Proceedings Paper:
Palczewska, A, Palczewski, J orcid.org/0000-0003-0235-8746, Aivaliotis, G et al. (1 more author) (2017) RobustSPAM for Inference from Noisy Longitudinal Data and Preservation of Privacy. In: IEEE ICMLA 2017 Conference proceedings. IEEE 16TH International Conference on Machine Learning and Applications - ICMLA 2017, 18-21 Dec 2017, Cancun, Mexico. Institute of Electrical and Electronics Engineers , pp. 344-351. ISBN 978-1-5386-1417-4

https://doi.org/10.1109/ICMLA.2017.0-137

RobustSPAM for inference from noisy longitudinal data and preservation of privacy

Anna Palczewska
School of Geography
University of Leeds, UK
Email: a.palczewska@leeds.ac.uk

Jan Palczewski and Georgios Aivaliotis
School of Mathematics
University of Leeds, UK
Email: {j.palczewski,g.aivaliotis}@leeds.ac.uk

Łukasz Kowalik
Institute of Informatics
University of Warsaw, Poland
Email: kowalik@mimuw.edu.pl

***Abstract*—The availability of complex temporal datasets in social, health and consumer contexts has driven the development of pattern mining techniques that enable the use of classical machine learning tools for model building. In this work we introduce a robust temporal pattern mining framework for finding predictive patterns in complex timestamped multivariate and *noisy* data. We design an algorithm RobustSPAM that enables mining of temporal patterns from data with noisy timestamps. We apply our algorithm to social care data from a local government body and investigate how the efficiency and accuracy of the method depends on the level of noise. We further explore the trade-off between the loss of predictivity due to perturbation of timestamps and the risk of person re-identification.**

***Index Terms*—robust, temporal pattern, noisy data, privacy**

I. Introduction

The widespread collection of data in social, health and consumer contexts has contributed to the availability of complex temporal datasets. Data instances collected in these datasets are characterized by a variable number of irregularly spaced timestamped events that include information about continuous activities and instantaneous events (e.g. Electronic Health Records, machine log files, credit/debit card use). The increase in the number of complex temporal datasets has prompted the development of methods that extend applicability of classical statistical, machine learning and data mining methods to those datasets [1]–[3]. This is particularly important in monitoring or detection problems such as: patient monitoring [4] or fraud detection [5]. These methods involve identification of relevant temporal patterns of events. Patterns represent sequences of time-point (or time-intervals) events. They are used to encode original variable-length data instances as fixed-length binary vectors representing presence or absence of chosen patterns therefore enabling application of existing classification and prediction tools.

Existing temporal pattern mining algorithms extend sequential pattern mining methods to a more complex case of time-related pattern mining. They are based on time point or interval representations of the data and are typically exploiting Allen-type relationships [6]. They require a temporal abstraction of the raw data into ordered coded events (e.g. two weeks of high blood pressure) [7]. This recoded data is searched for frequent sequential patterns using algorithms such as Generalized Sequential Pattern algorithm (GSP) [8], PrefixSpan [9], Sequential PAttern Mining (SPAM) [10] or Sequential PAttern Discovery using Equivalence classes (SPADE) [11]. The application of the above algorithms to monitoring and detection tasks has been successful for datasets with accurately recorded timestamps. Our objective is to build a frequent pattern mining algorithm that performs well in mining temporal data that include error/noise in the timestamps, a common occurrence in manually maintained databases. The existing methods will not work for noisy data as the noise may perturb frequent patterns into a large number of rare patterns, which therefore will be excluded from the inference or, if included, may lead to over-fitting of a trained model.

This paper makes two contributions. Firstly, we design a robust approach for analysis of longitudinal data that can account for noise/errors in the recorded timestamps. Such errors are common in information systems maintained manually such as health records or social care systems. Errors in inputting information in the system might result in different or even reverse temporal relations which are not true (for example event A was recorded after event B whereas they took place in the reverse order). Existing algorithms that allow for some tolerance [3] take into account constant error margins around intervals. Our contribution is both theoretical (formulation of robust temporal patterns) and algorithmic (efficient methods for identification of such robust patterns which go beyond and are significantly harder than classical problems of sequential pattern mining). Our approach focuses on using time points instead of intervals and fitting probabilistic models for the errors in the time stamp around these time points. Intervals are represented as a start point and an end point with possibly different errors around those points. This representation allows for intuitive and relatively compact representation of patterns while still retaining predictive power similar to interval-based methods. Moreover, we noticed significant gains in terms of computational and algorithmic complexity in our robust pattern mining algorithm compared to a similar approach that we explored for interval mining (not reported in the paper, but will be clear to the reader after reading this article).

