UNIVERSITY OF CALIFORNIA,
IRVINE

Multimodal event driven N-of-1 analysis of individual lifestyle and health

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Vaibhav Pandey

Dissertation Committee:
Professor Ramesh Jain, Chair
Professor Michael Carey
Professor Bin Nan

2021

DEDICATION

To my loving parents, mentors, teachers, and friends who have guided and supported me throughout my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Ramesh Jain, for being a great mentor, teacher, and friend. He has been the single greatest source of inspiration and motivation during my time as a graduate student. We spent many hours discussing my research, future prospects, and well-being, especially during the pandemic. His research inputs played a vital role in helping me write this dissertation, and every discussion with him left me with renewed motivation for doing meaningful work. His positive approach towards life and research has helped me gain a new perspective about myself and my place in the world. I am eternally grateful to have worked with him and hope to continue doing so in the future.

I would like to thank my committee members, Prof. Michael J Carey and Prof. Bin Nan. Mike is a fantastic thinker and systems researcher. His insights about formulating my research and dissertation were crucial and have helped me communicate my work effectively. Bin's insights and questions about the interplay between population, sub-population, and individual models added a new dimension to my research and have also given me a direction for my future research endeavors.

I would like to thank my colleagues and lab mates Nitish Nag, Dhruv Upadhyay, Ali Rostami, Hyungik Oh, and Preston Putzel for stimulating research discussion and support throughout my Ph.D. I would especially like to thank Nitish and Dhruv for their support and collaboration. Nitish has been a great friend. I truly cherish our conversations over countless meals, coffees, bike rides, and rock-climbing sessions (while I was desperately clinging for my life). Our discussions about understanding individuals and their behavior were a great help while formulating my dissertation. Dhruv made crucial contributions to the development of the event mining platform, and I am grateful to have worked with him on multiple projects.

Lastly, I would like to thank my parents for their love, support, and unwavering faith in me. They gave me the strength to believe in myself and strive to achieve my dreams. I would also like to thank my brother, sister, brother-in-law, sister-in-law, nephew, and niece for being in my life whenever I needed their support. Their constant presence gave me the determination to push through the pandemic and complete this dissertation.

VITA

Vaibhav Pandey

EDUCATION

Doctor of Philosophy in Computer Science **2021**
University of California, Irvine (UCI) *Irvine, CA*

Master of Science in Computer Science **2019**
University of California, Irvine (UCI) *Irvine, CA*

Bachelor of Technology in Computer Science and Technology (H) **2013**
Indian Institute of Technology Kharagpur (IIT KGP) *Kharagpur, India*

RESEARCH EXPERIENCE

Graduate Research Assistant **2019–2021**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant **2016–2019**
University of California, Irvine *Irvine, California*

PUBLICATIONS

Event Mining Driven Context-Aware Personal Food Preference Modeling **2020**
Vaibhav Pandey, Ali Rostami, Nitish Nag, Ramesh Jain
6th International Workshop on Multimedia Assisted Dietary Management, ICPR 2020, Milan, Italy

Personal Food Model **2020**
Ali Rostami, Vaibhav Pandey, Nitish Nag, Vesper Wang, Ramesh Jain
Proceedings of the 28th ACM International Conference on Multimedia, 2020

Personalized User Modelling for Context-Aware Lifestyle Recommendations to Improve Sleep **2020**
Vaibhav Pandey, Dhruv Upadhyay, Nitish Nag, Ramesh Jain
In Proceedings of the 1st International Workshop on Human-centric Multimedia Analysis, 2020 (pp. 13-20)

Personalized User Modelling for Sleep Insight **2020**
Dhruv Upadhyay, Vaibhav Pandet, Nitish Nag, Ramesh Jain
Proceedings of the 28th ACM International Conference on Multimedia, 2020

Continuous Health Interface Event Retrieval **2020**
Vaibhav Pandey, Nitish Nag, Ramesh Jain
Proceedings of the 2020 International Conference on Multimedia Retrieval (ICMR 2020)

Atmosome: The Personal Atmospheric Exposome **2020**
Hari Bhimaraju, Nitish Nag, Vaibhav Pandey, Ramesh Jain
medRxiv 2020

Respiration rate and volume measurements using wearable strain sensors **2019**
Michael Chu, Thao Nguyen, Vaibhav Pandey, Yongxiao Zhou, Hoang N Pham, Ronen Bar-Yoseph, Shlomit Radom-Aizik, Ramesh Jain, Dan M Cooper, Michelle Khine
NPJ digital medicine 2019

Cross-modal health state estimation **2018**
Nitish Nag, Vaibhav Pandey, Preston Putzel, Hari Bhimaraju, Srikanth Krishnan, Ramesh Jain
Proceedings of the 26th ACM international conference on Multimedia 2018

Ubiquitous event mining to enhance personal health **2018**
Vaibhav Pandey, Nitish Nag, Ramesh Jain
Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers

Endogenous and Exogenous Multi-Modal Layers in Context Aware Recommendation Systems for Health **2018**
Nitish Nag, Vaibhav Pandey, Ramesh Jain
arXiv preprint arXiv:1808.06468

Live personalized nutrition recommendation engine **2017**
Nitish Nag, Vaibhav Pandey, Ramesh Jain
Proceedings of the 2nd International Workshop on Multimedia for Personal Health and Health Care, 2017

Pocket dietitian: Automated healthy dish recommendations by location **2017**
Nitish Nag, Vaibhav Pandey, Abhisaar Sharma, Jonathan Lam, Runyi Wang, Ramesh Jain
International Conference on Image Analysis and Processing 2017, 444-452

Health multimedia: Lifestyle recommendations based on diverse observations **2017**
Nitish Nag, Vaibhav Pandey, Ramesh Jain
Proceedings of the 2017 ACM International Conference on Multimedia Retrieval

Cybernetic Health **2017**
Nitish Nag, Vaibhav Pandey, Hyungik Oh, Ramesh Jain
arXiv preprint arXiv:1705.08514

SOFTWARE

Interactive Event Mining Dashboard `https://theeventminer.com`
Web-based platform that enables exploratory analysis of events and data stream using event mining operations.

ABSTRACT OF THE DISSERTATION

Multimodal event driven N-of-1 analysis of individual lifestyle and health

By

Vaibhav Pandey

Doctor of Philosophy in Computer Science

University of California, Irvine, 2021

Professor Ramesh Jain, Chair

Population-based models are prevalent in different aspects of our lives. Decision-making in clinical health relies heavily upon the insights generated from population-based models such as recommended blood pressure and blood glucose values. These models and insights are easy to understand, but they do not necessarily capture the variance at an individual level. These are also likely to suffer from bias due to the participants' selection strategy (selection bias) or statistical limitations of the models used. Therefore, population-based models are not precise enough to make predictions at an individual level and highlight the need for personalized models for individual decision-making. N-of-1 analysis and modeling strategies are gaining popularity for addressing this problem. This modeling paradigm views every individual as a unique system and reduces variance at an individual level.

Multimodal data generated by individuals and systems can be utilized to enable such analysis. Different wearable and IoT devices and smartphone applications capture data about individual lifestyles and health as daily life events and associated data streams. These event streams and data streams create a *lifelog* for the user that captures their habits and behaviors in the form of frequent event patterns.

In this work, we propose an interactive event analysis system that leverages the multimodal events and data streams from varied sources and enables analysts to perform N-of-1 analysis for individuals. The system is based on an event patterns language [65] to represent temporal

event relationships and extends the language to allow the creation of aggregate features from event patterns. The proposed system has three major components that allow the analysts to perform N-of-1 analysis:

1. Event creation modules define new complex events from multimodal data using predefined event combination operators and motif discovery from data streams. Different event visualizations enable the analysts to explore the event space and identify event combinations for further analysis.

2. A hypothesis testing module allows the analyst to encode domain knowledge and personal beliefs in the form of a directed acyclic graph where every edge in the graph captures a causal relationship between the parameters (represented by the nodes). These relationships can then be tested using an implementation of the *do*-operator and estimate the effect of an intervention on the observed outcome.

3. A data-driven frequent event sequence detection module allows the analysts to discover frequent sequences of events and the time delays between the events leading up to an outcome event of interest. These frequent sequences describe the commonly observed events associated with the outcome and may represent a causal link between the events, which may be tested empirically or experimentally.

Chapter 1

Introduction

Knowing yourself is the beginning of all wisdom

Aristotle

The theoretical and scientific study of any phenomenon, system or situation revolves around a *model*, which is an abstract representation of the entity of interest and mimics its relevant aspects. For example, a geological map, a road map and a population density map represent different aspects of geographical entities such as cities, states and countries. Researchers in different scientific fields attempt to create models of agent behavior in different domains. There are models of cell behavior in different ecosystems, behavior of rational agents in different economic situations, etc. In this work, we are primarily discussing mathematical models, i.e. models that use the language of mathematics to describe different entities.

Models in different scientific disciplines can originate from the scientific knowledge and theories of the concerned domain (*knowledge-driven* models) or from a collection of empirical data about the entity in different situations (*data-driven* models). The former category of models derive from the existing knowledge and different assumptions in the concerned do-

main. Majority of such models clearly articulate the causal links between different properties and entities and are judged based on their capability to predict the system behavior. If a situation is observed where the model fails to describe the outcome, the underlying assumptions need to be questioned and modified until we arrive at a better model. However, knowledge-driven models rely on strict assumptions and ideal conditions that need not hold true in all situations. Therefore, these models may not be appropriate for highly complex systems such as human behavior and social sciences, where the behavior of the system depends on a large number of factors.

On the other hand, *data-driven* models attempt to replicate the behavior of the system by leveraging the data generated by the system. Advances in measurement and data collection approaches have led to a data revolution and are one of the main reasons for the rising popularity of data-driven modeling. Data-driven methods are particularly suited for domains where a large amount of data is being generated, Large scale collection of data from different systems have helped the evolution of artificial intelligence and machine learning methods and these developments have in turn improved the way experts in different domains tackle commonly observed problems. However, large scale data-driven models are difficult to interpret and are treated as "black-box" models of the underlying data generating system. This can lead to unexpected and often undesired outcomes due to unidentified biases in the collected data or the modelling strategy. Therefore, it is essential to take into account both data-driven and theory-driven approaches when designing solutions in different scientific domains. Most of the scientific community have realised this opportunity and made significant advances in combining the two approaches to address prevalent research challenges.

In this work, we propose a system to that enables data-driven modeling of a individual's behavior and health using a high level event pattern language. At the same time, the platform also allows the analyst or a domain expert to utilize their theoretical understanding of the domain to guide the modeling process. The models derived from the methodology are easily interpretable and enhance our understanding of the individual. However, it should be noted

that the proposed approach can be generalized to model any system that generates multi modal timestamped data.

1.1 Single subject modeling

The widespread adoption of commercially available sensors in form of wearable devices, IoT devices and smartphones have led to a drastic increase in the amount of personal lifestyle and well-being data generated by individuals. Numerous applications have been developed that utilize the multi-modal data streams to understand specific aspects of user behavior. However, different human activities are intricately linked with each other and studying them in isolation is only going to provide a limited understanding of the underlying processes. The first step in addressing this issue is to create a unified log of user activities and data streams.

1.1.1 Personal Data Collection

Lifelogs

Aggregating and recognizing events in our daily lives is a popular problem in the multi-media community. This problem has been termed"Lifelogging", which is explained as "a phenomenon whereby people can digitally record their own daily lives in varying amounts of detail, for a variety of purposes"[49]. Lifelogging is the first step towards recognizing various daily activities and how they define user's behavior. Lifelogging applications collect a variety of data streams about an individual, which has the potential to offer unique insights into human behavior. There are many visual lifelogging applications and projects which aim to

understand the user's life and activities by using the images taken by a wearable camera (E.g., GoPro) over a long period and the data collected could be used to identify events and daily activities happening in user's life for varied purposes[192][33][29][191].
The daily activity recognition could be further enhanced by merging the visual log with other multi-sensory data that can be collected using smartphones, wearable devices, and different IoT systems [9]. Some applications attempt to accomplish this task without using the visual logs to make the logging process more unobtrusive[135].

A complete lifelog of individual activities represents a personal chronicle of their lives, or a *Personicle*[135]. It requires a cross modal understanding of daily life events for efficient and precise recognition of human activities.

Quantified Self

Rapidly rising popularity of commercially available sensors and data-driven modeling approaches have influence many individuals to collect lifestyle, well-being and health data about themselves. This has to led to a movement called *quantified self*, comprising largely of self-motivated individuals who are sensitive towards their health and collect data about their daily lives and health outcomes.

Objective Self

Objective Self [63] takes the concept of quantified self beyond just collecting data and analyzing individual data. It aims to create an objective representation of an individual's day-today lives using the multimodal lifelogs. This representation of an individual can be used to understand and predict their behavior in different contexts, and can also be used for different applications such as food recommendation or medical diagnoses. Identifying the daily life activities (in the form of Personicle) is only the first step towards this goal. We need

to enrich the personicle with relevant event properties (such as, location, other participants in the event, effects of the event) which may not be readily apparent from personicle alone.

1.1.2 N-of-1 studies and Personal Science

Many scientific disciplines such as psychology employ single subject studies also known as *N-of-1* studies to understand individual behavior and identify best interventions for every individual independently. N-of-1 methods can be used to study highly idiosyncratic phenomena in a clinical setting. These studies involve randomized allocation of treatment to same individual on multiple occasions.

Personal Science[198], on the other hand, refers to a theoretical frameworks to facilitate analysis of self-tracked quantified self data. The problems being addressed in personal science are typically not addressed in medicine and are highly individualized in nature. The analyses are typically done by individuals themselves and tend to be not as rigorous as typical N-of-1 analyses.

1.2 Contributions

This dissertation addresses the problem of N-of-1 analysis using continuous multi-modal data and event streams. We present a domain independent event analysis framework capable of utilizing temporal data from disparate multi-modal sources. The system enables an analyst to utilize their knowledge of the domain and combine it with longitudinal multi-modal data to develop models capable of explaining system behavior. The major contributions of the system are:

1. Events as an abstract representation of real-world activities in the system. This allows

us to associate data from different sources to enrich events and add further dimensions to the events derived from single sources.

2. Event combination and creation operators that allow analysts to utilize their understanding of the domain to create new complex events that capture detailed activities in the system that are otherwise unlikely to be recognized from a single data source.

3. Designing a framework around a high-level event pattern language that enables the analysts to explore the longitudinal data. They can accomplish this by looking for associations and correlations in form of temporal event patterns, and discover unknown patterns using a novel semi-interval temporal pattern detection algorithm.

4. A proposed interactive approach for refining event patterns indexed in a tree-based event model. The analyst can interact with the tree model and filter anomalous events or event episodes that lead to spurious patterns.

5. Designed and developed a causal analysis framework based on the discussed event pattern language. The framework allows the analyst to encode their beliefs and knowledge of the domain in form of a Directed Acyclic Graph (DAG) and derive the required parameters from the event database using the event patterns and aggregate operations. This forms describes the analyst's hypothesis and can then be tested using Pearl's Causal Graphical Modeling framework[143].

1.3 Thesis outline

Chapter 2 discusses the various existing longitudinal data analysis and modeling paradigms employed in different scientific domains. We discuss each paradigm's assumptions, advantages, and disadvantages and how they can be combined in a single conceptual framework to create personalized individual models. Chapter 3 explores the related works in sequential

and temporal event pattern modeling and their applications in various domains. Chapter 4 describes the event mining concepts and definitions. We utilize an event pattern language [65] to model event relationships and extend the patterns with groups and aggregate operations. We discuss different event mining operations and system architecture for exploratory event mining analysis. Chapter 5 describes a causal analysis framework that allows analysts to test their beliefs about causal relationships between events. The known causal relationships can be encoded in the form of a DAG that also includes the relationship being tested. The variables in the hypothesis are described in the form of event mining expressions and can be computed using the events in the database. We can apply Causal Graphical Modeling principles on the data set and the causal structure to replicate the *do*-operator[145]. Chapter 6 describes a novel tree-based event episode indexing methodology that captures frequent temporal semi-interval event patterns as different branches of the tree. The tree model is used to find frequent semi-interval patterns, and the proposed algorithm is evaluated using a simulated data set with controllable error rates. Chapter 7 discusses the applications of different aspects of the proposed platform for studying problems in personal health and for deriving an explainable rule-based personal model. We demonstrate how we can use the event creation operators to extract events described in health and biomedical literature that have a causal impact on the user's health. We utilize the proposed causal analysis approach to understand a user's sleeping behavior in different situations and identify the effect of various environmental and behavioral factors on sleep quality metrics. We also investigate the impact of varying endurance training metrics derived from cycling events on performance. Lastly, we also demonstrate the use of event pattern language to find a user's taste preferences for food items in different environmental and behavioral contexts. We leverage a novel taste space definition and estimation method, which is discussed further in the appendix.
Chapter 8 concludes this dissertation and discusses the possible improvements and future works that allow the N-of-1 modeling paradigm to be used for generalized personal modeling.

Chapter 2

Understanding individuals: Longitudinal modeling paradigms

Observing and collecting data about a system's behavior is essential for understanding and predicting its behavior. We need to either observe the system under varying conditions or have prior knowledge about the system's behavior to make any predictions. Longitudinal modeling has been employed for studying the degree and direction of changes in dynamic systems in multiple scientific domains. Therefore, it is crucial to understand different longitudinal modeling paradigms and identify those valid for the research question under consideration.

Longitudinal studies are commonly observational and employ continuous or repeated measurements about the individual (system) over a long period. These types of studies are prevalent in medicine, psychology, and behavioral analysis and are particularly useful for evaluating the relationship between risk factors and the development of disease and the outcomes of treatment over different lengths of time. Similarly, because data is collected for given individuals within a predefined group, appropriate statistical methods may be employed to analyze change over time for the group as whole or particular individuals. Thus

longitudinal studies are very well suited for studying long-term phenomena and the effect of time on variables.

There are different types of longitudinal studies, each designed for a specific research goal. For example, 1) *Panel Studies*, where data is collected from the same random sample of individuals periodically, can be helpful to track the progression of individuals in the study and identify the changing sensitivity of individuals to particular stimuli, and 2) *Retrospective studies* where participants recall their past behavior and data at the time of data collection. Many longitudinal studies have led to seminal developments and discoveries in different scientific domains. Framingham Heart Study [102, 12] is one of the most well-known longitudinal studies and has led to a large number of discoveries about risk factors such as aging, smoking, diabetes, and hypertension for different cardiovascular diseases[115, 42, 17]. Similarly, the Grant Study from the Harvard Medical School [182] studied the physical and emotional development of individuals over eight decades and found significant correlations between lifestyle and personal factors (such as alcoholism, nature of personal and parental relationships, and intelligence) with life outcomes (such as financial success and "life satisfaction").

Despite their many successes in identifying relationships between variables, these studies have faced much criticism, especially related to selection bias[104] as the cohorts in these experiments are limited, and the insights gained from such experiments may not be easily generalized to populations with different genetic, social and cultural traits. Long-term longitudinal studies have traditionally been costly as repeated data measures over a large sample would require significant infrastructure and human costs. Historically, the cost factor has forced the researchers to limit longitudinal studies to a small cohort that introduces various biases in the study. However, advances in sensor technology, the Internet of Things, smart devices, and internet connectivity have led to an explosion of personal data generated by individuals. These devices capture continuous data streams about different user behaviors in a passive manner. The passive data collection and sensing modalities can create a more

accurate and comprehensive record of a user's day-to-day life. This presents a unique opportunity for researchers to conduct large-scale longitudinal studies as significant portions of the data collection infrastructure are already in place. The cohorts for the new age longitudinal studies can be scaled to large populations with minimal one-time investments. Therefore, data management and computational platforms are needed to ingest and process continuously generated sensor data for large populations. Ensuring data security and user privacy is one of the most critical challenges in such systems. These platforms should also incorporate different modeling and analysis techniques to help the analysts understand the multimodal longitudinal data. We attempt to address this problem in this chapter by reviewing the existing longitudinal modeling paradigms and their applications and limitations in different domains. We propose a novel theoretical longitudinal data analysis pipeline that takes advantage of the strengths of different approaches and produces personalized n-of-1 models that leverage existing scientific understanding.

2.1 Longitudinal modeling paradigms

There are three major modeling strategies used for longitudinal studies. 1) *Variable-centered analysis*, 2) *Person-centered analysis*, and 3) *Person-specific analysis* [56]. These three approaches differ in many aspects, ranging from data collection methods and frequency, sample size, and computational methods. However, all these differences stem from the assumptions about the population and the individuals.

We will examine the differences between the three methods regarding data requirements, assumptions about the data and the models, and the utility of the models derived from the experiments. We utilized two simple metrics to understand these differences *Parsimony* and *Specificity*.

Parsimony refers to the complexity of a model and can be defined as the number of pa-

rameters needed to describe the model. The parsimony of a model is generally inversely related to its interpretability, i.e., parsimonious models (models with fewer parameters) are generally easier to understand for human experts. For example, parameters or weights for different features in a linear regression model are more interpretable than the parameters of a neural network model with multiple hidden layers.

Specificity refers to the precision of the model in describing the subjects in the study. This metric is directly related to the robustness of the relationships captured by the model and if the generated insights are valid for a general population. Typically, the two metrics are inversely related, and the three longitudinal modeling approaches lie on different regions on the axes of parsimony and specificity (shown in fig. 2.1).

We will examine the modeling strategies along these two axes while keeping in mind that none of the approaches is the *best* approach. Different modeling strategies have their strengths and weaknesses, and the ideal longitudinal model would utilize the approach (or a combination of approaches) that best suits the research problem at hand.

2.1.1 Variable-centered analysis

Variable centered analysis is the predominant modeling approach in different scientific domains. It helps understand the relationships between variables of interest in a population and is appropriate for testing the hypotheses or research questions concerning the effects of one variable on another. This paradigm views variables as the primary agents and objects of change, while individuals are the medium in which the variables interact [85]. The data are typically collected from multiple subjects over one or more occasions, and the number of samples required for analysis depends on the degree of correlation and the variance in the data collected. The sample size can vary from as low as 30 to tens of thousands, and common correlations are identified across the sample to summarize the population with a

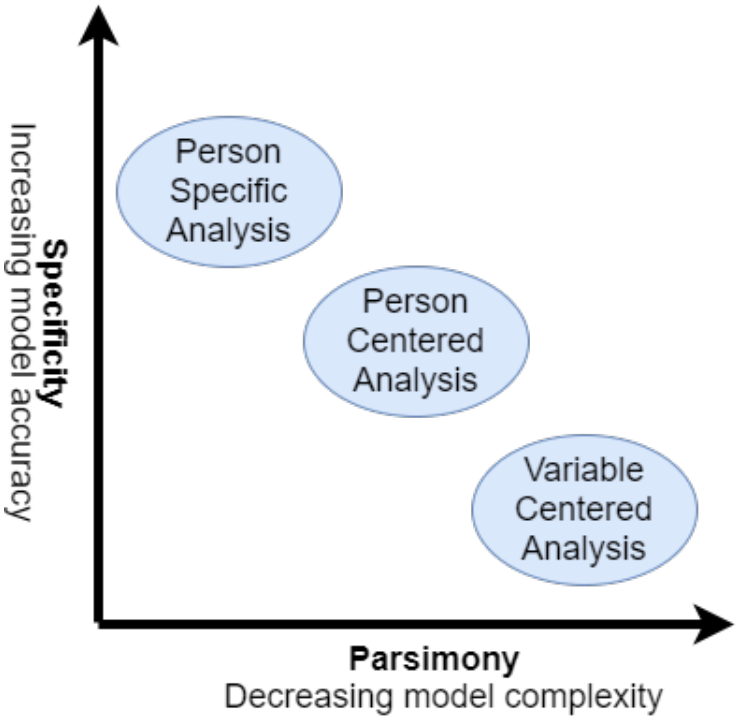

Figure 2.1: Parsimony vs Specificity for longitudinal modeling paradigms

single set of parameters.

We can illustrate the utility and the theoretical differences between the three approaches through an example of a researcher studying sleep and job performance. A vital question for this researcher would be, what are the emergent dimensions of sleep quality, and how are the different dimensions of sleep quality associated with job performance? These questions are focused on variables and relationships between them and suggest a variable-centered approach. The researcher could start by collecting data from multiple participants and using factor analysis to identify the latent dimensions of sleep quality. Similarly, they could utilize regression analysis to describe the sleep quality-job performance relationship for the sample and, by extension, the population.

The main differences between the three approaches stem from the assumptions concerning the relationship between the population and individuals. This approach assumes that the population is *homogeneous* concerning the causal relationships in the members of the population, i.e., all members in the population have identical underlying causal mechanisms,

and the differences between the samples are due to differences in the values of treatment, covariates or random noise. However, this assumption is not necessarily valid, especially when we are studying a population of humans.

Numerous studies have successfully leveraged variable-centered analysis to identify generic relationships between variables. However, this approach fails to capture relationships (and variations in general relationships) valid only for specific sub-populations in the sample. Since the entire population is summarized with a single set of parameters (for example, regression coefficients), variations in the underlying generative causal processes in different population segments are not adequately modeled. Therefore, the derived model is inadequate for explaining data from subpopulations with different traits with high accuracy. The resultant model is easy to understand (high parsimony) but has poor accuracy in different segments of the population (low specificity).

2.1.2 Person-centered analysis

Person centered analysis helps us understand the different categories of trajectories present in the collected longitudinal sample. The goal of this analysis is to identify the optimal number of sub-populations within the sample such that the associated finite set of parameters yield an accurate summary of the population[84, 107]. Therefore, this approach helps investigate hypotheses aimed at 1) categorizing individuals into sub-populations based on observed variables and 2) understanding the relationships of these sub-populations with treatment, covariates, or the outcomes [56]. In contrast to variable-centered analysis, this approach views variables not as agents and outcomes but as properties of individuals and the environment. These properties may have varying causal relationships for different types of individuals (sub-populations), and the goal is to identify these subpopulations and their progression concerning the variables of interest.

The data collection protocols for this type of analysis are similar to those used for variable-

centered analysis; data are collected from multiple subjects across one or more occasions. However, smaller sample sizes that may be appropriate for variable-centered analysis can potentially lead to convergence issues for smaller sub-groups[184].

Continuing with our previous example of studying the relationship between sleep quality and job performance, the researcher might be interested in understanding the relationship between the two metrics for different professions (day shift vs. night shift, white collar vs. blue collar, etc.). Is the effect of sleep latency (one of sleep quality dimensions) on performance different in different sub-populations? These research questions suggest the existence of identifiable sub-populations and hence indicate that the researcher should use a person-centered approach. The researcher could collect data from multiple participants over multiple occasions and identify sub-populations based on a measured parameter (e.g., profession) or finding representative clusters from collected data (e.g., movement/sleeping behavior patterns). They could then utilize an appropriate modeling algorithm for the research problem (whether the goal is to understand the progression of sub-population or high-level understanding between variables within the sub-population).

This approach relaxes the assumption about the homogeneity of data, as we are acknowledging the existence of fundamentally different individuals within the population and attempt to describe these sub-populations using a separate set of parameters. Therefore, the resulting model has better explainability (or specificity) over the population than the previous approach. However, this also means that the model is relatively more complicated and scores lower on the parsimony scale, and may not be as easy to interpret in the context of the domain (depending on the methods used).

2.1.3 Person-specific analysis

Person-specific analysis is usually employed to understand and investigate phenomena that are idiosyncratic to specific individuals. Person-specific analyses are best suited to hy-

potheses focused on individuals, and analyses that aggregate across samples to make inferences about variables or sub-populations are insufficient to address these research questions. It can also model phenomena that change too frequently or irregularly between individuals and cannot be captured using the previous approaches. These approaches provide the researcher with multiple models that describe each of the subjects in the sample. Inferences are meant to be specifically for the person and not the population. Person-specific analysis adopts a *holistic interactionist perspective*[181, 74], which [178] describe as "an individual's prior behaviors, genetic makeup, and contextual risk or protective factors operate as an integrated whole; taken in isolation, they may lose their meaning and consequence for that individual's behavioral course." Therefore, this approach views individuals as an integrative and complex system and the statistical methods need to embody the same assumptions.

Data for such analyses are typically collected from a small number of participants (as small as one) over a large number of occasions. The effect of the number of occasions on the statistical analysis is similar to that of the number of participants in variable and person-centered analyses.

A researcher or a professional in a clinical setting may want to find the emergent dimensions of sleep quality for a specific individual. What is the relationship of these sleep quality metrics with the individual's job performance metrics? Since the hypotheses are specific to an individual, person-specific analysis should be used to test the hypotheses. The researcher could collect data from the single participant over many time points (collected once a day spanning multiple weeks). They can identify the emergent dimensions of sleep quality using factor analysis and correlate each of these dimensions with job performance. They can also attempt to control confounding variables by capturing other environmental and behavioral attributes for the subject.

Since the person-specific analysis views an individual as a distinct system encompassing their behavioral and environmental history, it does not make any assumptions about the global relationships between the variables in the population or the sub-population. Therefore, it

relaxes the population homogeneity (variable-centered analysis) and sub-population homogeneity assumptions (person-centered analysis). On the other hand, due to lack of assumed generality, this analysis typically requires significantly more measurements than the variable or person-centered analysis.

The models derived from person-specific analysis tend to have better specificity than the other two approaches but also have the least parsimony, as each individual is described with a separate set of parameters. Thus the existing person-specific analysis methods generate models that best describe the individual but can be relatively difficult to interpret.

	Variable-Centered	Person-Centered	Person-Specific
Goal	Summarize the population using a single set of parameters	Identify similar groups in the population. Summarize each group using a set of parameters	Summarize each individual using a set of parameters
Data Collection	Multiple individuals, few occasions	Multiple individuals, multiple occasions	Single individual, many occasions
Strength	• Identify generic relationships valid for the whole population • Simple and easy to interpret	• Identify different subgroups having different causal mechanisms within the sample • Better specificity compared to variable-centered analysis	• Models to explain behavior of every individual in the study • Better specificity than the other two analyses
Weakness	• Not specific enough for heterogeneous groups in population	• Not specific enough for individual-level inferences • Not as parsimonious as variable-centered analysis	• Least parsimonious of the three approaches

Table 2.1: Comparison of the three longitudinal modeling approaches

2.1.4 Equivalence of approaches

While we have highlighted the differences between the three approaches, there are certain conditions under which these can provide identical results. Collectively, these conditions are known as *Ergodicity*. Ergodicity refers to the idea that a stochastic or a dynamic process will eventually reach all possible states in a uniform and random fashion. Equivalently, a large enough random sample from the process is sufficient to specify the process in its entirety. In the context of longitudinal studies, this implies that the sample is entirely homogeneous (all individuals are exact replications and interchangeable) and stationary (constant mean, variance).

However, this constraint is too restrictive for most real-world situations, especially when trying to model human health and behavior. Individuals are different due to variations in their genetic makeup, upbringing, and environment, making any human-generated data non-homogeneous in the population. Additionally, most individual health and behavioral mechanisms change over time. People undergo similar developments as they age, but they also exhibit short-term changes in many aspects due to their specific genetic and lifestyle traits. Both of these changes violate the stationarity assumption. Therefore, it is reasonable to assume that the three approaches are unlikely to produce identical results, and due to the nature of the analyses, the person-specific analysis is likely to produce the most accurate models at an individual level even though the results may be difficult to interpret[118, 3]. Thus, if our goal is to find more generic relationships, we may want to use a variable or person-centered analysis. However, a person-specific analysis should be used if specificity is vital to the researcher (if the goal is to use the model for a personal recommendation, specificity/accuracy is the most crucial factor).

2.2 N-of-1 modeling

Clinical or biomedical research conducted on one subject or individual is often called a single-subject, single-case or **n-of-1** study [24, 171]. N-of-1 analyses consider an individual as the sole unit of observation while studying the efficacy or side-effects of different interventions. It builds on the underlying philosophy and statistical methods of the person-specific analyses[90]. Multiple studies in health and psychology argue that only the person-specific approaches provide accurate results for individuals when the data are not ergodic[117, 178, 20]. Therefore, we need to utilize person-specific analysis to create personalized, explainable models with a high degree of specificity.

N-of-1 analysis has been utilized to varying degrees of success in different clinical trials and behavioral and biomedical research[45, 75, 114] such as nutrition [80], psychology[81] and oncology[37]. N-of-1 studies can be randomized experiments (N1RT) with well-defined methods to derive causal effects of treatments, or observational (N1OS) where we collect observational data without any randomization of treatment. Multiple causal inference frameworks can be applied on such observational data to find possible cause-effect relationships that can be further tested in an n-of-1 randomized trial [24].

N1OS share many traits with person-specific analysis. These provide more details with greater accuracy about individual responses to treatments and have been used in clinical practice to identify the best treatments and treatment protocol for individuals [114]. However, there are many barriers (both statistical and logistical) we need to overcome to effectively utilize n-of-1 analysis [82] for modeling individual behavior and health. There are multiple guidelines published by researchers in different scientific domains[4, 173, 189] as well as by the United States Department of Health and Human services agency for Healthcare Research that discuss these challenges and potential solutions in detail. We will discuss some of those challenges briefly.

- **Auto-correlation**: N-of-1 data is inherently in the form of a time series. Time series data generated from stochastic processes tend to have repeated patterns, and the future values generated by the process depend on the past values. This temporal relationship between data is unique to longitudinal data sets, and statistical methods commonly employed in randomized trials fail to account for it. This can introduce a bias in our estimate of the treatment effect. Any inference method applied for n-of-1 analysis needs to adjust for the auto-correlation when estimating the treatment effect.

- **Carry-over effect**: The impact of many interventions (especially in health and behavioral sciences) may have a lasting impact. Thus, it is very likely that the lingering effects of a treatment or intervention may be observed for some time event after it has been removed and influence the data being collected. Researchers conducting any n-of-1 study need to consider this carry-over while analyzing the data and interpreting the results. Multiple methods remove this effect either by analytical means or using the design of the study.

- **Slow effect onset**: This refers to the situations when the effect of a treatment may not be apparent immediately[31]. Slow-onset is not as well studied as other analytical challenges in n-of-1 trials. The current best approaches to handle the slow onset effect originate not from data but the understanding of the domain. Slow-onset and carry-over effects highlight the necessity of modeling temporal relationships between events in an n-of-1 trial.

- **Data collection**: Ecological Momentary Assessments (EMAs), surveys, and clinical measurements have been the traditional data collection tools for research studies. However, these are not the best data collection tools for n-of-1 studies as the data needs to be collected repeatedly, and these tools place the burden of data collection on the user. Thus, user adherence is likely to be affected negatively. This issue can be largely solved by multi-modal sensors and IoT devices and will be discussed next.

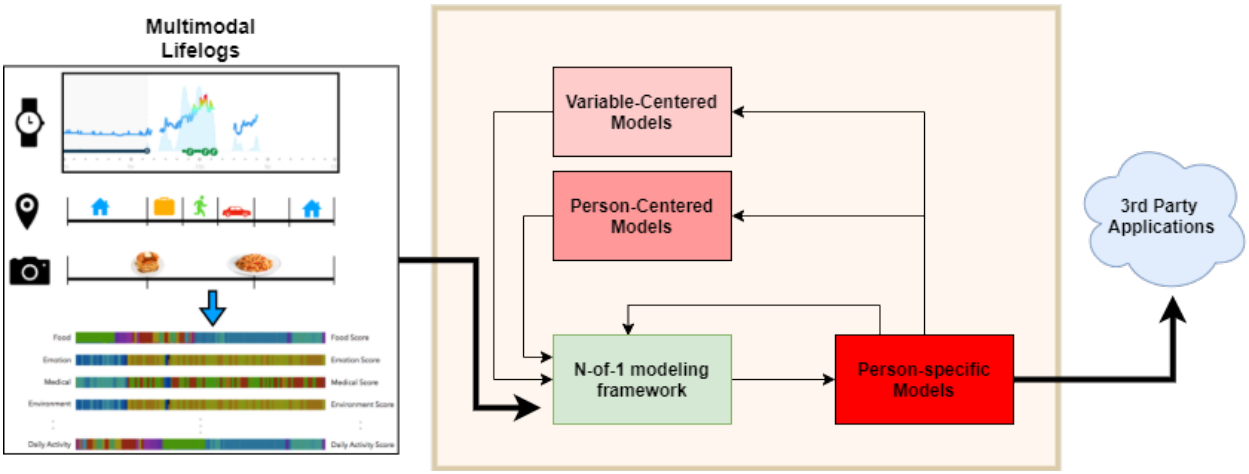

Figure 2.2: A conceptual framework for N-of-1 modeling using multi-modal longitudinal data. We can utilize the three longitudinal analysis approaches to create a N-of-1 modeling framework capable of deriving personalized models that are interpretable.

Various statistical methods have been employed by researchers to perform n-of-1 analysis, though there is no consensus among researchers as to what is the best approach. Researchers from psychology and health have used t-tests (paired and unpaired), Cohen's d, cross-correlation analysis, regression analysis, and simulation modeling analysis.

Vieira et al. [186] explore the use of dynamic modeling for n-of-1 analysis. This method allows researchers to use multivariate regression models for longitudinal analysis by including lagged variables as features in the model. Daza et al. [24] explore the application of Neyman-Rubin-Holland counterfactual framework on self-tracked n-of-1 observational data. The approaches discussed in this section are most closely aligned with person-specific analysis as the goal is to understand and describe a person's behavior with the highest accuracy. However, these studies typically require more data points than conventional multi-candidate trials due to temporal associations in the data. Therefore, we need to augment the person-specific models with variable or person-centered models. This will reduce the size of data required for the model and help prune some idiosyncratic but spurious correlations that can be explained by more generic relationships captured by the population-based models.

2.3 Personal Longitudinal data collection

Multimodal longitudinal data captures different facets of an individual's behavior over time. Consumer-grade wearable devices and smartphones accurately capture lifestyle and health data with minimal user participation required. Many biomedical and behavioral researchers have successfully leveraged this data in their experiments and gained unique insights into user behavior. Mobile health (mHealth) applications and platforms have significantly impacted cardiology, diabetes, mental health, and even infectious diseases[200, 158] research, diagnostics, and clinical practice. The WHO has published multiple reports and guidelines about the impact of mHealth applications and how to best leverage them for public health issues in different parts of the world.

The success of mHealth applications has primarily been due to the advances in sensor technology, connected smart devices such as wearable devices, smart scales, and smartphone applications such as Rescuetime, MyFitnessPal, and SleepCycle. The widespread adoption of these devices has led to an era of unprecedented human data generation. These devices and applications allow individuals to keep track of their behavior and habits. They collect continuous streams of data generated by human activities. GPS and locations services capture our location stream; accelerometers can capture our movement streams and patterns; microphones and light sensors can collect continuous data about our surroundings in the form of light and ambient noise. These human activities are represented by events that can appear as repeated signals or motifs in the data stream. Classification and retrieval techniques are commonly utilized to identify such events.

Kahneman et al. recognized 23 different categories of life events commonly observed in individuals and described them in their work on the daily reconstruction method[71]. A significant body of computational research addresses segmenting an individual's day into life events using multimodal signals and multimedia cues. Initial attempts to create a continuous log of an individual's life used visual and auditory information as the primary data[30].

Latest research efforts in this direction also leverage other multimodal signals as well as user input in the form of prompted EMAs[135] in an attempt to recognize the life events. The goal of segmenting a user's day is to attach semantic significance to each observed event and keep a log of user activities. This enhances our understanding of user's habits beyond just a quantified log and obtains an objective understanding of their lives[64].

The data streams attempt to capture information about events in our daily lives; however, every data stream can only capture a facet of human life. Westerman et al[197] and Xie et al[205] identify six aspects of multimedia events. Each aspect answers one of the W5H (what, where, when, who, why, and how) questions about the event. Structuring event-related information in this fashion allows us to reason on them and answer most queries related to events and event patterns. The multi-faceted event model allows us to attach multiple semantic values to the same event and lays the foundation of the event mining research. The event patterns, when complete, represent an individual's habits and are correlated with their situation. The goal of n-of-1 multimedia event mining is to arrive at a set of event patterns that describe the person's behavior and provide a model of the individual that applications can use in different domains such as health, finance, travel, etc.

2.4 Complex Events Processing

Complex Events Processing (CEP)[97] provides a mechanism for processing events and recognizing patterns in an event-driven system. Such platforms have utility in different domains ranging from financial services, distributed enterprises to health care, where the cyber-physical systems have a hierarchical structure. CEP platforms are capable of ingesting and processing events originating at different layers of operation. For example, in a distributed enterprise system, several network-level events (e.g., sending and receiving packets) constitute an application (API) level event (e.g., API calls). Several API level events constitute

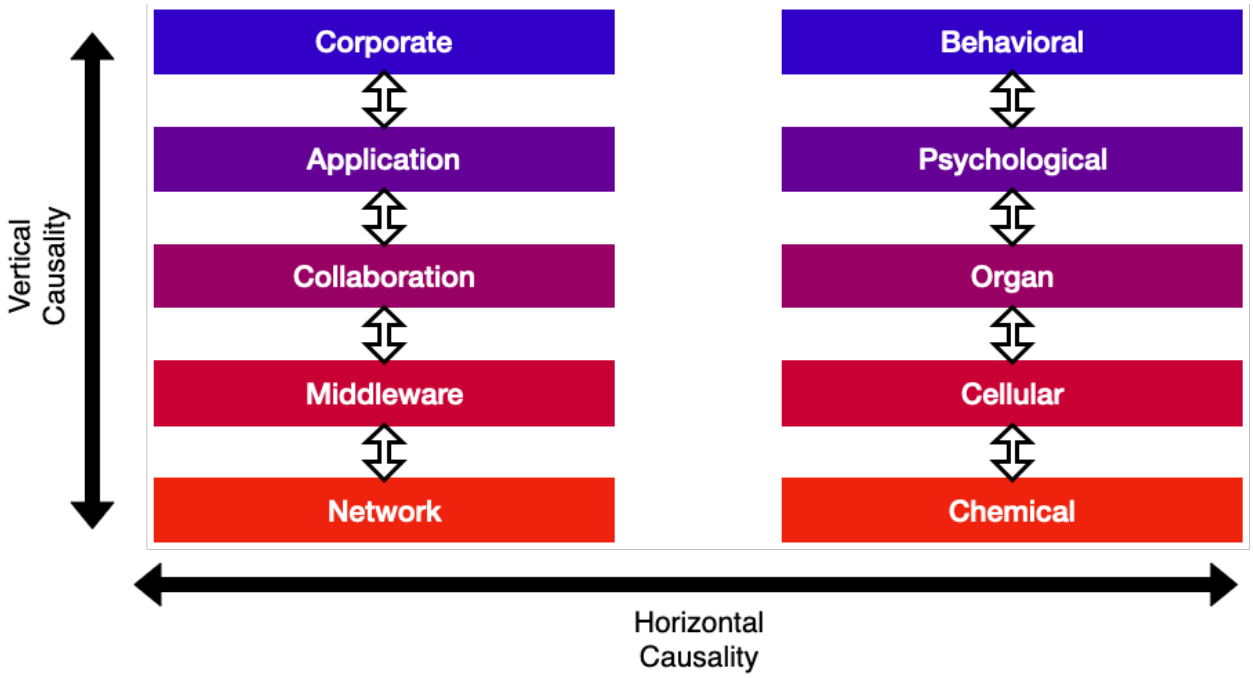

Figure 2.3: Complex Event Processing platforms allow us to model complex, event-driven systems that are divided into several hierarchical layers. These systems help identify event patterns, associate triggers with specific patterns and events, and maintain the chain of causality between events. These systems describe two types of causality between events, 1) **vertical causality** where events in different layers are causally related, and 2) **horizontal causality** where events in the same layer are causally related.

a business layer event (e.g., placing an order). Similar layered structure can also be seen in human and biological systems where layers can consist of chemical events representing essential biological reactions to behavioral events representing an individual's actions in the world (fig. 2.3). The event patterns represent an abstraction of the known data generating processes in the system, and additional statistical and data mining operators can be defined to discover and verify new patterns to be added to the model.

These platforms require an event pattern language suitable for the application domain to define event patterns and complex events as a function of simpler (lower hierarchy) events. These patterns also describe causal relationships between the events defined in different layers of the system, also referred to as **vertical causality**[96]. For example, in an online marketplace, placing an order is a user-level event that consists of several application-level events such as checking the availability of the product, checking the shipping status for the user's location, communicating with the financial services' server for payment, and notifying the user of the placed order and shipment details. Thus, the user-level event (placing an order) is described as a pattern of various application-level events, and if any of the constituent events for the pattern are missing, the user-level event does not occur. This concept also helps with event-aggregation and abstraction as it allows us to group recurring patterns of events as a new and meaningful event.

On the other hand, causal relationships between events in the same operational layer represent **horizontal causality**. Horizontal causality enhances our understanding of a single layer in the system and allows us to predict the system's behavior. Horizontal causality can also be represented using event patterns; however, these patterns may not necessarily represent an event abstraction. Various statistical methods can be employed to find candidate event patterns that may represent horizontal causal relationships. Events that signify related activities can happen at different times and appear in the event logs separated by many unrelated events. This is especially the case when the lower layer events are being generated due to a higher-layer event, but the vertical causal relationship is still unknown.

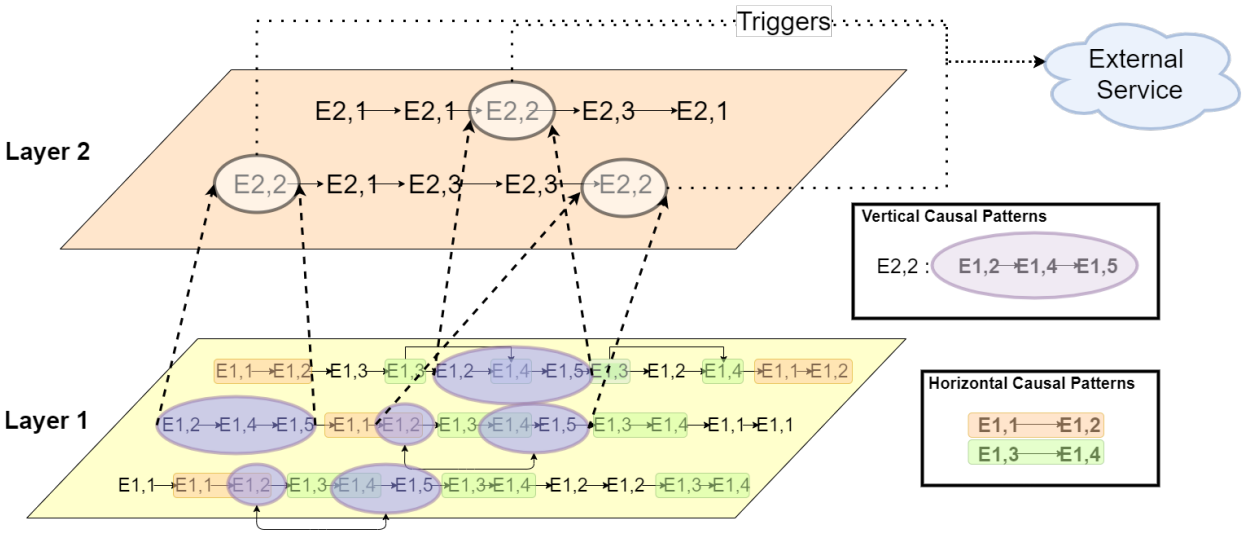

Figure 2.4: Event patterns can be used to represent vertical and horizontal causality in event-driven systems. Vertical causality patterns can also contribute to event abstraction and translate low-level system events to high-level user interactions. Horizontal causality patterns can enhance discovery of vertical patterns and make predictions about system behavior at the same level. Triggers can be associated with occurrences of various events and patterns that help coordinate actions with third party services.

