Towards a Library for the Analysis of Temporal Sequences

Thomas Guyet[1] (✉)[0000−0002−4909−5843] and Arnaud Duvermy[1,2][0009−0002−8509−5506]

[1] AIstroSight / Inria, HCL, UCBL
{thomas.guyet,arnaud.duvermy}@inria.fr
[2] Fondation APHP

Abstract. This article introduces TanaT, an open-source framework for temporal sequences analysis. Temporal sequences are made of complex events, described by qualitative and quantitative features, with a contiguous temporal footprint. Such kind of data are encountered in a wide range of applications (medicine, social science, traces analysis, education, etc.) and their analysis requires taking into account the longitudinality of the data. The proposed framework aims to empower data analysts with a coherent toolbox for handling such temporal sequences at all stages of the analysis process: data loading, data pre-processing and transformation, data analysis and data visualization. In this article, we introduce our framework, focusing on the distance-based clustering of temporal sequences. The complex nature of temporal entities requires versatility in defining distances between sequences and we highlight how TanaT addresses this challenge. https://gitlab.inria.fr/tanat/core/tanat

Keywords: Trajectory analysis · Data wrangling · Clustering · Metrics.

1 Introduction

With the continuous acquisition of increasingly large volumes of information, the analysis of temporal data has been attracting growing interest. In this work, we focus on temporal sequences (also referred to as *trajectories* [12], or as *complex event sequences*). Temporal sequences refer to event-based data where event occurrences are time-stamped and each event is described by attributes that may be either numerical or categorical. Such data arise in a variety of application domains, including computer logs analysis, life course analysis in the social sciences [5], career trajectory analysis [20], education [24], or patient care pathway analysis [21]. The temporal dimension contains rich information that is essential for understanding the dynamics of the underlying system or for supporting accurate and timely decision-making. In return, temporal sequences introduce additional challenges compared to classical tabular data or numerical time series data. Analyzing such data using tools originally designed for tabular formats is not always appropriate —either because the data can not be easily represented in tabular form, or because the assumptions of those analytical methods does

not suit temporal data. It thus appears necessary to develop a dedicated suite of tools specifically designed for the analysis of temporal sequences.

In various applications presented above, there is a shared interest in descriptive analysis, and especially in discovering groups of similar temporal sequences. This aims to uncover typical behaviors observed in a dataset. In this clustering problem, one important question is to define what "similar" means. Given the multifaceted nature of temporal sequences, clustering them —and designing appropriate distance metrics— has led to a wide range of approaches [7,10,17,21,26]. Similarly to what has been developed for time series [9,15,28], we strongly advocate for letting data scientists choose or build a dissimilarity metric that suits their own data and analysis needs. Our challenge is to propose a versatile framework to empower analysts with tools to fast prototyping and benchmark different metrics and clustering techniques.

In this paper, we introduce TanaT (*Temporal Analysis of Trajectories*), a Python library for the analysis of temporal sequences. The goal of this library is to provide a comprehensive set of generic tools for the analysis of temporal sequences. The remaining of the article is organized as follow. The next section will position TanaT with respect to state-of-the-art libraries dedicated to temporal data analysis. Then, we provide an overview of the library, detailing our data model and illustrating its current main functionalities. Section 6 introduces the concept of dissimilarity metrics in TanaT. Finally, we illustrate TanaT with a use case about healthcare trajectories and exemplifies the coding principles that enable its versatility and efficiency.

2 Related Works

The objective of this section is to present a comprehensive review of existing libraries and frameworks for temporal data analysis. In this review, we identified four different types of functionalities for a data analysis toolbox:

- Data preprocessing (a.k.a. data wrangling or ETL),
- Data visualization,
- Machine learning methods (classification, clustering, forecasting, etc.),
- Batch processing (e.g., pipelining, hyperparameter tuning, AutoML).

We believe that the first three functionalities must be integrated to establish a complete environment that covers most data analysis needs. The fourth functionality is beneficial for automating parts of the data analysis process, particularly for complex pipelines or the reproduction of similar studies.

To the best of our knowledge, there is no comprehensive Python package offering such a framework for temporal sequences.

2.1 Time Series vs Temporal Sequences

The literature generally distinguishes between two main types of temporal data: time series and temporal sequences. Time series (univariate or multivariate) refer

to regularly sampled numerical data. This type of data is already supported by a rich Python ecosystem of analytical libraries, including `tslearn` [28], `aeon` [15], `darts`, and `Greykite` [9], among others. These libraries organize and implement specific machine learning methods for time series. Preprocessing and visualization are ensured by standard numerical toolboxes (in Python: NumPy, SciPy, Matplotlib, etc.).

Nonetheless, temporal sequences are of a different nature. Temporal sequences refer to event-based data where event occurrences are time-stamped and each event is described by attributes that may be either numerical or categorical. While it is sometimes possible to transform temporal sequences into time series, this transformation is rarely trivial and requires dedicated tools to simplify the process. Moreover, temporal sequences often call for specific analyses that cannot be directly applied to time series representations. The objective of `TanaT` is to complement existing time series libraries by providing tools designed for data structured as temporal sequences, as well as tools to convert sequences into time series when needed.

