consisted of 6k measurements and the GPS dataset with 400k measurements were nearly instantaneous. The wind dataset was much larger (145MB), and thus took approximately 11 minutes to construct.

We performed an experiment, testing two datasets of different sizes while fixing the dimension and varying the number of *initial states* in the model. The first dataset (A) contains 285k measurements (155MB), while the second dataset (B) contains ≈ 3 M measurements (500MB). For each dataset, we selected seven attributes randomly and configured 10, 20 and 40 *initial states* respectively. We ran the experiment on a laptop with an Intel quad core 2.5GHz CPU with 8MB cache and 16GB of RAM. The results are summarized in Table 6.1.

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

Table 6.1: Measurements of the initialization time of a model indicate a strong dependence on the number of initial states.

The table indicates a strong dependence on the number of *initial states*. We note, however, that in the experiment 90% of this time was spent constructing the decision trees presented in Section 5.2 where one tree is created for each state in the hierarchy by classifying the measurements of the state against the measurements of all other states. By removing decision trees from the tool, the construction time could be reduced significantly.

Chapter 7

Conclusions

In this thesis, we presented a novel methodology and tool for the abstraction and visualization of large multivariate time series called StreamStory. StreamStory abstracts the data as states and transitions 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 help the user interpret the data by mapping the abstractions back to domain-specific concepts and suggesting 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-specific knowledge.

We demonstrated the utility of the proposed methodology on four real-world datasets of varying complexity. Using StreamStory, we identified several known as well as previously unknown patterns, presented feedback gathered from experts, presented a comparison to other related techniques and demonstrated its scalability.

Through the experiments and feedback gathered in Chapter 6 we confirm the three main hypotheses of this thesis. We demonstrated how the proposed abstraction can be used to identify long-term recurrent behavior while mitigating cluttering effects using the weather and traffic datasets. By using a wind dataset, we showed how traversing the scales can reveal interesting patterns. With all the use cases, we demonstrated how the structure and dynamics of the dataset can be mapped back to domain-specific concepts using the data-driven auxiliary views by providing an extensive discussion and interpretation of the resulting structure. In addition, we showed three of the most typical patterns that can be observed when interacting with StreamStory in the GPS dataset.

In the future we plan several extensions in different directions including new interactions, refining the visual representation, choice of parameters, etc. Some of these include:

Interactive Feedback: Rather than fixing the structure of the abstraction during construction, we will investigate how the user can interactively merge and split the states, modifying the structure manually.

Automatic Parameter Selection: Currently, there are still several parameters needed to construct the abstraction. Another possible direction is to explore mechanisms for automatically selecting and tuning these parameters.

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. For instance, one could compare the wind measurements taken at different times of the year.

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

Fuzzy States: The proposed methodology uses partitioning to construct the states of the representation, creating clear borders between the states. Could we construct a similar representation with fuzzy states, where a measurement could exist in more than one state at once?

Parametric Transitions: By modeling the transitions with Markov chains, StreamStory visualizes the expected behavior of the process. But what if the transition rates were parametric? Could we simulate how the process behaves under altered conditions?

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

Appendix A

Configuration and Parameters

We now discuss the configuration and parameters required to configure and construct the abstract StreamStory representation. Although we minimize the number of parameters required, there are still several settings which need to be configured to construct a StreamStory representation. This is in large part due to practical constraints (e.g. several datasets contain non-interesting attributes which users would like to ignore when modeling and visualizing the dataset) but also in part due to choosing algorithms from *machine learning* that support the abstraction of large datasets.

To visualize data using StreamStory, the user must login to the system and open the configuration form. Configuration includes several steps including selecting the dataset, choosing which attributes to use, selecting the attribute that represents the timestamp and configuring the type of the other attributes, selecting and configuring the clustering algorithm, etc. Figure A.1 shows the first part of the configuration form used to set up the system to visualize the weather dataset from Section 6.1. The form is dynamic, rendering different content depending on the current phase and configuration. We explain each of the configuration options in detail in the following sections.

A.1 Dataset and Attribute Selection

The first step when visualizing data with StreamStory is to select a dataset to visualize and configure which attributes to use. The dataset is uploaded through a form shown in Figure A.1. We require that the data be stored in CSV format and that each attribute (i.e. column) in the CSV be labeled as the labels are used as attribute names in the visualization. A sample of the weather dataset from Section 6.1 is shown in Figure A.2.

