CHALMERS
UNIVERSITY OF TECHNOLOGY

Anomaly Detection In Time Series Data

a practical implementation for pulp and paper industry

Master's thesis in Engineering Mathematics and Computational Science

Moa Samuelsson

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016

Anomaly Detection In Time Series Data

a practical implementation for pulp and paper industry

MOA SAMUELSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Mathematical Statistics
Chalmers University of Technology
Gothenburg, Sweden 2016

Anomaly Detection In Time Series Data
a practical implementation for pulp and paper industry
Moa Samuelsson

Supervisor: Maria Karlström, EUROCON MOPSsys
Examiner: Holger Rootzén, Mathematical Sciences

Master's Thesis 2016
Department of Mathematical Sciences
Division of Mathematical Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Anomaly detection in time series data
a practical implementation for pulp and paper industry
MOA SAMUELSSON
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Anomaly detection has shown to be a valuable tool in a variety of application domains, e.g. detecting credit card fraud, network intrusion and sensor malfunction.

This thesis provides an anomaly detection algorithm as a monitoring aid applied to time series data from the pulp and paper industry, developed for the company *Eurocon MOPSsys AB*. The algorithm is designed to be generally applicable to the targeted time series by providing methods for adapting parameters to the input data. The anomaly detection algorithm runs in an unsupervised setting using a statistical approach for detection. The algorithm works by fitting a statistical model to a training set of a given size and computing control limits for extracted features of the data. An anomaly is said to be found if a feature falls outside of its limits that are constantly updated to adapt to the current data. The thesis also gives an algorithm that detects changes in the trend of the time series by investigating residuals of linear fits to calculated trends of the data. The time complexities of the algorithms are linear in training size which make them suitable to run in an online environment.

The algorithm was evaluated using time series data provided by MOPSsys consisting of both laboratory and sensor values. As an aid for the evaluation, the time series were inspected visually to manually label deviating patterns. The anomaly detection algorithm is shown to be able to find these deviating patterns. However, it could not be determined whether these patterns are anomalies with respect to the underlying process as no labelled test data was available. Changes in the trend were also found to be in agreement with the beforehand expected outcome.

The developed algorithms show promising results but need labelled test data to give a more accurate evaluation of its performance.

Keywords: online anomaly detection, adaptive statistical process control, time series, segmentation, pulp and pulp industry, unsupervised learning.

Acknowledgements

First and foremost, I would like thank to my examiner Holger Rootzén for his valuable ideas and guidance throughout the project. I would also like to express my gratitude to my supervisor Maria Karlström for her continuous support throughout the project. I would like to acknowledge Ulf Johansson for conceptualising the idea of the project. In addition I would like to thank everybody involved at Eurocon MOPSsys for valuable input and for providing the necessary data.

Moa Samuelsson, Gothenburg, August 2016

Contents

List of Figures

List of Tables

Abbreviations

ARSS Average residual sum of squares

DTPP Difference to previous point

EWMA Exponentially weighted moving average

TrSz Training size

VOS Variance of segment

1
Introduction

Many experts say that we are currently experiencing the fourth industrial revolution, as advances in information and communication technology has allowed for more intelligent treatment of information and interconnectivity between machines and man [1]. There are said to be four design principles of the fourth industrial revolution: interoperability, information transparency, technical assistance and decentralised decisions. This thesis ties in to the third point, technical assistance.

Eurocon MOPSsys is a company that works to facilitate data analysis by supplying information systems for gathering, storing and presenting process information. Their target customers are the pulp and paper industries. Presentation of the stored data gives valuable information to the users about the current state of the production process. This information is used as a support in making decisions regarding the production. In addition, the system is used as an analysis tool for evaluation of the process.

Data collected in the process industry consists of time series with values that originate from sensors or that are manually rendered, for example laboratory values. A time series y is a collection of data where each data point has two entries, a time stamp t and a value $y(t)$ or y_t. Associated to each time series is a sampling frequency, which is the rate that data is gathered. A typical paper mill might have upwards of 10 000 - 20 000 collected and stored time series.

The paper and pulp industry is, as many other process industries, striving to achieve lower production costs, and reducing the number of employees is one possible measure. The result is that each employee will have more areas of responsibility, and thereby reduced possibility to monitor all parts of the process in detail. This induces a demand for better decision support and the ability to let the support systems request the operator's attention when needed. Detecting when a process variable deviates from its normal pattern may prove to be a valuable indicator if this detection can be made with a reasonably low frequency of false positives.

A helpful tool in finding these deviations is to apply anomaly detection. Anomaly detection is the concept of finding deviating patterns in data and is applied in a large variety of domains, e.g detecting credit card fraud, network intrusions and sensor malfunctions. Regardless of application domain, the ability to find anomalies has shown fruitful as anomalies contain valuable information. The definition of

an anomaly may vary with the application domain and sometimes even within the application domain. This makes implementation of anomaly detection techniques challenging. Another issue is that anomalies in general are rare events. This means that creating a data set with labels of normal and abnormal behaviour that is large enough to capture both the anomalies and the normal behaviour is not only difficult but also time consuming.

With this in mind, extracting labelled data of each possible anomaly occurring in the 10 000 - 20 000 collected time series from a paper mill is often not feasible. A more reasonable approach which this thesis explores is extracting the notion of normality from historical values, resulting in an *unsupervised* anomaly detection algorithm. In addition, for the anomaly detection to work as a monitoring aid it needs to be able to detected anomalies of streaming time series, i.e. run in an online environment.

1.1 Aim

This master's thesis aims to design and evaluate an online anomaly detection algorithm for process data from the pulp and paper industry. The algorithm should run in an unsupervised setting and the detected anomalies should be written to a database. To make this algorithm easily applicable in real-life situations, the aim is also to make this algorithm user friendly and time efficient. The thesis also aims to provide a summary and overview of the current state of the art regarding the field of anomaly detection. This last part of the aim is performed as a literature study.

1.2 Scope

This thesis aims to present a first approach to apply anomaly detection methods meant for practical use in pulp and paper production. The scope is to perform a literature study to get an overview of the field and from that choose a suitable method for implementation and evaluation. This thesis is limited to investigate data of univariate time series. Also, the detection techniques should be of the nature such that no previous knowledge of the process is needed. Only historical values of the variable being analysed should be considered. Applying process knowledge would probably improve the results but is outside the scope of this thesis.

1.3 Outline

This thesis is arranged as follows. Chapter 2 gives the theory on which this thesis relies. It also presents the results from literature study of anomaly detection. Chapter 3 concerns the methods for the practical implementation of the proposed algorithm. In chapter 4 the results of the evaluation of the algorithm are presented. Chapter 5 gives a discussion of the results and suggestions on future work. Chapter 6 provides a conclusion and a short summary of the work.

1.4 Contributions

This thesis provides a comprehensive overview of the field of anomaly detection. In addition, a first attempt of developing an anomaly detection algorithm for the pulp and paper industry was performed. The thesis provides an evaluation on how the algorithm is affected by different choices of parameters. This is useful information for further development of the algorithm. At this point the algorithm shows promising results and with further development it may become a useful monitoring aid at the pulp and paper industry. Furthermore, a first attempt on developing an algorithm that detects changes in the trend was performed. This algorithm also shows promising results but could use further evaluation before used in practice.

2
Theory

The purpose of this chapter is to give an overview of the concepts and methods used in this thesis. The chapter begins with an introduction to statistical process control, ideas on which the proposed anomaly detection algorithm is based. The chapter continues with an overview of the field of anomaly detection, which is a result of a literature study and serves as a background for the choices of the proposed method. The chapter ends with methods for time series analysis.

2.1 Statistical process control using Shewhart charts

In this section the basic concepts of statistical process control are introduced. The proposed method for anomaly detection relies on these ideas. Statistical process control is a methodology for monitoring processes to detect deviations. There are a variety of methods used in statistical process control, where this thesis uses so called *Shewhart* charts.

To keep a process in a desired state, monitoring and adjustment of the process is needed. Shewhart control charts is a monitoring tool used to detect deviations in the process. Figure 2.1 shows a typical Shewhart control chart, where μ is the *target value* and $3\sigma, -3\sigma$ are the upper and lower *control limits*. Values collected from a streaming time series make up the data points of the control chart. This data can be raw values of a time series, manipulated values or calculated statistics. New data points are plotted when they become available. When the current point lies between the control limits the process is said to be in a state of control. In contrast, when a data point lies outside a control limit the process is said to be out of control [2].

Even if the process is in a state of control we expect there to be some variation around the target value. This variation is often due to common causes, i.e. noise, in contrast to assignable causes. If assignable causes are detected their impact may be decreased, as they are believed to have an underlying avoidable cause. In the Shewhart control chart, these assignable causes are indicated by sufficiently extreme deviations. To obtain an indication of how large the deviation should be to be considered an indication of an assignable cause a statistical model is applied to model the noise. What kind of model to apply depends on the type of noise present. In this thesis the normal and χ^2 distributions are used to model the noise. Details

of these distributions and how parameters are estimated and their distributions are given in appendix A.

The model of the noise is used to estimate the target value and its standard deviation. From these estimations control limits are defined, shown as $\pm 3\sigma$ in Figure 2.1. In this case a normal distribution was applied to model the noise. If the input data is independently normally distributed the probability of falling outside the control limits defined by $\pm 3\sigma$ is about 1 in 370. This means that if the data is sampled once per day, less than one data point is expected to fall outside of the control limits per year. This frequency can be considered an acceptable rate of false positives, for this reason $\pm 3\sigma$ is a common measure for indication of an assignable cause. By instead choosing other values for the control limits, $\pm k \cdot \sigma$, $k > 0$, it is possible to affect the rate of false positives. However, the choice of k is a trade off between the number of false positives and possibly missing assignable causes.

Detecting assignable causes has three main advantages:

- The underlying cause may be identified and eliminated in the future
- It may be possible to adjust the process to the level where the impact of the cause is as low as possible and thus improve the overall process
- It will be easier to discover other assignable causes as the noise is reduced since the detected assignable cause now is known

Detecting when there are unexplained deviations in the process may thus not only give the possibility for short term corrections of the process but also provide knowledge for improving the process in the long term.

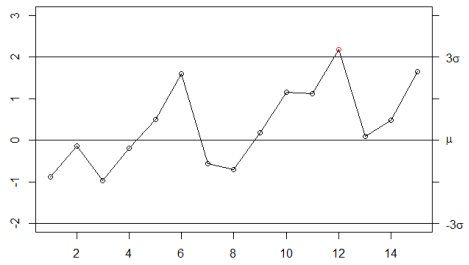

Figure 2.1: Example of a Shewhart chart.

2.2 Anomaly detection

This section gives an overview of the field of anomaly detection. Based on Chandola et al. Anomaly Detection: A survey [3], the section discusses the concept of anomaly and several detection techniques are presented. The techniques are evaluated with focus on their applicability and time complexity.

2.2.1 Anomaly detection setup

Anomalies are defined as patterns in data that do not conform to expected or normal behaviour. The problem of finding such patterns is referred to as anomaly detection. Anomaly detection can be applied to any type of data, binary, discrete or continuous, univariate or multivariate. A set of initial data, which will be referred to as the training set, must be provided. The set of data to be tested if it contains anomalous points is referred to as the detection set. The training and detection sets may be the same. These sets can change over time, e.g. in the case of streaming data. Detecting anomalies in streaming data is the problem of online anomaly detection, while detecting anomalies when the training and detection set is given beforehand is anomaly detection in an offline environment. There is more demand on an online anomaly detection algorithm, for example it needs to run with low time complexity and be able to adapt if new patterns in the data arise.

2.2.2 Types of anomalies

There are mainly three types of anomalies that are studied in the literature, namely point anomalies, sequential anomalies and contextual anomalies. A brief description of those is given below.

- **Point Anomalies** If a single point deviates from the considered normal pattern it is referred to as a point anomaly. This is the simplest form of an anomaly and is the most researched form [3]. An example of a point anomaly is if a process value suddenly is very low or high. An illustration of this is given in Figure 2.2, where the anomaly is marked in red.

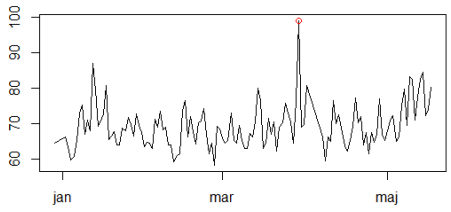

Figure 2.2: The figure illustrates a time series of logged sensor data. It contains a point anomaly, which is marked in red.