2.2 Standard Machine Learning vs Tailored for Temporal Sequences

To date, there are only a few libraries that provide machine learning methods for this type of data. We identified in the literature two different approaches to support data scientists in their analysis of temporal sequences. The first type of approach is to flatten temporal sequences in order to prepare them for applying a standard machine learning algorithm. The other type of approach is to design new algorithms tailored for temporal sequences.

Tools designed for the first class of approaches provides mainly data wrangling tools. For instance, MEDS-Tab[3] or MEDS [1] are libraries designed for sequence "tabularization" or "flattening". They propose tools to transform, filter, aggregate, the raw data. For these libraries, the pipeline of transformations yields tabular datasets that can be analyzed with standard machine learning algorithms. ESGPT (Event Stream GPT) [14] is also a data pre-processing and modeling library for continuous-time sequences of complex events. Contrary to the previous approaches, it is designed specifically to prepare data for the application of transformer-like deep learning architectures. The pre-processing focuses more on the collection of the sequences (queries to filter out sequences from the collection) than modifying the content of the sequences. Its advantage is to build a memory-efficient deep learning representation of sequences. Similarily, TemporIA [25] is a machine learning-centric toolbox for temporal data in medicine. It proposes a data model that includes time series, static features, and also events. The typical workflow of TemporIA contains data transformations and sequential (deep/shallow) machine learning models for classification tasks.

These libraries have been designed mainly for analyzing Electronic Health Records (EHR) datasets. The only generic framework is Temporian.[4] It is a

[3] `https://meds-tab.readthedocs.io`
[4] `https://temporian.readthedocs.io`

Python library for simple and efficient preprocessing and feature engineering of temporal data in Python. Similarly to the pandas library for tabular datasets, it proposes a set of wrangling tools for temporal sequences at a low level and thus can be used to prepare datasets for machine learning. Temporian supports multivariate time-series, temporal sequences, and cross-source event streams, but again it is more developed for time series. Finally, we can mention that foundation models [30] are also a kind of preprocessing for temporal sequences. In our work, we do not explore this direction due to their lack of interpretability.

Some other libraries choose to integrate methods that analyze data represented as temporal sequences. The most prominent library is `TraMineR` [5]. It is developed in the `R` environment and is specifically tailored for data represented as sequences of states.[5] It has been designed for analyzing data in social science and is now used in a wide range of applications (health, education, etc.). The success of this library is notably due to 1) its flexibility in defining metrics between sequences and 2) its coherent collection of tools (visualization, description, metrics, and clustering). Unfortunately, the `R` environment is not always the most used by data scientists. A translation has been proposed in Stata [8], but none for the Python environment, which is becoming more and more popular. One ambition of `TanaT` is to transfer and enrich the functionalities of `TraMineR` in a Python environment. More specifically, we noticed that modeling temporal data as state sequences may be restrictive in practice, and extending the data model is a core proposal of `TanaT`.

Another prominent framework is developed around the notion of Multiple Aspect Trajectory (MAT) [23], which includes semantic and spatial dimensions. This representation is close to the concept of "semantic trajectories", which incorporate a semantic layer into the description of trajectories [19,2]. The MAT framework is both a data representation and a collection of tools, including clustering [22], for this kind of data. The framework uses semantic web technologies and provides web interfaces. Such interfaces are probably too restrictive for data scientists who explore a dataset, and they would probably prefer a programming interface offering more versatility.

2.3 Interactive Analysis vs Batch Execution

We also identified that there are different levels of interactivity with the data. On the one hand, the complexity of the data at hand increases the interest in visualizing and interacting with them. On the other hand, in a field such as healthcare, in which the analyses have to be reproduced, sometimes on massive data, interactivity is replaced by the possibility to create scripts describing a pipeline of processes that will be run in batch, without user interaction. Visualization and visual analytics are also important dimensions for the analysis of temporal sequences. ESeVis [32] is an event sequence visualization framework. The article presents a review of visualization approaches for event sequence data. They conclude on the opportunity to integrate visual analytics and process mining tools

[5] We will revisit this notion of *state* in Section 4.2.

