Mining Temporal Exception Rules from Multivariate Time Series Using a new Support Measure

Thábata Amaral and Elaine P. M. de Sousa

University of São Paulo (USP), Brazil
thabataamaral@usp.br, parros@icmc.usp.br

Abstract. Association rules are a common task to discover useful and comprehensive relationships among frequent and infrequent data. Frequent patterns describe a usual behavior, while infrequent ones represent uncommon knowledge. Our interest lies in finding exception rules, a class of infrequent patterns that may have critical effects as a consequence. Existing approaches for exception rule mining usually handle "Itemsets databases", where transactions are organized without temporal information. However, temporality may be inherent to some real contexts and should be considered to improve the semantic quality of results. Moreover, these approaches implement a non-discriminatory support measure to estimate the relevance of an item, thus interpreting a large volume of data that may be merely occasional as patterns. Aiming to overcome these drawbacks, we propose `TRiER` (*TempoRal Exception Ruler*), an efficient method for mining temporal exception rules that not only discover exceptional behaviors and their causative agents, but also identifies how long consequences take to appear. We also present a new support measure to manipulate time series. This metric considers the context in which a pattern occurs, thus incorporating more semantics to the results. We performed an extensive experimental analysis in real multivariate time series to verify the practical applicability of `TRiER`. Our results show `TRiER` has lower computational cost and is more scalable than existing approaches while finding a succinct and relevant set of patterns.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords: Association Rule, Data Mining, Exception Rule, Time Series

1. INTRODUCTION

Time series occur in countless domains including economy, health and agribusiness. They are continuously produced and stored, generating a large volume of data. This scenario motivates the development of effective and efficient data mining methods to process, analyze and extract useful knowledge from these data. We are particularly interested in association rule mining [Agrawal et al. 1993]. This task has many practical applications since it expresses knowledge in an intuitive way for domain experts. The common purpose is to discover frequent and reliable relationships, called **strong rules**. An example of strong rule could be "with the help of antibiotics, the patient tends to recover" and it is noted as *antibiotics* $\rightarrow$ *recovery*. We call *antibiotics* as antecedent of the rule and *recovery* as consequent.

Although strong rules may be of interest to find unobserved frequent patterns, it is not applicable to discover hidden infrequent ones. Few approaches deal with the extraction of infrequent knowledge. We focus on those proposals that allow obtaining unusual and unexpected information, called exception rules [Suzuki 1996; Hussain et al. 2000]. In general, an exception rule is related to an association rule since for searching an exception we have to find an attribute, i.e. an agent, changing the consequent of a strong rule. An example of exception rule could be "the use of antibiotics in a patient with staphylococcus may lead to death" and can be noted as *antibiotics and staphylococcus* $\rightarrow$ *death*,

This research was supported by CNPq.

where *staphylococcus* is the agent causing the exception. Mining strong and exception rules explains the agents perturbing typical behaviors. As a result, previous control actions can be implemented in scenarios where unexpected agents have critical consequences. An example of prior control action to avoid staphylococcus is to properly maintain the patient hygiene in the hospital.

Existing methods for exception rule mining have some particularities that may not support the characteristics of real, large and complex databases, such as time series. Specifically, such methods only handle univariate data and consider exceptions are caused by a single agent (univariate agent). Moreover, they manipulate Itemset databases, where transactions have no temporal organization. However, temporality may be inherent to some real contexts and should be considered to improve the semantic quality of results. Another limitation is the high computational complexity (of exponential order) and the implementation of a non-selective support measure to estimate the relevance of a pattern. In some cases, the number of mined patterns is so large that it requires a second-order data mining task to select significant patterns and eliminate inconsistent and unnecessary ones.

Aiming to overcome these drawbacks, we propose `TRiER` (*TempoRal Exception Ruler*), a new method for mining temporal exception rules. `TRiER` differs from existing methods on the following aspects:

(1) it handles multivariate time series and discovers exceptions caused by more than one agent (multivariate ones).
(2) it allows the agent causing the exception to occur in a time lag. A time lag indicates a delay, in units of time, between the beginning of the antecedent and the end of the consequent of the rule.
(3) it proposes and implements a new support measure which is more selective than the classic one. As a consequence, our method produces a succinct and relevant set of rules, thus facilitating the understanding and interpretation of the domain experts.
(4) it is faster and more scalable than the current state of the art for exception rule mining, while finding more semantically complete rules regarding temporality.

This work is an extension of the paper "TRiER: A Fast and Scalable Method for Mining Temporal Exception Rules", published in the 34th Brazilian Database Symposium (SBBD 2019). As main differences we highlight an extended theoretical foundation about time series representation and further details of the proposed method, focusing on a new support measure especially designed for temporal data. In addition, we expanded our experimental analysis to validate our method in real datasets of agrometeorology and El-Niño phenomenon. Our results reveal that `TRiER` is faster and more scalable than related methods while finding a representative set of meaningful rules.

The remainder of the paper is organized as follows. Section 2 summarizes background concepts. Section 3 presents related work and their characteristics. In Section 4 we describe `TRiER` and the proposed support measure. Section 5 presents our experiments comparing `TRiER` with related methods. Finally, Section 6 concludes with the main contributions of the paper.

2. BACKGROUND

2.1 Time Series

Time series is a sequence of time-ordered observations with regular time intervals between each pair of observations [Mitsa 2010]. A time series is defined as $S = \{s_1, s_2, \dots, s_m\}$, where s_i for $i \in \{1, \dots, m\}$ corresponds to the occurrence of s_i at time t_i. A univariate time series is created by only one underlying variable, such as the dollar quote series. On the other hand, a multivariate time series is composed of several variables at each time t_i. An example is a weather time series with variables of rainfall, maximum temperature and air humidity in each observation. We formally represent each observation s_i of a multivariate time series as $s_i = \{s_{i1}, \dots, s_{iD}\}$, where s_i is a set with D variables.

Time series can be rewritten in a more concise and manipulable representation for data mining algorithms. There are several methods for time series representation, many of them with the objective of discretizing the data. Discretization is the process of mapping continuous values into discrete ones. Although discretization loses some details of the original data, discretized data can be more significant and easier to interpret, contributing to a more consistent representation of the results in different data mining tasks [Casanova et al. 2017].

The simplest discretization methods apply the equal-width and equal-frequency approaches. Equal-width methods divide the values so that all ranges have the same width. In contrast, equal-frequency ones divide the values so that all ranges have the same frequency of values. Thus, for a total of P data, ranging from a to b, equal-width discretization produces $\frac{b-a}{q}$ size ranges, where q is the desired number of ranges, while the equal-frequency approach creates ranges with $\frac{P}{q}$ data in each of them. Figure 1 shows an example of equal-width and equal-frequency discretizations for 155 days of rainfall, in which the equal-width approach divides the data into 8 ranges of width 10. In the first range, for instance, rainfall varied between 0 and 10mm during 5 days. In contrast, the equal-frequency approach divides the data into 5 ranges of different widths, so that all ranges have the same frequency, which is 31 days. Considering the first range, rainfall varied between 0 and 24mm during 31 days, while in the second one rainfall varied between 24 and 38mm in the same period.

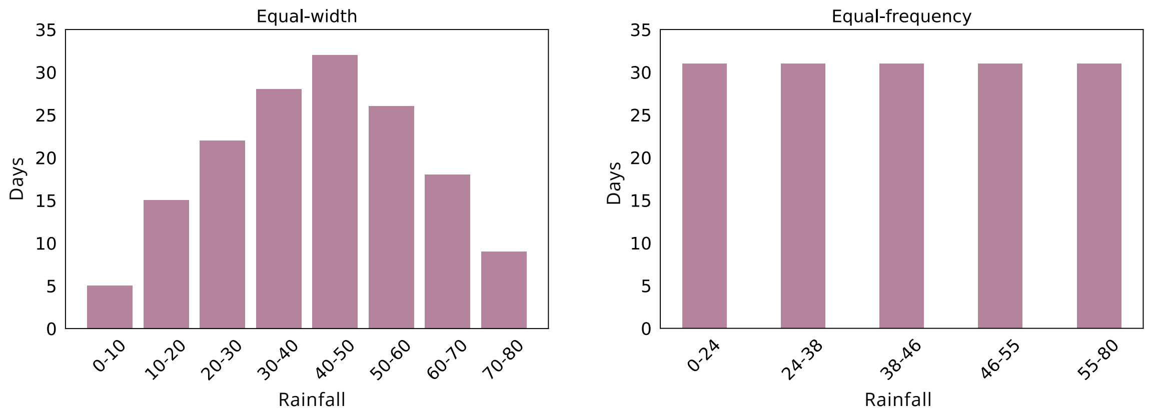

Fig. 1. Equal-width and Equal-frequency discretizations.