Thus, finding horizontal causality between such events may also help us discover new event abstractions.

The event patterns capturing vertical and horizontal causal relationships provide an easy-to-understand model of the underlying processes. The patterns also help keep track of the chain of causality among event occurrences by populating the causal aspect of an event with events that match the corresponding patterns. The causal aspect thus enhances our understanding of the system by describing allowing us to find the relevant causes for an outcome.

CEP platforms can also be helpful for external applications that depend on the context of the system. Different triggers and operations can be tied to occurrences of both low and high-level events and patterns. For instance, a third-party sleep management system may generate a bed-time notification using lifestyle events patterns that capture horizontal relationships between work hours, meals, and sleep events. The required routine may be initiated by a trigger associated with the pattern. Therefore, the CEP paradigm satisfies many of the design requirements for the N-of-1 event mining systems and lays the computational

foundations for such a framework.

2.5 Design requirements for N-of-1 event mining framework

We have identified a set of requirements for an N-of-1 event mining framework for building personalized models based on the commonly used computational paradigms, nature of data, and the goals and use cases of such models.

2.5.1 Data fusion

Different modalities of data capture different aspects of the behavior of any system and are especially valuable when studying the behavior of individuals and biological systems. Therefore, any system that aims to model and understand individual behavior must ingest multi-modal data from disparate sources. Events provide a uniformly structured abstraction over such data, and hence a CEP platform is naturally able to reason with multi-modal data provided we have robust and comprehensive event detection mechanisms in place.

2.5.2 Knowledge integration

As we discussed in this chapter, variable-centered models are not accurate enough for individual-level inference. However, this does not mean that we cannot utilize the results and insights from such models. These models usually capture relationships and dependencies between variables that are true for the general population. Using these relationships for person-specific analysis would improve the interpretability of the models and help remove

spurious correlations that these causal relationships can explain. Therefore, any individual-level modeling system would benefit from providing a mechanism for including knowledge and insights from external sources.

2.5.3 Pattern discovery and spurious pattern filtering

Data-driven pattern discovery is necessary for discovering common behavioral or operational patterns for any system. We can utilize well-known pattern discovery and unsupervised clustering algorithms for identifying such patterns. However, many such patterns are likely to represent spurious correlations, and therefore it is vital to have a mechanism in place to identify and filter such patterns. This can be done by keeping an expert in the loop or using external knowledge sources to verify the causal significance of such patterns.

2.5.4 Interpretability: Causally significant patterns from observational data

Interpretability is a very desirable trait in models representing dynamic systems. A model is a representation of a system and should enhance our understanding of the same. Therefore, the model derived from data must provide accurate predictions and capture causal relationships present in the system. Thus, the N-of-1 event mining platform should be capable of verifying the causal validity of derived patterns.

2.5.5 Reusable individual models

The derived models should be an abstract representation of the individual behavior and, thus, be application independent. Therefore, any external services that can benefit from an

understanding of the user's behavior should be able to use the models. However, ensuring user privacy is a critical factor here, and more research is needed to identify secure and privacy-preserving protocols for such model sharing.

Chapter 3

Literature Review

Event sequence data are found across a wide variety of fields and applications. Applications in domains as diverse as distributed systems, advertising, online marketplaces, and healthcare generate discrete data over time intervals and are arranged in sequence based on the entity associated with the data. For example, most websites record user activity in the form of events representing user interaction; network logs capture a timestamped sequence of events that describe network activity at different layers. Similarly, electronic health records and different wearable and IoT devices capture events about individuals' health over time.

The ubiquity of events is due to the ease of capturing data and the generality and flexibility of modeling with it. Events allow us to easily incorporate data from varied sources, modalities, and semantics, thus allowing us to understand the behavior of any system from multiple facets and address problems at different scales. However, the utility of this data can also be challenged by significant heterogeneity in events, event properties, sequences, and the goal of analyses. The events could be high-dimensional or low-dimensional, and the data could be temporally sparse or dense. Similarly, the goal of events analyses is as diverse as their real-world applications. Analysts could find the parameter variations in the same event type or patterns of past events that impact future event occurrences. They might want to find

event patterns that describe system behavior or anomalous patterns and their causes.
The diversity of data and challenges events and event sequence analysis have led to a broad range of research efforts that aim to solve one or more aspects of the problem. We will discuss different types of event sequence analysis problems and how different representations address these problems. We will also discuss various visual analytics systems that utilize event sequences and patterns for causal and associational analysis of temporal data.

3.1 Temporal Knowledge Structures

Different systems and underlying processes generate heterogeneous event streams captured in different forms appropriate for the analysis. The events are typically stored in a unified events database and represent a log of the system activity. These logs can be leveraged to understand aspects of system behavior and identify critical situations. Researchers in various domains have addressed this problem and have derived entities that structure the insights and knowledge gained from analyzing the event sequences.

3.1.1 Events

Point events

Temporal Point Events represent instantaneous activity in a system and are represented as a timestamp and an event label, (e, t). Some examples of such events are network logs of timestamped API requests, fall detection using accelerometer signal, accident or collision detected from surveillance camera videos or social media posts generated by users.

Interval events

Interval events represent activities spanning over a continuous time interval and are represented by a beginning and end timestamp. Examples of such events are, running events captured using wearable devices, social events marked on a calendar, etc. Computational systems represent these events as a tuple of start and end time and the event label, $[e, (t^+, t^-)]$.

Semi interval events

Semi interval events are capable of representing both point and interval events in the same representation scheme. Every instance in the events database represents either a point event, $[e, t]$, or one end of an interval event, $[e^+/e^-, t]$. Thus, semi-interval events allow us to reason with point and interval events in the same framework and offer more flexibility when describing temporal patterns, which will be discussed in the next sections.

We will utilize one of the above-described event representations when discussing event patterns and sequential analyses algorithms and methodologies.

3.1.2 Temporal Patterns

Temporal patterns capture the relationship between events in the system. Earlier works in defining patterns included operators that considered the relative order of events. Allen et al.[11] laid the foundations of describing complex relationships between temporal intervals using 13 operators describing different possible configurations of two intervals(fig. 3.1). Freksa et al. [43] described similar relationships for interval end-points or semi-intervals. A large volume of pattern mining research utilizes these relations to represent event relationships.

Patterns are labeled as significant based on significance measures relevant to the analysis.

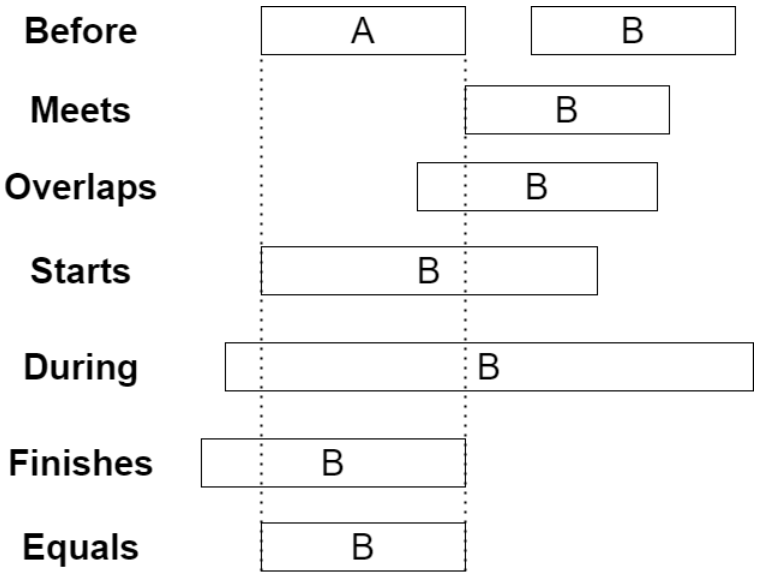

Figure 3.1: Examples of Allen's interval relationships between two interval events A and B. First six relations can be inverted.

Most commonly used significance measures are *Support*, *Confidence* and *Lift*. Support is the most commonly used significance measure for discovering frequent patterns.

Support for a pattern p in a sequence database S can be defined as the proportion of sequences where the pattern appears.

$$Support(p) = \frac{\{|s|s \in S \wedge p \sqsubset s|\}}{|S|}$$

Confidence of a rule captures how often a rule is true. For a rule of the form $X \rightarrow Y$, the confidence can be defined as

$$Confidence(X \rightarrow Y) = \frac{Support(X \cup Y)}{Support(\cup)}$$

However, these are typically used to capture frequent patterns or rules and a variety of significance measures are needed to capture different types of patterns such as anomalous patterns or surprising patterns[109][204].

Sequential Patterns

Sequential patterns and pattern mining algorithms were first introduced by Agrawal and Srikant [8] for discovering frequent subsequences as patterns from a sequence database. Utility of these patterns can be illustrated using the transactions data generated from an online marketplace, where each transaction consists of items purchased and the timestamp of the purchase. Such patterns can describe customers buying a dining table and chairs in one transaction, also purchase curtains in a subsequent transaction. Sequential Pattern Mining mainly focuses on finding patterns across multiple transactions in sequential order. Similar sequential patterns can also be found in other situations, such as web usage data describing a person's browsing behavior and genome sequence data encoding the elements of DNA sequence. We can now define some concepts related to sequences and sequential patterns.

Let I be a set of items or symbols $I = \{i_1, i_2, ..., i_m\}$.
An *itemset* X is a set of items such that $X \subseteq I$. The notation $|X| = k$ denotes the cardinality of X, ie the number of items in X. Length of an itemset (k), X, is defined by its cardinality.
A *sequence* is an ordered list of itemsets $s =< I_1, I_2, ..., I_n >$ such that $I_k \subseteq I, \forall 1 \leq k \leq n$.
A *sequence database* (SDB) is a list of sequences $SDB =< s_1, s_2, ..., s_p >$. A sequence $s_a =< A_1, A_2, .., A_n >$ is a *sub-sequence* of $s_b =< B_1, B_2, ..., B_m >$ if and only if there exist integers $1 \leq i_1 \leq i_2 \leq i_3 \leq .... \leq i_n \leq m$ such that $A_1 \subseteq B_{i_1}, A_2 \subseteq B_{i_2}, ..., A_n \subseteq B_{i_n}$.
Sequential patterns mining aims to find interesting subsequences in a sequence database (as defined by one of the pattern interestingness measures).

Fully dependent Patterns or d-Patterns

Fully dependent Patterns or *d-Patterns* [89, 88]refer to patterns that have low support but high specificity. Such patterns could be predictive of rare events but cannot be discovered using traditional mining algorithms as the patterns are infrequent. However, these patterns

could also be of great value to researchers trying to understand a system's behavior. Such patterns may also predict undesirable situations like node failure in a distributed system or a power grid service disruption. Setting a low support threshold may allow us to find such patterns but will also result in a large number of spurious patterns which hold no significance for the system. The rare event patterns may also be difficult to mine due to noisy data collection or human errors. Therefore, to avoid these issues [89] use hypothesis testing for dependency test instead of a minimum support threshold.

Periodic Patterns

Different systems and entities exhibit periodic behavioral patterns. Individual behavior follows diurnal, weekly, monthly, and annual patterns; agricultural activities are closely associated with seasonal patterns. Such periodic patterns reveal not only the repetitive behavior of the system but may also be indicative of external or environmental influences on the system. This also allows the analyst to find anomalous events or patterns due to deviation from the expected periodic pattern and predict the system's behavior.
However, many issues need to be considered for discovering periodic patterns [100]. Ma et al. discuss these issues and propose an algorithm for discovering partial patterns (p-patterns) in a temporal point events database. They divide the task into 2-steps 1) finding the possible periods p for every event type using Chi-square tests, and 2) find all event patterns with period p and minimum support w.

Mutually Dependent Pattern

Mutually dependent pattern or *m-Patterns* represent event patterns that are strongly correlated. These differ from d-patterns as the dependence between events is symmetric in case of m-patterns but can be uni-directional for d-patterns. Mining m-patterns using a mini-

mum support threshold has the same drawbacks as d-patterns and thus need to be mined separately.

We can define dependency between events E_1 and E_2 in a sequence of events S as:

$$P_S(E_1|E_2) = \frac{support(E_1 \bigcup E_2)}{support(E_2)}$$

Mutually dependent events can be defined[88, 99] as

Definition 3.1. *Given a sequence S and the minimum dependence threshold minp and E be the events from S. If any two events $E_1 \subseteq E$ and $E_2 \subseteq E$ are* ***significantly mutually dependent*** *with respect to S iff $P_S(E_1|E_2) \geq minp$ and $P_S(E_2|E_1) \geq minp$.*

If any two events $E_1 \subseteq E$ and $E_2 \subseteq E$ are significantly mutually dependent with respect to S then E is referred to as an **m-pattern**.

T-Patterns

In real-world systems, events can trigger subsequent events in near future and causal attribution can only be determined by mining pattern with an associated temporal constraint. *T-patterns* attempt to describe behavior using temporally constrained binary patterns. These patterns describe a sequence of events that occur within a specified time interval, for example, $E_1 \xrightarrow{T_1} E_2 \xrightarrow{T_2} E_3$. We can search for these patterns recursively and define larger and more complex patterns as a combination of repeated pairwise patterns. Magnusson et al. [101] describe an algorithm for mining these patterns and time intervals, and use these as the basis for their event exploration software THEME.

T-patterns can represent sequential relationships between events constrained by a temporal constraint however they are lacking when it comes to describing concurrent event relationships such as "Driving while it's Raining".

3.2 Event Sequence Modeling and Reasoning

Various approaches have been proposed for reasoning with event sequences and modeling different situations using events. Most of the approaches adopt a symbolic representation of events and use event patterns and temporal interval algebra to describe event relationships. However, multiple approaches adopt a numeric approach to modeling event relationships, especially for determining causal relationships between temporal event sequences. We will discuss some of the approaches in this section and evaluate their strengths and weaknesses.

3.2.1 Event Pattern Mining

Frequent Pattern Mining

Pattern mining consists of discovering significant and unexpected patterns in a database. The interest in these techniques originates from their ability to discover patterns hidden in large databases that are easily interpretable by humans, thus making them useful for understanding the data and decision-making. Agrawal and Srikanth [8] laid the foundations of the field in their seminal paper describing *Apriori algorithm* designed to discover frequent itemsets, i.e., items frequently appearing together, in a database of customer transactions. For example, we can use the Apriori algorithms to find that the itemset $\{eggs, bread, milk\}$ frequently appears in the database of transactions for a grocery store.

Although pattern mining has numerous applications in different domains, several pattern mining techniques such as frequent itemset mining[52, 146, 210] and association rule mining [7] do not account for the sequential nature of data. Thus, we need to explore algorithms that can discover frequent patterns in the data while also capturing the sequential order of the items or events.

Sequential Pattern Mining

Sequential Pattern Mining is the task of finding all the frequent sub-sequences in a sequence database. A sequence s is said to be a *frequent sequence* or *significant pattern* if and only if $support(s) \geq minsup$ for a threshold support $minsup$ set by the user[38]. Sequential Pattern Mining is essentially an enumeration problem where we want to generate all sequences with support greater than $minsup$. The naive approach to solve this problem would be to find the support of all possible sub-sequences in the database and output only those meeting the support requirement. However, this approach is inefficient as a sequence containing k items can have $2^k - 1$ distinct sub-sequences.

Numerous algorithms have been designed to efficiently explore the search space of sub-sequences, they utilize two basic operations *s-extensions* and *i-extensions*. These operations are used to $k+1$-length sequences from a k-length sequence.

- *s-extensions*: A sequence s_b is an s-extension of $s_a =< I_1, I_2, ..., I_h >$ with an item x, if $s_b =< I_1, I_2, ..., I_h, \{x\} >$.
- *i-extensions*: A sequence s_b is an i-extension of $s_a =< I_1, I_2, ..., I_h >$ with an item x, if $s_b =< I_1, I_2, ..., I_h \bigcup \{x\} >$.

The search algorithms explore the patterns in a *breadth first* or a *depth first* manner while pruning the sub-sequences with insufficient support due to the *anti-monotonicity* or *downward closure* property of sequential patterns. This property states that if s_a is a sub-sequence of s_b then $support(s_a) \geq support(s_b)$. Thus, if we see a sub-sequence that does not satisfy the support requirement, we need not explore any sequences generated from it.

AprioriAll and *GSP* are the first sequential pattern mining algorithms proposed by Agrawal and Srikanth [8, 177]. These algorithms adopt a breadth-first search approach for generating new candidate patterns and a horizontal database representation for the sequence database (fig. 3.2 a)).

a)

SID	Sequence
1	$\langle\{a,b\},\{c\},\{f,g\},\{g\},\{e\}\rangle$
2	$\langle\{a,d\},\{c\},\{b\},\{a,b,e,f\}\rangle$
3	$\langle\{a\},\{b\},\{f,g\},\{e\}$
4	$\langle\{b\},\{f,g\}\rangle$

b)

a	
SID	Itemsets
1	1
2	1,4
3	1
4	

b	
SID	Itemsets
1	1
2	3,4
3	2
4	1

c	
SID	Itemsets
1	2
2	2
3	
4	

d	
SID	Itemsets
1	
2	1
3	
4	

e	
SID	Itemsets
1	5
2	4
3	4
4	

f	
SID	Itemsets
1	3
2	4
3	3
4	2

g	
SID	Itemsets
1	3,4
2	
3	3
4	2

Figure 3.2: a) Horizontal Sequence database, b) Vertical Sequence database [38]

The GSP algorithm calculates the support for a sequence in a two step process:

1. For every candidate sequence s_a of length k, GSP checks if all its sub-sequences of length $k-1$ are also frequent. Due to the downward-closure property of patterns, if any of the sub-sequences have support less than *minsup*, then s_a will have support less than *minsup*. This step prunes a large number of infrequent patterns.

2. GSP will scan the database to calculate the support for s_a. If s_a is frequent then it is added to the output.

Despite its success in finding frequent sequential patterns, GSP has several important limitations:

- Repeatedly scans the database for calculating the support.
- May generate candidate patterns that do not exist in the database.
- Candidate patterns need to be stored in-memory for breadth-first exploration.

Numerous improvements to these algorithms have been proposed that target one or more of the above mentioned limitations. SPADE [211] utilizes a depth-first search approach to avoid some of the shortcomings of GSP. It utilizes a vertical database instead of a horizontal one, which allows us to directly find a pattern's support as the number of distinct identifiers in the IDList (fig. 3.2 b)). Additionally, we can find the IDList of a pattern s_a, obtained by extending (s-extension or i-extension) patterns s_b with item x, by joining the IDList of pattern s_b with that of item x.

Different algorithms such as SPADE [211], Spam[15], CM-Spam, and CM-Spade [39]utilize the above properties to generate frequent sequential patterns by scanning the database only once. These also allow us to generate the patterns without keeping all the patterns in memory and outperform the breadth-first search approach. Spam [15] utilizes a bit vector representation of the IDLists, that allows for efficient storage and matching while joining the IDLists as they tend to be very big in dense databases. It has been shown that the algorithms using the bit vector representation are more than an order of magnitude faster than the vertical pattern mining algorithms[15][13].

CM-Span and CM-Spade algorithms introduced the concept of *co-occurrence pruning* to reduce the number of infrequent candidate patterns generated[39]. They scan the database to create a *Co-Occurrence Map* (CMAP) that stores all frequent 2-sequences. For every candidate pattern s_a, if the last two items of s_a are not in the CMAP, we can safely ignore s_a without building its IDList.

Another well-known class of sequential pattern mining algorithms is the *pattern-growth algorithms*. These algorithms adopt a depth-first approach but do not generate candidate patterns that do not appear in the database. This is done by recursively scanning the database to find larger patterns; however, this can lead to multiple costly scans. These methods have adopted the concept of *projected database* to avoid multiple full database scans. The projected database of a pattern is the set of sequences where the pattern appears with all the items and itemsets appearing before the first occurrence of the pattern being

removed [147]. PrefixSpan[147] is the most popular pattern-growth algorithm for sequential pattern mining. The first step in the algorithm is to determine the set of single item patterns that satisfy the support threshold, this set of patterns is then used to seed the depth-first exploration of newer patterns and added to the output set. The algorithm then scans the projected database of the most frequent pattern p in the output set and finds a larger pattern of length $|p| + 1$ that starts with p and computes its projected database and support. The algorithm thus recursively searches for larger patterns and adds frequent patterns to the output set.

Many variations or extensions of sequential pattern mining have also been proposed for related applications where sequential pattern mining may not be appropriate. One fundamental limitation of the above algorithms is that they are likely to return a large number of patterns meeting the support threshold. Many approaches have been proposed to discover a concise representation of sequential patterns. *Closed sequential patterns* are the set of patterns not included in any other patterns with the same support. These can be defined as:

$$CS = \{s_a | s_a \in FS \wedge \nexists s_b \in FS | s_a \sqsubset s_b \wedge sup(s_a) = sup(s_b)\}$$

where FS is the set of all sequential patterns. Discovering closed sequential patterns reduces the size of the output set, and all interesting itemset and sequential relationships are still preserved. Bide[194] and CloSpan[209] are some of the first proposed solutions for closed sequential pattern mining and adopt a pattern-growth approach extending the PrefixSpan algorithm.

Maximal Sequential Patterns are the set of sequential patterns not contained in any other sequential patterns. These can be defined as:

$$MS = \{s_a | s_a \in FS \wedge \nexists s_b | s_a \sqsubset s_b\}$$

The number of maximal patterns is never more than the number of closed patterns ($MS \subseteq CS \subseteq FS$) and in practice can be one or two orders of magnitude less than all sequential patterns. However, maximal sequential patterns are not lossless as they do not preserve the support information for all the sub-maximal patterns. Several algorithms have been proposed for maximal sequential pattern mining that include breadth-first(AprioriAdjust), depth-first(VMSP [40]), pattern-growth(MaxSP[41]) and approximation approaches (DIMASP[46]).

Sequential pattern mining focuses solely on positive correlations between items and itemsets; however, negative correlations are more interesting in some applications. This problem is addressed by *negative sequential patterns*. A *negative sequential pattern* is a pattern containing the negation or absence of at least one item. This task is more challenging as including the absence of items drastically increases the search space. Negative-GSP [217] and PNSP[57]extend the GSP algorithm for mining negative patterns.

Periodic Pattern Mining

Periodic pattern mining refers to discovering patterns that appear frequently and regular periodic intervals. The periodicity of the pattern is measured by the time elapsed between two consecutive occurrences of the patterns, also called the period length. Periodic patterns can be further divided into Fully or Partially periodic patterns and Perfect or imperfect patterns[2]. In fully periodic patterns, all the symbols in the patterns sequence are periodic with the same period, while in partially periodic patterns, only a subset of symbols is periodic. Ma et al. [100] proposed an algorithm for finding all partially periodic patterns by first discovering frequent periods observed in sequences and then finding frequent associations or sequences for each period. Ozden et al. [136] define perfect periodic patterns as a pattern X that appears after every p in the sequence S. They proposed an algorithm to find cyclic association rules that appear in every cycle for the entire sequence.

Temporal Pattern Mining

Temporal Pattern mining has been employed in varied fields such as medical informatics [98], spatiotemporal data[130], and multimedia streams. Most of the temporal pattern mining research utilizes point or interval or both representations of temporal events. Allen's temporal logic is commonly used to describe complex relationships between event intervals [11]. Different event intervals are related by one of the temporal relations, and different algorithms have been designed to mine patterns among these events. Villafane et al. [187] proposed a graph mining method to discover temporal patterns. However, their approach was constrained to only two Allen relationships: 'contains' and 'during'. Kam and Fu [72] propose a hierarchical representation and algorithm for discovering temporal patterns. However, hierarchical representation suffers from two major drawbacks: 1) the same event relationship can be mapped to different patterns, 2) the same pattern can represent different event relationships. Hoppner et al. [55] propose an unambiguous representation of, called representation matrix, that exhaustively lists all pairwise relationships between events in a pattern. *TPrefixSpan* [202] employs a projected database similar to PrefixSpan algorithm to discover frequent temporal patterns. However, extending the temporal patterns is more complicated than extending sequential patterns and more computationally expensive.

Moerchen et al. [116] propose a temporal pattern mining framework utilizing semi-interval representation. They represent patterns as a partial order of semi intervals and successfully demonstrate the flexibility of semi-interval representation for describing patterns.

Allen and Freksa relations by themselves do not capture any information about the temporal gap and duration of the events, which could easily distinguish two or more patterns with the same symbolic representation. Namaki et al. [129] propose GTAR (Graph Temporal Association Rules) to extend association rules and include the temporal intervals between events in the discovered rules.

Hybrid temporal pattern mining refers to discovering patterns in both point and interval-

based sequences. MILPRIT*[25, 26] is a constraint-based hybrid temporal pattern mining method that allows users to specify pattern constraints as a regular expression. It represents patterns as a triple (K, D, T), where K represents the actors and entities references in the pattern, D represents the actions or events with associated timestamps, and T represents the temporal relationships between events as defined by Allen's relational algebra. Wu et al. [203] argue that discovering hybrid temporal patterns cannot be reduced to interval or point event patterns, and discovering temporal patterns cannot be reduced to discovering sequential patterns in either point or interval representation. They propose an approach called HTPM for mining hybrid temporal patterns from event sequences, consisting of both point-based and interval-based events. It iteratively generates all k-patterns from all pairs of $(k-1)$-patterns. It stores all frequent sequences for quick processing; however, this leads to considerable memory consumption.

Jalali and Jain [65] describe a temporal event mining framework and event pattern language that allows researchers to specify event patterns with temporal operators describing the sequential and concurrent relationships between interval and semi-interval events. Different types of patterns between the same pair of events can be captured by the differences in the time delay between them, which helps distinguish these patterns from each other instead of using Allen's or Freksa's relations. Including temporal gaps and intervals in the relationships addresses the concerns raised by Wu et al. [203]. We will utilize the event pattern language proposed by [65] in our work and design an event mining framework around it.

3.2.2 Situation Calculus

Situation calculus is a logic formalism utilized to reason about actions and changes. McCarthy [106] first proposed situation calculus, and has undergone many variations since then. In this section we will focus on the situation calculus as proposed by [152]. Fundamental elements of the situational calculus are:

- *Actions* that can be performed in the world. For example, *move(x,y)* describes an action for a robot to move to position x, y. Actions also have some special domain independent predicates associated with them such as $Poss(a, \sigma)$, that tell us if an action(a) is possible in a given state(σ).

- *Situations* are described by a sequence of actions, and can be changed by any future actions in the world. For example, in the robot world example, $do(move(x, y), \sigma_0)$ represents the situation after the action $move(x, y)$ has been performed. If in this situation $PickUp(Ball)$ is performed, then the situation becomes $do(PickUp(ball), do(move(x, y), \sigma_0))$.

- *Fluents* describe the state of the world. Fluents are represented as logical predicates whose truth values are state dependent. For example, $HasBall(\sigma_0)$ is true if the robot is holding a ball in situation σ_0. The example shown here is a *functional fluent* because its output is a truth value dependent on the state as $HasBall(\sigma_0)$ evaluates to $False$ but $HasBall(do(PickUp(ball), do(move(x, y), \sigma_0)))$ is $True$.

Researchers need to identify the required set of actions and fluents to model the agent behavior and state of the world accurately. In the classic example of blocks world [134], the possible actions for an agent are:

- $pickup(x)$: Pick up block x, provided the agent's hand is empty, the block is on the table, and no other block is on top of it.

- $putdown(x)$: Put down block x, provided the agent holds the block.

- $stack(x, y)$: Stack block x on top of y, provided the agent is holding x, y is on the table and no other block is on top of y.

- $unstack(x, y)$: Pick up x from on top of y, provided the agent's hand is empty, and no other block is on top of x.

Some the relational fluents required for this situation are:

- $HandsEmpty$: True if the agent's hands are empty in a situation.
- $Holding(x)$: True if the agent is holding the block x.
- $clear(x)$: True if the block x has no other block on top.
- $onTop(x, y)$: True if the block x is on top of y.
- $onTable(x)$: True if the block x is directly on the table.

3.2.3 Point process event sequence models

Different parametric methods have also been utilized to represent event sequences and model a system's behavior. Poisson processes are commonly employed to model point events in network applications to model random point processes. However, Poisson processes assume that the number of occurrences (or events) in disjoint intervals are independent of each other, also known as the *memorylessness* property. This is not necessarily true for situations where current and future events also depend on past events (e.g., sleep for individuals, traffic congestion, movement patterns, and social media events such as posts and retweets).
Hawkes Processes[53] are suitable for such cases where we are trying to model a self-exciting process[153]. The conditional intensity function $\lambda(t|\mathcal{H}_t)$ for the process can be defined as:

$$\lambda(t|\mathcal{H}_t) = \lambda_0(t) + \sum_{i:t>T_i} \phi(t - T_i)$$

where $T_i < t$ are the timestamps of events up to the current time t, $\lambda_0(t) : \mathcal{R} \to \mathcal{R}_+$ is the base intensity function describing the events triggered by external sources and $\phi(t-T_i) : \mathcal{R} \to \mathcal{R}_+$ is the memory kernel that modulates the effect that past events have on intensity at the current time. Hawkes process and its variants have been used to address different event

sequence problems such as clustering [208] and learning granger causality [207]. Zhang et al.[214] utilize a neural point process to identify granger causality between asynchronous, interdependent, multi-type event sequences. They argue that commonly used point-process models have very strong parametric assumptions that may not generalize well.

3.3 Event Analysis Tasks

Event sequence analysis methods can be used to perform different operations on a sequence database. Systems designed for helping analysts model situations using event-based data utilize different visualizations to represent event patterns. Different visual event and pattern representations are designed to address specific event analysis tasks. We will list some of the analysis tasks and briefly discuss what computational techniques and visualizations are commonly used to address the goals of the analysis.

3.3.1 Summarization

Event sequence summarization helps analysts uncover major progression patterns and featured groupings of sequence entities. The fundamental goal is to provide a quick overview of the entire sequence database. Sequential pattern mining, progression analysis, and sequence clustering have been used in different works to generate summaries of event sequence databases.

Explicit summarization refers to using visualizations to display all event sequences aggregated into one interface. Event timelines reveal temporal information among sequences such as distribution of events in varying time granularities [151][193], and Sankey-based visualizations can reveal the progression of events over time [199]. However, the visual interface for explicit summarization can become quite large and messy for large event sequences and

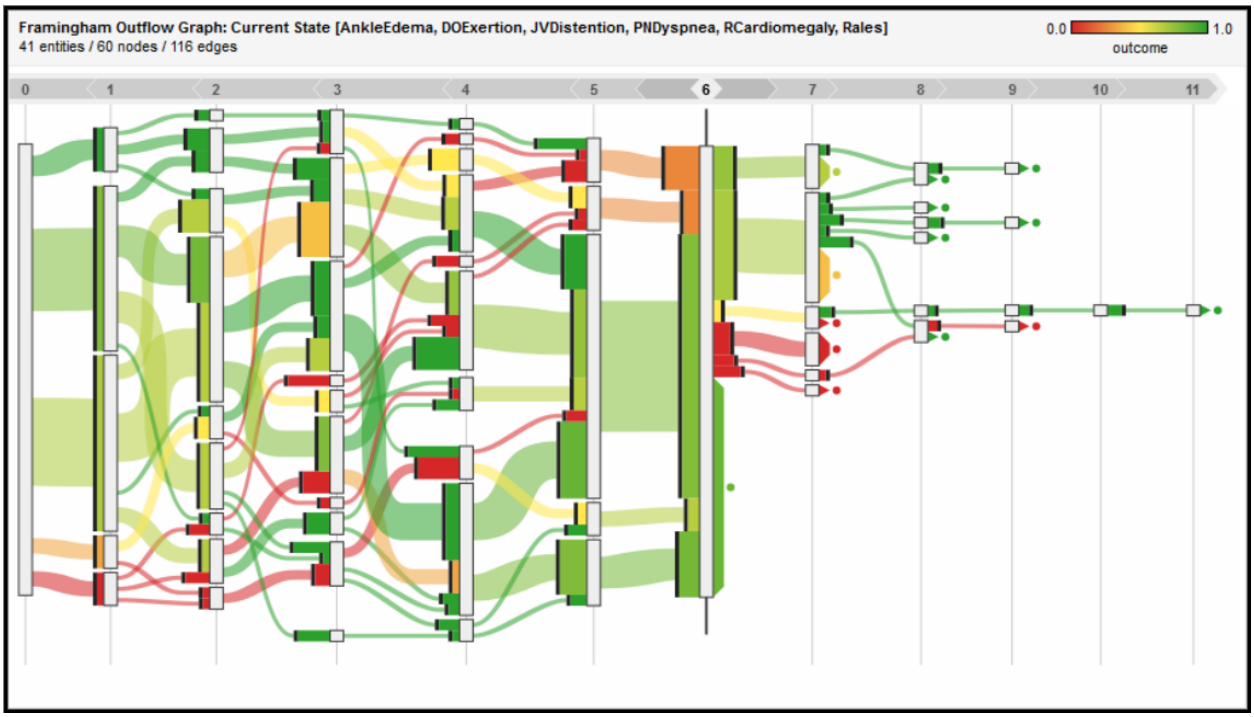

Figure 3.3: Outflow [199] utilizes Sankey diagram based visualization for explicitly describing event trajectories over time for the individuals in the Framingham study.

databases. *Inexplicit summarization* leverages data mining techniques for discovering informative patterns among event sequences. The analysis can be driven by user queries to extract relevant event sequences. These techniques provide a rich and interpretable query language that analysts can use to define sequence extraction logic[35][79]. Many applications utilize sequential mining and frequent mining algorithms to summarize multivariate symbolic temporal sequences[1][94][93]. *Clustering* is also very frequently used to find sequence-wide similarities and sequence groupings. Temporal event sequence clustering typically uses sequence characteristics such as event types and sequence attributes. Clustering of spatio-temporal events have been explored in many works[58][154][190].

3.3.2 Prediction

Prediction requires analyzing the observed event sequences to foresee the subsequent events in the sequence and are typically used for making predictions and recommendations to help users achieve specific goals. Event predictions can play a vital role in decision-making in many domains. Deep learning applications have been utilized to predict the risk that a patient may be diagnosed with a disease in the future [48][67]. Recurrent Neural Networks are frequently used for prediction tasks; however, interpretability remains a primary challenge for the deep learning approaches.

3.3.3 Anomaly detection

Anomaly detection for event sequences attempts to identify rare cases and occurrences that deviate from most sequence progressions. Anomaly detection can be divided into two major categories 1) Anomalous event detection, 2) Anomalous patterns detection.
Anomalous events can be identified in the context of sequence based on the expected event progressions[21][36][121][133] and multimodal events can also utilize event parameters to identify anomalous event occurrences. *Anomalous frequent patterns* are frequent patterns that describe unexpected behavior or pattern of events [77][140][175]. These methods have been employed to identify anomalous learning patterns in MOOCs and anomalous traffic patterns.

3.3.4 Causality Analysis

Timestamped event sequences can carry a great deal of information about the underlying causal mechanisms. Many methods have been developed to find the mutual causation of events, including graphical modeling, Hawkes-process-based, and deep learning approaches.

Graphical causal methods such as Peter & Clark (PC) and Functional Causal Models (FCM) are well-known causal discovery methods[143][176]. TiMINO [149] and VAR-LiNGAM [59] extend the FCM equations with time lags of causal relationships and PCMCI[161] and tsFCI[34] utilize conditional independence testing in the time-lagged correlation analysis.
Various approaches have been developed around Hawkes processes (discussed in the previous section). The self and mutual-excitation properties of the Hawkes process have prompted studies that attempt to recover the causal structure between the events. Eichler et al. [32], and Xu et al. [207] apply Granger causality to Hawkes processes using a least-squares estimation of the impact function. Jin et al.[66] have also developed a visual causal discovery framework around a Granger causality analysis algorithm based on Hawkes processes.
Recent causal discovery methods attempt to leverage deep neural networks to capture complex event dependencies. Zhang et al. [214] utilize neural point processes based on RNN[112] in place of Hawkes process for causal discovery. Nauta et al. [131] employ an attention-based convolutional neural network to model causal relationships and delays in temporal data.

3.4 Event Analysis applications

Applications in different domains utilize event sequence data for understanding the user's behavior and needs. We will review some examples of such applications in domains like health informatics, Social media, and E-commerce.

3.4.1 Health Applications

Electronic health records (EHR) and electronic medical records (EMR) are frequently used for analyzing patient visits and disease progressions. These can be represented as event sequences, and the details of the records can be stored as event parameters. Each event

can represent a medical event such as a lab test, a diagnosis or treatment, and a sequence would thus capture an individual's medical history. The combination of ample medical events sequence data and domain knowledge have enabled physicians and medical researchers to derive new knowledge, compare the efficacy of different treatments, and identify personalized treatment protocols. We will discuss the use of event sequence analysis for three major medical and health informatics tasks.

1. *Cohort Analysis* is used to discover the relationships between specific disease risks and the patient attributes that define the cohorts. The cohorts can be defined based on specific medical events, patient attributes (eg. age, gender), and individual event sequences' patterns. Cohort summarization techniques are used by systems such as CAVA[215] and Chronodes[62] to visually summarize the informative patterns within a cohort and discover the common exposure factors for disease.

   Cohort comparison is used to discover the differences between two cohorts of patients to find exposure factors for disease. COQUITO[79], PARAMO[70] and CoCo[103] utilize a combination of event sequence clustering and visual representation of patient cohorts to determine if the constructed cohorts carry exposure events for the disease.

2. *Prognosis Analysis* predicts the probability of a patient being diagnosed with certain diseases in the future using their medical history. Various applications utilize deep learning methods for prognosis analysis, [78][83] utilize RNNs to predict the future state of patients. RetainVis [83] enables analysts to modify the event sequences to observe how the future risks are affected. CarePre [67] also predicts the risk of patients being diagnosed with specific diseases and also identifies the most effective treatments based on their medical history.

3. *Outcome Analysis* studies the results of different medical progressions or interventions such as disease progression and treatment progression. Systems such as Outflow [199] and Frequence[148] utilize explicit sequence summarization using a Sankey diagram to

discover the outcomes of different procedures.

3.4.2 Social Media

Social Media platforms such as Twitter and Instagram also record user activities as event sequences. Each sequence consists of temporally ordered events representing users' interactions with the platform, such as posting or commenting. These sequences can be utilized to identify different types of user behaviors on such online forums. Two major types of behaviors exhibited by users are *Collective behaviors* and *Egocentric behaviors*.

Collective behaviors are activities conducted by a temporary and unstructured group of people. These include processes like information spread and human mobility. [185] describe the graph-based structure utilized by Google+ to understand the reposting behavior and capture the paths taken by different popular posts. Zhao et al. [216] propose a method to understand the rumor-spreading process on Twitter using a timeline visualization. Spatio-temporal event sequences from social media platforms like FourSquare have been used to discover user mobility patterns in different locations and have been used to optimize advertising strategies[86][148].

Egocentric behaviors are activities conducted or influenced by an individual. An egocentric perspective allows us to find detailed behavioral patterns and changes in user behaviors in response to significant public events and life events[163]. Saha et al. augment social media events with multimodal lifestyle data in their works to understand user behavior in situated communities such as college campuses and work places[165, 164, 162].

3.4.3 E-commerce

Similar to social media platforms, e-commerce websites also record a user's interactions with the platform as event sequences. Sequential events data have been leveraged to understand

user requirements and behavior, leading to improved advertising efficiency and personalized user experiences. Zgraggen et al. [213] propose a method for finding frequent patterns of clickstream data using a regular expression-based query language and discover frequent visiting traces. Different works explore sequential pattern mining coupled with innovative interfaces such as funnel-based visualisation[93] and tree-based visualization[92] to facilitate the discovery of prominent browsing patterns.

Chapter 4

Event Mining: Concepts and System

Events are the computational representation of the real-world activities of a system. They provide a dynamic temporal structure for the timestamped data instead of the static structure provided by daily, weekly or monthly calendars or time windows. Additionally, the intervals derived from events are semantically meaningful and make it easier for analysts to identify data relevant to their goals. Events can be recognized from different data sources and thus provide an inherently interpretable and more robust abstraction over possibly noisy data streams. Various temporal event processing and reasoning frameworks have been proposed that utilize the event sequences to get a better understanding of the system state or situation (fig. 4.2). Multimedia events have been the subject of study for many works and have been utilized to aggregate heterogeneous data from different sources and associate them with the corresponding events. Westermann and Jain [197] propose a multimedia event model with six different data aspects of an event. Each event answers a specific category of questions related to the event and thus leverages multi-modal data for cross-modal retrieval and analysis. This structure also enables the researchers to seamlessly integrate the latest developments in sensor technology and newer modalities in their computational pipeline without disrupting the existing models and patterns.

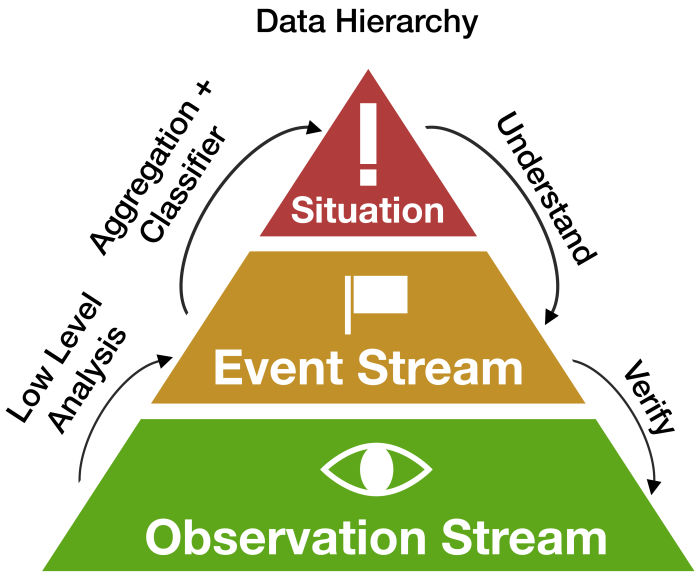

Figure 4.1: Events provide a natural abstraction over data streams and segment time in intervals with semantic meaning. Event abstraction also allows us to fuse data from multiple sources and enhance our understanding of real-world actions.

Using events for studying the behavior of a system or individuals is likely to result in models and patterns that can easily be interpreted in the context of the domain. Thus, any insights or recommendations generated from such systems would be in the form of easily actionable events.

In this chapter, we describe the temporal events and concepts as defined by [65] in their work on event mining for explanatory modeling. We extend the event pattern language proposed in [65] with group and aggregate operations that can be used for descriptive modeling of data and will be utilized in the later chapters of this dissertation. We also describe the high-level architecture of the novel n-of-1 observational modeling platform proposed in this work.

4.1 Concepts and definitions

We will introduce some concepts and definitions about events and patterns utilized in the rest of the paper. These lay the foundation for the event pattern language we use for pattern

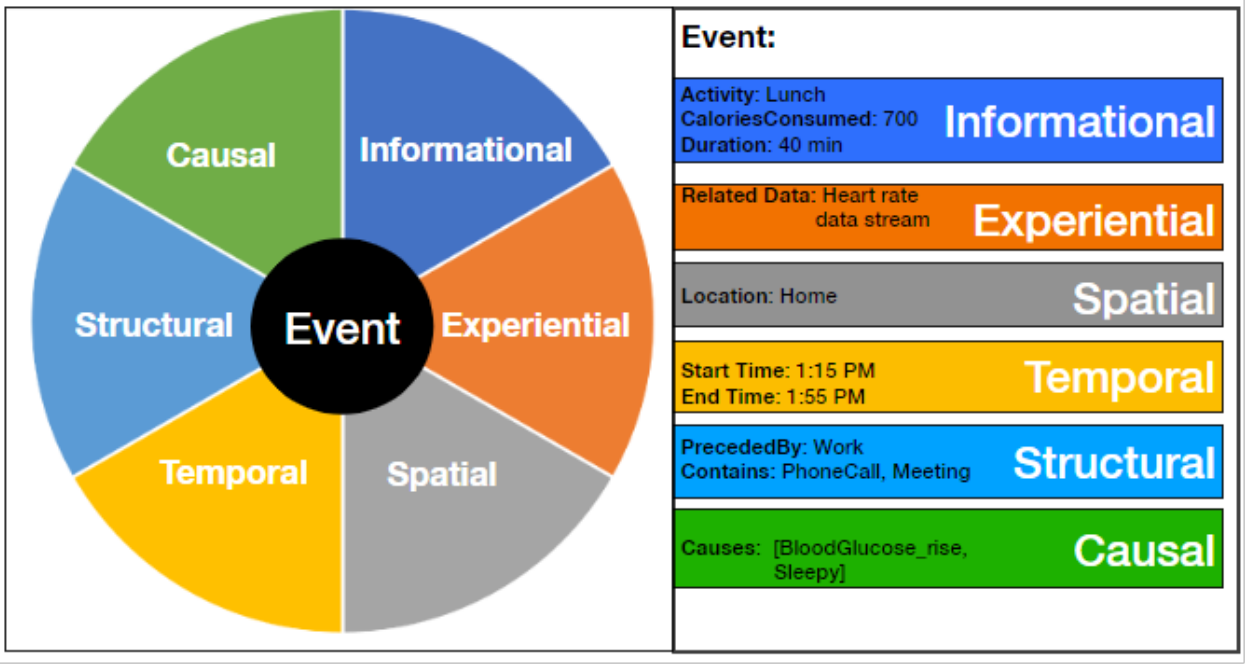

Figure 4.2: Events provide an abstraction for real world activities and event parameters capture information from disparate multi modal data streams. These parameters can be structured in six different aspects. Each of these aspects can be used to answer one of the W5H questions about the event occurrence.

specification, pattern discovery and hypothesis specification.

4.1.1 Events and Event streams

Events are computational objects that represent real-world activity in a system. These events can either be instantaneous (point event) or over some time (interval event). In some cases, we might have semi-interval events where we only know when an interval event starts or ends. We use the semi-interval event representation [10] to represent all three types of events in our work.

Each event also has a set of associated properties that describe the event.

Definition 4.1 (Event)**.** *An event can be defined as:*

$$e_i = (E, P, [e_i^+, e_i^-])$$

where E represents the class of events (the activity represented by the event, e_i), e_i^+ represents the timestamp of the beginning of the event, e_i^- is the timestamp of the end of event, and P represents the set of properties associated with events of class E. For point events, $e_i^+ = e_i^-$, and for semi-intervals we only know one the event end points.

Event streams are ordered collections of events that share some semantic significance. For example, all running, cycling and swimming events can be grouped together to create exercise event stream. We can define temporal order as follows :

$$e_1 \prec e_2 \Rightarrow (e_1^+ < e_2^+) \vee ((e_1^+ = e_2^+) \wedge (e_1^- < e_2^-))$$

Definition 4.2 (Event Streams)**.** *An event stream is a set of temporally ordered and non-overlapping events.*

$$ES_i = \{e_1, e_2, e_3, ..\}$$

where $(e_i \prec e_j) \wedge (e_i^- < e_j^+) \wedge (e_i.T = e_j.T) \forall i < j$.