	Data type		Machine learning methods				Interactivity		Language	
	Time series	Temporal sequences	Tabular ML	Tailored ML	Recurrent DL	Metrics	Visualization	Scripting	Python	Other
tslearn [28]	✓		☆	✓	☆	✓		✓	✓	
aeon [15]	✓		☆	✓	☆	✓	✓	✓	✓	
TraMineR [5]		✓		✓	✗	✓	✓			✓
MAT [23]	✓	✓			✓		✓		✓	✓
MEDS-Tab	✓	✓	✓						✓	
YAIB [29]		✓						✓	✓	
TemporAI [25]	✓	✓			✓			✓	✓	
MEDS [1]	✓	✓	✓					✓	✓	✓
ESGPT [14]	✓	✓			✓			✓	✓	✓
FIDDLE [27]		✓					✗	✓	✓	✓
ACES [31]		✓					✗	✓	✓	✓
Temporian	✓	✓					✗		✓	
ESeVis [32]		✓					✓	✗	✓	
EventAction [3]		✓					✓	✗		✓
EventFlow [16]		✓					✓	✗		✓
Coco [13]		✓					✓	✗		✓
TanaT (ours)		✓	✓	✓	☆	✓	✓	✓	✓	

Table 1. Characteristics of state-of-the-art frameworks for temporal data analysis. ✓ indicates functionalities provided by a framework, ✗ indicates those that seem not possible within a framework, and ☆ indicates functionalities obtained by planned connections with alternative frameworks. The last columns correspond to various environments such as Bash, R, or GUI.

in a unique framework. Such an approach of integrating visualization and data analytics leads to the proposal of visual analytics tools, such as EventAction [3], EventFlow [16], or Coco [13]. These approaches are easy to use for end-users; they allow conducting complex data analysis, such as clustering, without requiring specific skills in data science, but they are not versatile. The interfaces are designed to conduct one specific type of analysis and, more specifically, they do not allow the user to specify how to compare sequences. TanaT aims to include data visualization (instance and populational visualization) to support the user along their data analysis (in a notebook, for instance), but it is not made for interactive data analysis.

On the opposite side of interactivity, frameworks for creating processing pipelines have emerged in recent years, especially for EHR data. Indeed, one specificity of these data is their reusability for multiple studies. Tools to ease the reusability or reproducibility of studies make data scientists more efficient. ACES [31] has been proposed with these objectives and it defines a flexible task configuration language that enables defining diverse sets of prediction tasks on an event-stream dataset. YAIB [29] provides a framework for conducting clinical machine learning experiments on Intensive Care Unit (ICU) EHR data. It proposes recurrent neural network architectures to analyze sequential data. MEDS [1] also proposes a framework to run sequential machine learning models on various EHR datasets through a metadata schema. It also proposes to track the provenance of transformations to inform the trained models. Nonetheless, the preprocessing in YAIB or MEDS are not explicitly described in their pipeline. TemporAI [25] also proposed to implement workflows, but it seems less flexible than the previous methods. FIDDLE [27] is an interesting framework that allows making (low-level) preprocessing to prepare temporal data for machine learning tasks. The main difference with TanaT is the modeling of temporal sequences

as tensors. This representation discretizes time and thus loses some precious information about durations.

To conclude this section, Table 1 summarizes some properties of the main frameworks we reviewed.

3 Overview of TanaT

TanaT is a flexible toolbox for handling various types of temporal sequences. It is an open-source project available on gitlab (`https://gitlab.inria.fr/tanat/core/tanat`). Figure 1 illustrates the organization of the library into modules and their connections with each other and with external standard libraries. The library currently comprises four main modules:

- **Temporal sequence representation** is the core of the library. It implements classes to represent temporal sequences and trajectories (cf. section 4). Our classes are built on top of `pandas` for the sake of efficiency and versatility. This module comprises input-output routines: the loading of external raw data is possible from Pandas dataframes or through requests to a database; we provide several formats to export the internal representation for alternative machine learning libraries: tensors, graphs, tabular and time series.[6]
- **Data preprocessing** comprises a set of low-level preprocessing tools — selection of individuals, filtering of events, (static) feature engineering, etc. These tools transform the trajectory pools into other trajectory pools.
- **Visualization**: this module offers visualization of trajectories at individual or population levels. Visualizations are based on the Matplotlib framework.
- **Metrics**/**Clustering**: these modules are dedicated to the clustering of trajectories. TanaT allows specifying versatile dissimilarity metrics and implements clustering methods, primarily derived from `sklearn`, to cluster sequences or trajectories using the metrics defined by the user.

All preprocessing steps, including metrics definition and data analytics, can be used as an API to create interactive computational notebooks or in Python programs. At the same time, we also provide a domain-specific language to represent pipelines of processes (in YAML format). This enables TanaT to cover all the needs of a data scientist. On the one hand, the API enables the exploration of data by quickly writing some preprocessing steps, visualizing intermediate results, evaluating algorithms, etc. On the other hand, once the pipeline of processing from raw data to the data analysis results has been clearly defined, it can be reified into a workflow (coded as a YAML file) and executed from the command line. This enables the data scientist to share the processing (transparency) and to apply it to alternative databases, for instance (reproducibility).

[6] More specifically, TanaT does not aim to reimplement deep learning models (e.g., recurrent neural networks), but it provides the possibility to transform the trajectories into a format to apply the user's favorite models.