In the context of this work, the representation method must meet the following premises:

(1) Do not group infrequent patterns, as it distorts the frequency at which they occur. If it happens, exceptional and frequent patterns will have the same probability of occurring, affecting the quality and semantics of discovered rules.
(2) Do not compress time series observations, that is, avoid representing a set of observations by some criterion, such as the average, in a single observation.
(3) Act independently on each variable, such that it is possible to discern the meaning of each symbol for each variable of a multivariate time series.

The equal-frequency approach and classical methods dictated by the data, such as clipping [Mitsa 2010] tend to group infrequent and frequent patterns and do not meet the first premise. This grouping assumes all values in the range have the same probability of occurring, thus diluting unusual patterns. Non-adaptive methods, such as PAA [Keogh et al. 2001] and SAX [Lin et al. 2007], reduce the number of observations in a time series, compressing them by some criterion and do not satisfy the second premise. As the equal-width approach satisfies the three defined premises, it will be used in this work.

Although there are other approaches that meet the premises described, equal-width stands out for being intuitive and simple to implement.

2.2 Association Rules

Given a set I (set of items) and a database DB composed of a set of transactions T, each one being a subset of I, association rules are implications in the form $X \rightarrow Y$ that relate the presence of Itemsets X and Y in transactions of DB, assuming that $X, Y \in I$, $X \cap Y = \varnothing$ and $X, Y \neq \varnothing$. We call X as **antecedent** and Y as **consequent** of the rule. The classic measures to assess association rules are support (*supp*) and confidence (*conf*). Support calculates the frequency a rule occurred (*freq*) considering the number of transactions T in the database. Equation 1 defines the support of $X \rightarrow Y$.

$$supp(X \rightarrow Y) = \frac{freq(X \cup Y)}{|T|} \tag{1}$$

Confidence measures the probability of the consequent occurring in transactions that also contain the antecedent. Equation 2 represents the confidence of $X \rightarrow Y$.

$$conf(X \rightarrow Y) = \frac{freq(X \cup Y)}{freq(X)} \tag{2}$$

Given the minimum thresholds of support (*minsupp*) and confidence (*minconf*) informed by the user, we say $X \rightarrow Y$ is frequent if $supp(X \rightarrow Y) \geq minsupp$ and confident if $conf(X \rightarrow Y) \geq minconf$. Moreover, $X \rightarrow Y$ is **strong** if it is frequent and confident. An alternative to support-confidence was proposed in [Berzal et al. 2002] where the accuracy is measured by means of certainty factor (cf). Certainty factor aims to solve some of the confidence drawbacks. In particular, the support-certainty factor measures reduce the number of obtained rules, filtering those corresponding to statistical independence or negative dependence. As a consequence, extracted rules are stronger than those obtained with support-confidence.

Certainty factor ranges from -1 to 1 and measures how the probability of Y being in a transaction changes when it is known that X also is. Positive values indicate our belief increases, negative values mean it decreases and 0 means no change. Analogously, we say $X \rightarrow Y$ is certain if $cf(X \rightarrow Y) \geq mincf$, where $mincf$ is the minimum threshold for certainty factor informed by the user. Equation 3 defines the certainty factor of $X \rightarrow Y$.

$$cf(X \rightarrow Y) \begin{cases} \dfrac{conf(X \rightarrow Y) - supp(Y)}{1 - supp(Y)} & \text{if } conf(X \rightarrow Y) > supp(Y) \\ \dfrac{conf(X \rightarrow Y) - supp(Y)}{supp(Y)} & \text{if } conf(X \rightarrow Y) < supp(Y) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

3. RELATED WORK

The task of association rules is based on finding frequent and reliable relationships in databases. In this context, a database is defined as a collection of transactions, each one consisting of a set of items. Thus, to discover an association rule it is necessary to find items that cause the presence of others in the same transaction. Traditional methods for association rules relate items disregarding their occurrence orders [Agrawal et al. 1993]. However, time information is relevant for some applications and should be analyzed to better understand semantic issues. Thus, sequence mining was introduced in [Agrawal and Srikant 1995] to find sequences of items considering the order they occur. There are several approaches for sequence mining in literature. We thus discuss some of the most explored ones and those closely related to our work.

Classic sequence mining algorithms were based on *Apriori* [Agrawal et al. 1993], like *GSP* (Generalized Sequential Patterns) [Srikant and Agrawal 1996]. Apriori-like algorithms usually include candidate generation and validation steps, performed with support and confidence counts. A limitation of Apriori-like approaches is the need for support calculations through a complete database scan, comparing each item with all transactions. Aiming to reduce the processing time, approaches based on vertical format, as *ECLAT* [Zaki 2000] and *SPADE* [Zaki 2001], and on pattern growth concept, as *Prefix-Span* [Pei et al. 2001], were proposed.

Prefix-Span (Prefix-projected Sequential Pattern Growth) [Pei et al. 2001], for example, does not require a candidate generation step. The database is recursively projected into smaller parts and frequent sequences are directly extended to create larger ones. In general, sequence mining methods look for sequences considering the order in which items occur and do not establish cause and consequence relationships. In addition, they are pseudo-polynomial in time complexity [Dong 2009] and implement a non-discriminatory support measure. Therefore, when applied to mine sequences with low support thresholds, these algorithms generate an overwhelming amount of sequences in unfeasible time.

Low support thresholds concern the mining of infrequent patterns, that might be relevant to some applications. We focus on those proposals that allow obtaining some unexpected and uncommon information, called exception rules. In general, these approaches are able to manage rules that, being infrequent, provide a specific domain usually delimited by a strong rule. Exception rules were first defined as rules that contradict the user's common belief. It means that for searching an exception we have to find an attribute (also called agent) that changes the consequent of a strong rule. In general terms, the kind of knowledge an exception rule discovers can be interpreted as follows.

*"X **strongly** implies Y (and not E), but in conjunction with E, X **does not** imply Y".*

Concepts and research work on exception rule mining are presented in [Suzuki 1996] and [Hussain et al. 2000]. According to Suzuki, an exception rule is formally defined as:

$$\begin{gathered} X \rightarrow Y \text{ (high } supp \text{ and high } conf - \text{strong rule).} \\ X \wedge E \rightarrow \neg Y \text{ (low } supp \text{ and high } conf - \text{exception rule).} \\ X \nrightarrow E \text{ (high } supp \text{ and high } conf - \text{reference rule).} \end{gathered}$$

$X \rightarrow Y$ is a strong rule and indicates a common behavior. $X \wedge E \rightarrow \neg Y$ is an exception rule and shows that the presence of item E has modified the expected consequent of the strong rule, that is, E is the agent causing the exception ($\neg Y$). $X \nrightarrow E$ is a reference rule and determines the antecedent of the rule should have no association with the agent causing the exception. A similar definition is presented in [Hussain et al. 2000]. The difference lies in the reference rule, where the agent causing the exception may have association with the unexpected behavior ($E \rightarrow \neg Y$).

Recalling staphylococcus example, a valid set of rules based on Suzuki's definition would be:

Strong Rule: with the use of antibiotics the patient tends to recover.
Exception Rule: the use of antibiotics in patients with staphylococcus can lead to death.
Reference Rule: patients taking antibiotics do not acquire staphylococcus bacteria.

