Maximilian Toller, BSc

Parameter-Free Collective Anomaly Detection in Sequential Data

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing Dr.techn. Roman Kern

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, July 2018

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date　　　　　　　　　　Signature

Abstract

Anomaly detection is an essential research topic in data science. Detecting anomalies that occur collectively in a sequence is useful for many applications such as intrusion or fault detection. This thesis presents a parameter-free solution for detecting collective anomalies in sequential data based on stationarity and volatility estimation (STAVE). The STAVE algorithm extracts subsequences of a full sequence with a sliding window and clusters them according to a stationarity and volatility distance function. Collective anomalies are then detected by extracting the longest connected sequence within the smallest cluster. In a practical evaluation, STAVE achieved results comparable to commonly used parametric alternatives, while retaining low computational complexity and requiring no input other than the sequence to be investigated.

Contents

List of Figures

1 Introduction

In present days, an abundance of data is produced every day on the Web. A subgroup of this data is called sequential data, which describes ordered sequences of data. Sequential data is only required to have an ordered data sequence and the data does not need to follow any other particular structure. This data type is produced by a number of applications, ranging from econometrics over network traffic to user-action sequences. These many potential origins for sequential data give it an impact on several different areas of present society. By analyzing it, one can gain insights which are useful in various scenarios. For example, if a financial trader analyzes a stock market index and consequently realizes that it is likely to drop in the next days, he will be able to use this knowledge to his advantage. Another example for the benefits of sequential data analysis comes from medicine. If a heartbeat monitor regularly gives off a false alarm, one might be able to explain this behavior by devising a model for the monitor's output.

The above examples outline a central task of data science: Gaining a better understanding of why data behaves in a particular way. There are several ways to achieve this, but a good approach is devising mathematical models. The reason for this is that mathematical models are reasonable, predictable and reproducible. They can be used in mathematical proofs and can also allow a user to predict the future behavior of a data sequence. There are numerous different types of models, and they can be applied in countless scenarios [5] [8]. They can even be used to describe random or highly erratic data as long as the behavior is at least partially predictable.

When constructing models for sequential data, it is important to assure that they can truly describe what they are meant to describe. Therefore, models need to be evaluated after they were constructed. This is typically achieved by fitting a model's parameters to one dataset, and then testing it on a different dataset [2] [5]. Other evaluation steps can examine a model's scope or investigate assumptions about causality within a model.

Generally, when applying a fitted mathematical model to its designated sequential data, it is likely that the data will largely, but not completely match the model's predictions. This is only natural, since mathematical models are a simplification of data. Constructing a model so that it perfectly matches a dataset will usually lead to overfitting [5]. This would limit the model's scope to only the data it was constructed for, making it otherwise useless. Consequently, small deviations between model and data are tolerable, while great differences would question the model's validity. In such a case, devising a different model becomes necessary.

However, there are also phenomena which cannot be properly describe with conventional models. These occurrences are referred to as *anomalies*. We define anomalies as observations that are so different from the remaining data, that attempting to describe them with the original model causes the model to lose its validity on the remaining data [1]. This interpretation is different from classical anomaly definitions, such as:

- An observation that appears to deviate markedly from other members of the sample in which it occurs. [9]
- An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data [3]
- Patterns in data that do not conform to a well defined notion of normal behavior [7]
- An observation, which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism [11]

An exemplar data sequence with an anomaly is depicted in figure 1.1. There, a periodically oscillating signal with an anomaly between time 150 and 200 can be seen. Such anomalies pose problems for many sequential data applications, since they are difficult to explain and fitting a model on them makes it useless on normal data. Typically, anomalies are caused by an external interference on an otherwise closed system. For example, let us assume that a CO_2-measuring device is placed in a room to monitor air quality. The values it measures will depend on various influences, such

[1]We use this definition, since it conforms best with the approach presented in Chapter 4

as the number of people in the room, whether windows are opened or if any plants are in the room. Such situations could be considered as normal. However, if a person strongly exhales directly into the CO_2-measuring device, it would measure an abnormally high value. This would then be an anomaly and it would be impossible to explain with a model which describes normal CO_2-related activity in this room.

Investigating anomalies in a system can be a very beneficial activity. If one can devise a method for automatically detecting them, they can be separated from the normal data. Such an anomaly detection method would have multiple practical applications. For example, let us imagine a web server that offers a chat service to its clients. If this server is suddenly targeted by a distributed denial-of-service attack, it will receive an abnormally high amount of web traffic, which prevents it from responding to normal clients. In such a case, detecting and separating normal from abnormal web traffic would allow the server to continue to provide its service in spite of the attack.

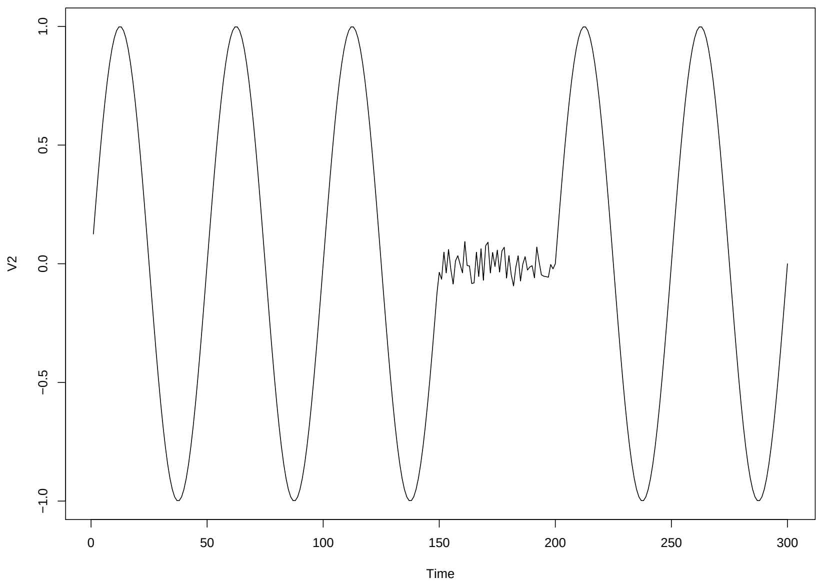


Figure 1.1: A data sequence with a collective anomaly between time 150 and 200

There are different types of anomalies in sequential data. Due to their vague definition, anomalies can occur in countless different forms, as long as they behave clearly different from the remaining data. Further, several distinct types of anomalies can even occur within a single data sequence [7]. This makes general anomaly detection challenging. Therefore, instead of trying to detect all possible types of anomalies at once, it might be easier to detect one particular anomaly type that is prevalent in various applications. One such type is the *collective* anomaly [7], which is defined as a collection of related data points that are anomalous. In the case of temporal data, collective anomalies are connected subsequences of one full data sequence that behave differently than the rest. One exemplar collective anomaly can be seen in above in Figure 1.1. In real temporal data, collective anomalies are usually caused by an external influence which affects the data over a certain time period. All of the examples for anomalies given above describe collective anomalies in temporal data sequences.

The problem of detecting collective anomalies in temporal data has received some attention in anomaly detection literature, although it is still less explored than other anomaly types [7]. The techniques which are used to detect them differ from those which are used for other types. What most of them have in common is that they work with sequential data collections. There, it is assumed that a collection contains many normal sequence, and few anomalous sequences. While this is an interesting setting, it requires a collection of sequences to be implemented in practice.

In this thesis, collective anomalies are investigated in a different scenario. The given data consists of only a single data sequence which may or may not contain one or several collective anomalies. This particular setup is very interesting since it requires only very little data. Additionally this setup is theoretically more challenging than databases yet still seems to frequently be trivial for humans. For example, when looking at Figure 1.2, several anomalies are immediately evident. While the main part of the noisy sequence moves unpredictably inside a restricted range, a few data points are far outside of this range. These points are point anomalies, since they only affect a single observation. The other anomaly with is also clearly visible is the linear behavior between approximately time 300 and 400. This is a collective anomaly, since the anomalous observations are connected over time. All of these anomalies can be easily identified by looking at them. However, for

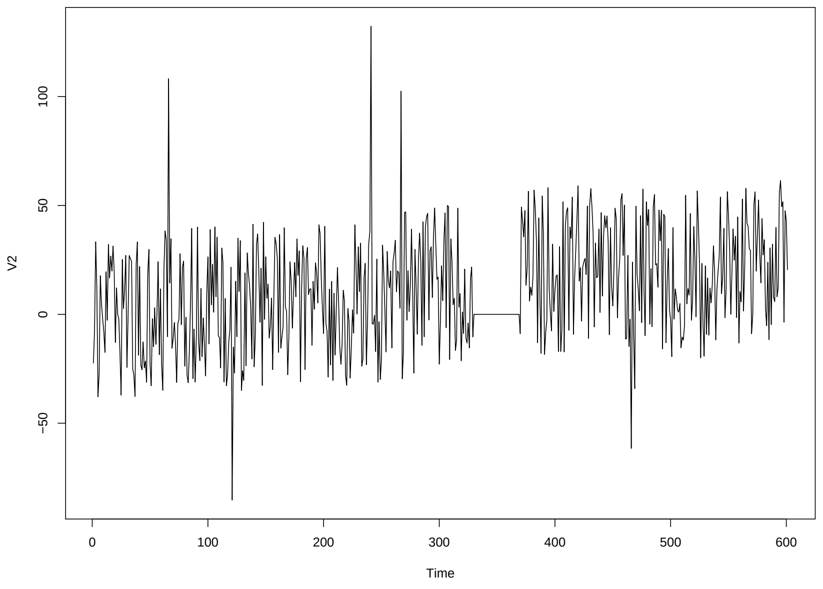


Figure 1.2: A noisy time series with point anomalies and one collective anomaly

a computer, this is not an easy problem. Mathematically detecting collective anomalies is challenging since one cannot rely on the fact that anomalies ″look″ different from the remaining data. When only presented with a numeric data sequence without visualization, detecting collective anomalies becomes much harder. Humans will be slower at spotting anomalies in raw numerical data than at ′seeing′ them in a data visualization.

With only a single given data sequence, the difficulty of detecting collective anomalies largely depends on what one knows about the data in advance. If, for example, it is known that the data originates from a climate database, one can expect the data to have seasonal influences. Such influences can then be used to facilitate the collective anomaly detection. With seasonal influences, one can deduce a contextual model describing where the data should be at a given point in time. If there are collective deviations from the model, then these are likely collective anomalies. However, in this thesis we will assume than there is no prior knowledge about the data. Of course, this rules out the use of approaches such as the seasonal method that was just described. The motivation for complicating the problem in such a way is clear: If nothing is known about the data, and one is still able to reliably solve the detection problem, then this solution will be beneficial for countless different practical applications. Instead, solving collective anomaly detection with prior contextual assumptions about the behavior of the data would restrict the solution to data that truly behaves as assumed.

Additionally, since nothing is known about the data in advance, one can also not expect that such information is given to the solution before it can solve the problem. This means that an algorithm, which should solve collective anomaly detection without prior knowledge about the data at hand, may not depend on parameters which are specified by a human. Otherwise, the task of acquiring appropriate knowledge about the data is simply handed to the user and this excludes a fully automated solution to our problem. Consequently, the solution may only have one input: the data in which it should detect collective anomalies. Further, it will be assumed that this data always contains a single, continuous anomaly.

The solution we devised is based on modeling sequence properties. These properties are computed from the entire sequence, and from all subsequences. The distance between the properties of the full sequence and the subsequences is then used to split the subsequences in two clusters. From the smaller cluster, the longest connected subsequence is extracted, which corresponds to a collective anomaly according to the previously modeled sequence properties.

So, to summarize, the problem which is addressed in this thesis is parameter-free collective anomaly detection a single data sequence without any knowledge about the data's origin. A reliable solution for this problem will be beneficial for many practical applications such as the examples given above. In the following chapters we will discuss which related solutions have already been proposed by literature, how anomaly detection can be achieved in a parameter-free way, how the new method proposed in this thesis compares against related methods, and what this implies for future work.

2 Background

In the data science literature, there are several publications which address anomaly detection in sequential data. In this chapter, we draw borders between different types of anomalies which are investigated in literature. Then, we discuss the solutions proposed by these publications and the techniques upon which they are based. Generally, the related publications are split into two categories: Probabilistic solutions and discord-based solutions. The former use the frequency of observations to detect anomalous behavior, while the latter use similarity measures.

2.1 Terminology

The term *anomaly* is not always used in computer science literature to describe anomalous data behavior. Another frequently-used term is *outlier*. According to Aggarwal [1], these two terms are synonymous and can be used interchangeably. However, most publications which are cited in this thesis and mention outlier detection refer to *point* anomalies/outliers. This is one particular anomaly type, which is discussed below. Therefore, in this thesis abnormal behavior of sequential data will be referred to as anomaly.

2.2 Anomaly Types

As indicated above, there are countless different types of anomalies which can occur in sequential data. In this section, the types which are most prominently investigated in literature are presented. In a survey by Chandola [7] which contrasts different approaches to anomaly detection, anomalies are divided into three distinct categories:

Point Anomalies are individual abnormal points in a data sequence. They are the simplest and most common anomaly type, according to Chandola. Here, *simple* refers to how difficult it is to detect, remove or ignore them. Commonly, point anomalies take abnormally high or low values, making them clearly visually distinguishable from the remaining data. A practical example would be a temperature sensor, which usually measures daily temperatures between -50°C and 60°C, yet during one single measurement it malfunctions and measures 840°C, which is far too high. Point anomalies can be problematic for non-robust algorithms, such as simple linear regression or moving-averages.

Contextual Anomalies are also individual abnormal points in a data sequence. Unlike point anomalies, contextual anomalies do usually not take abnormally high or low values. Instead, they are only abnormal within a given context. For example, if the above mentioned temperature sensor would measure -10°C, then this alone is not an anomalous event. If, however, it is known that this measurement was made in summer, then this additional context makes the measured value abnormal. Detecting contextual anomalies is more challenging than identifying point anomalies, since the context of the measurement has to be known. This either requires an expert who individually identifies contextual anomalies, or a sophisticated model which puts a data sequence into context. Consequently, the success of contextual anomaly detection depends on a specification of a context or at least contextual attributes.

Collective Anomalies are related anomalous points which occur in groups. For sequential data, this means that they occur consecutively after each other. The individual observations can be abnormal by themselves, yet this is not required. They can be collective groups of point anomalies, groups of contextual anomalies, both or neither. A collective anomaly is abnormal because the mutual occurrence of its observations cannot be explained with a model describing the remaining observations. An example would be a skipped heartbeat measured by an electrocardiograph. The value of each observation made during the skipped heart is normal by itself and therefore no point anomaly. Further, they are also no contextual anomalies, since in the nearby observations no heartbeat was occurring. Yet when one sees the individual observations as a connected unit, it becomes evident that a interval between the last and the next heartbeat was too long. Exploring and potentially detecting collective anomalies has been subject of a few publications. These will be discussed in the following sections.

2.3 Probabilistic Methods

In a nutshell, probabilistic methods for collective anomaly detection are based on the assumption that one is less likely to observe anomalies than normal data. Consequently, one should more likely see regular observations than anomalous data instances. Based on this fact, information theory measures such as entropy can be used to distinguish between normal and anomalous.

Let

$$t = t_1, t_2, ..., t_n$$

be a sequence of real-valued observations and

$$c = c_1, c_2, ..., c_m = t_k, t_{k+1}, ..., t_{k+m-1}$$

a subsequence of t. Then the entropy $H(t)$ is the unpredictability of sequence t. Formally, this can be defined as

$$H(t) = -\sum_{i=1}^{n} P(t_i) \log_2 P(t_i), \tag{2.1}$$

where $P(t_i)$ is the empirical probability of observing t_i. This probability is commonly estimated with

$$P(t_i) = \frac{\mathcal{F}(t^{-1}(t_i))}{n},$$

where $\mathcal{F}(t^{-1}(t_i))$ is the number of observations in t which are equal to t_i.

If the entropy of a sequence is low, this suggests that sequence t_i behaves predictably. Conversely, if $H(t)$ is high, then t is hard to predict precisely, which implies that it is difficult to describe by a model. Therefore, anomalous behavior increases the entropy of a sequence, and this can be used for collective anomaly detection. If removing c from t reduces the entropy, then c is less normal than the remaining observations. Hence, entropy based-anomaly detection assumes that c is anomalous if

$$H(t \setminus c) < H(t). \tag{2.2}$$

2.3.1 Entropy-Based Anomaly Detection

From Equation 2.2, the first brute-force solution for collective anomaly detection can be derived. m is the maximum length of the collective anomaly.