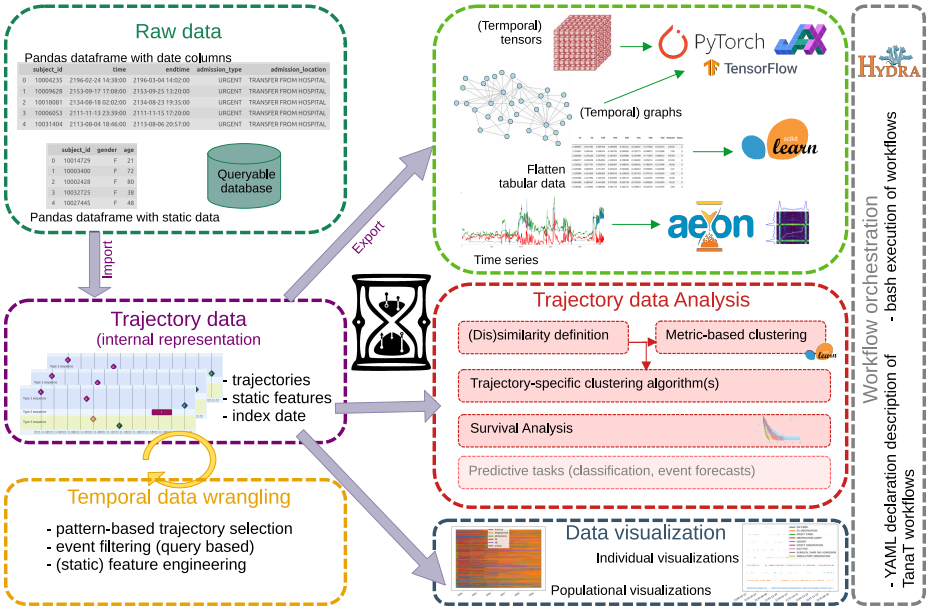

Fig. 1. Illustration of the TanaT ecosystem: TanaT modules and connections with external libraries. The core of TanaT is the representation of trajectories (purple square), which can be imported from structured data (Pandas dataframes or databases) and exported in other classical machine learning formats (green frame). Libraries for analyzing trajectories comprise the library of data wrangling (yellow square) to transform trajectories; different trajectory analysis tools: clustering, survival analysis (prediction tasks are for future developments). Analyses are complemented with a data visualization module (in dark blue). Finally, the grey square illustrates the transversal capabilities to represent an analysis pipeline within a YAML file to be executed in batch.

4 TanaT Data Model

We introduce the core data structures of TanaT. The library is designed to be extensible, and other modules/data structures could be added in the future. This section concludes with the conceptual data model (see Figure 5).

4.1 Entities, Sequences, and Trajectories

The TanaT library distinguishes three different types of temporal data structures: temporal *entities* (the atomic unit), *sequences*, and *trajectories*.

The fundamental structure for representing temporal data is the *temporal entity*.[7] An entity describes something that has occurred for an individual and has a temporal extent.

[7] We use the term "temporal entity" rather than "event" since the latter is reserved in our framework for a specific subtype of temporal entity, i.e., those with zero duration.

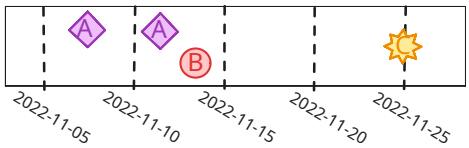

Fig. 2. Illustration of a sequence of events. Each colored-shape represents an event. The event type of an occurrence is represented by a symbolic feature (letters A, B or C). Each event type as a color. The y position has no specific meaning here.

- The nature of an entity is described as a vector of features (qualitative or quantitative). At least one descriptive feature must be defined. In the simplest case, an entity is represented by a symbolic value drawn from a vocabulary, i.e., a categorical feature.
- The temporal extent of an entity is defined by a timestamp or a time interval. In TanaT's temporal model, all entities are timestamped using calendar dates and/or times. Time representation can be integers, floating-point numbers or date-time format. In addition, there is no fixed time scale: TanaT can handle time resolutions ranging from nanoseconds to geological eras, depending on the dataset. In our motivating examples, such as care trajectories, the granularity is typically daily.

A *sequence* is the primary structure for representing collections of entities. All entities within a sequence must share the same type of temporal extent (e.g., point events, intervals, or states) and the same set of descriptive features.

Example 1. Figure 2 illustrates a *sequence* composed of four instantaneous entities. Each entity type is denoted by a symbol (e.g., A, B, C, D). All entities share a single categorical feature. They are displayed along a timeline to highlight their temporal positions.

The sequence may contain multiple entities with the same feature value (e.g., A), and distinct entities may occur at the same timestamp. The only constraint is that two identical entities (with the same descriptive features) cannot occur at the exact same time.

A *trajectory* is a more complex structure that models a multi-sequence object for a single individual. Each component sequence may differ in terms of features and temporal extent. This allows trajectories to capture heterogeneous types of temporal information.

For example, a trip might be represented as a traveler trajectory comprising:

1. weather conditions (a state sequence, see next section for details),
2. visited cities (an interval sequence), and
3. points of interest visited (an event sequence).