Considering Hussain's definition, only the strong and exception rules would be the same. A valid example of a reference rule in this application domain would be:

Reference Rule: patients taking antibiotics can acquire staphylococcus bacteria.

Different definitions for exception rule mining focus only on finding unusual and contradictory behavior [Daly and Taniar 2008], while Suzuki's and Hussain's also allow inferring the agent causing the exceptional behavior.

The Exception Rule Search Algorithm (*ERSA*) [Calvo-Flores et al. 2011] is based on the definition introduced in [Suzuki 1996] but it does not use a reference rule. Authors argue this rule does not offer a semantic enrichment when defining exceptions and reformulate the definition as the pair of strong and exception rules as follows.

$$X \rightarrow Y \text{ (frequent and certain } - \text{ strong rule).}$$
$$E \rightarrow \neg Y \text{ (certain in the domain of the strong rule antecedent } - \text{ exception rule).}$$

ERSA basic operation is described below. The transaction database is first converted into a binary format to search the set of frequent items and select strong rules. *ERSA* maintains a single set with all candidate rules. Thus, for each rule $X \rightarrow Y$, *ERSA* completely scans this set (which tends to be large) to search for a valid exception, that is, a rule whose antecedent contains X and the consequent is different from Y. *ERSA* applies confidence or certainty factor to filter relevant rules. Its complexity is $O(Trl2^l)$, where T is the number of transactions in the database, l is the quantity of items and r is the number of discovered rules.

Recent methods for exception rules are based on fuzzy logic, as the Fuzzy Exceptional Rule Search Algorithm (*FERSA*) [Ruiz et al. 2016]. Although it is similar to *ERSA* in development and implementation terms, fuzzy logic is applied to avoid problems of inaccuracy and data inconsistency.

To the best of our knowledge, there is no method in literature to find exception rules caused by more than one agent. Recall staphylococcus example: although a combination of agents may cause the patient death, related methods only recognize staphylococcus as causative agent. Moreover, those methods do not consider temporality in the exception rule mining process.

4. THE PROPOSED METHOD

We propose `TRiER` (*<u>T</u>empo<u>R</u>al <u>E</u>xception <u>R</u>uler*), a fast and scalable method for mining temporal exceptions rules and their corresponding strong rules. Mining both rules allows a better understanding of the agents that perturb the strong rule's usual behavior. `TRiER` expands the state of the art on exception rule mining by discovering unusual knowledge caused by multivariate agents and revealing how long the consequences take to appear. The kind of knowledge `TRiER` discovers is described below.

"X strongly implies Y after a time t. But X and E (even if E occurs later) implies something different from Y after a time v".

Recall staphylococcus example, a valid set of rules considering `TRiER`'s definition is:

Strong Rule: with the use of antibiotics the patient tends to recover in two weeks.
Exception Rule: the use of antibiotics in patients who acquire staph bacteria after 5 days may lead to death in the next month.

`TRiER` effectively deals with univariate and multivariate time series and implements the concepts of window and time lag. A **window** (w) of a time series S is a sequence of events that occurs in a continuous interval, starting at time t_b and ending at time t_e, such that events t_b and t_e belong to w. The window is used to restrict the maximum size of the sequential pattern, that is, the maximum number of consecutive observations the pattern can have. **Time lag** (t_{lag}) is a delay between the beginning of the antecedent and the end of the consequent. Time lag follows the temporal granularity of time series, such as days, months or years.

We divide our method in three main steps: (1) *Sequence Mining*, (2) *Rules Discovery* and (3) *Exception Rule Mining*, as illustrated in Figure 2. Additionally, TRiER requires some representation (discretization) method for continuous data and pre-processing activities to treat missing values and noise, for example.

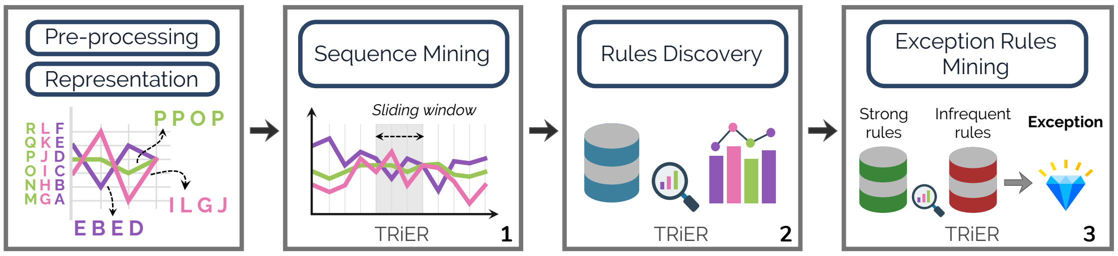

Fig. 2. TRiER overview.

TRiER does not restrict a representation method, but the application user must consider the infrequent nature of the patterns to be mined in order to choose an appropriate method, i.e., a method that fulfills the three premises presented in Section 2.1. We also suggest to apply a missing value treatment method, such as interpolation. Otherwise, there may be a semantic loss in the patterns obtained. Once pre-processing and representation activities are performed, the TRiER steps can be effectively started, as described in the following sections.

4.1 Sequence Mining

In *Sequence Mining* we create sequences that will generate strong and infrequent rules. It is the most expensive step, as it consists of a combinatorial process. We apply two parameters in the discretized dataset: the window size (w) and the minimum support for a sequence to be considered frequent.

One of the main contributions of TRiER is the proposal of a new support measure. The classic support measure (Equation 1) is not intended for manipulating time series and, therefore, it may generalize information about the pattern's occurrence. In order to better understand the context in which a pattern occurs, the measure we propose considers two scenarios:

(1) Number of times a pattern occurs.
(2) Number of time series containing the pattern.

The classic measure makes no distinction from the first scenario. It means that if a pattern occurs numerous times or only once in a time series, it will have the same contribution in support calculation. This assumption represents a semantic loss, because when manipulating temporal data it is relevant to understand how the pattern occurs at different observations of a time series.

The second scenario allows to understand how the pattern occurs in the entire database. Specifically, if a pattern occurs numerous times in a single series, but does not occur regularly in the database, it may indicate a very specific behavior in that time series. Therefore, considering it as a pattern may be inconsistent.

Thus, our new measure ($supp_{TRiER}$) considers these scenarios to establish a more comprehensive view of the occurrence of the pattern. Equation 4 formalizes our measure: L is the number of time series in the database, P is the number of observations in the database (the sum of the observations of each time series that makes up the database), j is the number of time series in which a sequence occurs and f is the number of times a sequence appears.

$$supp_{TRiER} = \frac{f}{P} \cdot e^{\left(\frac{j}{L}-1\right)} \quad (4)$$

The factor $\frac{f}{P}$ represents the probability of the pattern occurring in the database, while the exponential varies according to the probability of the pattern occurring in a time series. As the function e^x is strictly increasing, $supp_{TRiER}$ increases as the number of series in which the pattern occurs also increases, that is, if two patterns have the same probability of occurring in the database, the pattern with the greatest support will be the one that appears in more time series.