Our second contribution concerns privacy preservation/statistical disclosure problems in the context of longitudinal data. As such data contain rich temporal information, they

AP, JP and GA gratefully acknowledge support by grant EP/N013980/1 from the UK Engineering and Physical Sciences Research Council.

may be used in person re-identification [12], [13]. A typical approach of using aggregate statistics is often insufficient for complex datasets as experience in [14] shows. Instead, we borrow from a classical approach in statistical disclosure and perturb timestamps with a random noise. Analysis with such perturbed data is possible through our robust timestamps approach. We measure the risk of re-identification with the Unicity measure [15]. This represents the percentage of uniquely identified records given a number of points in time. We will report on the performance of this method in predictive modeling; in particular, we will discuss the loss of predictive power and the sensitivity to specification of the noise.

As a test ground for our methods we will use a risk stratification problem in Adult Social Care. The dataset is provided by a local authority and contains for each client: timestamped referrals, assessments, reviews and services provided as well as static health and socio-economic descriptors. The aim is to identify clients that are at the highest risk of moving into expensive care (such as nursing or residential care) in order to provide additional services to extend their independence.

The paper is organized as follows: Section II introduces the classification problem. In Section III, we provide the mathematical foundation for defining robust order and robust pattern. Section IV describes the TestPattern algorithm and Robust Sequential Pattern Mining (RobustSPAM) algorithms. In Section V we explore the performance of the RobustSPAM algorithm, analyze the loss of predictive power due to the noise and the trade-off between the noise and privacy preservation. Section VI concludes this work.

II. Problem Definition

The general approach for modeling temporal datasets is defined as follows. Let $D = \{\langle x_i, y_i\rangle\}_{i=1}^n$ be a training dataset: $x_i \in X$ is a collection of multidimensional timestamped instances and $y_i \in Y$ is a class label associated with x_i. The main aim is to learn a function $f : X \to Y$ that can classify unlabeled instances.

The dataset considered in this paper was provided by a local government body and comprises information from an Adult Social Care system. Every data instance x_i is a complete record of interactions with and services provided for a client. There are four main activities reported in the systems: referrals, assessments, services and reviews. The class label y_i denotes whether or not a client is in receipt of one of two expensive services (nursing or residence housing). A similar problem, although on a shorter time scale, has been studied for electronic health records (EHR) in [16].

Data instances are characterized by a variable number of irregularly timestamped events. We may have a situation that a client was referred for a need of care but was not eligible to receive any service then, but after a number of years ended up with the same referral reason and received a number of services. There are groups of clients (e.g mental health or disability problems) that receive support throughout their lives and groups of people that have single events like equipment or home adaptation only. This heterogeneity of client data means that a classifier cannot be learnt directly from the data.

A common approach for timestamped datasets is to apply a (dataset-based) transformation $\phi : X \to X'$ that maps each instance x_i into a fix-length vector x'_i while retaining as much as possible temporal characteristics of x_i. In this work we use a dynamic transformation as proposed in [16]. We learn transformation ϕ from data using temporal pattern mining by following the steps:

1) convert activities (events and intervals) into point-time sequences of events,
2) find frequent temporal patterns from the event sequence data.

Having transformed data $D' = \{\langle x'_i, y_i\rangle\}_{i=1}^n$ we can use classical machine learning methods to learn the classifier.

Electronic health records, social care records and other manually maintained datasets often contain errors in reporting timestamps. We also consider the case when data contain noise introduced in the process of anonymization. Current approaches for temporal pattern mining do not take these scenarios into account. Temporal patterns are searched within sequences of events ordered using time-point and/or interval relationships (e.g Allen's relations [17]) which assume accurate timestamps, therefore affecting pattern discovery. In this paper we address this problem. We propose a robust pattern mining algorithm for pattern discovery and we provide the analysis of the classifier predictive power vs the added noise to the timestamps. In the following sections we will explain in details steps of our modeling approach.

III. Robust Temporal Patterns

A. Event Abstraction

Let Σ be an **abstraction alphabet** that represents a finite set of permitted abstractions. A multivariate variable x_i is transformed into an event representation $e_i = \langle c_i, t_i\rangle$ with $c_i \in \Sigma$. In what follows we will write $t(e_i)$ for t_i and e_i for c_i. Temporal abstractions have been well studied for transformation of numerical time series variables to a high level qualitative representation [7]. In case of EHR data, each clinical variable is transformed to an interval-based representation [16]. For categorical data, the alphabet Σ consists simply of all categories. Unfortunately, in many cases an event e_i is described by more than one categorical value (e.g., location and type of medical intervention). Constructing an alphabet through a product of possible categories leads usually to impractically large number of event codes (c.f. column 3 of Table IV) making the pattern mining part very computationally intensive and inference often infeasible. In such cases there is a need to recode categories and arrange them in hierarchies which can then be used at a different level of detail.