4.1.2 Event operators

Event operators are used for creating new events or event streams from existing ones. There are two main categories of event operations 1) *combination*, and 2) *filter*.

Event combination operators can be used to combine two or more time intervals (representing events) to create a new event. These operators utilize a combination of interval union, intersection, and complement operations to define the event combination logic. Figure 4.3 depict how two interval events can be combined using AND and OR operations. Additionally, the NOT operator can be used to find the inverse of event intervals and utilized as

any other event.

Definition 4.3 (Event Combination)**.** *Event combination operators combine one or more event streams to derive a new event stream. The event combinations can be specified as a logical expression using event streams as operands, and AND(∧), OR(∨) and NOT(¬) operators.*

We will define the logical operators below.

1. **AND**(∧): Finds the intersection of intervals in the operand event streams. This operator returns only the intersection of the events that overlap with each other.
   $ES_{out} = ES_1 \wedge ES_2$
   $ES_{out} = \{e_{1,i} \cap e_{2,j} | \forall e_{1,i} \in ES_1, e_{2,j} \in ES_2, e_{1,i} \cap e_{2,j} \neq \emptyset\}$

2. **OR**(∨): Combines all the events from two event streams in the result. In case of overlapping intervals, union of the intervals is returned.
   $ES_{out} = ES_1 \vee ES_2$
   $ES_{out} = \{e_{1,i} \cup e_{2,j} | \forall e_{1,i} \in ES_1, e_{2,j} \in ES_2, e_{1,i} \cap e_{2,j} \neq \emptyset\} \bigcup$
   $\{e_{1,i} | \forall e_{1,i} \in ES_1, e_{2,j} \in ES_2, e_{1,i} \cap e_{2,j} = \emptyset\} \bigcup$
   $\{e_{2,j} | \forall e_{1,i} \in ES_1, e_{2,j} \in ES_2, e_{1,i} \cap e_{2,j} = \emptyset\}$

3. **NOT**(¬): Returns the complement of all the intervals in the operand event stream. This is a unary operator and is typically used in combination with one of the other two operators.
   $ES_1 = \{(e_i^+, e_i^-) | \forall e_i \in ES_1\}$
   $ES_2 = \neg ES_1 = \{(e_i^-, e_{i+1}^+) | \forall e_i \in ES_1\}$

Event filter operation can be used to select only events from an event stream that satisfy a set of conditions. These conditions consist of constraints on different event parameters, including event start and end time and event name.

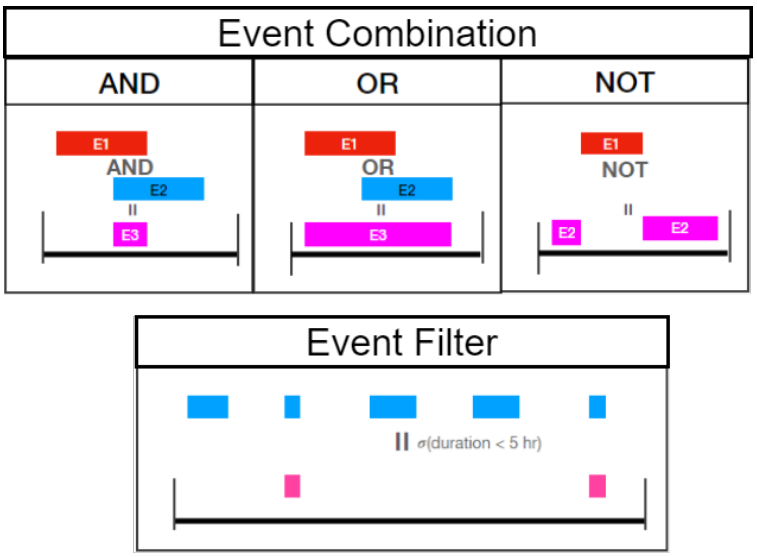

Figure 4.3: Event operators for creating new event streams from the existing ones.

Definition 4.4 (Event filter)**.** *Event filter operations apply a set of constraints,* Φ*, on event parameters and result in a new event stream with events that satisfy the constraints.*

$$ES_2 = \sigma(ES_1, \Phi)$$

$$ES_2 = \{e_i | \forall e_i \in ES_1, \Phi(e_i) = True\}$$

The constraints for event filter operations can be as simple as equating values or more fuzzy matching using similarity scores between parameters. The constraint filter can help curate the events data set for any analysis and redefining events based on similar parameter values.

Different event operations can be used in a single expression to define new complex event streams as a function of existing events. For example, if we want to find intervals where a person is running, their heart rate is in the aerobic zone (140-160 bpm), and they are not in their home city (Irvine). As the problem suggests, we need running event stream (ES_{run}) and heart rate event stream (ES_{hr}) for creating the new event stream.

We can define the new event stream ($ES_{AerobicRun}$) as:

$$ES_{AerobicRun} = \sigma(ES_{run}, \Phi := location! = \text{“}Irvine\text{”}) \wedge \sigma(ES_{hr}, \Phi := event = \text{“}Aerobic\text{”})$$

Thus, the event operations help us derive new and more complex events relevant to the analysis and can be used in subsequent patterns and aggregation analyses.

4.1.3 Pattern

Patterns allow us to specify tuples of events that are related to each other via a temporal constraint. The interval representation of the events allows us to utilize Allen's interval operations[10] for describing temporal relationships between different events.

Definition 4.5 (Patterns)**.** *A pattern is a relation between two event streams that is defined by a temporal constraint C. Thus, a pattern P between event streams ES_1 and ES_2 defined by temporal constrain C can be represented as:*

$$P = \{(e_{1,i}, e_{2,j}) \forall e_{1,i} \in ES_1 \wedge e_{2,j} \in ES_2 \wedge (e_{1,i} C e_{2,j})\}$$

where $(e_{1,i} C e_{2,j})$ represents that the pair of events, $e_{1,i}$ and $e_{2,j}$, satisfy the temporal constraint C.

We utilize three event pattern operations (temporal constraints) described in [65] that capture possible interval relationships.

1. **Concurrent** operation: Identify pairs of events that overlap.

$$P_{Conc.} := ES_1 \perp ES_2$$

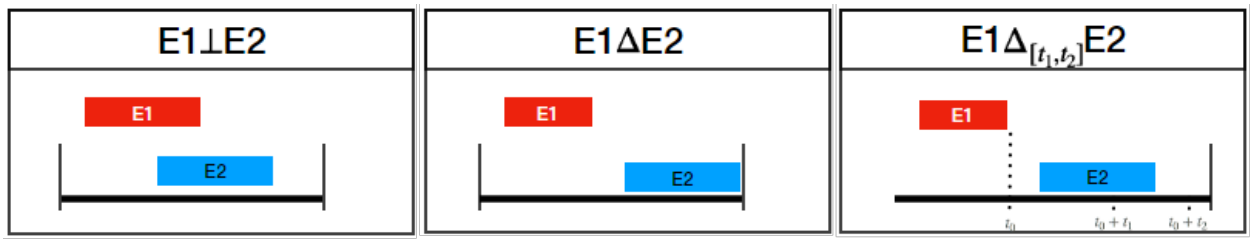

Figure 4.4: Event pattern operators to capture different temporal relationships between the events.

$$P_{Conc.} = [(e_{1,i}, e_{2,j})\forall i, j, e_{1,i} \cap e_{2,j} \neq \phi]$$

2. **Sequential** operation: Identify pairs of events that occur in the specified sequence.

$$P_{Seq.} := ES_1 \Delta ES_2$$

$$P_{Seq.} = [(e_{1,i}, e_{2,j})\forall i, j, e^+_{1,i} < e^+_{2,j}]$$

3. **Conditional Sequential** operation: Identify pairs of events that occur in the specified sequence and within the specified time interval.

$$P_{Cond.Seq.} := ES_1 \Delta_{[t_1,t_2]} ES_2$$

$$P_{Cond.Seq.} = [(e_{1,i}, e_{2,j})\forall i, j, e^+_{1,i} + t_1 \leq e^+_{2,j} < e^+_{1,i} + t_2]$$

4.1.4 Groups and Aggregations

Groups are helpful when the nature of the relationship between events is not one-to-one. This can also be viewed as an event having a carry-over effect, as discussed in section 2.2. This is a prevalent situation in the context of health applications and biological systems. For example, a person's aggregated sleeping behavior in the past week would affect their current mood, not just their most recent sleep. Groups allow us to capture the aggregated nature

of these relationships. Groups are created from a pattern and define a set of events of one event stream related to a specific event of the second event stream.

Definition 4.6 (Group)**.** *Consider a pattern, P, such that*

$$P = \{(e_{1,i}, e_{2,j}) \forall e_{1,i} \in ES_1 \wedge e_{2,j} \in ES_2 \wedge e_{1,i} C e_{2,j}\}$$

where C is the temporal constraint relating two events.
We can define a group, G, as follows

$$G = P.groupBy(ES_1)$$

$$G = \{(e_{1,i}, \{e_{2,j} \forall j, (e_{1,i}, e_{2,j}) \in P\}) \forall i, e_{1,i} \in ES_1\}$$

Every element in the group relates an event from ES_1 to a set of events from ES_2. The events from ES_1 are called the *indexing events*, as these are used to index the events from ES_2. We can also use ES_2 as indexing events, mapping all the associated events from ES_1 to an events in ES_2. Thus, groups can map any outcome event to all the events that are causally related to it.

Definition 4.7 (Aggregates)**.** *An aggregate over a Group G can be defined by a function F, that takes as input a set of events E, and maps it to a vector $V \in R^n$. An aggregate, A, is*

$$A = G.apply(\mathbb{F})$$

$$A = \{(e_{1,i}, \mathbb{F}(E)) \forall (e_{1,i}, E) \in G\}$$

Thus, applying an aggregate results into one vector, V, associated with each indexing event in the group.

We can apply any generic functions on the related set of events in a Group to calculate

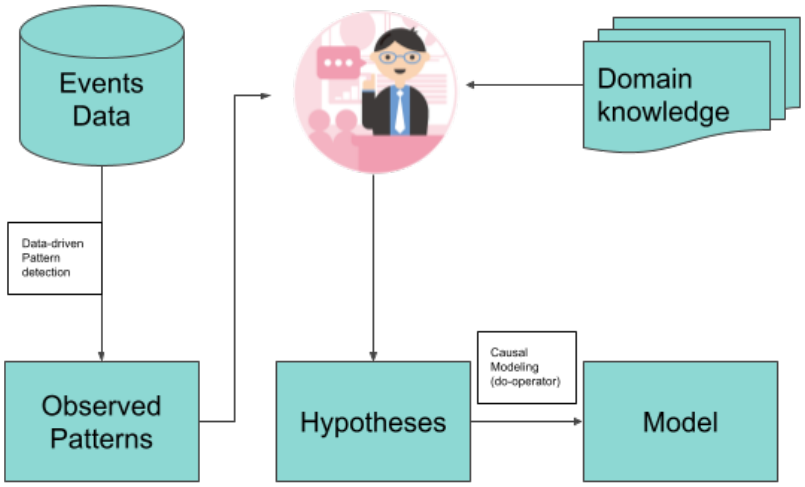

Figure 4.5: High level overview of the event mining analysis workflow. The analyst in the loop identify the significant or non-spurious patterns from the set of discovered patterns. They can combine the significant patterns and their knowledge of the domain to create a hypothesis that can then be tested using the platform.

summary statistics relevant to our experiments. These functions include (but are not limited to) finding the average/sum of an event parameter, the average time of day for associated events, etc. This allows us to find the relevant factors from the related events in the past and will be utilized to define causal hypotheses and design experiments.

4.2 Event Mining System: Functional requirements

The Event-based knowledge discovery and modeling paradigm we employ is interactive and relies on a human-in-the-loop to ensure that spurious patterns and event relations are not used in the model. We have used the operators described above to design an event mining framework that can be used for n-of-1 modeling of individuals. We will discuss different components of the system in the subsequent sections and describe the system architecture.

Event stream Schema

Field	DataType	Constraints	Description
participant_id	String	INDEX	Unique id of the participant
start_time	Timestamp	INDEX	Time at the beginning of an event
end_time	Timestamp	INDEX	Time at the end of an event
event_type	String		Event Stream name
event_name	String		Event Name
parameters	JSON		Parameter values for the event
datastreams	JSON		Datastreams used to find event parameters
id	UID	UNIQUE	Unique event id

Data stream Schema

Field	DataType	Constraint	Description
participant_id	String	INDEX	Unique id of the participant
timestamp	Timestamp	INDEX	Timestamp of the measurement
value	<DataType>		Measured Value
unit	String		Unit of the measurements
source	String		Source device/application for the measurement
id	UID	UNIQUE	Unique row id

Pattern Schema

FieldName	DataType	Constraints	Description
row_id	UUID	Unique,Primary	
event1_type	String		First Event stream Name
event1_id	Event UUID	Foreign Key	Event 1 unique id
event2_type	String		Second Event stream Name
event2_id	Event UUID	Foreign Key	Event 2 unique id

Figure 4.6: Database schema used to combine data and event streams from different sources.

4.2.1 Data Fusion

The event mining platform is capable of ingesting data from different sources using a unified event schema. The events are stored in a PostgreSQL table as it allows to keep structured event fields (name, type, start and end time) as well as unstructured parameters in a JSON field in a unified schema (fig. 4.6).

However, each data stream is stored in a separate table, though if multiple sources are measuring the same data stream, they are stored in the same table and can later be chosen or aggregated as required by the analysis.

4.2.2 Event Creation

Events form the fundamental unit of analysis and are the primary actors on the individual in the event mining analysis. Events representing user activities such as sleep, exercise, and

meals could originate directly from data sources such as sensors, smartphone applications, and IoT devices. However, these life activity events may not be sufficient to capture low-level activities or details such as variations in heart rate or intensity levels during any physical activity. A prominent example of such events is different sleep stages that can be identified using EEG signals but are lost if we only capture the sleep event. However, the subjective outcome of sleep quality and restfulness depends on the sleep-stage events. Therefore, event creation operations allow us to create new relevant events for the analysis and enhance the existing events by associating new information.

The event mining system presented in this work has three primary operations (fig. 4.7) for creating new events:

1. **Threshold based segmentation**: This operator utilizes a set of non-overlapping thresholds to be applied over a data stream. Each threshold range has an associated label that represents the event name. Continuous intervals with a duration greater than a minimum duration (e.g., 5 seconds) are stored as a new event.

2. **Motif-based segmentation**: This operator relies on motif discovery algorithms such as **S**ymbolic **A**ggregate appro**X**imation or **SAX**[91]. Motif discovery algorithms attempt to find repeatedly observed patterns and variations in time series data. The user can use the operator to find such patterns and identify patterns corresponding to real-world events.

3. **Event combination operators**: As discussed in the previous section, event combination operators can also be used to create new events. These operations can find event occurrences that are a direct combination of existing events and can be used in the models and patterns instead of the parent life events.

The newly created events also follow the same schema as other events and are stored in a table that holds temporary results.

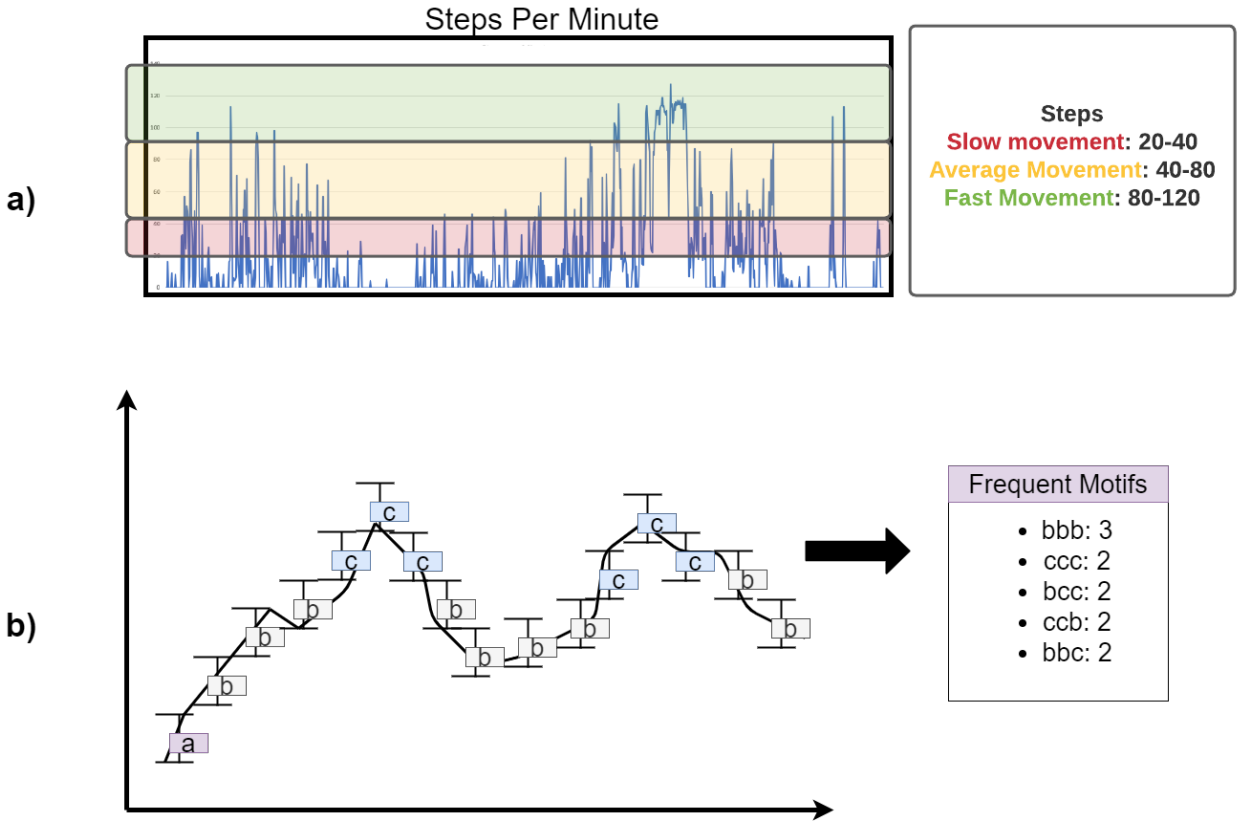

Figure 4.7: Event creation operators in the event mining platform. a) Threshold based segmentation. Matches the values of the time series to a label using non-overlapping ranges. Continuous intervals defined as having the same label are stored as new events. b) SAX based segmentation. Converts the numeric time series to symbolic series and finds frequent sub strings as motifs. These sub strings represent possible new events.

4.2.3 Pattern Creation

Events from a system may have a causal impact on other events, which changes the frequency of event occurrence and their parameters. For example, weather-related events like rainfall or storms are likely to affect exercise behavior. These effects are commonly observed as changes in event occurrences or parameter value distributions. We can use temporal event patterns, event creation, and filter operations to find such relationships. We utilize the operators discussed in the previous section to define different patterns from event streams. The patterns are stored in a PostgreSQL table, and the schema is described in fig. 4.6.
This relational representation of patterns allows us to easily implement *aggregation* and *group* operators using SQL queries. We will use these operations when defining and testing hypotheses that relate aggregated event parameters to certain outcomes.

4.3 System Architecture

This section describes the overall system architecture and implementation details for different system components. Figure 4.8 shows a high-level view of the different components of the system and their interactions with each other. We have implemented a subset of the event pattern language in a web-based platform that allows an analyst to explore the events data set interactively. The dashboard can be accessed at `https://theeventminer.com` and is discussed in detail in Appendix B. There are six main components of the system:

1. **Data fusion layer**: This module is responsible for parsing the data from different sources and storing it in the unified database. Currently, we have implemented the parsers for popular lifestyle data platforms such as Apple HealthKit and Strava so that the users can connect their accounts with the event mining platform using OAuth verification. Analysts can also upload CSV (comma-separated values) files with headers

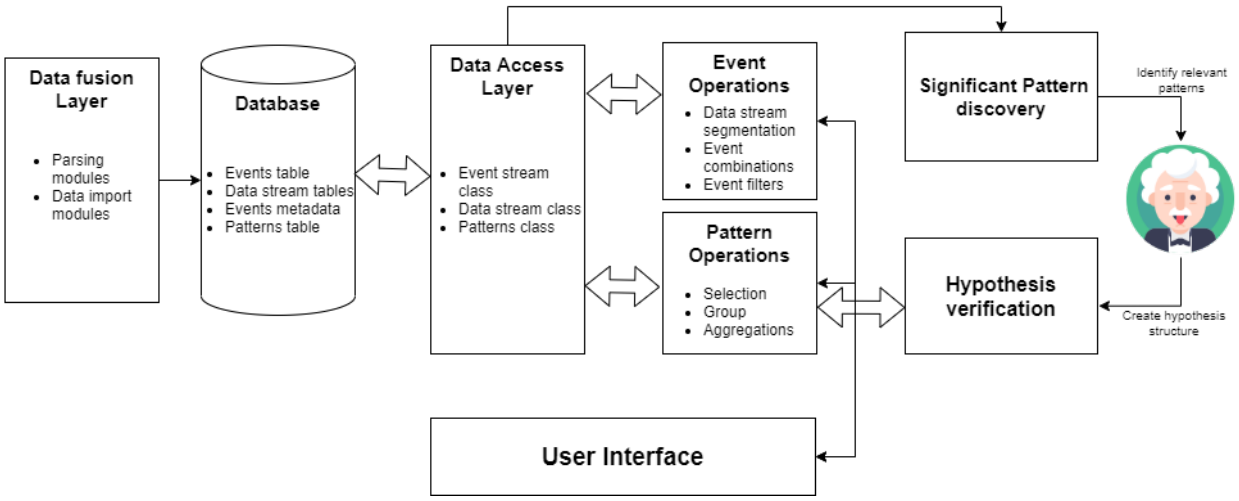

Figure 4.8: A high-level view of interactions between component systems of the event mining platform.

matching the event and data stream schema.

2. **Database**: A PostgreSQL database instance is used as the permanent event and data stream and pattern storage for the platform. PostgreSQL allows us to use the structured format of an event's time model for event and pattern operations while storing varying event parameters using a JSON field. We can also store the source for different events and data points, allowing us to use data streams and events based on the expected accuracy of the sources.

3. **Data Access layer**: Event stream, data stream, and pattern classes are implemented on top of the unified data store. This module interacts with the database and provides access to the required events, data streams, and patterns. Event and pattern operations access the data through these modules and write the results to the database. This layer also interacts with the User Interface (UI) and displays a selection of events to the analyst.

4. **Event operations**: Event operators create new event streams from existing events and data streams. These operations utilize the event creation operations discussed above and create an event stream object.

5. **Pattern operations**: Pattern operations are used to create new patterns from existing patterns and event streams. Pattern class from the data access layer is used to communicate with the database to read and write pattern occurrences and communicates with the UI layer to display and visually compare different pattern frequencies.

6. **User Interface** (UI): The UI layer allows the analyst to interact with the events, create new events and patterns, and examine and select patterns that are significant for the analysis. An events analysis session begins with a data selection panel, where the user can select the events, data streams, and the time range for the analysis. The selected events are displayed on a timeline and a polar plot to examine the individual's event patterns relative to the circadian rhythm. The interface also provides panels for event and pattern operations for creating new event streams and patterns. Different panels of the dashboard are depicted in figure 4.9

The event mining platform described in this chapter allows an analyst to perform exploratory analysis on the events set and initial analysis for finding causally significant event patterns. We will utilize the pattern, group and aggregate formulation while defining and testing hypothesis that encode the analyst's beliefs about the system or individual.

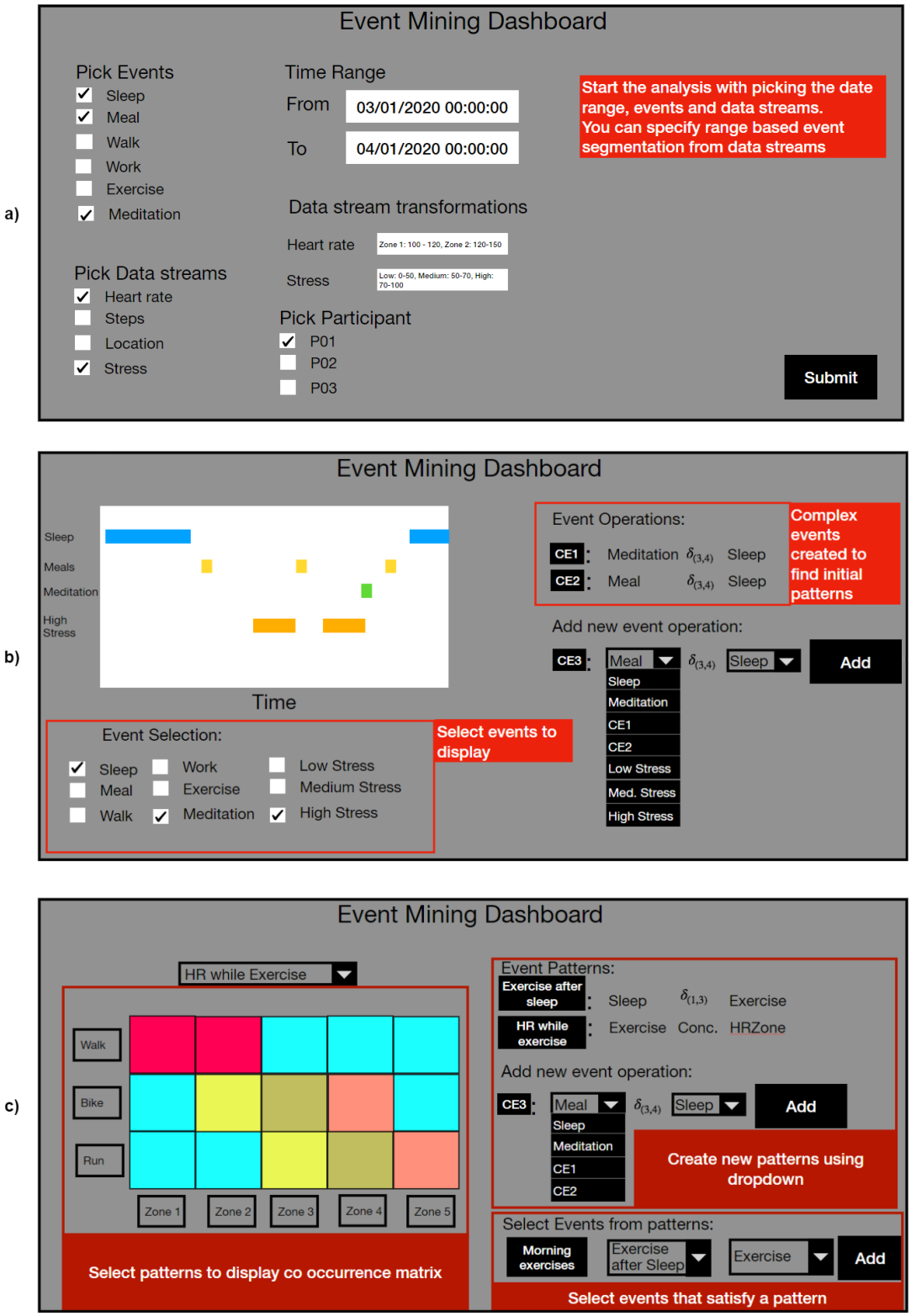

Figure 4.9: Different panels for exploratory analysis in the event mining dashboard. a) Data selection panel, where the analyst can select events, data streams and time range for the analysis, b) Events panel, where the analyst can visualize the selected events and create new events, and c) Patterns panel, where the analyst can create and visualize patterns between various event streams.

Chapter 5

Knowledge Integration and Hypothesis testing

Anecdotal beliefs, personal experiences, and commonly observed associations are frequently used to make personal and policy decisions. However, such untested assumptions could lead to unexpected errors in specific sub-populations and contexts. We attempt to include such patterns in personalized modeling after testing them in different contextual situations.

Testing and verifying long-held and commonly observed patterns is integral to the personalized modeling process. An n-of-1 modeling platform should enable users to test their beliefs about individual behavior, health, and environment. The hypotheses can be represented as relationships between events and event parameters in an event-driven framework. The event relationships or hypotheses can be derived from population-based knowledge, personal and anecdotal beliefs, and experience. Verifying the hypothesis allows us to make predictions and inferences about the system's behavior and state in different contexts. This is an essential step towards building recommendation and navigation systems to guide the system's behavior.

Knowledge and insights derived from population-based experimental trials are excellent sci-

entific tools for determining causal links. However, the quantitative effects determined from these experiments may not generalize very well. The same intervention can have very different responses across individuals due to personal attributes and is a widely recognized phenomenon in the scientific community [171]. The inconsistency of interventions has led to increased research in N-of-1 studies where observational or experimental data from a single individual is utilized to identify mechanisms and interventions specifically for the person. We use the event pattern language described in the previous chapter to identify events that are causally related to an outcome and compute the aggregated effect of such events on the outcome. This analysis is done at an individual level and can be viewed as an N-of-1 observational experiment. We refer to Granger's causality [172] when identifying the causal effect of past events on subsequent event occurrences and event parameters and utilize Causal Graphical Modeling principles for our analysis.

We propose a novel approach to build personal health models by integrating knowledge derived from external sources (e.g., Randomised Control Trials, personal beliefs) with multimodal observational data. The external knowledge sources provide us with a causal structure of the personal model in a Directed Acyclic Graph (DAG). We personalize that model by learning the coefficients and causal effects associated with different edges in the model using multimodal personal data. We utilize the Causal Graphical Modeling (CGM) [143] approach for estimating the causal effects of suspected relationships between multimodal parameters and computationally replicate the *do*-operator[143] to find the causal effects in various contexts. The combination of context, action, and outcome values describes the rules that define the personal model and can be updated over time using personal multimodal data.

5.1 Events in a cybernetic system

The goal of event mining is to understand systems that interact with their environment and their next actions are decided by the current state of the system and the feedback received from the environment. These interactions and the behavior of the system is monitored in form of events. Cybernetics and control theory have long attempted to describe such systems in a mathematical framework. The mathematical model in classic systems theory states that:

$$X[k+1] = A[k]X[k] + B[k]U[k]$$

$$Y[k] = C[k]X[k] + D[k]U[k]$$

Where X, U, and Y are the system true state, inputs, and measured output vectors respectively. A, B, C, and D are matrices that provide the appropriate transformation of these variables at a given time k.

Events are the agents of change in such dynamic systems. Every interaction with the environment (external event) affects the underlying system state. The state of the system, in turn, determines the next actions of the system (system generated events) which may affect the environment as well as the system state. This is true even for human bodies and various biological functions. A cybernetic view of the human systems and behavior allows us to interpret different individual, social and environmental events as an input to the system. Every event propagates a change in the system and impacts the occurrences of future events.

We will use the relationship between cardio-respiratory fitness and endurance activities among humans to highlight event interactions in a dynamic system and how we can utilize event mining operations to model the behavior of such system and identify the effect of external and system events on the system state.

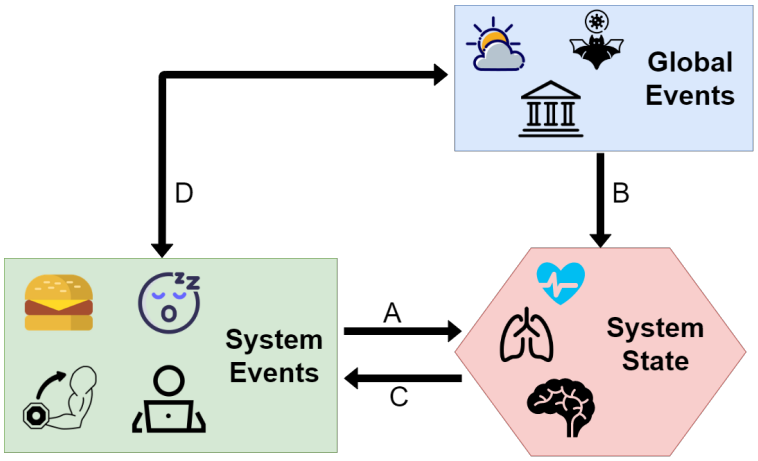

Figure 5.1: Event interactions between a dynamic system and the external environment. These interactions determine the future state and behavior of the system. Event mining operations can be used to articulate and test these relationships.

5.2 Causal relationships between events

Events (such as sleep and exercises) impact health state parameters such as aerobic fitness in human bodies. These health state changes, in turn, affect the occurrence of other events (e.g., heart rate changes during exercise) during our daily lives. Let us consider the effects of exercise habits on VO_2 Max, an essential aspect of cardio-respiratory fitness (CRF). VO_2 Max describes the volume of oxygen required by our bodies to function at maximal heart rate capacity and is a significant predictor of long-term cardiovascular health [155]. It is also widely accepted as a good predictor of performance in endurance sports [166, 23]. At the same time, repeated exposure to endurance training affects our pulmonary, cardiovascular, and neuro-muscular systems leading to improved delivery of atmospheric oxygen to mitochondria and improved exercise performance[69]. The magnitude of fitness improvement depends on multiple factors, notably the existing fitness levels and intensity, duration, and frequency of endurance exercises[196]. Though there is no consensus about the ideal duration of a training plan, some studies have observed a more remarkable improvement in VO_2 Max in the first six weeks of training. After that point, the CRF tends to stabilize, and any improvements observed in performance are due to other factors such as exercise economy

and lactate threshold[150, 68].

Recovery/resting periods also play a vital role in adapting to various exercise inputs. Too much training with little to no rest leads to overtraining[110] and does not allow the body to recover from exercise strain. On the other hand, too much rest leads to de-training[132] and does not have long-term health benefits. Sleep is essential to recovery between exercises and contributes to changes in CRF. Various sleep disorders affect post-exercise recovery, and successful interventions also lead to improvement in performance[167].

Therefore, we can infer from the above discussions that exercise events in the past six weeks have a causal impact on a person's current expected endurance capacity. However, the past exercise events can affect current cardio-respiratory fitness via multiple mechanisms that we will capture using different aggregation operations.

The impact of an event on the health state can be viewed as an impulse response that asymptotically returns to its equilibrium state (fig. 5.2). Thus the effect of any event on our health state (and other events) decreases over time, and the aggregated effect of the events occurring only in a specific time window (with non-zero impulse response value) in the past can be observed in the outcome. This natural tendency to achieve homeostasis keeps the effects of an event bounded and allows us to apply cybernetic principles while modeling the effects of lifestyle on health state parameters. It is possible to estimate the impulse response function and decay rate following specific events as shown by [212] in the context of forecasting blood-glucose response to different food items. However, this requires detailed high-frequency sampling of data streams for every individual. Due to this constraint, we have limited our approach to estimating the average effect of events over a larger time interval.

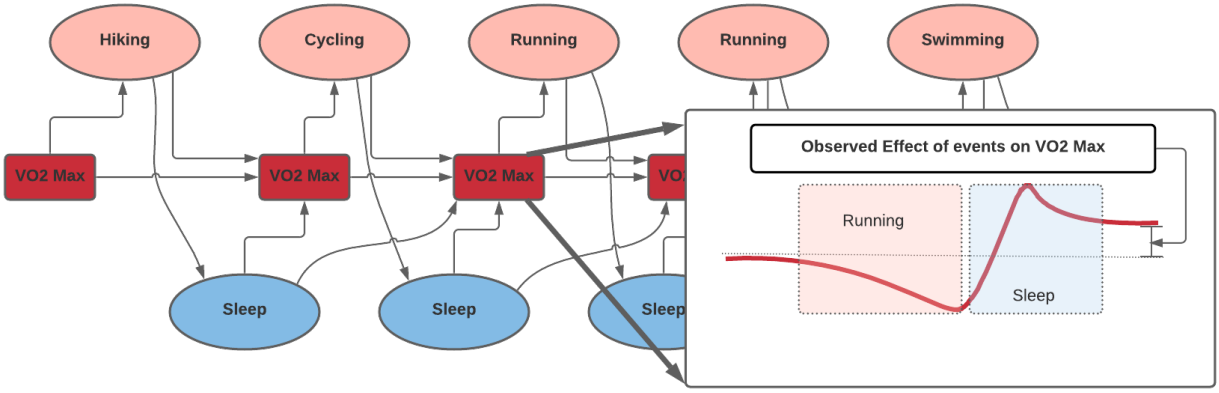

Figure 5.2: Exercise and Sleep events affect cardio-respiratory fitness. Every endurance exercise event (such as running) puts a volumetric stress on our heart, that reduces the maximal work capacity temporarily, but adequate amount of rest (sleep) can lead to an improvement in maximal work capacity. However, in absence of future exercise events, VO2 Max would asymptotically return to its baseline value. Thus, the effect of exercise and sleep on VO2 Max can be viewed as an impulse response that slowly goes down to zero.

5.2.1 Event patterns as causal links between events

As discussed in the previous section, events in the past impact current health outcomes and events. Any causal event relationship has at least three components, 1) Cause (event), 2) Outcome (event), 3) Temporal delay. An event pattern can represent these relationships if the cause and outcome events or parameters of an event. For example, endurance exercises in the last six weeks have significantly improved CRF [18]. This effect can be captured as a temporal event pattern, P_1, as:

$$P_1 = ES_{Exerc}\Delta_{0,42days}ES_{CRF}$$

We can convert any known biological event relationships to event patterns, which can be utilized for health state modeling and estimation. The bounded nature of the event impulse response allows us to use patterns for computational efficiency as only the events in a specific time window can have an observable effect on the outcome. Some example patterns are shown in table 5.1.

Cause	Outcome	Temporal Relationship	Pattern
Exercise	CRF	Past 42 days	$ES_{Exe.}\Delta_{0,42days}ES_{CRF}$
Resting	CRF	Past 7 days	$ES_{Rest.}\Delta_{0,7days}ES_{CRF}$
Exercise	Sleep	Past 7 days	$ES_{Exer.}\Delta_{0,7days}ES_{Sleep}$

Table 5.1: Examples of causal event relationships captured using patterns.

Different attributes of an event can affect the outcome event through separate mechanisms. For example, heart rate variations during an exercise event (measured by hr zone events) play a part in changing the physiology of our heart. At the same time, exposure to the sun during exercise (measured using exercise start and end time) impacts melatonin production, contributing to sleep onset and sleep efficiency. These factors, derived from the same event, have different effects on the outcome (CRF). These effects can be derived from the pattern that relates exercise events to CRF events (exercise events with a CRF value).

$$G_{hrzone} = P_1.groupBy(ES_{CRF}).apply(hrzoneDuration)$$

$$G_{exer.Time} = P_1.groupBy(ES_{CRF}).apply(averageStartTime)$$

The effect of an event attribute that has a causal effect on the outcome can be aggregated from the events that satisfy the temporal pattern. Thus, every group-aggregation operation represents a causal pathway from one event (exercise) to the outcome event (sleep).

5.3 Hypothesis Specification

Causal hypothesis representation should allow the expert users to incorporate their prior beliefs and external knowledge in the experiments. We have discussed how event patterns can be used to identify event pairs that are causally related; we will now attempt to extend

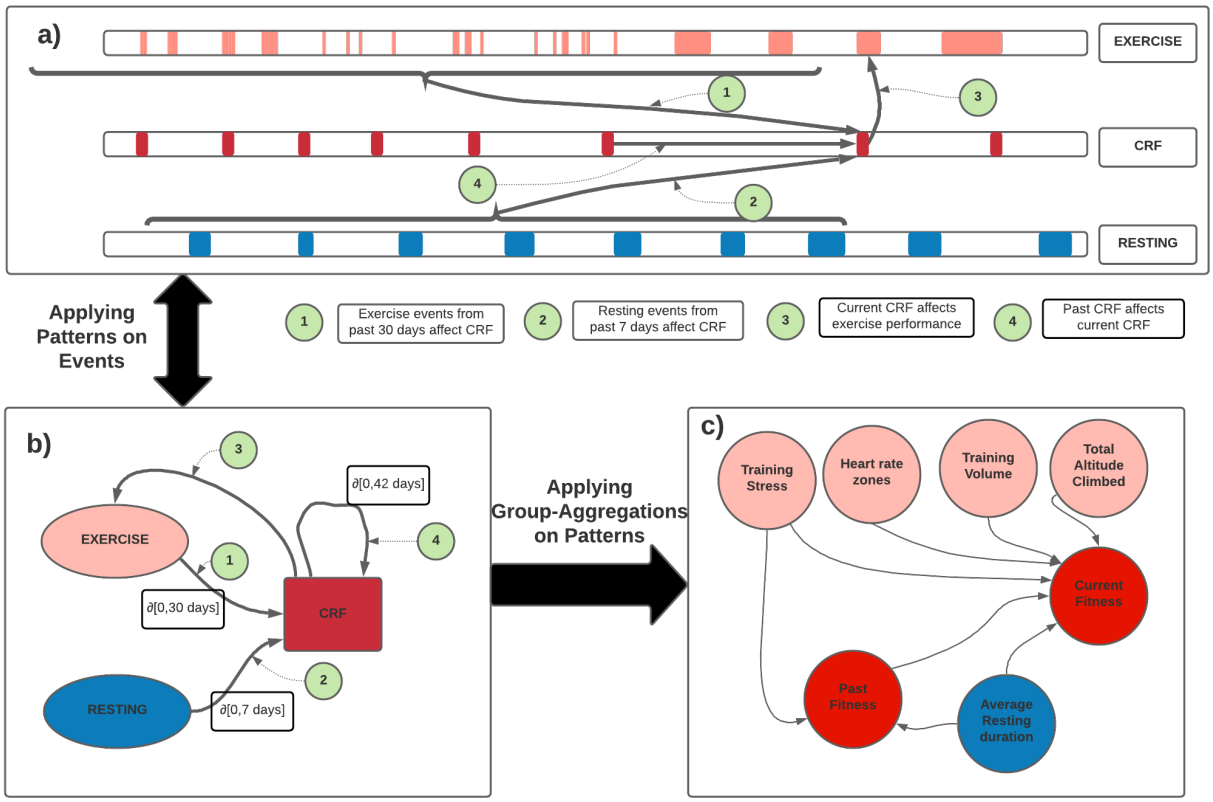

Figure 5.3: a) Events have causal relationships among themselves, and these relationships manifest themselves in different parameters and event occurrences. b) Pairs of such events can be captured using event pattern operators with appropriate temporal delay. c) Group and aggregation operators allow us to derive multiple causal factors from an event pattern that represent multiple causal pathways between the events.

this line of thinking using group-aggregate operations to find the aggregate causal effect of the past events causally related to the outcome.

5.3.1 Capturing knowledge as DAG

Graphs have been used to encode relationships between entities in different domains. Knowledge graphs are a commonly used data structure for encoding different types of relationships between different entities in the system—similarly, a hypothesis of known and suspected causal event relationships that need to be tested. These can be represented using a Directed Acyclic Graph (DAG) structure commonly known as *Causal Graphical Model*.

Causal Graphical Models (CGM) have been used extensively in varied scientific literature and experiments. CGMs are traditionally used to represent the causal assumptions between variables in an experimental setting. Different parameters are represented as nodes and a directed edge represents a causal dependence from *cause* to *effect*[143]. Missing edges between variables represent the lack of a known causal relationship between the variables. The assumptions in the causal graph originate from external knowledge sources (e.g., scientific literature, randomized trials, personal beliefs). The CGM can estimate the causal effect of one variable on an outcome using functional (Structural Equation Modelling) and probabilistic (Bayesian) estimation mechanisms.

We can describe such causal relationships between events in the form of CGMs. We can describe multiple causal pathways between events using "group-aggregate" operations, each operation generating one parameter in the graph. Pattern operations allow us to identify events that have a causal impact on the outcome of interest, and group-aggregate operations allow us to find out parameters of interest from the related events. These parameters are represented as nodes in the resulting CGM, and edges represent the causal relationships between these parameters(fig. 5.3). Since we are computing the variables by aggregating or selecting values from patterns, the possible causal links between the parameters are defined by the temporal relationship between the variables i.e., variables derived from the preceding events in a pattern can affect the parameters of the future event.

The causal graphical model thus generated can be considered a causal hypothesis. We can choose one of the relationships to test in the graphical hypothesis and find the causal effect of the selected parameter while applying the principles of Pearl's *do-calculus* [143] to determine the confounding variable set. Figure 5.2 illustrates how we can convert known event relationships to causal hypotheses using event patterns and group-aggregation operations.

5.3.2 Causal hypothesis

The specification of the hypothesis includes causal relationships between different parameters described as CGM and the definitions of the parameters in the form of event mining operations. We utilize the two components and apply the required statistical methods to test one or more of the links specified in the CGM. This hypothesis formulation allows us to easily include new relationships using the graphical structure and additional parameters using the event mining operators.

Hypothesis testing is accomplished in two steps:

1. Creation of data set using event mining operators: We can compute the values for each variable in the hypothesis using their corresponding event formulation. Due to the nature of the group-aggregate functions, each variable value is associated with an outcome event instance. Thus, processing all the event formulations results in a relational data set with each row of variables corresponding to an outcome event and can be considered a treatment unit.

2. Testing the statistical validity of relationships in the data set: We can use the graphical structure of the hypothesis to identify the minimum set of variables that would block all *backdoor paths*[144] between the intervention and the outcome node. This allows us to simulate the *do*-operator and find the effect of changing the intervention variable (or events) on the outcome.

The two-step verification process allows us to incorporate new causal relationships and enrich the model to understand the system better and observe novel behavioral patterns. We can also utilize the verified structure of the hypothesis to create models for predicting the state and behavior of the system. We will now present a formal definition of the hypothesis in terms of a DAG and event mining operations.

Formal hypothesis definition

The causal structure for the hypothesis is defined as a directed graph $\mathbb{G} = (\mathcal{E}, \mathcal{V})$ where $\mathcal{V}$ is the set of vertices in the graph and $\mathcal{E}$ is the set of edges in the graph that represent the causal relationships. Every node in the graph has an associated parameter(ν) defined as a pattern-group-aggregate operation. We first need to specify the causal event link as a pattern ($\mathcal{P}_i$) and apply the aggregation operation($\mathbb{F}$) to derive the parameter(ν). Therefore, for every causal link, e_i, let :

$$
\begin{aligned}
& e_i = (N_i^{in}, N_i^{out}) \in \mathcal{E} \wedge N_i^{in}, N_i^{out} \in \mathcal{V} \\
& \mathcal{P}_i := ES_i^{in} \mathbb{C} ES_i^{out} \\
& \mathcal{G}_i := \mathcal{P}_i.groupBy(ES_i^{out}) \\
& N_i^{in}.\nu := \mathcal{G}_j.apply(\mathbb{F}) \\
& N_i^{out}.\nu := ES_i^{out}.Param
\end{aligned}
\tag{5.1}
$$

As described in chapter 4, every value (ν_i^j) computed using a group-aggregation function is associated with an event from the outcome event stream ($e_j \in ES_{out}$). Thus, values(ν) associated with any node N_i has the structure:

$$N_i.\nu := \{(e_j, \mathbb{F}(\{e_k\}))\forall e_j \in ES_{out}, \forall e_k | (e_k, e_j) \in \mathcal{P}_i\}$$

This property of the aggregated groups allows us to relate each node's values with an event in the outcome event stream. Extending this over all the nodes in the hypothesis allows us to create a relational data set of all the parameters indexed by the outcome events and can be used for testing the hypothesis where every row in the data set can be considered a treatment unit.