Algorithm 1 Brute-Force Entropy

```
Require: $t, m$
  $\text{min}_H \leftarrow H(t)$
  $c_{min} \leftarrow \{\}$
  for $i \leftarrow 1...n-1$ do
    for $j \leftarrow i+1...min(n, i+m-1)$ do
      $c \leftarrow \{t_i, ..., t_j\}$
      if $H(t \setminus c) < \text{min}_H$ then
        $\text{min}_H \leftarrow H(t \setminus c)$
        $c_{min} \leftarrow c$
      end if
    end for
  end for
  return $c_{min}$
```

The idea behind Algorithm 1 is simple: Remove every possible connected subsequence of t once, and check how the entropy changed. The subsequence which decreased the entropy the most is then the collective anomaly.

While this approach is simple, it also has considerable deficits. m must exactly match the length of the collective anomaly, since otherwise the entropy of the collective anomaly could be further increased by appending it to normal observations. For example, given a sequence

$$t' = 0, 0, 0, 0, 0, 0, 39, 33, 29, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0$$

it is clear that the collective anomaly in t' is $t'_7, t'_8, t'_9 = 39, 33, 29$. With the correct anomaly length $m = 3$, Algorithm 1 would indeed return $c' = t'_7, t'_8, t'_9$. However, if m is not known in advance and guesses that $m = 5$, then Algorithm 1 would return $c' = t'_5, t'_6, t'_7, t'_8, t'_9$, since the entropy of this sequence is even higher.

Another more subtle problem of Algorithm 1 is that collective anomalies do not have to take unexpected values, they are only required to be collectively anomalous. An example for such a case is depicted in Figure 2.1. Collective

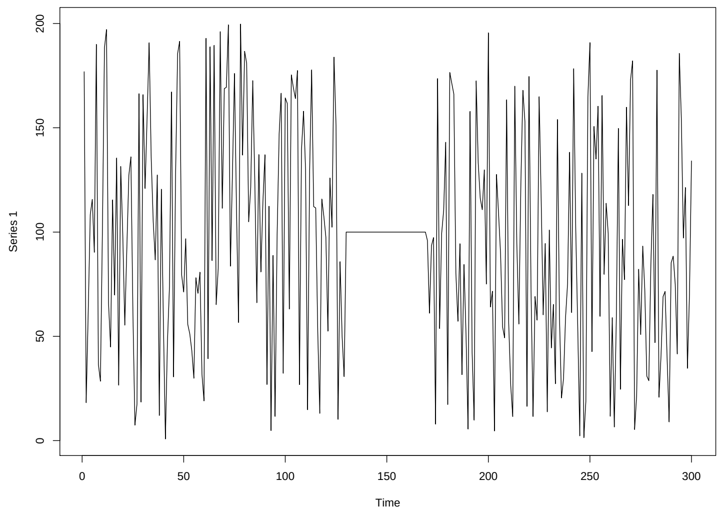


Figure 2.1: A noisy data sequence with a collective anomaly from observation 140 to 169

anomalies of this type are difficult to detect with entropy based methods, since they may seem to be more regular than the remaining data.

2.3.2 Markov Chain-based Anomaly Detection

In literature, more sophisticated methods than Algorithm 1 can be found. One such approach is based on first order Markov chains , which are stochastic processes with a memory of one observation. The term *stochastic* describes a probabilistic process that is influenced by past events and present probabilities. Since a model which only considers dependencies between two consecutive observations cannot be sufficient for collective anomaly detection, n-Markov chains can be used instead [1].

Many real-world data sequence from various domains exhibit a behavior called *short memory*, which can be mathematically expressed as

$$P(t_n|t_{n-k}...t_{n-1}) \approx P(t_n|t_{n-L}...t_{n-1}) \forall k > L. \tag{2.3}$$

In essence, this means that the conditional probability of an observation may depend on a varying number of previous observations. So, if one assumes the data at hand exhibits *short memory*, then using one constant n-Markov chain is insufficient. Therefore, Ron et al.[22] proposed a variable-order Markov chain. They also presented an appropriate representations of this model: The Probabilistic Suffix Tree (PST). This is a tree, where each edge represents a (discrete) symbol from the symbol alphabet G, and every internal node has one edge for each possible symbol that could follow it. The nodes also store the next symbol probability γ, with $\sum_{i=1}^{|G|} \gamma_i = 1$. The probability that a probabilistic suffix tree creates a sequence t^0 can be expressed as

$$P_S(t^0) = \prod_{i=1}^{|G|} \gamma_{i-1}(t_i^0) \tag{2.4}$$

which simply is the product of the edge probabilities that progress down S and produce t^0.

The PST was used in several following publications for the classification of sequences. If one combines the PST with a similarity measure, it can be used

[1] An n-Markov Chain has a memory of n observations.

for detecting anomalous sequences. This was first done by Sun et al. [24], who used the similarity measure SIM_N which is defined as

$$\text{SIM}_N(t^0, S) = \frac{1}{m}(\log P_S(t_1^0) + \sum_{i=2}^{m} P_S(t_i^0|t_1^0...t_{i-1}^0)) \tag{2.5}$$

where t^0 is the query sequence. An advantage of this measure is that it also considers the length of a sequence, unlike many other similarity measures. In a following experimental study, Sun et al. showed that their method is fast and well-suited for deciding if a query sequence is abnormal with respect to a sequence database. Additionally, they addressed the problem of detecting the top-n anomalous sequences in a database.

2.3.3 Drawbacks of PST and Probabilistic Methods

To transfer the method of Sun et al. to the setting of this thesis, where only a single sequence without a database is known and a anomalous subsequence is searched, several adjustments are required. Firstly, a database must be generated. This can be achieved by generating $\mathcal{P}(t)$, the set of all subsequences of t. Secondly, only the top-n anomalous sequence problem they addressed is viable in our setting, since no query sequence is available. Finally, the real-valued data has to be discretized, or at least rounded to a few significant digits, to keep the number of branches in the PST reasonably small [2].

These required adjustments cause additional problems. On the one hand, discretization is can be problematic, since much information may be lost during the process and the associated discrete intervals have set to appropriate values. On the other hand, $\mathcal{P}(t)$ is not equivalent to a sequence database. This is, in fact, a general problem of relating sequence database research to research on a single sequence. In a database, all sequences have an explicitly defined beginning and end, whereas subsequences extracted from a source sequence do not. Of course, a subsequence can be seen as an individual

[2]To represent a PST with real-valued sequences, every node would require one edge for every possible real-valued observation that could follow it, which is infeasible in a practical context.

sequence with a start and an end, yet these two important points are chosen by a user or an algorithm and not already determined in advance. This is a great difference from $\mathcal{P}(t)$ to a database. The subsequences in $\mathcal{P}(t)$ should not be seen as individual instances, since all the information they contain comes from the original sequence. They may overlap, or can be subsequences of each other.

Attempts to circumvent this problem by generating a database in a different way than using $\mathcal{P}(t)$, such as splitting t into equal-length subsequences, will not resolve the difficulties. With only a single given data sequence, there simply is no more information available than this one original sequence. Consequently, to successfully detect collective anomalies in a single sequence t, it must by itself contain enough information so that one can distinguish between normal and anomalous behavior. This means that the length m of the collective anomaly c has to be short compared to the sequence length n, which means

$$m \ll n. \tag{2.6}$$

2.4 Discords

Since probabilistic anomaly detection methods suffer from the deficits mentioned in Section 2.3, researchers have come up with a more practical approach. This approach is based on discords. Let $\mathcal{S}(c, d)$ be a similarity measure and $\hat{d}$ a subsequence of t with length m. If a sequence t is split into $\frac{n}{m}$ subsequences $\mathcal{T} = \mathcal{T}_1, ..., \mathcal{T}_{\frac{n}{m}}$ with length m, then $\hat{d}$ is a discord if

$$\hat{d} = \arg\min_{d}(\mathcal{S}(\mathcal{T}_d, \arg\max_{e}(\mathcal{S}(\mathcal{T}_d, \mathcal{T}_e))) \tag{2.7}$$

This means that a discord $\hat{d}$ is the sequence in $\mathcal{T}$ whose nearest neighbor is the least similar. Using discords for collective anomaly detection is simple. The discord is assumed to be equivalent to the collective anomaly. A brute force solution suggest by Keogh et al. [14] for finding discords, and thus also for finding collective anomalies, can be seen in Algorithm 2. This solution

Algorithm 2 Brute-Force Discords

```
Require: $T, m$
  $\text{min}_{sim} \leftarrow \infty$
  $\text{location}_{\text{min}} \leftarrow \text{NaN}$
  for $d \leftarrow 1...n - m + 1$ do
    $\text{max}_{sim} \leftarrow 0$
    for $e \leftarrow 1...n - m + 1$ do
      if $S(T_d, T_e) > \text{max}_{sim}$ then
        $\text{max}_{sim} \leftarrow S(T_d, T_e)$
      end if
    end for
    if $\text{max}_{sim} < \text{min}_{sim}$ then
      $\text{min}_{sim} \leftarrow \text{max}_{sim}$
      $\text{location}_{\text{min}} \leftarrow d$
    end if
  end for
  return $[\text{min}_{sim}, \text{location}_{\text{min}}]$
```

will reliably identify the discord of a sequence t as long as m is given. However, it is also a slow algorithm with a worst-case complexity of $\mathcal{O}(n^2)$, which is why Keogh et al. also presented a method for speeding it up. This acceleration is based on a symbolization of the sequence.

2.4.1 Symbolic Aggregate Approximation

A commonly used discretization technique for sequential data is Symbolic Aggregate Approximation (SAX), which was suggested by Lin et al. [16]. It consists of two steps: Piecewise aggregate approximation and symbolization. The idea of the first step is to reduce the dimensionality of t to a w-dimensional space with $w < n$. This can be achieved by replacing neighboring observations, called "frames", by their means. Mathematically, this can be expressed as

$$\bar{t}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} t_j, \tag{2.8}$$

where $\bar{t} = \bar{t}_1, ..., \bar{t}_w$ is the piecewise aggreate approximation of t. In essence, this means that, unlike splitting t into $\frac{n}{m}$ subsequences T of length m as in the probabilistic approaches, t is summarized in w observations $\bar{t}$ of length 1.

After the dimension reduction, $\bar{t}$ is symbolized . Based on the assumption that a normalized [3] sequence t^N follows a known distribution [15], breakpoints $\beta = \beta_1, ..., \beta_{l-1}$ for the l symbols can be chosen is such a way that every symbol has an equal probability. This can be achieved with a statistical table for the Gaussian distribution. The breakpoints β are then used to discretize the real-values of t^N to a symbolized sequence $\mathcal{S}_t$.

2.4.2 Heuristically Ordered Time Series using SAX

We introduced SAX in the above section to speed up Algorithm 2. In essence, what makes this algorithm slow are the two *for* loops, both of which have a run-time of $\mathcal{O}(n)$. At this point it is important to observe that in fact no full iteration over d and e is necessary. As soon as a value was found that is larger than the current $\max_{sim}$, the loop can be aborted. So, if the equal length subsequences $\mathcal{T}$ were ordered in an ideal way, then Algorithm 2 would be significantly faster with $\mathcal{O}(n)$ run-time.

Keogh et al. [14] solved this with "Heuristically Ordered Time series [4] using Symbolic Aggregate Approximation" (HOT SAX). They extract subsequences of t with a sliding window, and then symbolize these subsequences with SAX. Next, these symbols are organized in a occurrence map and a prefix tree, which are used to identify which symbols occurs how frequently and where it appears. Both of these structures can be computed in $\mathcal{O}(n)$ space and time. The discord $\hat{d}$ can then be identified in $\mathcal{O}(n)$ time by only iterating over all symbol sequences which occur once. If the algorithm has not found the discord in the set of all symbol sequences which occur exactly once, then the remaining sequences are iterated in random order.

[3] A sequence t can be normalized to t^N with $t_i^N = \frac{t_i - \mu_t}{\sigma_t}$, where μ_t is the mean and σ_t the standard deviation of t

[4] A time series is a real-valued time-discrete temporal sequence

2.4.3 Drawbacks of HOT SAX and Discords

The effectiveness of HOT SAX was demonstrated in several practical experiments with data from various different domains. To provide concrete values for the parameters introduced above, which were the alphabet size l and the word length w, Keogh et al. provide two practical observations. While they admit that w strongly depends on the dataset, they also claim that, for l, a value of either 3 or 4 is best "for virtually any task on any dataset" [14].

This is not true and easily refuted. Let $\mathcal{D}$ be a multimodal [5] distribution with 5 or more distinct modes. Any data sequence from this distribution with sufficiently many observations cannot be properly represented by three or four symbols. An exemplar data sequence where SAX with $l = 3$ or 4 does not work is depicted in Figure 2.2. Consequently, using SAX adds two parameters which require tuning to the collective anomaly detection problem. Another critical parameter required by discord-based methods is the length of the anomaly, as otherwise one could not divide a sequence into subsequences and search for the least similar.

Further, detecting discords fully depends on the selected similarity or distance measure. Many solutions in literature use Euclidean distance, which is defined as

$$\delta(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{2.9}$$

While this measure is suitable for many different problems, it only depends on the geometric distance between x and y. This can cause undesirable side effects, such as an over-interpretation of y-shifts. An example for this can be seen in Figure 2.3. Consequently, other anomaly detection methods that fully rely on Euclidean distances for measuring similarity, such as Matrix Profile-based discord detection [27], are not ideal for highly variant [6] or non-stationary [7]data. In spite of their disadvantages, discords have a distinct advantage over probabilistic methods. Detecting anomalies requires

[5]A multimodel distribution is the sum of several Gaussian distributions with distinct μ and σ.

[6]Variance is the second statistical moment and defined as $var(x) = \frac{1}{n}\sum_{i=1}^{n}(\bar{x} - x)^2$

[7]Stationary processes are defined in Chapter 4.

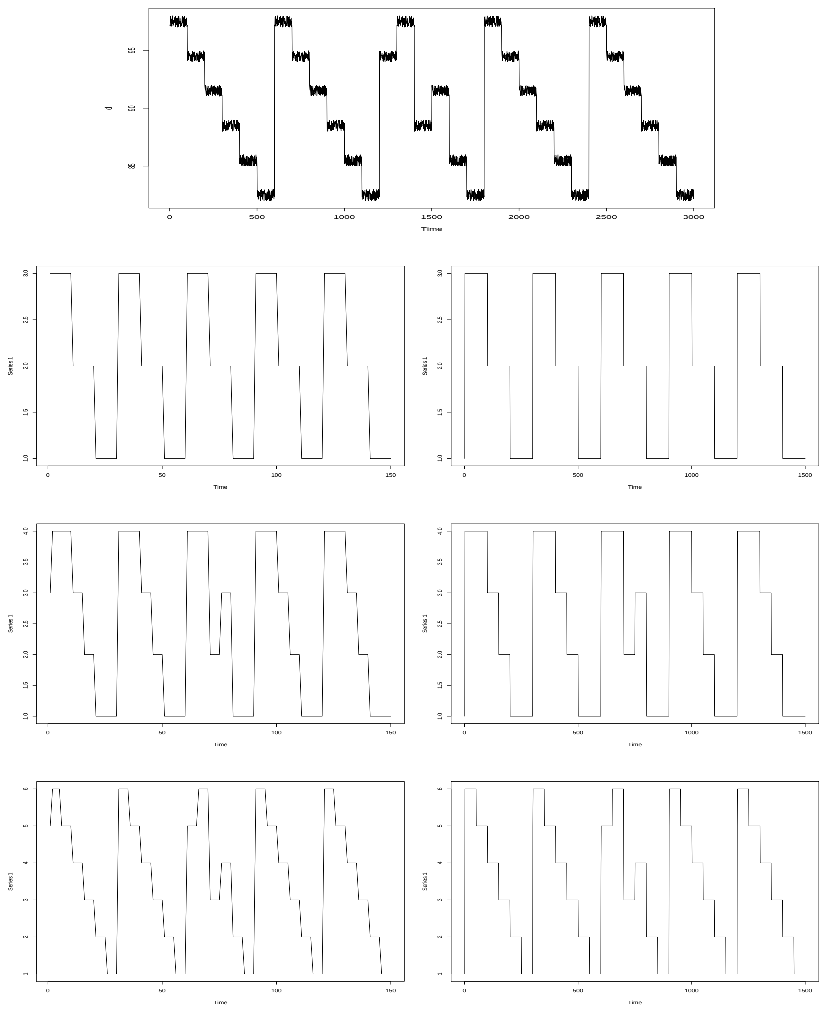


Figure 2.2: An exemplar data sequence where SAX with $l = 3$ or 4 is a bad choice. The top plot shows the original multimodal data sequence with a collective anomaly in the middle. The second row depicts SAX with $l = 3$ and $w = 150, 1500$, where no anomaly can be seen. The third row shows SAX with $l = 4$ and $w = 150, 1500$, where only half of the anomaly is visible. Finally, the bottom row shows SAX with a better parametrization $l = 6$ and $w = 150, 1500$, where the entire anomaly is visible.

a method for distinguishing between normal and abnormal observations, and very frequently this is accomplished with a threshold. Such a threshold

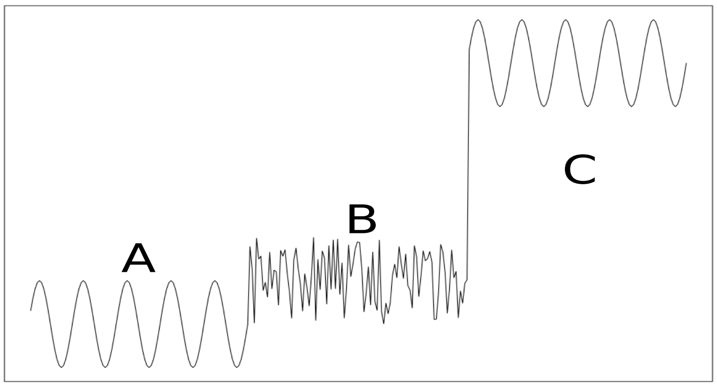


Figure 2.3: A sequence where Euclidean distance can be misleading. Subsequences A and B have a higher Euclidean similarity than A and C, since they are geometrically closer. Intuitively, A and C should be more similar to each other than either to B

is a hard decision boundary and needs to be tuned to the data. Discord-based detection does not need such a threshold, since the entire detection approach is based on the assumption that the least similar subsequence is anomalous. In essence, this implies the additional assumption that an anomaly is present in the data. This assumption can obviously be disadvantageous in some practical settings. However, in this thesis we will also assume that the observed data always contains an anomaly.