In this work we consider a social care dataset from a local government body with categorical variables only. To demonstrate the complexity of the data mentioned in the previous paragraph, a referral activity has been coded using three significant variables: source (by whom the referral was

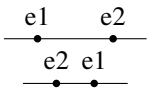

Fig. 1. Effect of noise in timestamps on relative position of events on time axis. The top picture shows real times when events occurred. The effect of noise in timestamps in presented on the bottom picture.

made), reason (why) and outcome (action taken after the referral was made). The source categories were grouped into two categories: health (H) when referral came from primary or secondary care and other (O) representing e.g. police, self-referral, family/neighbors, legal agent, etc. The variable Reason was grouped in a number of categories such as (need for assessment (A.1), referral to mental health (A.3), carer assessment (C) etc.). The outcome variable includes: support plan amended (A.1), accept for assessment (A.2), accept for disability services (B.1), other decision (J). We represent the code for a referral activity as *activity_type-source-reason-outcome*, e.g. Ref-O-A.1-J or Ref-H-C-B.1.

After abstracting all variables/activities, we represent an instance x_i as a **sequence of events (SE)** sorted according to their timestamps t_i:

$$SE_i = \langle e_1, e_2, e_3, \ldots, e_l \rangle,$$

where $t(e_l) \leq t(e_{l+1})$.

B. Robust Temporal Relationship

Consider two events e_1 and e_2 that occur at times $t(e_1)$ and $t(e_2)$. Their relationship can be described in the following way:

- e_1 is before e_2 (one event happens before the other event);
- e_1 co-occurs with e_2 (both events happen at the same time).

In standard approaches, both relations are described with the following *order*:

$$e_2 \geq e_1 \Leftrightarrow \; t(e_2) - t(e_1) \geq 0 \tag{1}$$

The underlying assumption above is that observed data are exact. However, when timestamps are recorded with an error or are noisy, the relation between *recorded* timestamps $t(e_2) \geq t(e_1)$ may not imply that e_2 *really* happened after e_1. In this paper we introduce a robust relation between events that is immune to such noisy perturbation of timestamps.

Definition 1: (Robust ordering) Event e_1 happens before e_2 in *robust ordering* with respect to β, denoted as $e_1 \preceq e_2$, if

$$t(e_2) - t(e_1) \geq \beta(e_1, e_2). \tag{2}$$

An ordering is called *separable* if

$$\beta(e_1, e_2) = \hat{\beta}(e_1) + \hat{\beta}(e_2) \tag{3}$$

for a function $\hat{\beta}$ depending on the event code.

Example.
Assume that the noise introduced to the timestamps is normally distributed with mean 0 and variance σ^2, independently from the type of event. Denoting by $t(\cdot)$ a recorded timestamp and by $t^*(\cdot)$ a real timestamp, we have

$$t(\cdot) = t^*(\cdot) + \epsilon, \quad \epsilon \sim N(0, \sigma^2).$$

We want to find a threshold K such that

$$P(t(e_2) - t(e_1) \geq \underbrace{K}_{\beta(e_1,e_2)} \mid t^*(e_1) \leq t^*(e_2)) \geq \alpha$$

for some confidence level α, e.g., $\alpha = 95\%$. Then setting $\hat{\beta}(\cdot) = K/2$ defines a separable robust ordering such that $t^*(e_1) \leq t^*(e_2)$ implies that $e_1 \preceq e_2$ with probability α, i.e., we identify the true ordering of events with probability α. Notice that a normally distributed noise does not allow for identification with probability 1 as the noise can potentially be arbitrarily large (however with a very small probability).

We compute

$$\begin{aligned}
&P(t(e_2) - t(e_1) \geq K \mid t^*(e_1) \leq t^*(e_2)) \\
&\geq P(t(e_2) - t(e_1) \geq K \mid t^*(e_1) = t^*(e_2)) \\
&= P(t^*(e_2) + \epsilon_2 - t^*(e_1) - \epsilon_1 \geq K \mid t^*(e_1) = t^*(e_2)) \\
&= P(\underbrace{\epsilon_2 - \epsilon_1}_{N(0,2\sigma^2)} \geq K) \\
&= P(\underbrace{\frac{\epsilon_2 - \epsilon_1}{\sqrt{2}\sigma}}_{N(0,1)} \geq \frac{K}{\sqrt{2}\sigma}) = \Phi\left(-\frac{K}{\sqrt{2}\sigma}\right)
\end{aligned}$$