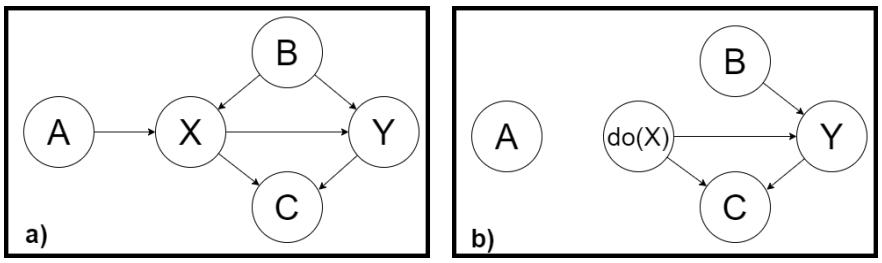

Figure 5.4: Causal relationships between variables can be captured in a DAG structure called **Causal Graphical Model (CGM)**. The *do*-operator can be used to find the causal effect of intervention (X) on the outcome (Y). Conceptually, the *do*-operator is equivalent to removing all incoming causal links to the intervention, that could lead to a *backdoor path* to the outcome. In the above figure, we remove the link from B to X (by controlling for B) but not on C as it is the collider node between X and Y, and blocking on it would open another back-door path from X to Y.

5.4 Hypothesis testing

There is a rich body of literature about CGMs and identifying minimal causal path blocking variables from the graphical structure itself[142]. Wright et al. first championed the use of causal path analysis [201] to answer causal queries and quantify causal effects. Pearl et al.[143] combined the causal path analysis with Bayesian modeling and provided the researchers a way to articulate causal relationships as separate entities from statistical concepts. Assuming that we capture causal event relationships as temporal patterns and capture the aggregated causal parameters using group operations, we can utilize the CGM principles to identify the causal effects of aggregated event parameters on the outcome.

Figure 5.3(c) describes a hypothesis that relates different endurance training metrics (derived from multimodal data streams collected with exercise events) with cardiovascular performance and cardiovascular fitness. These relationships are derived from the literature in the exercise physiology domain and tested in different clinical trials (discussed in the previous section). However, the causal effect of any one of the parameters is unlikely to be consistent across all individuals. Therefore, *we need to perform an N-of-1 analysis to utilize observational data collected for an individual and find the causal effect of different relationships in*

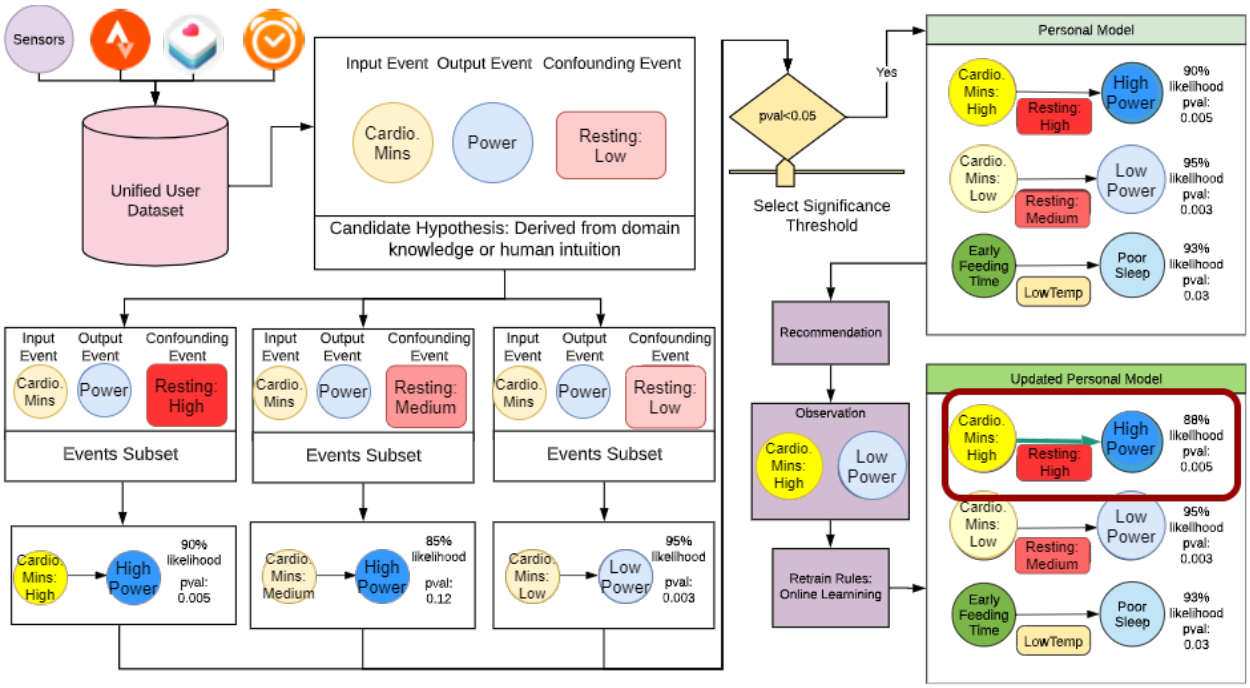

Figure 5.5: Detailed view of hypothesis testing framework. Treatment, outcome and covariates are derived from event operations associated with every node in the causal graph. The parameter values are then converted to categorical labels and the units are then matched using CEM (Coarsened Exact Matching). For each matched set we find the validity of the encoded relationship using Fisher's exact test and significant relationships are added to the set of rules that describe the model of the system. These rules can be continuously updated using incoming data and provide a causal understanding of the system (shown in the red outline).

our hypothesis. We can use the graph structure to find the minimal set of variables needed to block all causal pathways except the one being tested and simulate the *do*-operator to find the direct causal effect. For example, to find the causal effect of **training stress** (source) on **current fitness** (outcome), we need to observe **past fitness**, as it lies on an alternate causal pathway, in order to find the direct causal effect of the source on the outcome.

5.4.1 *do*-operator

As described in Pearl et al. [143], we can consider the *do*-operator, $do(X = x)$, as an intervention in the causal mechanism that sets the random variable X to value x while not

disturbing any of the non-descendant nodes in the causal graph. Graphically, intervening on X can be viewed as removing all incoming edges to the node representing X as shown in figure 5.4. It can be defined as follows:

$$P(Y = y|do(X = x)) = \sum_{z} P(Y = y|X = x, Z = z)P(Z = z)$$

where Z is the set of intermediary variables intercepting any indirect causal paths between X and Y. The additional set of variables to be controlled or covariates (Z) is determined from the causal graph using *backdoor principle*[142]. We can find the direct causal effect of any intervention (*do(X=x)*) on the outcome (Y) as

$$CDE = E(Y|do(X = x), do(Z)) - E(Y|do(X = x'), do(Z))$$

where x' is the baseline against which the effect of an intervention is to be measured.
In addition to the causal effects, this modeling paradigm also allows us to answer queries about counterfactual events and attribute specific outcomes to individual events or a set of events. These queries are impossible to answer in a purely statistical model but are essential for understanding the system's functioning.

5.4.2 Unit Matching and testing

The *do*-operator provides us with a conceptual framework for identifying the causal effects of different interventions and explains that it is essential to remove any bias due to covariates to understand causal relationships. Randomized experiments provide an experimental avenue for removing this bias by using an ignorable treatment assignment that is independent of the covariates. However, this approach is not possible when using observational data to estimate causal effects as all the events and data collection happened in the past. *We attempt to replicate the randomized treatment assignment by obtaining treatment and control groups*

with similar covariate distributions[179]. Once we have matched control units with similar treatment units (with respect to the covariates), we can use different techniques (e.g., t-test, linear regression, CART, etc.) to estimate the treatment effect for each treatment unit. The treatment effect reported in this case is the *Average Treatment effect for the Treated* or ATT. The first step in this analysis is identifying the appropriate set of covariates for matching the units. Typically, researchers tend to be liberal when choosing the covariates as including variables that are not associated with the outcome can lead to a slight increase in variance, but excluding a potentially important (correlated) variable can lead to a significant increase in bias[179]. However, we should also be careful not to include any variables that the treatment or intervention of interest may impact. Using CGM for identifying the covariate set alleviates these concerns as we are only choosing the set of variables that remove the causal impact from variables except the ones included in the intervention[169].
Causal analysis using matching methods can be largely divided into four steps that are discussed below:

Distance Metric

Distance metrics define "closeness" or similarity between different units based on the covariate values. We will discuss three common ways to define the distance $D_{i,j}$ between units i and j.

1. *Exact Matching*: This method directly compares the covariate values(Z) and returns a binary similarity score. It can be defined as :

$$D_{i,j} = \begin{cases} 0, & \text{if } Z_i = Z_j \\ 1, & \text{otherwise} \end{cases}$$

   Exact matching is the ideal distance metric in many ways[60], but it can lead to

many unmatched units in the case of high dimensional covariates. However, *coarsened exact matching*(CEM) allows us to utilize exact matching on a more extensive range of variables, for example, defining categories based on total exercise minutes in the last 42 days rather than using the continuous values of the variable. We have used CEM in our experiments discussed in chapter 7 for causal effect estimation.

2. *Mahalanobis distance*: It can be defined as:

$$D_{i,j} = (Z_i - Z_j)'\Sigma^{-1}(Z_i - Z_j).$$

where Σ is the covariance matrix of Z in the complete control group.

Mahalanobis distance can work well with relatively few covariates, but it does not perform well when the covariates are not normally distributed or are high dimensional.

3. *Propensity score*: Propensity score($e_i(Z_i)$) for a unit i is defined as the probability of receiving the treatment given the observed covariates, $e_i(Z_i) = P(T_i = 1|Z_i)$, and can be estimated using methods like logistic regression, CART and generalized boosted models. We can use propensity scores to match units in place of the covariate values because the distribution of covariates is identical for both treatment and control groups at every value of the propensity scores. If the treatment assignment is ignorable given the covariates, then the same holds for the propensity scores. We can define the distance between two units i and j as:

$$D_{i,j} = |e_i - e_j|.$$

A variation of propensity score, known as "linear propensity score" has been found to be very effective in reducing bias[160, 159] and can be defined as

$$D_{i,j} = |logit(e_i) - logit(e_j)|.$$

Matching method

We can use the selected distance metric for matching similar units. The goal of the matching is to find a set of control units that resemble the treatment units in covariate distribution and thus allow us to simulate randomized experiments from observational data. Some commonly used matching techniques are:

1. *k:1 nearest neighbor*: In this method, we match a treatment unit with k nearest treatment units in terms of the distance metric selected. Nearest neighbor methods always simulate the ATT as we attempt to find the nearest control units to every treatment unit and may exclude control units that are not close enough to any treatment unit.

2. *Subclassification and Full matching*: Subclassification matching methods search for groups of units that are similar as defined by the distance metric though the number of groups or clusters need to be specified before matching. Full matching applies a similar methodology; however, they select the number of groups automatically. Each matched set of units contains at least one treatment and one control unit and can be used to estimate both ATT and ATE (Average Treatment Effect).

3. *Weighting*:Propensity scores can also be included directly in estimates as inverse weights. One commonly used weighting scheme is *Inverse probability of treatment weighting* and the corresponding weight can be defined as

$$w_i = \frac{T_i}{e_i} + \frac{1 - T_i}{1 - e_i}$$

   There are many such weighting schemes found in causal inference literature and can be found in [179].

Quality of matching

Determining the quality of matching is very important for the matching analysis. Conventional metrics of accuracy or model performance do not necessarily capture matching quality, as our goal is to minimize the differences in covariate variable distributions between treatment and control groups. We can compare the multidimensional histograms of all covariates between the treatment and control groups; however, this approach might not be viable in all problems as the histograms may be very coarse and may have numerous regions with zero value. Additionally, different numeric and graphical methods can be used to evaluate the similarity of distributions.

1. *Numeric methods*: Some commonly used numerical metrics are standardized difference of means and ratio of variances between treatment and control groups for all covariates or propensity scores.

2. *Graphical methods*: QQ (quantile-quantile) plots can be used to compare the empirical distribution of each covariate in the treatment and control group. Plots of standardized difference of means of variables between unmatched and matched sets can also inform us whether matched units are closer to each other than unmatched units.

Outcome Analysis

Matching methods result in treatment and control groups with adequate covariate balance, and we can now move on to the analysis stage. We utilize Fisher's exact test to identify the differences in the outcome distribution for the treatment and control units which is interpreted as the average treatment effect for the treated (ATT). The weighted sum of estimates for each matched set leads to the final estimates ATT.

5.5 Use cases

These steps result in rules that represent causally significant relationships between the aggregated event features and the outcome events included in the patterns. Patterns derived from external knowledge sources are verified using the above mentioned steps and are included for future hypothesis tests if found to have a significant average treatment effect. These rules describe the system behavior and are inherently interpretable. Thus, the rule-based model enhances our understanding of the system and can help identify effective interventions and recommendations.

The verified hypothesis can also be used as a template for models to predict system state and behavior. The incoming links to a node included in the verified hypotheses can be viewed as known causal factors for the parameter represented by the node and can be used as features to train a prediction model for the parameter. We have performed several experiments to show the utility of this modeling framework in the context of personal health and lifestyle interactions and will be discussed in chapter 7.

Chapter 6

Data-driven temporal event pattern discovery: Hypothesis discovery

Events from different sources and recognized from multimodal data streams form the history or the *lifelog* of the individual. The lifelog of an individual contains a digital representation of their actions, habits, and lives and can thus be utilized to discover their repetitive behaviors. The data-driven discovery of frequent event patterns adds value to the N-of-1 modeling by enabling the analyst to discover unknown event relationships idiosyncratic to the individual and identify contextual situations determined by the events preceding the outcome of interest that may not have a direct causal impact but affects the outcome indirectly.

The events are available to the analyst in the form of a large sequence of timestamped events. For example, $ES =< (E_1, t_1), (E_2^+, t_2^+), (E_3, [t_3^+, t_3^-) >$ represents a sequence of 3 events with associated temporal information. Each event can be one of the three types (point, interval, or semi-interval event).

In this chapter, we will discuss how we can derive frequently observed patterns of events. We propose an episode mining approach to derive frequent patterns. The episodes are defined relative to an outcome of interest and a time range that includes the relevant events. We

discuss how multimodal data associated with events can reduce the variance within events of the same type (e.g., find different sleep events). After obtaining the symbolic representation for similar events, we can discover the frequent event patterns. We utilize semi-interval events to describe event patterns as described by Moerchen et al. [116]. We adopt a novel tree-based indexing of semi-interval event sequences to index commonly observed episodes and discover frequent patterns by traversing the tree in a depth-first manner. We will evaluate the proposed approach using a simulated dataset with controllable error rates and randomly injected fixed event patterns.

6.1 Temporal Pattern discovery

Temporal patterns describe associations between different temporal events observed in the system. The majority of works in sequential and temporal pattern mining have focused on discovering frequent patterns as defined by the support of the pattern. These patterns and corresponding sequential pattern discovery algorithms were discussed in chapter 3.

However, not all *interesting* patterns are likely to be frequent. For example, event patterns containing a relatively rare event are unlikely to be frequent in the set of all event patterns. Therefore, we adopt an outcome-dependent frequent episode discovery approach, i.e., discovering patterns that lead to an outcome of interest. This makes it possible to discover patterns leading to the outcome, and the analyst can then interact with the pattern discovery operator to further refine the patterns by specifying the events they want to exclude from the analysis.

6.1.1 Multimodal Event clustering

Different frequent pattern and episode mining paradigms for temporal events assume a symbolic representation for temporal events. All events representing the same user activity (e.g., Sleep) are represented by the same symbol in such methods. However, a significant distinction between such symbolic events and real-life events is the variation among the events representing the same activity. For example, individuals may describe their sleep events as excellent or poor based on how rested they feel after the event. Oftentimes, the goal of the analysis is to find patterns that lead to these variations in the outcome event. Therefore, the analysis needs to identify different categories of the outcome event as required by the goal of the analysis.

Multimedia and multimodal event representations are useful for addressing this problem, as an event's informational and experiential aspects capture the associated information for every event. We can utilize an event's informational and experiential parameters to redefine the event categorization for lifestyle events (such as sleep, walking, and running).

We can define multiple ways to relabel events using the event mining operations and commonly employed machine learning techniques:

1. **User-defined categorization**: Analysts can use event filter operations (define in chapter 4) to create new event streams from existing events that match user-defined constraints. This operation can also relabel the events based on constraints derived from domain knowledge or the analyst's intuition. For example, if the analysts want to find the differences between patterns leading to longer sleep events (sleep duration ≥ 8 hours) and shorter sleep events (sleep duration < 6 hours), they can define a new

event stream (ES_{new_sleep}) as follows (Eq. 6.1):

$$
\begin{aligned}
ES_{long\_sleep} &= \sigma(ES_{sleep}, \Phi := duration \geq 8hrs) \\
ES_{short\_sleep} &= \sigma(ES_{sleep}, \Phi := duration < 6hrs) \\
ES_{new\_sleep} &= ES_{long\_sleep} \vee ES_{short\_sleep}
\end{aligned}
\tag{6.1}
$$

2. **Sub-event based categorization**: Experiential information associated with an event can be used to derive sub-events within an event in the database. These sub-events can then be utilized to derive additional informational parameters for the events and used as any other parameter for relabeling the events. For example, different running events with identical duration and calories burned could drastically impact a person's body depending on altitude variations during the run. This aspect of the event can be captured using an event mining operation (Eq. 6.2) to discover uphill and downhill running sub-events and using the aggregate duration as event parameters.

$$
\begin{aligned}
&ES_{uphill\_run} = Run \wedge detect - climb(Altitude) \\
&ES_{downhill\_run} = Run \wedge detect - descent(Altitude) \\
&Ascent\_duration = (Run \perp ES_{uphill\_run}).groupBy(Run).apply(\Phi := sum(duration)) \\
&Descent\_duration = (Run \perp ES_{downhill\_run}).groupBy(Run).apply(\Phi := sum(duration))
\end{aligned}
\tag{6.2}
$$

3. **Event Clustering**: The informational event parameters can be utilized as features to describe a multimedia event. These features can be used to train an unsupervised model to find prominent event clusters. This method can relabel events in a completely data-driven manner and assumes that events with similar features have similar semantic meaning and have similar event patterns associated with them. Figure 6.1 depicts the clustering approach implemented on PMData dataset[180] collected for 16 individuals

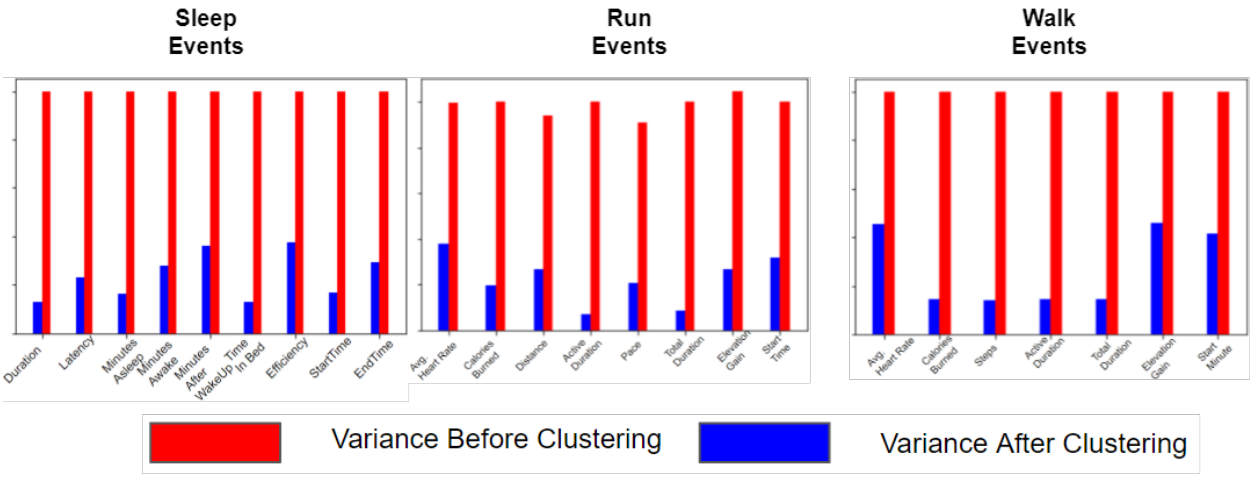

Figure 6.1: Events of the same type can still have a lot of variance in terms of the experience of the events. This variance is captured by the event features and concurrently captured multi modal data streams. Clustering on these related parameters can help us find new event labels that lead to reduced variance in events with the same label. This figure shows the reduction in values of event parameters for sleep, run and walk events after clustering the events to find clusters of events with similar experiential and informational aspects.

over a period of 5 months[1].

6.1.2 Event Episode

Event episodes are the set of temporally ordered events that should be considered when discovering the frequent patterns. These are defined by a time window, or *episode duration*, relative to an instance of the outcome event. Event episodes include all events from relevant event streams (as determined by the analyst) that occur in the specified episode duration before the outcome event. All frequent patterns relevant to understanding the outcome event occurrences are assumed to occur in the time window defined by the episode duration. Formally, this can be defined as :

Definition 6.1 (Event Episodes)**.** *Given an outcome event stream* ES_{out}*, episode duration* $\tau = [t_1, t_2]$ *and the set of relevant event streams* $ES_{in} =< ES_1, ES_2, ..., ES_k >$*, an episode*

[1] `https://datasets.simula.no/pmdata/`

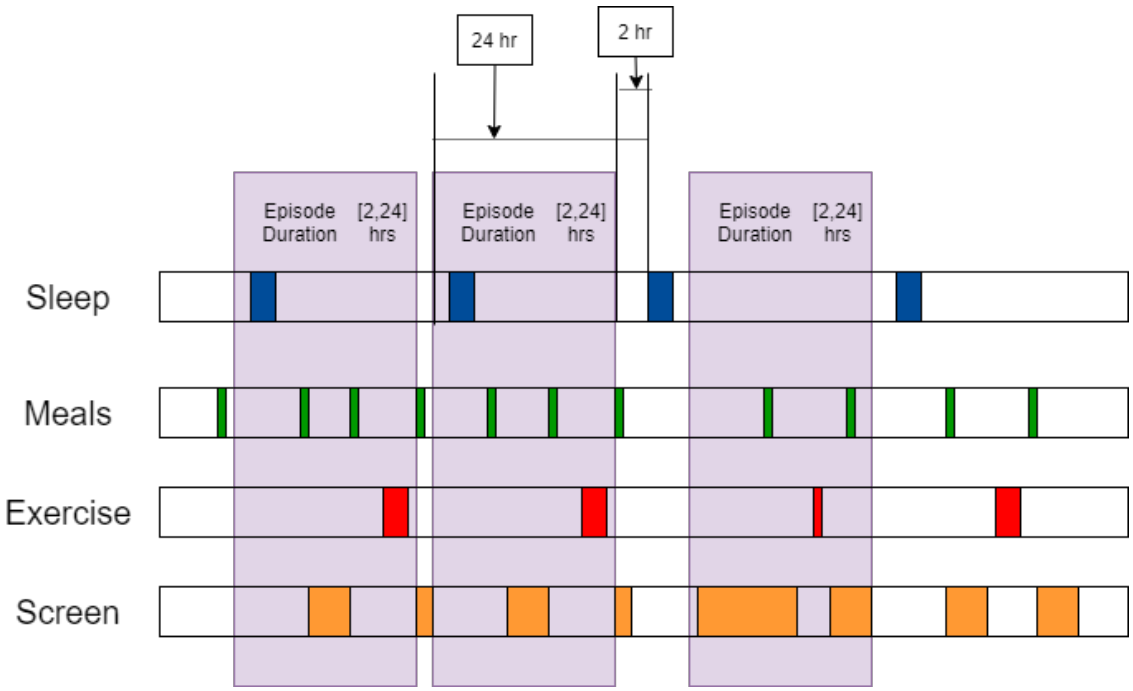

Figure 6.2: Event episodes highlighted on a timeline. This figure shows an example of event episodes relative to **sleep** events and includes **meals**, **exercise** and **screen activity** events. Every event in the sleep event stream has a corresponding event episode. Episode duration is [2, 24] hours, therefore events from the selected event streams occurring in the specified window are included in an event episode.

(ε) contains all events that are followed by an event from the outcome event stream in the time interval, τ.

$$\varepsilon = \{(e^j_{out}, \varepsilon_j) \forall e^j_{out} \in ES_{out}, \varepsilon_j = \{e_{in} | \forall e_{in} \in \bigcup_{i=1}^{k} ES_i \wedge e_{in} \Delta_\tau e^j_{out}\}\}$$

where Δ_τ represents the conditional sequential constraint between events.

Figure 6.2 depicts a pictorial representation of event episodes. Commonly used pattern mining frameworks result in many spurious patterns if we lower the threshold to discover rare or anomalous patterns. However, defining the event episodes concerning an outcome of interest allows us to discover patterns related to a rare event using the same frequent pattern mining framework and lower support thresholds as the number of episodes is determined by the occurrence frequency of the outcome event.

6.1.3 Event and Pattern model

Once we have converted the multimedia events to their symbolic representation and created the required episodes, we can use the episodes to index the sequential event patterns in a tree structure where every path from the root node (the outcome event) depicts a sequential event pattern. We will discuss the event and pattern representation used to create the event episodes and index sequential patterns.

Event Model

As discussed in the previous sections, temporal events are usually represented as points, intervals, or semi-interval events. Temporal point event representation is commonly used in pattern discovery applications. However, it is insufficient to incorporate interval events in the same framework. Different works integrate both interval and point representations for pattern mining, or *hybrid temporal pattern mining*[203]. However, the interval relations described using interval algebra are more restrictive as compared to semi-interval event algebra and can lead to some pattern occurrences not being recognized despite matching the desired event order[116].

We will be using the semi-interval event representation in this work. Therefore, any interval event E spanning over time interval (t_i, t_j) will be represented as two tuples, (E^+, t_i) and (E^-, t_j), where E^+ and E^- represent the start and finish end points of the interval event E. Thus, the event episodes are likely to contain interval end-points, and the patterns need to be defined as semi-interval event sequences. These sequences can be easily translated to interval event pattern operations described earlier in this work.

Pattern Model

The semi-interval event representation allows us to describe different event relationships with sequential operators, which can be extended to incorporate the temporal gap distribution between two event pairs. A single semi-interval event pattern can incorporate patterns described by multiple interval arrangements. Semi-interval patterns also allow us to include partially ordered event patterns. Thus, the flexibility of the semi-interval event patterns allows them to be discovered at a smaller support threshold as compared to the corresponding interval event patterns (fig. 6.3). This has also been observed in experimental studies comparing the semi-interval pattern matching with interval patterns described using Allen's algebra [119].

Morchen et al. [116] discuss two categories of patterns described using semi-interval events *Semi-interval Sequential Patterns* (SISP) and *Semi-interval Partial Order* patterns (SIPO). SISPs are sequences of itemsets or events with *strict order* between all semi-interval events in the pattern. SIPOs, on the other hand, allow a *partial order* between itemsets or events in the pattern which can be represented by a directed acyclic graph. The example SIPO in figure 6.3 specifies strict orders between A^+ and A^-, but no order is specified between A^+ and B^+. We extend this representation by associating a temporal constraint with each edge in the SIPO and SISP.

In our work, we propose an approach for discovering frequent pairwise conditional sequential semi-interval event patterns leading to the corresponding outcome event for the event episodes. We utilize clustering methods to find common temporal delays between a pair of events that describe frequent conditional sequential patterns between the events. The pairwise patterns can be used to remove events belonging to infrequent patterns from the episodes.

The filtered episodes are used for indexing events in a tree-like structure (fig. 6.4) that stores different SISP leading to the outcome event. The root of the tree represents the outcome

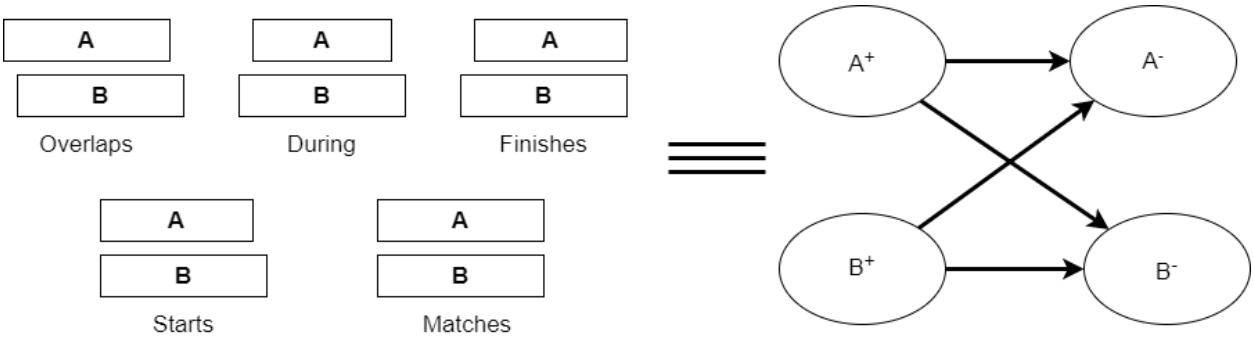

Figure 6.3: Semi-interval Partial Order (SIPO) patterns allow increased flexibility compared to interval algebra patterns as situations requiring multiple interval algebra patterns can be described using a single SIPO and thus can be discovered at a higher support threshold and reducing the number of spurious patterns. SIPO also have the added advantage of requiring only two types of event relationship operator (sequential and concurrent), which can be easily extended to incorporate the observed temporal gap distribution between the events.

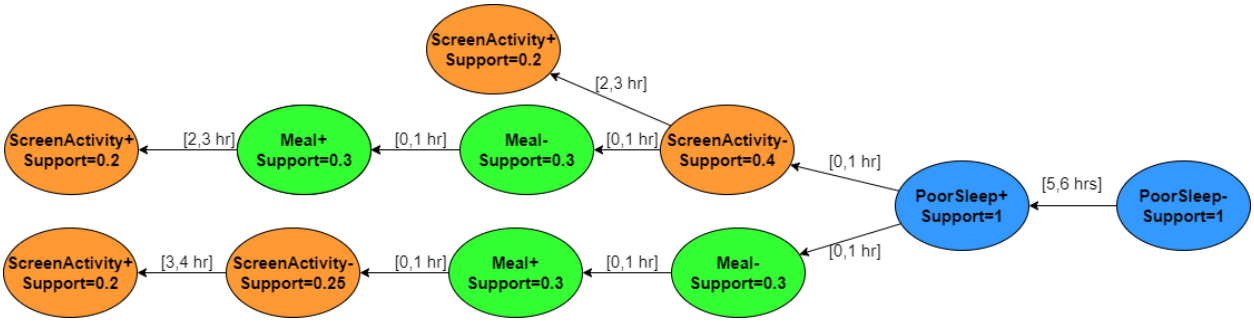

Figure 6.4: A sample tree model of semi-interval events leading to **poor sleep** events. Every path on the tree starting from the root node represents a sequential pattern.

event used as the reference for creating the episodes. Every edge in the model connects an event to a preceding event and stores the time gap distribution observed for the event pair. Every path originating from the root node represents a SISP originating from the outcome event and connects to the previous event in the sequential pattern.

6.2 Pattern Discovery Algorithms

This section will discuss the algorithms used for discovering the pairwise frequent conditional sequential event patterns and the tree-based event indexing that can be used to find larger

sequential patterns.

6.2.1 Pairwise Event Pattern Discovery

Pairwise frequent conditional sequential patterns capture the temporal relationship between two events. Finding all pairwise event patterns can have $O(N^2)$ time complexity for N events, and adding the complexity of searching for common delays between events can significantly increase the time complexity of the problem. However, we are only concerned with patterns that lead to the desired outcome event, which allows us to find all delay values between event pairs in a single pass of data; we can then utilize appropriate statistical method (e.g., clustering, t-tests, peak finding) to find different distributions of the observed delay values.

We present an approach for discovering all pairwise conditional sequential patterns that lead to the outcome event (described in algorithm 1). This is accomplished using a two-step approach. The first step requires iterating through all the episodes in reverse chronological order and store all observed delays between pairs of events (lines 4-9) in a hash map. We then utilize a mixture-model based clustering approach to find constituent distributions for the observed delay values and utilize the maximum, and minimum values observed for each component of the mixture-model to find the different delay ranges (lines 10-18). Thus, for each event pair (e_{in}, e_{out}) we find a set of time-intervals $\{\tau_j\}$ that define the commonly observed delays between the two events.

The derived patterns can be used as a filter to remove non-frequent event occurrences before indexing the episodes. This is possible due to the *downward closure* property of support of patterns. This property states that for two sequential patterns $\rho_i =< e_{i,1}, e_{i,2}, ..., e_{i,m} >$ and $\rho_j =< e_{j,1}, e_{j,2}, ..., e_{j,n} >$ and ρ_i is a sub-sequence of ρ_j i.e. $\exists\{k_1, k_2, ..., k_m\}$ such that $e_{i,1} \subseteq e_{j,k_1} \wedge e_{i,2} \subseteq e_{j,k_2} \wedge e_{i,3} \subseteq e_{j,k_3} ..... \wedge e_{i,m} \subseteq e_{j,k_m}$, then $support(\rho_i) \geq support(\rho_j)$.

Algorithm 1 Frequent pairwise conditional sequential pattern discovery

Require: $episodes = \{(e_i, t_i) | \forall i, t_i \geq t_{i+1}\}$ ▷ Events are in reverse chronological order
Require: MinSupport
delayMap := HashMap(string, list[numeric])
delayIntervals := HashMap(string, list[numeric])
outputInterval := HashMap(string, list[numeric])
for $i, episode \in enumerate(episodes)$ **do**
    $e_{out} := episode[1]$
    for e_{in} in episode[2:] **do**
        delayMap[e_{in}].append($e_{out}.t - e_{in}.t$)
    end for
end for
for k in delayMap.keys() **do**
    delayValues := delayMap[k]
    delayIntervals, intervalMembership := clusterDelays(delayValues)
    for $j, range \in enumerate(delayIntervals)$ **do**
        if $|\{m_i | \forall m_i \in$intervalMemberships$\wedge m_i = j\}| \geq$MinSupport **then**
            outputInterval[k].append(range)
        end if
    end for
end for
Output outputInterval

This is also true for temporal patterns, thus if any temporal pattern, ρ, contains a sub-pattern $e_i \Delta_\delta e_{out}$ that is not a part of any frequent pairwise patterns between e_i and e_{out} then $support(\rho) < min.support$. Thus, we can safely remove all occurrences of e_i from episodes that do not match one of the frequent pairwise patterns. We use this property when creating the tree model of event episodes to reduce the number of spurious patterns in the model.

6.2.2 Tree based episode indexing

As discussed in the previous section, filtering the episodes using frequent pairwise patterns leads to reduced noise in the episodes, and the variations in the episodes are caused only due to frequent patterns. Therefore, we can use the filtered event episodes to discover larger frequent patterns (algorithm 2). We present an algorithm for indexing all filtered episodes in a tree structure (described in algorithm 3), which can then be utilized to find frequent temporal patterns of semi-interval events.

The root node represents an empty episode, and the subsequent events in the episode are processed in reverse chronological order. Events in an episode are added to the tree in a depth-first manner, and a new node is added when the episode branches from or extends previously observed episodes (lines 3-10). Every node in the tree stores the episode identifier and the timestamp for every event matched to the node and links to the subsequent nodes representing the next events in different episodes (lines 8-10). The timestamp information is utilized to find the corresponding delay values in the tree between neighboring nodes (representing consecutive events in the episodes). The distribution of delay values between a node and its parent node is associated with the node and describes the temporal relationship between the events. The nodes also contain the number of occurrences of the sequential pattern starting from it and leading to the outcome event corresponding to its support in

Algorithm 2 Filter episodes using frequent conditional sequential patterns

Require: $episodes = \{(e_i, t_i) | \forall i, t_i \geq t_{i+1}\}$
Require: $eventPairs = \{(e_{in}, \tau) | supp(e_{in} \Delta_\tau e_{out}) \geq$MinSupport$\}$ ▷ Frequent event pairs from algorithm 1

```
filteredSequence := list[list[events]]
for i, episode ∈ enumerate(episodes) do
    filteredEpisode := list[events]
    e_out := episode[1]
    filteredEpisode.append(e_out)
    for e_in in episode[2:] do
        delay := e_out.t − e_in.t
        for τ ∈ eventPairs[e_in] do
            if τ[0] ≤ delay ≤ τ[1] then
                filteredEpisode.append(e_in)
                break
            end if
        end for
    end for
    filteredSequence.append(filteredEpisode)
end for
Output filteredEpisode
```

the episode database.

Thus, the tree model stores all observed episodes that lead to the outcome of interest and can be used to discover different types of patterns (closed or maximal). Every path from a root node describes a SISP that leads to the outcome event. We can take advantage of this property to identify closed patterns from the tree model. Additionally, the occurrence count associated with each node allows us to calculate the support value for the pattern and can be used to identify frequent patterns.

6.2.3 Frequent closed SISP extraction

We extract frequent closed SISPs from the tree model using a recursive depth-first traversal algorithm (algorithm 4). A depth-first traversal of the tree explores all sequential episodes available in the tree to extract sequential closed patterns from the tree. We incrementally

Algorithm 3 Index events in the suffix tree structure to discover frequent sequential patterns

Require: $filteredEpisodes = \{(e_i, t_i) | \forall i, t_i \geq t_{i+1}\}$ ▷ filtered episodes from algorithm 2

```
1: TreeModel := HashMap(string,EventTreeNode)
2: for i, episode ∈ enumerate(filteredEpisodes) do
3:     e_out := filteredEpisodes[1]
4:     curNode := TreeModel[e_out]
5:     curNode.addTimeStamp(i, e_out.t)
6:     curNode.incrementCount()
7:     for e_in in filteredEpisodes[2:] do
8:         curNode := curNode.next[e_in]
9:         curNode.incrementCount()
10:        curNode.addTimeStamp(i, e_in.t)
11:    end for
12: end for
13: Output TreeModel
```

Algorithm 4 Recursive depth-first traversal of the event tree model to obtain frequent patterns

Require: EventNode ▷ Node from tree model of episodes from Algorithm 3
Require: Min.Support
Require: AllPatterns ▷ Contains all frequent closed SISP upon completion
Require: CurrentPattern

```
if EventNode.support < Min.Support then
    if CurrentPattern.size > 0 then
        AllPatterns.add(CurrentPattern)
    end if
else
    CurrentPattern.push(EventNode.delayRange)
    CurrentPattern.push(EventNode.event)
    if EventNode.next.size ≠ 1 ∨ then
        AllPatterns.add(CurrentPattern)
    end if
    for n ∈ EventTree.next do
        AllPatterns := RecursiveTraverse(n, Min.Support, AllPatterns, CurrentPattern)
    end for
end if
Output AllPatterns
```

add events and delays to the pattern until we observe a drop in support, at which point the currently explored pattern is added to the set of all patterns. The patterns are extended until we have exhausted the current branch of the tree or the support drops below $Min.Support$. However, it should be noted that the patterns extracted in this manner do not include all possible closed sequential patterns from the tree, and further research is required for extracting the complete set of closed SISP from the tree model.

6.3 Analysis process

The approach described in this chapter can be utilized to power an interactive frequent SISP discovery framework. The analyst can interact with a visual representation of the tree model and identify the events or nodes of the tree that represent spurious patterns. This allows the analyst to refine the patterns until they arrive at a suitable model of the event sequences preceding the outcome event. Since every node stores the episode identifiers and time stamps for every associated event, re-indexing the tree can be accomplished by removing the events represented by the deleted node from their respective episodes and adding the concerned episodes to the tree model.

The tree indexing algorithm (algorithm 3) has a run time complexity of $O(N)$. Therefore, the interactive re-indexing of the tree is feasible and can allow the analysts to explore event patterns on the go.

6.4 Simulated data experiments

We tested the utility of the pattern discovery algorithm described in this chapter for discovering frequent patterns in event sequences using a simulated data set. We also utilize the PMData data set[180] for discovering frequent SISP that lead to different types of sleep

events, categorized using a clustering approach.

We utilized a simulated data set to understand the approach's efficacy in discovering patterns of different lengths in the presence of random noise. The noise here refers to randomly generated events that do not represent any of the frequent event patterns.

6.4.1 Data generation

The simulated data consists of a single sequence of semi-interval events controlled by a set of user-defined parameters. The number of distinct events in the sequence is defined by n_e. There can be $2n_e$ distinct symbols in the event sequence representing the beginning and end point of every interval event. The total number of events in the sequence are determined by the parameter N. The algorithm randomly generates an event after an interval defined by parameter δ, called the minimum interval. The probability of random event generation is determined by parameter α where $0 < \alpha < 1$. Higher values of alpha imply a greater density of randomly generated events or noise.

We can inject *fixed sequences* of semi-interval events in the data set. The frequency of fixed sequence generation is controlled by parameter β where $0 < \beta < 1$. Whenever random event generation is triggered (determined by α), the algorithm will generate one of the fixed sequences with a probability β. The fixed sequence to be generated is chosen randomly from a set of fixed sequences. In our experiments, injected two sequential patterns:

$$S_1 := E_1^+ \Delta_{25} E_2^+ \Delta_{10} E_2^- \Delta_{15} E_1^- \Delta_5 E_5^+ \Delta_{20} E_5^-$$

$$S_2 := E_3^+ \Delta_{26} E_1^+ \Delta_{15} E_1^- \Delta_{10} E_3^- \Delta_{19} E_4^+ \Delta_{10} E_7^+ \Delta_{15} E_4^- \Delta_{18} E_7^- \Delta_{10} E_5^+ \Delta_{70} E_5^-$$

We varied the noise in the data using the parameter α and compared the number of pairwise patterns and SISPs discovered at different noise levels and for different values of episode duration. Values of β, n_e, N and δ are set to 0.1, 10, 100000 and 5 respectively. All time

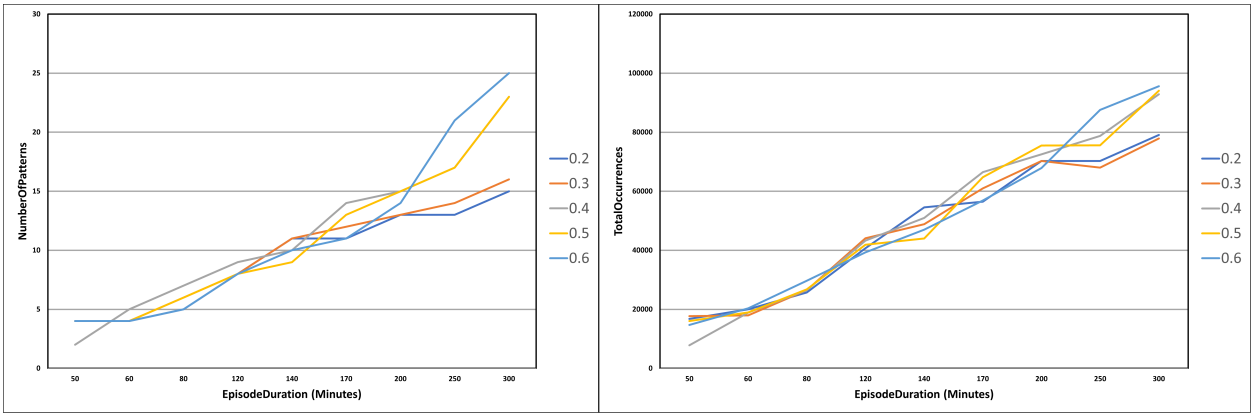

Figure 6.5: Distribution of number of pairwise frequent event patterns discovered and their occurrences versus induced error rates and episode duration in the synthetic data.

intervals and delays are measured in minutes.

6.4.2 Results and Discussion

Figure 6.5 shows the number of distinct pairwise patterns and their aggregated occurrences for different values of episode duration and at different error rates (α). Every line in the plots represents the number of patterns and pattern occurrences at different error rates. We can see in the figure that for smaller values of the episode duration, the error rate does not significantly impact the number of patterns and pattern occurrences. This shows that for smaller episode duration, the significant pairwise patterns are extracted with high precision. However, for larger episode duration ($>$ 170 minutes), greater error rates lead to a greater number of spurious (non-injected) patterns and consequently a larger number of pattern occurrences. Based on this insight, we expect the number of spurious SISPs to increase with increasing episode duration, though it is necessary to increase the episode duration for extracting longer patterns.

Figure 6.6 depicts the occurrences of patterns of different length for different episode duration. We observe occurrences of larger patterns only for longer episodes. Especially, patterns originating from S_2 are only observed for episode duration $>$ 100 minutes. This matches our

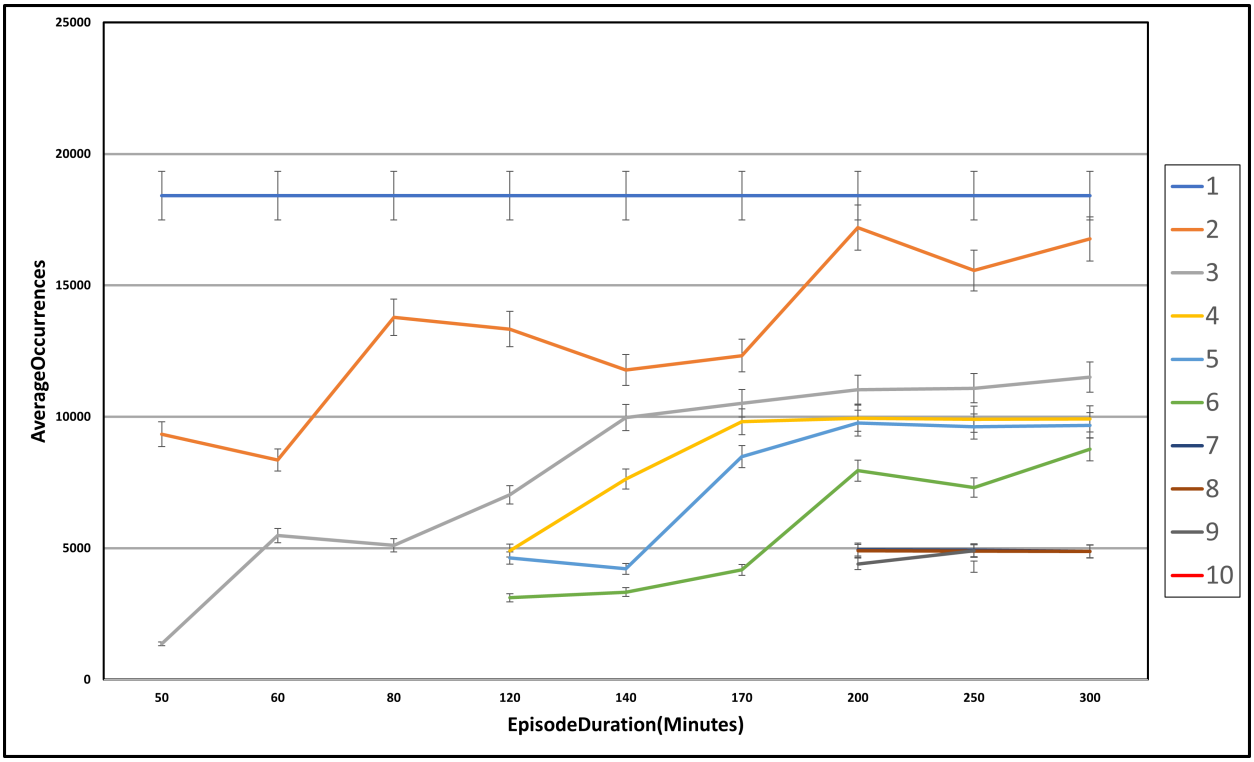

Figure 6.6: Occurrences of patterns of different lengths (displayed in legend) for varying episode duration.

expectation as injected temporal delays in S_2 are relatively larger than in S_1. We can see an example of extracted patterns for support threshold $Min.Support = 0.05$ and episode duration 120 and 250 minutes in tables 6.1 and 6.2 respectively. As expected, the longer episode duration allows us to capture longer patterns but, at the same time, also reduces the precision of the extracted patterns.