- **Sequential Anomalies** If a sequence or collection of points is anomalous with respect to the rest of the data, but not the points themselves, it is referred to as a sequential or collective anomaly [3]. Since this thesis deal with anomalies in time series we will refer to this type of anomaly as a sequential anomaly. An example of a sequence anomaly is if a sensor that records process values fails and from that point outputs the same process value, which is illustrated in Figure 2.3 Note that these values are not considered anomalous themselves, but the sequence of them is.

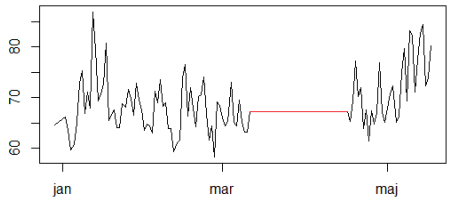

Figure 2.3: The figure illustrates a time series of logged sensor data. It contains a sequential anomaly, which is marked in red. Note that the values of the anomalous sequence are not anomalous themselves but the combination of them is.

- **Contextual Anomalies** If a point or a sequence of points are considered as an anomaly with respect to its local neighbourhood, but not otherwise, it is referred to as a contextual anomaly [3]. For example, suppose a process can target different qualities at different times resulting in changes of the process values for each quality. Let the qualities result in process target values of three different levels 1, 2 and 3. If the process is running a quality at level 2 and there is an instance of a process value close to those of level 3 this is an contextual anomaly, however, globally a process value close to level 3 is not anomalous when running the quality of that level. This is illustrated in Figure 2.4.

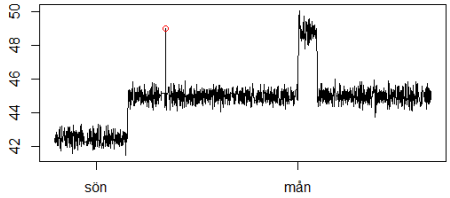

Figure 2.4: Example of a contextual anomaly (marked in red) in a time series.

To summarise, the most straightforward form of an anomaly is point anomaly. It is also the simplest one to detect since there is no need to find what sequence or context it belongs to. When detecting sequential anomalies it is common to transform subsequences of the data to points containing information about the subsequence. Then the techniques for finding point anomalies can be used to find sequential anomalies. However, how to construct subsequences is a problem on its own and can be difficult to solve. Similarly to anomaly detection there are several segmentation techniques that approach this problem. The efficiency of the segmentation technique affects the result of the anomaly detection. Keogh et al. [4] gives a survey of existing segmentation algorithms in time series. To find a contextual anomaly the context must be identified using the contextual attributes of the data. Then an ordinary anomaly detection technique can be applied for each of the contexts. Similarly to

sequence anomalies and constructing subsequences, to identify the contexts can be difficult.

2.2.3 Foundation of anomaly detection

The anomaly detection problem has been investigated in many different fields of mathematics and with different application areas. The anomaly detection techniques have many times been developed to be applied to a specific area of concern resulting in problems when applying the same technique to a different area. Anomaly detection was first research in the field of statistics as outlier detection. Lately the statistical approaches have been expanded by machine learning methods. Hodge and Austin [5] gives an extensive survey of these anomaly detection techniques. Recently other approaches have been explored such as neural networks presented by Markou [6] and methods for cyber-intrusion detection [7].
What is similar for all anomaly detection techniques is that they consists of two parts, a training phase and detection phase. During the training phase, the anomaly detection uses a set of training data to define a model which specifies what is considered normal and/or abnormal with respect to the training set. In the detection phase, new or incoming data is classified using the model from the training phase. Depending on the technique and implementation, the result is either binary or returned as a level of anomaly. A binary result implies that the tested data instance is either reported normal or abnormal, while a result as a level of anomaly is an anomaly score produced from the detection technique. Data instances with anomaly scores above some threshold level could then be classified as anomalies.

2.2.3.1 Learning methods in anomaly detection

The type of data available influences what anomaly techniques that can be applied. There is a main difference in the types of data, labelled or unlabelled data instances. For labelled data there are labels associated with each data point which gives information if the instance is normal or abnormal. For unlabelled data instances there is no such information. From the type of data available there are three different approaches for the training phase:

- **Supervised learning** When applying supervised learning the system is fed with labelled data on which the algorithm defines what is normal or not. The challenge of supervised learning is that it is usually very time consuming to label data and it is normally hard to include all types of anomalies, which is needed for the algorithm to perform well.

  Advantages
  - Could use powerful anomaly detection techniques to learn the underlying model
  - Can be used when anomalies are more frequently occurring than normal instances

  Disadvantages
  - Time consuming and sometimes impossible to label data
  - Hard to find labelled data of all possible normal and abnormal instances

- **Semi-supervised learning** When applying semi-supervised learning the system is solely fed with points of normal behaviour. This gives the system a way to learn what is normal. Similarly to the supervised learning it is hard to find data points that cover every instance of normality.

Advantages	Disadvantages
– More widely applicable than supervised learning – Does not need to specify all types of anomalies that might arise	– Might be challenging to find normal data that covers all normal instances

- **Unsupervised learning** Unsupervised learning does not use labelled data. Instead, this method assumes that the normal behaviour is the most frequently occurring. Normal instances are then defined as the most frequently occurring patterns, and points or sequences deviating from these patterns are reported as anomalies.

Advantages	Disadvantages
– No labelled data needed – Widely applicable	– Relies on the assumption that normal instances are far more frequent than abnormal ones

In conclusion, the learning technique that can be used depends on the available data. If labelled data is available and and it is sufficient to assume that this data represents most of the instances that were given then supervised learning is the most suitable. Semi-supervised learning is used with advantage if it is easy to extract a great variety of normal instances. Unsupervised learning is the only possible alternative if none of the above is applicable.

2.2.3.2 Anomaly detection techniques

This section gives an overview of some of the existing anomaly detection techniques. As part of the literature study, advantages and disadvantages of each approach is given.

- **Classification based** The majority of classification based anomaly detection techniques runs in a supervised or semi-supervised environment. It uses a training set of labelled data to learn a model or classifier. This model is then used to classify new or incoming points. The classification-based techniques rely on the assumption that it is possible to distinguish between normal and anomalous points in the given feature space. Examples of classifiers are Neural Networks [8] [9], Bayesian Networks [10], Support Vector Machines [11] and Rule-based [12] classifiers.

Advantages
- Powerful algorithms can be used to distinguish between instances
- Low time complexity of the detection phase

Disadvantages
- Needs labelled data
- Computationally heavy training phase
- Usually returns binary results, i.e. no level of anomaly is available

- **Clustering based** The clustering-based method groups similar points of the data to form clusters. The clustering-based method can be divided into three categories as they rely on different assumptions. The first clustering-based method relies on the assumption that normal points belong to a cluster and anomalous points do not. Thus it is sufficient to determine if a point to be classified belongs to a cluster or not. [3]
  The second clustering-based method relies on the assumption that normal points lie close their closest cluster centroid while anomalous points do not. In this setup, the centroids of each cluster must be calculated with respect to some measure. The distance to the closest centroid is then the anomaly score for each point. [3]
  The third clustering-based method relies on the assumption that normal data instances belong to clusters that are large and dense while anomalies either belong to clusters that are small or sparse. This technique requires the calculation of the density and size of the clusters. A point is reported as anomalous if the density and/or size of the cluster it belongs to is below some threshold. [3]

  Advantages
  - Runs in an unsupervised environment
  - Low time complexity of the detection phase
  - Widely applicable, can be used on several data types

  Disadvantages
  - Depends on the performance of the clustering algorithms
  - Not optimised for anomaly detection but rather to find clusters
  - Computationally heavy training phase
  - Defining distance measure is not always straightforward

- **Statistical approach** The statistical methods for anomaly detection rely on the assumption that "normal data instances occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model"[3].
  The statistical methods could be divided into two types of techniques, parametric and non-parametric. The parametric techniques assume that the normal data instances are generated from a parametric distribution with parameters Θ. If the parameters are unknown, they are estimated from the training set. The anomaly score of a point x is given by the inverse of the probability density function at $f(x, \Theta)$. Statistical hypothesis test could also be used to classify data instances.

On the other hand, non-parametric methods do not define the underlying model a priori, but finds the structure from the data. The simplest non-parametric technique uses histograms. A histogram is made from the training data set and the anomaly score of a point is the inverse of the height or the number of other points in the bin it falls into. This method is sensitive to the choice of the bin length. [3]

Advantages
- Runs in an unsupervised environment
- If the statistical assumption is true, this technique provides a statistically justified solution
- Low time complexity of both the training and detection phase

Disadvantages
- Relies on the assumption that data comes from the assumed statistical distribution
- Hard to determine the correct test statistic to use
- Multivariate anomalies might not be detected

The most suitable anomaly detection technique to use depends on what training data is available and also what restrictions there are on the time complexity. The proposed techniques usually need some modifications when applied in a real life situation.

2.3 Time series analysis

This section gives techniques for time series analysis. It focuses on the problem of processing and analysing data to gain useful information. The methods presented are trend estimation, segmentation, aggregation and extraction of features. For each of these methods their time complexities are given. The Big O notation is used to analyse the time complexity, which is explained in appendix B.

2.3.1 Definition of a time series

A time series is a collection of data consisting of time stamped entries ordered in time. The time stamp is denoted t and the value of the time series at that point in time is denoted $y(t)$ or y_t. The rate at which data is collected in a time series is referred to as the sampling frequency. The sampling frequency is usually given as the average number of samples per second.

2.3.2 Trend estimation

To analyse long term changes in a time series it can be useful to calculate the trend of the time series. The computed trend is itself a time series that explains underlying tendencies and can be viewed as a smoothed version of the original time series. There are multiple ways of computing the trend, this section presents the methods *moving average*, *moving median* and *exponentially weighted moving average*.

- **Moving average** One of the most intuitive ways of computing the trend is to use moving average. Utilising this method the trend component, $\tilde{y}(t)$, at

each point in time is the average of the n previous points. Formally, let y be a time series of process values and n be the number of previous points to use. The trend component or moving average at time t is given by

$$\tilde{y}(t) = \text{MA}(t) = \frac{y_t + y_{t-1} + ... + y_{t-n}}{n} = \frac{1}{n}\sum_{i=t}^{t-n} y_i$$

Advantages	**Disadvantages**
– Easy to understand	– Not trivial to choose n to get a good result
– Time complexity of $O(1)$	– Sensitive to outliers

- **Moving median** An other intuitive way to calculate the trend is to use moving median. This is analogous to the moving average method, but the average is exchanged for the median. Formally, let y be a time series of process values and n be a fixed time frame. Then the trend component or moving median at time t is given by the median of the n previous points,

$$\tilde{y}(t) = \text{MM}(t) = \text{median}(y_t, y_{t-1}, ..., y_{t-n}).$$

Advantages	**Disadvantages**
– Easy to understand	– Not trivial to choose n to get a good result
– Robust against outliers	– Time complexity of $O(n \log n)$

- **Exponentially weighted moving average** Another way of calculating a trend is to use an exponentially weighted moving average. This method is related to the moving average, but uses a smoothing coefficient θ instead of a number of previous points n. The exponentially moving average is the average of all historical points but the influence of the historical points decay exponentially with time. Let y_t be the value of the time series at time t and $\theta \in [0, 1]$ be the smoothing constant. The exponentially weighted average is given by

$$\tilde{y}_t = \text{EWMA}(t) = (1-\theta)(y_t + \theta y_{t-1} + \theta^2 y_{t-2} + ...).$$

The weights sum up to one since the geometric series

$$1 + \theta + \theta^2 + \theta^3 + ... = \frac{1}{1-\theta}$$

when $\theta \in [0, 1]$, much like the n weights of $1/n$ of the moving average sum up to one. The smoothed value can be calculated recursively as

$$\tilde{y}_t = (1-\theta)y_t + \theta\tilde{y}_{t-1},$$

since

$$\begin{aligned}
\tilde{y}_0 &= y_0 \\
\tilde{y}_1 &= (1-\theta)(y_1 + \theta \cdot \overbrace{y_0}^{\tilde{y}_0}) \\
\tilde{y}_2 &= (1-\theta)(y_2 + \theta(y_1 + \theta y_0)) = (1-\theta)y_2 + \theta\overbrace{(1-\theta)(y_1 + \theta y_0)}^{\tilde{y}_1} \\
&\vdots \\
\tilde{y}_t &= (1-\theta)(y_t + \theta y_{t-1} + \theta^2 y_{t-2} + ... + \theta^{t+1} y_0) = ... \\
... &= (1-\theta)y_t + \theta\overbrace{(1-\theta)(y_{t-1} + \theta y_{t-2} + ... + \theta^t y_0)}^{\tilde{y}_{t-1}}
\end{aligned}$$

This makes it possible to compute the EWMA in $O(1)$ as long as the EWMA of the previous point in time is known. A θ close to 0 will give the most recent value greater influence on the value of the exponentially weighted moving average while a θ close to 1 will give the historical values the most influence. When calculating a trend, θ should preferably be close to 1.