Algorithm 1: `TRiER` sequence mining

```
Input  : Discretized data, Window ($w$), Minimum support ($minsupp_{TRiER}$)
Output:
Frequent Sequences ($SeqFreq$)
1  begin
2    foreach (discretization symbol $\alpha$) do
3      Support calculation
4      if ($\alpha$ support $\geq minsupp_{TRiER}$) then
5        $SeqFreq \leftarrow \alpha$

6    foreach (sequence $seq_i \in SeqFreq$) do
7      foreach (sequence $seq_j \in SeqFreq$ with the same number of variables as $seq_i$) do
8        $seq_{new} \leftarrow seq_i \cup seq_j$
9        if (total of variables of $seq_{new}$ = total of variables of $seq_i + 1$) then
10         Support calculation
11         if ($seq_{new}$ support $\geq minsupp_{TRiER}$) then
12           $SeqFreq \leftarrow seq_{new}$

13   foreach (sequence $seq_{d+1} \in SeqFreq$) do
14     foreach (sequence $seq_{o+1} \in SeqFreq$ with the same number of observations as $seq_{d+1}$) do
15       $seq_{new} \leftarrow seq_{d+1} + seq_{o+1}$
16       if (total of observations of $seq_{new}$ = total of observations of $seq_{d+1} + 1$) then
17         Support calculation
18         if ($seq_{new}$ support $\geq minsupp_{TRiER}$) then
19           $SeqFreq \leftarrow seq_{new}$
```

Algorithm 1 describes Sequence Mining step, which is composed of three tasks: (1) mining sequences with one variable (lines 1 to 5), (2) mining sequences with n variables and one observation (lines 6 to 12) and (3) mining sequences with n variables and p observations (lines 13 to 19). In the first task, we calculate the support of each representation symbol, selecting those that meet the minimum support threshold ($minsupp_{TRiER}$). In the second task, we create sequences with one observation and n variables. To create a frequent sequence with n variables, we join two frequent sequences seq_i and seq_j with $n-1$ variables, so that the intersection of such sequences has $n-2$ variables. Specifically, the junction of seq_i and seq_j will compose the set of frequent sequences ($SeqFreq$) if seq_{new} is frequent and has the number of variables of seq_i +1.

In the last task, we obtain sequences with multiple observations and variables. We thus join two frequent sequences seq_{d+1} and seq_{o+1} with $p-1$ observations to form a candidate with p observations, so that the first $p-2$ observations of a sequence match the last $p-2$ observations from the other sequence. This task ends when we reach the window size.

As mentioned earlier, Sequence Mining is a costly step. In our Java implementation of `TRiER`, our strategy to improve the efficiency of this step is described as follows. Initially, time series are stored in a map structure, called **Hash Map**. This structure is composed of key and value sets that permit to return objects quickly, with constant complexity. Keys are formed by items and an item indicates an observation in a time series. The value of each key is a **Tree Set** composed of pairs in the format $(i{:}j)$, where j is a time series identifier containing the key in the instant i. Tree Set implements a red black tree to sort $(i{:}j)$ pairs and performs the most common operations in logarithmic complexity.

The Sequence Mining step is $O((wLmD^w)^2 log(Lm))$, where w is the window size, L is the number of time series in the database, m is the number of observations in a time series and D is the number of variables in each observation. Although it is a high complexity, TRiER is exponential in relation to the window size, while related works are exponential in relation to the number of items/sequences discovered, a potentially much larger number.

4.2 Rules Discovery

In this step we extract strong and infrequent rules with time lag (if any) from sequences previously mined. From a sequence with p observations it is possible to extract rules with time lag ranging from 1 to $p-1$. If time lag is 1, the antecedent and the consequent of the rule occur in subsequent observations, respectively. If time lag is $p-1$, the antecedent is in the first observation and the consequent in the last observation. For each time lag possibility, potential rules are generated. From sequence $P\ S, P\ L, E, P\ R$, for example, it is possible to extract the following rules:

—$P\ S, P\ L, E \rightarrow PR$ (rule with time lag 1, since $P\ R$ occurs immediately after $P\ S, P\ L, E$).
—$P\ S, P\ L \rightarrow P\ R$ (rule with time lag 2, because $P\ R$ occurs two units of time after $P\ S, P\ L$).
—$P\ S \rightarrow P\ R$ (rule with time lag 3, since $P\ R$ occurs three units of time after $P\ S$).

Traditional measures for rules validation (confidence and certainty factor) are composed of the support metric. Since TRiER proposes and implements a new support measure, confidence and certainty factor implemented by our method follow the new measure and are respectively called $conf_{TRiER}$ and cf_{TRiER}. Thus, rules possibilities are evaluated with the following parameters:

(I) Minimum support of the strong rule ($minsupp_{TRiER-str}$).
(II) Minimum support of the infrequent rule ($minsupp_{TRiER-inf}$).
(III) Maximum support of the infrequent rule ($maxsupp_{TRiER}$).
(IV) Minimum confidence ($minconf_{TRiER}$).
(V) Minimum certainty factor ($mincf_{TRiER}$)

Figure 3 illustrates how we classify strong and infrequent rules. A rule is **strong** if its support-confidence or support-certainty factor meets these minimum thresholds (parameters I, IV or V).

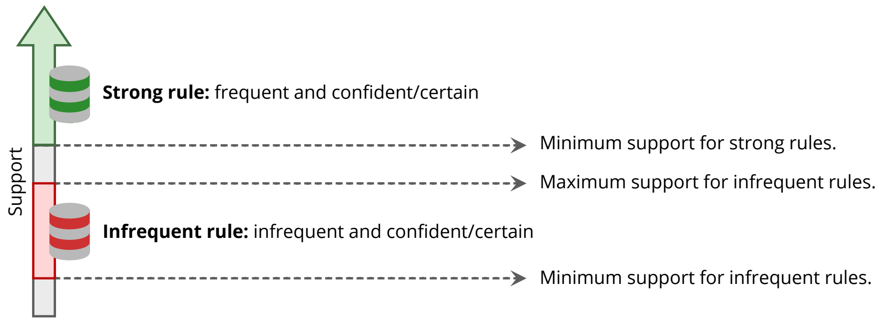
Strong rule: frequent and confident/certain
Minimum support for strong rules.
Maximum support for infrequent rules.
Support
Infrequent rule: infrequent and confident/certain
Minimum support for infrequent rules.

Fig. 3. Classification of strong and infrequent rules.

In addition, a rule is **infrequent** if it meets a minimum support threshold for infrequent rules ($minsupp_{TRiER-inf}$) and its support is less than a maximum support ($maxsupp_{TRiER}$). We apply a maximum support threshold to assure the support of infrequent rules does not exceed the $minsupp_{TRiER-str}$ of strong rules.

4.3 Exception Rule Mining

Once the rules are classified in strong and infrequent we start the *Exception Rule Mining* step, where we consider a different definition for exception rules. The main difference of our definition to literature ones is that we consider an exception rule as infrequent. We argue those rules are naturally atypical and indicate an unusual behaviors, so the maximum support of exception rules should not exceed the minimum support of a strong rule. Thus, an exception rule is formalized as follows.

$$\begin{gathered} X \to Y \text{ (frequent and confident/certain } - \text{ strong rule)} \\ X \wedge E \to \neg Y \text{ (infrequent and confident/certain } - \text{ exception rule)} \end{gathered}$$

For each strong rule found, the set of infrequent rules is analyzed to identify a rule that meets the following restrictions: (1) The antecedent of the infrequent rule must contain the antecedent of the strong rule and (2) the consequent of the infrequent rule must not contain or be included in the consequent of the strong rule. The infrequent rule satisfying those constraints indicates an exception to the analyzed strong rule.

We remark the agent causing the exception may occur in a time lag, as long as it occurs before the rule consequent. We also enable exceptions to be caused by more than one item. Algorithm 2 summarizes the main idea of Rules Discovery and Exception Rule Mining steps.

Algorithm 2: TRiER rules discovery and exception rule mining

```
Input  : Frequent sequences, $minsupp_{TRiER-str}$, $minsupp_{TRiER-inf}$, $maxsupp_{TRiER}$, $minconf_{TRiER}$ or
         $mincf_{TRiER}$
Output: Temporal exceptions and their corresponding strong rules
1  begin
2      foreach sequence $seq \in$ freqSequences do
3          for ($p = 0$ to $p \leq seqsize - 2$) do
4          generate rules from the current sequence with time lag $t_{lag}$
5          filter generated rules using $mincf_{TRiER}$ or $minconf_{TRiER}$
6          if (rule support $\geq minsupp_{str}$) then
7              strong rules ← rule
8          else if (rule support $\geq minsupp_{inf} \wedge$ rule support $\leq maxsupp$) then
9              infrequent rules ← rule
10     foreach strong rule $str \in$ strong rules do
11         foreach infrequent rule $inf \in$ infrequent rules do
12             if ($str$ antecedent $\subset inf$ antecedent $\wedge$ $str$ consequent $\not\subset inf$ consequent) then
13                 exception rules ← strong rule and its exception rule
```

The resulting complexity of Rules Discovery and Exception Rule Mining steps is $O(yw)^2$, where y is the number of mined sequences and w is the window size. As the number of frequent sequences is reduced due to the new support measure, these steps are performed quickly and this is the main difference in terms of efficiency and effectiveness between TRiER and competing methods.