Pattern	Significant	Occurrences
E_5^-	True	16776
$E_5^+ \Delta_{[18,76]} E_5^-$	True	8202
$E_1^- \Delta_{[2,10]} E_5^+ \Delta_{[18,76]} E_5^-$	True	4026
$E_2^- \Delta_{[13,20]} E_1^- \Delta_{[2,10]} E_5^+ \Delta_{[18,76]} E_5^-$	True	4000
$E_2^+ \Delta_{[8,15]} E_2^- \Delta_{[13,20]} E_1^- \Delta_{[2,10]} E_5^+ \Delta_{[18,76]} E_5^-$	True	3844
$E_1^+ \Delta_{[23,30]} E_2^+ \Delta_{[8,15]} E_2^- \Delta_{[13,20]} E_1^- \Delta_{[2,10]} E_5^+ \Delta_{[18,76]} E_5^-$	True	3164

Table 6.1: SISPs extracted for episode duration = 120 minutes, $\alpha = 0.4$ and $\beta = 0.1$.

Pattern	Significant	Occurrences
E_5^-	True	16776
$E_5^+ \Delta_{[18,76]} E_5^-$	True	8106
$E_6^+ \Delta_{[0,33]} E_5^-$	False	922
$E_0^+ \Delta_{[0,68]} E_5^-$	False	1294
$E_1^+ \Delta_{[0,48]} E_5^-$	False	1808
$E_9^+ \Delta_{[0,91]} E_5^-$	False	1071
$E_8^+ \Delta_{[0,29]} E_5^-$	False	973
$E_2^- \Delta_{[0,50]} E_5^-$	False	1070
$E_2^- \Delta_{[0,50]} E_5^-$	False	1070
$E_1^- \Delta_{[5,30]} E_5^+ \Delta_{[18,76]} E_5^-$	True	4037
$E_7^- \Delta_{[6,54]} E_5^+ \Delta_{[18,76]} E_5^-$	True	4012
$E_2^+ \Delta_{[1,19]} E_2^- \Delta_{[0,50]} E_5^-$	False	959
$E_2^- \Delta_{[13,20]} E_1^- \Delta_{[5,30]} E_5^+ \Delta_{[18,76]} E_5^-$	True	3832
$E_4^- \Delta_{[16,23]} E_7^- \Delta_{[6,54]} E_5^+ \Delta_{[18,76]} E_5^-$	True	3999
$E_2^+ \Delta_{[8,15]} E_2^- \Delta_{[13,20]} E_1^- \Delta_{[5,30]} E_5^+ \Delta_{[18,76]} E_5^-$	True	3443
$E_7^+ \Delta_{[13,20]} E_4^- \Delta_{[16,23]} E_7^- \Delta_{[6,54]} E_5^+ \Delta_{[18,76]} E_5^-$	True	3994
$E_1^+ \Delta_{[23,30]} E_2^+ \Delta_{[8,15]} E_2^- \Delta_{[13,20]} E_1^- \Delta_{[5,30]} E_5^+ \Delta_{[18,76]} E_5^-$	True	2507
$E_4^+ \Delta_{[8,15]} E_7^+ \Delta_{[13,20]} E_4^- \Delta_{[16,23]} E_7^- \Delta_{[6,54]} E_5^+ \Delta_{[18,76]} E_5^-$	True	3985
$E_3^- \Delta_{[17,24]} E_4^+ \Delta_{[8,15]} E_7^+ \Delta_{[13,20]} E_4^- \Delta_{[16,23]} E_7^- \Delta_{[6,54]} E_5^+ \Delta_{[18,76]} E_5^-$	True	3970
$E_3^+ \Delta_{[24,31]} E_1^+ \Delta_{[13,20]} E_1^- \Delta_{[8,15]} E_3^- \Delta_{[17,24]} E_4^+ \Delta_{[8,15]} E_7^+ \Delta_{[13,20]} E_4^- \Delta_{[16,23]} E_7^- \Delta_{[6,54]} E_5^+ \Delta_{[18,76]} E_5^-$	True	3339

Table 6.2: SISPs extracted for episode duration = 250 minutes, $\alpha = 0.4$ and $\beta = 0.1$.

Chapter 7

Personal Models: Exploration and Examples in Personal Health

This chapter will discuss salient traits of personal health, behavioral models derived using the proposed N-of-1 approach, and how these can be utilized in a cybernetic framework to provide continuous health navigation. We have demonstrated the efficacy of our proposed N-of-1 analysis approach in various publications studying the effects of different lifestyle and environmental factors on aspects of individual health and behavior such as sleep quality, cardio-respiratory fitness, and endurance activity performance. We also demonstrate how we can utilize contextual information to create a model of users' preferences in different situations and provide context-aware food recommendations.

7.1 Personalized Health Models

Our health is the sum of our genetics, lifestyle, and environmental exposures. Different activities and experiences continuously impact different aspects of our health, but we rarely

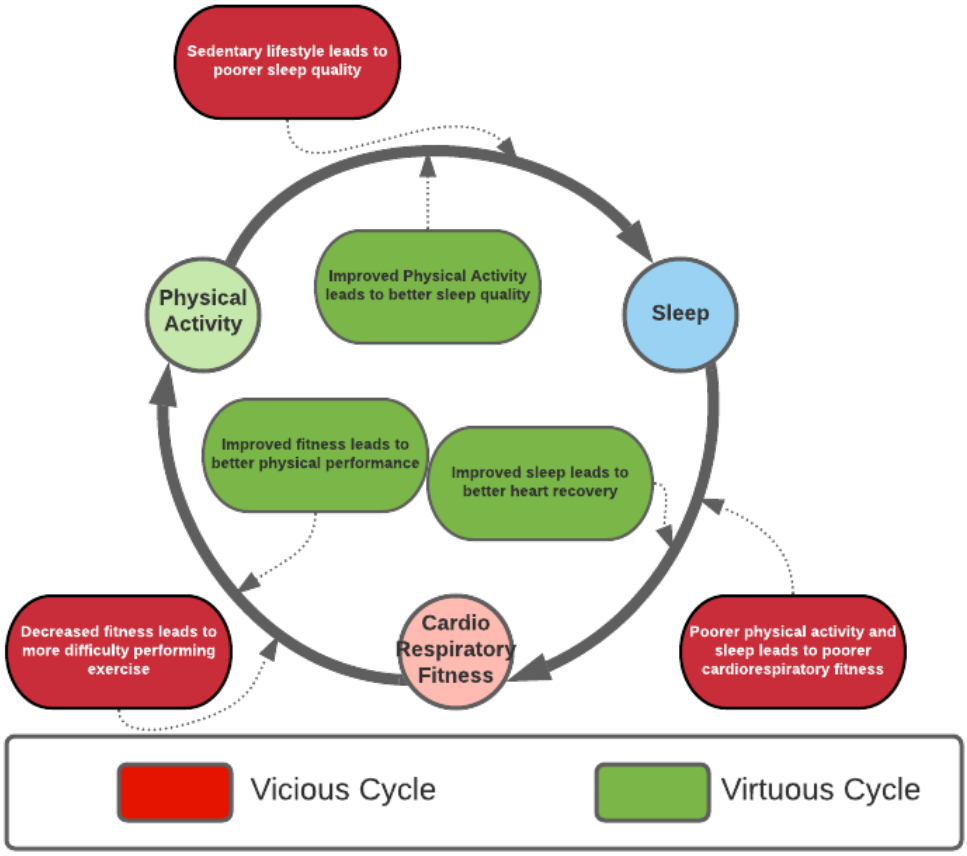

Figure 7.1: Our lifestyle events have a large impact on our health. Our habits can either lead us to a *virtuous cycle*, where positive health results are a natural outcome of our habits and in turn help sustain those habits, or a *vicious cycle*, where our lifestyle leads to a decline in health which in turn makes sustaining healthy habits even more difficult.

recognize such changes unless we start showing symptoms of a disease. The symptomatic stage indicates significant disease progression, and it might already be too late for an effective intervention (especially for chronic diseases such as type-II diabetes and hypertension). Thus, we need to monitor our health continuously in order to manage it effectively over time.

Continuous monitoring and estimation of health is a difficult task for the existing healthcare paradigm as the conventional methods of observing health parameters (e.g., blood tests and X-rays) require large apparatus and significant expertise. On the other hand, the widespread adoption of wearable and smart devices has empowered individuals to collect rich multimodal data about their lifestyles and health. This data can be utilized to identify different lifestyle

activities and events[135] as well as different health parameters[128]. Our lives are segmented into various events such as sleeping, eating, and working, and such events provide us a way to reason with lifestyle and environmental information. These events affect our lives, and health [138], and repeated behaviors can lead us to vicious or virtuous health cycles (fig. 7.1). Therefore, events are an inherently explainable and customizable abstraction while moving from data streams to personal models, and any intervention designed in the form of lifestyle events can be easily translated to the required user action. To leverage this data effectively, we need to develop personal health models that capture the interplay between these events and our health and can be used in a cybernetic framework to identify appropriate interventions for achieving individual health goals [124].

Such a model can be used in a cybernetic health navigation framework to provide the proper guidance at the right time for health management. Health recommendation systems provide us a way to apply cybernetic principles to manage a person's health [127]. Using lifestyle interventions, we can build a navigation system that guides us through our day much in the same way that modern navigation systems inform drivers about the most optimum path towards their destination [125]. We need to design context-aware personal recommendation systems that change the person's context with every event during the day. The dynamic context allows us to provide an optimal recommendation at every point of the day and calibrate the recommendations as different events occur.

We have demonstrated the utility of the proposed N-of-1 modeling approach in various studies investigating different aspects of a person's health. We have leveraged our approach to derive rule-based models of the person's health and demonstrated how it could be used to provide context-aware recommendations to improve health outcomes such as sleep quality. A rule-based representation of the verified hypotheses results provides an easy-to-understand model of the person's health. Since the quantities defined in the rule-based models are derived from user-generated events, the recommendations generated by the models can be

easily converted to real-life user events.

7.2 Case Study I: Continuous Health Interface Event Retrieval

Wearable devices and lifelogging applications capture a large amount of data about events in our lives; however, the data typically stay in their silos. Combining all such events and data streams is necessary to enable multimodal applications to perform effective health estimation and guidance. We may want to combine events or data streams from different sources to identify more complex events with a deterministic relationship with health parameters. Such events act as an interface between lifestyle events and biological systems; hence we are calling them **interface events**. These events determine the mechanism through which lifestyle events impact our biology, and variations in these interface events could lead to entirely different outcomes for similar lifestyle events.

This can be observed when considering the long-term cardiac effects of different types of physical activities. Physical activities can strain our heart muscles in two ways, cardiovascular volume overload and pressure overload. A cardiovascular volume overload is a biological event where the heart must pump large volumes of blood through the circulatory system. This event would cause structural changes in the heart in the form of eccentric hypertrophy, where the left ventricular chamber volume increases. The volume overload interface event could be caused by various exercise events such as cycling, jogging, soccer, dance, or hiking. Other exercise events such as weight lifting, sprinting, bouldering, or even bowel constipation would cause pressure overload in the heart. These events would cause concentric hypertrophy of the heart[113][27]. This is illustrated in Figure 7.2. In this case, we see that similar lifestyle events can have a vastly different impact on our heart, and it is important that we find the cardiovascular volume and pressure overload events if we want to find the impact of any exercise on our heart.

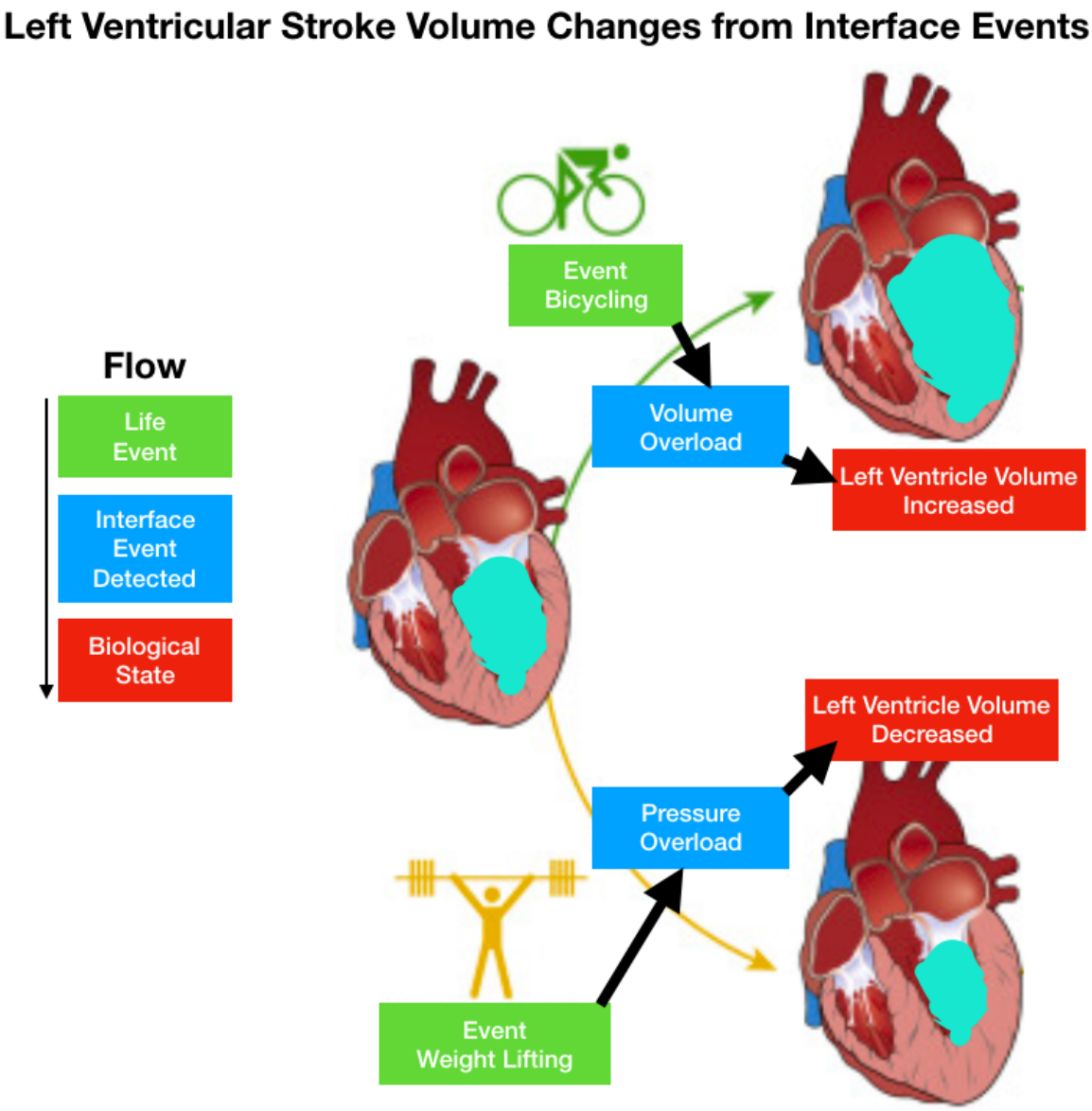

Figure 7.2: This figure shows that detecting relevant interface events is necessary to estimate the physiological state of an individual from their lifelog information. Here two examples of life events are given that cause opposite outcomes in cardiac adaptation. Both events at a crude level can be considered exercise, but precisely extracting the interface event is critical if we are to use the lifelog information to estimate evolving health states.

7.2.1 Knowledge driven event extraction

A lot of the interface events have been recognized in the medical literature. For example, PM2.5 exposure and circadian rhythm disruption are caused by different lifestyle events such as outdoor exercises, changes in sleeping and eating habits, and screen activity and impact the functioning of various biological systems. Similarly, the literature has well-established links between diet and various chronic diseases such as type II diabetes, hypertension, and ASCVD. Thus, we must utilize the existing knowledge to guide the retrieval of these interface events from user-generated data.

The interface events can be specified as transformations and combinations of existing events and data streams. These transformations can be as varied as applying a threshold on a data stream (e.g., binning heart rate in different ranges) to recognizing events and actions from a video stream (e.g., fall detection). We currently use the event patterns language and operators described in chapter 4 to describe the interface events. Especially NOT ($\neg$), AND($\wedge$) and OR($\vee$) operators, which can be used in conjunction with user-defined event detection operators to describe complex events. For example, if we want to identify events where the individual's heart rate is above 120 bpm, and PM2.5 concentration is above 10 $\mu g/m^3$ of air, then we would need first to create a PM2.5 stream using their location stream. We could then define these exposure events using the formulation given below. Figure 7.3 shows some of the retrieved events.

$$ExposureEvent := (Heartrate > 120) \wedge (PM2.5 > 10)$$

These event operators can be combined with user-defined event recognition operators allowing us to utilize more data streams and recognize more complicated events. For example, we define a climb-detection operator for the altitude data stream. This operator detects time intervals when the user is climbing up a slope. The detected events could be combined with cycling events and events where the person's heart rate is above 170 bpm to find out

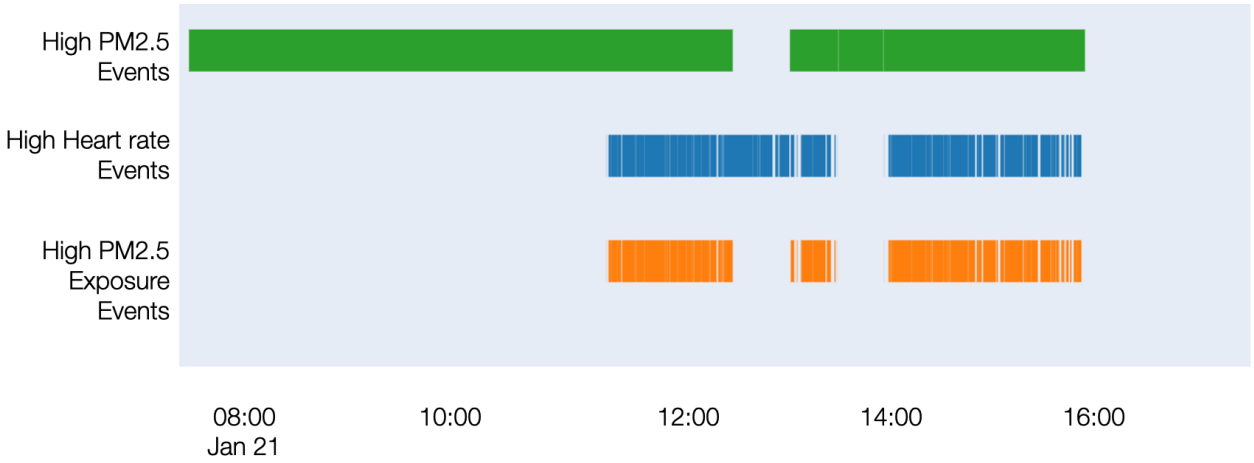

Figure 7.3: This figure depicts the instances during a day where the individual's PM2.5 intake is expected to be high. This is determined by combining heart rate zone events (HR $\geq$ 120 bpm) with High PM2.5 concentration events, which in turn is determined using location stream and air pollution data.

high-intensity uphill cycling events, which cause cardiovascular volume overload. The event formulation is shown below, and some of the retrieved events are shown in figure 7.4.

$$UphillCycle := Cycling \wedge detect - climb(Altitude)$$

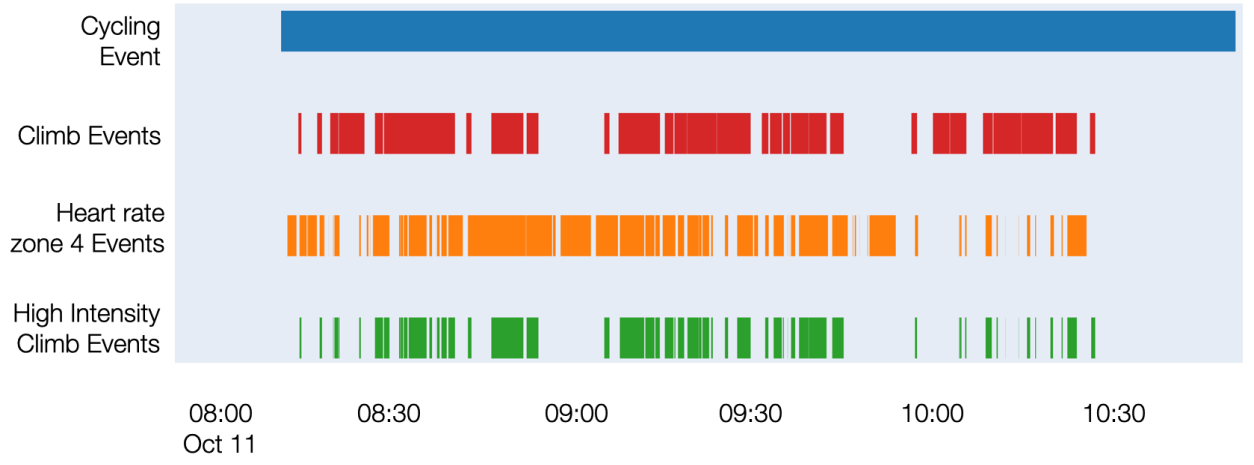

Figure 7.4: This figure shows the application of AND operator to combine events. We are trying to detect interface events where the heart rate of the individual is in zone 4 (170-190 bpm) while they are climbing a slope on a bicycle.

7.2.2 Methodology

We obtain the definition for different such events from medical literature and identify lifestyle events that could be utilized to retrieve those. For example, exposure to different pollutants is a vital interface event that impacts different aspects of a person's health. We can calculate the pollutant exposure from the user's continuous GPS location stream and combine it with the EPA's publicly available pollutant information.
Once the interface events have been defined, we can use the definition to retrieve those events and then utilize them for further estimation or guidance.

7.2.3 Dataset

We have used lifestyle and physiological data streams collected by an individual over ten years for our experiments. Different data streams have been collected over different periods and vary in sampling frequency. We expect our system to be used for experiments in an N-of-1 setting [90] for providing individualized health guidance using observational data; thus, the evaluation of this system should be done for an individual over time.
There are two primary sources of personal data that we have utilized in our experiments:

- Detailed physiological data generated during an exercise event sampled at per second frequency. These data streams and events are collected from Strava and can better understand the user's health state.
- Lifestyle data collected as events or data streams during the day. These are typically sampled at a lower frequency. These data streams and events are collected from various sources such as Apple HealthKit and Google timeline etc. These capture the lifestyle events we expect to encounter during our day-to-day lives, such as sleep, meal times, and commute.

Table 7.1: Data streams and sources

Data stream	Sources
Heart rate	Strava, Apple Health-Kit
Power	Strava
Cadence	Strava
Altitude	Strava
Location	Strava, Google Location History
Step Count	Apple Health-Kit
Weight	Apple Health-Kit
Stairs	Apple Health-Kit

These data streams and corresponding sources are listed in table 7.1.

We have also used the pollution data captured by various environmental agencies across the world and aggregated by EPA[1]. We find the monitoring station nearest to the user's location and use the file generated by the station at the given time to estimate the user's exposure to different pollutants.

7.2.4 Interface Events

We have defined interface events derived from biomedical literature using the previously mentioned event operators and specific event-detection operators for data streams. We are retrieving two lifestyle-based interface events and two environmental interface events, which are described below.

- **Lifestyle interface events** *Volume Overload* events require an individual's heart to put a sustained effort. These can be identified from the heart rate stream (by finding intervals of high heart rate) or by combining altitude stream (to detect climb events) and cycling (or running or hiking) events. If these events are repeated over time, they are likely to result in an increase in left ventricle volume (as depicted in fig. 7.2). These

[1] `https://aqs.epa.gov/aqsweb/documents/data_api.html`

events can be represented as

$$VolOverload := (HR > 140) \vee (Cycling \wedge detect - climb(Altitude))$$

Pressure Overload events require the heart to put in a short and intense effort, which presents as a spike in the heart rate stream. These can also be determined by using the power output stream collected during cycling events and identifying intervals where power output is higher than 400W. Over time, these events lead to a reduction in left ventricular volume. These events can be represented as

$$PressOverload := detect - spike(HR) \vee (Power > 400W)$$

- **Environmental interface events** An individual's exposome streams can be created from their location history combined with publicly available GIS data provided by different government organizations. *PM2.5 intake* can be computed from air pollutant data (provided by EPA) and the heart rate stream (used to estimate the breathing rate using results from [195]). We are estimating individual tidal volume using formulation given in [28]. Multiplying the estimated tidal volume and breathing rate gives us the total air intake per minute for the individual, using which we can find their PM2.5 intake. Prolonged PM2.5 exposure has many long-term and short-term health effects[206].

  Blood O_2 level of individuals is known to decrease with altitude as air pressure decreases with an increase in altitude. It leads to reduced blood oxygen saturation levels, which can be quantified by using the relationship described in [51]. Low blood oxygen levels can lead to hypoxia, which can affect various biological functions [50].

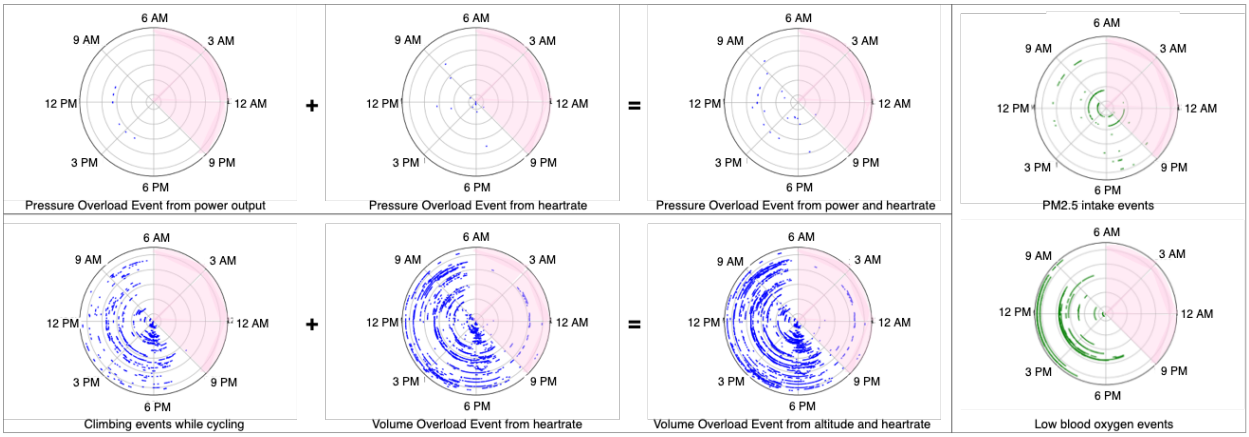

Figure 7.5: This figure shows different interface events retrieved using user-generated data. Different days of the year are represented as concentric circles, and different sectors of the circle represent different times of the day. Events in a day are represented as colored arcs on that circle. Volume and pressure overload events are retrieved using lifestyle data and events such as heart rate and exercise events. We can see that we can retrieve a significantly larger number of interface events by combining events from multiple data streams. This figure also shows the environmental interface events. High PM2.5 intake events are recognized over one week. These are the instances where per minute PM2.5 intake was higher than 0.7 μg. Low blood oxygen events are recognized over one year, where blood oxygen saturation goes below 95%. The highlighted sectors roughly represent the time between sunrise and sunset; thus, we can see how different events overlap with circadian patterns.

7.2.5 Results

RQ1: Data fusion

We can use multiple data streams from different sources to recognize different instances of an interface event. This is evident in fig. 7.5, which depicts the occurrences of cardiovascular volume and pressure overload events. Volume overload events are detected in two ways, 1) We identify intervals with a sufficiently high heart rate, and 2) We identify intervals where the person is climbing up a slope under their power (e.g., cycling). These two event definitions are combined using an OR operator to give all occurrences of volume overload events. We could add more methods of recognizing volume overload events using the same operator with minimal additional effort. Similarly, pressure overload events are recognized from the heart rate data stream (spike detection) and the power stream (high effort intervals). These events

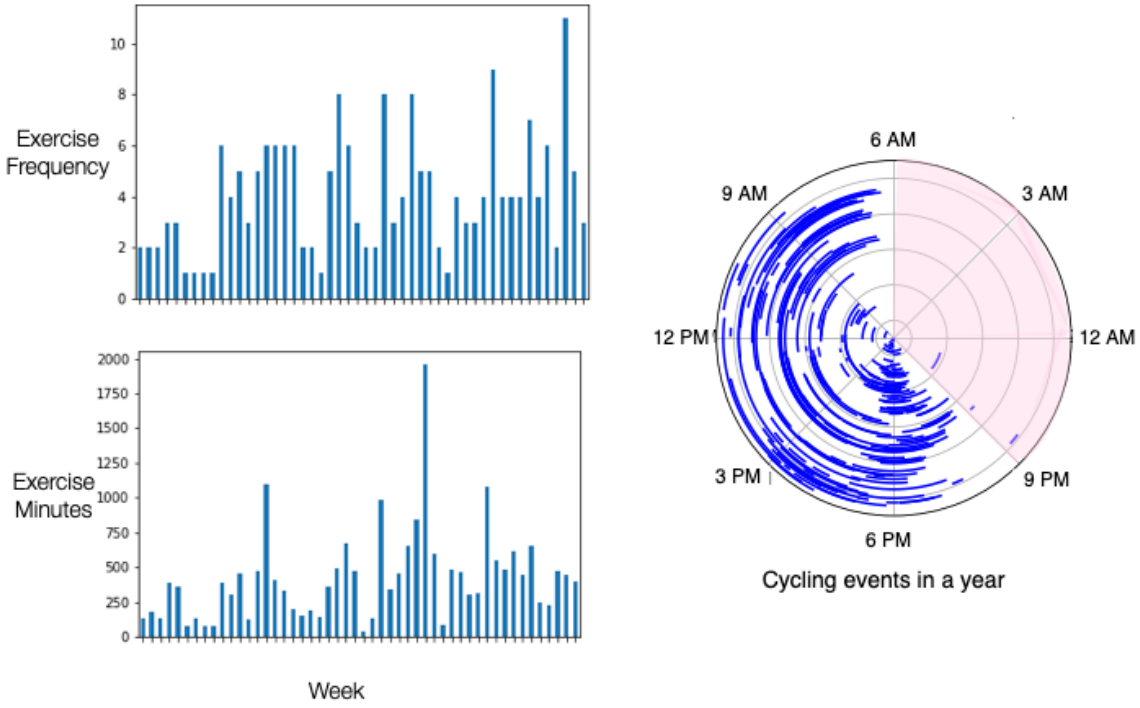

Figure 7.6: This figure shows the results of a real-world query about exercise behavior. The bar plots show the exercise frequency and exercise minutes per week over a year. The polar plot shows all the exercise events during the day, where the highlighted sector roughly matches the time between sunset and sunrise. Thus we can see how likely are these events to cause any disruptions in circadian patterns.

are, by definition, short, and the same can also be seen in fig 7.5. We can also combine the user-generated data with environmental data to generate their exposome streams. Figure 7.5 shows the occurrences of PM2.5 intake events over a week and low blood oxygen events over a year.

RQ2: Continuous Retrieval

We are retrieving the events continuously over a year, as shown in figure 7.7. A zoomed-in view is shown in figures 7.4 and 7.3 which highlight the combinations of events while retrieving an interface event. We apply event detection operators on incoming data streams and use the definition of the interface events to combine the generated events. Once we have

retrieved the interface events over a sufficiently long period, we can analyze their occurrences and find useful patterns, such as the relationship of volume overload events with circadian patterns. We can see in fig. 7.5, the density of the volume overload events is very low after sunset, which may lead us to think that any disruptions in sleep patterns are unlikely to be caused by these events. We can overlay a similar plot of sleep events and see the relationship between the event streams.

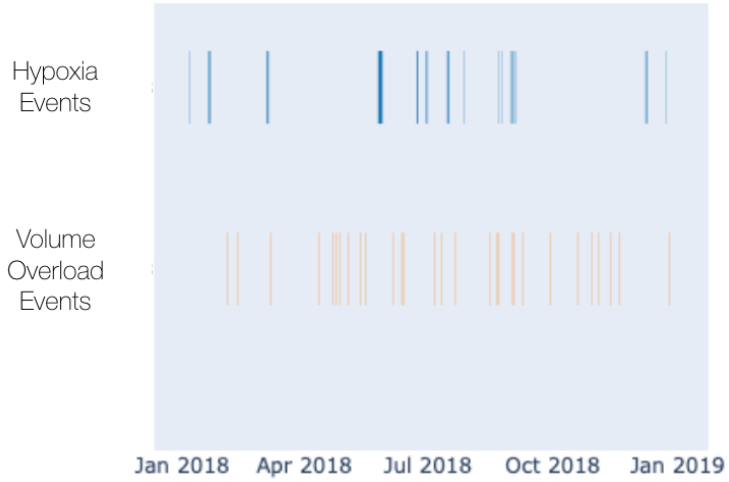

Figure 7.7: This figure shows the continuous retrieval of volume overload and hypoxia events over the course of a year.

RQ3: Real world query

The system is also helpful for answering a real-world query about a person's life in terms of attributes and patterns of relevant interface events. One such example is in fig. 7.6 where we try to respond to a physician's questions about a person's exercise habits. Typically a doctor would ask the person about the frequency and intensity of their exercise routine. These answers are in figure 7.6, where we display the exercise frequency and total duration of exercise events for every week of the year. We also show all the exercise (cycling) events in a format that can be compared with any other lifestyle habit (e.g., sleep) to discover new behavioral/physiological patterns.

7.3 Case Study II: Personalized models for understanding sleep behavior

Lifestyle factors have a high impact on our health outcomes. Our eating, sleeping, and movement patterns determine large parts of our short and long-term health [174, 54, 128, 123]. Keeping track of how we behave in different contextual situations and the impact these behavioral patterns have on our health is difficult for medical professionals and individuals. At the same time, with the increasing prevalence of chronic diseases such as diabetes and hypertension, understanding the effect of lifestyle on different aspects of our health becomes a critical research challenge [87, 168]. A multitude of consumer devices such as smartwatches and smart home systems measure aspects of our daily life as events and data streams and control our local environment [19]. Using the data streams and events captured by these devices, we can find recurring behavioral patterns and associated health outcomes to create an explainable rule-based model of the person [137]. Explainability is a desired quality in health prediction and recommendation systems as it can verify the quality of predictions and builds user engagement and trust in the system.

The field of sleep and health recommendation systems is relatively new. Studies have explored the pitfalls of using the conventional recommendation systems for health and devised alternatives using entity properties and relationships [95]. Context-awareness is an essential quality for health recommendation systems [170]. Context-aware recommendation systems (CARS) have been explored in different domains [188]. CARS have traditionally incorporated context information in collaborative filtering models in one of three ways, 1) Contextual Pre-filtering, 2) Contextual Post-filtering, and 3) Contextual Modelling [6]. There can be different types of contextual information relevant to a recommendation system. These usually fall into one of the following categories: temporal, location, individual (user characteristics), activity (about the activity), and relational (when multiple entities are involved)[188]. We have adopted a contextual modeling approach and incorporated the contextual information

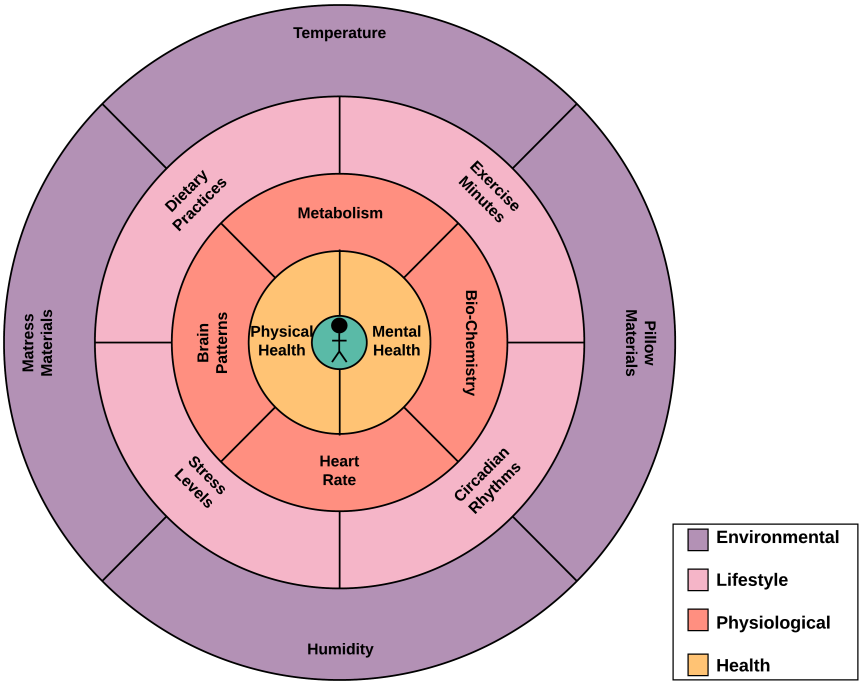

Figure 7.8: User Centered Semantic Rings with a focus on sleep. These rings show the most relevant factors that contribute to an individuals sleep state.

in the rule-based model itself. Multiple studies have explored personalized recommendation systems for different aspects of user-health, such as diet [76]. These utilize different learning techniques to develop personalized models for individuals but usually lack explainability, an essential characteristic of health recommendation systems.

7.3.1 Causal Rule-based modelling: Event Mining

Creating a model of the person's behavior and health is central to building personalized health recommendation systems. We apply the hypothesis testing approach described in chapter 5 to perform N-of-1 experiments on a user's data [108] that allows us to find causal relationships between different lifestyle events and biological outcomes. These relationships define a rule-based explainable model of the user that can predict sleep outcomes in different contextual situations. The process is described in figure 7.9.

Event mining allows us to discover and specify patterns between different events in a person's

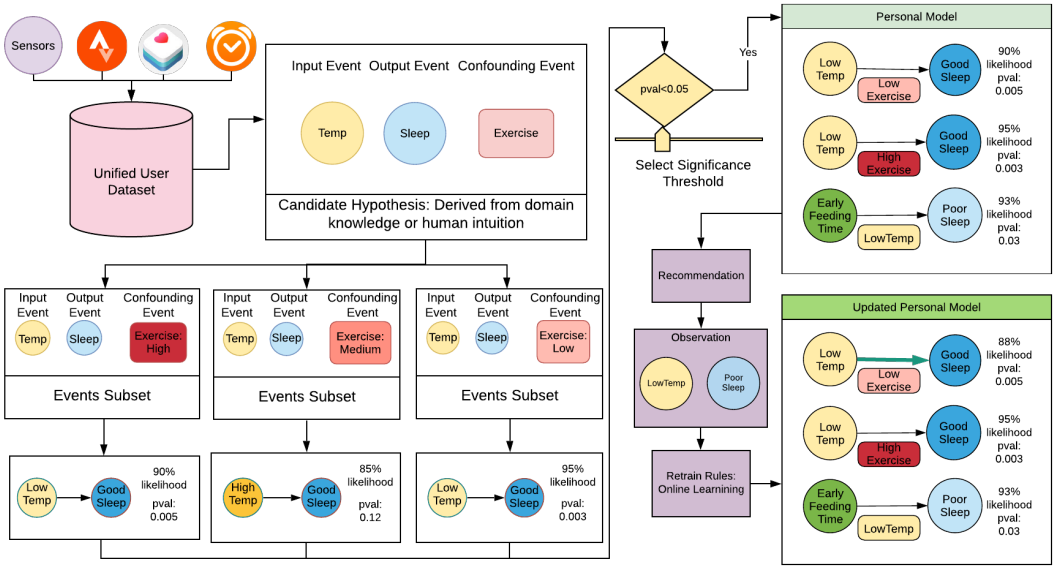

Figure 7.9: Rule based personal model for predicting sleep outcomes. We divide the occurrences of the outcome events into smaller subsets based on the values of co-occurring contextual factors. This minimizes the variance in the outcome due to the confounding variables within each subset. Subsets that exhibit significantly different distribution for different values of input events are converted to rules and added to the model.

life. We utilize these patterns to create hypotheses that might describe a person's behavior. A hypothesis needs to specify the intervention event and the associated confounding factors that affect the relationship between the intervention and the outcome. The confounding factors are specified using the temporal delay operator, $\Delta[t_b, t_e]$, that relates the events that occur within the specified time interval $[t_b, t_e]$. The confounding factors are specified as patterns P between the lifestyle events and the outcome. Thus, a hypothesis would be specified as $E_i \underset{P}{\rightarrow} E_o$, where we want to measure the causal effect of intervention E_i on the outcome event E_o while controlling for the events specified by the set of patterns P. These patterns and hypotheses can be derived from existing knowledge and human intuition, allowing us to leverage the results of population studies performed in clinical and controlled settings.

Combining the event mining operators with causal inference principles allows us to perform N-of-1 experiments on the user's longitudinal data. Thus, we can find the effect of the

intervention on the outcome in an unbiased manner, and if we can capture all the confounding variables in the set of patterns, we would obtain the causal effect of the intervention on the outcome. The results are stored as a conditional rule that uses confounding variables and the intervention event as the predicate. The distribution of the outcome events in the subset is used to make a prediction.

We use a set of these contextual rules to predict health outcomes. We find the most relevant rule by matching the user's current context with the rules and utilizing it to make the prediction. A rule-based model, while lacking in complexity, offers the advantage of explainability and online training. Every prediction and recommendation generated from this model can be explained using the associated contextual factors, thus eliminating recommendations based on spurious relationships. This is an essential characteristic of health models and recommendation systems. As the user behavior changes over time, the model needs to adapt to the changing user parameters and be trained using the latest observed data. We can easily update the rules by updating the outcome variable's distribution whenever the rule matches the user's current context.

7.3.2 Multi-Item Health Recommendations

The rule-based health model allows us to find the health outcomes in different contextual situations. The user's activities during the day (such as exercise, meals, work-related stress) and their local environmental parameters (such as temperature, humidity, and ambient light and sounds) determine these contextual variables. Thus, we can utilize this model in a recommendation system setting to determine the parameters (both user behavior and environmental variables) for optimizing a health outcome (e.g., sleep quality).

Every action taken by the user and every environmental exposure changes the user's health state [123], which changes the context for future actions and recommendations. We need to

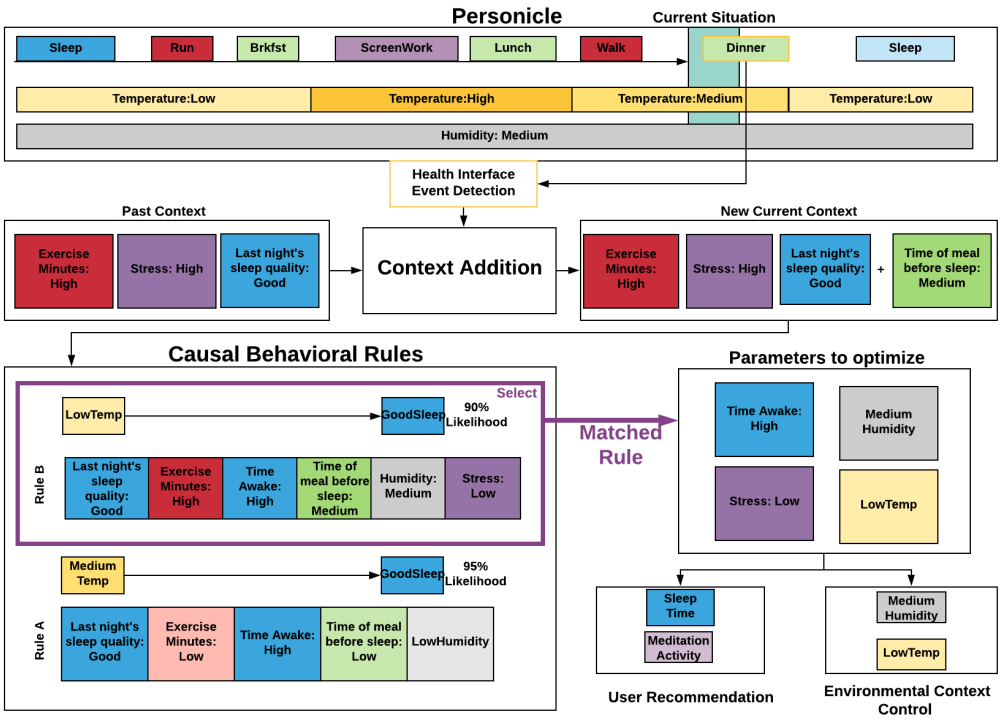

Figure 7.10: Live context calculation. The system updates user context every time they log an event. We retrieve all the sub-events and parameters relevant for context calculation (e.g., time of meal from dinner event). The retrieved contextual information is added to the existing context, and the updated context is used to generate a new set of recommendations. These recommendations are then sent either to the user or to a device controlling the user's environmental factors.

retrieve the relevant events from the user's events and data streams that impact their health state [138]. Different contextual parameters are defined as aggregations of these events. For example, Total Screen Time during the day is an important confounding factor for understanding an individual's sleeping habits. It can be determined by aggregating the duration of all the screen activity events (such as working, watching TV, and social media activity) during the day. These aggregations can be performed using events-based triggers encoded as condition-action rules. As new events appear in the person's events log, the retrieved events can be aggregated to change the user's live context parameters. We can use the latest context values to provide recommendations that would optimize the user's health outcomes. We match the live contextual parameters for the person with the contextual parameters

of the various rules present in the model. If the current context matches multiple rules, we utilize the rule with the highest likelihood of the desired outcome. Once we have identified the rules that match the current context, we can utilize the unmatched contextual parameters and the intervention event to find the parameters that can lead to the optimal outcome. We can either present the recommendation to the user (if the recommendation is an action to be taken by the user) or communicate with a smart device that controls the user's environmental context (e.g., smart home devices, HVAC systems, smart bulbs). The recommendation system produces a set of actions that would maximize the likelihood of the optimal outcome; thus, the proposed recommendation system is different from typical recommendation systems as the recommendation consists of multiple items.

Since any event during the day can change the user's context, the recommendations are recomputed anytime an event changes the user's context. This process is depicted in figure 7.10. Thus, at any point during the day, the recommendation system would provide a list of timestamped actions to be performed by different agents (the user or an automated device) to optimize the sleep outcomes.

7.3.3 Methodology

We ran experiments to create a personal rule-based model for optimizing a person's sleep quality metrics by providing lifestyle and local environmental recommendations. We utilized data collected by one individual for more than two years using readily available consumer applications and wearable and IoT devices. We performed two sets of experiments on the collected dataset to create and evaluate the model. The first set of experiments find the average causal effect of input variables on sleep quality metrics. We used Welch's t-tests and a p-value of 0.05 to determine statistical significance. The second set of experiments tested the prediction accuracy for a static pre-trained model vs. an online training model.