Advantages
- Adapts to the recent behaviour of the process
- Time complexity of $O(1)$

Disadvantages
- Conceptually complicated in comparison to the moving average and the moving median
- Depends on the choice of θ

2.3.3 Segmentation of time series

Segmentation of time series is a useful tool for detection of sequential anomalies as mentioned in section 2.2.2. The purpose of a segmentation algorithm is to partition the time series y in smaller subseries $< y^1, y^2, ..., y^k >$ where y^i is a subsequence of y such that

$$y = \bigcup_{i=1}^{k} y^i$$

and

$$y^i \bigcap y^j = \emptyset, \quad i \neq j.$$

The resulting subsequences should consist of similar values with respect to some measure. This section gives an overview of some of the existing segmentation methods. We will focus on partitioning the time series into piecewise linear segments, but the methods are applicable for other attributes as well, e.g. partitioning into subsequences with similar variance.

The segmentation can either utilise *linear interpolation* or *linear regression* as a tool to measure similarities in the subsequences. Using linear interpolation the fitted lines are drawn between the initial and end point of the segment. While using linear regression the line is fitted in order to minimise the sum of the squared residuals. A residual is the vertical distance from a point to the fitted line.

The segmentation methods use a maximum error bound to decide when to create a new segment. For example, an error measure could be the sum of squared residuals. Another common error measure is the largest distance for a point in the segment to the line. These or some other error measure can be used to determine the error of the segment. An ideal segmentation has small errors and a small number of segments. The result of the segmentation depends on the choice of threshold level for the error. A too small value of the threshold level may result in a large number of segments that might not provide much additional information to the original time series. In contrast, a too large value of the threshold level may result in few large segments which can mask critical differences.
There are a multitude of different segmentation methods with different advantages and disadvantages. In the list below a few of them are described in more detail.

- **Sliding window segmentation algorithm** The sliding window method builds up segments from an initial point t_0 and then proceeds forward in time. A new segment is created when the previous segment exceeds some predefined error limit. Let $y(t), t \in [t_0, T]$ be a time series to segment. The first segment starts at t_0 and consecutive points in time are added to the segment until the error of the segment exceeds some predefined threshold level. A new segment is created from the end point of the previous segment and onwards. This continues until the entire time series has been segmented.

  Advantages
  - Can run in an online environment
  - Time complexity of $O(Ln)$, where n is the length of the time series and L is the average length of a segment

  Disadvantages
  - Usually produces a segmentation far from optimal

- **Top down segmentation algorithm** The top down method differs from the sliding window method by that the whole sequence is considered at once. All possible partitions are evaluated and then the time series is split at the best location, i.e. where the error decrease is the largest. These new segments are then tested to see if their approximation error is below the predefined threshold, if not the algorithm recursively continues to split the segments into sub-segments until all segments fulfill the threshold level.

  Advantages
  - Produces a solution with low total residual error

  Disadvantages
  - Time complexity of $O(n^2K)$, where n is the length of the time series and K is the number of segments
  - Is not suitable to run in an online environment

- **Bottom up segmentation algorithm** The bottom-up method is opposite to top-down in the sense that it starts with the finest partition and builds up the solution by fusing these. In more detail, if n is the length of the time series, at first $n/2$ segments are created. These segments are then fused with one of

their adjacent segments in a way such that the increase in error is the smallest. This procedure continues until some stopping criteria is fulfilled. Such criteria could be a maximum number of segments, a maximum increase in error when segments are fused and/or a total maximum error of the segments.

Advantages
- Time complexity of $O(Ln)$ where n is the length of the time series and L is the average length of a segment
- Produces a solution with low total residual error

Disadvantages
- Is not suitable to run an online environment

2.3.4 Aggregation

A major topic of time series analysis is aggregation. Aggregation is important when mining data and extracting information, but also for presentation of the data. Time series often consist of a very large amount of data points which may be troublesome to handle. Aggregation methods provide a way to compress a time series by replacing consecutive data points by some representative value, e.g. their mean or median value. Aggregation is beneficial since it reduces the number of points and clarifies patterns in the time series by suppressing noise. In this section the method *piecewise aggregate approximation* is presented together with a proposed method for adaptive choice of the aggregation parameter n, i.e. the number of points to use for aggregation.

Piecewise aggregate approximation

One of the simplest methods for aggregation, proposed by Keogh et al., is piecewise aggregate approximation [13]. The time series is divided into equally sized subsequences over which the mean is taken. This reduces the length of the time series with a factor n that is the length of the subsequences, see Figure 2.5. This method both has low time complexity and can be run in an online environment. The choice of the length n of the subsequences is a trade off between number of reduced points and risk of masking critical patterns. A small value of n may leave an unnecessary amount of points and noise while a large value of n may be a too rough approximation and mask significant patterns. A suitable choice of n can differ a lot between time series, sometimes by several powers of 10. This makes it hard, and sometimes impossible, to determine a fixed value of n that gives an adequate result to different time series. This gives rise to the need of finding a method that in an adaptive way sets a suitable choice of n.

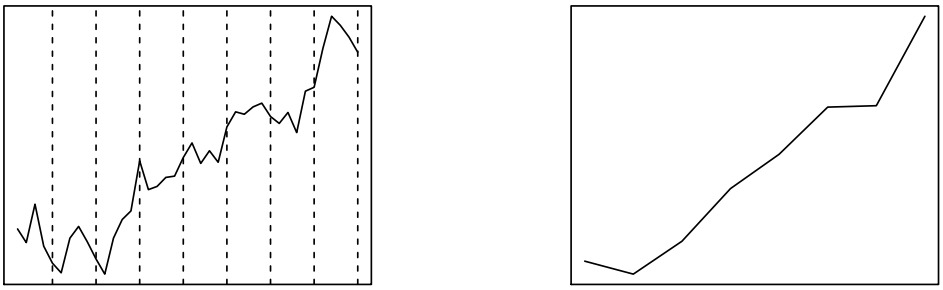

Figure 2.5: Time series aggregation using piecewise aggregate approximation. The time series to the left is aggregated by taking the mean of the points between the vertical lines. The result is displayed to the right.

Determining n adaptively using the autocorrelation function

This is a proposed method to approach the problem of finding an adaptive way to determine the time frame n used for aggregation. A key point for a good aggregation is to find few points that efficiently describe the pattern of the time series. For example, consecutive points that follow a linear pattern can efficiently be aggregated by the end points of a fitted line of this section, as illustrated in Figure 2.6.

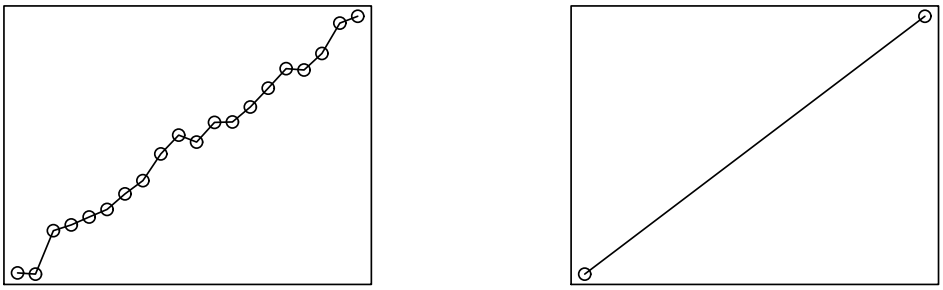

Figure 2.6: Time series aggregation of points that follows a linear pattern. The time series to the left is aggregated by taking the end points of the fitted line of this section. The result is displayed to the right

The proposed method seeks to find an average length of linear relationships in the time series by utilising the *autocorrelation function*. Such an average length of linear relationships is believed to be a good measure for the aggregation length. The autocorrelation of a time series measures the degree of linear relation between points that are some distance or lag τ apart. [14] The autocorrelation function with lag τ is given by

$$\rho_\tau = \frac{\mathrm{Cov}(y_t, y_{t+\tau})}{\mathrm{Var}(y_t)}.$$

For a time series that originates from a continuous process for which the trend is eliminated, the absolute values of the autocorrelation tend to be close to 1 for adjacent points in time and decay to zero as the time lag increases. To find a suitable choice for n the autocorrelation is calculated for r subsequences of the time series of length l with increasing values of τ until ρ_τ drops below some predefined level α. The value of τ is registered at this point and an average of them is taken as the value of n.

Advantages

- Adapts to the data
- Time complexity of $O(n)$

Disadvantages

- Relies on the assumption that the autocorrelation will drop and that this measure is a suitable choice for n
- Depends on the length of the subsequences which is not trivial to choose

2.3.5 Extraction of features

A time series consists, in its raw state, of a collection of time stamps and values associated with these. To detect changes in amplitude or shape of the time series it is necessary to extract information about the patterns of the time series. This section defines some of the possible *features* of a time series used to quantify and detect these patterns. The described features will be considered both using the original time series and when the trend of the time series is eliminated.

- **Raw data** The most trivial feature is the raw data values of the time series. This feature can be used to detect global extreme values of the time series, e.g. as shown in Figure 2.2. In the case when the trend is eliminated, this feature will explain how the time series varies around the trend. This makes it possible to find local extreme values as well, e.g. as shown in Figure 2.4.
- **Difference to previous point** Using this feature as a measure it is possible to detect extreme fluctuations. It is given by

  $$\mathrm{DTPP}(t) = y_t - y_{t-1},$$

  where y_t is the process value at time t. This feature is used to detect extreme fluctuations between consecutive points. Singular extreme values, e.g. as shown in figures 2.2 and 2.4, may also be detected by this feature.
- **Variance of subsequences** This feature can detect if the variance of certain sections of the time series changes. A change in variance for an adjusted process could indicate that the regulator does not work properly. This feature requires a segmented time series. The variance of each subsequence is given by

  $$\mathrm{Var}(X) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{X})^2.$$

  This feature is used to detect changes in the variance, e.g. as shown in Figure 2.3.

3

Method

This chapter outlines the practical methods used for anomaly detection in time series data and their implementation in the proposed algorithms. The chapter begins with a presentation of the proposed anomaly detection algorithm together with motivations for choices of methods. Furthermore, the proposed method for detecting changes in the trend is presented. The chapter ends with an outline of the practical implementation considerations.

3.1 Proposed method for anomaly detection in time series

This section presents the proposed anomaly detection algorithm. The algorithm consists of three main steps. At first the data is preprocessed to extract information and features of the time series data. Afterwards the preprocessed data is analysed to find if any of the extracted features lie outside any control limit. In that case an anomaly is reported. Lastly the control limits are updated.

To find different aspects of the analysed data, the algorithm is run on both the original data and on trend eliminated data. The algorithm run on the original data is referred to as algorithm a and is presented in the flowchart in Figure 3.1. The algorithm run on the trend eliminated data is referred to as algorithm b, presented in the flowchart in Figure 3.2. For each step in the flowchart there is a reference to a section that presents the details of the step.