Example 2. Figure 3 shows a trajectory consisting of three sequences of different types: two event sequences and one interval sequence. Each sequence has a distinct feature set.

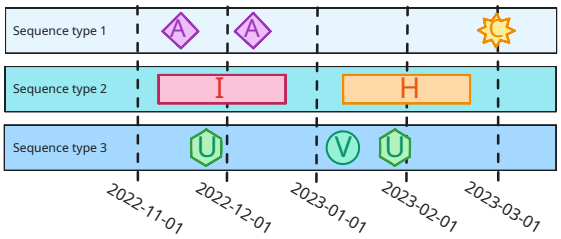

Fig. 3. Illustration of a trajectory with the three types of sequences.

A trajectory can also include *static features* (i.e., attributes not associated with a temporal extent). In the context of care pathways, static features may include birth date, gender, chronic conditions, etc. These are represented as key-value pairs.

4.2 Types of Temporal Support

TanaT currently supports three main types of temporal sequences, as illustrated in Figure 4:

- **Event sequences**: entities are point-based in time (e.g., a medical appointment).
- **Interval sequences**: entities span a duration (e.g., an hospital stay). Intervals may overlap, and gaps between successive intervals are allowed. Degenerate intervals (with identical start and end times) are permitted.
- **State sequences**: entities also span time intervals but are mutually exclusive. There is exactly one state active at any point in time. This imposes contiguity between intervals and enforces that two successive intervals have distinct feature values.

The notion of *state* was introduced by *TraMineR* [5], and we implement this representation primarily for compatibility with analyses typically conducted by users of this library.

Timestamps are internally represented as dates. We have added the possibility to define an index date, i.e., a reference point in time specific to each individual, that allows the temporal position to be expressed as a delay relative to this reference. An index date is typically defined as the first occurrence of a specific event.

4.3 Object Collections

In TanaT, a collection of sequences or trajectories is referred to as a *pool*. Each object in a pool represents an individual (a patient, a traveler, a student, etc.), and all individuals are described using the same structural schema, i.e., same types of sequences, temporal support types, and feature sets, both temporal and static.

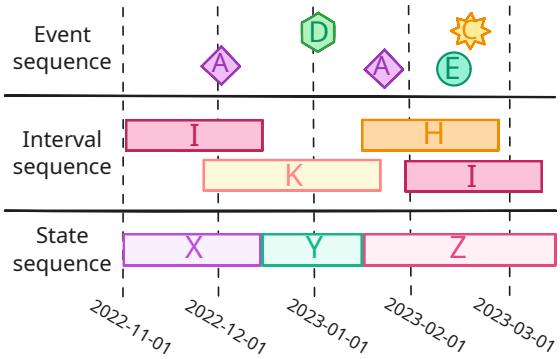

Fig. 4. This figure illustrates the various types of temporal supports modeled in TanaT. From bottom to top: events, intervals, and states.

The *pool* object is intended for efficient operations over collections of trajectories or sequences, enabling optimized computations and collective analyses.

Figure 5 provides a conceptual overview of the data organization in the TanaT library (except static features). The central backbone highlights the core trio — trajectory, sequence, and entity— which is instantiated (on the right) into the various types of sequences. On the left, the concept of a *pool* is illustrated, while the bottom part details the components of an entity, including *features* (defined at the sequence type level) and the *temporal extent*.

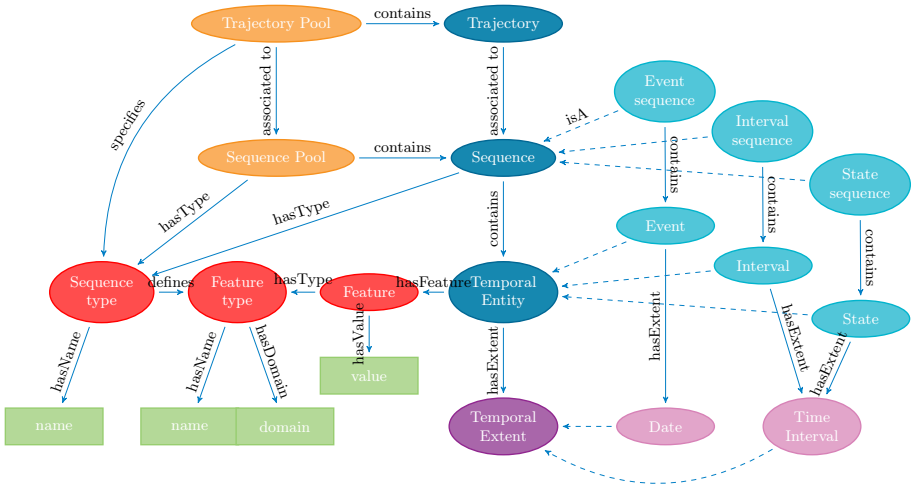

Fig. 5. Conceptual data model in TanaT (see text for explanation).