7.3.4 Data Set

The data set includes exercise and lifestyle parameters for a 31-year-old male collected continuously over two years via the user's Garmin Fenix 5 smartwatch, their smartphone, and an IoT sensor that collected local temperature and humidity values. Sleep Cycle was primarily used to keep track of sleep events. Apple Health Kit was used to help compile sleep quality measures recorded by the Garmin smartwatch, daily step counts, and daily floors climbed. The accelerometer measures of the smartwatch and the audio recordings from SleepCycle were used to create sleep quality measures. Strava was used to keep track of exercise events. An image-based food log recorded feeding times with phone camera metadata, and a SensorPush IoT sensor was used to collect temperature and humidity during sleep events. All of these data sources were then temporally matched to record lifestyle events throughout the day accurately.

We used the thresholds mentioned in Table 7.2 and Table 7.3 to convert the data streams to relevant events for the event mining analysis. We used nine lifestyle/environmental events: Previous Night's Sleep Quality Measures, Exercise Minutes in the Day, Interval Between Eating and Sleeping, Minutes Awake Between Sleep Events, Temperature, and Humidity when going to bed. The possible output events are sleep quality measures (Table 7.2). We used 70% of the data to build the model and 30% of the data to test the model. The train-test split was created based on temporal order.

7.3.5 Causal Rules and Effects from N-of-1 Experiments

We perform N-of-1 tests on the user's data to find the average effects of different lifestyle and environmental events on sleep quality parameters while controlling for other lifestyle parameters as confounding factors. We treat one of the input event's possible values as the baseline and compare the distribution of the outcome variable for other values of the

Table 7.2: Sleep Quality Measure and Event Thresholds

Variable	**Classification Ranges**	**Event Name**
Sleep Latency	[0, 15] (15, 30] (30, ∞)	Good Average Poor
Awake Minutes	[0, 20] (20, ∞)	Good Poor
Awakenings >5 mins	[0, 1] (1, ∞)	Good Poor
Sleep Efficiency	[0.85, 1.00] [0, 0.85)	Good Poor

Table 7.3: Lifestyle Factors and Event Thresholds

Variables	**Classifications Ranges**	**Event Name**
Exercise Minutes Per Day	[0] (0, 50] (50, 150] (150, ∞)	None Poor Average Good
Exercise Minutes Per Week	[0, 150] (150, 300] (300, ∞)	Poor Average Good
Interval Between Eating and Sleeping	[0] (0, 180] (180, ∞)	Missing Poor Good
Minutes Awake Between Sleep Events	[0, 900] (900, 1020] (1020, ∞)	LT 15 Hours Btwn 15-17 Hours GT 17 Hours
Starting Temperature	[0, 60] (60, 67] (67, ∞)	Cold Comfortable Warm
Starting Humidity	[0, 30] (30, 50] (50, 100]	Low Ideal High

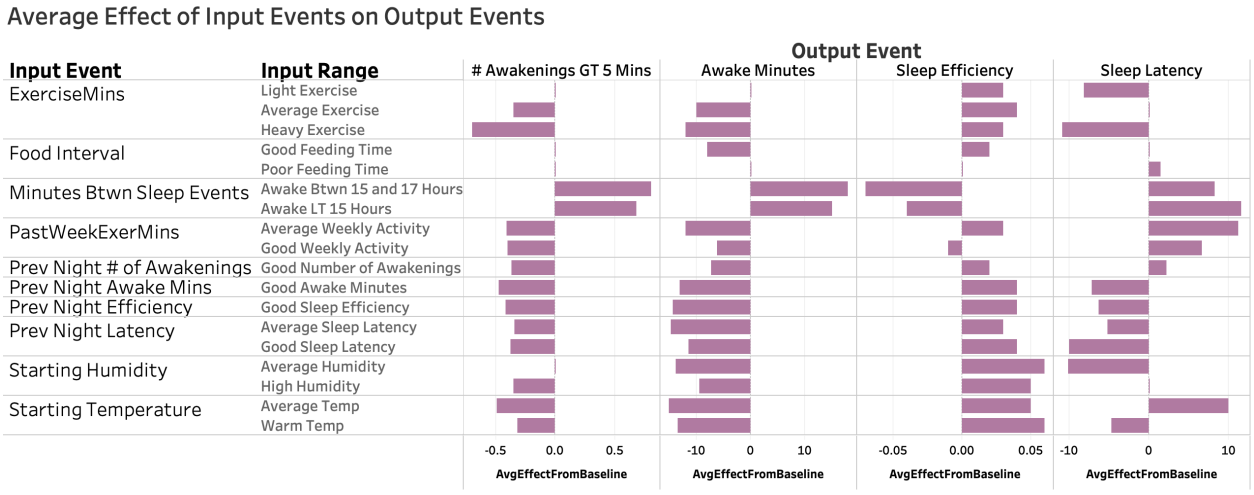

Figure 7.11: Average Effects that each input event has on the output event when compared to each input event's base category. If a metric is 0 then no significant relations were found.

event with the baseline distribution. If changing the input event value causes a significant change in the outcome distribution while controlling for confounding variables, the rule is deemed significant. This gives us the causal effect of different values of an input event on the observed outcome. We repeat this experiment while controlling for different variables and aggregating the causal effects to find the input event's average causal effect. If the difference is not significant, then we merge the two distributions and use the combined distribution at the time of contextual matching.

The results of these experiments are in Figure 7.11. Interestingly, an average temperature(60-67 F^o) seems to improve every sleep quality measure except for sleep latency. This is an important observation as it shows that not all quality measures are correlated with each other and that an improvement in one does not necessarily equate to an improvement in all other sleep quality measures. Another fascinating insight is that exercise improves sleep latency the most. On average, we can tell that exercising a lot will reduce sleep latency by 10.5 minutes, with just a small workout will help reduce sleep latency by an average of 8 minutes.

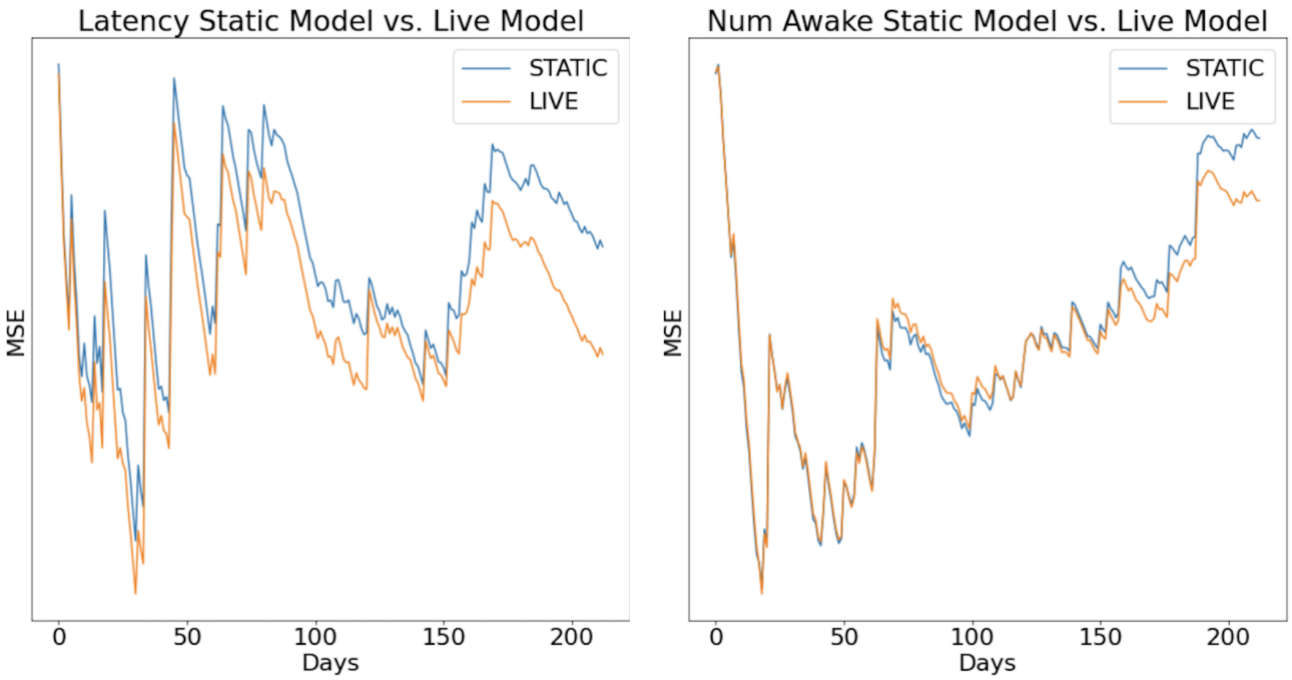

Figure 7.12: Comparison of pre-trained model vs. online training. Online training allows the model to adapt to user's changing sleep behavior resulting in lower error in predictions.

7.3.6 Context Matching and Sleep Predictions

We also want to demonstrate the contextual matching of rules and test the accuracy of the model's predictions, as that will determine the efficacy of any recommendations we generate. We train a linear regression model corresponding to every rule in the model and use the data subset that matches the rule to train the model. This model is then used to predict sleep outcomes for situations matching with the rule.

We used two training strategies for the prediction model; 1) Pre-trained static models and 2) the warm start online training. We expect that the user's sleep behavior would change over time, and thus, an online learning strategy would eventually outperform the pre-trained model.

The model's input features are Exercise minutes during the day, Feeding Time, Time Awake, Humidity, and Temperature while going to bed. We match the user's context with the context of the rules, and the most significant rule that matches the context is used to provide the

recommendation.

We create a set of contextual variables for each day in the dataset at the end of the day. These values are then used to find a matching rule. If multiple matches are found, then we used the rule with a higher statistical significance. The linear model associated with the matched rule would then be used to predict the sleep outcome parameter. The critical difference between the pre-trained and online models is that the online model would be updated continuously using the data in the test set. This way, the online model has the opportunity to adapt to the user over time. The results of the model predictions are in figure 7.12. The results illustrate an improvement in the performance of the online model over the pre-trained model. Eventually, we expect the online model would achieve a much smaller MSE as it adapts to the changing sleep behavior exhibited by the user.

7.4 Case Study III: Optimizing training for endurance activity performance

The effect of training and lifestyle on cardio-respiratory fitness and future endurance activity performance has been studied extensively in exercise physiology literature. However, most studies are conducted for a cohort of users and do not account for variations in individual traits such as genetics, age, and past exercise behavior. Thus, the results and insights generated from them do not generalize well for many individuals.

This study examines the effects of different parameters related to training behavior and exercise patterns on future exercise performance for every individual in the data set. We apply the N-of-1 hypothesis testing approach discussed in chapter 5 to test the relationships derived from exercise physiology literature.

7.4.1 Dataset

We used the Goldencheetah[2] exercise data set [105] for our experiments. The data set is publicly shared on the Open Science Framework and includes different exercise activities marked as interval events and various multimodal data streams that capture performance such as heart rate, power output, distance, and speed. The data set was curated by contributions from users of a popular sports analytics tool, *GoldenCheetah*, and contains rich longitudinal data about activity behavior and performance for a large number of people. The complete data set has more than 4000 participants and is updated frequently. We included the top-five participants in our analysis based on the heart rate and power data available. We applied the event and pattern operations on this data (as described in the previous sections) to derive different exercise performance, training load, and fitness features.

The progression of some of these features is shown in figure 7.13. The selected users have

[2] `https://osf.io/6hfpz/`

collected and volunteered exercise data over more than eight years and provide an excellent opportunity for longitudinal modeling of exercise behavior and performance using observational data.

7.4.2 Experiment

We applied event mining principles and hypothesis testing operations to find causal effects of different training parameters (such as training stress and duration) on endurance training performance and cardio-respiratory fitness.
We computed four types of parameters from this data:

1. **Exercise Performance** is measured by average power generated in bins of heart rate recovery percentage (HRR%). HRR% is defined as

$$HRR\% = 100 * \frac{hr - HR_{rest}}{HR_{max} - HR_{rest}}$$

   We define ten bins of HRR% (0-10%, 10-20%, etc.) and find the average power generated concurrently in HRR% bins for every ride with associated power and heart rate data streams. Events that have the performance information as a parameter are collectively referred to as ES_{CRF} as these will be used to estimate the cardio-respiratory fitness.

2. **Training Load** is captured by the volume and intensity of endurance exercises. We have defined multiple parameters under this category such as Exercise Duration, Coggan's Training Stress Score [44], TRIMP[120], Time in heart rate zones (HRZones) such as warmup, cardio, maximal, etc.

3. **Cardio-Respiratory Fitness** over an interval of time is determined by considering the slope of the regression line in HRR% vs. Power data for rides that lie in the

specified period. This slope captures the expected amount of power generated for a unit increase in HRR% and is expected to increase if a person has improved their cardio-respiratory fitness.

4. **Resting duration** captures the average amount of rest the person gets over an interval of time. We are estimating this feature as the average amount of time between two exercise events.

All the features are defined and calculated using event pattern language and are described in table 7.4.

7.4.3 Hypothesis

Our hypothesis aims to find the factors that contribute the most to changes in performance (both positive and negative). The causal graph for the hypotheses is in figure 5.3.c). The hypothesis captures the effects of three major factors:

1. Impact of training factors on fitness and current performance
2. Impact of fitness on performance
3. Impact of resting duration on fitness and performance

Our goal is to find training interventions that lead to the most impact on performance outcomes.

We find the causal effect of a parameter by finding the difference between the distribution of the outcome under the intervention (defined by the *do*-operator) vs. a baseline intervention. This allows us to find the best interventions for the individual by using known causal relationships and deriving the quantitative effects from the observational data.

Pattern	Group	Features
$P = ES_{Exe.} \xrightarrow{0,42days} ES_{CRF}$	$G = P.groupBy(ES_{CRF})$	$Exe.Duration = G.apply(sum, E.duration)$ $TSS = G.apply(COGGAN - TSS)$ $TRIMP = G.apply(TRIMP - func)$ $HRZone = G.apply(HRZone - div)$ $Cal.Burned = G.apply(sum, calories)$ $Ele.Gain = G.apply(sum, elevationGained)$
$P = ES_{CRF'} \xrightarrow{0,42days} ES_{CRF}$	$G = P.groupBy(ES_{CRF})$	$Fitness = G.apply(fitnessSlope)$
$P = ES_{Rest.} \xrightarrow{0,42days} ES_{CRF}$	$G = P.groupBy(ES_{CRF})$	$Avg.Rest.Duration = G.apply(average, duration)$

Table 7.4: Event patterns and group representation of features used in experiments.

7.4.4 Results and Conclusion

The results of our experiment are shown in fig. 7.14. The figure shows the five best and worst possible training interventions in terms of their impact on performance. We can see from the figure that the effects of interventions vary greatly for different individuals. For example, most effective interventions for users 1, 2, and 3 originate from training volume (total amount of training), while for user 5 the most effective intervention is increasing cardiovascular stress (TRIMP) during the training. This distinction makes sense because while user 5 does have significant training volume, the intensity (determined by heart rate) of their training is relatively low compared to other participants. The interventions with negative impact for user 5 also support this observation. We observed that spending more time in the 'fitness' heart rate zone (60-70% of maximal heart rate) lead to a negative impact on performance in HRR% zone 6.

These experiments demonstrate the utility of events while creating longitudinal personal models. Event and pattern operators provide a very convenient and inherently explainable mechanism to describe causal relationships and also allow us to encode known relationships in the personal model. However, the current methodology assumes that the temporal constraint that captures causal relationships between events does not change with time. This is not necessarily true and needs to be investigated further.

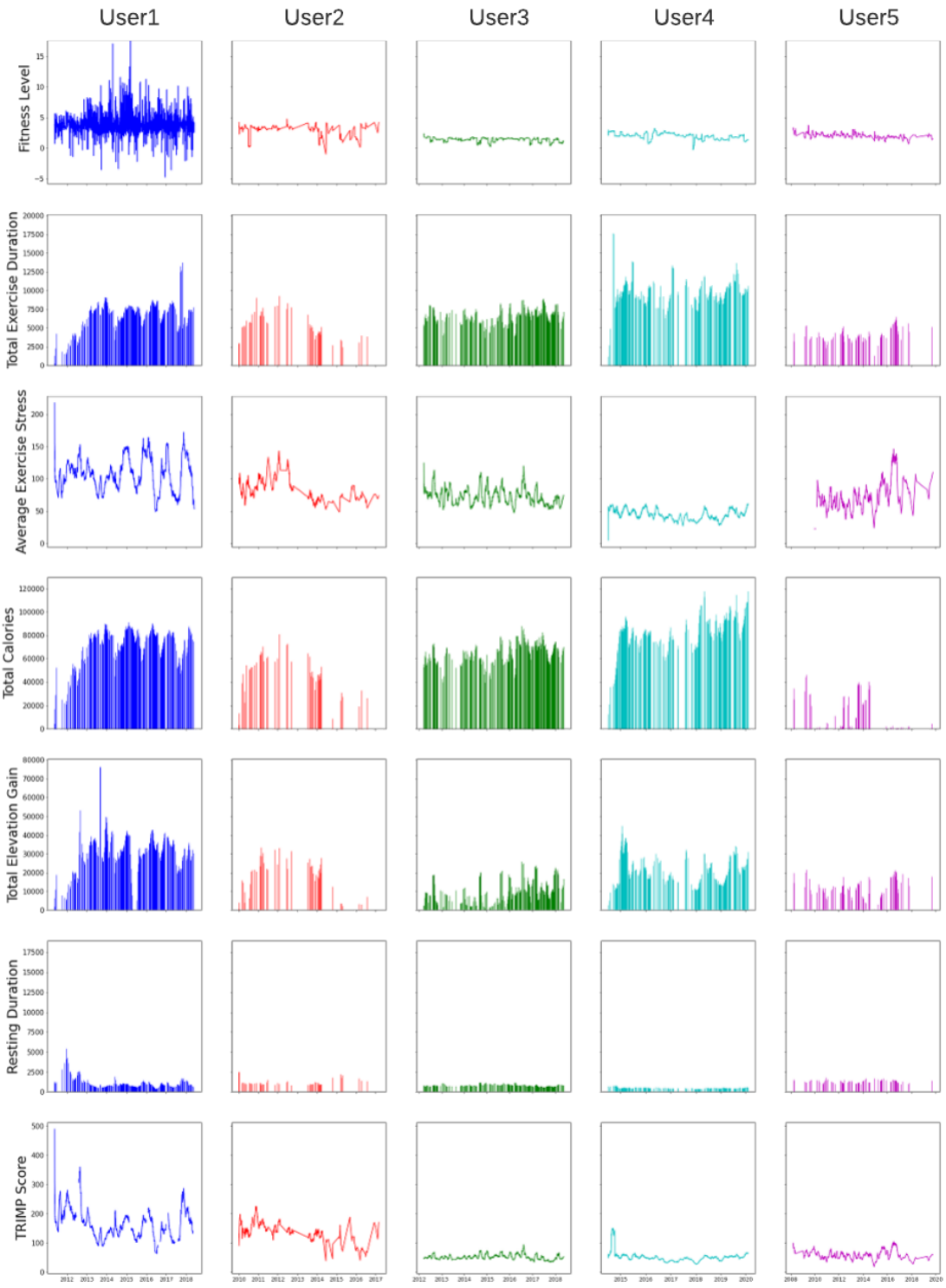

Figure 7.13: Progression of features for the 5 individuals over time. The features were created using pattern-group-aggregate event operations. Each value shown in these plots is related to an outcome event (Exercise with CRF value) and is used in a graphical hypothesis to find the causal effect of the parameters on the performance parameters of the outcome event. All duration features (resting and total exercise) are measured in minutes, Elevation gain is measured in feet.

Figure 7.14: Average causal effects of interventions on different performance parameters. This figure is divided into 4 quadrants based on the Power HRR% zone for the users. Each row of graphs, in a given quadrant, represents the most significant positive (L) and negative (R) intervening factors that causally affects the power produced in the corresponding HRR% zone. Each bar in the graph shows the average causal effect of the intervention represented by the label of the corresponding bar. For example, the most effective intervention for User1 is exercising between 8010 to 8340 minutes in the past 42 days.

7.5 Case Study IV: Context dependent taste preference modeling

Food serves many functions at an individual level. It provides us with the energy and building blocks to sustain our lives while also serving as a source of personal fulfillment and social glue. Our taste and sensory preferences are significant causal factors behind our food decisions and affect our health. For this reason, there is a rapidly growing need for personalized food services that guide users towards a healthier lifestyle while also ensuring the food's enjoyment. With the advancement of technology, especially in the recommendation and sensing fields, it is possible to guide users towards a healthier lifestyle by understanding their underlying taste profile and their daily lifestyles to provide healthier recommendations that still appeal to the user's tastes [126]. Food is an essential part of our lives, and advancements in applications such as food logging platforms and recipe recommendations can help us identify and improve our eating behavior.

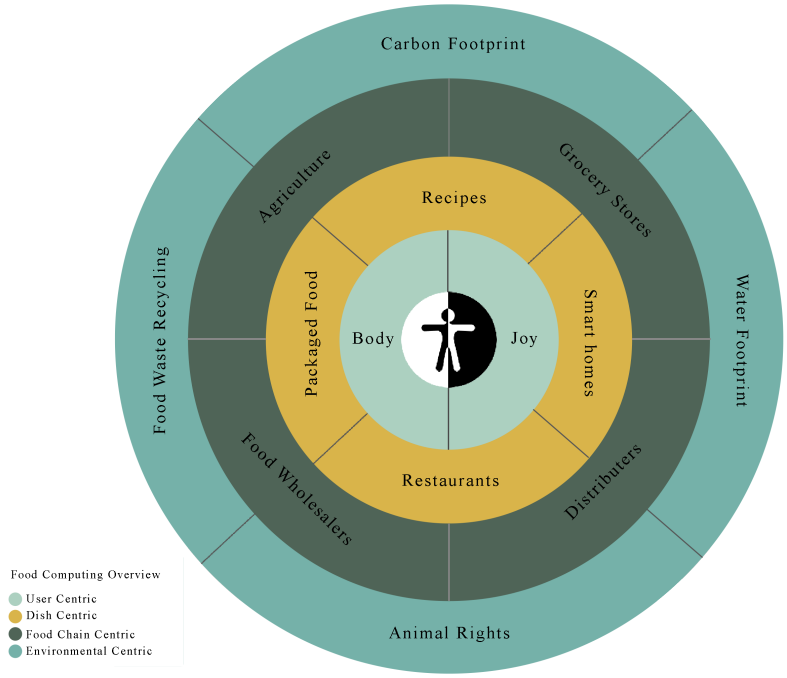

Figure 7.15: Personal food computing overview and relevant proposed layers.

At the heart of these personalized food services lies the Personal Food Model[156]. As shown in figure 7.15, it (PFM) has two main components: 1) the biological component and 2) the preferential component. The biological component determines how different food items interact with our biology and health [128, 123]. In contrast, the preferential component captures how different contextual and environmental factors impact our food preferences and, in turn, affect our choices. There has been a lot of work done on context-aware preference modeling and recommendations. However, the current approaches are still far from truly personalizing these recommendations. Especially for health and food-related applications, the contextual factors can be captured by different multi-modal devices and applications. Typically, these applications store individual data in their silos, which do not interact with other applications. We propose a comprehensive food event model that could provide a mechanism for these applications to cross-utilize each other's data. We also demonstrate how we could utilize the data collected using such applications to create a user preference model and how it varies with different contextual factors such as stress and temperature. We use event mining principles to model the contextual relationships in an unsupervised manner.

7.5.1 Food Event Model

A single food event has multiple facets captured by different applications. Some examples are the food being eaten, the time of the day, the amount of food, the location type, the ambiance, the person eating the food, and other people involved in that event. If we capture sufficient contextual information about the food event, it will be possible to find what caused it. Furthermore, collecting information about the body's response to the food event opens the door to understanding the biological responses to the food event. Inspired by the multimedia event model described by Westermann and Jain [197], we propose a unified food event model that contains all the factors defining the food event, as shown in figure 7.16. The food

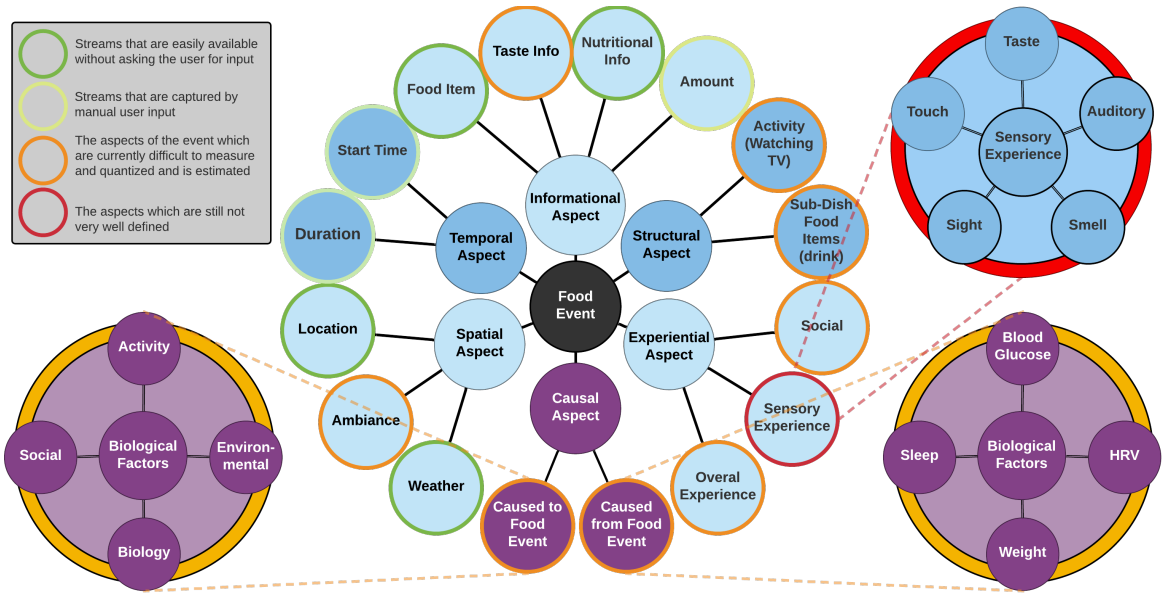

Figure 7.16: Food Event Model: It is essential to capture all the different aspects that a food event contains in order to build powerful models. The causal aspect of a food event is especially challenging to capture. In the bottom right and left corners, we see prior events that cause food events to occur on the bottom left (such as a user's taste model), and we see what future events the food event is responsible for affecting (such as a user's health).

event model consists of 6 main aspects: **spatial**, **experiential**, **informational**, **structural**, **temporal** and **causal** aspect. Each of these aspects has sub-components, as illustrated in figure 7.16. Some of these aspects have been studied extensively and are widely captured, such as location and time. We can capture the biological aspects in free-living conditions thanks to the advances in Internet-of-Things (IoT) and wearable technologies such as sleep monitoring [14]. Physiological data-streams such as Electrocardiography (ECG) and Photoplethysmography (PPG) are non-invasive and low-cost techniques and enable continuous health, and well-being data collection [122], [73]. However, some other sub-components are challenging to collect as they are not understood very well, such as the sensory experience. Auditory and visual information are the only exceptions and are well understood in the multimedia field; however, no such model exists for the sense of taste and smell. The taste information of a food event is crucial for building the preferential side of the personal food model, but to the best of our knowledge, there is currently no method to map food items to ingredients to taste information. We utilize a novel approach to capture information about

the taste experience of a dish driven by the informational aspect discussed in detail in the appendix.

7.5.2 The Causal Aspect

Identifying an event's cause(s) is not easy to answer, even in trivial cases. Numerous factors could affect a food event, such as physical activity, social gathering, weather, or the time of the day. Using current methods, modeling this aspect of a food event is extremely challenging; however, it is critical for building a Personal Model. As shown in figure 7.16, the causal aspect has two sub-components: events that caused the food event and events that the food event has caused. Events caused by a food event are primarily reflected in the biological impact of food. This includes changes in heart rate variability, sleep quality, and other effects on the body and health. On the other hand, the events that caused a food event could be more complicated, as external factors could also influence and initiate a food event. These include social events, environmental factors, and weather conditions. Many environmental and biological factors have been known to affect a food event. Some biological factors may be easier to model, for example, age and weight, which are shown to affect the food decision-making process [183]. Psychological aspects may be more challenging to measure, like mental stress; however, many studies have shown that it can be measured using wearable devices and even social media usage [163]. [5] brings excellent intuition on how stress can have a strong influence on food choice. A food event also depends on environmental factors such as the weather [61]. Complex environmental causal factors are often missed, which can bring substantial help in the context reconstruction. For example, a pandemic, such as the COVID-19 can drastically affect the food habits of populations. [111] shows that for geographical regions with higher numbers of daily COVID-19 cases, the historical trends in search queries related to bars and restaurants are strongly correlated with re-openings happening in those areas. Therefore the environmental knowledge is an essential factor in

the causal aspect of food. In this study, we picked two crucial causal factors: stress and weather, to demonstrate how the context can change the user's food preference profile and analyze how they affect the dietary choices.

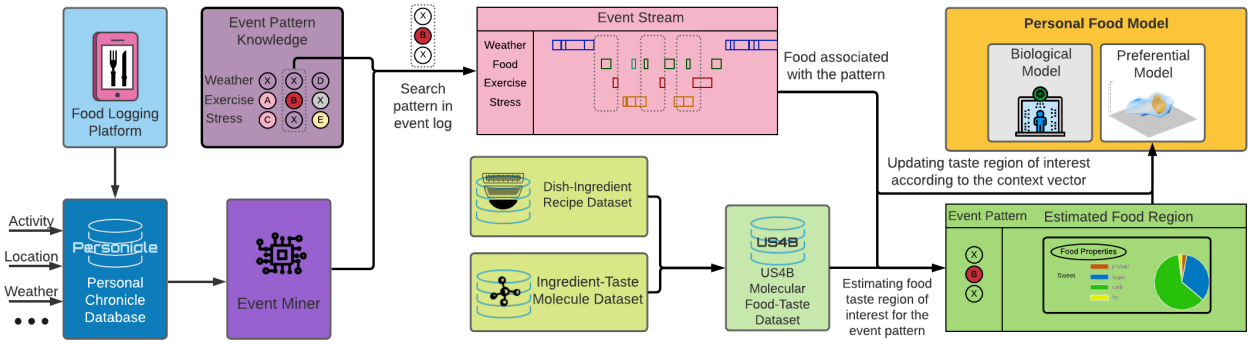

Figure 7.17: Causal Preferential Model Architecture: The food logging platform captures the different aspects of the food events. We use event mining to find contextual patterns and build a taste profile for each pattern and update the preferential subsection of the personal food model.

7.5.3 Experimental Design

We present a novel food preference model that considers causal factors to estimate taste preferences in a particular context. The food model captures the user's preferred taste region, which could change with context. Figure 7.17 illustrates the overall architecture of the preferential food model. Food logger [157] collects information about the food event and stores in the Personicle. The Personicle is a database containing different data streams about the user over a long time in one place[135]. We apply event mining operators on the user's personicle [137][139] to identify contextual factors that impact food preferences. We create event patterns relating different contextual factors with the meal events and find all occurrences of these contextual event patterns in the event streams. This allows us to find food items consumed in different contexts. We can then aggregate the corresponding taste vectors to find the contextual taste preference.

We opted to utilize synthesized data for the experiments because we can use the ground

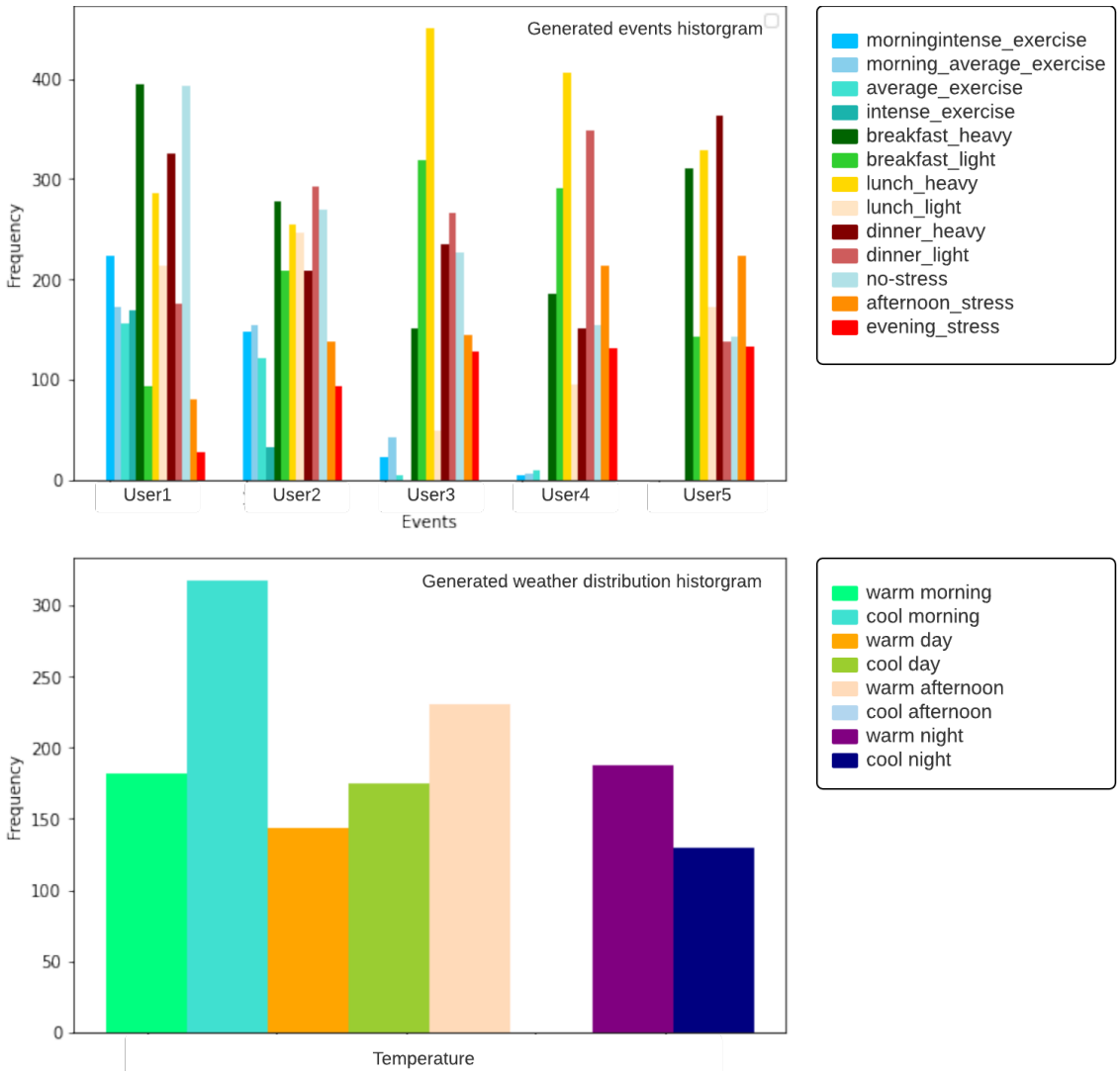

Figure 7.18: Dataset Summary: This figure displays a summary of our events dataset. This includes the frequency distribution for the different events that are present in the events log for the five people in our dataset. The event relationships were encoded as probabilistic transitions in a Markov-chain model. Concurrent and past contextual events also affect the parameters of the lifestyle events.

truth of contextual factors' impact on taste to validate the model, just like in many other works such as [16]. By using synthesized training data, the dataset size can be significantly increased with little human labor [22]. [141] introduces a system that automatically creates synthetic data to enable data scientists. [141] suggests that synthetic data can successfully replace original data for data science if it meets two requirements: First, it must somewhat resemble the original data statistically to ensure realism and keep problems engaging for

data scientists. Second, it must also formally and structurally resemble the original data so that any software written on top of it can be reused.

We designed a rich event stream database, and the occurrences and parameters of these events depend on contextual factors such as time of the day, temperature, and stress. We use the novel US4B taste estimation method to estimate the taste-related molecules' quantity in a dish as the taste cue for the personal model. We showcase multiple experiments on our rich event stream data set generated using a randomized Markov-chain-based event generator. We created five different lifestyle profiles, which would determine the generated events for five different people over 500 days with approximately three food events a day for a total of 7373 food events (Figure 7.18). The lifestyle profiles consist of the parameters needed for the Markov-chain model to generate the event streams. These parameters include the probability distribution of each event occurrence based on the previous event, designed to imitate a natural event stream. These events include food events that are controlled by contextual variables such as stress and weather. Research shows that stress correlates with eating more palatable and delicious food [5]. Even though the relationship could be both ways, either overeating or not eating as much depending on the person. We designed the parameters associated with the stress-related causal aspect of a food choice based on the available findings such that if a person had a stressful day, it would impact their food choice towards more palatable foods for some subjects and towards less appetite for others. Accordingly, some of our synthetic subjects have a higher probability of having a stressful day than the others to achieve a greater variety within the dataset. The causal relation of weather context with food choice has also been studied. [61] shows that combining weather context in food profile modeling yields better results. We also have distribution parameters regarding the weather condition and parameters that affect each subject's food choice based on the weather context. We then use event mining operators to find causal relationships between contextual factors and meal events in the generated data. The underlying causal relationships in the synthetic data were hidden from the event mining system and the person

doing the analysis.

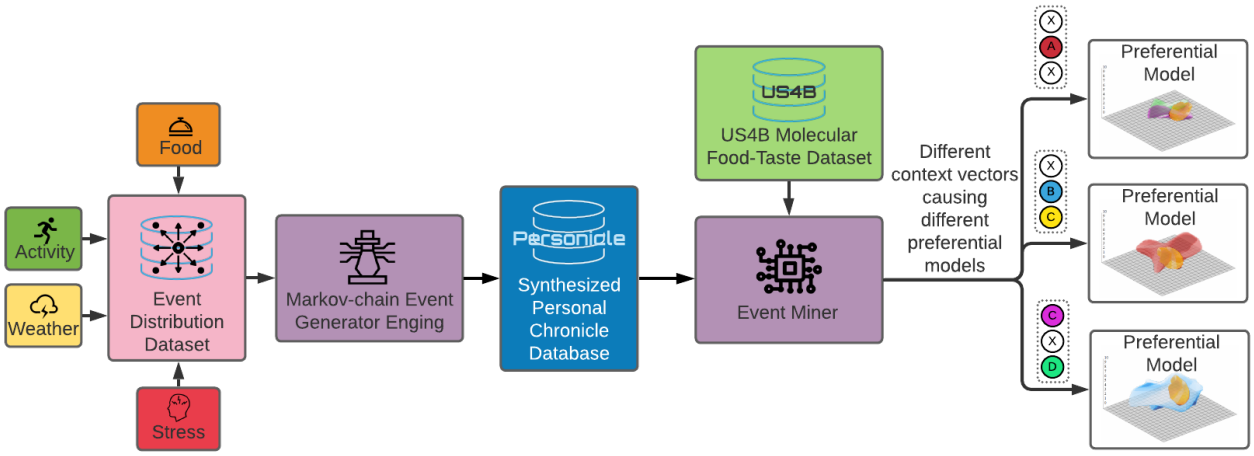

Figure 7.19: Experimental Design: This figure illustrates how we perform hypothesis testing using synthetic data. The events dataset contains the list of event types which are to be generated such as food and activity. The events must resemble the real data statistically so the parameters are carefully selected and are fed to the Markov-chain event generator engine to create the synthesized dataset. Then we use event mining to apply our model to the dataset and test its viability in action.

7.5.4 Results

We attempted to answer three research questions (RQ) in our experiments:

1. How does the individual taste preference vector change with changes in contextual parameters?

2. How does adding context-awareness change the predictive performance of the preference model?

3. How much data is needed to create a stable model?

RQ1: Contextual variation in taste profile

Figure 7.20 shows the variation in the preferred taste profile with different contextual variables for the five individuals in our dataset. We created the individuals' contextual taste profile by averaging the US4B taste vectors for meals consumed in different contextual situations. Thus, every individual has nine contextual preference vectors (3 temperature levels * 3 stress levels) for every meal (breakfast, lunch, and dinner). The contextual preference vectors are compared against the average preference vector for the three meals in the radar plots in fig. 7.20. We have included the radar plots for two users. We can see that **user5** has an increased preference for umami flavored food for dinner when it is cool outside, but that preference goes down with an increase in temperature, and **user1**'s preference for sweet, bitter, and umami flavors during lunch goes down with an increase in temperature.

RQ2: Comparison of prediction performance

We compared the context-aware preference model's predictive performance against the "No-Context" preference model using Top-5 accuracy as our performance measure. We used an 80-20 train-test split while maintaining the chronological order (test set samples were from a period after the training set) and reported the models' performance on the test set. We used a nearest-neighbor approach to match an individual's preference vector with the available food items using cosine similarity. We predicted the five most likely food items for every meal event in the test set and compared the predictions against the meal event's actual dish. Figure 7.21 compares the performance of models with different levels of contextual information. As we expected, adding contextual information leads to a better performance than the "No-Context" model for all five individuals in our dataset.

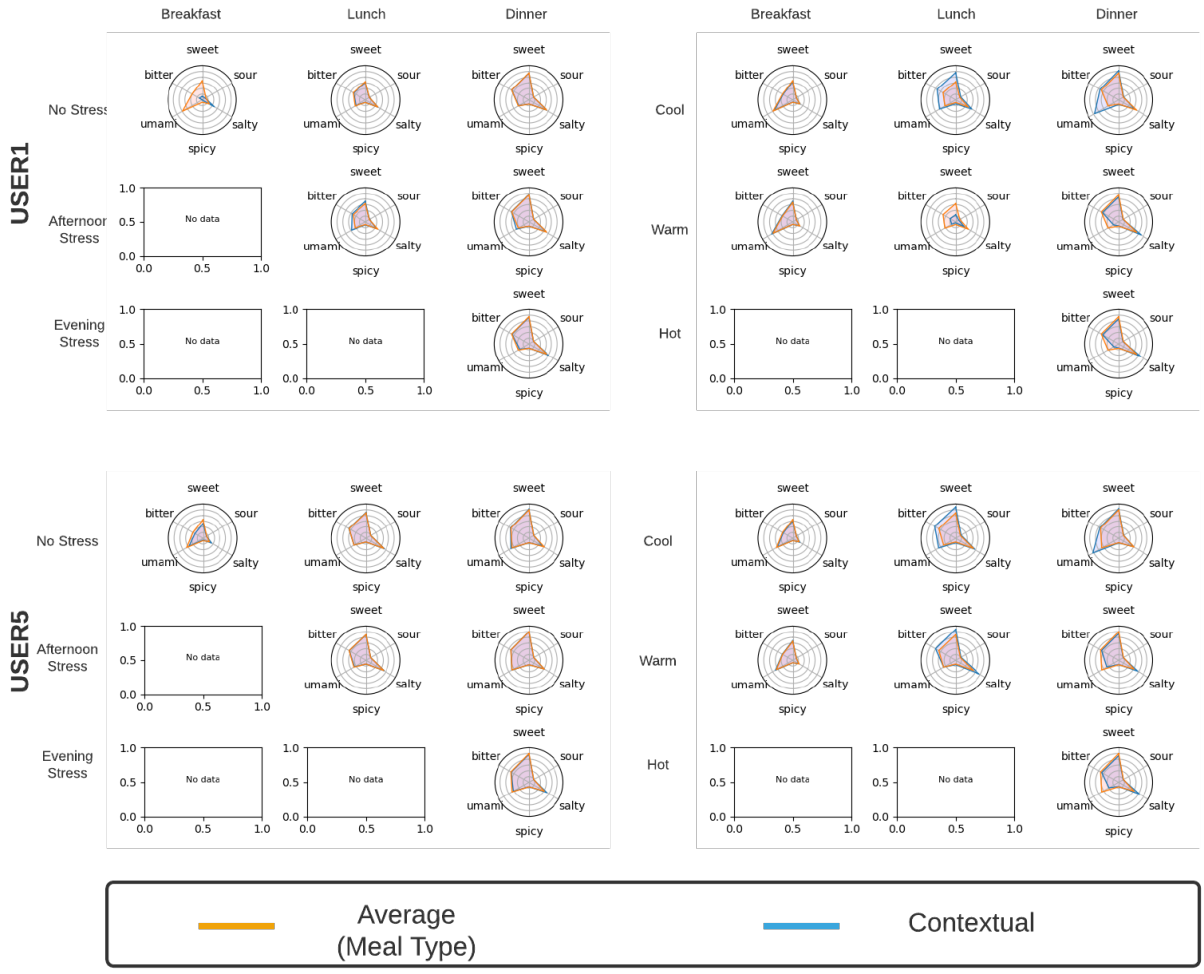

Figure 7.20: Variation in taste preferences with context. This figure shows how the preferences for different taste aspects change with context. We can see that for User1, the preference for sweet, bitter, and umami flavors during lunch goes down with increase in temperature.

RQ3: Model accuracy with training data volume

We also performed experiments to find how the model accuracy varies with the amount of training data. We used a fixed test set containing events data for 100 days. The training set size was varied on a logarithmic scale from 4 to 400 (with a factor of 2). We used the top-5 accuracy metric, and the results are reported in fig. 7.22 for all users. We observe that initially, the non-contextual model outperforms the context-aware model. This could be due

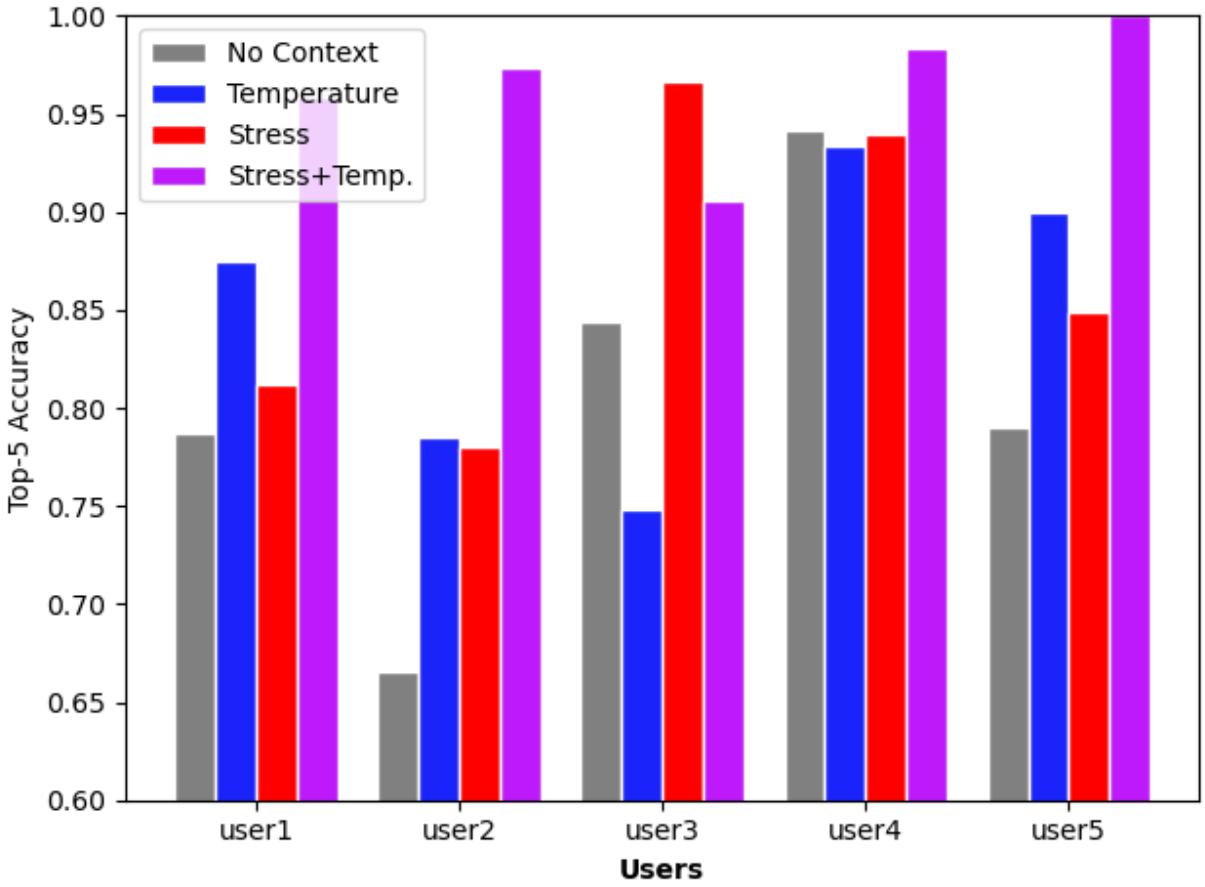

Figure 7.21: Model performance using Top-5 predictions accuracy. We can see that for all users adding all contextual factors (Stress+Temperature) leads to a better model than no contextual information.

to a lack of data in different contextual situations. The observation supports this explanation that as the size of the training dataset increases, the context-aware model outperforms the non-contextual model. The accuracy graph starts flattening at 128 days; thus, we would need to collect about 100 days of events to train and use this model effectively.