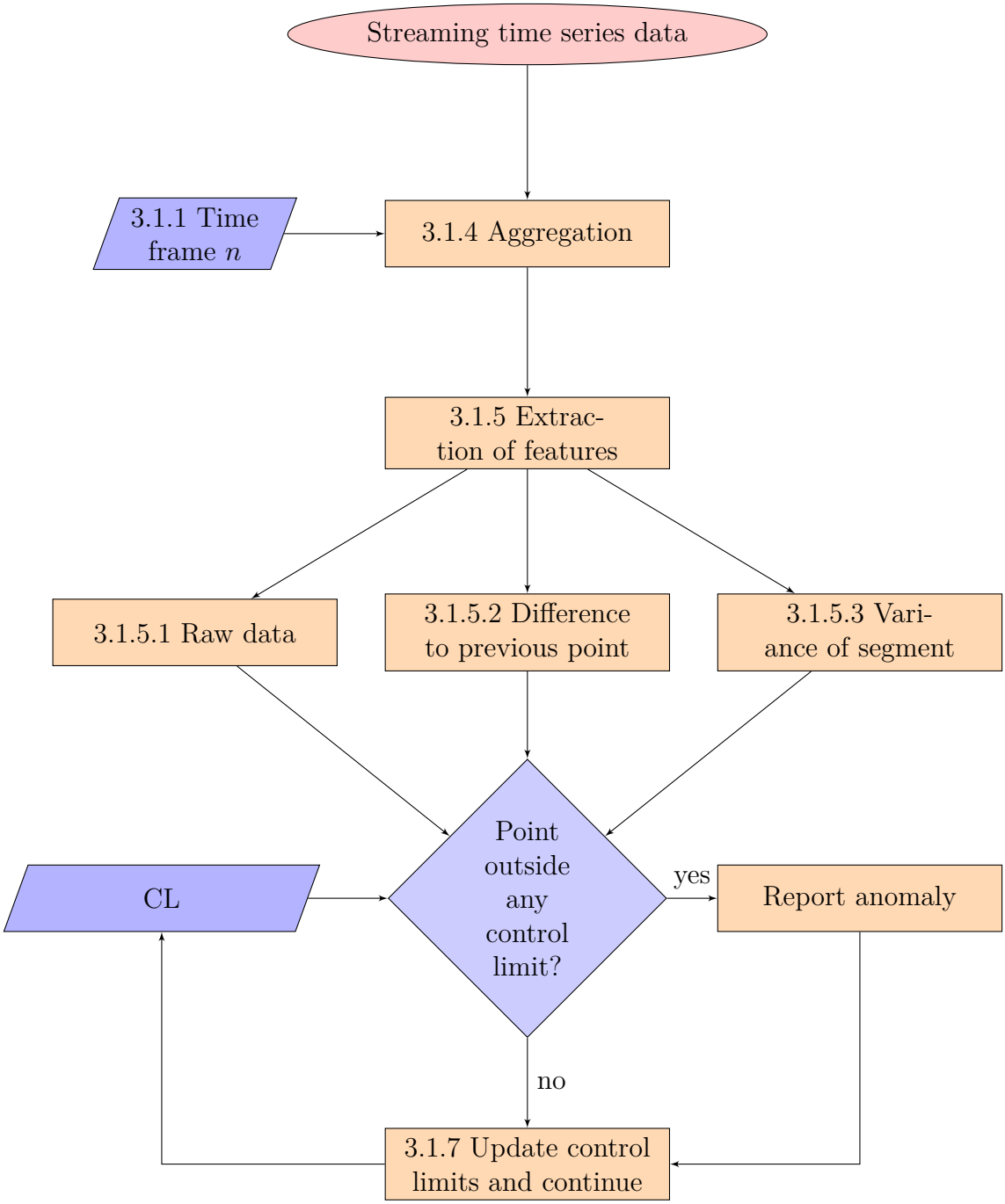

Figure 3.1: A flowchart of the proposed anomaly detection algorithm a.

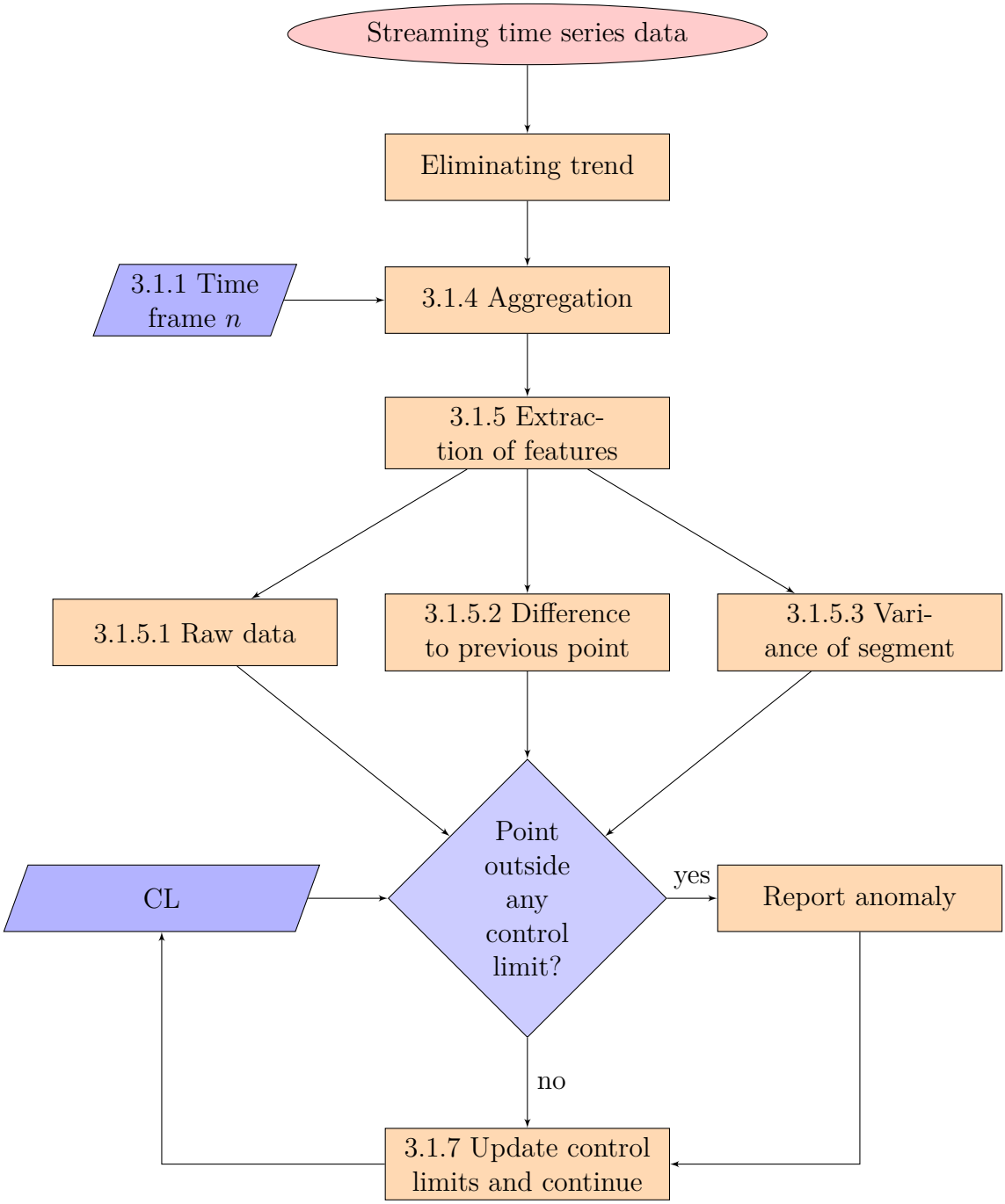

Figure 3.2: A flowchart of the proposed anomaly detection algorithm b.

3.1.1 Calculating the time frame

Time frame n used for aggregation is deduced in an offline step in the beginning of the algorithm. The reason for not constantly update this parameter is due to two main reasons:

- **Reduce number of operations of the algorithm** This parameter is believed to have a small variance, i.e. it will lie approximately around the same value. Thus constantly performing these calculations will not have a large

impact on the accuracy
- **Increase user friendliness** Frequent changes of time frame for aggregation may be confusing for the users

An outline of the algorithm is shown in the flowchart in figure 3.3. For each step of the algorithm there is a reference to the section that presents the step in detail.

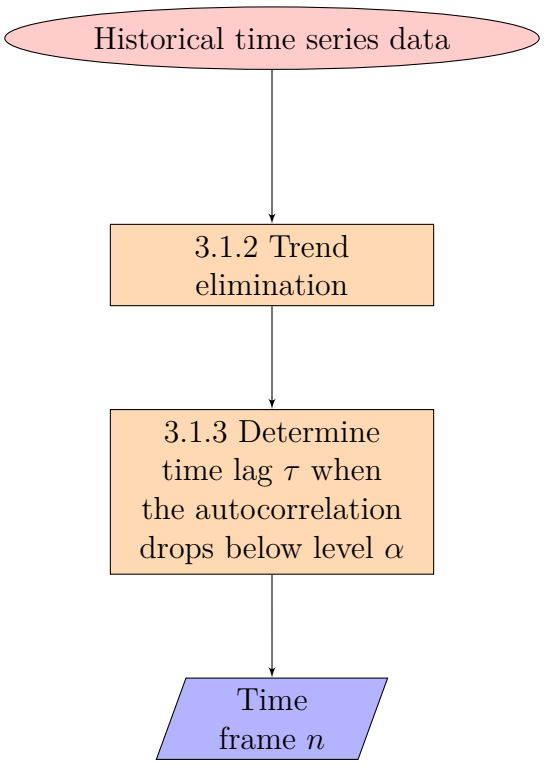

Figure 3.3: A flowchart of the proposed algorithm to determine the time frame n for aggregation.

3.1.2 Trend elimination

Trend elimination is performed as the first step for deducing time frame n for aggregation and for the anomaly detection algorithm b. The trend is eliminated by calculating a trend using exponentially weighted moving average, 'EWMA', explained in section 2.3.2 with $\theta = 0.8$. The 'EWMA' method for computing the trend is used due to two reasons. It has low time complexity and choosing a value of the smoothing coefficient θ giving a reasonable result for a variety of time series is straightforward in comparison of finding a suitable number of points n used for moving average or moving median. The computed trend is subtracted from the original time series. Pseudocode for eliminating the trend is found in the listing 3.1.

Listing 3.1: Pseudocode for eliminating trend of a time series

```
Input: Streaming time series TS, trend parameter theta
Output: Time series with trend eliminated eTS

trend_elimination(TS, theta)

    s[t] <- (1-theta)*TS[t] + theta*s[t-1] # s(t) is the trend
    eTS <- TS[t] - s[t]

    return(eTS)
```

3.1.3 Determine time lag τ when autocorrelation drops below level α

To find the time frame n used for aggregation, repeated autocorrelation calculations of subsequences of the time series is performed. The subsequences are determined by randomly selecting r points from the time series. These points are used as initial points for subsequences of length l which are extracted from the time series. The autocorrelation is calculated for these subsequences with increasing time lag τ. When the correlation drops and is below some predefined level α for five consecutive points the value of τ is registered. Finally the median is taken of the registered values of τ. Pseudocode is found in the listing 3.2.

Listing 3.2: Pseudocode for calculating n using autocorrelation function

```
Input: Time series with trend eliminated eTS, number of repetitions r, subsequence length l, alpha
Output: Time frame n

determine_n(eTS, r, l, alpha)
    randomly select r points from eTS
    create subsequences of length l from the randomly selected points
    for i in 1:r
        autocorrelation <- inf
        tau <- 1
        while(absolute value of autocorrelation > alpha)
            calculate autocorrelation for subsequence i and lag tau
            tau <- tau + 1
        endWhile
        register tau
    endFor

    return median of registered tau
```

The median of the different values of τ is translated to the nearest time frame, that is translating the numerical value median(τ) into a unit of a time period. For example, if the sampling frequency of the time series is 1/day and the median of the values of τ is 6. Then the numerical value 6 may be translated into the time period of 1 week.

3.1.4 Aggregation

A step in preprocessing the data is aggregating the time series. This reduces noise and clarifies patterns. The aggregation is performed using piecewise aggregate approximation, which is described in section 2.3.4. This method uses parameter n, which is the time frame to aggregate over. This parameter is set using the proposed method that utilises the autocorrelation of the time series. This method is explained in section 3.1.1. Pseudocode for the aggregation is shown in the listing 3.3. The aggregated time series is the data used from now on for the anomaly detection.

Listing 3.3: Pseudocode for aggregation using piecewise aggregate approximation

```
Input: Streaming time series TS, length of subsequence n
Output: Aggregated time series ATS

piecewise_aggregate_approximation(TS, n)
    t_0 <- time tick since last aggregated point
    t <- current time tick
    If t-t_0 >= n
        ATS <- mean(TS[t_0:t])
        t_0 <- t

    return ATS
```

3.1.5 Extraction of features

The second step of the algorithm is to extract features of the data. Three features are extracted, namely raw data, difference to previous point and variance of segment, explained in section 2.3.5. Details of the implementation is given in this section.

3.1.5.1 Raw data

Extracting this feature is trivial, as this feature consists of the raw input values. The raw data is assumed to be independently normally distributed $N(\mu, \sigma^2)$, where μ is the target values and σ describes the amount of noise. However, the independence assumption is unlikely to hold true for values close in time but for simplicity independence is assumed.

3.1.5.2 Difference to previous point

The next feature is difference to previous point that captures the behaviour of fluctuation of the time series. The values of this feature are assumed to be independently normally distributed $N(0, 2\sigma^2 - 2\text{Cov}(y_t, y_{t-1}))$. Note that this feature eliminates linear trends and thus should be symmetrically centred around 0 if the normality assumption holds true. Pseudocode for the implementation of the extraction of this feature is given in the listing 3.4.