5. EXPERIMENTAL ANALYSIS

Our experimental results on real data show the usefulness, effectiveness and efficiency of `TRiER` when compared to related methods. We first analyze the relevance of sequences mined by `TRiER` and *Prefix-Span*, one of the most efficient methods of sequence mining. Finally, we compare `TRiER` with the state of the art method for exception rules, *ERSA*. Experiments were performed on an Intel Core i7, 3.40 GHz computer with 16 GB of RAM and a SATA hard disk.

5.1 AgroData and El-Niño Datasets

The experimental study in real data aims to investigate whether `TRiER` and related methods can offer effective and efficient support for discovering knowledge in agrometeorology and climate. We thus performed experimental analysis on two datasets, namely AgroData and El Niño. This study can help domain experts gain new insights into common and exceptional behaviors.

For our first dataset we chose data from agrometeorology, since agriculture plays a fundamental role in the economy of several countries. In Brazil, for example, agribusiness accounts for 23% of GDP and 40% of the labor. Mining exception rules in agrometeorology may help us to identify which climatic conditions most affect crops and how quickly implications arise. As a result, previous control actions could be carried out in regions with similar characteristics to minimize severe impacts caused by extreme weather, as droughts. Our analysis focus on sugarcane, due to its importance for ethanol production and Brazil's economy.

We then constructed AgroData, a multivariate time series dataset combining climatic and vegetative index variables. We collected climatic time series from the National Institute of Meteorology[1] (INMET) and they are composed of rainfall, maximum and minimum temperatures variables.

Vegetation index data were obtained with SATVeg[2], a tool maintained by Embrapa[3] that extracts NDVI time series from TERRA/MODIS satellite images. NDVI (Normalized Difference Vegetation Index) is a widely used index in agricultural research that represents the soil vegetative vigor. NDVI ranges from -1 to 1: values close to 1 indicate strong vegetative activity while negative or close to 0 values describe regions where there is weak or no chlorophyll activity. We collected NDVI time series of sugarcane from 2014 to 2018. Time series are composed by 60 monthly observations from the state of Sao Paulo, the largest sugarcane producing state in Brazil.

In order to compose AgroData dataset, the climatic data were associated with NDVI ones considering the restriction a sugarcane record must be at most 70 km distant (geodesic distance) of a meteorological station. The resulting dataset is multivariate and composed of 60 monthly observations of NDVI, rainfall, maximum and minimum temperatures ($MaxTemp$ and $MinTemp$, respectively), from 2014 to 2018. Figure 5.1 illustrates the mapping for AgroData, in which sugarcane regions are highlighted in green and the weather stations monitoring them are symbolized by purple stars.

Our second dataset includes the El Niño data available in Kaggle[4]. The dataset is originally composed of daily ocean and meteorological readings salted by buoys positioned throughout the equatorial Pacific from 1980 to 1998 and has the information of intensity of southern winds, relative humidity, air temperature and sea surface temperature. In our experiments we remodeled the temporal granularity of the data to monthly, since cause and consequence relationships in climatic data are not instantaneous. We selected the period from 1997 to 1998 as research period because of the strong intensity of the phenomenon in these years. The resulting dataset is composed of 12 monthly observations

[1]http://www.inmet.gov.br/portal/
[2]https://www.satveg.cnptia.embrapa.br/satveg/
[3]https://www.embrapa.br/
[4]https://archive.ics.uci.edu/ml/datasets/El+Nino

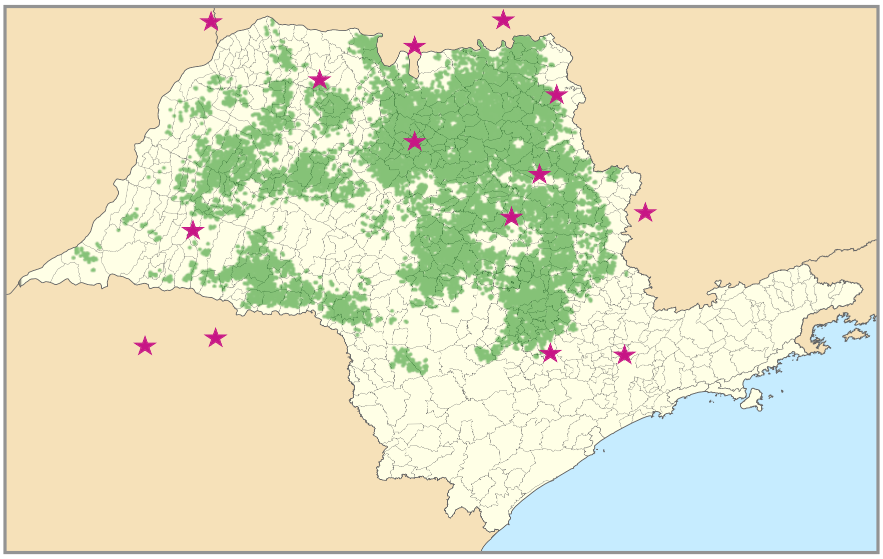

Fig. 4. Weather stations and sugarcane areas in Sao Paulo.

of intensity of southern winds $VentMer$, humidity $Umid$, air temperature $TemSup$ and sea surface temperature $TempMar$. Table I describes AgroData and El-Niño datasets.

Table I. Datasets descriptions.

Dataset	Time series size (m)	Variables (D)	Dataset size (L)
AgroData	60 monthly observations	$MaxTemp$, $MinTemp$, $NDVI$ and rainfall	$24,420$ time series
El-Niño	12 observations	$VentMer$, $Umid$, $TempMar$, $TempSup$	$71,885$ time series

5.2 Sequence Mining

This experiment aims to investigate the efficiency of *Prefix-Span* and `TRiER` in the most costly phase of the mining process, i.e. sequence mining. Accordingly, we measure running time as we increase the window size (w) and the minimum support threshold. As the temporal granularity of series is monthly, the window size is given in months ($w = 1$, for instance, means one month). The implementation of *Prefix-Span* used is available in SPMF[5] tool, where the method is implemented in Java. This tool is a reference in pattern mining, specifically in Sequence Mining and Itemsets [Gan et al. 2019].

Each algorithm was executed 5 times and Table II summarizes their average time in minutes for AgroData. The green data represent the method which performed better and the red data indicate the opposite. The maximum window size evaluated is 5 due to an efficiency bottleneck of *Prefix-Span*, as after this threshold its processing time grows abruptly. Thus, the evaluation of larger windows would make the execution of this experiment unfeasible.

As we increase the window size the performance of *Prefix-Span* degrades, as shown in Figure 5. The reason is that *Prefix-Span* implements a non-discriminatory support measure and if an item occurs only once or countless times, it has the same contribution in support calculation. As a consequence, it generates an overwhelming number of sequences, which include a large amount of irrelevant and unnecessary information. The increase in the window size is directly related to the sequence size. As Sequence Mining is a combinatorial task, larger sequences require more execution time.

[5]http://www.philippe-fournier-viger.com/spmf/

Table II. Time spent in sequence mining for AgroData dataset, in minutes.