Therefore,

$$P(t(e_2) - t(e_1) \geq K \mid t^*(e_1) \leq t^*(e_2)) \geq \alpha$$

if

$$\Phi\left(-\frac{K}{\sqrt{2}\sigma}\right) = \alpha,$$

i.e.,

$$K = -\sqrt{2}\sigma\Phi^{-1}(\alpha).$$

This yields a robust separable ordering with a constant function

$$\hat{\beta} \equiv -\frac{\sqrt{2}}{2}\sigma\Phi^{-1}(\alpha).$$

Taking $\alpha = 50\%$ gives $\hat{\beta} = 0$ and reclaims an ordinary ordering of real numbers. In practice, one will usually use $\alpha > 50\%$; for example $\alpha = 90\%$ gives $K \approx -1.38\sigma$. $\triangleleft$

Notice that unlike the exact relation between true timestamps $t^*(\cdot)$, the robust ordering defined above is not transitive if $\hat{\beta} < 0$. Indeed, assume that $e_1 \preceq e_2 \preceq e_3$. This means that

$$\begin{aligned}
t(e_2) - t(e_1) &\geq \hat{\beta}(e_1) + \hat{\beta}(e_2), \\
t(e_3) - t(e_2) &\geq \hat{\beta}(e_2) + \hat{\beta}(e_3).
\end{aligned}$$

From here we can only deduce that

$$t(e_3) - t(e_1) \geq \hat{\beta}(e_1) + 2\hat{\beta}(e_2) + \hat{\beta}(e_3),$$

while the right hand side is smaller (because $\hat{\beta}(e_2) < 0$) than $\hat{\beta}(e_1) + \hat{\beta}(e_3)$ which defines the relation $e_1 \preceq e_3$.

C. Robust Temporal Patterns

The lack of transitivity of robust ordering means that representations of temporal patterns may be very complicated as temporal relationship between every pair of events has to be specified. In this paper, motivated by computational tractability, we restrict attention to patterns in which consecutive events are assumed to be ordered according to the robust ordering $\preceq$. This means that a pattern can be represented by a sequence of events and the robustness only enters at the stage of verifying if it is contained in a given instance.

Definition 2: (Temporal Pattern) A temporal pattern is a sequence of events $P = \langle p_1, p_2, \ldots, p_k \rangle$. The size $|P|$ (also denoted $length(P)$) of P is defined as the number k of events in the sequence.

Definition 3: (Robust inclusion) An instance x_i with the event representation $SE_i = \langle e_1, e_2, \ldots, e_l \rangle$ contains a pattern $P = \langle p_1, p_2, \ldots, p_k \rangle$, denoted $P \in SE_i$, if there is a one-to-one mapping p from $\{1, \ldots, k\}$ to $\{1, \ldots, l\}$ such that

$$\forall_{i \in 1,\ldots,k} : e_{p_i} \preceq e_{p_{i+1}},$$

where $\preceq$ is a robust ordering.

IV. Temporal Pattern Mining Algorithm for Robust Timestamps

In this section, we present an algorithm for mining frequent robust temporal patterns. The algorithm takes as an input D: the sequence of events (SE) representation for each client and the function $\hat{\beta}$. It finds all frequent l-length patterns with the length l within given bounds similarly as in the classical SPAM [10]. The procedure is iterative and follows the following two phases:

1) (candidate generation phase) generate a candidate pattern by extending a frequent l-pattern using depth-first-search (DFS) algorithm (as in SPAM),
2) (counting phase) accept the new pattern if it has support at least τ, i.e. the proportion of records that contain it is at least τ. This is where our algorithmic contribution lies.

An l-pattern is a sequence $s = \langle e_1, e_2, \ldots, e_l \rangle$ and $s \in SE$. Its extension comprises adding a new event code at the end. If the sequence s is not frequent any child sequence generated from s cannot be frequent, which allows for truncation of the search tree. This is called an a priori principle [18].

In the following subsection we describe details of these steps.

A. Candidate Generation

To generate candidates we used the depth-first search (DFS) approach as it was proposed in [10]. We start at the root node with a 0-pattern. At each node n we extend the l-pattern adding a new event e_k at its tail and we calculate its support. If the support of a generated pattern s is greater than minimum support threshold τ, we store that sequence and repeat DFS recursively on s. The maximum length of any sequence is limited since the input database is finite. If the support of s is less than τ, then we do not need to repeat DFS on s as indicated by the a priori principle. If none of the generated children are frequent, then the node is a leaf and we backtrack up the tree and follow further steps of DFS procedure. Details are provided in Algorithm 1, RobustSPAM (Robust Sequential Pattern Mining). The algorithm takes the following inputs: a prefix (l-pattern to be extended), data D, a list possible events for extensions Sn, maximum pattern length maxlen and the minimal support τ. In the first iteration $Sn = \Sigma$.