Listing 3.4: Pseudocode for extracting feature: difference to previous point

```
Input: Streaming time series TS
Output: Difference to previous point DPP

difference_to_previous_point(TS)

    t <- current time tick
    DPP <- TS[t] - TS[t-1]

    return DPP
```

3.1.5.3 Variance of segment

The third feature that is extracted is the variance of segments. The time series is divided into segments in a fashion of sliding windows with window size $n = 10$ and jump size $l = 5$. Since the raw data is assumed to be normally distributed and the variance includes squaring these numbers, the values of this feature are assumed to be chi-squared distributed $\chi^2(n-1)$, where n is the window size. Pseudocode for the implementation of the extraction of this feature is given in the listing 3.5.

Listing 3.5: Pseudocode for extracting feature: variance of segment

```
Input: Streaming time series TS, window size/segmentation length n, hop length l
Output: Variance of segment VOS

variance_of_segment(TS, n, l)
    t0 <- last time tick variance was calculated
    t <- current time tick
    if(t-t0 >= l)
        VOS <- variance(TS[(t-n:t])
    endIf
    return VOS
```

3.1.6 Detecting anomalies

The next step of the algorithm is to decide if the streaming points are anomalous or not with respect to the most recent data. This is simply done by checking if the most recent points of the extracted features falls outside the control limits determined in the last iteration. How the control limits are calculated is explained in section 3.1.7. If a point falls outside of the control limits it is reported anomalous.

3.1.7 Calculation of control limits

The last step of the algorithm is to update the control limits. As described in section 2.1 the control limits are given by $\pm k\sigma$. The control limits are dynamically determined by estimating σ for each streaming point from the data using the n most recent points and a given value of k. Pseudocode for calculating the control limits is given in the listing 3.6. The value of k is set so that an accepted rate of expected false positives is reached.

Listing 3.6: Pseudocode for calculating the control limits

```
Input: Streaming data of extracted features EF, training size n, constant for control limit k
Output: Control Limits CL

calculation_of_control_limits(EF,n, k)

    t <- current time tick
    sigma_hat <- standard_deviation(EF[t-n:t])
    CL <- concatenate(-k*sigma_hat, k*sigma_hat)

    return CL
```

3.2 Detect changes in the trend

An important aspect to consider for many time series from the paper and pulp production is changes in the trend. A change in trend can indicate a change in the process that may affect the quality negatively, however a change in trend may also define a normal behaviour, e.g. a shift in target value. This section proposes a method for detecting changes in the trend.

The trend detection consists of three main steps. The first step is to calculate a trend. The second step is to linearly segment the trend. Finally, a change in trend is reported if a new segment is created, i.e there are more than one segment after the segmentation is completed. This method is based on the assumption that if the trend does not change, i.e. the slope does not change, a line can be fitted with streaming points of the trend without a significant increase in sum of residuals per points. Contrarily, if the trend changes, there will be an increase in sum of residuals per points with the result that a new segment is created and a change in trend is reported. An example of the algorithm is illustrated in figure 3.4

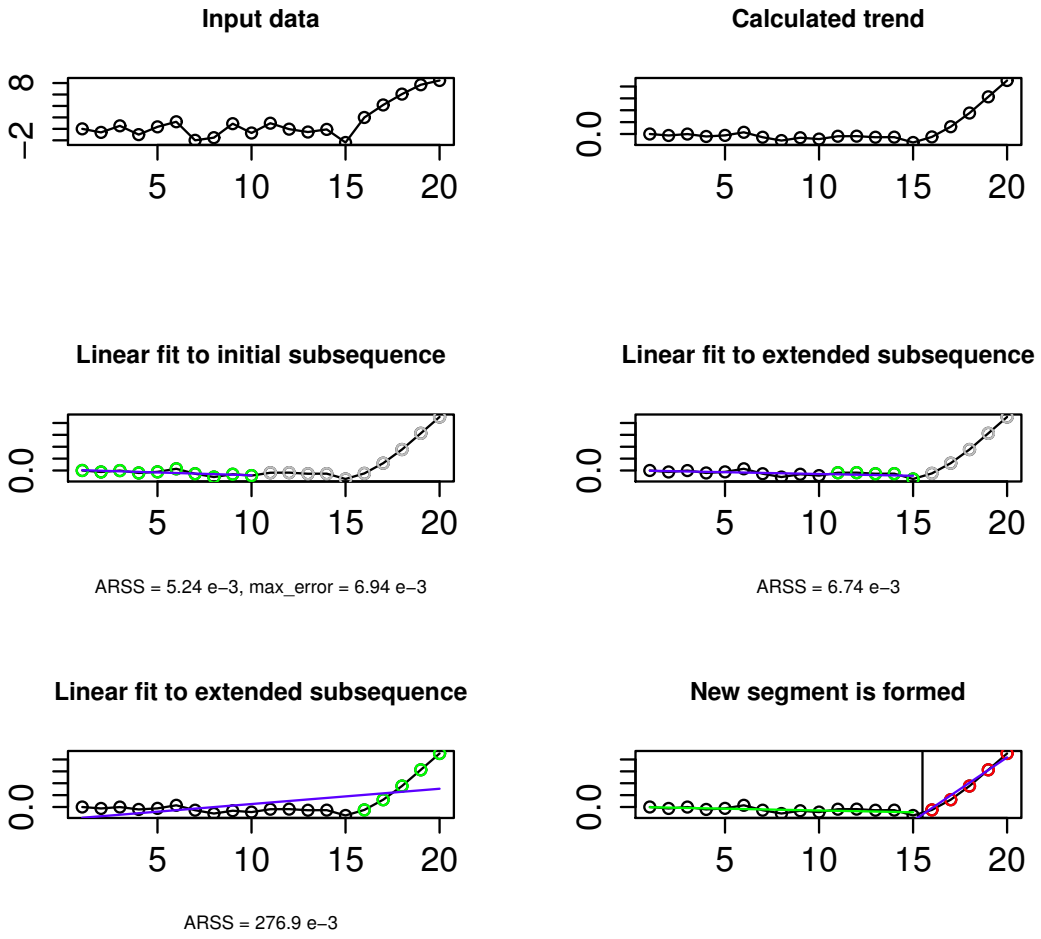

Figure 3.4: Illustrated example of the proposed algorithm for trend change detection. The top left figure shows the input data and the top right shows the calculated trend for this data using exponentially weighted moving average with $\theta = 0.9$. The middle left figure shows the linear fit (blue line) to the initial subsequence. The middle right figure shows the linear fit to the extended subsequence. The added points are marked in green. The bottom left figure shows the linear fit to the further extended subsequence. Since $ARRS > max\_error$ a change in trend is reported an a new segment is formed, shown in the bottom right figure.

A flowchart of the algorithm is shown in Figure 3.5, for each step there is a reference to the section that gives details of the practical implementation.

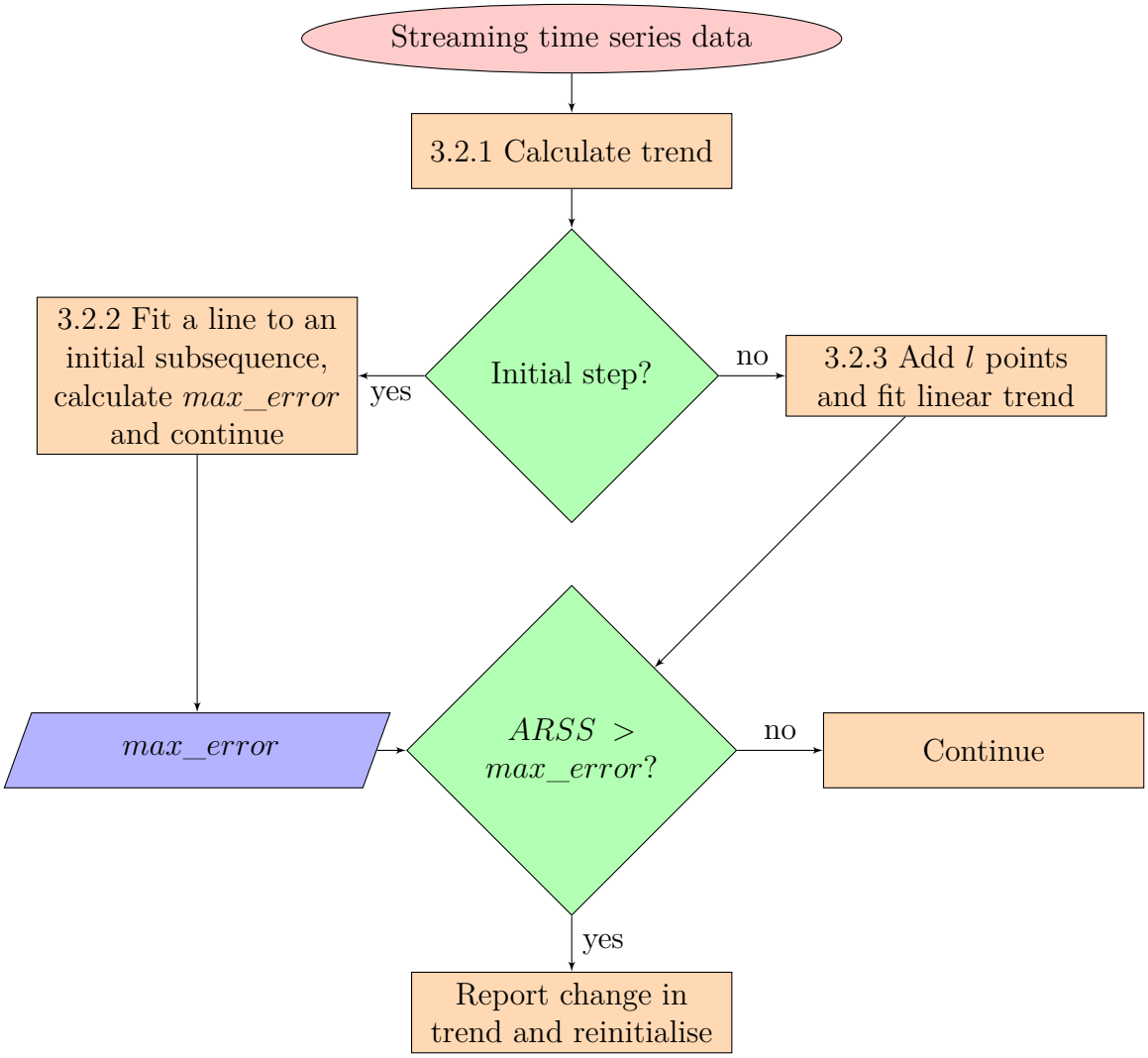

Figure 3.5: A flowchart of the proposed method for detecting changes in the trend.

3.2.1 Trend calculation

The first step of the algorithm to detect changes in trend is to calculate a trend. This is done using exponentially weighted moving average as described in section 2.3.2 with $\theta = 0.9$. For this method it is convenient to use a larger value of θ to get a smoother trend.

3.2.2 The initial step

The purpose of the initial step of the segmentation is to determine a threshold, max_error, which will be the decision boundary for when a new segment should be produced. A line is fitted in the least square sense to the initial subsequence of the trend with length n. The threshold is calculated by

$$max\_error = \alpha \cdot \frac{\sum_{i=1}^{n} r_i^2}{n}$$

where r_i is the residual at point i and α is a constant ≥ 1 to tune the sensitivity of the detection. That is, α allows for minor deviations in the average residual sum of squares ($ARSS$) of future fittings without causing a report of change in trend. In comparison with the anomaly detection algorithm α is similar to k. A small value of α may falsely report changes in the trend while a large value of α may influence the algorithm to detect changes in the trend later in time.

3.2.3 The subsequent steps

The trend is segmented by adding l consecutive points to the initial subsequence of the trend. A new line is fitted to this extended subsequence. The average residual sum of squares is calculated as

$$ARSS = \frac{\sum_{i=1}^{n} r_i^2}{n}.$$

If $ARSS$ of this new segment is greater that max_error the recently added points are rejected to be part of the current segment. A change in the trend is reported and a new initial segment is created. If the trend does not change the extended subsequence will eventually be very large and a trend change may then be masked. Therefore does the algorithm use a maximum length of the subsequence such that when the extended subsequence reaches the maximum length a sliding window approach is applied.

3.3 Practical implementation procedure

This section gives the details of the practical implementation, such as software used and data format.

Software language R The algorithm and the evaluation of it is implemented in R and the library 'sats' is used. R is used due to its powerful tools for statistics and data analysis. It is also convenient since it allows for creating objects and functions. These features make R ideal for prototyping the algorithm. For increased performance in future implementations, R has the ability to integrate with other programming languages such as C, C++, Java and Python.