5 Temporal Data Wrangling

Raw data is rarely directly usable for analysis. Cleaning and structuring such data is essential to obtain meaningful and relevant results. Whereas alternative libraries typically require pre-processing before representing data as sequences, the philosophy behind TanaT is to represent data directly as temporal sequences and to manipulate them using TanaT's dedicated wrangling tools, which are designed to be both expressive and intuitive.

For instance, in the context of an epidemiological study, a standard procedure involves identifying the subset of individuals to include in the study and then extracting their characteristics for descriptive analysis. This description may take the form of a trajectory composed of events relevant to the analysis, or of static features derived from longitudinal data.

TanaT supports these preprocessing steps through generic tools for selecting individuals, labeling séquence, and filtering entities.[8] Sequence labeling consists in assigning static information to individuals based on all their available data, including sequences. This notably enables feature engineering to apply machine learning methods to tabular data (i.e., static data across the entire population).

These tools are designed to be both generic and flexible, allowing analysts to configure filtering and selection criteria using customizable patterns.

The current pattern language supports the specification of the occurrence of a given event. The library is designed to be extensible and to accommodate other pattern semantics. In particular, we plan to introduce patterns inspired by the notion of chronicles [6], enabling selection capabilities that are not possible with traditional dataframe-based representations.

6 Clustering Trajectories

This section presents the modules develop to conduct metric-based clustering of trajectories. In many data analysis, clustering is an important step to describe the data at hand. In the case of trajectory data, we would like to propose a versatile framework, similar to those proposed for time series [15,28]. This implies first to give the user the possibility to specify a metric.[9]

TanaT provides a broad collection of metrics that can be easily tested and compared to identify the most appropriate choice for a given task. The hierarchical representation of sequential data in TanaT has led us to define metrics for the three main types of objects being manipulated.

– First, defining a metric between entities enables the quantification of differences between their attributes. When attributes are numerical, an Euclidean

[8] The construction of new events is a planned feature for future versions of the library.

[9] It is important to note that the term "metric", as used in the library, refers to a dissimilarity measure, which does not necessarily satisfy all the mathematical properties of a distance. Similarly, the term "metric space" refers to a space equipped with such a measure.This terminology differs from the strict mathematical usage but is commonly accepted in time series libraries.

distance can be used; for categorical attributes, Hamming distance or a cost matrix is more appropriate. `TanaT` allows users to define custom entity-level metrics to adapt to specific requirements.
- The dissimilarity between sequence is obtained by aggregating pairwise entities dissimilarities. For sequences, a wide range of solution to aligning sequences of temporal entities. In the current version, we have focused on implementing the one available in `TraMineR` and similar to the ones encountered for time series. Specifically, we have implemented the following: LCP (Longest Common Prefix), LCSS (Longest Common Subsequence), an adaptation of the Euclidean distance, the χ^2 distance, and edit distances [18,11]. We have also included DTW and drop-DTW [4] for sequences.
- Finally, trajectory-level metrics are defined as aggregations of the distances computed on the sequences composing a trajectory (e.g., sum or minimum, possibly weighted). In practice, comparing two trajectories requires defining a metric for each sequence type and an aggregation function.

This general vision of dissimilarity between trajectories or sequences is highly versatile and allows users to fine-tune it according to the specific characteristics of each trajectory type. It is worth noting that entities does not required to be transformed into numerical form. For instance, it is possible to define a cost-matrix between entities attributes to evaluate the dissimilarities between entities with symbolic attributes.

One important issue is the computational efficiency of dissimilarity calculations. To address this, `TanaT` implements a hidden caching system: if the dissimilarity between two individuals is requested multiple times, the cached value is reused for subsequent calls. This internal mechanism is fully transparent to the data scientist.

Once a dissimilarity metric has been defined, it can be used within clustering algorithms such as hierarchical clustering or k-means. It is the responsibility of the data scientist to choose an appropriate algorithm that aligns with the properties of the dissimilarity measure.

7 Use Case Example

In this section, we illustrate a common use case: clustering patient care trajectories. In this simple example, we work with sequences of patient admissions. Each admission is characterized by its type and the hospital department to which the patient was admitted. These admissions are represented as events (arrival dates in a department).

The code below shows how to load the data from a Pandas dataframe that contains at least three columns: one for the dates (`admittime`), one for the patient identifier (`subject_id`), and one or more columns describing event attributes. The user provides the necessary information to identify which columns to use in the dataset. The summary function displays a few basic statistics to verify that the data was loaded correctly. As a result, a sequence *pool* is directly constructed.

```
from tanat.sequence import EventSequencePool, EventSequenceSettings
from tanat.dataset import access
admissions = access("mimic_admissions") # admissions is a pandas dataframe

adm_settings = EventSequencePoolSettings(
   id_column="subject_id",
   time_column="admittime",
   entity_features=["admission_type"]
)

seqpool = EventSequencePool(data=admissions, settings=adm_settings)
print(seqpool.summarize())
```