Algorithm 1 RobustSPAM

Input: prefix, D, Sn, maxlen, τ, β
1: sTemp = $\emptyset$
2: sPattern = $\emptyset$
3: **for** $(c_i : Sn)$ **do**
4: pattern = (prefix, c_i)
5: support = **GetSupport**(D, pattern, β)
6: **if** (support $> \tau$) **then**
7: store pattern
8: sTemp = sTemp $\cup$ c_i
9: sPattern = sPattern $\cup$ pattern
10: **end if**
11: **end for**
12: **for** $(i : sPattern)$ **do**
13: **if** (maxlen $> |i|$) **then**
14: **RobustSPAM**(i, D, sTemp, maxlen, τ, β)
15: **end if**
16: **end for**

Algorithm 2 GetSupport

Input: D, pattern[], β
1: count $= 0$
2: **for** $(i : D)$ **do**
3: Convert SE_i to tuple representation T[] = {(t, st, c)} using β
4: count=count + **TestPattern**(T[], pattern[])
5: **end for**
6: **return** count/|D|

B. Pattern Count

In contrast to other temporal pattern mining algorithms the counting phase is challenging due to robust timestamps. In existing algorithms only the time order of events matters and standard subsequence matching algorithms suffice; their complexity is linear with respect to the length of the record. However, in the case of robust timestamps a definitive ordering of events cannot be established (c.f. Definition 3) and existing sequence matching algorithms cannot be applied. We therefore designed an algorithm for pattern matching which, thanks to the particular choice of function $\beta(e_1, e_2) = \hat{\beta}(e_1) + \hat{\beta}(e_2)$ with $\hat{\beta} < 0$, has also a linear complexity – the feature crucial for the efficiency of our approach.

The following observation lies at the roots of our algorithm:

$e_1 \preceq e_2$ if and only if

$$t(e_2) - \hat{\beta}(e_2) \geq t_1 + \hat{\beta}(e_1).$$

(Recall that $\hat{\beta}(\cdot) < 0$.) Figure 2 presents the robust order $e_1 \preceq e_2$ between events e_1 and e_2.

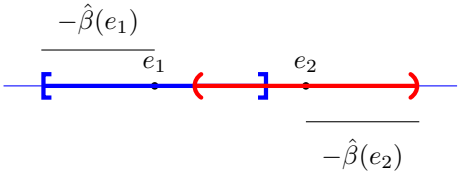

Fig. 2. Robust order for pattern count algorithm.

Following the above observation, we recode events for each client in the following way. For each event, we create two tuples (t,st,c) representing timestamp (t), status (st) and event code (c) as follows:

$$\big(t(e_i) + \hat{\beta}(e_i), "start", e_i)\big), \qquad \big(t(e_i) - \hat{\beta}(e_i), "end", e_i)\big).$$

For each client, those tuples are sorted with respect to (t) and stored in table T[] (see Algorithm 2), which forms the input to Algorithm 3. We use two arrays Active[] and Used[] that keep track of active event codes that can be used to match a pattern and those events which have already been matched, respectively. We walk through the table T[]. When we encounter a starting tuple (status="start") we increase the count for the code of this event within Active[] array. Then we test if there is a match with a corresponding event code in the pattern. If yes then we increase the event count in Used[] array and we test consecutive event code from the pattern against all the active but not already used codes. This is repeated until all event codes in Active[] are used. We then proceed with the next element from T[]. If the tuple status represents the closing intervals then we reduce the count in Active[] and Used[] arrays (the event is no more available for a matching).

V. Experimental results

In this section, we present results for a dataset on provision of adult social care by a local government. We test our approach when timestamps are perturbed by various levels of additive (Gaussian) noise thus enabling comparison to a benchmark model obtained under the assumption that the original dataset is free of inaccuracies in timestamp recording. We further explore the trade-off between privacy preservation through perturbation of timestamps and the loss of predictive power.

A. Dataset

The original dataset consists of 100,000 records of adult social care clients. For each client, there is an independent variable (label) yes/no describing if the client is in receipt of an expensive social care support package. Each client's record comprises a number of activities and their timestamps. There are four types of activities: referrals, assessments, services and reviews. Activities are described in the original dataset by a number of attributes. We transformed them into a coded event dataset, see Table I. The referral event code is composed of three parts: source (who made a referral) – 2 categories; reason (why the service is needed) – 12 categories and outcome (decision for assessment) – 15 categories. The assessment activity was coded using only one variable – eligibility with 16 categories. The service activity was coded with 116 categories and the review activity – with 17 categories. The numbers of all possible codes for each event type is presented in the second column of Table I. The actual number of codes found within the dataset are showed in the last column of the same table.