2.5 Artificial Neural Networks

In many areas of computer science, the current state of the art research includes the application of machine learning techniques. A very commonly used supervised technique is the construction and training of an Artificial Neural Network (ANN). ANNs can be seen as networks with an input layer, an output layer, and arbitrarily many "hidden"-layers in between. A simple variant of an ANN is the perceptron. It is a single artificial neuron, which receives one or multiple inputs, and then activates if this input triggers its activation function. An example is depicted in Figure 2.4

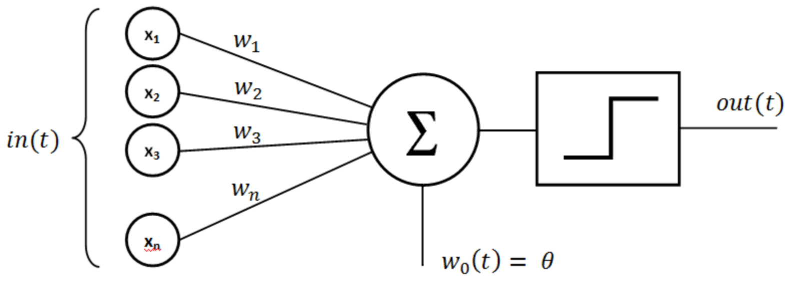


Figure 2.4: A model of a perceptron. x is the input, w the weights. These are multiplied, summed up and then fed into the activation function.

Formally, a perceptron can be described as

$$o = f(xw + b) \tag{2.10}$$

where x is the input vector, w the vector of weights, b the bias and f the activation function. To allow a perceptron to learn by training it, one presents it several inputs and then changes the weights w according to the correctness of the output. With a learning rate a and the expected output ϕ, the weights of a perceptron can be updated with

$$w_i = w_i + a(\phi - o)x_i. \tag{2.11}$$

In essence, a perceptron learns to draw a line between input that should make it activate and input that should not. It is a linear separator (or binary classifier). Of course, such a simple model has limitations. Perceptrons can only solve classification problems which are linearly separable. A famous example were this does not work is XOR-problem. Hence, ANNs need more layers to solve non-linearly separable problems.

Layers in between the in- and output layers are called hidden layers. With multi-layered ANNs, detection/classification works the same way as with perceptrons. However, to implement learning in such a setting, one needs to refine the perceptron learning algorithm. The hidden-layer before the output layer is trained in the same way as a perceptron. Then, the weight changes are propagated back through the network until all weights were

Algorithm 3 Back Propagation

```
Require: $j$
  for all $k$ do
    $w_{k,j} = w_{k,j} + af'(\sum_{i=1}^{\lambda} x_i)(\phi - o)x_k$
  end for
```

updated. If $w_{k,j}$ is the weight from the kth node in the previous layer to the kth node in the current layer, then the algorithm for this can be written as

In essence, this is Equation 2.11 propagated back through the network. This algorithm is repeated for all neurons in all hidden layers, and iterated for all training samples until the weights converge [8].

2.5.1 Recurrent Neural Networks

In the ANN architecture introduced above, all information from the input propagates forward through the network until the output layer is reached. Such a feed-forward architecture is not the only way how an ANN can be constructed. A recurrent neural network (RNN) is a network where backward connections between hidden layers are allowed. An example can be seen in Figure 2.5

A distinct advantage of RNNs is their ability to memorize previous inputs. This allows them to understand ordering, which is essential when working with sequences. Malhotra et al. [17] used RNNs with added long short-term memory cells to implement collective anomaly detection in sequential data. They trained the network by first showing it normal sequences, and then using its learned knowledge to predict how anomalous sequences should develop if they would not contain an anomaly. The deviations between predicted and actual values where then returned as anomalous.

Of course, such an approach requires tuning of several parameters, such as initial network weights or the deviation threshold. Yet one distinct advantage of this approach is that the length of the anomaly does not have to be

[8]Convergence is reached when the changes between iterations are smaller than a predefined threshold.

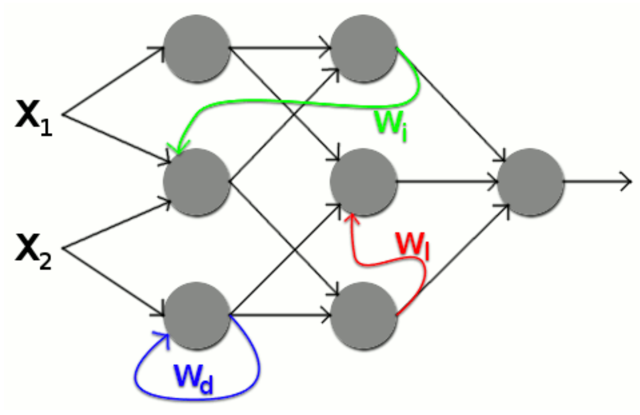


Figure 2.5: A recurrent neural network. Edges may go forward (black), backward (green), within layers (red) or stay at the same node (blue).

specified in advance. Further, no window size for the predictor has to be specified.

A disadvantage of the method proposed by Malhotra et al. is that its performance was not extensively evaluated against other methods. Instead, this work can be seen as proof-of-concept for ANN-based anomaly detection in data sequences.

In general, deep learning [9] is likely a promising approach to collective anomaly detection. There are many other tasks in data science were the state of the art was improved by ANNs, in particular with additional long short-term memory cells [26]. However, in this thesis, we will not use deep learning. Currently, there is an extensive hype for deep learning in data science. Yet it has also been criticized for weaknesses, such as being data-hungry, failing to distinguish causation from correlation [10], and not being able for transfer learned facts into other settings [18]. Instead, we will explore an approach based on unsupervised learning in the following chapter.

[9]Machine learning using neural networks with multiple layers

[10]This weakness is not specific to deep learning. Many other approaches to machine learning face this problem as well.

3 Preliminary Work

Prior to this thesis, several different methods were devised, investigated or attempted. In this chapter, less successful approaches to collective anomaly detection are discussed. They are presented in chronological order according to the date on which they were devised. We also note that this chapter uses terms that are only defined in following chapters.

3.1 Season Shapelets

In a preliminary master project, an approach for detecting multiple collective anomalies in seasonal time series was developed. This setting is different from that of this thesis, where seasonality is not required and all sequences contain exactly one collective anomaly. However, parts of the approach were adapted and reused in this thesis. The main idea was to construct a shapelet, which is a sequence that represents a ″typical″ season. Computing the distance between each season and the shapelet could then indicate which seasons are normal and which abnormal. These distances were further clustered with two means to clearly divide the subsequences according to their normality. An exemplar seasonal sequence and an appropriate shapelet can be seen in Figure 3.1.

An advantage of shapelets is that they are very intuitive, yet applying them in practice can be challenging. Firstly, constructing an appropriate shapelet is not trivial and can be particularly difficult if the data used for deducing it contains an anomaly, since the anomaly will likely influence the shapelet's construction. Secondly, it is problematic to place the shapelet at the correct position, if the data is not stationary. Finally, shapelets require

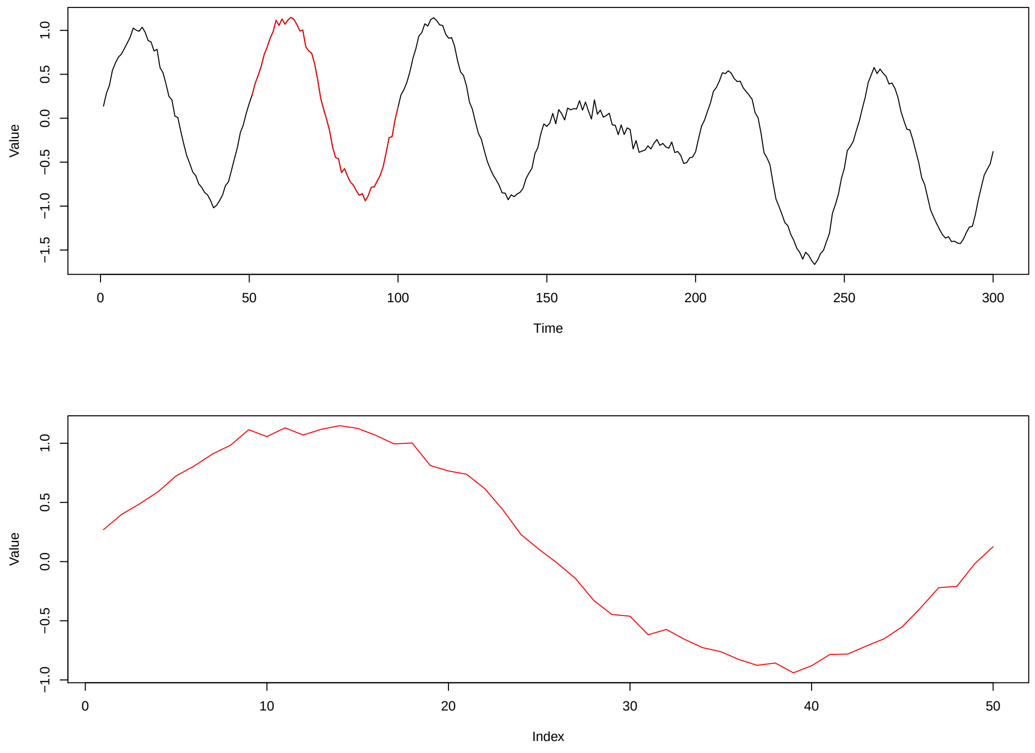


Figure 3.1: A seasonal sequence (top) and the corresponding shapelet (bottom)

recurrent motifs in the input data, and this makes it difficult to use them on non-periodic data.

This method inspired the clustering of subsequence-properties with two means in the final method of this thesis.

3.2 Regions of Interest

Since shapelets are susceptible to non-stationarity, it might be advantageous to devise an algorithm that works with non-stationary sequences. The second major idea that was investigated was based on regions of interest. This primitive originally comes from computer vision, where it is a rectangular area in an image. To transfer regions of interest to sequential data, a restricted sliding window was used. This window was not only limited to a specified number of observations (x-limit), but also to a specific observation value range (y-limit). It could move in x- and y-direction, and values outside of the specified region were considered as missing values. As the window progressed, every region of interest would be stored, and later used for mining anomalous windows with a different algorithm. An example can be seen in Figure 3.2.

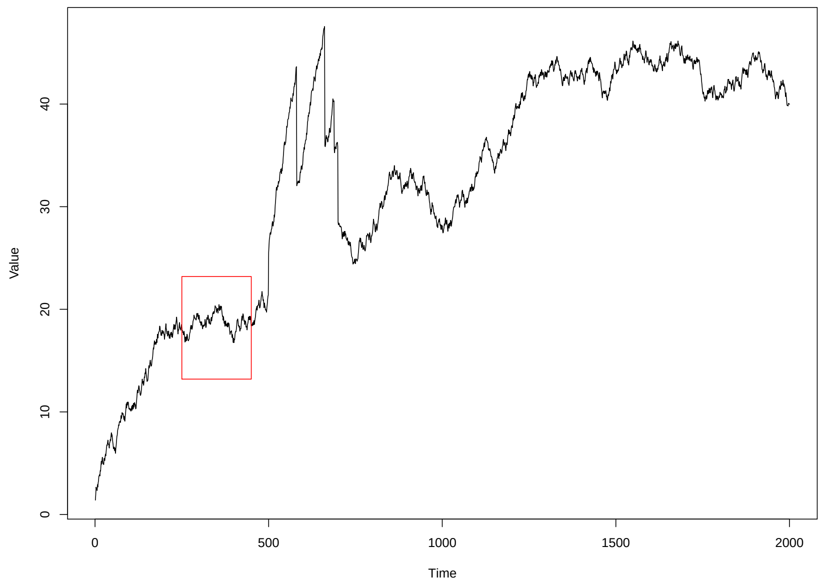


Figure 3.2: A sequence (black) and the current region of interest (red)

One main problem with this approach was that many steps were undefined. While a 1-dimensional window has a clear direction in which it moves over the sequence, a 2-dimensional window requires additional decisions on when to move in the value space. It is not trivial to move the window along the data without following a moving average, which would make the 2-dimensional window meaningless. Additionally, simply moving the window to every possible location would not only produce many regions with all values missing, which could be easily discarded. It would also produce many regions with only very few non-missing observations, which are difficult to handle. A further disadvantage is the large number of missing values this approach generates. However, the main reason why this idea was abandoned was that it overcomplicates the entire task. While sequences are often depicted in a 2-dimensional plots, it is not necessary to investigate them with 2 degrees of freedom, since they can only move forward in time. Further, this approach has several parameters, such as height and width of the window, which made it unsuited for the task of this thesis.

3.3 Backward similarity

The third main alternative idea for detecting collective anomalies was based on backwards similarity. The intuition behind the idea was to compare every observation with a number of previous observations and compute the Euclidean distance. This was repeated for a number of backward-lags, which were then summarized by taking the mean of all lags. The resulting distance function was then searched for collective statistical outliers o, which were determined with $o \leq \delta \vee o \geq \delta$, where δ is a predefined threshold.

Unlike the other methods in this chapter, backwards similarity seemed promising in preliminary tests. It was considered as main solution candidate until the approach presented in Chapter 4 was devised. An advantage of backwards similarity was that it was more simple than shapelets or regions of interest, and that it transfers periodicity anomalies to statistical outliers. However, choosing the threshold δ was critical to the success of the method. The initial idea was to use the standard three-sigma rule [20], but preliminary testing showed that this is not the best choice for all data. Therefore, instead of a using a threshold, backwards similarity was also tested with clustering similar to the shapelet-based approach (and similar to the final method of this thesis).

Since backwards similarity showed no clear theoretic flaws, it was also included in the evaluation in Chapter 5.