Once uploaded the user should select which attributes to visualize. The reasons to support this choice are purely practical. Several datasets contain many attributes, some of which may not be interesting to the user. By allowing them to choose which attributes to use, we eliminate the need of a manual preprocessing step.

A.2 Attribute Configuration

After the dataset is uploaded and the user selects which attributes they would like to visualize, they are required to configure the attributes.

First the user must specify which attribute represents time. We require that one of the attributes is a numeric, millisecond representation of time. While we could use heuristics to detect the temporal attribute, early experiments showed that because of occasional misguesses such heuristics cause confusion to first time users and do not increase the

Configure Model

Choose file

Choose file

rainfall.csv

Upload the CSV file you would like to model. The system expects the columns of the CSV file to be named in the first line of the file and that one attribute is a numerical representation of time. This can be any millisecond time representation.

100%

Select Attributes

Ignored Attributes

Empty list

Filter

→ →

Selected Attributes

Showing all 3

Filter

← ←

Time

Temperature

Rainfall

Configure Time Attribute

Select time attribute (must be timestamp)

☐ Time

☐ Temperature

☐ Rainfall

Select time unit

Hour

☒ Include time attributes in model construction

Done

Figure A.1: Screenshot of StreamStory’s initial configuration form. When starting configuration the user is required to upload a dataset in CSV format, select which attributes to use, select the attribute that represents the timestamp, select the temporal granularity (i.e. time unit) and decide whether to include attributes extracted from the timestamps of the measurements as presented in Section 4.6.2 into the feature vectors. Once this basic configuration is entered, the form is dynamic, rendering different content for different configuration options.

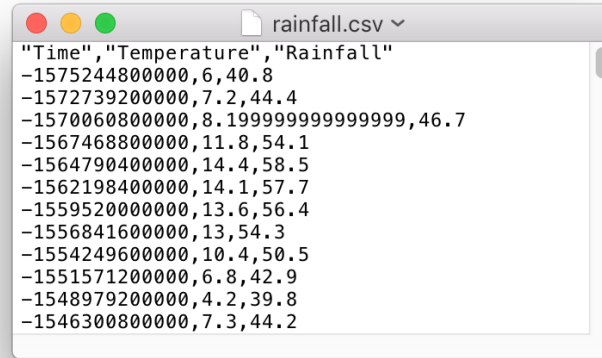


Figure A.2: A sample of the weather dataset. The uploaded datasets are required to be in CSV format with labeled columns. One of the attributes is required to represent a numeric representation of time. This can be any millisecond time representation.

productivity of expert users because of the manual check that is required to ensure that the guess was correct.

Then the temporal granularity should be chosen to reflect the natural dynamics of the data. The available granularities are: *second*, *minute*, *hour*, *day* and *month*. The temporal granularity influences some visual elements and defines the time unit used for modeling the transition rates between the states (e.g. number of jumps per minute).

By clicking on the checkbox on the bottom of Figure A.1 the user specifies whether to include static temporal features in the point cloud representation as discussed in Section 4.6.2. These features highlight periodic behavior at a fixed granularity. The granularity depends on the particular choice of time unit presented above. For instance, when selecting the time unit *hour*, StreamStory will highlight behavior which repeats in every 24 hour period.

We then require the user to select the type of each attribute. As shown in Figure A.3, StreamStory supports two types of attributes: (a) *numeric* and (b) *categorical*. The main difference is in their internal representation. While we use raw values to represent *numeric* attributes, for *categorical* attributes we use categorical feature vectors. Categorical feature vectors are binary vectors with as many dimensions as there are possible categories. All entries of the vector are set to zero except the dimension that represents the current category which is set to one. The system tries to guess the type of attributes by scanning the first ten lines of the uploaded CSV for non-numeric entries. The corresponding attribute is pre-set as *categorical* if such an entry is found. The user should check the types of attributes in case of incorrect guesses.

In the final part of the attribute configuration section the user can select attributes for which they would like to include derivative information as described in Section 4.6.1. By default no attribute is selected. For all options except selecting the attribute that represents the timestamp, the system suggests sensible defaults which are suitable for a wide range of datasets. With the configuration of attributes finished, the user can move on to configuring the data abstraction phase.