For each client we represented the data as a sequence of events, sorting them by the timestamp. We selected 42,585 clients – those with more than three events $|SE_i| > 3$. We used this dataset to test model classification performance with an increasing level of timestamp perturbation. Further, to explore in more detail the risk of person re-identification and to analyze the loss of predicting power for our classifier to specification of the introduced noise, we selected 2000 clients and restricted their data to services only: there were 60 event

Algorithm 3 TestPattern

Input: T[], pattern[]

```
Active[]=0, Used[]=0, j = 0, x = 0
for i = 1 to n do
   (t,st,c) := T[i]
   if (st=="start") then
      Active[c]++
      if (pattern[j]==c) then
         ps = pattern[j]
         do
            Used[ps]++
            j++
            ps = pattern[j]
         while Active[ps]-Used[ps]> 0 &j < |pattern|)
         if (j==|pattern|) then
            x = 1
            exit
         end if
      end if
   else
      Active[c]−−
      Used[c] = max(0, Used[c]-1)
   end if
end for
return x
```

TABLE I
Re-coding activities to the event representation.

Event Type	Potential no. codes	No. codes in data
referral	260	18
assessments	16	10
services	116	90
reviews	17	8

type codes describing a type of service the client received. To increase the accuracy of the reference model (so that the effect of noise is clearer) we artificially assigned labels 0 and 1 to clients using clustering methods based on frequent temporal patterns.

B. Workflow of the Experiment

The experiment was performed in two steps:

1) Classification performance - to test ability of the RobustSPAM algorithm to find patterns in noisy (randomly perturbed) data.
2) Detailed analysis of the trade-off between the loss of predictive power and the ability to re-identify individuals from timestamps.

In the first step we used original label defining if a person receives an expensive care support package. We followed the workflow provided in Section II. We converted activities into point-time sequences of events. We then perturbed timestamps by adding noise as follows:

$$\text{timestamp} \longrightarrow \text{timestamp} + \text{noise } N(0, \gamma^2) \text{ in days},$$

where $N(0, \gamma^2)$ stands for the normal distribution with the mean 0 and the variance γ^2. We applied the RobustSPAM algorithm for $\hat{\beta}(\cdot) = 2\gamma$ to find frequent patterns. Then we transformed the sequences of events into a binary matrix where each column represented presence (1) or absence (0) in a client record of a frequent pattern. The random forest model was used as a classifier.

In the second step, we used a similar procedure but extended the range of γ and assessed the probability of re-identification.

In both cases we mined frequent robust temporal patterns from each class label separately. Using local minimum support rather than global minimum support for the entire training dataset allows for finding patterns relevant for a given class, particularly in the case of unbalanced data.

C. Classification Performance

In this section we test the ability of our RobustSPAM algorithm to capture temporal patterns that are important for prediction. We compare our results with the original SPAM algorithm applied on unperturbed dataset. The dataset included 107 codes representing: referrals, assessment, service and review activities for 42,585 clients. The total number of 17503 clients was labeled as not using expensive care, whereas 25083 were labeled as recipients of an expensive care support package.

Firstly, we used the SPAM algorithm [10] to find frequent patterns for non-perturbed data. Then we used these patterns to build random forest classifier. To build a classifier we split dataset into two sets: training and testing. The training dataset included 4000 clients that were balanced equally between two classes. The testing dataset contained the remaining records. We used the same training and testing datasets for the RobustSPAM algorithm. Introducing different levels of noise to timestamps we tested the ability of the RobustSPAM to find important predictive patterns. The results of the analysis are presented in Table II. For each level of noise γ (here γ is the standard deviation of the Gaussian noise) we collected frequent patterns and the model performance for the training and testing datasets. To decide if a pattern is frequent we used 10% minimum support. The minimum pattern length was set to 3 and we allowed patterns to grow to the maximum length. Sensitivity (true positive rate) represents the proportion of clients who are at risk of expensive care and are correctly classified. Specificity (true negative rate) represents a proportion of clients that are not at risk of expensive care and are correctly classified. The difference between sensitivity and specificity is bigger for the testing dataset than for the training dataset because the majority of clients in the testing dataset are not at risk of expensive care. The significant drop in the model accuracy is observed only for the noise with the standard deviation ≥ 50 days.

The original SPAM and RobustSPAM with $\gamma = 0$ find the same patterns within the data (as expected), see the column Patterns in the first two rows of Table II (different values of accuracy are caused by randomness involved in the construction of the random forest classifier). When we add noise (reported as the standard deviation γ in days) we notice an increase of the number of patterns and a drop in the model predictive power suggesting the loss of important temporal information from the data. The increase in the number of identified patterns is due the algorithm relaxing conditions for matching a pattern (so that the noise does not prevent patterns to be found in data) allowing, therefore, more patterns to reach the required support level. However, some predictive power is retained even for large noise, which may be explained by the fact that the presence of certain (atemporal) combinations of events is significant for the prediction.

D. Predictivity and Re-identification vs Noise

In this section we restrict attention to a smaller population of 2000 clients and with reduced number of codes as explained in Subsection V-A. This enables us to make a more thorough analysis of the link between the noise, the predictive power and the risk of re-identification. The latter is assessed based on a modification of the unicity measure for spacio-temporal data [13].