Minimum Support	$w = 2$		$w = 3$		$w = 4$		$w = 5$	
	Prefix-Span	TRiER	*Prefix-Span*	TRiER	*Prefix-Span*	TRiER	*Prefix-Span*	TRiER
0.05	**0.067**	7.416	**1.494**	19.166	34.324	**26.5**	811.613	**29.75**
0.1	**0.066**	3.5	**1.428**	6.916	34.319	**8.583**	806.415	**10.2**
0.15	**0.066**	2.25	**1.427**	3.833	33.906	**4.333**	784.452	**6.3**
0.2	**0.065**	1.116	**1.421**	1.916	33.346	**1.935**	735.166	**3.383**
0.3	**0.065**	0.75	1.352	**0.8**	32.744	**0.816**	648.352	**1.25**
0.4	**0.064**	0.383	1.319	**0.416**	32.380	**0.4666**	565.446	**0.65**

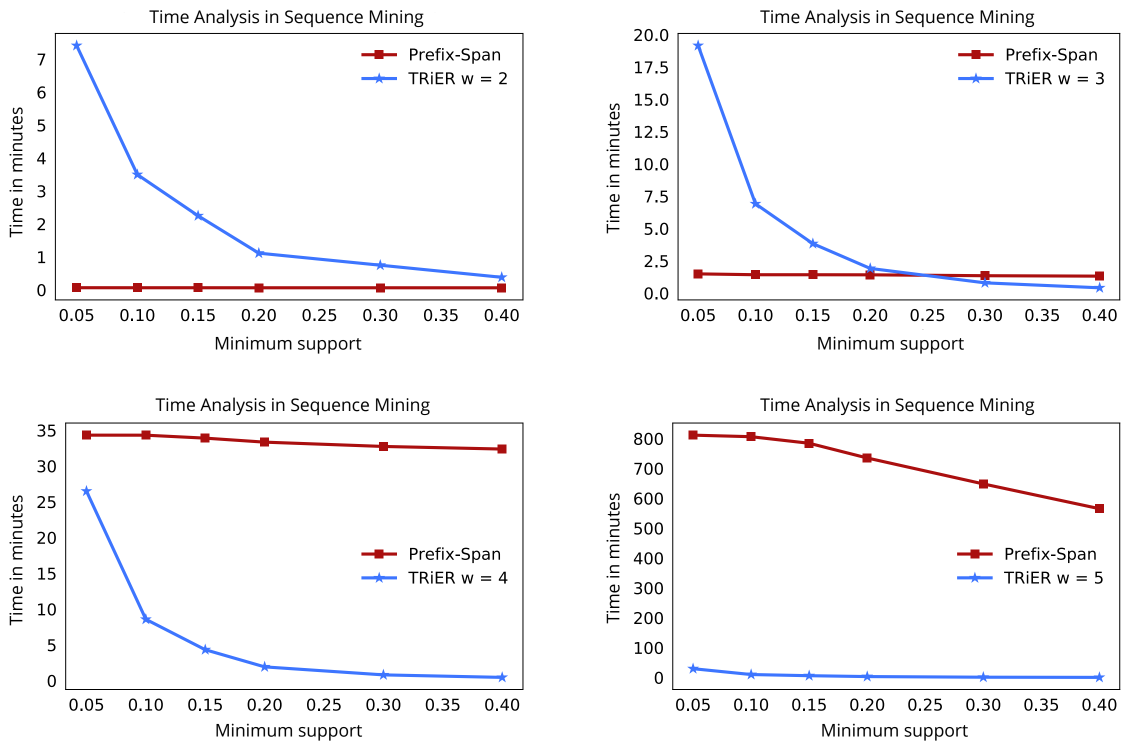

Fig. 5. AgroData dataset: time spent in sequence mining, in minutes.

For small windows, TRiER mines sequences at a viable run time, although it does not achieve the performance of *Prefix-Span*, which uses a database projection approach. However, when the window size increases, TRiER's growth rate of time spent is more stable and considerably less than the one of *Prefix-Span*. The efficiency bottleneck of *Prefix-Span* is in mining larger sequences (as size 4) with low support thresholds. Mining of larger sequences are relevant in domains where long-term analysis is desired. Mining low support sequences is useful where it is intended to discover atypical patterns. TRiER, on the other hand, has superior performance in these cases, because it implements a selective support measure that limits the number of mined sequences and, consequently, minimizes the time needed to generate combinations of larger sequences.

Figure 6 illustrates the significant growth in the number of sequences mined by *Prefix-Span* as the window size increases. For small windows (2 and 3) the number of sequences is feasible, 1397 and 50515 respectively. When the window size increases (4 and 5), the number of sequences is immense, 1693404 and 52520544 respectively. As the rules will come from this volume, they will also be huge. Thus, instead of creating knowledge to facilitate expert analysis, this amount of information will require a second-order data mining task to weed out irrelevant patterns and filter necessary ones.

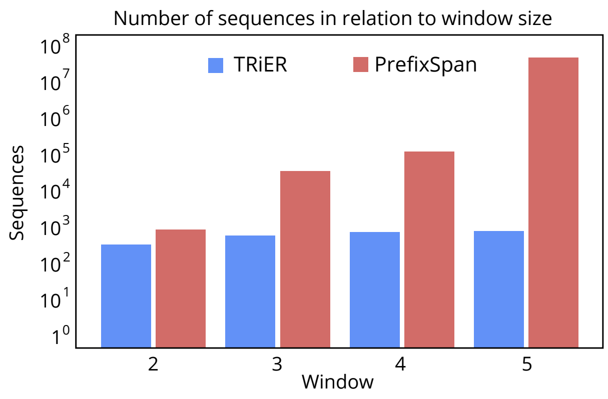

Fig. 6. Sequences generated by `TRiER` and *Prefix-Span* for 5% of minimum support for AgroData dataset.

Figure 7 presents the time spent for *Prefix-Span* and `TRiER` in Sequence Mining for El-Niño data. Similarly to the AgroData data, results show large window sizes and low thresholds of support are efficiency bottlenecks for *Prefix-Span* method.

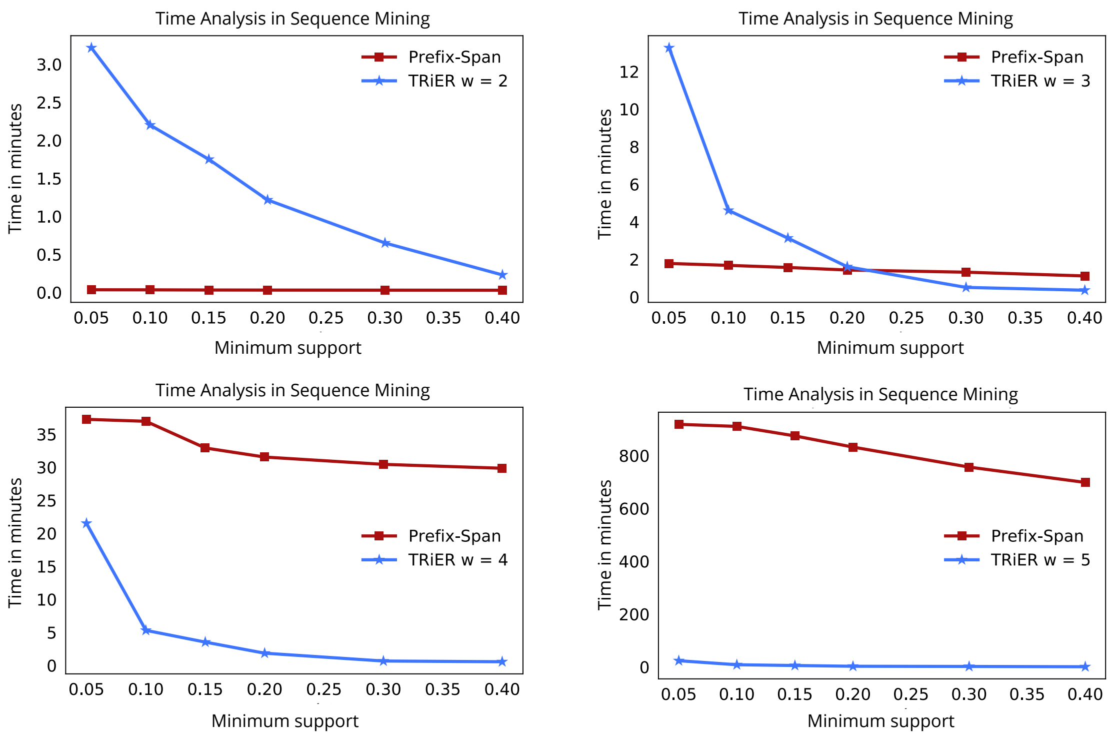

Fig. 7. El-Niño dataset: time spent in sequence mining, in minutes.