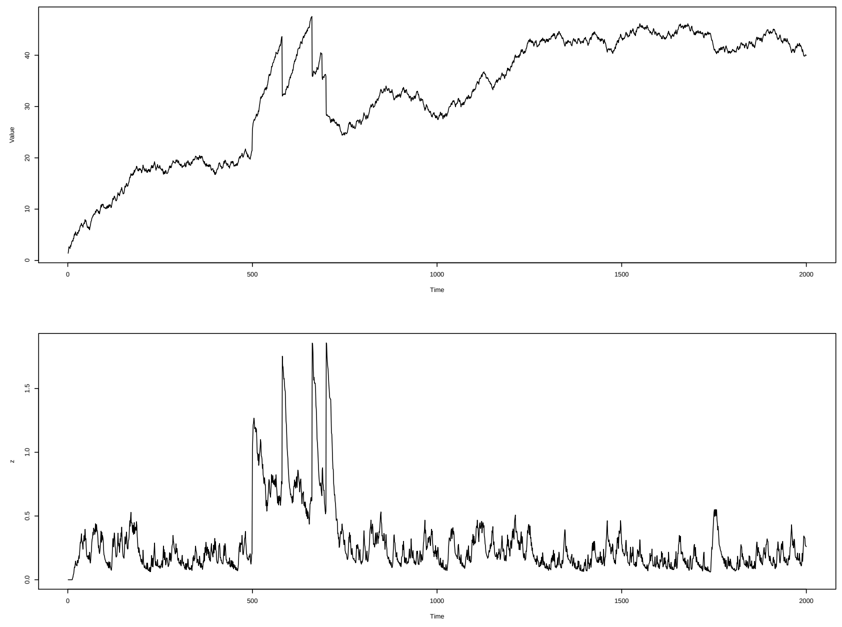


Figure 3.3: A sequence with an anomaly (top) and the backwards distance function (bottom).

4 Methods

From the previous chapters, we have learned that parameter-free collective anomaly detection is an interesting and challenging problem. In this chapter, we will make several key assumptions and observations, which we will use to create a new method for detecting collective anomalies. Additionally, all related concepts are explained and assigned to a reasonable position, which allows the reader to reconstruct the methods used in this thesis.

4.1 Stationarity and Correlation

Let $t = t_1, ..., t_n$ be a sequence of observations. If the statistical properties of t do not change over time, then t is considered to be stationary. A strictly (or strongly) stationary process is a sequence whose joint probability is independent of any lag, which means

$$P(t_1, ..., t_k) = P(t_{1+\tau}, ..., t_{k+\tau}) \tag{4.1}$$

where τ is an arbitrary lag. Strict stationarity implies that the mean and variance do not change over time. Obviously, not all sequences are strictly stationary. Another commonly encountered property is wide-sense (or weakly) stationarity. A wide-sense stationary process has a time-independent mean and an autocorrelation that only depends on lag τ. The autocorrelation can be computed with

$$A_t(\tau) = \sum_{i=1}^{n} t_i t_{i-\tau} \tag{4.2}$$

where τ is the lag at which the sequence is compared with itself. This means that one shifts sequence in time and compares if high and low values occur in nearby positions. Usually, the autocorrelation A is further normalized by its maximum value, which can also be found at $\tau = 0$

$$\mathcal{A}_t(\tau) = \frac{A_t(\tau)}{A_t(0)} \tag{4.3}$$

4.2 Modeling Sequence Behavior

We previously already defined anomalies as behavior that cannot be meaningfully described by a model that describes the remaining data. Since there are countless different structures, patterns or motives than can occur in data sequences, it is difficult to develop a model that can describe all normal data sequences and is still valuable in a practical application. Therefore, we will consider three different theoretic types of sequence behavior. These types will then form the basis for detecting collective anomalies.

4.2.1 Seasonal Sequences

In the context of sequential data, the term seasonality refers to regular patterns which repeat over time. If sequence t contains a pattern that appears regularly every s observations, then t is seasonal with season length s. A typical example for seasonal behavior are monthly temperature measurements. Every summer, the temperatures rise, and every winter they fall. Mathematically, there are several ways to represent seasonality. One representation is based on the trigonometric sine-function and can be written as

$$y_i^s = a \sin(\frac{2\pi}{s} t_i + b) \tag{4.4}$$

where a is the amplitude, season length s the period, and b the phase offset. An example can be seen in Figure 4.1 In econometrics, the data sequences are often decomposed in trend, seasonal, cyclical and irregular components [4] [10]. This can be written as

$$y^s = T + S + C + R. \tag{4.5}$$

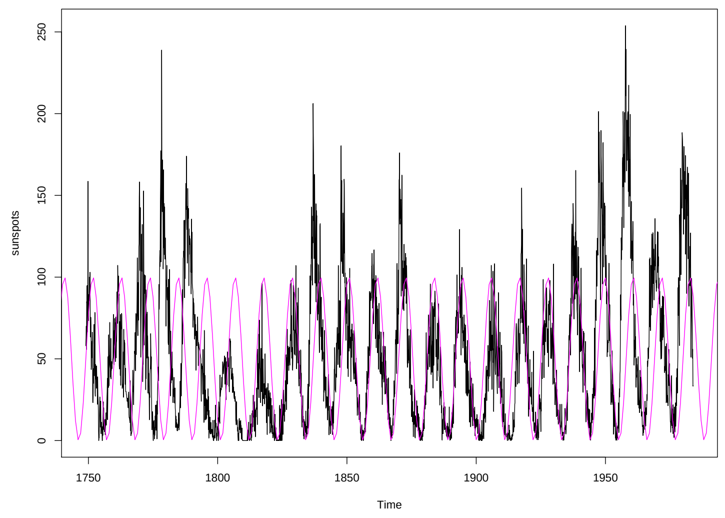


Figure 4.1: A sequence of monthly sunspots (black) was approximated by a sinusoidal sequence (magenta).

While this model can be used to describe many data sequences, it is also difficult to tune and requires several parameters, such as a function describing the trend or the season length. Therefore, we will prefer the less accurate yet simpler sinusoidal model for describing seasonality in the context of anomaly detection.

An interesting property of seasonal data is that its correlation with itself is also seasonal. Since a seasonal sequence t is periodic, its autocorrelation $\mathcal{A}_t$ will be periodic as well if t is wide-sense stationary. This can be useful for anomaly detection, as $\mathcal{A}_t$ often is easier to analyze than t itself. However, if one solely relies on seasonal models, it is difficult to succeed without knowing the season length, which is another parameter that requires tuning. Yet even without this knowledge, two observations can be made about seasonal sequences:

- If y^s has no trend or similar component, it is usually stationary.
- y^s has a constant, periodic behavior.

4.2.2 Noisy Sequences

Another phenomenon that can be frequently observed in sequential data is noise. Noisy sequences contain random deviations from an expected 'smooth' behavior. An example can be seen in Figure 4.2. There are various

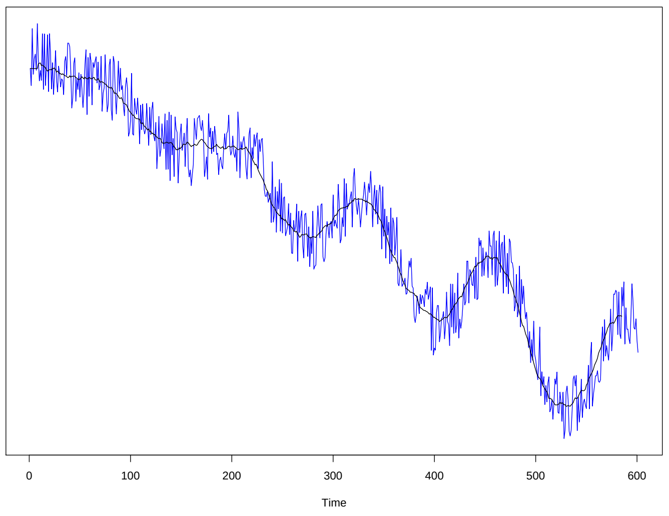


Figure 4.2: A sequence (black) with added ergodic white Gaussian noise (blue)

different types of noise. Here, we will assume that noise is additive white Gaussian noise. Let $\mathcal{N}(\mu, \sigma^2)$ be the Gaussian (normal) distribution and $R \sim \mathcal{N}$ be a sequence of independent and identically distributed random numbers drawn from $\mathcal{N}$. A sequence t that is under the influence of additive white Gaussian noise can then be expressed as

$$y_i^R = t_i + R_i. \tag{4.6}$$

Noisy sequences are less predictable than smooth sequences, which is why many non-robust algorithms perform poorly on them.

Additive white Gaussian noise, and in a wider-sense every independent and identically distributed process, is an ergodic process. This means that its statistical properties can be deduced from a sufficiently long subsequence, which can be written as

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} R_i = \mu_R. \tag{4.7}$$

Ergodic processes are strictly stationary, and are very long-term predictable, although they can seem erratic and fluctuating when one observes only individual values. Markov Chains, as introduced in Chapter 2, are also ergodic. However, a noisy sequence y^R does not have to be ergodic, since the white Gaussian noise R is only added to the original sequence t. If t is not stationary, then y^R will not be stationary either.

Distinguishing noisy from smooth sequences, or even measuring the amount of noise in a sequence, is not trivial. Cellucci et al. [6] have explored these tasks for stationary processes, and Hu et al. [12] addressed non-stationary processes. Yet in general, two key observations can be made about noisy sequences:

- R is stationary, and y^R is stationary if t is stationary.
- The individual observations of y^R fluctuate and are highly erratic.

4.2.3 Random Walk Sequences

While additive white Gaussian noise is a random stationary sequence, not all purely random sequences are stationary. If noise is not added to an existing signal, but rather accumulated over time, one obtains a fluctuating process that is not stationary. Let $\omega \sim \mathcal{N}(0, \sigma^2)$ be a random variable that is accumulated n times. Then

$$y^\omega = y_1^\omega, y_2^\omega, ..., y_n^\omega = \omega_1, \omega_1 + \omega_2, ..., \sum_{i=1}^{n} \omega_i \tag{4.8}$$

is a Gaussian random walk. An exemplar sequence is depicted in Figure 4.3. In practice, sequences similar to random walks can be frequently found in exchange rates on the market [25].

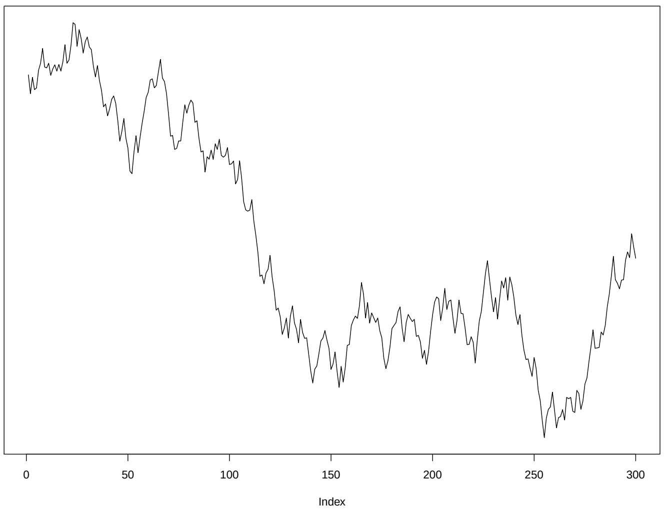


Figure 4.3: An exemplar random walk data sequence

Random walks are nonstationary since they have a unit root. Let X be a first order autoregressive process of the form

$$X_i = aX_{i-1} + \epsilon_i, \tag{4.9}$$

where a is an arbitrary factor and ϵ the sequence of errors. If one introduces the backshift operator $\mathcal{B}X_i = X_{i-1}$, then Equation 4.9 can be rewritten as

$$(1 - a\mathcal{B})X_i = \epsilon_i. \tag{4.10}$$

If we assume that all errors $\epsilon = 0$ and that the sequence X is not zero, we obtain the characteristic equation

$$1 - a\zeta = 0 \tag{4.11}$$

by replacing the backshift operator with the unknown root [1] ζ of the associated polynomial. Consequently, $\zeta = \frac{1}{a}$. If $|a| < 1$, then X is stationary, and if $|a| > 1$, then X is nonstationary and diverges. For a random walk y^{ω},

[1]The roots of a polynomial $p(x, f) = x^p + a_0x^{p-1} + ... + a_f$ are the zeros of p, which means $p(x', f) = 0$

$|a| = 1$, which means that y^ω is not stationary, but will also not explosively grow towards infinity.

Using the roots of a sequence's characteristic equation can be an excellent indicator for assessing stationarity. However, in a practical setting, sequences are not generated by perfectly autoregressive or cumulative processes, which limits their practical applicability. Still, as with the previous sequence models, we can make two observations about random walk-like sequences:

- y^ω is not stationary.
- y^ω is not regular and hard to predict.

4.3 Practical Implementation

In the previous section, we have made several observations about different sequential data models regarding their stationarity and regularity. Since the goal of this thesis is to devise a practical collective anomaly detection method, purely theoretic considerations such as unit roots are not helpful. However, the observations we made about the different sequential data models are not useless. We observed different degrees of stationarity and volatility, which measures the irregularity of a sequence. In this section, we will use these differences to devise measures for discriminating between the associated models.

4.3.1 Estimating Stationarity

When discussing seasonal sequences, we observed that the autocorrelation $\mathcal{A}_t$ of a sequence t is seasonal with the same period as t, if t is stationary. We also observed that wide-sense stationarity implies that $\mathcal{A}_t$ is time-independent. Further, one should also note that a z-normalized sequence $z = \frac{t-\mu_t}{\sigma_t}$ also has a z-normalized autocorrelation. From this, we can deduce: If t is stationary, then $\mathcal{A}_z$ will be 'almost' stationary with mean $\mu_{\mathcal{A}_z} = 0$. If one now considers that $\mathcal{A}_z(0) = 1$, we arrive at the following insight: *If t is stationary, $\mathcal{A}_z$ will quickly produce value $\mathcal{A}_z(\tau_0) \leq 0$* [13].

This brings us to a simple algorithm for estimating stationarity:

Algorithm 4 Stationarity Estimation: $\Gamma(t)$

```
Require: $t$
  $n \leftarrow \text{length}(t)$
  $z \leftarrow (t - \mu_t)/\sigma_t$    z-normalization
  $\alpha \leftarrow \mathcal{A}_z(\tau = 1, ..., n)$    autocorrelation
  $\xi \leftarrow \alpha^{-1} \leq 0$    find first zero interception
  return $1 - \min(\xi)/n$
```

Algorithm 4 z-normalizes the data, computes the autocorrelation function, finds the first value less than or equal to zero and returns its index divided by the number of observations. The returned value ranges in $[0, 1]$, where values close to 0 indicate low stationarity and values close to 1 indicate high stationarity. It is important to note that there is no threshold beyond which sequences are considered stationary or nonstationary, since such a threshold cannot be objectively determined for arbitrary data. As we will see later, classifying sequences as either ′stationary′ or ′nonstationary′ with a decision boundary is not even required for anomaly detection. Instead, we will use the ′amount′ of stationarity to compute distances between ′more stationary′ and ′less stationary′ segments.

Of course, there exist other methods for determining the stationarity of a sequence such as the Augmented Dickey–Fuller test. However, this method and similar stationarity test are usually computationally expensive. Therefore, Algorithm 4, which has a computational complexity of $\mathcal{O}(n \log n)$will be preferred in the this thesis.

4.3.2 Estimating Volatility

From the above observations about the regularity of sequences, the usefulness of a measure for the amount of noise, smoothness and similar characteristics becomes evident. For this, we will introduce the concept of sequence volatility. Let $T(i)$ be a continuous, piece-wise linear function that passes through all points of $t = t_1, ..., t_n$ and is linear in between. T can be obtained with a linear interpolation ψ of t

$$T(k) = \psi(k) = t_i + \frac{t_j - t_i}{j - i}(k - i), \tag{4.12}$$

where indices $i < k$ and $j > k$ mark the location of the two closest observations in t.

T is what is usually depicted when a data sequence is plotted. An example for this can be seen in Figure 4.4.

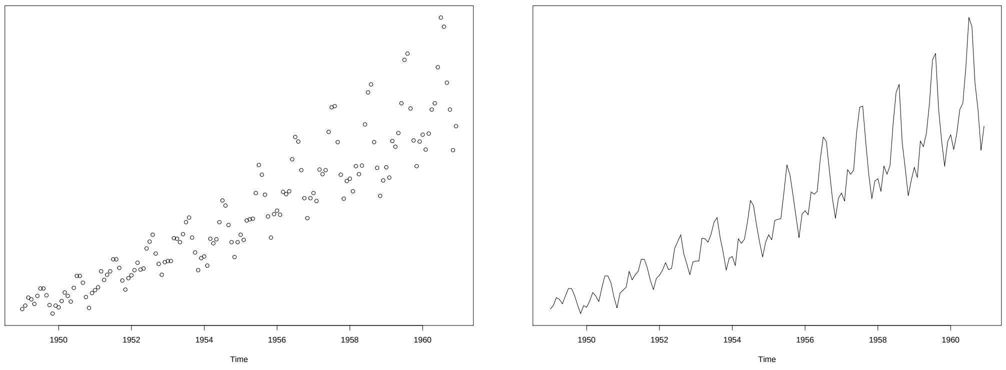


Figure 4.4: A raw data sequence of monthly air passengers t (left) and the interpolation T (right).