1) Unicity Measure: The unicity measure was introduced in [13] as the risk of client re-identification knowing p pieces of spatio-temporal information about a user. The algorithm proposed in [15] is based on the Monte Carlo paradigm. For p randomly selected points from a randomly selected client's sequence of events (e.g. mobile phone or credit card usage), one counts how many clients have the same subset of events within their records. This is repeated many times and the percentage of those with an unique p-sequence of events is reported.

We use the unicity measure to analyze risk of re-identification using coded longitudinal data. In our framework, codes replace the spatial information in the original algorithm and time is measured with daily accuracy. We applied the

TABLE II
CLASSIFICATION PERFORMANCE.

algorithm	noise γ	Patterns	Train Acc	Train Sn	Train Sp	Test Acc	Test Sn	Test Sp
SPAM	NA	4454	0.8012	0.8193	0.7851	0.8019	0.8939	0.7069
RobustSPAM	0	4454	0.8018	0.8178	0.7872	0.7974	0.8908	0.7017
RobustSPAM	2	5101	0.8058	0.8365	0.7801	0.7967	0.8887	0.7016
RobustSPAM	7	6622	0.7975	0.8073	0.7883	0.7935	0.8804	0.7012
RobustSPAM	14	9019	0.7887	0.8203	0.7629	0.7884	0.8815	0.6928
RobustSPAM	25	13622	0.7827	0.789	0.7768	0.7866	0.8791	0.6912
RobustSPAM	50	29764	0.6158	0.6226	0.6097	0.5986	0.7173	0.5004

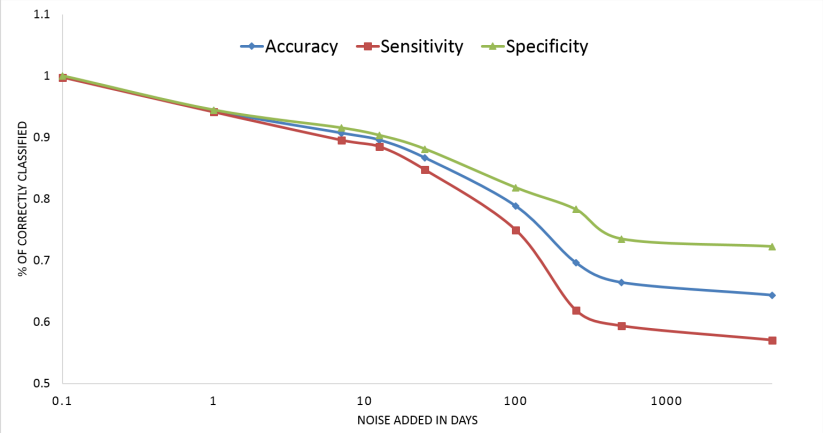

Fig. 3. Predictivity vs level of noise

TABLE III
RISK OF RE-IDENTIFICATION VS LEVEL OF NOISE.

Noise γ	p=2	p=3	p=4	p=5	p=6
0	0.62	0.95	0.99	0.99	1
1	0.21	0.95	0.99	0.99	0.99
7	0.07	0.46	0.79	0.92	0.97
12.5	0.05	0.32	0.66	0.85	0.93
25	0.03	0.21	0.47	0.69	0.82
100	0.01	0.06	0.17	0.30	0.42
250	0.006	0.02	0.07	0.13	0.21
500	0.005	0.02	0.04	0.08	0.13
5000	0.003	0.01	0.03	0.05	0.08

TABLE IV
TESTING SET ACCURACY VS LEVEL OF NOISE.

Noise γ	Patterns	Acc	Sn	Sp
0	13	0.99	0.99	1
1	27	0.94	0.94	0.94
7	40	0.91	0.90	0.92
12.5	45	0.90	0.89	0.90
25	59	0.87	0.85	0.88
100	218	0.79	0.75	0.82
250	565	0.70	0.62	0.78
500	819	0.66	0.59	0.73
5000	1283	0.64	0.57	0.72

algorithm for various levels of perturbation of timestamps. Table III collects the results.

2) Loss in Model Prediction: For a selection of noise standard deviations γ reported in first column of Table IV, we perturbed the data and applied our inference approach with robust timestamps. Each model was generated on the same training sample of 800 clients and tested on the remaining 1200 clients. Both datasets were balanced.