Once the data is loaded, we can define metrics to specify how two sequences should be compared. To do so, we must define a metric both at the entity level and at the sequence level. In the code below, we define a Hamming distance over the `admission_type` attribute to compare individual temporal entities. The sequences are then compared using a "Euclidean" distance[10] based on the entity metric defined earlier.

```
from tanat.metric.entity import HammingEntityMetric,
                                HammingEntityMetricSettings
from tanat.metric.sequence import LinearPairwiseSequenceMetric,
                                  LinearPairwiseSequenceMetricSettings

## -- Create the entity metric (with default settings)
hamming_settings = HammingEntityMetricSettings(
  default_value=0.0
)
hamming_metric = HammingEntityMetric(hamming_settings)

## -- Create the sequence metric
linear_settings = LinearPairwiseSequenceMetricSettings(
  entity_metric=hamming_metric, agg_fun="sum"
)
sequence_metric = LinearPairwiseSequenceMetric(linear_settings)
```

This metric can then be used to compare two sequences from the pool:

```
# -- Get the sequences
seq1 = seqpool[18253212]
seq2 = seqpool[11540283]

# -- Compute the metric between the two sequences
val = sequence_metric(seq1, seq2)
```

[10] This sequence metric sums the distances between pairs of events taken sequentially in the two sequences. If the sequences have different lengths, the additional events in the longer sequence are ignored (`default_value=0.0`).

Finally, this metric can be used within a clustering algorithm. Here, we perform agglomerative hierarchical clustering based on the admission data.

```
from tanat.clustering import HierarchicalClusterer,
                             HierarchicalClustererSettings

hc_settings = HierarchicalClustererSettings(metric=sequence_metric,
                                            n_clusters=5)
clusterer = HierarchicalClusterer(settings=hc_settings)
clusterer.fit(seqpool)
```

It is worth noting that all classes are parameterized through a single settings object. This indirect parameterization strategy, adopted in TanaT, allows for managing the full complexity of certain parameter sets and supports representing the processing configuration in an external file (e.g., YAML), which can then be used to instantiate objects. This design enables a dual execution mode: either interactively or in batch.

An extended notebook, covering additional modules of the library, is provided as supplementary material.

8 Conclusion and Future Work

In this paper, we introduced TanaT, A first Python library dedicated to the analysis of sequential data. The library is designed to be flexible and currently includes functionality for sequence representation, visualization, metric definition, and clustering. Initial experiments show that the approach is highly flexible, though performance in terms of computation time still needs improvement.

Future development will focus on improving the performance of metric computations and adding complementary features such as classification, sequence manipulation, and gradual integration of additional metrics.

Acknowledgments. The authors would like to thank the APHP Foundation for funding the AIRacles Chair, within which this project was initiated and is currently being developed.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Arnrich, B., Choi, E., Fries, J.A., McDermott, M.B., Oh, J., Pollard, T., Shah, N., Steinberg, E., Wornow, M., van de Water, R.: Medical event data standard (meds): Facilitating machine learning for health. In: ICLR 2024 Workshop on Learning from Time Series For Health. pp. 03–08 (2024)
2. Cayèré, C., Sallaberry, C., Faucher, C., Bessagnet, M.N., Roose, P., Masson, M., Richard, J.: Multi-level and multiple aspect semantic trajectory model: application to the tourism domain. ISPRS International Journal of Geo-Information **10**(9), 592 (2021)