Furthermore, let $\mathcal{M} = \mathcal{M}_1, ..., \mathcal{M}_\nu$ be all solutions for

$$T'(i) = 0 \wedge T''(i) \neq 0$$

where T' and T'' are the first and second derivative of T respectively. Consequently, $\mathcal{M}$ contains the locations of all minima and maxima of T. The sequence volatility v_t is then given by

$$v_t = \frac{\nu}{n - 1} \tag{4.13}$$

Intuitively, sequence volatility can be seen as the percentage of observations in a sequence where the direction changes from "up" to "down" or vice

Algorithm 5 Volatility Estimation: $\Lambda(t)$

```
Require: $t$
  $n \leftarrow \text{length}(t)$
  $d \leftarrow \Delta_1(t)$    take discrete difference
  $d(d^{-1} > 0) \leftarrow 1$    label all inclines with 1
  $d(d^{-1} < 0) \leftarrow -1$    label all declines with -1
  $\eta \leftarrow d(d^{-1} \neq 0)$    remove saddle points
  $\eta \leftarrow \Delta_1(\eta)$    find incline/decline changes
  return $\text{length}(\eta^{-1} \neq 0)/n$
```

versa. Measuring the volatility of a sequence can be achieved with Algorithm 5. Δ_1 is the discrete difference of a sequence at lag 1. This means that for an exemplar sequence $\grave{t} = 1, 3, 4, 2$, the discrete difference at lag 1 would be

$$\Delta_1(\grave{t}) = (3-1), (4-3), (2-4) = 2, 1, -2$$

The sequence volatility in this case would be $v_{\grave{t}} = \frac{1}{3}$. This makes sense since in a sequence of length 4, the "up" or "down" direction can at the most change 3 times, and changes only once.

4.4 Collective Anomaly Detection with STAVE

After introducing algorithms for stationarity and volatility estimation, it is possible to construct a collective anomaly detection algorithm based on them. Intuitively, the idea is to compute these properties of local subsequences, and then compare them against the properties of the remaining sequence. Let

$$s = s_1, s_2, ..., s_{n-w+1} = t_1...t_w, t_2...t_{w+1}, ..., t_{n-w+1}...t_n$$

be the set of subsequences of sequence t extracted by a sliding window of length w. Then the Stationarity and Volatility (STAVE) distance sequence θ can be computed with Algorithm 6. This algorithm estimates stationarity and volatility of sequence t, then estimates them also for all sliding window subsequences of t, and returns the Euclidean distance from each subsequence to the full sequence.

Algorithm 6 Stationarity and Volatility Distance: $\theta(t)$

```
Require: $t$
  $\Gamma_t = \Gamma(t)$
  $\Lambda_t = \Lambda(t)$
  for $i = 1...n - w + 1$ do
    $\theta_i = \sqrt{(\Gamma_t - \Gamma(s_i))^2 + (\Lambda_t - \Lambda(s_i))^2}$
  end for
  return $\theta$
```

It is important to note that Γ_t and Λ_t are the STAVE estimates of t and not of $t \setminus s_i$. If we assume that the window is much shorter than the total sequence, then comparing the window against the remaining sequence, is almost equivalent to comparing it with the entire sequence including the window itself. Mathematically, this can be written as

$$\Gamma(t \setminus s_i) \approx \Gamma(t) \quad \text{and} \quad \Lambda(t \setminus s_i) \approx \Lambda(t) \qquad \text{if} \quad w \ll n.$$

This property is important to achieve an acceptable computational complexity, which is a problem in several of the methods discussed in Chapter 2.

The STAVE sequence θ is useful for collective anomaly detection, since it separates regions with different stationarity or volatility. An example for this is depicted in Figure 4.5.

This property of the STAVE sequence can be used to detect collective anomalies. Let

$$\Omega = \Omega_1, \Omega_2, ..., \Omega_{n-2w+1} = \theta_1...\theta_w, \theta_2...\theta_{w+1}, ..., \theta_{n-2w+1}...\theta_{n-w+1}$$

be all subsequences extracted from θ with a sliding window with the same length w as above. If one clusters Ω with two centroids, then the cluster with less instances assigned to it will likely contain the entire collective anomaly c [2].

Clustering Ω with two centroids can be achieved with the k-means Algorithm 7. This algorithm assigns every sequence in Ω to one of two clusters.

[2] The difference in cluster sizes is an indicator of how anomalous the smaller cluster is. If both clusters have similar sizes, then likely neither of them is anomalous. However, this would require a hard decision boundary beyond which a cluster is either small enough or not

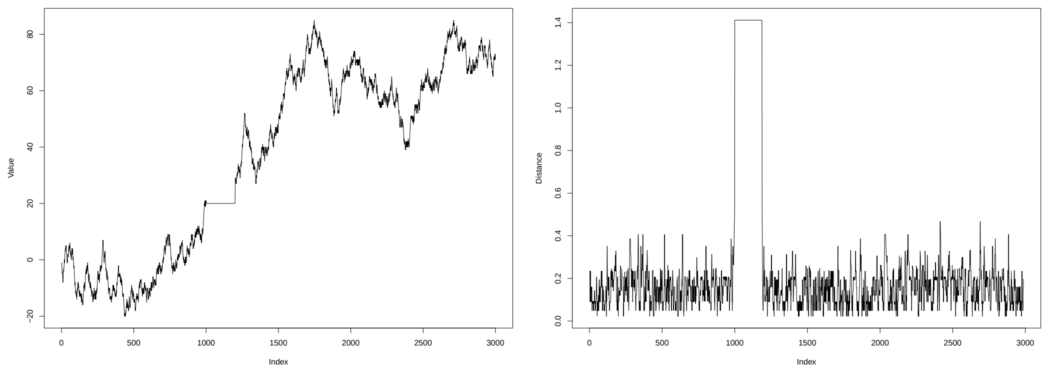


Figure 4.5: A random walk sequence with an anomaly (left) and the corresponding STAVE distance(right). The location of the collective anomaly matches the peak in the distance function.

Algorithm 7 K-Means Clustering: kmeans(Ω, K) [23]

```
for $i \leftarrow 1...k$ do
    $\mu_k \leftarrow$ randomSeed()
    $\nu_k \leftarrow \{\}$
end for
while $\mu \neq \nu$ do
    $\nu \leftarrow \mu$
    for $i \leftarrow 1...k$ do
        $\alpha_i \leftarrow \{\}$
    end for
    for $i \leftarrow 1...n - 2w + 1$ do
        $j \leftarrow \arg\min_{j'} |\mu_{j'} - \Omega_i|$
        $\alpha_j \leftarrow \alpha_j \cup \Omega_i$
    end for
    for $i \leftarrow 1...k$ do
        $\mu_i \leftarrow \frac{1}{|\alpha_i|} \Sigma_{\Omega \in \alpha_i} \Omega$
    end for
end while
return $\mu, \alpha$
```

As initial clusters, one can compute the mean of each window, and choose the windows with the highest and the lowest mean. Since STAVE separates regions by normal and anomalous STAVE properties, one cluster α^N will represent normal subsequences. The other cluster $\alpha^A = \alpha_1^A, ...\alpha_l^A$ will contain all stationarity and volatility anomalies. Of course, there may be other types of anomalies in t, yet finding all of them with a parameter-free method remains a task for the future.

To extract collective anomalies from α^A, one simply has to find the longest connected sequence of windowed observations. This can be achieved with Algorithm 8.

Algorithm 8 Collective Anomaly Extraction: $E(\alpha^A)$

```
    $\chi \leftarrow \Delta_1(\alpha^A)$
    $\chi \leftarrow \text{append}(1, (\chi^{-1} \neq 1), l)$
    $\chi \leftarrow \text{append}(1, \Delta_1(\chi))$
    $\text{sum} \leftarrow \sum \chi$
    **return** $\alpha^A_{\text{sum}}, \alpha^A_{\text{sum}+1}, ..., \alpha^A_{\text{sum}+\max(\chi)}$
```

A critical parameter of the procedure mentioned above is the window length w, since it determines the minimum length of the anomaly. Therefore, it is set to the smallest possible value $w = 4$. Windows shorter than 4 observations would cause errors in the above algorithms. This constant window length is also very convenient from the perspective of computational complexity. However, it is debatable if this is truly the best choice. In the next chapter, several different window lengths will be investigated, up to $w = \sqrt{n}$, which is considered as upper boundary for Equation 2.6. Stationarity estimation with Algorithm 4 has a complexity of $\mathcal{O}(n \log n)$, and this is computed for all $n - w + 1$ windows of length w, resulting in $\mathcal{O}(nw \log w)$. If the window length w is constant, than this can be reduced to $\mathcal{O}(n)$. An overview of the computational complexities of all methods used in this thesis is depicted in Table 4.1.

[3]In a general case, Algorithm 7 has a computational complexity of $\mathcal{O}(ndki)$ [23], where n is the number of instances to be clustered, d the dimensions of the instances, k the number of clusters, and i the number of iterations. In our setting d, k and are $2 = \mathcal{O}(1)$. The number of iterations i is also constant, since it does not depend on the input since.

Step	Computational Complexity
STAVE properties of full sequence	$\mathcal{O}(n \log n)$
STAVE properties of subsequences	$\mathcal{O}(nw \log w)$
STAVE distance θ	$\mathcal{O}(n)$
K-means clustering[3]	$\mathcal{O}(n)$
Extracting anomalies	$\mathcal{O}(1)$
Total	$\begin{cases} \mathcal{O}(n \log n) & w = 4 \\ \mathcal{O}(n\sqrt{n} \log \sqrt{n}) & w = \sqrt{n} \end{cases}$

Table 4.1: The computational complexities of all methods used for collective anomaly detection.

To summarize, the method presented here for collective anomaly detection consists of the following steps:

- Estimate stationarity and volatility of full sequence
- Estimate the same properties of all subsequences extracted by a sliding window.
- Compute the distance from all subsequences' to the full sequences' properties, giving the STAVE sequence.
- Extract subsequences from the STAVE sequence with a sliding window.
- Cluster these subsequences with 2 centroids.
- Find the longest connected series of subsequences in the smaller cluster.

For the two sliding windows, the same window length is used. Clustering is achieved with the k-means algorithm.

5 Evaluation

In this chapter, the above mentioned methods and techniques are evaluated in a practical setting. First, the test setup is described. Then, the results are presented in various tables and plots. Finally, the results are discussed and interpreted.

5.1 Setup

5.1.1 Procedure

To evaluate the algorithm proposed in Chapter 4, it is contrasted with other comparable anomaly detection methods in an experimental study. These methods are Entropy-based anomaly detection (ENTROPY, Algorithm 1), the brute force discords algorithm (DISCORDS, Algorithm 2), and HOT SAX. At first, this might seem an unfair comparison, since all of these methods have parameters that need to be tuned. However, completely parameter-free collective anomaly detectors are yet to be proposed by literature.

ENTROPY was not used as presented in Algorithm 1. Rather, we used an accelerated version which only tests for subsequences of the predefined length m. Consequently, the length of the collective anomaly was provided to the algorithm in advance.

Similar to ENTROPY, the only parameter of DISCORDS was the length of the discord. This value was also provided in advance. Further, the implementation of Algorithm 2 was accelerated with the runtime improvements suggested by Zhu et al. [28].

For evaluating HOT SAX, several parametrizations where tested. The discord length was always set to the length of the collective anomaly in the current test case. While this facilitated the tests for HOT SAX, it still was necessary to achieve comparable results. The remaining parameters, PAA size, alphabet size and normalization threshold, were tuned towards the test set. The best performing setup was PAA size 6, alphabet size 3, and normalization threshold 1.

The preliminary method BACKSIM was also included in the test setup. The upper and lower window boundaries were set to the default values $\sqrt{n}$ and $\sqrt[3]{n}$ respectively. The outlier threshold was set to $\mu \pm 3\sigma$ according to the standard three-sigma-rule [20]. A different outlier threshold $\mu \pm 2\sigma$ was also tested, to investigate the validity of the three-sigma rule in this setting. Further, instead of a threshold, k-means clustering for mining anomalies in the backwards similarity distance function was also tested, since this does not require a threshold.

To compare STAVE with the other two approaches, it was first tested with window length $w = 4$ and $w = \sqrt{n}$. Afterwards, an additional test was conducted, where the window length was varied from 4 to 68 [1]. The purpose of this test was to assess the impact of the selected window length, since the STAVE is supposed to be a parameter-free algorithm.

Further, since it was not self-evident that the STAVE measures are well-suited for anomaly detection, they were also replaced with more conventional measures. For stationarity, the alternative measure was the Augmented Dickey-Fuller (ADF) test. This test has a lag-parameter, which was set to zero, reducing the test to the non-augmented Dickey-Fuller test. The reason for this was that the lagged version of the test has a considerably slower runtime, making it infeasible to apply it to all sliding-window subsequences of a long sequence. For volatility, the alternative measure was variance (VAR), the second statistical moment. With the alternative measures and window sizes, a total of 8 different combinations were available and evaluated on the test data.

[1]The longest required window length predicted by theory should be 110 observations, since no sequence was longer than 110^2 observations. However, the results converged at $w \approx 55$

Finally, a method returning a uniformly random subsequence was also included, to assure that all other methods are meaningful for detecting collective anomalies.

An overview of all tested methods and their parameters can be seen in Table 5.1.

Method	**Description**	**Parameters/Default values**
ENTROPY	Algorithm 1 with only one window size	Anomaly Length
DISCORDS	Algorithm 2	Anomaly Length
HOT SAX	Algorithm 2 with SAX and runtime optimization	Anomaly Length, PAA size=6, alphabet size=3, threshold=1
BACKSIM$_{2\sigma}$	As in Section 3.3	$p = \sqrt[3]{n}$ $q = \sqrt{n}$ $\delta = \mu \pm 2\sigma$
BACKSIM$_{3\sigma}$	As in Section 3.3	$p = \sqrt[3]{n}$ $q = \sqrt{n}$ $\delta = \mu \pm 3\sigma$
BACKSIM$_k$	As in Section 3.3 with Algorithm 7	$p = \sqrt[3]{n}$ $q = \sqrt{n}$
STAVE$_4$	Algorithm 6 with Algorithm 4 and Algorithm 5	$w = 4$
STAVE$_{\sqrt{n}}$	Algorithm 6 with Algorithm 4 and Algorithm 5	$w = \sqrt{n}$
ADFVE$_4$	Algorithm 6 with ADF and Algorithm 5	$w = 4$ $\text{lag} = 0$
ADFVE$_{\sqrt{n}}$	Algorithm 6 with ADF and Algorithm 5	$w = \sqrt{n}$ $\text{lag} = 0$
STAVAR$_4$	Algorithm 6 with Algorithm 4 and variance	$w = 4$
STAVAR$_{\sqrt{n}}$	Algorithm 6 with Algorithm 4 and variance	$w = \sqrt{n}$
ADFVAR$_4$	Algorithm 6 with ADF and variance	$w = 4$ $\text{lag} = 0$
ADFVAR$_{\sqrt{n}}$	Algorithm 6 with ADF and variance	$w = \sqrt{n}$ $\text{lag} = 0$
RAND	Returns a random subsequence	

Table 5.1: All test methods and their parameters

5.1.2 Test Data

The dataset used in the evaluation consisted of 40 test sequence, each containing one distinct collective anomaly. 20 of these sequences originate from a mixture of practical domains, ranging from medical data over financial time series to space shuttle valve cycles. The main criterion for selecting them was the occurrence of an anomaly, that one can clearly visually distinguish from the remaining observations. The other 20 sequences were synthetic data. They were generated as mixtures of seasonality, noise, random walks and trends, with one clear anomaly of random length injected at a random position. Visual depictions of all test sequences can be found in the Appendix.

A histogram summarizing the lengths of the test sequences is depicted in Figure 5.1. The average sequence length was 2215 observations. The longest sequence consisted of 11251 data points, while the shortest had 60 observations. On average, the sequences from practical domains where longer than the synthetic sequences, with 2669 and 1762 observations respectively.