Configure Time Attribute

Select time attribute (must be timestamp)

☒ Time
☐ Temperature
☐ Rainfall

Select time unit

Hour

☒ Include time attributes in model construction

Select attribute types

Temperature Numeric: ☒ Categorical: ☐
Rainfall Numeric: ☒ Categorical: ☐

Include derivative

Temperature ☐
Rainfall ☐

Figure A.3: Attribute configuration. To configure the attributes, the user has to specify which attribute represents time, select the temporal granularity (i.e. time unit), indicate whether to include features extracted from timestamps in the embeddings, determine the type of the attributes and select for which attribute they would like to include derivative information into the feature vectors. For all options but the first we provide sensible defaults that can be used with most datasets.

A.3 Configuring the Data Abstraction Phase

With the configuration of attributes complete the user can configure the parameters of the data abstraction phase. This phase includes two steps: (a) configuring the clustering algorithm and (b) configuring the multiscale hierarchy.

Configuration of the clustering depends on which algorithm the user selects. By default, k -means is used. The appropriate section of the configuration form is shown in Figure A.4. When selecting k -means, the user is required to specify the number of clusters k which represents the number of *initial states* (i.e. the number of states on the lowest scale). In contrast DP-means does not specify the number of *initial states* but rather controls the size of the individual states by specifying the maximal radius of a state. For convenience, when selecting DP-means, we also let the user specify the minimal and maximal number of states. We incorporate these two parameters directly into the algorithm.

We offer two choices to construct the multiscale hierarchy. The first and default is a distance-based approach which uses mean linkage agglomerative clustering to construct a hierarchical partition of the *embedding space*. The second option exploits the dynamics between the states to construct the hierarchical partition. It uses a standard method from spectral graph theory to iteratively split a state of the Markov chain. Both methods are described in Section 3.5.

Here, we also allow the user to specify “ignored” attributes. Attributes specified as “ignored” are not used by the embedding function and therefore omitted from the abstraction, but are still visualized in the visual representation. Superimposing such attributes is useful for studying correlations in the dataset.

(a)

Configure States ?

Select Algorithm

K-Means

Number of States

12

(b)

Configure States ?

Select Algorithm

DP-Means

Minimum Number of States

10

Maximum Number of States

30

State Radius

0.8

Figure A.4: Configuring the clustering algorithm. We support two choices of the clustering algorithm which computes the *initial states*. When choosing *k*-means (a), the user can select the number of *initial states* by specifying the parameter *k*, while specifying the maximal radius of the cluster when choosing DP-means (b). In case of DP-means, we also allow the user to enforce a lower and upper bound on the number of clusters.

A.4 Finishing Configuration

Finally, the user can name the model and enter a description before finishing the configuration. Once finishing the configuration, the construction phase begins. The model construction is the most time consuming step when visualizing data with StreamStory. It scans the entire dataset, constructing the states and transitions and several *machine learning* models which support the auxiliary views. Indeed, after this initial step, the visual representation is fully interactive. Naming and describing the model is important as it allows the user to create, store, organize and view an entire collection of models. These two properties can be changed later when reorganizing the collection.