The main difference in sequences volume mined by related works and our method is the support measure. The support measure we propose is more selective and significantly reduces the number of mined patterns. The advantage is the knowledge discovered is feasible to manipulate and analyze. Specifically, the new support measure is based on discovering relevant knowledge instead of an overwhelming volume of patterns, which potentially contains a large number of redundant and unnecessary information. Figure 8 shows the number of patterns generated by `TRiER` with the traditional support measure (dark blue) and the proposed measure (light blue) for the AgroData dataset from 2017 to 2018 using different window configurations for 5% of minimum support.

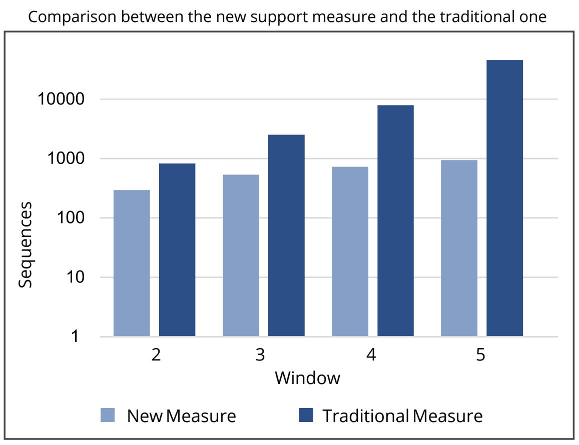

Fig. 8. AgroData dataset: Mined sequences with the traditional support measure and the new one.

We can note the proposed measure generates significantly fewer patterns than the traditional measure. For window 5, for example, the new support measure generates 936 sequences while the traditional measure discovers 45328 sequences.

5.3 Rules Discovery and Exception Rule Mining

This experiment aims to investigate the effectiveness and usefulness of `TRiER` and the state of the art method for mining exceptions, *ERSA*. We analyze results from two perspectives: the time taken to discover rules and their semantic relevance. Figure 9 illustrates time spent by `TRiER` and *ERSA* to discover exception rules in AgroData dataset and Figure 10 represents the results for the El-Niño dataset. `TRiER` was tested with different windows $w = \{2, 3, 4, 5\}$ to analyze how this parameter influences on the rule discovery step. Although *ERSA* generates all the possibilities of frequent Itemsets, we restricted the maximum Itemset size to 5 so that the time taken was not impractical.

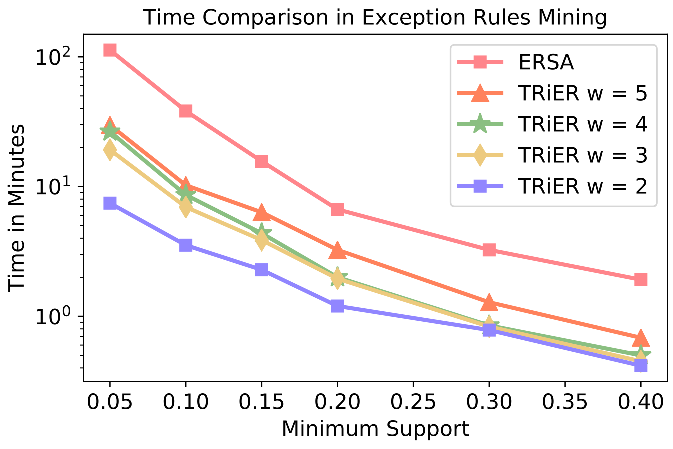

Fig. 9. AgroData data: Time spent by *ERSA* and `TRiER` to discover exception rules, in minutes.

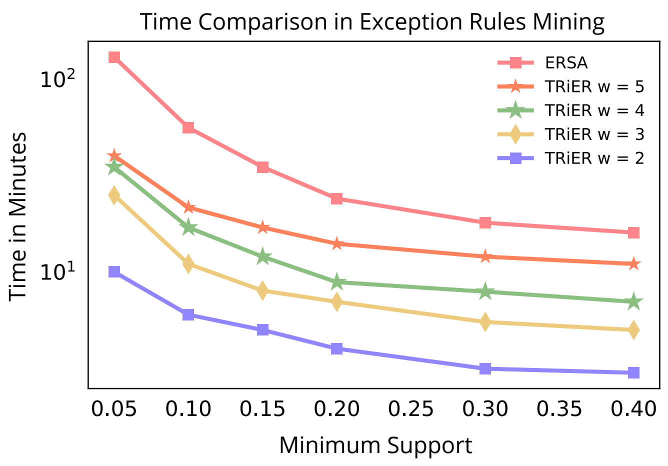

Fig. 10. El-Niño data: Time spent by *ERSA* and `TRiER` to discover exception rules, in minutes.

As *ERSA* generates a huge number of rules, we measured times with 60% of certainty factor for both algorithms. Rules generated only by confidence or lower values of certainty factor exceeded the computational resources available to perform the experiment. Unlike `TRiER` that deals with multivariate data and understands a time series is composed of 60 observations and each observation has 4 variables, *ERSA* does not make this distinction and considers a time series as one Itemset with 240 items, that is, 60 observations of each variable (dimension).

To discover exception rules, *ERSA* first generates frequent Itemsets. The way it was planned *ERSA* would generate 2^x candidates, where x is the number of symbols used in representation. For that reason we limited the maximum Itemset size to 5. Then, the number of possible candidates is the number of sets with up to 5 elements we can form with the number of symbols used in representation.

Another problem that degrades *ERSA* performance is the way it searches for exceptions. The rules are stored in a single set, without distinction of strong and infrequent rules. Thus, for each strong rule, the set is scanned again to identify an exception rule. This step is one of *ERSA's* efficiency and effectiveness bottlenecks. Efficiency bottleneck because the set tends to be large and is wholly scanned every time a strong rule is found. Effectiveness bottleneck because many unnecessary and irrelevant rules can be interpreted as an exception, which can hinder and confuse the expert's analysis.

In contrast, `TRiER` classifies discovered rules between strong and infrequent. Thus, for each strong rule the set of infrequent rules is analyzed to discover an exception that meets the premises defined in Section 4. Specifically, `TRiER` does not scan the whole rules set but only those classified as infrequent, a much smaller set. In order to base the semantic analysis of discovered rules, we present some examples of rules generated by `TRiER` and *ERSA*. Table III presents examples of rules mined by `TRiER`.

Table III. Rules discovered by `TRiER` for AgroData and El-Niño datasets.

Rule	X	Z	$\neg Y$	Y	t_{lag}	$supp$	$conf$	cf
1	$MinTemp[18, 20]$	$rainfall[0, 50]$	$NDVI[0.2,0.4]$	$NDVI[0.6, 0.8]$	1	5.09%	74.38%	69.17%
2	$MaxTemp[28, 30]$	$rainfall[450, 500]$	$NDVI[0.6, 0.8]$	$NDVI[0.4, 0.6]$	1	9.82%	84.53%	70.96%
3	$MinTemp[20, 22]$	$MaxTemp[32, 34]$ and $rainfall[100, 150]$	$NDVI[0.8, 1.0]$	$NDVI[0.6, 0.8]$	1	14.25%	85.47%	79.63%
4	$VentMer[8, 10]$	$VentMer[2, 4]$	$TempSup[27.5, 29.5]$	$TempSup[23.5, 25.5]$	3	8.13%	81.34%	76.45%
5	$VentMer[10, 12]$	$Umid[15, 30]$	$TempMar[26.5, 28.5]$	$TempMar[22.5, 24.5]$	2	8.25%	82.30%	76.71%

Values are presented in intervals, due to the discretization process. Temperatures are measured in degree Celsius, rainfall in millimeters and time lag in months. Support, confidence and certainty factor relate to exception rules. Recall exception rules definition: X is the rule antecedent, Z is the agent causing the exception, $\neg Y$ is the exceptional consequent and Y is the expected consequent. Rules presented in Table III can be respectively described as follows:

— **Rule 1**

Strong rule: If the minimum temperature varies between 18°C and 20°C, NDVI remains between 0.6 and 0.8 in next month.