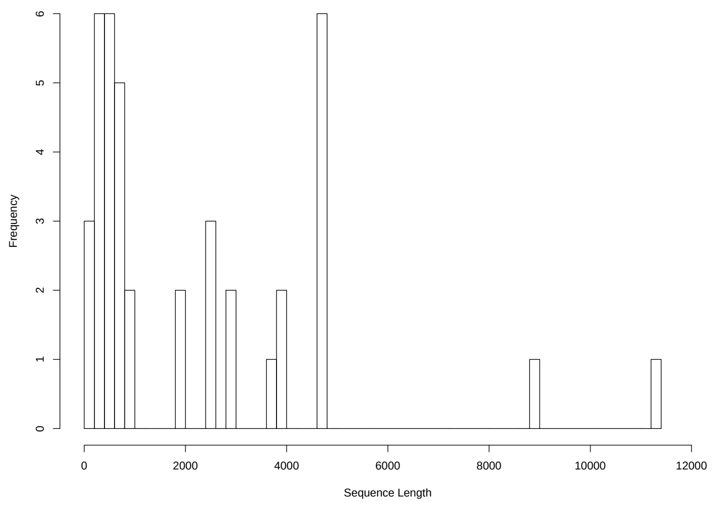


Figure 5.1: Histogram of sequence lengths found in the test data

The lengths of the anomalies in the test sequences are summarized in Figure 5.2. The mean anomaly length of all test cases was 143 observations. The shortest and the longest anomaly had 9 and 950 data points respectively. Similar to the full sequence lengths, the anomalies found in the practical data were on average longer than the synthetically induced anomalies. The former had a mean of 172 observations, while the latter averaged at 114 observations.

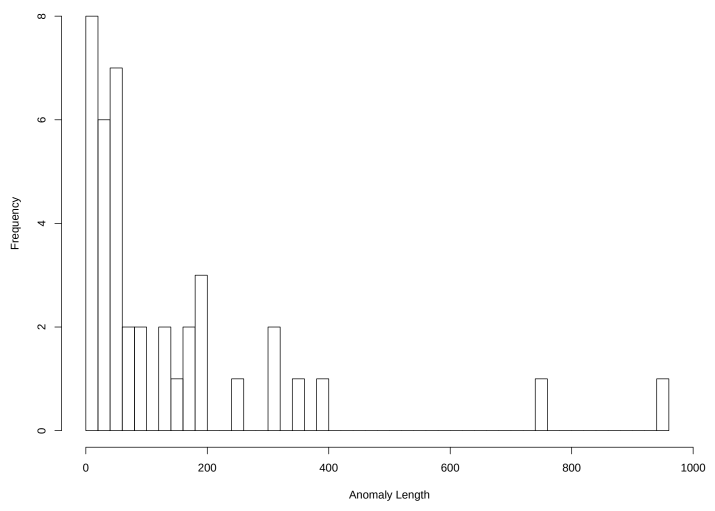


Figure 5.2: Histogram of anomaly lengths found in the test data

A histogram of the ratio of anomaly length to full sequence lengths can be seen in Figure 5.3. On average, the 9.9% of the data points in a sequence were anomalous. The longest anomaly made up 40% of the full sequence, while the shortest had a length ratio of 0.41%.

5.1.3 Measures

The detection task for which STAVE was devised is the identification an anomalous subsequence in a longer data sequence. To test this, we compared the expected anomalous observations with the identified observations. Commonly used measures for such a task are precision and recall. Precision is defined as the partition of identified observations that are correct. Mathematically, this is frequently expressed as

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{5.1}$$

where ”tp” indicates a true positive, and ”fp” a false positive. Recall is the partition of anomalous observations that where identified, mathematically

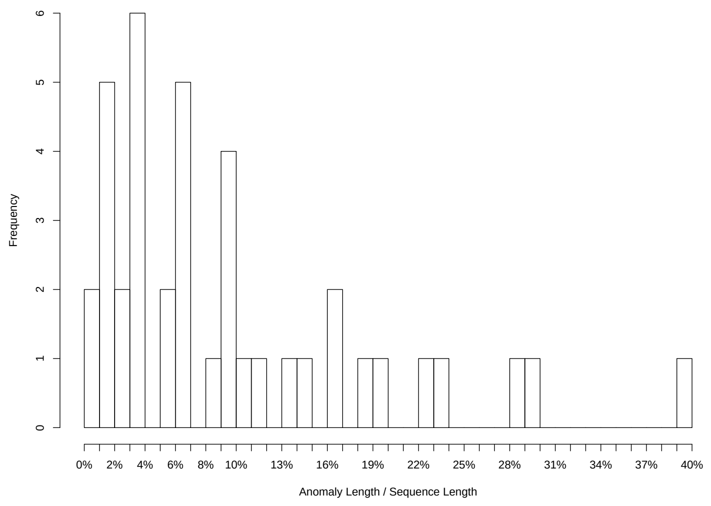


Figure 5.3: Histogram of the anomaly length ratios $\frac{m}{n}$

expressed as

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{5.2}$$

wherer "fn" means false negative. While precision and recall are measure different aspects of classifiers [2], it is desirable to combine them to have a general representation of a classifiers performance. This can be done with the F_ι measure, which is defined as

$$F_\iota = (1 + \iota^2) * \frac{\text{precision} * \text{recall}}{\iota^2 * \text{precision} + \text{recall}} \tag{5.3}$$

While the F_ι measure is widely used in data science literature, it has the disadvantage that true negatives (tn) are discarded. This can lead to a misinterpretation of results, if a detection algorithm is skewed towards positive classification [21]. Therefore, the Matthews Correlation Coefficient (MCC) [19] is included as alternative evaluation measure. It is defined as

$$\text{MCC} = \frac{\text{tp} * \text{tn} - \text{fp} * \text{fn}}{\sqrt{(\text{tp} + \text{fp})(\text{tp} + \text{fn})(\text{tn} + \text{fp})(\text{tn} + \text{fn})}} \tag{5.4}$$

[2] A classifier detects the class of an object. In the context of anomaly detection, the classes are "normal" and "anomalous"

Unlike the above mentioned measures, it does not range from 0 to 1, but instead from −1 to 1, which is similar to the Pearson Correlation Coefficient.

Since precision, recall, F_ι and MCC can be computed for every individual sequence, we use mean-precision, mean-recall, mean-F_ι and mean-MCC to summarize detection performances on various sequences.

5.2 Results

Table 5.2 shows the scores achieved by the individual variants of the Algorithms discussed above. The highest precision and $F_{0.1}$ scores were achieved

	Precision	Recall	$F_{0.1}$	MCC	Runtime
ENTROPY	0.245	0.241	0.245	0.253	57.198s
DISCORDS	0.458	0.446	0.458	0.501	4482.697s
HOT SAX	0.500	0.500	0.500	**0.503**	**20.614**s
$\text{BACKSIM}_{2\sigma}$	0.405	0.253	0.391	0.264	90.822s
$\text{BACKSIM}_{3\sigma}$	0.402	0.204	0.361	0.225	90.825s
BACKSIM_{k}	0.354	0.338	0.352	0.318	82.702s
STAVE_4	0.402	0.345	0.395	0.356	25.024s
$\text{STAVE}_{\sqrt{n}}$	0.416	**0.507**	0.415	0.421	29.058s
ADFVE_4	0.296	0.236	0.293	0.245	298.624s
$\text{ADFVE}_{\sqrt{n}}$	0.288	0.365	0.287	0.302	154.765s
STAVAR_4	**0.574**	0.375	**0.554**	0.433	27.341s
$\text{STAVAR}_{\sqrt{n}}$	0.370	0.453	0.368	0.376	31.231s
ADFVAR_4	0.459	0.276	0.445	0.319	315.805s
$\text{ADFVAR}_{\sqrt{n}}$	0.402	0.501	0.401	0.411	155.219s
RAND	0.104	0.337	0.104	0.133	0.004s

Table 5.2: Mean measure scores per algorithm

by STAVAR$_4$. The highest recall was achieved by STAVE$_{\sqrt{n}}$. HOT SAX had the best MCC score and the shortest runtime [3].

The MCC scores of the algorithms per individual test case are depicted in Figure 5.4. There were several test cases were no algorithm variant found any true positives, such as test case 31 or test case 34. There was also a visible positive correlation between STAVE$_{\sqrt{n}}$ and STAVE$_4$. No test case was fully solved (MCC$=1$) by all algorithms.

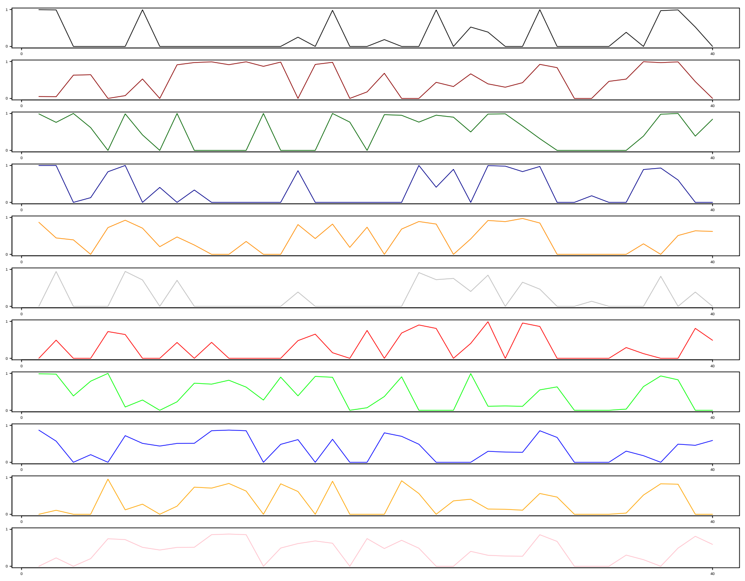


Figure 5.4: MCC scores per test case. Algorithms from Top to Bottom: ENTROPY, DISCORDS, HOT SAX, STAVE$_4$, STAVE$_{\sqrt{n}}$, ADFVE$_4$, ADFVE$_{\sqrt{n}}$,STAVAR$_4$, STAVAR$_{\sqrt{n}}$,ADFVAR$_4$, ADFVAR$_{\sqrt{n}}$

Figure 5.5 shows the mean MCC achieved by STAVE tuned with various different window lengths. The test was aborted after $w = 68$, since the mean MCC converged after $w \approx 55$. The average score of all tested window lengths was 0.382, although this would very take the value at the point of convergance $MCC = 0.346$ if longer windows were added. The runtime varied from 25 seconds to 39 seconds, positively correlating with the window length.

[3]RAND was not included in the runtime ranking.

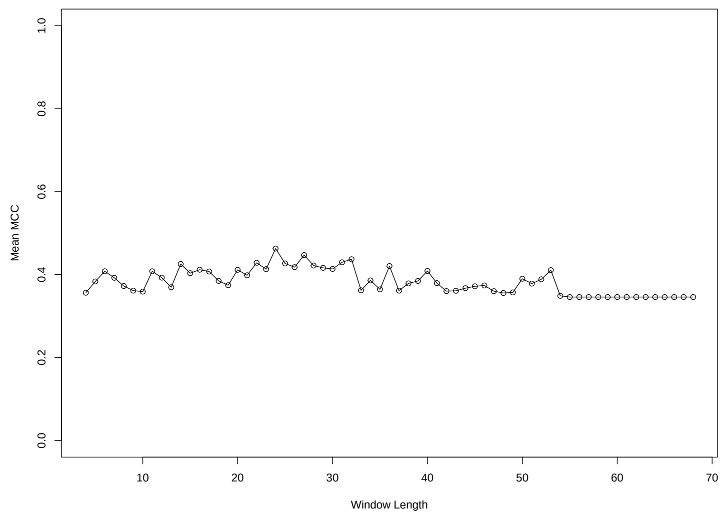


Figure 5.5: Mean STAVE MCC scores by window length

5.3 Discussion

The purpose of this thesis was to develop an algorithm that can detect collective anomalies in sequential data without any user input other than the sequence itself. If one only looked at results in Table 5.2, one could assume this objective was not achieved. It appears that both discord-based approaches DISCORDS and HOT SAX were superior to STAVE, since their overall scores were higher.

However, the 8 variations of STAVE did not require the length of the anomalies in advance. This is a crucial advantage over the other algorithms. In practical settings, anomalies cannot be expected to have a predefined length. The performance of the STAVE variants was not clearly inferior to HOT SAX and DISCORDS, although it was not given any prior information about the data. Additionally, HOT SAX performed well because all of its 4 parameters were optimally tuned. It is therefore not surprising that a likely overfitted method achieves high precision and recall on the dataset it was tuned to. Further, the runtime of HOT SAX can be misleading. Several similar parameterizations did not terminate in reasonable time.

When comparing DISCORDS with HOT SAX, it appears that DISCORDS is the better overall choice. It has only one parameter, and when running Algorithm 2, one can also create a matrix profile [4] in parallel with only linear additional space. The runtime of DISCORDS was much slower, yet there have already been several suggestions how the algorithm can be further accelerated [28]. ENTROPY appeared to be inferior to the other algorithms. This was likely due to entropy not truly being a measure for normality as discussed above.

Both threshold variants of BACKSIM achieved acceptable mean precision scores, yet recall and MCC scores were similarly bad as those of ENTROPY. MCC likely is more relevant than the other measures, since it is the only of the 4 measures that takes all classified observations into consideration. It appears that BACKSIM with a threshold detects too few anomalies, and that the threshold is of critical importance, which is disadvantageous since it depends on the data. The clustered variant of BACKSIM appears to improve upon these deficits, since its overall scores seem more reliable and it requires one less parameter. However, in total BACKSIM appears to be inferior to most other algorithms in its current implementation.

Overall, it appears that most STAVE variants were able to compete with the parametric alternatives, which is certainly a success for parameter-free methods. Whether STAVE is truly parameter-free is debatable. The window length seems to influence the overall performance, yet the results in Figure 5.5 did not exhibit great variance and appeared to be stationary. If we considers that famous data science algorithms have in the past been claimed to be parameter-free, such as the matrix-profile, we can conclude that STAVE is at least ”more” parameter-free than other methods. Further, STAVE does not use any internal empirically chosen constants during its computation. A clear drawback of STAVE is that it cannot determine if an anomaly is present in the data. We did not succeed in finding an answer for this problem which does not need thresholds, constants or parameters.

When comparing STAVE_4 and $\text{STAVE}_{\sqrt{n}}$ it seems as if the latter variant performs better. Using window length 4 has a faster runtime, but adjusting the window length to the full sequence length seems advisable. Further,

[4] A sequence of the distance to a subsequence’s closest neighbor as proposed by Yeh et al. [27]

STAVE struggled with test sequences which had repeating patterns that were much longer than the window length. This was mainly due to stationarity and volatility changing within a cycle, which then caused the clustering to split the seasonality in two clusters instead of one. For strongly seasonal sequences, a window length close to the number of observations in one season is likely a good choice.

Using the Dickey-Fuller test for stationarity estimation appears to have more deficits than benefits. ADFVE and ADFVAR were both inferior to STAVE and STAVAR, regardless of the window choice. This might be due to not using the lagged version of the test, but this was simply too computationally expensive. We aborted the lagged ADF during testing, since it was evident that it would not terminate in any reasonable time, having spent over an hour on the first test sequence without result.

Variance appears to be a good alternative measure for volatility. One can assume that it is even superior to Algorithm 5, since STAVAR_4 was the only algorithm that achieved the highest value in two different measures. However, when comparing precision and $F_{0.1}$ scores, it becomes evident that these two highly correlate and likely contain much mutual information. This may limit the impact of these two measures to that of a single measure.

Altogether, the results demonstrate that parameter-free collective anomaly detection is possible without deep learning and millions of data points. STAVE was capable of detecting anomalies in short, single data sequences without knowing anything about them in advance. The only requirement for detecting anomalies is that they differ in stationarity and/or volatility from the remaining sequence, which is true for most anomalies that can be visually identified by a human.