References

- [1] R. Rosenholtz, Y. Li, J. Mansfield, and Z. Jin, “Feature congestion: A measure of display clutter,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’05, Portland, Oregon, USA: ACM, 2005, pp. 761–770, ISBN: 1-58113-998-5. DOI: 10.1145/1054972.1055078. [Online]. Available: <http://doi.acm.org/10.1145/1054972.1055078>.
- [2] L. Stopar, P. Škraba, M. Grobelnik, and D. Mladenić, “StreamStory: Exploring multivariate time series on multiple scales,” *IEEE Transactions on Visualization and Computer Graphics*, 2018. DOI: 10.1109/TVCG.2018.2825424.
- [3] L. Stopar, “Multi-scale methodology for explaining data streams,” in *Proceedings of SiKDD 2015*, Ljubljana, Slovenia, 2015, pp. 37–40.
- [4] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale, “A Review of Temporal Data Visualizations Based on Space-Time Cube Operations,” in *EuroVis - STARs*, Eurographics, 2014, ISBN: 978-3-03868-028-4.
- [5] W. Aigner, S. Miksch, H. Schumann, and C. Tominski, *Visualization of time-oriented data*, 1st. Springer, 2011, ISBN: 0857290789, 9780857290786.
- [6] H. Hochheiser and B. Shneiderman, “Dynamic query tools for time series data sets: Timebox widgets for interactive exploration,” *Information Visualization*, vol. 3, no. 1, pp. 1–18, Mar. 2004, ISSN: 1473-8716. DOI: 10.1145/993176.993177.
- [7] S. Havre, B. Hetzler, and L. Nowell, “Themeriver: Visualizing theme changes over time,” in *Proc. of the IEEE Symp on Information Vis*, ser. INFOVIS ’00, Washington, DC, USA: IEEE Computer Society, 2000, ISBN: 0-7695-0804-9.
- [8] L. Byron and M. Wattenberg, “Stacked graphs - geometry amp; aesthetics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1245–1252, Nov. 2008, ISSN: 1077-2626. DOI: 10.1109/TVCG.2008.166.
- [9] R. Peng, “A method for visualizing multivariate time series data,” *Journal of Statistical Software, Code Snippets*, vol. 25, no. 1, pp. 1–17, Feb. 2008, ISSN: 1548-7660.
- [10] 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 TVCG*, vol. 15, no. 6, pp. 1049–1056, Nov. 2009, ISSN: 1077-2626. DOI: 10.1109/TVCG.2009.187.
- [11] M. Burch, F. Beck, and S. Diehl, “Timeline trees: Visualizing sequences of transactions in information hierarchies,” in *Proc. of the Working Conf. on Adv. Vis. Interfaces*, ser. AVI ’08, Napoli, Italy, 2008, pp. 75–82, ISBN: 978-1-60558-141-5. DOI: 10.1145/1385569.1385584.
- [12] S. Haroz, R. Kosara, and S. L. Franconeri, “The connected scatterplot for presenting paired time series,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 9, pp. 2174–2186, Sep. 2016, ISSN: 1077-2626. DOI: 10.1109/TVCG.2015.2502587.

- [13] 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 TVCG*, vol. 22, no. 1, pp. 559–568, 2016, ISSN: 1077-2626.
- [14] H. Jänicke and G. Scheuermann, "Steady visualization of the dynamics in fluids using epsilon-machines," *Computers and Graphics*, vol. 33, no. 5, pp. 597–606, 2009.
- [15] 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, 2011.
- [16] M. Pylvänen, S. Äyrämö, and T. Kärkkäinen, "Visualizing time series state changes with prototype based clustering," in *Proc. of the Conf. on Adaptive and Natural Comp. Algorithms*, Kuopio, Finland, 2009, pp. 619–628, ISBN: 3-642-04920-6, 978-3-642-04920-0.
- [17] O. Maimon and L. Rokach, *Data mining and knowledge discovery handbook*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005, ISBN: 0387244352, 9780387244358.
- [18] I. Davidson, "Visualizing clustering results," in *SIAM International Conference on Data Mining*, 2002.
- [19] M. G. Luka Stopar Blaz Fortuna, "Newssearch: Search and dynamic re-ranking over news corpora," in *Conference on Data Mining and Data Warehouses*, 2012.
- [20] N. Cao, D. Gotz, J. Sun, and H. Qu, "Dicon: Interactive visual analysis of multi-dimensional clusters," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2581–2590, Dec. 2011, ISSN: 1077-2626.
- [21] J. Johansson, P. Ljung, M. Jern, and M. Cooper, "Revealing structure within clustered parallel coordinates displays," in *IEEE Symposium on Information Visualization. INFOVIS.*, Oct. 2005, pp. 125–132.
- [22] 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*, Kuala Lumpur, Malaysia, 2012, pp. 505–516, ISBN: 978-3-642-30216-9.
- [23] M. Trampuš, F. Fuat, D. Pighin, T. Štajner, J. Berčič, B. Novak, D. Rusu, L. Stopar, and M. Grobelnik, "DiversiNews: Surfacing diversity in online news," *AI Magazine*, vol. 36, no. 4, 2015.
- [24] M. Trampuš, F. Fuat, J. Berčič, D. Rusu, L. Stopar, and T. Štajner, "(i)DiversiNews - a stream-based, on-line service for diversified news," in *Proceedings of SiKDD 2013*, Ljubljana, Slovenia, 2013.
- [25] T. Liu, "Application of markov chains to analyze and predict the time series," *Modern Applied Science*, vol. 4, no. 5, p. 162, 2010.
- [26] V. Soloviev, V. Saptsin, and D. Chabanenko, "Markov chains application to the financial-economic time series prediction," arXiv.org, Papers, 2011.
- [27] S. Vasanthi, M. Subha, and S. T. Nambi, "An empirical study on stock index trend prediction using markov chain analysis," *Journal of Banking Financial Services and Insurance Research*, vol. 1, no. 1, pp. 72–91, 2011.
- [28] W. Ching, S. Zhang, and M. Ng, "On multi-dimensional markov chain models," *Pacific Journal of Optimization*, 2007, ISSN: 1348-9151.
- [29] C. Wang, T.-Z. Huang, and W.-K. Ching, "A new multivariate markov chain model for adding a new categorical data sequence," *Mathematical Problems in Engineering*, 2014.