Data format The data covers 14 different time series collected at a pulp and paper mill, which are stored on a hard drive. The time series consists of time stamps, values at the time stamps and status codes. Values of status codes other than 'ok' are removed. Streaming data is simulated using a `for`-loop.

4

Results

This chapter presents the results of the thesis. The chapter begins with an explanation of how time series was selected used to evaluating the algorithm. Afterwards, results of the impact of various parameters on the anomaly detection algorithm is presented. This is followed by results from the algorithm that detects changes in the trend. The chapter ends with an outline of the time complexities of the developed algorithms.

4.1 Selection of data

The fourteen provided time series were inspected visually. Three of these time series were selected as representatives for analysing the algorithm as they capture a variety of potential anomalies and behaviours that the algorithm should be able to adapt to. Furthermore, subsequences of these time series were selected that captures critical patterns and contain a reasonable low amount of data points. For one of the time series, two subsequences were selected. The time series are plotted in figures 4.1-4.3. In the anomaly detection algorithm, each of the time series are considered in their original state, then referred to as time series x.a, and with their trend eliminated, then referred to as time series x.b.

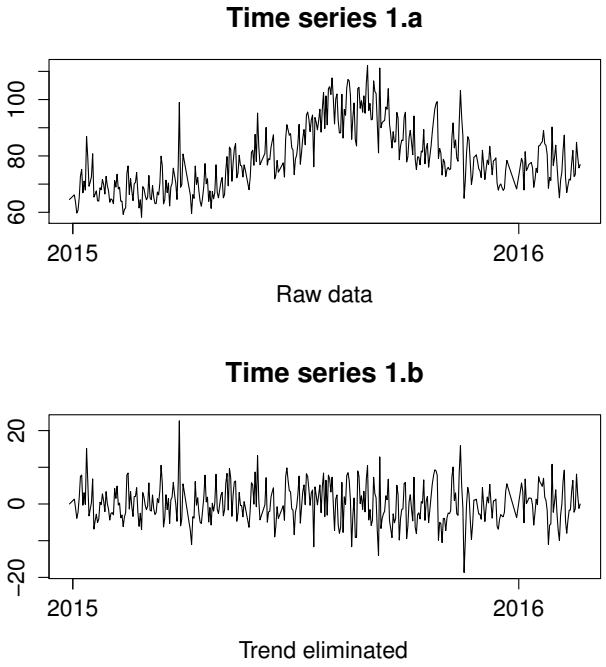

Figure 4.1: Time series 1.a and 1.b. This time series contains a seasonal trend in its original state, time series 1.a.

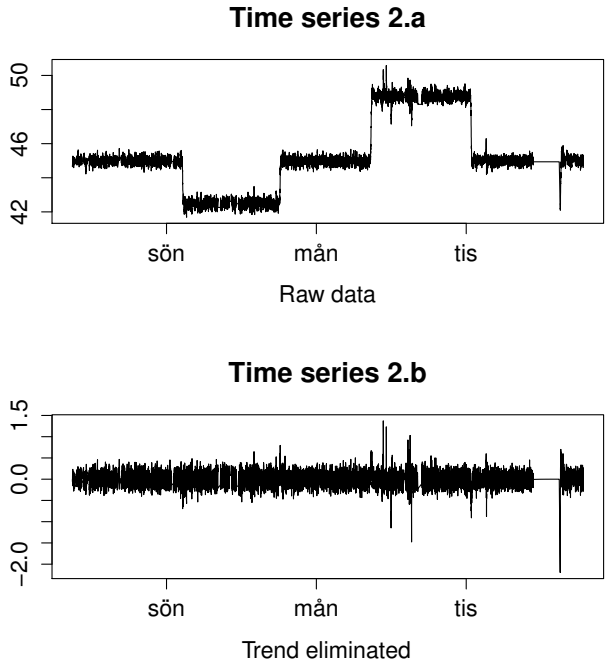

Figure 4.2: Time series 2.a and 2.b. The underlying process of this time series has different target values. When the trend is removed, time series 2.b, it is no longer possible to detect these levels.

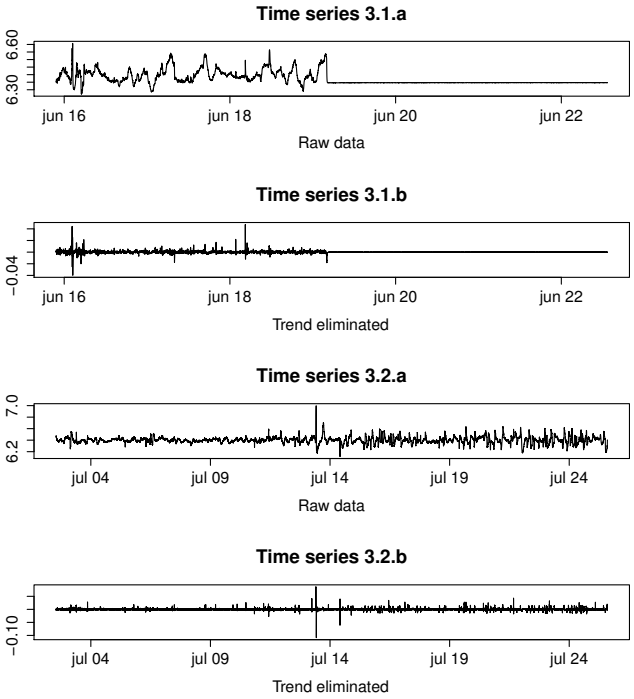

Figure 4.3: Time series 3.1.a, 3.1.b, 3.2.a and 3.2.b. Time series 3.1 and 3.2 originates from the same process and are extracted from different time periods. This time series has a high sampling frequency.

4.1.1 Details about the time series

Time series 1, Figure 4.1, consists of results from a manually performed test in the laboratory that measures impurity of recycled pulp gathered during a day. Impurity of the pulp may be due to deviations in the process and may affect the quality if the levels are too high. In an ideal state these measurements should have low and constant values. This time series is particularly interesting for anomaly detection as it shows a seasonal trend which may mask local spikes/outliers.

Time series 2, Figure 4.2 consists of online sensor measurements of a quality parameter of paper. The quality is adjusted to different target values depending on the currently produced quality. For each target level the values should preferably be constant. This time series is interesting for the anomaly detection algorithm as it contains different levels, and so the algorithm must be able to adapt to the current level.

Time series 3.1 and 3.2, Figure 4.3 consists of online sensor measurements of concentration of pulp. The process is adjusted to a single target value. This time series is interesting for the anomaly detection as it shows changes in the variance which would be useful to detect.

Compiled information about the time series is given in Table 4.1.

Table 4.1: Detailed information about time series 1-3.2. The table sates if the values of the time series are collected from a sensor or comes from laboratory tests. It also gives if the underlying process is adjusted to a target value or not. In addition the sampling frequency is given and the total number of points in the considered region.

Time Series	Value type	Adjusted	Frequency (Hz)	Number of points
1	Laboratory	No	1/day ($1.16 \cdot 10^{-5}$)	390
2	Sensor	Yes	1/20s (0.05)	12000
3.1	Sensor	Yes	1/10s (0.1)	26000
3.2	Sensor	Yes	1/10s (0.1)	149000

4.1.2 Suggested anomalies to detect

For the selected time series a couple of points that seems anomalous were pointed out manually, these are shown in figures 4.4 - 4.6. Note that these suggested anomalies might not be anomalous points in the context of the pulp and paper production and there might be anomalous point that are not discovered by this manual method. These deviating points are solely discovered by the pattern recognition by the human eye and do not include any knowledge about the process.

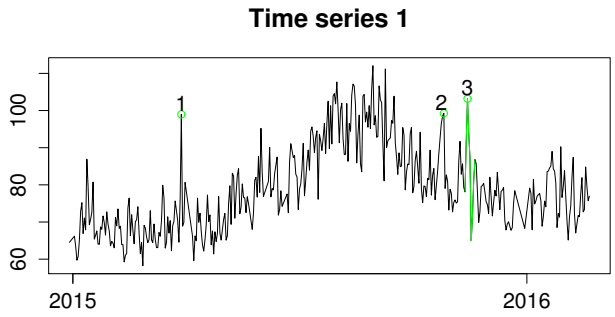

Figure 4.4: Suggested anomalies for time series 1. Anomalies 1 and 2 are point anomalies that locally have extreme values. Anomaly 3 is a sequential anomaly where there is an unusual rapid fluctuation.

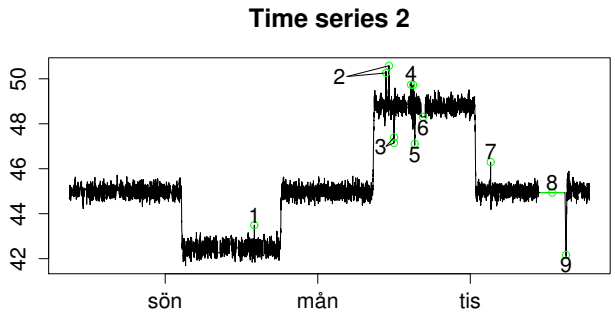

Figure 4.5: Suggested anomalies for time series 2. Anomalies 1-5, 7 and 9 are point anomalies that have extreme values locally. Anomalies 6 and 8 are sequential anomalies with sequences that have the same value for an unusually long period of time. The jumps are not considered anomalous as they correspond to changes in process settings.

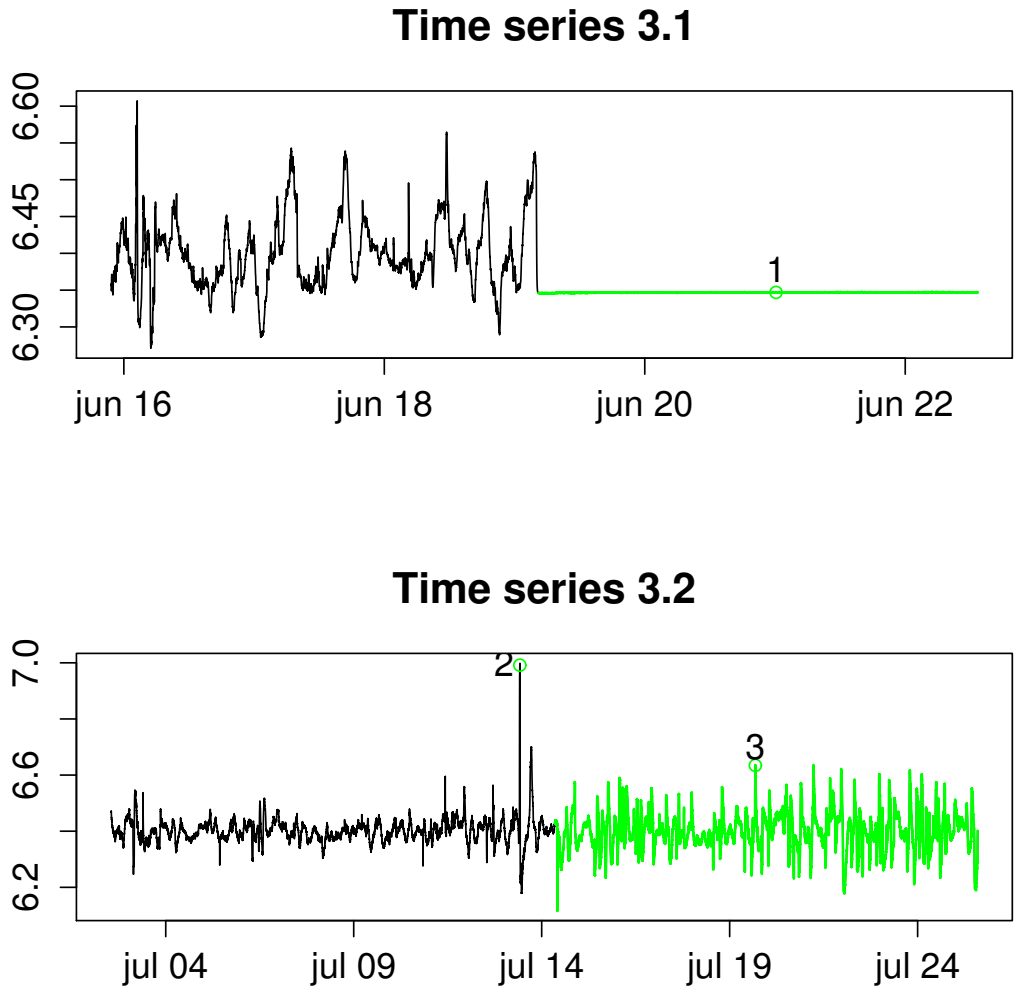

Figure 4.6: Suggested anomalies for time series 3. Anomaly 2 is a point anomaly since it has an extremely high value compared to the rest of the series. Anomalies 1 and 3 are sequential anomalies showing a decrease and increase in variance respectively.