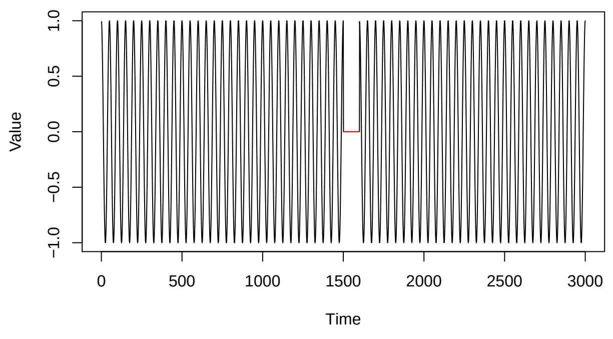
6 Conclusion

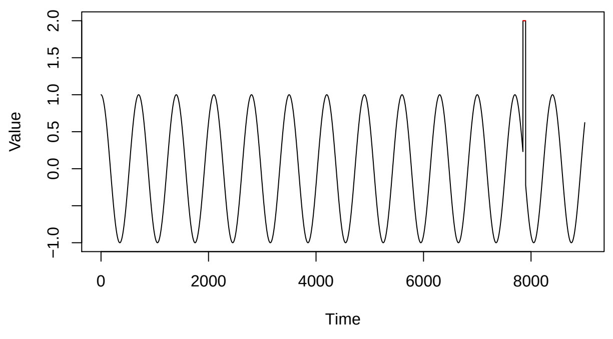
Developing a parameter-free algorithm that detects collective anomalies in sequential data is a challenging problem. Stationarity and Volatility Estimation (STAVE) can detect collective anomalies without tuning any parameters, and is still able to compete with parametric methods. Further, STAVE was not designed for a specific domain and can be used in various different areas. STAVE can be seen as a proof-of-concept, demonstrating that parameter-free collective anomaly detection without any prior knowledge about the data is indeed possible - even if it is much easier to propose parametric solutions that can achieve similar results. Further, STAVE can be used on a single short data sequence, which cannot be done with contemporary deep learning techniques.

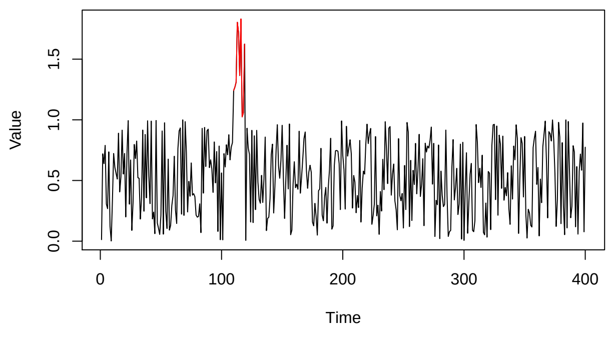
Future work might include the development of a streaming algorithm, that can detect anomalies in linear time and with constant space complexity. With a few adjustments, STAVE can be redesigned to a streaming algorithm, since it summarizes many observations with two real numbers. The newly developed algorithms for estimating stationarity and volatility might be used in various different applications in the future.