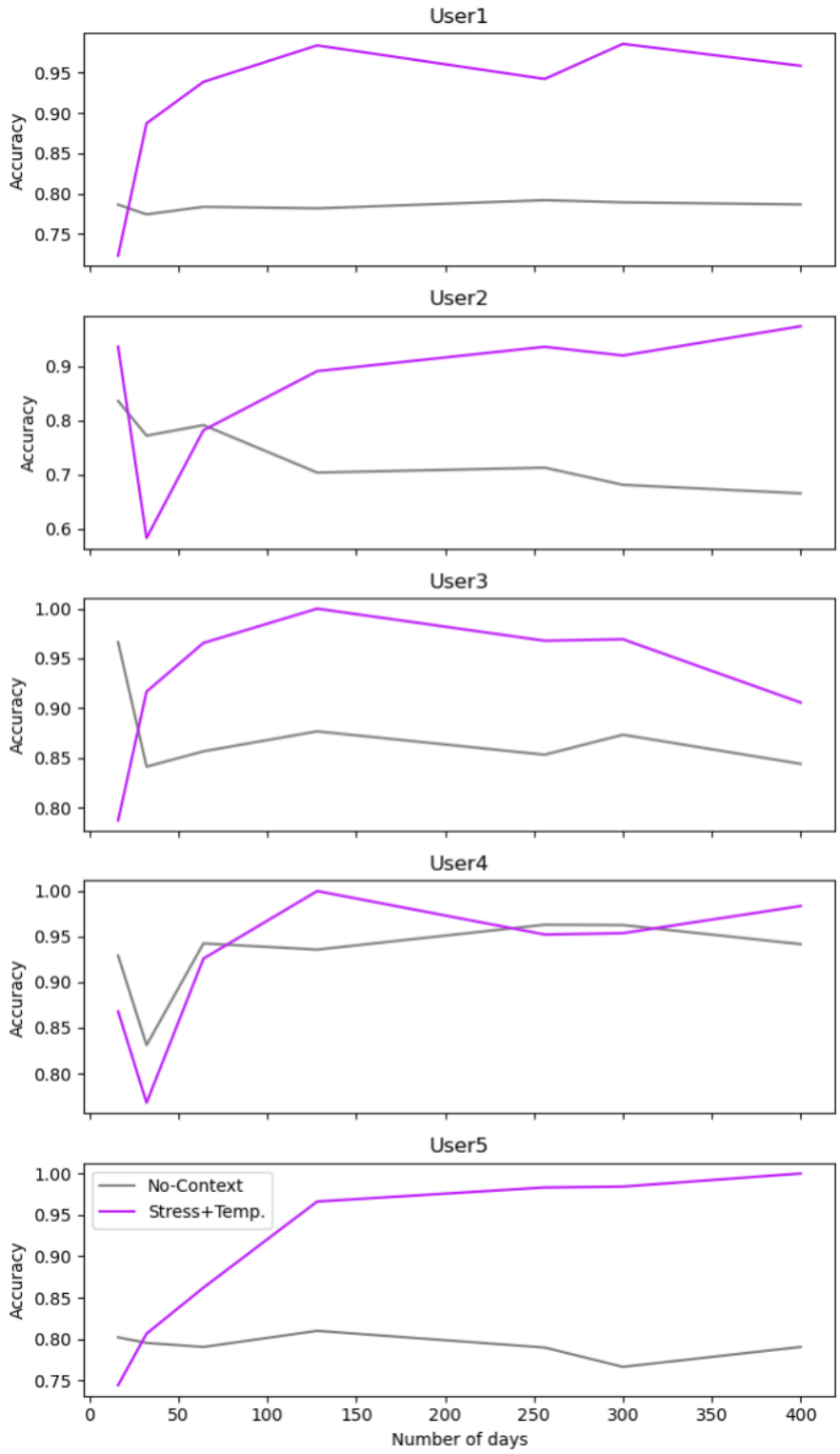

Figure 7.22: Model accuracy vs training data volume. The context-aware model appears to stabilize at 128 days as mentioned in Section 6.3. As expected, initially the non-contextual model outperforms the context-aware model, but with more training data, the context-aware model has the higher accuracy.

Chapter 8

Conclusion

Data is everywhere in today's world. Every electronic device and application we use observes an aspect of our behavior and daily lives. Governments and public agencies continuously monitor environmental and societal factors that affect our behavior, lifestyle, and health. Different applications and research efforts have leveraged such data to solve problems related to personal health in clinical settings. Deep Neural Networks have been utilized in multimedia-rich fields such as radiology for diagnosing various diseases such as cancer and Alzheimer's. Neural networks have also contributed significantly to genetic sequencing and AI-driven drug discovery. However, a large amount of lifestyle data being generated by individuals is not being utilized efficiently for health estimation and guidance. Nag et al. [123] propose a framework for continuous health estimation and guidance that utilizes multimodal lifestyle data. Such works require a system for processing the raw multimodal streams into meaningful events representing the user and environmental activities. The system should reason with the extracted events and identify event patterns representative of user behavior. The events can also estimate the effects of various habits and activities on individual behavior and health.

In this work, we attempt to address the problems of utilizing multimodal data from disparate sources for personalized longitudinal modeling. We propose an N-of-1 modeling paradigm for understanding user behavior and estimating the effects of user events on individual health and behavioral attributes. Traditional modeling strategies that study phenomena at a population or sub-population level are great tools for studying general causal relationships that hold for populations but are not sufficiently precise for understanding and making predictions at an individual level (Chapter 2). We have addressed several aspects of this problem, such as :

1. *Multimodal data fusion* in the form of multimodal events that are capable of relating information from different sources to a real-world activity. Data from different sources capture an aspect of the event, and thus the resulting event has more information than events recognized from a single source (Chapter 4).

2. *Knowledge Integration* in the form of user-defined events as well as hypotheses. Event creation operations allow analysts to define new complex events as a combination of existing ones. Extracting these events in a purely data-driven manner is a challenging task; however, analysts can easily define these events and include them in the analysis using the event operators. Similarly, the graphical representation of a hypothesis and the hypothesis testing operations described in chapter 5 allow analysts and expert users to incorporate domain knowledge as event patterns in the hypothesis (Chapter 5).

3. *Frequent Event Patterns discovery* allows the analysts to discover co-occurring events and discover temporal relationships between such events. The patterns are useful because these inform the analysts about frequent user behaviors and can be abstracted as a complex event if the co-occurrence is strong enough. These patterns may also include unknown causal relationships between the events that can be tested either using the hypothesis testing operation or experimentally (Chapter 6).

Different parts of the framework are evaluated in multiple published works to study events

and hypotheses related to personal health and behavior, such as exercise behavior, cardio-respiratory health, and contextual dietary preferences (Chapter 7).

However, many aspects of a generalized N-of-1 modeling system have not been explored in this work and require further research.

- **Visual pattern representation**: Innovative visualizations for event sequences and event patterns data would allow analysts to explore data interactively. We have utilized a linear and radial timeline visualization for events and a co-occurrence matrix visualization for patterns in this work. However, these can easily become too large to manage, and larger patterns cannot be displayed on a matrix; thus, we need to explore specialized visual representations such as Sankey and tree diagrams to display event patterns.

- **Interactive hypothesis building**: An interactive graph interface for specifying new hypotheses would allow analysts and domain experts to easily incorporate their beliefs into a hypothesis without being familiar with the background event formulations and operators.

- **Additional pattern operators**: In this work, we have focused on pattern operators that capture positive event correlations; however, negative event correlations are equally important in health and lifestyle. For example, identifying interventions that reduce symptoms (headache, heartburn, etc.) can only be done by finding negative event patterns.

- **Data Security and Ownership**: Recent technological and policy developments such as GDPR have raised awareness among users about the value of data security and privacy. Any application dealing with personal lifestyle and health data has to ensure the highest standards of data security. We must explore security and database architectures that allow secure data sharing with third-party applications and enable the

end-users to have complete ownership and control over their data.

- **Secure Model Sharing**: As discussed in Chapter 2, aggregating individual models can provide insights into generalized behavioral patterns across populations, and the aggregated population and sub-population models can also serve as a solution to the *cold start* problem commonly observed when there is a lack of individual-specific data.

Bibliography

[1] Peekquence: Visual Analytics for Event Sequence Data — Bum Chul Kwon, Ph.D. - IBM Research.

[2] Periodic Pattern Mining – Algorithms and Applications — Global Journal of Computer Science and Technology.

[3] The New Person-Specific Paradigm in Psychology on JSTOR.

[4] *The Essential Guide to N-of-1 Trials in Health*. Springer Netherlands, 2015.

[5] T. C. Adam and E. S. Epel. Stress, eating and the reward system. *Physiology and Behavior*, 91(4):449–458, 8 2007.

[6] G. Adomavicius and A. Tuzhilin. Context-Aware Recommender Systems. In *Recommender Systems Handbook*, pages 191–226. Springer US, Boston, MA, 2015.

[7] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules.

[8] R. Agrawal and R. Srikant. Mining sequential patterns. *Proceedings - International Conference on Data Engineering*, pages 3–14, 1995.

[9] X. Alameda-Pineda, E. Ricci, and N. Sebe. Multimodal behavior analysis in the wild: An introduction. In *Multimodal Behavior Analysis in the Wild*, pages 1–8. Elsevier, 2019.

[10] J. F. Allen. INTERVAL-BASED REPRESENTATION OF TEMPORAL KNOWLEDGE. volume 1, 1981.

[11] J. F. ALLEN and G. FERGUSON. Actions and Events in Interval Temporal Logic. *Journal of Logic and Computation*, 4(5):531–579, 10 1994.

[12] C. Andersson, A. D. Johnson, E. J. Benjamin, D. Levy, and R. S. Vasan. 70-year legacy of the Framingham Heart Study, 11 2019.

[13] S. Aseervatham, A. Osmani, and E. Viennet. BitSPADE: A lattice-based sequential pattern mining algorithm using bitmap representation. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 792–797, 2006.

[14] M. Asgari Mehrabadi, I. Azimi, F. Sarhaddi, A. Axelin, H. Niela-Vilén, S. Myllyntausta, S. Stenholm, N. Dutt, P. Liljeberg, and A. M. Rahmani. Sleep Validation of Commercially Available Smart Ring and Watch Against Medical-Grade Actigraphy in Everyday Settings (Preprint). *JMIR mHealth and uHealth*, 8 2020.

[15] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential PAttern mining using a bitmap representation. page 429, 2002.

[16] K. Barnard, V. Cardei, and B. Funt. A comparison of computational color constancy algorithms - Part I: Methodology and experiments with synthesized data. *IEEE Transactions on Image Processing*, 11(9):972–984, 8 2002.

[17] E. J. Benjamin, D. Levy, S. M. Vaziri, R. B. D'agostino, A. J. Belanger, and P. A. Wolf. Independent Risk Factors for Atrial Fibrillation in a Population-Based Cohort: The Framingham Heart Study. *JAMA: The Journal of the American Medical Association*, 271(11):840–844, 3 1994.

[18] H. Carte, A. M. Jones, and J. H. Doust. Effect of 6 weeks of endurance training on the lactate minimum speed. *Journal of Sports Sciences*, 17(12), 1 1999.

[19] F. Casino, C. Patsakis, E. Batista, O. Postolache, A. Martínez-Ballesté, and A. Solanas. Smart Healthcare in the IoT Era: A Context-Aware Recommendation Example. In *2018 International Symposium in Sensing and Instrumentation in IoT Era, ISSI 2018*. Institute of Electrical and Electronics Engineers Inc., 11 2018.

[20] R. B. CATTELL. P-technique, a new method for analyzing the structure of personal motivation. *Transactions of the New York Academy of Sciences*, 14(1):29–34, 1951.

[21] D. Chankhihort, B.-M. Lim, G.-J. Lee, S. Choi, S.-O. Kwon, S.-H. Lee, J.-T. Kang, A. Nasridinov, and K.-H. Yoo. A Visualization Scheme with a Calendar Heat Map for Abnormal Pattern Analysis in the Manufacturing Process 21. *International Journal of Contents*, 13(2), 2017.

[22] Q. Chen, W. Qiu, Y. Zhang, L. Xie, and A. Yuille. SampleAhead: Online Classifier-Sampler Communication for Learning from Synthesized Data. *British Machine Vision Conference 2018, BMVC 2018*, 8 2018.

[23] H. R. E. Costill, DL; Thomason. Fractional utilization of the aerobic capacity during distance running. *Med Sci Sports*, pages 248–252, 1973.

[24] E. J. Daza. Causal Analysis of Self-tracked Time Series Data Using a Counterfactual Framework for N-of-1 Trials. *Methods of Information in Medicine*, 57(1):e10–e21, 2018.

[25] S. de Amo, W. P. Junior, and A. Giacometti. MILPRIT*: A Constraint-Based Algorithm for Mining Temporal Relational Patterns. *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jdwm.2008100103*, 4(4):42–61, 1 1.

[26] S. De Amo, W. P. Junior, A. Giacometti, and T. G. Clemente. Mining Temporal Relational Patterns over Databases with Hybrid Time Domains.

[27] G. de Simone. Concentric or eccentric hypertrophy: how clinically relevant is the difference? *Hypertension (Dallas, Tex. : 1979)*, 43(4):714–5, 4 2004.

[28] A. H. Diacon, C. F. Koegelenberg, K. J. Klüsmann, and C. T. Bolliger. Challenges in the estimation of tidal volume settings in critical care units [10], 10 2006.

[29] C. Dobbins, R. Rawassizadeh, and E. Momeni. Detecting physical activity within lifelogs towards preventing obesity and aiding ambient assisted living. *Neurocomputing*, 230:110–132, 3 2017.

[30] A. R. Doherty, A. F. Smeaton, K. Lee, and D. P. W. Ellis. Multimodal Segmentation of Lifelog Data. Technical report.

[31] N. Duan, R. L. Kravitz, and C. H. Schmid. Single-patient (n-of-1) trials: A pragmatic clinical decision methodology for patient-centered comparative effectiveness research. *Journal of Clinical Epidemiology*, 66(8 SUPPL.8), 8 2013.

[32] M. Eichler, R. Dahlhaus, and J. Dueck. Graphical Modeling for Multivariate Hawkes Processes with Nonparametric Link Functions. *Journal of Time Series Analysis*, 38(2):225–242, 5 2016.

[33] K. El Asnaoui, A. Hamid, A. Brahim, and O. Mohammed. A survey of activity recognition in egocentric lifelogging datasets. In *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems, WITS 2017*. Institute of Electrical and Electronics Engineers Inc., 5 2017.

[34] D. Entner and P. O. Hoyer. On Causal Discovery from Time Series Data using FCI.

[35] J. A. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A Visual Interface for Multivariate Temporal Data: Finding Patterns of Events across Multiple Histories.

[36] F. Fischer, J. Fuchs, P. A. Vervier, F. Mansmann, and O. Thonnard. VisTracer: A visual analytics tool to investigate routing anomalies in traceroutes. *ACM International Conference Proceeding Series*, pages 80–87, 2012.

[37] E. Fountzilas and A. M. Tsimberidou. Overview of precision oncology trials: challenges and opportunities, 8 2018.

[38] P. Fournier-Viger, J. Chun, W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas. A Survey of Sequential Pattern Mining. 1(1), 2017.

[39] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas. Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8443 LNAI(PART 1):40–52, 2014.

[40] P. Fournier-Viger, C.-W. Wu, A. Gomariz, and V. S. Tseng. VMSP: Efficient Vertical Mining of Maximal Sequential Patterns. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8436 LNAI:83–94, 2014.

[41] P. Fournier-Viger, C.-W. Wu, and V. S. Tseng. Mining Maximal Sequential Patterns without Candidate Maintenance.

[42] S. S. Franklin, M. G. Larson, S. A. Khan, N. D. Wong, E. P. Leip, W. B. Kannel, and D. Levy. Does the relation of blood pressure to coronary heart disease risk change with aging?: The Framingham Heart Study. *Circulation*, 103(9):1245–1249, 3 2001.

[43] C. Freksa and R. Fulton. Temporal Reasoning Based on Semi-Intervals Freksa Temporal Reasoning Based on Semi-Intervals 2 TIME IS A MASK WORN BY SPACE. *Artificial Intelligence*, 54:199–227, 1992.

[44] T. J. Gabbett, B. T. Hulin, P. Blanch, and R. Whiteley. High training workloads alone do not cause sports injuries: how you get there is the real issue. *British Journal of Sports Medicine*, 50(8), 4 2016.

[45] N. B. Gabler, N. Duan, S. Vohra, and R. L. Kravitz. N-of-1 trials in the medical literature: A systematic review, 8 2011.

[46] R. A. García-Hernández, J. F. Martínez-Trinidad, and J. A. Carrasco-Ochoa. A New Algorithm for Fast Discovery of Maximal Sequential Patterns in a Document Collection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3878 LNCS:514–523, 2006.

[47] N. Garg, A. Sethupathy, R. Tuwani, R. Nk, S. Dokania, A. Iyer, A. Gupta, S. Agrawal, N. Singh, S. Shukla, K. Kathuria, R. Badhwar, R. Kanji, A. Jain, A. Kaur, R. Nagpal, and G. Bagler. FlavorDB: A database of flavor molecules. *Nucleic Acids Research*, 46(D1):D1210–D1216, 8 2018.

[48] S. Guo, F. Du, S. Malik, E. Koh, S. Kim, Z. Liu, D. Kim, H. Zha, and N. Cao. Visualizing uncertainty and alternatives in event sequence predictions. *Conference on Human Factors in Computing Systems - Proceedings*, page 12, 5 2019.

[49] C. Gurrin, A. F. Smeaton, and A. R. Doherty. LifeLogging: Personal big data, 2014.

[50] V. H. Haase. Hypoxia-inducible factors in the kidney, 2006.

[51] P. Hackett, R. Roach, and J. Sutton. High-altitude medicine. *Wilderness medicine*, 4:2–43, 1995.

[52] HanJiawei, PeiJian, and YinYiwen. Mining frequent patterns without candidate generation. *ACM SIGMOD Record*, 29(2):1–12, 5 2000.

[53] A. G. Hawkes. Spectra of Some Self-Exciting and Mutually Exciting Point Processes. *Biometrika*, 58(1):83, 4 1971.

[54] A. P. Hills, S. J. Street, and N. M. Byrne. Physical Activity and Health: "What is Old is New Again". In *Advances in Food and Nutrition Research*, volume 75, pages 77–95. Academic Press Inc., 2015.

[55] F. Höppner and F. Klawonn. Finding informative rules in interval sequences. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2189:125–134, 2001.

[56] M. C. Howard and M. E. Hoffman. Variable-Centered, Person-Centered, and Person-Specific Approaches: Where Theory Meets the Method. *Organizational Research Methods*, 21(4):846–876, 10 2018.

[57] S. C. Hsueh, M. Y. Lin, and C. L. Chen. Mining negative sequential patterns for e-commerce recommendations. *Proceedings of the 3rd IEEE Asia-Pacific Services Computing Conference, APSCC 2008*, pages 1213–1218, 2008.

[58] O. Huisman and P. Forer. The complexities of everyday life : balancing practical and realistic approaches to modelling probable presence in space - time. *undefined*, 2005.

[59] A. Hyvärinen, S. Shimizu, and P. O. Hoyer. Causal Modelling Combining Instantaneous and Lagged Effects: an Identifiable Model Based on Non-Gaussianity.

[60] K. Imai, G. King, and E. A. Stuart. Misunderstandings between experimentalists and observationalists about causal inference. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 171(2):481–502, 4 2008.

[61] T. Ito, Y. Fukazawa, D. Zhu, and J. Ota. Modeling Weather Context Dependent Food Choice Process. *Journal of Information Processing*, 26(0):386–395, 8 2018.

[62] P. J. J., ChenShang-Tse, KahngMinsuk, B. De, BasoleRahul, SharminMoushumi, and C. Horng. Chronodes. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 8(1), 2 2018.

[63] R. Jain and L. Jalali. Objective self. *IEEE Multimedia*, 21(4):100–110, 2014.

[64] R. Jain and L. Jalali. Objective self. *IEEE MultiMedia*, 21(4):100–110, 10 2014.

[65] L. Jalali and R. Jain. Event Mining for Explanatory Modeling. *Event Mining for Explanatory Modeling*, 5 2021.

[66] Z. Jin, S. Guo, N. Chen, D. Weiskopf, D. Gotz, and N. Cao. Visual Causality Analysis of Event Sequence Data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1343–1352, 9 2020.

[67] Z. Jin, J. Yang, S. Cui, D. Gotz, J. Sun, and N. Cao. CarePre: An Intelligent Clinical Decision Assistance System. *ACM Transactions on Computing for Healthcare*, 1(1):1–20, 11 2018.

[68] A. M. Jones. A five year physiological case study of an Olympic runner. *British Journal of Sports Medicine*, 32(1), 3 1998.

[69] A. M. Jones and H. Carter. The Effect of Endurance Training on Parameters of Aerobic Fitness. *Sports Medicine*, 29(6):373–386, 2000.

[70] N. K, G. A, S. SR, S. WF, M. B, and S. J. PARAMO: a PARAllel predictive MOdeling platform for healthcare analytic research using electronic health records. *Journal of biomedical informatics*, 48:160–170, 2014.

[71] D. Kahneman, A. B. Krueger, D. A. Schkade, N. Schwarz, and A. A. Stone. A Survey Method for Characterizing Daily Life Experience: The Day Reconstruction Method. *Science*, 306(5702), 12 2004.

[72] P. S. Kam and A. W. C. Fu. Discovering temporal patterns for interval-based events. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1874:317–326, 2000.

[73] E. Kasaeyan Naeini, S. Shahhosseini, A. Subramanian, T. Yin, A. M. Rahmani, and N. Dutt. An Edge-Assisted and Smart System for Real-Time Pain Monitoring. In *Proceedings - 4th IEEE/ACM Conference on Connected Health: Applications, Systems and Engineering Technologies, CHASE 2019*, pages 47–52. Institute of Electrical and Electronics Engineers Inc., 8 2019.

[74] E. K. Keenan. Seeing the forest and the trees: Using dynamic systems theory to understand "stress and coping" and "trauma and resilience". *Journal of Human Behavior in the Social Environment*, 20(8):1038–1060, 12 2010.

[75] L. Kempf, J. C. Goldsmith, and R. Temple. Challenges of developing and conducting clinical trials in rare disorders, 4 2018.

[76] M. A. Khan, E. Rushe, B. Smyth, and D. Coyle. Personalized, Health-Aware Recipe Recommendation: An Ensemble Topic Modeling Based Approach. Technical report, 2019.

[77] T. Kim and C. H. Park. Anomaly pattern detection for streaming data. *Expert Systems with Applications*, 149:113252, 7 2020.

[78] J. Krause, A. Perer, and K. Ng. Interacting with predictions: Visual inspection of black-box machine learning models. *Conference on Human Factors in Computing Systems - Proceedings*, pages 5686–5697, 5 2016.

[79] J. Krause, A. Perer, and H. Stavropoulos. Supporting Iterative Cohort Construction with Visual Temporal Queries. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):91–100, 1 2016.

[80] T. Krone, R. Boessen, S. Bijlsma, R. van Stokkum, N. D. Clabbers, and W. J. Pasman. The possibilities of the use of N-of-1 and do-it-yourself trials in nutritional research. *PLoS ONE*, 15(5):e0232680, 5 2020.

[81] I. M. Kronish, M. Hampsey, L. Falzon, B. Konrad, and K. W. Davidson. Personalized (N-of-1) Trials for Depression: A Systematic Review. *Journal of Clinical Psychopharmacology*, 38(3):218–225, 6 2018.

[82] D. Kwasnicka, J. Inauen, W. Nieuwenboom, J. Nurmi, A. Schneider, C. E. Short, T. Dekkers, A. J. Williams, W. Bierbauer, A. Haukkala, F. Picariello, and F. Naughton. Challenges and solutions for N-of-1 design studies in health psychology. *Health Psychology Review*, 13(2):163–178, 4 2019.

[83] B. C. Kwon, M.-J. Choi, J. T. Kim, E. Choi, Y. B. Kim, S. Kwon, J. Sun, and J. Choo. RetainVis: Visual Analytics with Interpretable and Interactive Recurrent Neural Networks on Electronic Medical Records. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):299–309, 5 2018.

[84] S. T. Lanza, B. P. Flaherty, and L. M. Collins. Latent Class and Latent Transition Analysis. In *Handbook of Psychology*. John Wiley & Sons, Inc., 4 2003.

[85] B. P. Laursen and E. Hoff. Person-Centered and Variable-Centered Approaches to Longitudinal Data. *Merrill-Palmer Quarterly*, 52(3), 2006.

[86] P.-M. Law, Z. Liu, S. Malik, and R. C. Basole. MAQUI: Interweaving Queries and Pattern Mining for Recursive Event Sequence Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 25(1), 1 2019.

[87] D. c. Lee, A. G. Brellenthin, P. D. Thompson, X. Sui, I. M. Lee, and C. J. Lavie. Running as a Key Lifestyle Medicine for Longevity, 6 2017.

[88] T. Li Li. EVENT MINING ALGORITHMS AND APPLICATIONS EVENT MINING Chapman & Hall/CRC Data Mining and Knowledge Discovery Series Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Technical report.

[89] F. Liang, S. Ma, and J. L. Hellerstein. Discovering Fully Dependent Patterns. *Proceedings*, pages 511–527, 4 2002.

[90] E. O. Lillie, B. Patay, J. Diamant, B. Issell, E. J. Topol, and N. J. Schork. The n-of-1 clinical trial: The ultimate strategy for individualizing medicine? *Personalized Medicine*, 8(2):161–173, 3 2011.

[91] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery 2007 15:2*, 15(2):107–144, 4 2007.

[92] Z. Liu, B. Kerr, M. Dontcheva, J. Grover, M. Hoffman, and A. Wilson. CoreFlow: Extracting and Visualizing Branching Patterns from Event Sequences. *Computer Graphics Forum*, 36(3):527–538, 6 2017.

[93] Z. Liu, Y. Wang, M. Dontcheva, M. Hoffman, S. Walker, and A. Wilson. Patterns and Sequences: Interactive Exploration of Clickstreams to Understand Common Visitor Paths. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):321–330, 1 2017.

[94] LiuZhicheng, KerrBernard, DontchevaMira, GroverJustin, HoffmanMatthew, and WilsonAlan. CoreFlow. *Computer Graphics Forum*, 36(3):527–538, 6 2017.

[95] M. López-Nores, Y. Blanco-Fernández, J. J. Pazos-Arias, and A. Gil-Solla. Property-based collaborative filtering for health-aware recommender systems. *Expert Systems with Applications*, 39(8):7451–7457, 6 2012.

[96] D. Luckham. *The power of events*. 2002.

[97] D. C. Luckham and B. Frasca. Complex Event Processing in Distributed Systems. Technical report, 1998.

[98] K. Ma, W. Wang, N. Yuan, and X. Chi. Study on temporal patterns of medical emergency call. *Proceedings - The 2015 10th International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2015*, pages 403–407, 1 2016.

[99] S. Ma and J. Hellerstein. Mining Mutually Dependent Patterns. *Proceedings 2001 IEEE International Conference on Data Mining*, pages 409–409, 11 2001.

[100] S. Ma and J. L. Hellerstein. Mining partially periodic event patterns with unknown periods. *Proceedings - International Conference on Data Engineering*, pages 205–214, 2001.

[101] M. S. Magnusson. Discovering hidden time patterns in behavior: T-patterns and their detection. *Behavior Research Methods, Instruments, and Computers*, 32(1):93–110, 2000.

[102] S. S. Mahmood, D. Levy, R. S. Vasan, and T. J. Wang. The Framingham Heart Study and the epidemiology of cardiovascular disease: A historical perspective, 3 2014.

[103] S. Malik, F. Du, M. Monroe, E. Onukwugha, C. Plaisant, and B. Shneiderman. Cohort Comparison of Event Sequences with Balanced Integration of Visual Analytics and Statistics. *Proceedings of the 20th International Conference on Intelligent User Interfaces*.

[104] J. E. Manson and S. S. Bassuk. Invited commentary: The framingham offspring study-A pioneering investigation into familial aggregation of cardiovascular risk, 6 2017.

[105] Mark Liversedge. Golden Cheetah Open Dataset, 2018.

[106] J. McCarthy, P. H. R. i. a. intelligence, and u. 1981. Some philosophical problems from the standpoint of artificial intelligence. *Elsevier*.

[107] A. L. Mccutcheon. Latent Class Analysis In: Latent Class Analysis. 1987.

[108] S. McDonald, F. Quinn, R. Vieira, N. O'Brien, M. White, D. W. Johnston, and F. F. Sniehotta. The state of the art and future opportunities for using longitudinal n-of-1 methods in health behaviour research: a systematic literature overview, 10 2017.

[109] K. McGarry. A survey of interestingness measures for knowledge discovery. *Knowledge Engineering Review*, 20(1):39–61, 2005.

[110] D. C. Mckenzie. Markers of Excessive Exercise. *Canadian Journal of Applied Physiology*, 24(1), 2 1999.

[111] M. A. Mehrabadi, N. Dutt, and A. M. Rahmani. The Causality Inference of Public Interest in Restaurants and Bars on COVID-19 Daily Cases in the US: A Google Trends Analysis. 8 2020.

[112] H. Mei and J. Eisner. The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. *Advances in Neural Information Processing Systems*, 2017-December:6755–6765, 12 2016.

[113] C. Mihl, W. R. Dassen, and H. Kuipers. Cardiac remodelling: Concentric versus eccentric hypertrophy in strength and endurance athletes, 2008.

[114] R. D. Mirza, S. Punja, S. Vohra, and G. Guyatt. The history and development of N-of-1 trials. *Journal of the Royal Society of Medicine*, 110(8):330–340, 8 2017.

[115] G. F. Mitchell, S. J. Hwang, R. S. Vasan, M. G. Larson, M. J. Pencina, N. M. Hamburg, J. A. Vita, D. Levy, and E. J. Benjamin. Arterial stiffness and cardiovascular events: The framingham heart study. *Circulation*, 121(4):505–511, 2 2010.

[116] F. Moerchen and D. Fradkin. Robust mining of time intervals with semi-interval partial order patterns. *Proceedings*, pages 315–326, 2010.

[117] P. C. Molenaar and C. G. Campbell. The new person-specific paradigm in psychology. *Current Directions in Psychological Science*, 18(2):112–117, 4 2009.

[118] P. C. M. Molenaar. A Manifesto on Psychology as Idiographic Science: Bringing the Person Back Into Scientific Psychology, This Time Forever. *Measurement: Interdisciplinary Research & Perspective*, 2(4):201–218, 10 2004.

[119] F. Mörchen and A. Ultsch. Efficient mining of understandable patterns from multivariate interval time series. *Data Mining and Knowledge Discovery*, 15(2):181–215, 2007.

[120] R. H. Morton, J. R. Fitz-Clarke, and E. W. Banister. Modeling human performance in running. *Journal of Applied Physiology*, 69(3), 9 1990.

[121] C. Muelder, B. Zhu, W. Chen, S. Member, H. Zhang, and K.-L. Ma. Visual Analysis of Cloud Computing Performance Using Behavioral Lines. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 22(6), 2016.

[122] E. K. Naeini, I. Azimi, A. M. Rahmani, P. Liljeberg, and N. Dutt. A Real-time PPG Quality Assessment Approach for Healthcare Internet-of-Things. In *Procedia Computer Science*, volume 151, pages 551–558. Elsevier B.V., 8 2019.

[123] N. Nag. Health State Estimation. 3 2020.

[124] N. Nag and R. Jain. A Navigational Approach to Health: Actionable Guidance for Improved Quality of Life. *Computer*, 52(4):12–20, 4 2019.

[125] N. Nag and R. Jain. A Navigational Approach to Health: Actionable Guidance for Improved Quality of Life. *Computer*, 52(4):12–20, 4 2019.

[126] N. Nag, V. Pandey, and R. Jain. Live personalized nutrition recommendation engine. In *MMHealth 2017 - Proceedings of the 2nd International Workshop on Multimedia for Personal Health and Health Care, co-located with MM 2017*, pages 61–68. Association for Computing Machinery, Inc, 10 2017.

[127] N. Nag, V. Pandey, H. Oh, and R. Jain. Cybernetic Health. 5 2017.

[128] N. Nag, V. Pandey, P. J. Putzel, H. Bhimaraju, S. Krishnan, and R. Jain. Cross-modal health state estimation. In *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, pages 1993–2002. Association for Computing Machinery, Inc, 10 2018.

[129] M. H. Namaki, Y. Wu, Q. Song, P. Lin, and T. Ge. Discovering Graph Temporal Association Rules. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017.

[130] L. Nardi, M. R. E. S. P. Landforms, , and u. 2015. Spatio-temporal patterns of channel changes in response to a major flood event: the case of the Magra River (central–northern Italy). *Wiley Online Library*, 40(3):326–339, 3 2014.

[131] M. Nauta, D. Bucur, and C. Seifert. Causal Discovery with Attention-Based Convolutional Neural Networks. *Machine Learning and Knowledge Extraction 2019, Vol. 1, Pages 312-340*, 1(1):312–340, 1 2019.

[132] P. D. Neufer. The Effect of Detraining and Reduced Training on the Physiological Adaptations to Aerobic Exercise Training. *Sports Medicine*, 8(5), 11 1989.

[133] P. H. Nguyen, C. Turkay, G. Andrienko, N. Andrienko, O. Thonnard, and J. Zouaoui. Understanding user behaviour through action sequences: From the usual to the unusual. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2838–2852, 9 2019.

[134] N. J. Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, 2014.

[135] H. Oh and R. Jain. From Multimedia Logs to Personal Chronicles. In *Proceedings of the 2017 ACM on Multimedia Conference - MM '17*, pages 881–889, New York, New York, USA, 2017. ACM Press.

[136] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. *Proceedings - International Conference on Data Engineering*, pages 412–421, 1998.

[137] V. Pandey, N. Nag, and R. Jain. Ubiquitous event mining to enhance personal health. In *UbiComp/ISWC 2018 - Adjunct Proceedings of the 2018 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2018 ACM International Symposium on Wearable Computers*, pages 676–679, New York, New York, USA, 2018. ACM Press.

[138] V. Pandey, N. Nag, and R. Jain. Continuous Health Interface Event Retrieval. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, New York, NY, USA, 6 2020. ACM.

[139] V. Pandey, D. Upadhyay, N. Nag, and R. Jain. Personalized user modelling for context-aware lifestyle recommendations to improve sleep. In *CEUR Workshop Proceedings*, volume 2684, 2020.

[140] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng. On Mining Anomalous Patterns in Road Traffic Streams. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7121 LNAI(PART 2):237–251, 12 2011.

[141] N. Patki, R. Wedge, and K. Veeramachaneni. The synthetic data vault. In *Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016*, pages 399–410. Institute of Electrical and Electronics Engineers Inc., 8 2016.

[142] J. Pearl. [Bayesian Analysis in Expert Systems]: Comment: Graphical Models, Causality and Intervention. Technical Report 3, 1993.

[143] J. Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3(0):96–146, 2009.

[144] J. Pearl. The Causal Foundations of Structural Equation Modeling. Technical report, 2012.

[145] J. Pearl. *The Book of Why*. Basic Books, 2018.

[146] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang. H-mine: Hyper-structure mining of frequent patterns in large databases. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 441–448, 2001.

[147] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, 11 2004.

[148] A. Perer and F. Wang. Frequence: Interactive Mining and Visualization of Temporal Frequent Event Sequences.

[149] J. Peters, D. Janzing, and B. Schölkopf. Causal Inference on Time Series using Restricted Structural Equation Models. *Advances in Neural Information Processing Systems*, 26, 2013.

[150] E. Pierce, A. Weltman, R. Seip, and D. Snead. Effects of Training Specificity on the Lactate Threshold and VO ¡sub¿2¡/sub¿ Peak. *International Journal of Sports Medicine*, 11(04), 8 1990.

[151] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. LifeLines: visualizing personal histories. *Conference on Human Factors in Computing Systems - Proceedings*, pages 221–227, 1996.

[152] R. Reiter. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*, volume 20. MIT Press, 2001.

[153] M.-A. Rizoiu, Y. Lee, S. Mishra, and L. Xie. A Tutorial on Hawkes Processes for Events in Social Media. *The Australian National University*, 8 2017.

[154] A. C. Robinson, D. J. Peuquet, S. Pezanowski, F. A. Hardisty, and B. Swedberg. Design and evaluation of a geovisual analytics system for uncovering patterns in spatio-temporal event data. *Cartography and Geographic Information Science*, 44(3):216–228, 5 2017.

[155] R. Ross, S. N. Blair, R. Arena, T. S. Church, J. P. Després, B. A. Franklin, W. L. Haskell, L. A. Kaminsky, B. D. Levine, C. J. Lavie, J. Myers, J. Niebauer, R. Sallis, S. S. Sawada, X. Sui, and U. Wisløff. *Importance of Assessing Cardiorespiratory Fitness in Clinical Practice: A Case for Fitness as a Clinical Vital Sign: A Scientific Statement from the American Heart Association*, volume 134. 2016.

[156] A. Rostami, V. Pandey, N. Nag, V. Wang, and R. Jain. Personal Food Model. *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4416–4424, 8 2020.

[157] A. Rostami, B. Xu, and R. Jain. Multimedia Food Logger. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4548–4549, New York, NY, USA, 8 2020. ACM.

[158] S. P. Rowland, J. E. Fitzgerald, T. Holme, J. Powell, and A. McGregor. What is the clinical value of mHealth for patients? *npj Digital Medicine*, 3(1), 12 2020.

[159] D. B. Rubin. Using Propensity Scores to Help Design Observational Studies: Application to the Tobacco Litigation. *Health Services and Outcomes Research Methodology 2001 2:3*, 2(3):169–188, 2001.

[160] D. B. Rubin and N. Thomas. Matching Using Estimated Propensity Scores: Relating Theory to Practice. *Biometrics*, 52(1):249, 3 1996.

[161] J. Runge, P. Nowack, M. Kretschmer, S. Flaxman, and D. Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5(11):eaau4996, 11 2019.

[162] K. Saha, A. E. Bayraktaroglu, A. T. Campbell, N. V. Chawla, M. De Choudhury, S. K. D'Mello, A. K. Dey, G. Gao, J. M. Gregg, K. Jagannath, G. Mark, G. J. Martinez, S. M. Mattingly, E. Moskal, A. Sirigiri, A. Striegel, and D. W. Yoo. Social media as a passive sensor in longitudinal studies of human behavior and wellbeing. *Conference on Human Factors in Computing Systems - Proceedings*, 5 2019.

[163] K. Saha and M. De Choudhury. Modeling stress with social media around incidents of gun violence on college campuses. *Proceedings of the ACM on Human-Computer Interaction*, 1(CSCW), 11 2017.

[164] K. Saha, R. Mulukutla, K. Nies, P. Robles-Granda, A. Sirigiri, D. W. Yoo, P. Audia, A. T. Campbell, N. V. Chawla, S. K. D'Mello, A. K. Dey, M. D. Reddy, K. Jiang, Q. Liu, G. Mark, E. Moskal, A. Striegel, M. De Choudhury, V. Das Swain, J. M. Gregg, T. Grover, S. Lin, G. J. Martinez, S. M. Mattingly, and S. Mirjafari. Imputing Missing Social Media Data Stream in Multisensor Studies of Human Behavior. *2019 8th International Conference on Affective Computing and Intelligent Interaction, ACII 2019*, 9 2019.

[165] K. Saha, J. Torous, S. K. Ernala, C. Rizuto, A. Stafford, and M. De Choudhury. A computational study of mental health awareness campaigns on social media. *Translational Behavioral Medicine*, 9(6):1197–1207, 11 2019.

[166] B. Saltin and P. O. Astrand. Maximal oxygen uptake in athletes. *Journal of Applied Physiology*, 23(3), 9 1967.

[167] C. Samuels. Sleep, Recovery, and Performance: The New Frontier in High-Performance Athletics. *Neurologic Clinics*, 26(1), 2 2008.

[168] J. Sarris, A. O'Neil, C. E. Coulson, I. Schweitzer, and M. Berk. Lifestyle medicine for depression, 4 2014.

[169] B. Sauer, M. A. Brookhart, J. A. Roy, and T. J. VanderWeele. Covariate Selection. 2013.

[170] H. Schäfer, S. Hors-Fraile, R. P. Karumur, A. Calero Valdez, A. Said, H. Torkamaan, T. Ulmer, and C. Trattner. Towards Health (Aware) Recommender Systems. In *Proceedings of the 2017 International Conference on Digital Health - DH '17*, pages 157–161, New York, New York, USA, 2017. ACM Press.

[171] N. J. Schork. Personalized medicine: Time for one-person trials, 4 2015.

[172] A. Seth. Granger causality. *Scholarpedia*, 2(7):1667, 2007.

[173] L. Shamseer, M. Sampson, C. Bukutu, C. H. Schmid, J. Nikles, R. Tate, B. C. Johnston, D. Zucker, W. R. Shadish, R. Kravitz, G. Guyatt, D. G. Altman, D. G. Altman, and S. Vohra. CONSORT extension for reporting N-of-1 trials (CENT) 2015: explanation and elaboration. *Journal of Clinical Epidemiology*, 76:18–46, 8 2016.

[174] T. Shochat. Impact of lifestyle and technology developments on sleep, 2012.

[175] R. R. Sillito and R. B. Fisher. Semi-supervised Learning for Anomalous Trajectory Detection.

[176] P. Spirtes, C. Glymour, and R. Scheines. Causation, Prediction, and Search. 81, 1993.

[177] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1057 LNCS:1–17, 1996.

[178] S. K. Sterba and D. J. Bauer. Matching method with theory in person-oriented developmental psychopathology research. *Development and Psychopathology*, 22(2):239–254, 5 2010.

[179] E. A. Stuart. Matching methods for causal inference: A review and a look forward. *Statistical science : a review journal of the Institute of Mathematical Statistics*, 25(1):1, 2 2010.

[180] V. Thambawita, S. Hicks, H. Borgli, S. A. Pettersen, H. K. Stensland, D. Jha, D. Johansen, H. D. Johansen, T. Kupka, T.-M. Grønli, P. M. Fredriksen, R. Eg, K. Hansen, S. Fagernes, A. Biorn-Hansen, D. T. D. Nguyen, H. L. Hammer, R. Jain, M. Riegler, and P. Halvorsen. PMData: A sports logging dataset. 2020.

[181] E. Thelen. Dynamic systems theory and the complexity of change, 2005.

[182] G. E. Vaillant. *Triumphs of Experience*. Harvard University Press, 10 2012.

[183] F. van Meer, L. Charbonnier, and P. A. M. Smeets. Food Decision-Making: Effects of Weight Status and Age, 8 2016.

[184] A. Vargha, L. R. Bergman, and S. Takács. Performing Cluster Analysis Within a Person-Oriented Context: Some Methods for Evaluating the Quality of Cluster Solutions. *Journal for Person-Oriented Research*, 2(1-2):78–86, 4 2016.

[185] F. Viégas, M. Wattenberg, J. Hebert, G. Borggaard, A. Cichowlas, J. Feinberg, J. Orwant, and C. R. Wren. Google+ Ripples: A Native Visualization of Information Flow. *Proceedings of the 22nd international conference on World Wide Web - WWW '13*.

[186] R. Vieira, S. McDonald, V. Araújo-Soares, F. F. Sniehotta, and R. Henderson. Dynamic modelling of n-of-1 data: powerful and flexible data analytics applied to individualised studies. *Health Psychology Review*, 11(3):222–234, 7 2017.

[187] R. Villafane, K. A. Hua, D. Tran, and B. Maulik. Knowledge Discovery from Series of Interval Events. *Journal of Intelligent Information Systems 2000 15:1*, 15(1):71–89, 2000.

[188] N. M. Villegas, C. Sánchez, J. Díaz-Cely, and G. Tamura. Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems*, 140:173–200, 1 2018.

[189] S. Vohra, L. Shamseer, M. Sampson, C. Bukutu, C. H. Schmid, R. Tate, J. Nikles, D. R. Zucker, R. Kravitz, G. Guyatt, D. G. Altman, and D. Moher. CONSORT extension for reporting N-of-1 trials (CENT) 2015 Statement. *BMJ (Clinical research ed.)*, 350:h1738, 5 2015.

[190] K. Vrotsou. Everyday mining: Exploring sequences in event-based data. 2010.

[191] P. Wang and A. F. Smeaton. Using visual lifelogs to automatically characterize everyday activities. *Information Sciences*, 230:147–161, 5 2013.

[192] P. Wang, L. Sun, A. F. Smeaton, C. Gurrin, and S. Yang. Computer Vision for Lifelogging: Characterizing Everyday Activities Based on Visual Semantics. In *Computer Vision For Assistive Healthcare*, pages 250–282. Elsevier Inc., 1 2018.

[193] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, B. Shneiderman, and S. Murphy. Aligning temporal data by sentinel events: Discovering patterns in Electronic Health Records. *Conference on Human Factors in Computing Systems - Proceedings*, pages 457–466, 2008.

[194] WangJianyong, HanJiawei, and LiChun. Frequent Closed Sequence Mining without Candidate Maintenance. *IEEE Transactions on Knowledge and Data Engineering*, 19(8):1042–1056, 8 2007.

[195] K. Wasserman, B. J. Whipp, S. N. Koyal, and W. L. Beaver. Anaerobic threshold and respiratory gas exchange during exercise. *Journal of Applied Physiology*, 35(2):236–243, 1973.

[196] H. A. Wenger and G. J. Bell. The Interactions of Intensity, Frequency and Duration of Exercise Training in Altering Cardiorespiratory Fitness. *Sports Medicine*, 3(5), 1986.