The anomaly detection algorithm will be evaluated according to its ability to detect these anomalies.

4.2 Anomaly detection algorithm

In this section the results of the anomaly detection algorithm, explained in section 3.1, is presented. The anomaly detection algorithm is performed with different values of the parameters to evaluate how the algorithm is affected by the choices of these parameters.

4.2.1 Results of determining time frame for aggregation

To derive the behaviour of the autocorrelation function for the selected time series, they are plotted in Figure 4.7. Note that the autocorrelation drops quickly for time series 1 and 2.

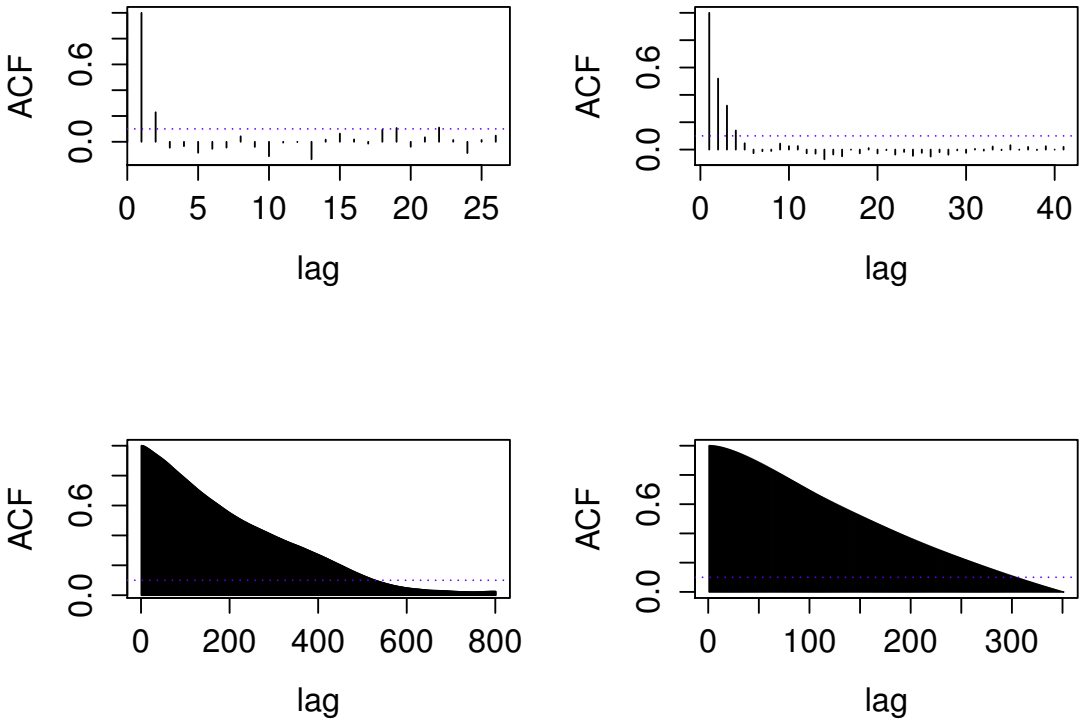

Figure 4.7: Autocorrelation functions for time series 1 (top left), 2 (top right), 3.1 (bottom left) and 3.2 (bottom right).

To determine the time frame for aggregation the method described in section 3.1.4 was applied with parameters number of repetitions $r = 10$, subsequence length $l = 1$month and threshold level $\alpha = 0.1$. The result is shown in Table 4.2.

Table 4.2: Time lags τ_α at which the autocorrelation function drops below the predefined level $\alpha = 0.1$ for the time series.

Time Series	Estimated τ
1	2
2	7
3.1	432
3.2	329

4.2.2 Impact of aggregation

In this section results of the algorithm for different aggregations are presented.

The numerical results in Table 4.2 are translated into the corresponding time frames, given in Table 4.3.

Table 4.3: Calculated approximate time frames for aggregation using the correlation method described in section 4.2.1.

Time Series	n	Corresponding time frame
1	2	2 days
2	7	3 minutes
3.1	432	1.2 hours
3.2	329	55 minutes

Based on the results in Table 4.3 three time frames were selected for each time series to evaluate the impact of aggregation, see Table 4.4.

Table 4.4: Time frames for aggregation that were used for evaluation. Time series 3.1 and 3.2 are considered jointly for this evaluation.

Time Series	n_1	n_2	n_3
1	1 day (original data)	2 days	1 week
2	1 minute	6 minutes	12 minutes
3.1 & 3.2	30 minutes	1 hour	2 hours

The anomaly detection algorithm was executed with the different time frames for aggregation. The results describe whether the anomalies pointed out in the figures 4.4 - 4.6 are detected or not, the number of additional anomalies that were found and the number of anomalies found per number of input points. A sequential anomaly as anomaly 1 in Figure 4.6 is not considered as detected if only a single point in that interval is detected as anomalous. The results for the three time series, with and without trend component, are shown in tables 4.5 - 4.10. For these results the other parameters were fixed as follows:
Training size 60
k for control limit 3
trend estimation method 'EWMA'
θ **for 'EWMA'** 0.8
Visual representations of the detected anomalies are found in appendix C.

Time series 1

Table 4.5 shows the result for different aggregations for time series 1.a. From this table we see that the number of found anomalies decreases with increased length of the time frame for aggregation. Regarding the feature DTPP anomalies 2 and 3 are found for the aggregation with longer time frame, which suggest that the aggregation in this case clarifies patterns. However, anomaly 1 that was detected for the shorter time frame for aggregation is lost, which in contrast suggests that the aggregation also masks patterns.

Table 4.5: Anomalies found for time series 1.a with different aggregation. Anomalies 1,2 and 3 refer to the anomalies in Figure 4.4. • indicates that the anomaly was found and ∘ that it was not. The number of other anomalies that were found is also presented. In these trials k was set to 3 and the training size was 60.

		Anomalies found				
Feature	Aggregation	1	2	3	Other	Frequency
Raw data	1 day	•	∘	∘	1	0.5%
	2 days	∘	∘	∘	1	0.5%
	3 days	∘	∘	∘	1	0.7%
DTPP	1 day	•	∘	∘	3	1.0%
	2 days	∘	•	•	0	1.0%
	3 days	∘	•	•	0	1.5%
VOS	1 day	∘	∘	∘	1	0.5%
	2 days	∘	∘	∘	0	0.0%
	3 days	∘	∘	∘	0	0.0%
Total	1 day	•	∘	∘	4	1.3%
	2 days	∘	•	•	1	1.5%
	3 days	∘	•	•	1	2.3%

Table 4.6 shows the result for different aggregations for time series 1.b, i.e. time series 1 with the trend removed. Comparing the results between time series 1.a and 1.b shows that anomaly 3 was found studying the feature raw data when the trend was removed which was not the case for time series 1.a.

Table 4.6: Anomalies found for time series 1.b with different time frames for aggregation. Anomalies 1,2 and 3 refer to the anomalies in Figure 4.4. • indicates that the anomaly was found and ∘ that it was not. The number of other anomalies that were found is also presented. In these trials k was set to 3 and the training size was 60.

		Anomalies found				
Feature	Aggregation	1	2	3	Other	Frequency
Raw data	1 day	•	∘	•	1	0.8%
	2 days	∘	∘	∘	0	0.0%
	3 days	∘	∘	∘	0	0.0%
DTPP	1 day	•	∘	∘	3	1.0%
	2 days	∘	•	•	0	1.0%
	3 days	∘	•	•	0	1.5%
VOS	1 day	∘	∘	∘	0	0.0%
	2 days	∘	∘	∘	0	0.0%
	3 days	∘	∘	∘	0	0.0%
Total	1 day	•	∘	•	4	1.5%
	2 days	∘	•	•	0	1.0%
	3 days	∘	•	•	0	1.5%

Impact of aggregation on time series 2

Table 4.7 shows the result for different aggregations for time series 2.a. From this table we see, similarly to time series 1, that the number of found anomalies decreases with increased time frame for aggregation.

Table 4.7: Anomalies found for time series 2.a with different aggregation. Anomalies 1-9 refer to the anomalies in Figure 4.5. • indicates that the anomaly was found and ∘ that it was not. The number of other anomalies that were found is also presented. In these trials k were set to 3 and the training size was 60.

		Anomalies found										
Feature	Aggregation	1	2	3	4	5	6	7	8	9	Other	Frequency
Raw data	1 min	•	•	•	•	•	∘	•	∘	•	79	2.0%
	6 min	∘	∘	∘	∘	•	∘	∘	∘	•	33	4.6%
	12 min	∘	∘	∘	∘	∘	∘	∘	∘	•	28	7.5%
DTPP	1 min	•	•	•	•	•	∘	•	∘	•	22	0.7%
	6 min	•	∘	∘	∘	∘	∘	∘	∘	•	12	1.8%
	12 min	∘	∘	∘	∘	∘	∘	∘	∘	•	9	0.3%
VOS	1 min	∘	∘	∘	∘	∘	∘	∘	•	∘	29	0.7%
	6 min	∘	∘	∘	∘	∘	∘	∘	•	∘	4	0.7%
	12 min	∘	∘	∘	∘	∘	∘	∘	∘	∘	2	0.5%
Total	1 min	•	•	•	•	•	∘	•	•	•	130	3.2%
	6 min	•	∘	∘	∘	•	∘	∘	•	•	49	6.9%
	12 min	∘	∘	∘	∘	∘	∘	∘	∘	•	39	10.3%

Table 4.8 shows the result for different time frames for aggregation for time series 2.b, i.e. time series 2 with the trend removed. Comparing the results between time series 2.a and 2.b we see that more of the suggested anomalies are found when the trend is eliminated.

Table 4.8: Anomalies found for time series 2.b with different time frames for aggregation. Anomalies 1-9 refer to the anomalies in Figure 4.5. • indicates that the anomaly was found and ∘ that it was not. The number of other anomalies that were found is also presented. In these trials k was set to 3 and the training size was 60.

		Anomalies found										
Feature	Aggregation	1	2	3	4	5	6	7	8	9	Other	Frequency
Raw data	1 min	•	•	•	•	•	∘	•	∘	•	38	1.1%
	6 min	∘	∘	•	∘	•	∘	∘	∘	•	11	1.8%
	12 min	∘	∘	∘	∘	∘	∘	∘	∘	∘	7	1.8%
DTPP	1 min	•	•	•	•	•	∘	•	∘	•	22	0.7%
	6 min	∘	•	•	•	•	∘	∘	∘	•	8	1.7%
	12 min	∘	∘	∘	∘	∘	∘	∘	∘	∘	9	2.3%
VOS	1 min	∘	∘	∘	∘	∘	∘	∘	•	∘	10	0.3%
	6 min	∘	∘	∘	∘	∘	∘	∘	•	∘	4	0.7%
	12 min	∘	∘	∘	∘	∘	∘	∘	∘	∘	1	0.3%
Total	1 min	•	•	•	•	•	∘	•	•	•	70	1.8%
	6 min	∘	•	•	•	•	∘	∘	•	•	23	3.8%
	12 min	∘	∘	∘	∘	∘	∘	∘	∘	∘	17	4.4%

Impact of aggregation on time series 3

Time series 3.1 and 3.2 are considered jointly when evaluating the impact of aggregation. The only feature that shows a difference in detected suggested anomalies for the different aggregations is VOS.

Table 4.9: Anomalies found for time series 3.a with different aggregation. Anomalies 1,2 and 3 refer to the anomalies in Figure 4.6. • indicates that the anomaly was found and ∘ that it was not. The number of other anomalies that were found is also presented. In these trials k was set to 3 and the training size was 60.

		Anomalies found				
Feature	Aggregation	1	2	3	Other	Frequency
Raw data	30 min	∘	•	∘	14	1.1%
	1 hour	∘	•	∘	6	1.0%
	2 hours	∘	•	∘	3	1.1%
DTPP	30 min	∘	•	∘	22	1.6%
	1 hour	∘	•	∘	10	1.5%
	2 hours	∘	•	∘	3	1.1%
VOS	30 min	•	∘	•	3	0.4%
	1 hour	∘	∘	•	0	0.1%
	2 hours	∘	∘	•	0	0.0%
Total	30 min	•	•	•	49	3.7%
	1 hour	∘	•	•	22	3.4%
	2 hours	∘	•	•	7	2.5%

Table 4.10: Anomalies found for time series 3.b with different time frames for aggregation. Anomalies 1,2 and 3 refer to the anomalies in figure 4.6. • indicates that the anomaly was found and ∘ that it was not. The number of other anomalies that were found is also presented. In these trials k was set to 3 and the training size was 60.