Exception Rule: If the minimum temperature remains with this value and rainfall varies between 0 and 50mm, NDVI decreases to 0.2 and 0.4 in next month.

— **Rule 2**

Strong rule: If the maximum temperature varies between 28°C and 30°C, NDVI remains between 0.4 and 0.6 in next month.

Exception Rule: If the maximum temperature remains with this value and rainfall varies between 450 and 500mm, NDVI increases to 0.6 and 0.8 in next month.

— **Rule 3**

Strong rule: If the minimum temperature varies between 20°C and 22°C, NDVI remains between 0.6 and 0.8 in next month.

Exception Rule: If the minimum temperature remains with this value, maximum temperature varies between 32°C and 34° and rainfall varies between 100 and 150mm, NDVI increases to 0.8 and 0.1 in next month.

Rule 1 shows that periods of low rainfall (causative agent) contributed to the decrease in NDVI (exceptional behavior). This rule can be validated by studies conducted by Embrapa[6], in which experts state that periods of low rainfall damage sugarcane crops and are responsible for reducing NDVI [Lucas and Schuler 2007]. In contrast, experts say that periods of intense rainfall are related to the increase in NDVI (Rule 2) and periods with regular rainfall are ideal for sugarcane (Rule 3).

— **Rule 4**

Strong rule: If the southern winds vary between $8m/s$ and $10m/s$, the surface temperature varies between 23.5°C and 25.5°C after three months.

Exception Rule: If the southern winds remain at this value, but decrease after two months and vary between $2m/s$ and $4m/s$, the surface temperature increases and varies between 27.5°C and 29.5°C after three months.

— **Rule 5**

Strong rule: If the southern winds vary between $10m/s$ and $12m/s$, the sea surface temperature varies between 22.5°C and 24.5°C after two months.

Exception Rule: If the southern winds remain at this value and the air humidity varies between 15% and 30%, the sea surface temperature increases and varies between 26.5°C and 28.5°C after two months.

[6]https://www.embrapa.br/

These rules are corroborated by the existing knowledge about the El-Niño phenomenon which states that low intensity winds are associated with the increase in the Earth's surface temperature (Rule 4). Also, when low intensity winds are combined with low air humidity, sea surface temperature increases (Rule 5) [Marcuzzo and Romero 2013].

Rules discovered by *ERSA* are semantically restricted since they only allow inferring about relationships and do not consider the temporal aspect. Examples of rules mined by ERSA for AgroData and El-Niño datasets are:

— **Rule 1**

Strong Rule: If the minimum temperature varies between 18°C and 20°C, NDVI varies between 0.4 and 0.6. Support: 58.53%, confidence: 61.20% and certainty factor: 60.43%.

Exception Rule: If the minimum temperature remains at this value and rainfall varies between 200mm and 250mm, NDVI varies between 0.6 and 0.8. Support: 54.13%, confidence: 65.78% and certainty factor: 59.58%.

— **Rule 2**

Strong Rule: If the surface temperature varies between 23.5°C and 24.5°C, the sea surface temperature varies between 22.5°C and 24.5°C. Support: 55.12%, confidence: 66.22% and certainty factor: 62.17%.

Exception Rule: If the surface temperature remains at this value and the air humidity varies between 30% and 45%, the sea surface temperature varies between 26.5°C and 28.5°C. Support: 63.56%, confidence: Support: 61.28% and certainty factor: Support: 60.45%.

6. CONCLUSION

In this paper we proposed `TRiER`, the precursor in the area of Exception Rule Mining with support for multivariate time series. Our method applies the concept of window as a constraint in sequence mining process. Such window enables to discover rules and exceptions that occur in a time lag.

In this context, `TRiER` not only discovers common behaviors (strong rules) and their contradictions (exception rules), but also identifies how long consequences take to appear and the multiple agents that may have caused the exceptional behavior. Identifying temporality in patterns increases semantics to the results and allows a better understanding of the circumstances in which a pattern occurs. Our method enriches the state of the art by discovering rules with greater semantic relevance, in addition to being faster and more scalable than the main competing method for exception rule mining, *ERSA*.

We also presented a new support measure ($supp_{TRiER}$), which significantly reduces the number of mined patterns, resulting in a manipulable set of rules for domain experts. The proposed measure can also be reused by other methods that manipulate time series. Besides support-confidence, `TRiER` implements support-certainty factor. We apply certainty factor as an alternative to confidence to filter rules corresponding to statistical independence or negative dependence.

Finally, the computational complexity of our method is lower when compared with related methods for Sequence Mining. While `TRiER` is exponential in relation to the window size, competing methods are exponential in relation to the number of items/sequences, a potentially much larger number.

REFERENCES

Agrawal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases. In *Proceedings of International Conference on Management of Data*. ACM Press, New York, NY, USA, pp. 207–216, 1993.

Agrawal, R. and Srikant, R. Mining sequential patterns. In *Proceedings of ICDE.* pp. 3–14, 1995.

Berzal, F., Blanco, I., Sánchez, D., and Vila, M. Measuring the accuracy and interest of association rules: a new framework. *Intelligent Data Analysis*, 2002.

Calvo-Flores, M., Ruiz, M., and Sánchez, D. New approaches for discovering exception and anomalous rules. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* vol. 19, pp. 361–399, 04, 2011.

Casanova, I. J., Campos, M., Juarez, J., Fernandez-Fernandez-Arroyo, A., and Lorente, J. Impact of time series discretization on intensive care burn unit survival classification. *Progress in Artificial Intelligence*, 2017.

Daly, O. and Taniar, D. Exception rules in data mining. *Applied Mathematics and Computation*, 2008.

Dong, G. *Sequence data mining.* Springer-Verlag, Berlin, Germany, 2009.

Gan, W., Lin, J. C., Fournier-Viger, P., Chao, H., and Yu, P. S. A survey of parallel sequential pattern mining. *Transactions on Knowledge Discovery from Data*, 2019.

Hussain, F., Liu, H., Suzuki, E., and Lu, H. Exception rule mining with a relative interestingness measure. In *Proceedings of Knowledge Discovery and Data Mining.* ACM Press, New York, NY, USA, pp. 86–97, 2000.

Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 2001.

Lin, J., Keogh, E., Wei, L., and Lonardi, S. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 2007.

Lucas, A. and Schuler, C. Análise do NDVI/NOAA em cana-de-açúcar na ata Atlântica no litoral norte de Pernambuco, Brasil. *Revista Brasileira de Engenharia Agrícola e Ambiental*, 2007.

Marcuzzo, F. F. N. and Romero, V. Influência do El Niño e La Niña na precipitação máxima diária do estado de Goiás. *Revista Brasileira de Meteorologia*, 2013.

Mitsa, T. *Temporal data mining.* Chapman & Hall/CRC, Minneapolis, U.S.A., 2010.

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of ICDE.* pp. 215–224, 2001.

Ruiz, M. D., Sánchez, D., Delgado, M., and Martin-Bautista, M. J. Discovering fuzzy exception and anomalous rules. *IEEE Transactions on Fuzzy Systems* 24 (4): 930–944, 2016.

Srikant, R. and Agrawal, R. Mining sequential patterns: Generalizations and performance improvements. In *Advances in Database Technology — EDBT '96*, P. Apers, M. Bouzeghoub, and G. Gardarin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–17, 1996.

Suzuki, E. Discovering unexpected exceptions: a stochastic approach. In *Proceedings of Rough Sets, Fuzzy Sets, and Machine Discovery.* Tokyo University Press, Tokyo, pp. 259–262, 1996.

Zaki, M. J. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 2000.

Zaki, M. J. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 2001.