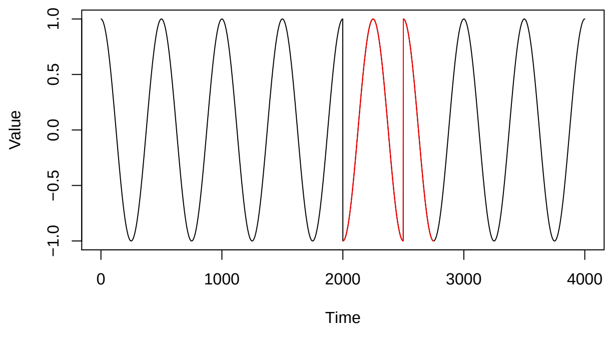
Appendix

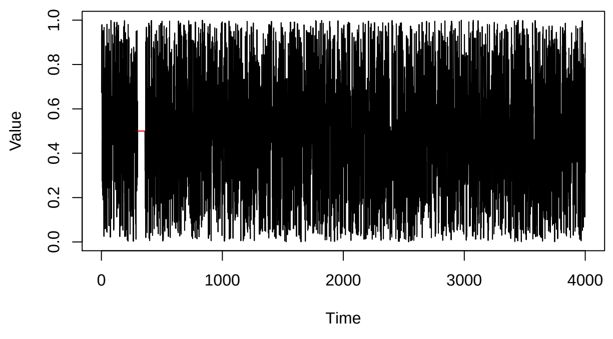
Sequences - Black
Anomalies - Red

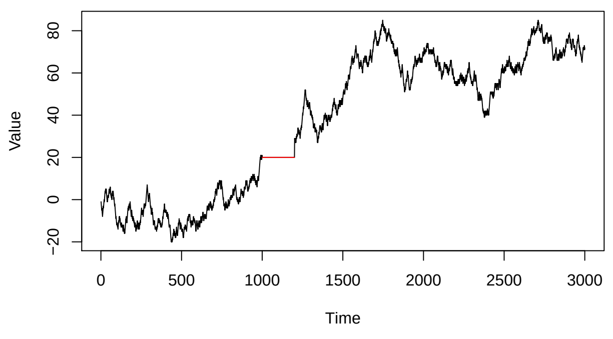


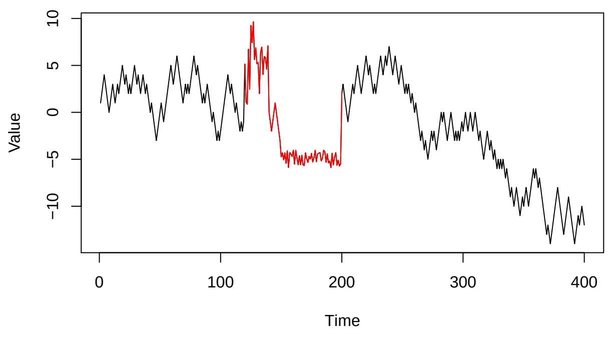


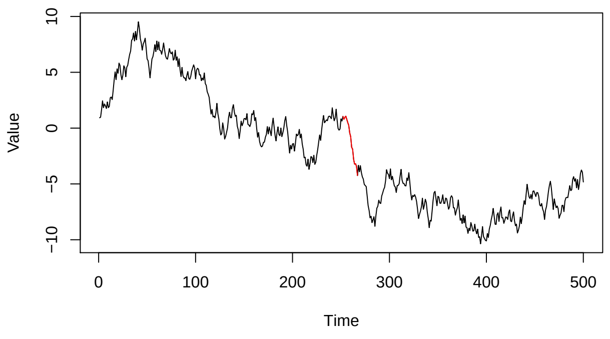


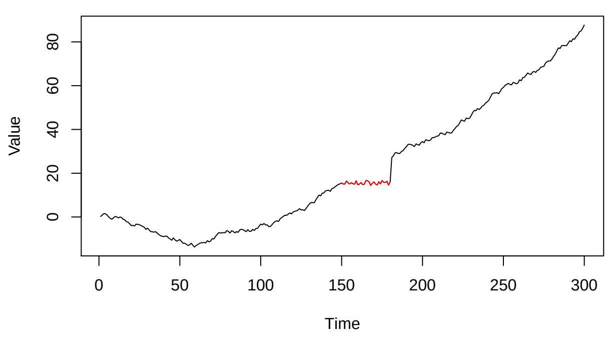


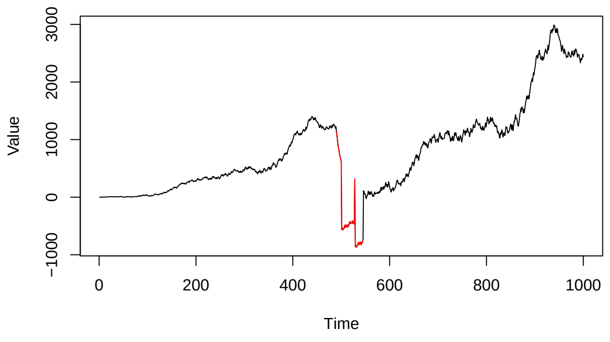


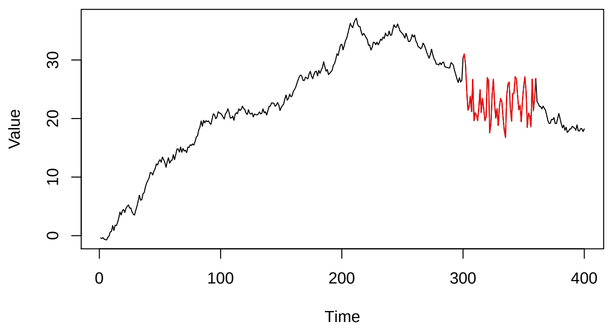


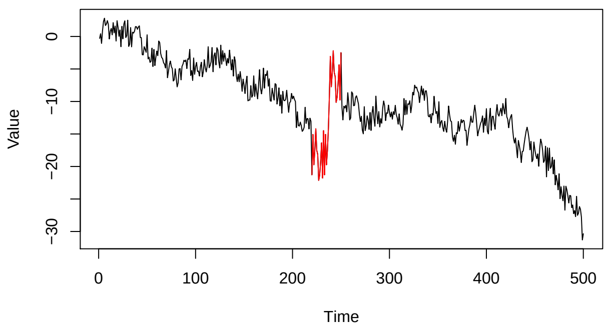

Value
Time
10
5
0
−5
−10
0
100
200
300
400

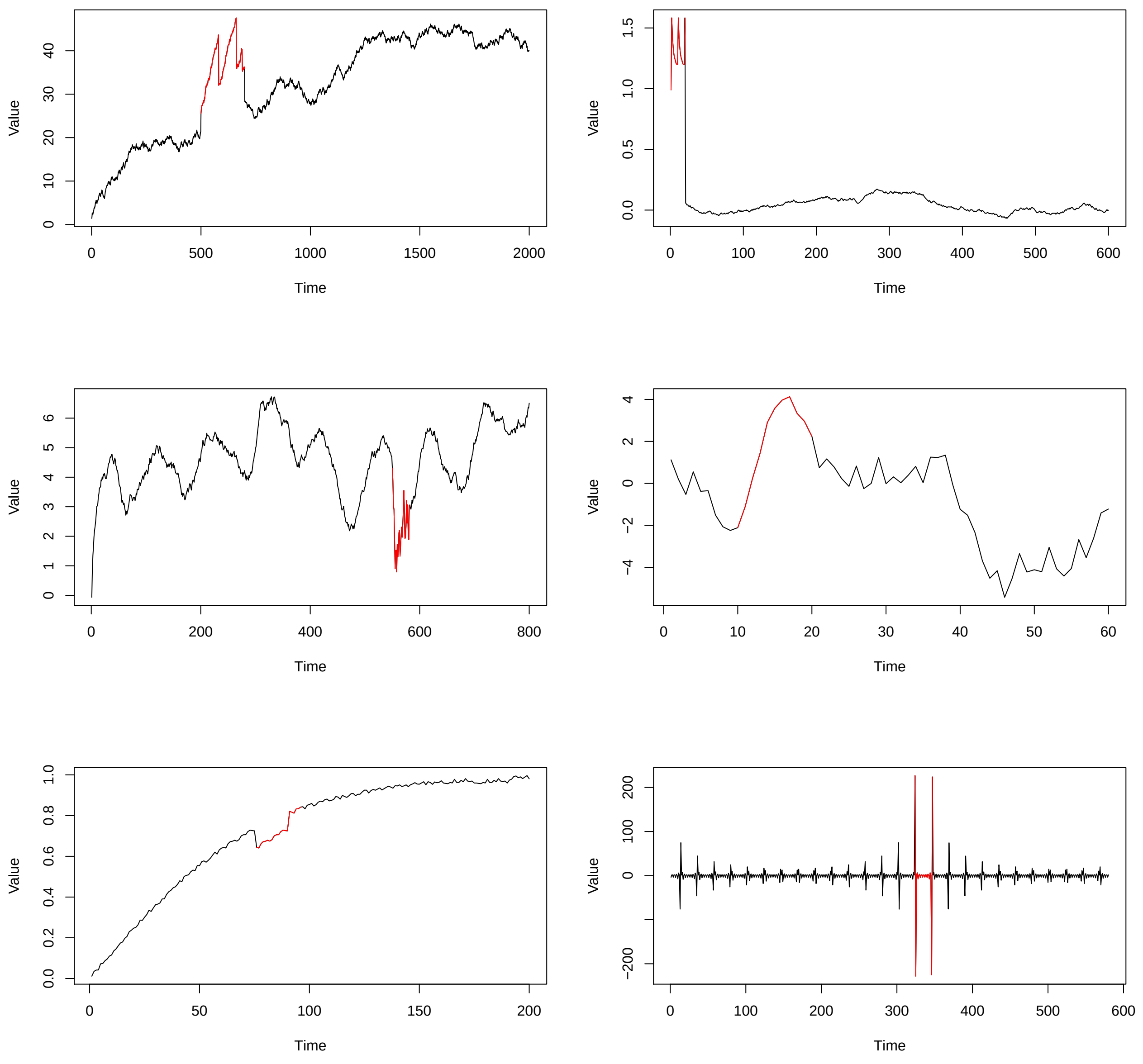

Value
Time
10
5
0
−5
−10
0
100
200
300
400
500

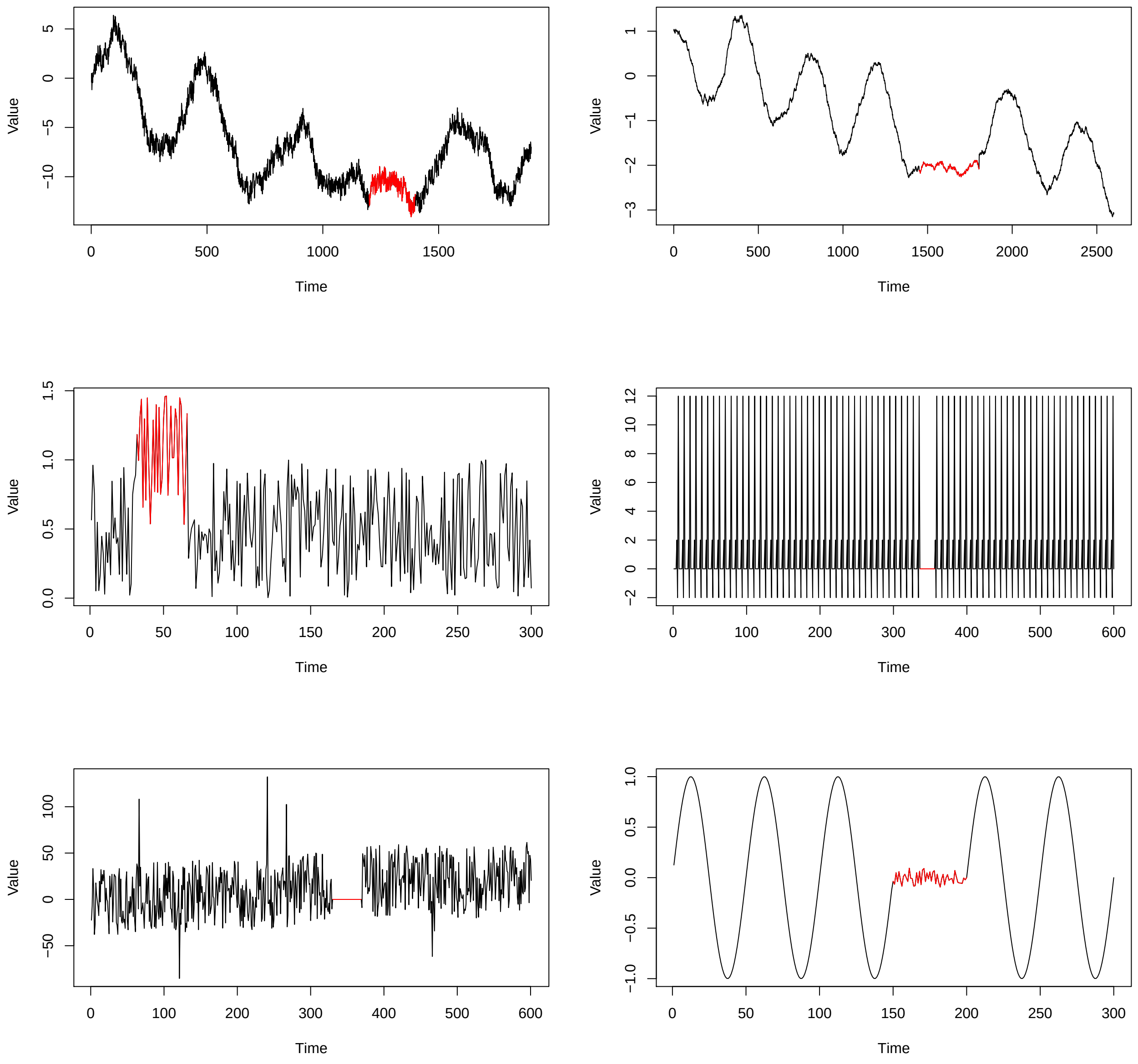

Value
Time
80
60
40
20
0
0
50
100
150
200
250
300


Value
Time
3000
2000
1000
0
−1000
0
200
400
600
800
1000

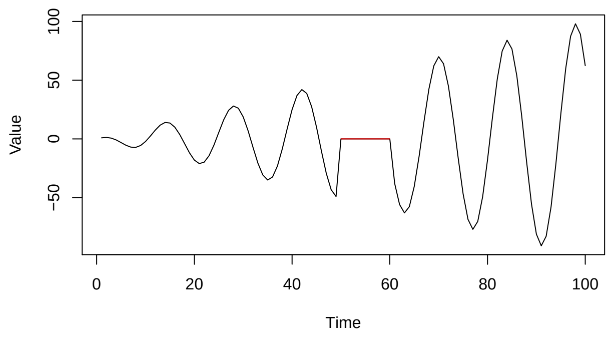

Value
Time
30
20
10
0
0
100
200
300
400

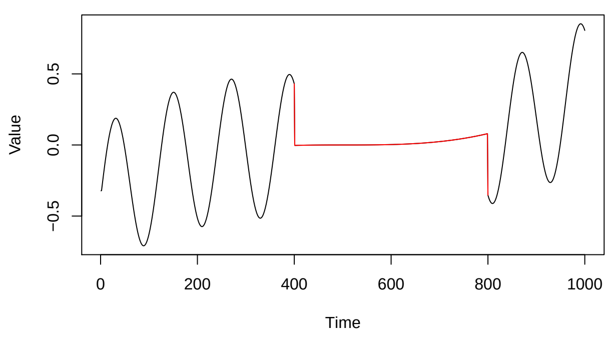

Value
Time
0
−10
−20
−30
0
100
200
300
400
500

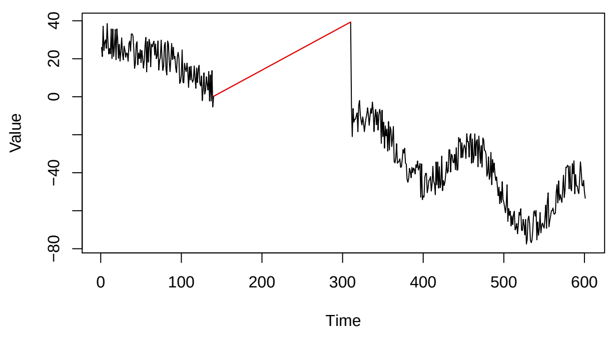

Value
Time
0
500
1000
1500
2000
0
10
20
30
40
0.0
0.5
1.0
1.5
100
200
300
400
500
600
800
0
1
2
3
4
5
6
−4
−2
2
4
10
20
30
40
50
60
0.0
0.2
0.4
0.6
0.8
1.0
50
100
150
200
−200
0
100
200

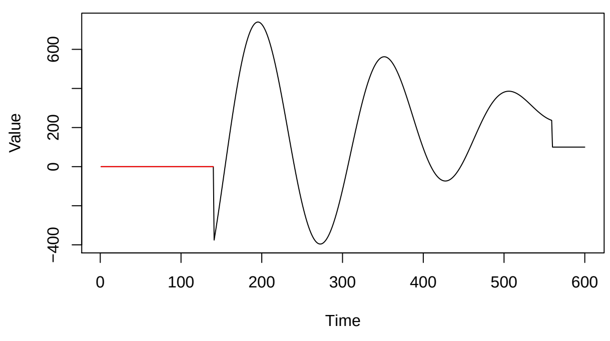


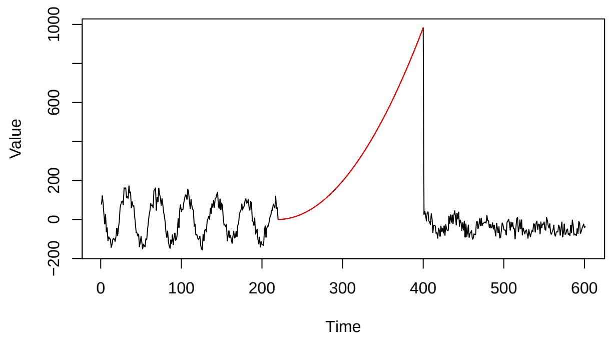
Value
Time

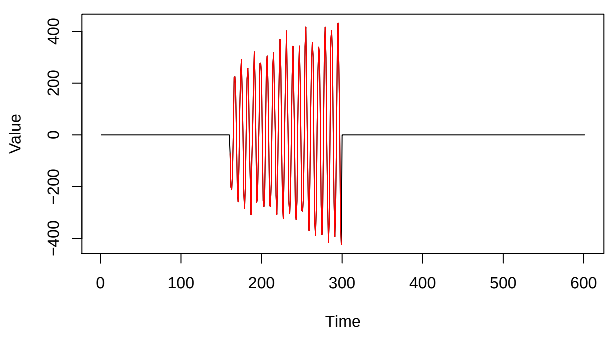

Value
100
50
0
−50
0
20
40
60
80
100
Time

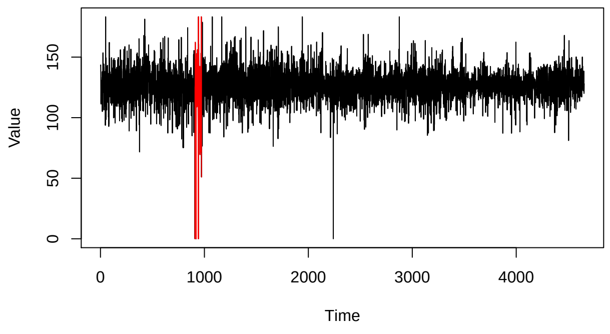

Value
0.5
0.0
−0.5
0
200
400
600
800
1000
Time

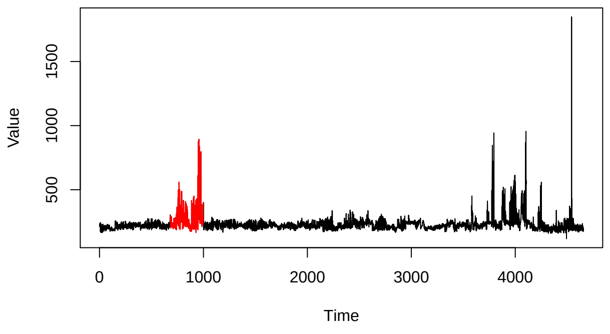

Value
40
20
0
−40
−80
0
100
200
300
400
500
600
Time

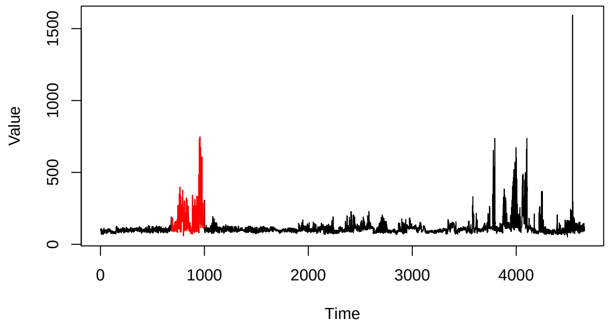

Value
600
200
0
−400
0
100
200
300
400
500
600
Time

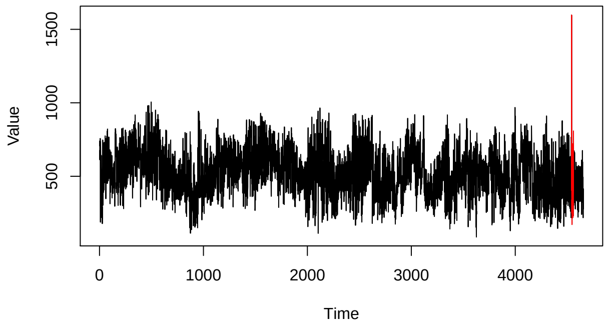

Value
1000
600
200
0
−200
0
100
200
300
400
500
600
Time

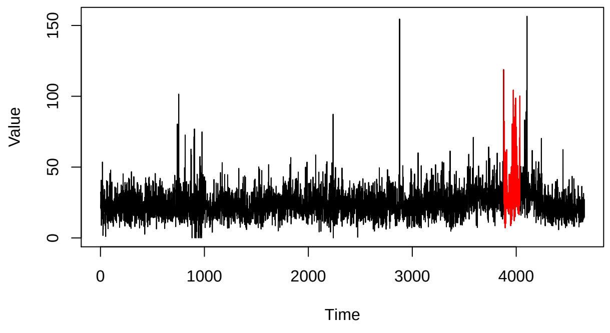

Value
400
200
0
−200
−400
0
100
200
300
400
500
600
Time

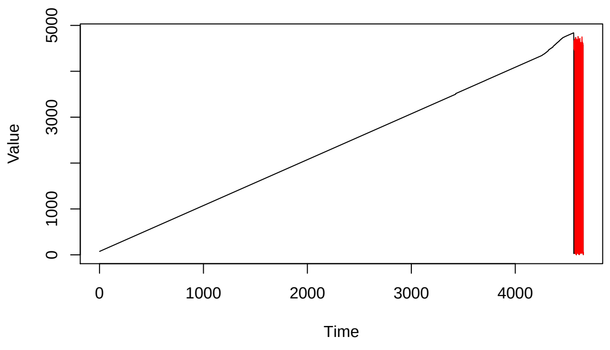

Value
0
50
100
150
0
1000
2000
3000
4000
Time

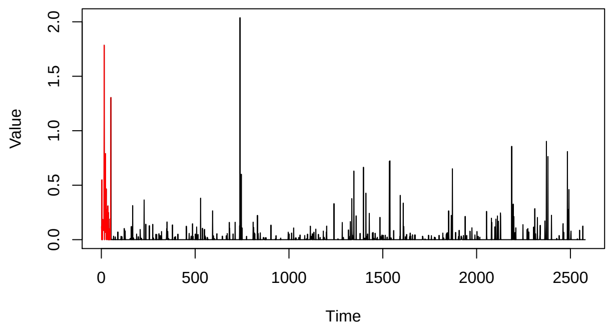

Value
500
1000
1500
0
1000
2000
3000
4000
Time


Value
0
500
1000
1500
0
1000
2000
3000
4000
Time

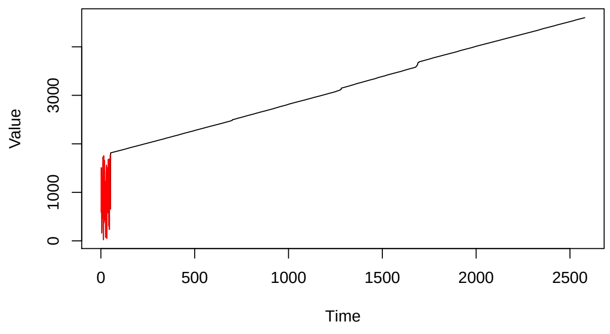

Value
500
1000
1500
0
1000
2000
3000
4000
Time


Value
0
50
100
150
0
1000
2000
3000
4000
Time

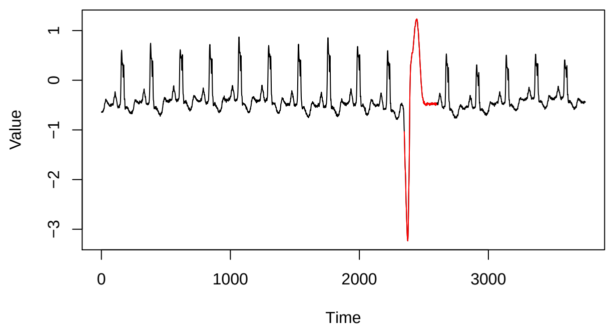

Value
0
1000
3000
5000
0
1000
2000
3000
4000
Time



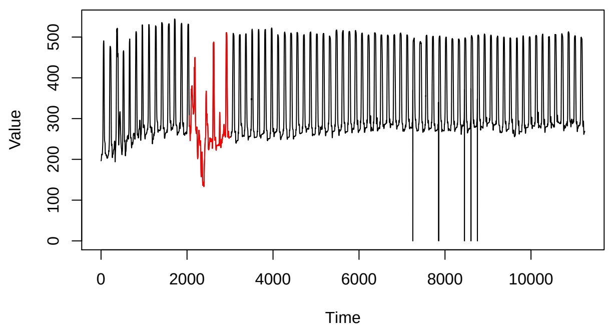
Value
0.0
0.5
1.0
1.5
2.0
0
500
1000
1500
2000
2500
Time



Value
0
1000
3000
0
500
1000
1500
2000
2500
Time



Value
1
0
−1
−2
−3
0
1000
2000
3000
Time



Value
0
100
200
300
400
500
0
2000
4000
6000
8000
10000
Time

Bibliography

[1] Aggarwal, C. C. Outlier analysis. In *Data mining* (2015), Springer, pp. 237–263.

[2] Aris, R. *Mathematical modelling techniques*. Courier Corporation, 2012.

[3] Barnett, V., and Lewis, T. *Outliers in statistical data*. Wiley, 1974.

[4] Brockwell, P. J., and Davis, R. A. *Time series: theory and methods*. Springer Science & Business Media, 2013.

[5] Burnham, K. P., and Anderson, D. R. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media, 2003.

[6] Cellucci, C., Albano, A., Rapp, P., Pittenger, R., and Josiassen, R. Detecting noise in a time series. *Chaos: An Interdisciplinary Journal of Nonlinear Science 7*, 3 (1997), 414–422.

[7] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR) 41*, 3 (2009), 15.

[8] Gershenfeld, N. A., and Gershenfeld, N. *The nature of mathematical modeling*. Cambridge university press, 1999.

[9] Grubbs, F. E. Procedures for detecting outlying observations in samples. *Technometrics 11*, 1 (1969), 1–21.

[10] Hamilton, J. D. *Time series analysis*, vol. 2. Princeton university press Princeton, 1994.

[11] Hawkins, D. M. *Identification of outliers*, vol. 11. Springer, 1980.

[12] Hu, J., Gao, J., and White, K. Estimating measurement noise in a time series by exploiting nonstationarity. *Chaos, Solitons & Fractals 22*, 4 (2004), 807–819.

[13] Hyndman, R. J., and Athanasopoulos, G. *Forecasting: principles and practice*. OTexts, 2014.

[14] Keogh, E., Lin, J., and Fu, A. Hot sax: Efficiently finding the most unusual time series subsequence. In *Data mining, fifth IEEE international conference on* (2005), IEEE, pp. 8–pp.

[15] Larsen, R. J., Marx, M. L., et al. *An introduction to mathematical statistics and its applications*, vol. 2.

[16] Lin, J., Keogh, E., Lonardi, S., and Chiu, B. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery* (2003), ACM, pp. 2–11.

[17] Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. Long short term memory networks for anomaly detection in time series. In *Proceedings* (2015), Presses universitaires de Louvain, p. 89.

[18] Marcus, G. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631* (2018).

[19] Matthews, B. W. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure 405*, 2 (1975), 442–451.

[20] Nikulin, M. Three-sigma rule. `http://www.encyclopediaofmath.org/index.php?title=Three-sigma_rule&oldid=17366`. Accessed:2018-7-21.

[21] Powers, D. M. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

[22] Ron, D., Singer, Y., and Tishby, N. Learning probabilistic automata with variable memory length. In *Proceedings of the seventh annual conference on Computational learning theory* (1994), ACM, pp. 35–46.

[23] Schütze, H., Manning, C. D., and Raghavan, P. *Introduction to information retrieval*, vol. 39. Cambridge University Press, 2008.

[24] Sun, P., Chawla, S., and Arunasalam, B. Mining for outliers in sequential databases. In *Proceedings of the 2006 SIAM International Conference on Data Mining* (2006), SIAM, pp. 94–105.

[25] Taylor, M. P., and Peel, D. A. Nonlinear adjustment, long-run equilibrium and exchange rate fundamentals. *Journal of international money and finance 19*, 1 (2000), 33–53.

[26] Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[27] Yeh, C.-C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., Silva, D. F., Mueen, A., and Keogh, E. Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on* (2016), IEEE, pp. 1317–1322.

[28] Zhu, Y., Zimmerman, Z., Senobari, N. S., Yeh, C.-C. M., Funning, G., Mueen, A., Brisk, P., and Keogh, E. Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on* (2016), IEEE, pp. 739–748.