[197] U. Westermann and R. Jain. Toward a common event model for multimedia applications. *IEEE Multimedia*, 14(1):19–29, 1 2007.

[198] G. I. Wolf and M. De Groot. A Conceptual Framework for Personal Science. *Frontiers in Computer Science*, 0:21, 6 2020.

[199] K. Wongsuphasawat and D. H. Gotz. Outflow: Visualizing Patient Flow by Symptoms and Outcome.

[200] C. S. Wood, M. R. Thomas, J. Budd, T. P. Mashamba-Thompson, K. Herbst, D. Pillay, R. W. Peeling, A. M. Johnson, R. A. McKendry, and M. M. Stevens. Taking connected mobile-health diagnostics of infectious diseases to the field. *Nature*, 566(7745), 2 2019.

[201] S. Wright. Path Coefficients and Path Regressions: Alternative or Complementary Concepts? Technical Report 2, 1960.

[202] S. Y. Wu and Y. L. Chen. Mining nonambiguous temporal patterns for interval-based events. *IEEE Transactions on Knowledge and Data Engineering*, 19(6):742–758, 6 2007.

[203] S. Y. Wu and Y. L. Chen. Discovering hybrid temporal patterns from sequences consisting of point- and interval-based events. *Data & Knowledge Engineering*, 68(11):1309–1330, 11 2009.

[204] T. Wu, Y. Chen, and J. Han. Re-examination of interestingness measures in pattern mining: a unified framework. *Data Mining and Knowledge Discovery*, 21(3):371–397, 11 2010.

[205] L. Xie, H. Sundaram, and M. Campbell. Event mining in multimedia streams. *Proceedings of the IEEE*, 96(4):623–647, 2008.

[206] Y. F. Xing, Y. H. Xu, M. H. Shi, and Y. X. Lian. The impact of PM2.5 on the human respiratory system, 2016.

[207] H. Xu, M. Farajtabar, and H. Zha. Learning Granger Causality for Hawkes Processes. *33rd International Conference on Machine Learning, ICML 2016*, 4:2576–2588, 2 2016.

[208] H. Xu and H. Zha. A Dirichlet Mixture Model of Hawkes Processes for Event Sequence Clustering. *Advances in Neural Information Processing Systems*, 30, 2017.

[209] X. Yan, J. Han, and R. Afshar. CloSpan: Mining: Closed Sequential Patterns in Large Datasets. *Proceedings*, pages 166–177, 5 2003.

[210] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.

[211] M. J. Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning 2001 42:1*, 42(1):31–60, 2001.

[212] D. Zeevi, T. Korem, N. Zmora, D. Israeli, D. Rothschild, A. Weinberger, O. Ben-Yacov, D. Lador, T. Avnit-Sagi, M. Lotan-Pompan, J. Suez, J. A. Mahdi, E. Matot, G. Malka, N. Kosower, M. Rein, G. Zilberman-Schapira, L. Dohnalová, M. Pevsner-Fischer, R. Bikovsky, Z. Halpern, E. Elinav, and E. Segal. Personalized Nutrition by Prediction of Glycemic Responses. *Cell*, 163(5):1079–1094, 11 2015.

[213] E. Zgraggen, E. Zgraggen, S. M. Drucker, D. Fisher, and R. Deline. (s—qu)eries: Visual Regular Expressions for Querying and Exploring Event Sequences.

[214] W. Zhang, T. Panum, S. Jha, P. Chalasani, and D. Page. CAUSE: Learning Granger Causality from Event Sequences using Attribution Methods, 11 2020.

[215] Z. Zhang, D. Gotz, and A. Perer. Article Iterative cohort analysis and exploration.

[216] J. Zhao, N. Cao, Z. Wen, Y. Song, Y.-R. Lin, and C. Collins. #FluxFlow: Visual Analysis of Anomalous Information Spreading on Social Media. *IEEE Transactions on Visualization and Computer Graphics*, 2013.

[217] Z. Zheng, Y. Zhao, Z. Zuo, and L. Cao. Negative-GSP: An Efficient Method for Mining Negative Sequential Patterns.

Appendix A

Taste Space Modeling

A.1 Food Preference Space: Taste space

The taste of food items can be very complex. The taste sensory aspect has not been modeled as robustly as the visual and auditory senses, which have standard models such as the RGB color space model. In [156], the authors demonstrated how a unified and robust taste space model is required to create a preferential personal food model. They presented the US4B taste space, which includes six dimensions: umami, salty, sweet, sour, spicy, and bitter. However, that work was the initial step and did not provide any actual taste dataset or concrete approach to build such a dataset. There is currently no available method that could approximate the US4B values of a set of dishes using available resources. We provide a novel approach that uses the taste molecules to estimate a dish's US4B taste space using its ingredients. FlavorDB [47] is the only existing publicly available extensive data set on food taste that contains the list of taste molecules associated with each food item and a list of taste and smell attributes to each molecule. However, we could not derive the US4B values for dish items directly from flavorDB as there are a few limitations to this data set. This

dataset only has the information for ingredients and lacks the information for dish items and recipes. The dataset does not have any information about the intensity of the taste for different molecules. As shown in figure A.1, we start by counting the taste molecules associated with each element of the US4B taste model and create a taste vector for each ingredient item. Then we use a recipe data set containing a list of ingredient items for each dish and use the calculated US4B vector for ingredients in the previous step to calculate a US4B value for each dish based on adding the taste vectors of the ingredients. The taste values for dishes create the personal food model based on the food items in different situations. We used this approach to create a taste profile of 60 different dishes. We picked 20 dish recipes for each meal type: breakfast, lunch, dinner. Each meal type has ten dishes for a heavy meal option and ten for a light meal option. This data is fed to a randomized Markov-chain event generator, described in the following section, to create a randomized events log, including food events.

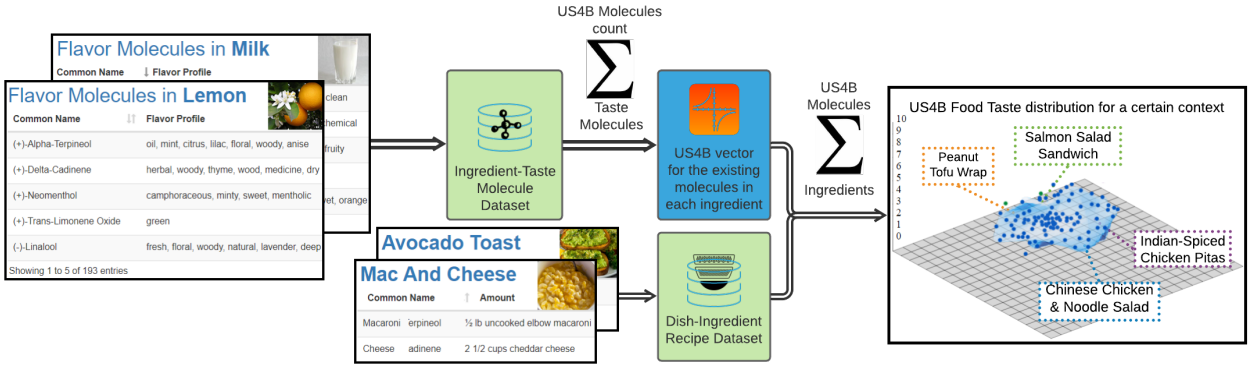

Figure A.1: Taste Space Generation: This figure illustrates our approach to map the food items to a corresponding US4B vector in the taste space using the taste information associated with the present molecules in each ingredient.

Appendix B

Event Mining System Implementation

We implemented the event pattern language (described in chapter 4) in a web-based event analysis dashboard, available at `https://theeventminer.com`. All users are given access to a public database in the dashboard and should contact the author if they are interested in using the dashboard for their analysis. In this chapter, we will discuss the user workflows for the dashboard and describe the implementation details for different components of the dashboard.

B.1 User Interface and Analysis workflow

The dashboard enables the analyst to explore and enrich any events data set using the event pattern language. We provide access to different event mining operations via an easy-to-use and understand user interface and provide a visual programming interface. The user interface consists of four tabs, each serving a distinct purpose.

1. **Data Selection Tab**: The analyst needs to provide the scope of the analysis by

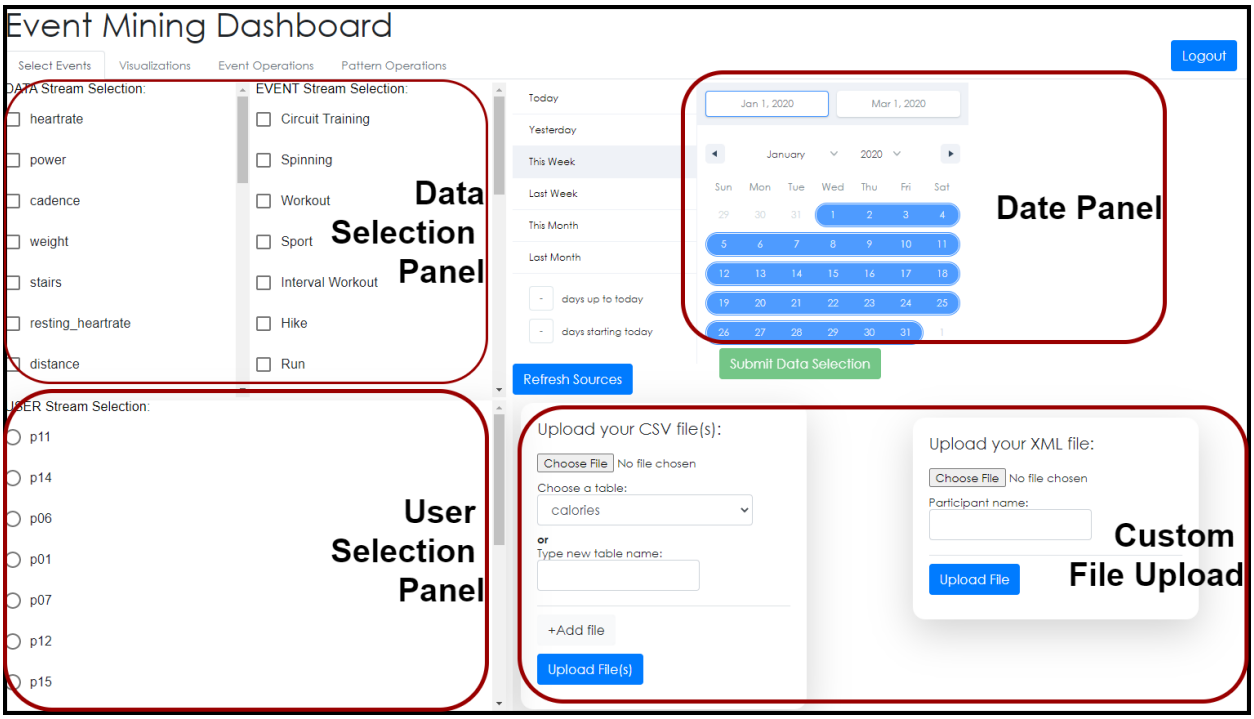

Figure B.1: Screenshot of the data selction tab from the **EventMining Dashboard**. The analyst can select the data and event streams for the concerned individual.

providing the events and data streams required for the analysis. They also need to provide the date range for which the analysis will be performed. The dashboard is designed to analyze one individual's data at a time; therefore, the analyst must select one user from the individuals in their data set. This tab is described in figure B.1.

2. **Event Visualization Tab**: The analyst can explore the selected events in the events visualization tab. This tab displays the events on a timeline and also on a tree ring visualization. The tree ring visualization displays event occurrences relative to the daily circadian cycle and identifies the regularity of events (such as sleep or exercise). The analysts can interact with these figures and zoom in and out as required. Figure B.2 shows a screenshot of this tab.

3. **Event Creation Tab**: This tab allows the analyst to create new events from existing events, data streams, or patterns. The data segmentation panel allows analysts to create events from data streams by performing simple range-based data segmentation

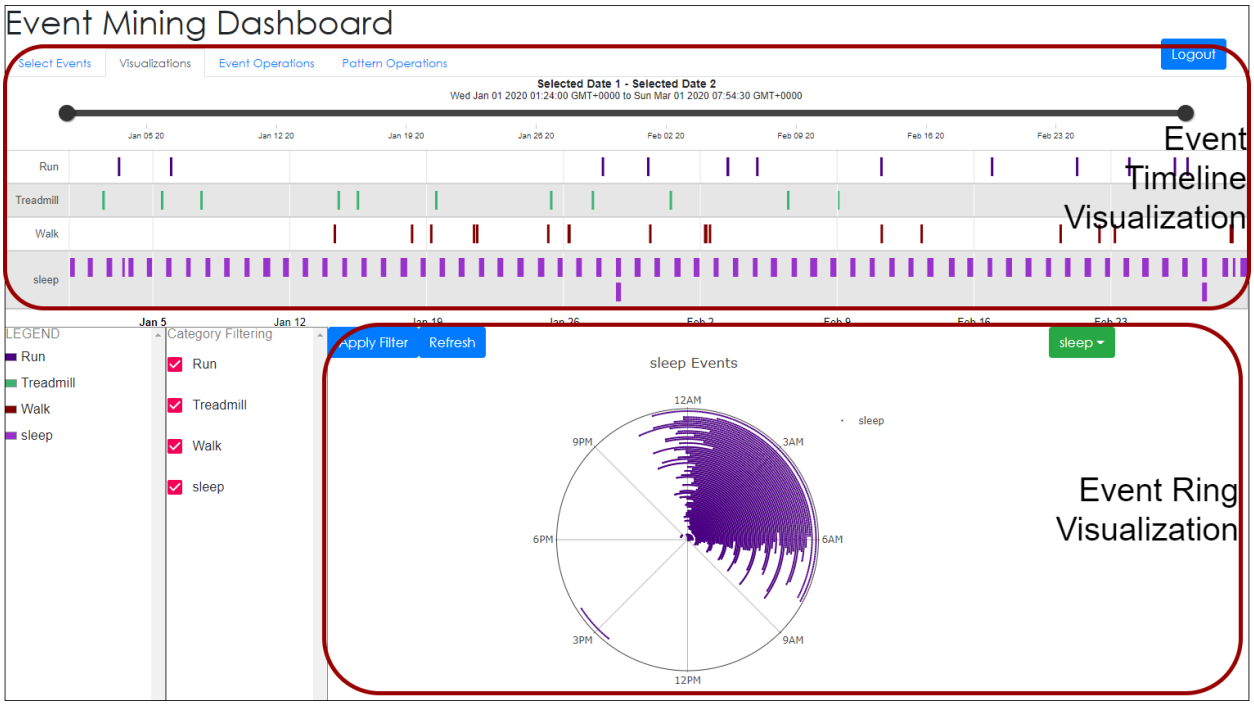

Figure B.2: Screenshot of the visualization tab from the **EventMining Dashboard**. This tab displays the selected events in a timeline and a tree ring visualization and allows the analyst to explore the event space.

operations (as described in chapter 4). The event combination panel allows analysts to create new events using operators such as AND and OR operations. The created events and their formulation is listed in this panel. The event filter panel allows the analyst to create new event streams from patterns by selecting constituent events that match a pattern. The event creation tab is shown in figure B.3.

4. **Pattern Creation Tab**: The analyst can create new patterns from the events in the database. This tab supports the concurrent and conditional sequential pattern operators. The patterns are stored in the database and can be visualized as a co-occurrence matrix as shown in figure B.4.

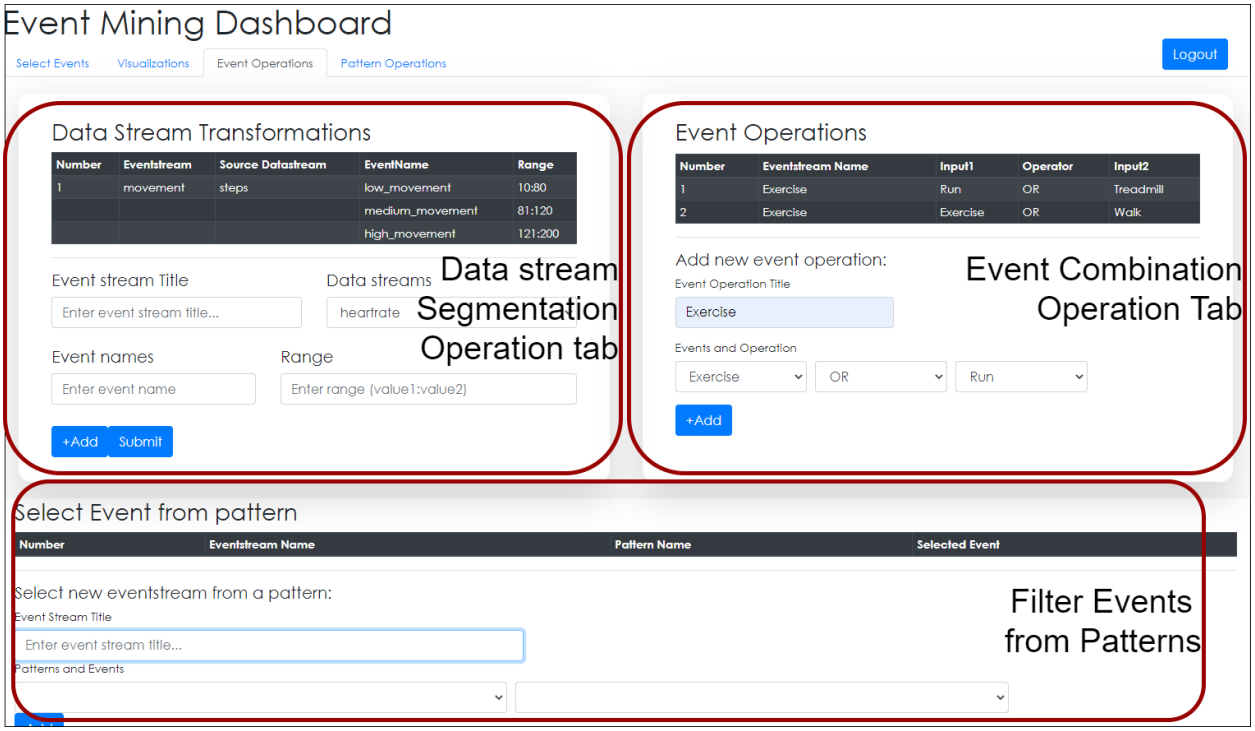

Figure B.3: Screenshot of the event creation tab from the **EventMining Dashboard**. This tab allows the analyst to create new events from data streams, existing events, and from created patterns.

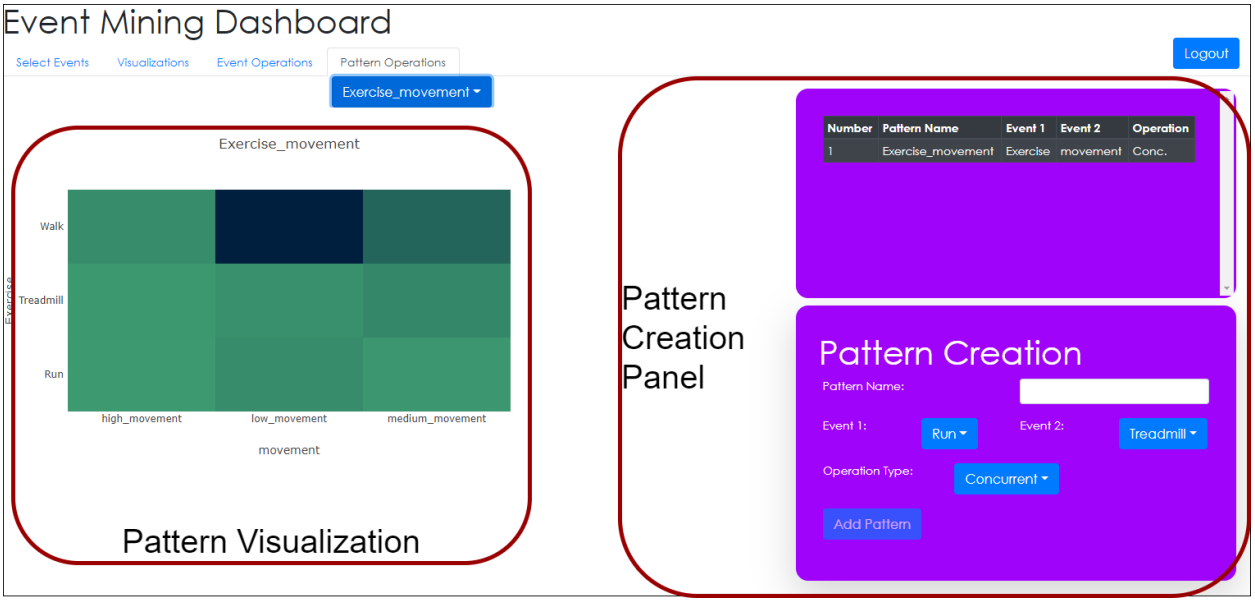

Figure B.4: Screenshot of the pattern creation tab from the **EventMining Dashboard**. This tab allows the analysts to create pairwise pattern from event streams selected and created for the analysis. The patterns are visualized in a co-occurrence matrix.

B.2 System Modules

The dashboard is developed primarily using a Flask backend and ReactJS front-end application. PostgreSQL was used as the primary database due to its flexibility in handling JSON fields in a relational schema. The front-end handles all user interaction and visualization functionalities, while the backend handles event mining operations, data fusion, and web request handling.

The backend has three major components 1) Flask API endpoints, 2) Event mining core libraries, and 3) Database layer for data fusion. We will describe components 2 and 3 in detail.

B.2.1 Data Fusion Module

As mentioned above, we utilize the PostgreSQL database to store the events and data streams for end-users. The events are typically stored in a single table called "event_stream" to create a singular log of events for a user, while every data stream is stored in a separate time indexed table. The schema for these tables are provided in a configuration file (metadata.json), as shown below.

The metadata file has two main fields: 1) *schema* contains a JSON string representation of different schema that can be used to create a new table. These represent the commonly used schema for events and data stream tables. 2) *Tables* contains a list of tables in the system and their corresponding schema (described in schema section). When the analyst wants to import data to a new table, they need to specify the new table name and schema (shown in fig. B.1) for the table. The data fusion layer utilizes this information to create the table and adds it to the metadata file. Thus, the analysts can extend the database quickly and include new data sources without directly interacting with the database.

```
{
"schema":{
    "datastream_schema_real": {
        "userID": "VARCHAR",
        "timestamp": "TIMESTAMPTZ",
        "value": "REAL",
        "unit": "VARCHAR",
        "source": "VARCHAR"
    },
    "datastream_schema_location": {
        "userID": "VARCHAR",
        "timestamp": "TIMESTAMPTZ",
        "value": "POINT",
        "unit": "VARCHAR",
        "source": "VARCHAR"
    },
    "datastream_schema_json": {
        "userID": "VARCHAR",
        "timestamp": "TIMESTAMPTZ",
        "value": "JSON",
        "unit": "VARCHAR",
        "source": "VARCHAR"
    },
    "datastream_schema_interval_real":{
        "userID": "VARCHAR",
```

```
        "start_time": "TIMESTAMPTZ",
        "end_time": "TIMESTAMPTZ",
        "timestamp": "TIMESTAMPTZ",
        "value": "REAL",
        "unit": "VARCHAR",
        "source": "VARCHAR"
    },
    "event_schema": {
        "userID": "VARCHAR",
        "event_type": "VARCHAR",
        "event_name": "VARCHAR",
        "start_time": "TIMESTAMPTZ",
        "end_time": "TIMESTAMPTZ",
        "parameters": "JSON",
        "datastreams": "JSON"
    },
    "simula_schema_resting_hr": {
        "userID": "VARCHAR",
        "timestamp": "TIMESTAMPTZ",
        "value": "REAL",
        "error": "REAL",
        "unit": "VARCHAR",
        "source": "VARCHAR"
    }
},
"tables":{
    "heartrate": "datastream_schema_real",
```

```
        "power": "datastream_schema_real",
        "cadence": "datastream_schema_real",
        "weight": "datastream_schema_real",
        "distance": "datastream_schema_real",
        "speed": "datastream_schema_real",
        "altitude": "datastream_schema_real",
        "slope": "datastream_schema_real",
        "temperature": "datastream_schema_real",
        "location":"datastream_schema_location",
        "pm25": "datastream_schema_real",
        "stress": "datastream_schema_real",
        "event_stream": "event_schema",
        "lightly_active_minutes": "datastream_schema_real",
        "moderately_active_minutes": "datastream_schema_real",
        "very_active_minutes": "datastream_schema_real",
        "resting_heart_rate": "datastream_schema_json",
        "sedentary_minutes": "datastream_schema_real",
        "steps": "datastream_schema_real",
        "time_in_heart_rate_zones": "datastream_schema_json"
        }
}
```

B.2.2 Core libraries

We have implemented the core event mining operations as python libraries, and these rely on three Python classes that represent data streams, event streams, and patterns, as discussed in chapter 4. These classes enable the analyst to perform event and pattern operations without

exposing the details of the event language. The Flask backend exposes these operations using APIs utilized for analysis via the front-end application (shown in the screenshots above). We will discuss the details of these three classes as they form the backbone of the dashboard.

Abstract Stream Class

An abstract stream class defines the basic temporal structure of both event and data streams. It describes the interface that different event streams and data streams utilize to interact with the database and perform operations.

Abstract Stream Class

```
from abc import ABC, abstractmethod
import pandas as pd

"""Contains the abstract class Stream which is the superclass for
data streams and event streams"""
class Stream(ABC):

    def __init__(self, user_id, relation_name, conn, time_range=None):

        self._user_id = user_id
        self._relation_name = relation_name
        self._db_conn = conn

        # Should convert the records from the db to dataframes
        self._read_obj = pd.DataFrame()
```

```
        # Should store these as dataframes and convert in the commit method
        # this object should only be set from a setter method
        self._commit_obj = pd.DataFrame()
        # Define query in this, select while creating a data stream
        # and insert while committing the changes
        self.query_object = {}


    def get_data(self):
        combined_data = self._read_obj.append(self._commit_obj,
            ignore_index=True)
        return combined_data.sort_values(by=["timestamp"])

    @abstractmethod
    def __str__(self):
        pass


    def _set_commit_obj(self, data, *params, **kwargs):
        if data is None or data.shape[0]==0:
            return
        self._commit_obj = data
        self._commit_obj['userID'] = self._user_id


    @abstractmethod
    def apply_transform(self, transform):
```

```
        pass


@abstractmethod
def access(self, **kwargs):
    """access the values in the specified time interval"""
    start_time = kwargs['start_time']
    end_time = kwargs['end_time']

    self.query_object['table'] = self._relation_name
    self.query_object['field'] = 'timestamp'
    self.query_object['operator'] = 'between'
    self.query_object['value'] = "'{}' AND '{}'".format(
        start_time.strftime("%Y-%m-%d %H:%M:%S %z"),
        end_time.strftime("%Y-%m-%d %H:%M:%S %z"))
    self.query_object['time_field'] = kwargs['time_field']
    self.query_object['userID'] = self._user_id

    cur = self._db_conn.select(self.query_object)
    results = cur.fetchall()
    cur.close()

    transformed_results = list(zip(*results))
    del results

    if len(transformed_results) == 0:
        self._read_obj = pd.DataFrame({
```

```
                                    'userID': [],
                                    'timestamp': [],
                                    'value': [],
                                    'unit': [],
                                    'source': []
                                   })
    return

# Schema is time, value, unit, source add start_time
# and end_time for interval streams
if len(transformed_results) == 5:
    temp_res = {
                'userID': transformed_results[0],
                'timestamp': transformed_results[1],
                'value': transformed_results[2],
                'unit': transformed_results[3],
                'source': transformed_results[4]
               }
else:
    temp_res = {
                'userID': transformed_results[0],
                'timestamp': transformed_results[3],
                'value': transformed_results[4],
                'unit': transformed_results[5],
                'source': transformed_results[6]
               }
self._read_obj = pd.DataFrame(temp_res)
```

```
def commit(self):
    """Writes the values in the internal storage to the db"""
    self.query_object = {'table': self._relation_name,
                         'num_rows': self._commit_obj.shape[0]}

    for f in self._commit_obj.columns:
        self.query_object[f] = self._commit_obj[f]

    try:
        self._db_conn.insert(self.query_object)
    except Exception as e:
        print("Error while commiting")

        print(e)
        raise e

    self.query_object = {}
    self._commit_obj = None
```

Data Stream Class

The data stream class extends the abstract stream class to incorporate datastream specific properties. The **apply_transform** method allows the analyst to define custom operations to be performed (such as segmentation, SAX, etc.) on the data stream that can be used to recognize new events.

Data Stream Class

```
import pandas as pd
import numpy as np
from datetime import datetime
import os
import json
from .abstract_stream import Stream


class DataStream(Stream):

    def __init__(self, user_id, relation_name, conn, time_range=None):
        super().__init__(user_id, relation_name, conn, time_range)

    def __str__(self):
        return "{}:{} \nAccessed values: \n{} \nValues to commit: \n{}"
        .format("Data stream", self._relation_name, self._read_obj.head(),
            self._commit_obj.head())

    def access(self, **kwargs):
        """access the values in the specified time interval
        arguments are  start_time, end_time,
        datetime strings in the format %Y-%m-%d %H:%M:%S %z,
        for eg 2019-07-08 11:08:43 -07"""

        kwargs['time_field'] = 'timestamp'
        super().access(**kwargs)
```

```
def apply_transform(self, transform, *params, **kwargs):
    """applies the transform function on the read data;
    parameters: relation_name for output data
    (future) type of output stream (event/data stream)"""
    if self._read_obj.shape[0] != 0:
        data = self._read_obj.copy(deep=True)
    else:
        data = self._commit_obj.copy(deep=True)

    assert (data is not None) and (data.shape[0] > 0),
        "Cannot transform an empty data stream"

    data_out = transform(data, *params, **kwargs)


    if kwargs.get('output', 'data') == 'data':
        new_stream = DataStream(self._user_id,
            kwargs.get('relation_name', 'test_ds'),
            self._db_conn)
    else:

        new_stream = EventStream(self._user_id, self._db_conn,
            kwargs.get("event_type",
                "temp_event_{}".format(self._relation_name)),
                kwargs.get("event_name",
                    "temp_event_{}".format(self._relation_name)),
```

```
                relation_name=kwargs.get('relation_name',
                    'event_stream'))
        new_stream._set_commit_obj(data_out,
            set_name=kwargs.get('set_name', False),
            debug=kwargs.get('debug', False))

        return new_stream
```

Event Stream Class

The event stream class also extends the abstract stream class; however, the event stream class has many special methods that are not required for the abstract and the datastream class. These include metadata generation methods such as *get_events* and *get_eventstream*. The class also provides a method for filtering the events by name and adding new events to an existing event stream object.

Event Stream Class

```
import pandas as pd
import numpy as np
from datetime import datetime
import os
import json
from .abstract_stream import Stream


class EventStream(Stream):
```

```
def __init__(self, user_id, conn, event_type=None, event_name=None,
    time_range=None, relation_name="event_stream"):
    super().__init__(user_id, relation_name, conn, time_range)
    self._event_type = event_type
    self._event_name = event_name

def __str__(self):
    return "{}:{}, {} \nAccessed values: \n{} \nValues to commit: \n{}".
        format("Event stream", self._event_type, self._event_name,
            self._read_obj.head(), self._commit_obj.head())

def access(self, **kwargs):
    """access the values in the specified time interval
    arguments are  start_time, end_time, datetime objects in the format
    %Y-%m-%d %H:%M:%S %z, for eg 2019-07-08 11:08:43 -07"""

    start_time = kwargs['start_time']
    end_time = kwargs['end_time']

    self.query_object['table'] = self._relation_name
    self.query_object['field'] = 'start_time'
    self.query_object['operator'] = 'between'
    self.query_object['value'] = "'{}' AND '{}' {} {}".
        format(start_time.strftime("%Y-%m-%d %H:%M:%S %z"),
            end_time.strftime("%Y-%m-%d %H:%M:%S %z"),
            "AND event_type = '{}'".format(self._event_type)
                if self._event_type is not None else "",
```

```
            "AND event_name = '{}'".format(self._event_name)
                if self._event_name is not None else "")
        self.query_object['time_field'] = 'start_time'
        self.query_object["userID"] = self._user_id

        cur = self._db_conn.select(self.query_object)
        results = cur.fetchall()
        cur.close()

        transformed_results = list(zip(*results))
        del results
        if len(transformed_results) == 0:
            self._read_obj = pd.DataFrame({
                    'userID':[],
                    'event_type': [],
                    'event_name': [],
                    'start_time': [],
                    'end_time': [],
                    'parameters': [],
                    'datastreams': []
                   })
            return

        temp_res = {
                    'userID': transformed_results[0],
                    'event_type': transformed_results[1],
                    'event_name': transformed_results[2],
```

```
                'start_time': transformed_results[3],
                'end_time': transformed_results[4],
                'parameters': transformed_results[5],
                'datastreams': transformed_results[6]
               }
    self._read_obj = pd.DataFrame(temp_res)


def _set_commit_obj(self, data, set_name = False, debug=False):
    if data is None or data.shape[0] == 0:
        return
    data['userID'] = self._user_id
    if debug:
        print(data)
        print("Event type: {}".format(self._event_type))
        print("Event name: {}".format(self._event_name))
    self._commit_obj = data
    if self._event_type is not None:
        self._commit_obj['event_type'] = self._event_type
    if self._event_name is not None and set_name:
        self._commit_obj['event_name'] = self._event_name


def add_new_eventstream(self, estream):
    """Add another eventstream's data to the current eventstream"""
    assert type(estream) == type(self),
        "Object of type EventStream required instead {} passed"
```

```
            .format(type(estream))

        if self._read_obj.shape[0] == 0:
            self._read_obj = estream._read_obj
        else:
            self._read_obj = self._read_obj.append(
                estream._read_obj, ignore_index=True)

        if self._commit_obj.shape[0] == 0:
            self._commit_obj = estream._commit_obj
        else:
            self._commit_obj = self._commit_obj.append(
                estream._commit_obj, ignore_index=True)

        if self._event_type is not None:
            self._commit_obj['event_type'] = self._event_type
        if self._event_name is not None:
            self._commit_obj['event_name'] = self._event_name
        return


    def apply_transform(self, transform, *params, **kwargs):
        """applies the transform function on the read data;
        parameters: relation_name for output data
        (future) type of output stream (event/data stream)"""

        # print("In event apply_transform")
```

```
        if self._read_obj.shape[0] != 0:
            data = self._read_obj.copy(deep=True)
        else:
            data = self._commit_obj.copy(deep=True)

        data_out = transform(data, *params, **kwargs)

        if kwargs.get('output', 'event') == 'data':
            new_stream = DataStream(self._user_id,
                kwargs.get('relation_name', 'test_ds'),
                self._db_conn)
        else:
            new_stream = EventStream(self._user_id,
                self._db_conn,
                event_type=kwargs.get("event_type",
                    "temp_event_{}".format(self._event_type)),
                event_name=kwargs.get("event_name", None),
                relation_name=kwargs.get('relation_name',
                    'event_stream'))
        new_stream._set_commit_obj(data_out,
            debug=kwargs.get('debug', False))

        return new_stream


def commit(self):
```

```
        self._commit_obj['parameters'] =
            self._commit_obj['parameters'].apply(
                lambda x: json.dumps(x))
        self._commit_obj['datastreams'] =
            self._commit_obj['datastreams'].apply(
                lambda x: json.dumps(x))

        self.query_object = {'table': self._relation_name,
                             'num_rows': self._commit_obj.shape[0]}

        for f in self._commit_obj.columns:
            self.query_object[f] = self._commit_obj[f]

        try:
            self._db_conn.insert(self.query_object)
        except Exception as e:
            print("Error while commiting")

            print(e)
            raise e

        self.query_object = {}
        self._commit_obj = None


    def filterByName(self, names):
        if self._read_obj.shape[0] > 0:
```

```
            self._read_obj =
                self._read_obj.loc[self._read_obj['event_name'].isin(names)]
        if self._commit_obj.shape[0] > 0:
            self._commit_obj =
                self._commit_obj.loc[self._commit_obj['event_name'].isin(name

        return


    def get_data(self, debug=False):
        if debug:
            print("in_get_data")
            print(self._read_obj)
            print(self._commit_obj)
        combined_data = self._read_obj.append(
                self._commit_obj, ignore_index=True)
        if debug:
            print(combined_data)
        return combined_data.sort_values(by=["start_time"])


    def get_events(self):
        data = self.get_data()
        events = data['event_name'].unique()
        return events.tolist()


    def get_eventstream(self):
        data = self.get_data()
        return data['event_type'].iloc[0]
```

Patterns Class

The patterns class provides the functionality of defining event patterns and computing them from the event stream class. We utilize a SQLite database to store temporary pattern results. The patterns class is implemented separately from pattern matching algorithms as the pattern data structure is independent of the pattern matching algorithm used. The algorithm to be used is decided based on parameter values during run-time.

Patterns Class

```
import json
import os
from datetime import datetime
import sys
from pathlib import Path

proj_loc = str(Path(__file__).parent.parent.resolve())
sys.path.append(proj_loc)
from pattern.utils import operators

import sqlite_scripts.add_df as sq
import sqlite3 as sq3

UNARY_OP = set(['not'])
BINARY_OP = set(['temporal_seq', 'sequential', 'concurrent'])


# Mapping of operator name to the implementation,
#this is used in the Pattern.run_pattern method
```

```
operator_mapping = {
    'temporal_seq': operators.temporal_sequential_operator,
    'concurrent': operators.co_occurence_operator
}

class Pattern(object):
    """Runs a pattern that matches across 2 event streams
       Assumes that the event streams are in sqlite db"""
    def __init__(self, name, db_loc, *params,
        db_name='session_data.db', **kwargs):
        """operator = delay or follows or co-occur
           params = [eventstream1, (eventstream2)]
           time_window = [t1, t2]"""


        self._name = name
        self._db_loc = db_loc
        self._db_name = db_name

        self._operator = kwargs.get("operator", None)

        if (self._operator is not None) and
            (self._operator not in set(list(UNARY_OP) + list(BINARY_OP))):
            raise ValueError("Operator should be one of {}".
                format(UNARY_OP + BINARY_OP))

        self._es1 = kwargs.get("eventstream_1", None)#params[0]
```

```
        self._es1_name = self._es1.get_eventstream()
            if self._es1 is not None else None

        self._es2 = kwargs.get("eventstream_2", None)
        self._es2_name = self._es1.get_eventstream()
            if self._es2 is not None else None

        self._time_window = kwargs.get("time_window", None)
        self._results = None


def run_pattern(self):
    """
    Runs the pattern specified by the operator along the
    specified eventstreams
    """
    print("Running pattern {}".format(self._name))
    if self._operator is None:
        print("Unspecified operator for the pattern")
        return None

    pattern_op = operator_mapping[self._operator]
    if self._operator in BINARY_OP:
        self._results = pattern_op(self._es1, self._es2,
            time_window=self._time_window)

    elif self._operator in UNARY_OP:
```

```
        self._results = pattern_op(self._es1,
            time_window=self._time_window)

    # results would be a table of event parameters
    # We would store this in a sqlite database

    sq.add_to_db(self._name, self._results,
        db_loc=self._db_loc, dbname=self._db_name)



def select(self, *params):
    """
    Select event stream(s) from the results of a pattern;
    Returns a dictionary of events data;
    key is the event name passed to the function
    """
    if self._results is None:
        self._results = sq.get_from_db(self._name, self._db_loc,
            dbname=self._db_name)
        if (self._results is None) or (self._results.shape[0] == 0):
            return None
    columns = self._results.columns
    return_set = {}
    for e in params:
        try:
            assert "{}_start_time".format(e) in columns and
```

```
                "{}_end_time".format(e) in columns and
                "{}_event_name".format(e) in columns and \
                "{}_event_type".format(e) in columns and
                "{}_parameters".format(e) in columns and
                "{}_datastreams".format(e) in columns
            except AssertionError as e:
                print("{}_not_in_pattern".format(e))
                continue
            temp_df = self._results[['userID', "{}_event_type".format(e),
                "{}_event_name".format(e), "{}_start_time".format(e),
                "{}_end_time".format(e), "{}_parameters".format(e),
                "{}_datastreams".format(e)]]
            temp_df.columns = ["userID", "event_type", "event_name",
            "start_time", "end_time", "parameters", "datastreams"]
            return_set[e] = temp_df

        return return_set



def cooccurrences(self, event1, event2):
    """
    Returns a cooccurrence matrix in long form ie
    (Event1, Event2, count)
    """
    if self._results is None:
        try:
            self._results = sq.get_from_db(self._name, self._db_loc)
```

```
        except Exception as e:
            print("Error while reading pattern {}".format(self._name))
            raise e

        if self._results.shape[0] == 0:
            return None
    group_cols = ["{}_event_name".format(event1),
        "{}_event_name".format(event2)]
    try:
        assert all([f in self._results.columns for f in group_cols])
    except AssertionError:
        print("Events not found: {}, {}".format(event1, event2))
        print(self._results.columns)
        return None

    aggregated_data = self._results.groupby(group_cols).size()

    aggregated_data = aggregated_data.reset_index()
    aggregated_data.columns = group_cols + ["count"]
    return aggregated_data
```

We have implemented two pattern operators 1) Concurrent and 2) Conditional Sequential. These operations are implemented as in-memory SQLite join queries.

Pattern Operations

```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import sqlite3 as sq3
from sqlite3 import InterfaceError
import os
import sys
import json



def temporal_sequential_operator(eventstream1, eventstream2,
    time_window=None):
    """Applies temporal sequence operator on two event streams"""
    es1_name = eventstream1.get_eventstream()
    es2_name = eventstream2.get_eventstream()

    es1 = eventstream1.get_data().sort_values(by='start_time')
    es2 = eventstream2.get_data().sort_values(by='start_time')

    if time_window is None:
        print("No time window provided")
        return None
    es1['interval_start'] = es1['start_time'] + \
```

```
    timedelta(hours=float(time_window[0]))
es1['interval_end'] = es1['end_time'] + \
    timedelta(hours=float(time_window[1]))

# convert json to json string for sqlite
es1['parameters'] = es1['parameters'].apply(lambda x: json.dumps(x))
es2['parameters'] = es2['parameters'].apply(lambda x: json.dumps(x))
es1['datastreams'] = es1['datastreams'].apply(lambda x: json.dumps(x))
es2['datastreams'] = es2['datastreams'].apply(lambda x: json.dumps(x))
# convert datetime to datetime string for sqlite
for f in ['start_time', 'end_time', 'timestamp',
    'interval_start', 'interval_end']:
    if f in es1.columns:
        es1[f] = es1[f].apply(lambda x:
            x.strftime("%Y-%m-%d %H:%M:%S %z"))
    if f in es2.columns:
        es2[f] = es2[f].apply(lambda x:
            x.strftime("%Y-%m-%d %H:%M:%S %z"))

# convert the dfs to in-memory sqlite tables,
# join the tables, then read as df
conn = sq3.connect(':memory:')
#write the tables
try:
    es1.to_sql(es1_name, conn, index=False)
    es2.to_sql(es2_name, conn, index=False)
except InterfaceError as e:
```

```
        print("Eventstream 1")
        print(es1.head())
        print(es1.dtypes)

        print("Eventstream 2")
        print(es2.head())
        print(es2.dtypes)
        raise e

qry = '''
    select
        {es1}.userID,
        {es1}.event_type {es1}_event_type,
        {es1}.event_name {es1}_event_name,
        {es1}.start_time {es1}_start_time,
        {es1}.end_time {es1}_end_time,
        {es1}.parameters {es1}_parameters,
        {es1}.datastreams {es1}_datastreams,

        {es2}.event_type {es2}_event_type,
        {es2}.event_name {es2}_event_name,
        {es2}.start_time {es2}_start_time,
        {es2}.end_time {es2}_end_time,
        {es2}.parameters {es2}_parameters,
        {es2}.datastreams {es2}_datastreams

    from
```

```
            {es1} join {es2} on
            {es2}.start_time between {es1}.interval_start
            and {es1}.interval_end
        '''.format(es1=es1_name, es2=es2_name)
    result_df = pd.read_sql_query(qry, conn)
    conn.close()

    for f in result_df.columns:
        if f.endswith('start_time') or f.endswith("end_time") or
            f.endswith("timestamp"):
            result_df[f] = result_df[f].apply(lambda x:
                datetime.strptime(x, "%Y-%m-%d %H:%M:%S.%z"))
        elif f.endswith("parameters") or f.endswith("datastreams"):
            result_df[f] = result_df[f].apply(lambda x: json.loads(x))

    return result_df


def co_occurence_operator(eventstream1, eventstream2, **kwargs):
    """Applies concurrency operator on two event streams"""
    es1_name = eventstream1.get_eventstream()
    es2_name = eventstream2.get_eventstream()

    es1 = eventstream1.get_data().sort_values(by='start_time')
    es2 = eventstream2.get_data().sort_values(by='start_time')

    # convert json to string for sqlite
```

```
es1['parameters'] = es1['parameters'].apply(lambda x: json.dumps(x))
es2['parameters'] = es2['parameters'].apply(lambda x: json.dumps(x))
es1['datastreams'] = es1['datastreams'].apply(lambda x: json.dumps(x))
es2['datastreams'] = es2['datastreams'].apply(lambda x: json.dumps(x))
# convert datetime to string for sqlite
for f in ['start_time', 'end_time', 'timestamp',
    'interval_start', 'interval_end']:
    if f in es1.columns:
        es1[f] = es1[f].apply(
            lambda x: x.strftime("%Y-%m-%d %H:%M:%S %z"))
    if f in es2.columns:
        es2[f] = es2[f].apply(
            lambda x: x.strftime("%Y-%m-%d %H:%M:%S %z"))

# convert the dfs to in-memory sqlite tables,
# join the tables, then read as df
conn = sq3.connect(':memory:')
#write the tables
es1.to_sql(es1_name, conn, index=False)
es2.to_sql(es2_name, conn, index=False)
# This query runs the operator using an in-memory sqlite table
qry = '''
    select
        {es1}.userID,
        {es1}.event_type {es1}_event_type,
        {es1}.event_name {es1}_event_name,
        {es1}.start_time {es1}_start_time,
```

```
            {es1}.end_time {es1}_end_time,
            {es1}.parameters {es1}_parameters,
            {es1}.datastreams {es1}_datastreams,

            {es2}.event_type {es2}_event_type,
            {es2}.event_name {es2}_event_name,
            {es2}.start_time {es2}_start_time,
            {es2}.end_time {es2}_end_time,
            {es2}.parameters {es2}_parameters,
            {es2}.datastreams {es2}_datastreams

        from
            {es1} join {es2} on
            {es2}.start_time between {es1}.start_time
            and {es1}.end_time or
            {es1}.start_time between {es2}.start_time
            and {es2}.end_time
        '''.format(es1=es1_name, es2=es2_name)
    result_df = pd.read_sql_query(qry, conn)
    conn.close()

    for f in result_df.columns:
        if f.endswith('start_time') or f.endswith("end_time") or
            f.endswith("timestamp"):
            result_df[f] = result_df[f].apply(
                lambda x: datetime.strptime(x, "%Y-%m-%d %H:%M:%S %z"))
        elif f.endswith("parameters") or f.endswith("datastreams"):
```

```
            result_df[f] = result_df[f].apply(lambda x: json.loads(x))

    return result_df
```