3. Du, F., Plaisant, C., Spring, N., Crowley, K., Shneiderman, B.: Eventaction: A visual analytics approach to explainable recommendation for event sequences. ACM Transactions on Interactive Intelligent Systems (TiiS) **9**(4), 1–31 (2019)
4. Dvornik, M., Hadji, I., Derpanis, K.G., Garg, A., Jepson, A.: Drop-DTW: Aligning common signal between sequences while dropping outliers. Advances in Neural Information Processing Systems (NIPS) **34**, 13782–13793 (2021)
5. Gabadinho, A., Ritschard, G., Müller, N.S., Studer, M.: Analyzing and visualizing state sequences in R with TraMineR. Journal of Statistical Software **40**(4), 10 (2011)
6. Guyet, T., Besnard, P.: Chronicles: Formalization of a Temporal Model. Springer International Publishing (2023)
7. Guyet, T., Pinson, P., Gesny, E.: Clustering of timed sequences – application to the analysis of care pathways. Data and Knowledge Engineering **156**, 102401 (2025)
8. Halpin, B.: SADI: Sequence analysis tools for stata. The Stata Journal **17**(3), 546–572 (2017)
9. Hosseini, R., Chen, A., Yang, K., Patra, S., Su, Y., Al Orjany, S.E., Tang, S., Ahammad, P.: Greykite: deploying flexible forecasting at scale at linkedin. In: Proceedings of the 28th Conference on Knowledge Discovery and Data Mining (SIGKDD). pp. 3007–3017 (2022)
10. Le, M., Nauck, D., Gabrys, B., Martin, T.: Sequential clustering for event sequences and its impact on next process step prediction. In: Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU). pp. 168–178 (2014)
11. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Proceedings of the Soviet physics doklady (1966)
12. Levy, R.: Regard sociologique sur les parcours de vie. Regards pluriels sur l'approche biographique: entre discipline et indiscipline **95**, 1–20 (2001)
13. Malik, S., Shneiderman, B., Du, F., Plaisant, C., Bjarnadottir, M.: High-volume hypothesis testing: Systematic exploration of event sequence comparisons. ACM Transactions on Interactive Intelligent Systems (TiiS) **6**(1), 1–23 (2016)
14. McDermott, M., Nestor, B., Argaw, P., Kohane, I.S.: Event stream gpt: a data pre-processing and modeling library for generative, pre-trained transformers over continuous-time sequences of complex events. Advances in Neural Information Processing Systems **36**, 24322–24334 (2023)
15. Middlehurst, M., Ismail-Fawaz, A., Guillaume, A., Holder, C., Guijo-Rubio, D., Bulatova, G., Tsaprounis, L., Mentel, L., Walter, M., Schäfer, P., et al.: aeon: a python toolkit for learning from time series. Journal of Machine Learning Research **25**(289), 1–10 (2024)
16. Monroe, M., Lan, R., Lee, H., Plaisant, C., Shneiderman, B.: Temporal event sequence simplification. IEEE transactions on visualization and computer graphics **19**(12), 2227–2236 (2013)
17. Moreau, C., Chanson, A., Peralta, V., Devogele, T., de Runz, C.: Clustering sequences of multi-dimensional sets of semantic elements. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing. pp. 384–391 (2021)
18. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology **48**(3), 443–453 (1970)
19. Noel, D., Villanova-Oliver, M., Gensel, J., Le Quéau, P.: Design patterns for modelling life trajectories in the semantic web. In: Proceedings of the International Symposium on Web and Wireless Geographical Information Systems (W2GIS). pp. 51–65 (2017)

20. Oubelmouh, Y., Fargon, F., De Runz, C., Soulet, A., Veillon, C.: Identifying survival-changing sequential patterns for employee attrition analysis. In: Proceedings of the 10th International Conference on Data Science and Advanced Analytics (DSAA). pp. 1–10 (2023)
21. Rama, K., Canhão, H., Carvalho, A.M., Vinga, S.: AliClu - temporal sequence alignment for clustering longitudinal clinical data. BMC Medical Informatics and Decision Making **19**(1), 289 (2019)
22. Santos, Y., Giuliani, R., Portela, T., Renso, C., Carvalho, J.: MAT-CA: a tool for multiple aspect trajectory clustering analysis. In: ACM Conferences. pp. 40–43 (2023)
23. dos Santos Mello, R., Bogorny, V., Alvares, L.O., Santana, L.H.Z., Ferrero, C.A., Frozza, A.A., Schreiner, G.A., Renso, C.: MASTER: A multiple aspect view on trajectories. Transactions in GIS **23**(4), 805–822 (2019)
24. Saqr, M., López-Pernas, S., Helske, S., Durand, M., Murphy, K., Studer, M., Ritschard, G.: Sequence analysis in education: principles, technique, and tutorial with r. In: Learning analytics methods and tutorials: a practical guide using R, pp. 321–354. Springer Nature Switzerland Cham (2024)
25. Saveliev, E.S., van der Schaar, M.: Temporai: Facilitating machine learning innovation in time domain tasks for medicine. arXiv preprint arXiv:2301.12260 (2023)
26. Soubeiga, A., Guyet, T., Antoine, V.: Soft-ECM: An extension of evidential C-means for complex data. In: Proceedings of the International Conference on Fuzzy Systems (Fuzz-IEEE) (2025)
27. Tang, S., Davarmanesh, P., Song, Y., Koutra, D., Sjoding, M.W., Wiens, J.: Democratizing ehr analyses with fiddle: a flexible data-driven preprocessing pipeline for structured clinical data. Journal of the American Medical Informatics Association **27**(12), 1921–1934 (2020)
28. Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., et al.: Tslearn, a machine learning toolkit for time series data. The Journal of Machine Learning Research **21**(1), 4686–4691 (2020)
29. van de Water, R., Schmidt, H.N.A., Elbers, P., Thoral, P., Arnrich, B., Rockenschaub, P.: Yet another icu benchmark: A flexible multi-center framework for clinical ml. In: The Twelfth International Conference on Learning Representations (2024)
30. Wornow, M., Xu, Y., Thapa, R., Patel, B., Steinberg, E., Fleming, S., Pfeffer, M.A., Fries, J., Shah, N.H.: The shaky foundations of large language models and foundation models for electronic health records. npj Digital Med. **6**(135), 1–10 (2023)
31. Xu, J., Gallifant, J., Johnson, A.E., McDermott, M.: Aces: Automatic cohort extraction system for event-stream datasets. arXiv preprint arXiv:2406.19653 (2024)
32. Yeshchenko, A., Mendling, J.: A survey of approaches for event sequence analysis and visualization. Information Systems **120**, 102283 (2024)