Figure 3 presents the model predictivity as a function of the noise standard deviation γ; exact values and the number of patterns found are collected in Table IV. Comparison of Tables III and IV yields a conclusion that the noise with standard deviation of $\gamma = 25$ days retains good predictivity while providing good prevention of re-identification based on 2 external observations and acceptable for 3 observations. Increasing noise to $\gamma = 100$ strongly improves the privacy (even at the level of 6 observations) but at a cost of a visible loss of predictive power. Taking into account that this test was performed on a small dataset of 2000 individuals, we may conclude that the perturbation of timestamps in the whole dataset of over 42,000 clients should provide significantly stronger privacy protection. Further investigations will be taken up in future research.

All experiments were conducted on Intel(R) Core(TM) i7-4790 3.0Ghz CPU and 16GB of RAM.

VI. CONCLUSIONS

Techniques for mining (time-point) sequential data as well as methods to mine time-interval data have been developed over the last twenty years. These algorithms have been successfully used for datasets with accurately recorded timestamps. However, existing methods do not work for noisy data as the noise may blow a true frequent pattern into a large number of rare patterns, which therefore will be excluded

from the inference or, if included, may lead to over-fitting of a trained model.

In this paper we proposed a new robust temporal pattern mining framework for finding predictive patterns in the presence of error/noise in timestamps, a common occurrence in manually maintained databases. We presented an a priori algorithm RobustSPAM that mines time-point patterns and is extendable to time-interval data via encoding the beginning and the end of an interval as two time-point events. Evaluation on adult social care data, comprising both time-point and time-interval data, showed that the algorithm successfully finds patterns that are important for client risk stratification. Due to increased concerns about data privacy and recent research showing that longitudinal data are at high risk of re-identification, we explored the trade-off between the loss of predictivity due to random perturbation of timestamps and the risk of person re-identification.

Future work includes optimization of the RobustSPAM algorithm and extension to other specifications of the function $\beta(e_1, e_2)$, which will bring algorithmic and conceptual challenges. We also plan tests of the methodology on Delirium EHR dataset when screening process is completed and we obtain an official permission to use that dataset.

REFERENCES

[1] I. Batal, G. F. Cooper, D. Fradkin, J. Harrison, F. Moerchen, and M. Hauskrecht, "An efficient pattern mining approach for event detection in multivariate temporal data," *Knowledge and Information Systems*, vol. 46, no. 1, pp. 115–150, 2016.

[2] Y.-C. Chen, J. T.-Y. Weng, and L. Hui, "A novel algorithm for mining closed temporal patterns from interval-based data," *Knowledge and Information Systems*, vol. 46, no. 1, pp. 151–183, 2016.

[3] R. Moskovitch and Y. Shahar, "Classification of multivariate time series via temporal abstraction and time intervals mining," *Knowledge and Information Systems*, vol. 45, no. 1, pp. 35–74, Oct. 2015.

[4] M. Hauskrecht, I. Batal, M. Valko, S. Visweswaran, G. F. Cooper, and G. Clermont, "Outlier detection for patient monitoring and alerting," *Journal of Biomedical Informatics*, vol. 46, no. 1, pp. 47–55, 2013.

[5] A. Srivastava, A. Kundu, S. Sural, and A. Majumdar, "Credit card fraud detection using hidden markov model," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 1, pp. 37–48, Jan 2008.

[6] J. F. Allen, "Towards a general theory of action and time," *Artificial Intelligence*, vol. 23, no. 2, pp. 123 – 154, 1984.

[7] Y. Shahar, "A framework for knowledge-based temporal abstraction," *Artificial Intelligence*, vol. 90, no. 1, pp. 79 – 133, 1997.

[8] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, ser. EDBT '96. London, UK, UK: Springer-Verlag, 1996, pp. 3–17.

[9] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. chun Hsu, "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *Proceedings of the 17th International Conference on Data Engineering*, ser. ICDE '01, 2001, pp. 215–224.

[10] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '02. New York, NY, USA: ACM, 2002, pp. 429–435.

[11] M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Mach. Learn.*, vol. 42, no. 1-2, pp. 31–60, Jan. 2001.

[12] J. Domingo-Ferrer and R. Trujillo, "Anonymization of trajectory data," https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.46/2011/32_Domingo-Trujillo.pdf, October 2011.

[13] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific Reports*, vol. 3, Mar. 2013.

[14] M. Bardsley, J. Billings, and L. S. A. Chassin, "Predicting social care costs: a feasibility study," https://www.nuffieldtrust.org.uk/research/predicting-social-care-costs-a-feasibility-study, February 2011.

[15] Y.-A. de Montjoye, "Computational privacy: Towards privacy-conscientious uses of metadata," Ph.D. dissertation, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, September 2015.

[16] I. Batal, D. Fradkin, J. Harrison, F. Moerchen, and M. Hauskrecht, "Mining recent temporal patterns for event detection in multivariate time series data," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 280–288.

[17] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983.

[18] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of the Eleventh International Conference on Data Engineering*, ser. ICDE '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 3–14.