- [30] 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, ISSN: 1758-8715. DOI: 10.1504/IJIEI.2011.040177.
- [31] J. Woodring and H.-W. Shen, “Multiscale time activity data exploration via temporal clustering visualization spreadsheet,” *IEEE Trans. on Vis. and Computer Graphics*, vol. 15, no. 1, pp. 123–137, Jan. 2009, ISSN: 1077-2626. DOI: 10.1109/TVCG.2008.69.
- [32] M. Luboschik, C. Maus, H. J. Schulz, H. Schumann, and A. Uhrmacher, “Heterogeneity-based guidance for exploring multiscale data in systems biology,” in *2012 IEEE Symposium on Biological Data Visualization (BioVis)*, Oct. 2012, pp. 33–40. DOI: 10.1109/BioVis.2012.6378590.
- [33] D. Auber, Y. Chiricota, F. Jourdan, and G. Melançon, “Multiscale visualization of small world networks,” in *Proceedings of the Ninth Annual IEEE Conference on Information Visualization*, ser. INFOVIS’03, Seattle, Washington: IEEE Computer Society, 2003, pp. 75–81, ISBN: 0-7803-8154-8.
- [34] C. Stolte, D. Tang, and P. Hanrahan, “Multiscale visualization using data cubes,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 176–187, Apr. 2003, ISSN: 1077-2626. DOI: 10.1109/TVCG.2003.1196005.
- [35] G. Leban, B. Zupan, G. Vidmar, and I. Bratko, “Vizrank: Data visualization guided by machine learning,” *Data Min. Knowl. Discov.*, vol. 13, no. 2, pp. 119–136, 2006.
- [36] B. Fortuna, M. Grobelnik, and D. Mladenic, “Ontogen: Semi-automatic ontology editor,” in *HCI (9)*, ser. Lecture Notes in Computer Science, vol. 4558, Springer, 2007, pp. 309–318.
- [37] J. Norris, *Markov chains*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.
- [38] B. Fortuna, J. Rupnik, J. Brank, C. Fortuna, V. Jovanoski, M. Karlovčec, B. Kažič, K. Kenda, G. Leban, A. Muhič, B. Novak, J. Novljan, M. Papler, L. Rei, B. Sovdat, L. Stopar, M. Grobelnik, and D. Mladenić, “QMiner: Data analytics platform for processing streams of structured and unstructured data,” in *Neural Information Processing Systems*, Ljubljana, Slovenia, 2014.
- [39] B. Kulis and M. I. Jordan, “Revisiting k-means: New algorithms via bayesian non-parametrics,” *CoRR*, vol. abs/1111.0352, 2011.
- [40] D. Arthur and S. Vassilvitskii, “K-means++: The advantages of careful seeding,” in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA ’07, New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035, ISBN: 978-0-898716-24-5.
- [41] B. Fortuna, “Semi-automatic ontology construction,” PhD thesis, Jožef Stefan International Postgraduate School, 2011.
- [42] L. Jiang, Y. Dong, N. Chen, and T. Chen, “Dace: A scalable dp-means algorithm for clustering extremely large sequence data,” *Bioinformatics*, vol. 33, no. 6, pp. 834–842, 2017. DOI: 10.1093/bioinformatics/btw722. eprint: /oup/backfile/content_public/journal/bioinformatics/33/6/10.1093_bioinformatics_btw722/2/btw722.pdf. [Online]. Available: <http://dx.doi.org/10.1093/bioinformatics/btw722>.
- [43] B. S. Everitt, S. Landau, and M. Leese, *Cluster analysis*. Wiley, 2009, ISBN: 0340761199, 9780340761199.

- [44] K. Deng, P. G. Mehta, and S. P. Meyn, “Optimal kullback-leibler aggregation via spectral theory of markov chains,” *IEEE Transactions on Automatic Control*, vol. 56, no. 12, pp. 2793–2808, Dec. 2011, ISSN: 0018-9286.
- [45] R. Agaev and P. Chebotarev, “On the spectra of nonsymmetric laplacian matrices,” *Linear Algebra and its App.*, vol. 399, pp. 157–168, 2005, ISSN: 0024-3795.
- [46] F. R. K. Chung, *Spectral graph theory*. AMS, 1997.
- [47] D. Boley, G. Ranjan, and Z.-L. Zhang, “Commute times for a directed graph using an asymmetric laplacian,” *Linear Algebra and its Applications*, vol. 435, no. 2, pp. 224–242, 2011, ISSN: 0024-3795.
- [48] F. Takens, “Detecting strange attractors in turbulence,” in *Dynamical systems and turbulence, Warwick 1980*, Springer, 1981, pp. 366–381.
- [49] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner, “Hierarchical parallel coordinates for exploration of large datasets,” in *Visualization '99. Proceedings*, Oct. 1999, pp. 43–508.
- [50] M. Mole, “Study of the properties of air flow over orographic barrier,” PhD thesis, University of Nova Gorica, 2017.

Bibliography

Publications Related to the Thesis

Journal Articles

- L. Stopar, P. Škraba, M. Grobelnik, and D. Mladenić, “StreamStory: Exploring multivariate time series on multiple scales,” *IEEE Transactions on Visualization and Computer Graphics*, 2018. DOI: 10.1109/TVCG.2018.2825424.
- M. Trampuš, F. Fuat, D. Pighin, T. Štajner, J. Berčič, B. Novak, D. Rusu, L. Stopar, and M. Grobelnik, “DiversiNews: Surfacing diversity in online news,” *AI Magazine*, vol. 36, no. 4, 2015.

Conference Papers

- L. Stopar, “Multi-scale methodology for explaining data streams,” in *Proceedings of SiKDD 2015*, Ljubljana, Slovenia, 2015, pp. 37–40.
- M. Trampuš, F. Fuat, J. Berčič, D. Rusu, L. Stopar, and T. Štajner, “(i)DiversiNews - a stream-based, on-line service for diversified news,” in *Proceedings of SiKDD 2013*, Ljubljana, Slovenia, 2013.
- B. Fortuna, J. Rupnik, J. Brank, C. Fortuna, V. Jovanoski, M. Karlovčec, B. Kažič, K. Kenda, G. Leban, A. Muhič, B. Novak, J. Novljan, M. Papler, L. Rei, B. Sovdat, L. Stopar, M. Grobelnik, and D. Mladenić, “QMiner: Data analytics platform for processing streams of structured and unstructured data,” in *Neural Information Processing Systems*, Ljubljana, Slovenia, 2014.

Biography

The author of this thesis was born in Ljubljana, Slovenia, on August 5, 1988.

He completed the Bachelor of Science degree in an interdisciplinary academic program of Computer Science and Mathematics at the Faculty of Computer and Information Science and Faculty of Mathematics and Physics at the University of Ljubljana in 2013. Afterwards, he enrolled into the PhD programme Information and Communication Technologies at the Jožef Stefan International Postgraduate School.

His research interests include Machine Learning, Data Mining, Data Fusion, Predictive Analytics, application of Data Mining in different domains. Most of his research work is connected to integrating statistical machine learning methods to visualization with a focus on scalability.

During his PhD studies he was employed as a research assistant at the Artificial Intelligence Laboratory at the Jožef Stefan Institute, Ljubljana, Slovenia. He collaborated in a number of EU-funded projects including ProaSense (The Proactive Sensing Enterprise), COG-LO (Cognitive Logistics), RENDER (Reflecting Knowledge Diversity), AQUASMART (Aquaculture Smart and Open Data Analytics as a Service) and others.