		Anomalies found				
Feature	Aggregation	1	2	3	Other	Frequency
Raw data	30 min	∘	•	∘	17	1.3%
	1 hour	∘	•	∘	12	1.8%
	2 hours	∘	•	∘	6	2.0%
DTPP	30 min	∘	•	∘	28	2.0%
	1 hour	∘	•	∘	15	2.2%
	2 hours	∘	•	∘	4	1.4%
VOS	30 min	•	∘	•	4	0.4%
	1 hour	•	∘	∘	0	0.1%
	2 hours	∘	∘	∘	0	0.0%
Total	30 min	•	•	•	50	3.7%
	1 hour	•	•	∘	27	4.1%
	2 hours	∘	•	∘	10	3.1%

4.2.3 Impact of training size

The training size influences the estimation of μ and σ that are used for determining the control limits. A large training size gives a more accurate estimation of the parameters but has the down side of taking more time to adapt to changes in the data. To evaluate how the training size influences the estimation 1000 samples with size equal to the investigated training size were taken for which the parameters were estimated. The average and the standard deviation of the estimations were recorded and can be found in Table 4.11. The estimated μ and σ for the whole series is given as a reference. However note that the time series itself is a sampling of an underlying distribution for which we want to estimate the parameters and thus an increased number of entries in the time series would introduce an uncertainty to the largest training sets as well. As expected, the standard deviation of the estimated parameters drops as the training size increases.

Table 4.11: The average and standard deviation of parameter estimations from 1000 samples from the time series with different sample sizes.

Time series	Training size	$\bar{\mu}$	$s_{\bar{\mu}}$	$\hat{\sigma}$	$s_{\hat{\sigma}}$
1.a	30	79.2	2.03	11.4	1.29
	60	79.2	1.36	11.5	0.855
	120	79.3	0.865	11.5	0.541
	390	**79.3**	—	**11.5**	—
1.b	30	0.268	1.09	6.02	0.832
	60	0.29	0.707	6.05	0.566
	120	0.269	0.466	6.07	0.359
	390	**0.275**	—	**6.09**	—
2.a	30	45.3	0.369	2.02	0.242
	60	45.3	0.273	2.04	0.17
	120	45.3	0.186	2.04	0.12
	12000	**45.3**	—	**2.04**	—
2.b	30	-0.00133	0.0367	0.203	0.0511
	60	0.00096	0.0266	0.0206	0.040
	120	0.00023	0.00191	0.207	0.029
	12000	**0.00018**	—	**0.209**	—
3.1.a	30	6.39	0.00865	0.0486	0.00721
	60	6.39	0.0062	0.0486	0.00494
	120	6.39	0.0044	0.0485	0.00356
	26000	**6.39**	—	**0.0487**	—
3.1.b	30	$2.41 \cdot 10^{-6}$	0.00049	0.00236	0.00122
	60	$1.17 \cdot 10^{-5}$	0.00034	0.00252	0.00104
	120	$5.23 \cdot 10^{-6}$	0.000251	0.00259	0.000813
	26000	$\mathbf{-4.23 \cdot 10^{-7}}$	—	**0.0027**	—
3.2.a	30	6.4	0.0117	0.0592	0.0137
	60	6.4	0.0080	0.06	0.0101
	120	6.4	0.0055	0.0605	0.0073
	149000	**6.4**	—	**0.0608**	—
3.2.b	30	-0.00012	0.00578	0.0302	0.0104
	60	-0.000178	0.0041	0.0312	0.00793
	120	-0.000041	0.00297	0.0316	0.0061
	149000	**-0.00115**	—	**0.0322**	—

To further study how the training size affects the behaviour of the control limits for time series with a changing trend, a graphical representation for different training sizes is presented in Figure 4.8. Notice that for time series 1.a the control limits

lie closer to the input data for small training sizes, i.e. they adapt to the current pattern. However, also note that the control limits are more influenced by extreme values when the training size is small. Time series 2.a shows that the method is very influenced by quality shifts causing the control limits to be far from the input data a short period after the quality shift. This effect is caused by the rapid change in target value, causing the estimated $\hat{\mu}$ and $\hat{\sigma}$ to deviate from the expected value while the algorithm is adapting. This effect is increased for larger training sizes. This usually makes the control limits unusable for this period of time.

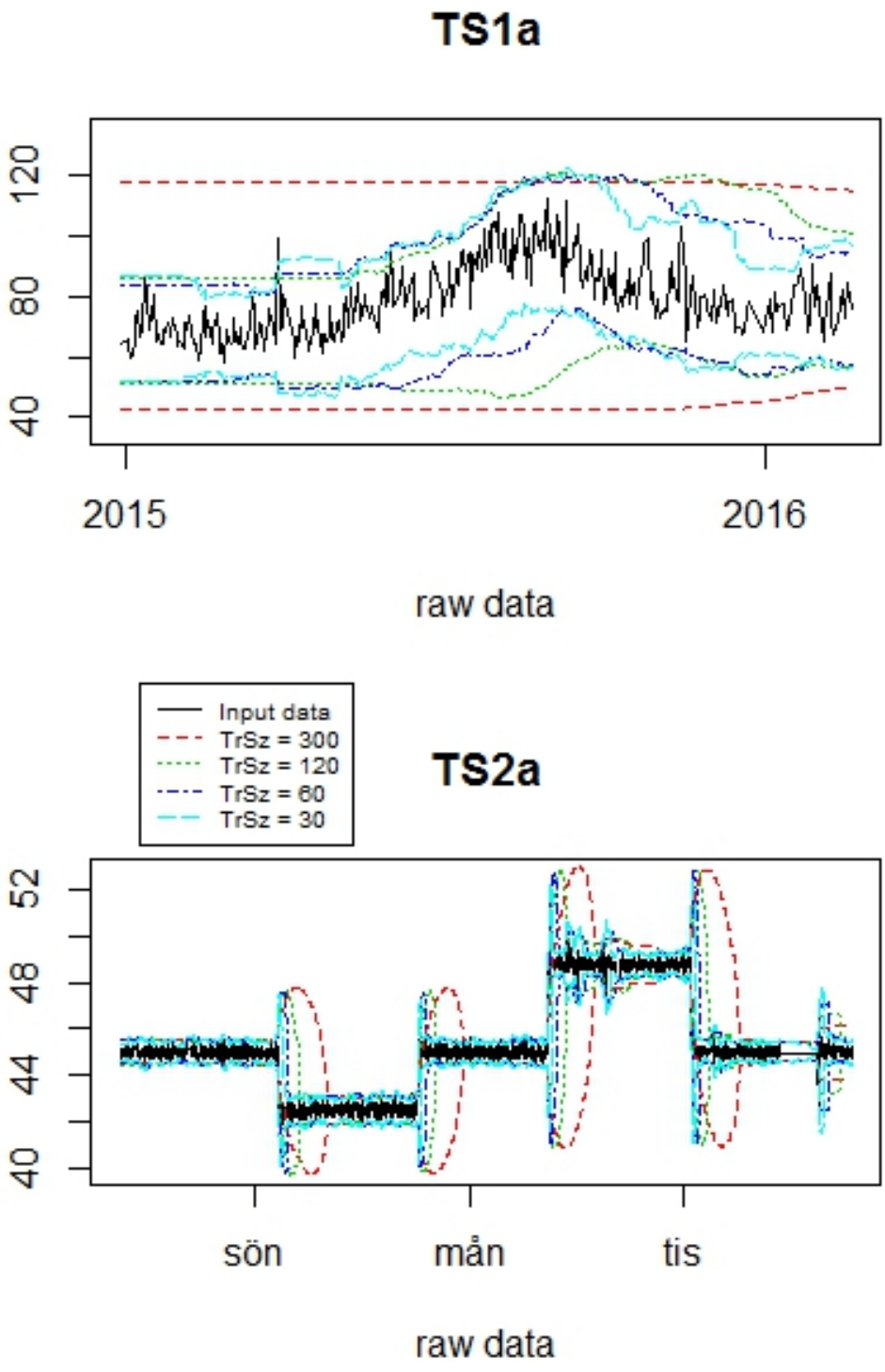

Figure 4.8: Control limits for different training sizes for time series 1.a (top) and 2.a (bottom), both for raw data.

This method was also applied to the feature difference to previous point of the time series, shown in Figure 4.9. Notice that this feature eliminates the trend and thus

also the negative influence of the trend shifts on the control limits.

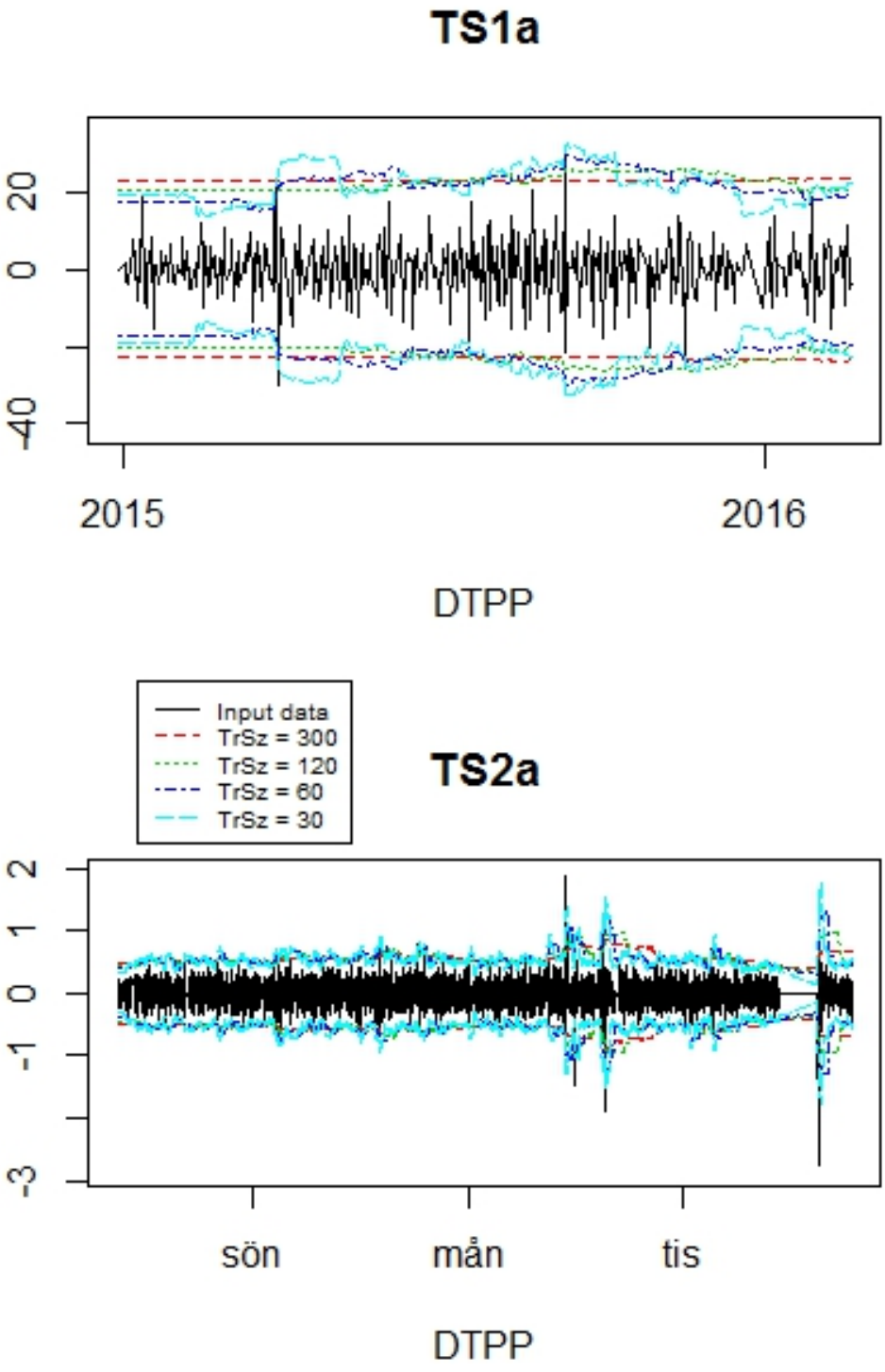

Figure 4.9: Control limits for different training sizes for time series 1.a (top) and 2.a (bottom), both for the feature difference to previous point.

Lastly, the method was applied to the feature variance of segment for time series 3.2.a, shown in Figure 4.10. This feature shows several spikes which influences the control limits in a similar way to the quality shifts.

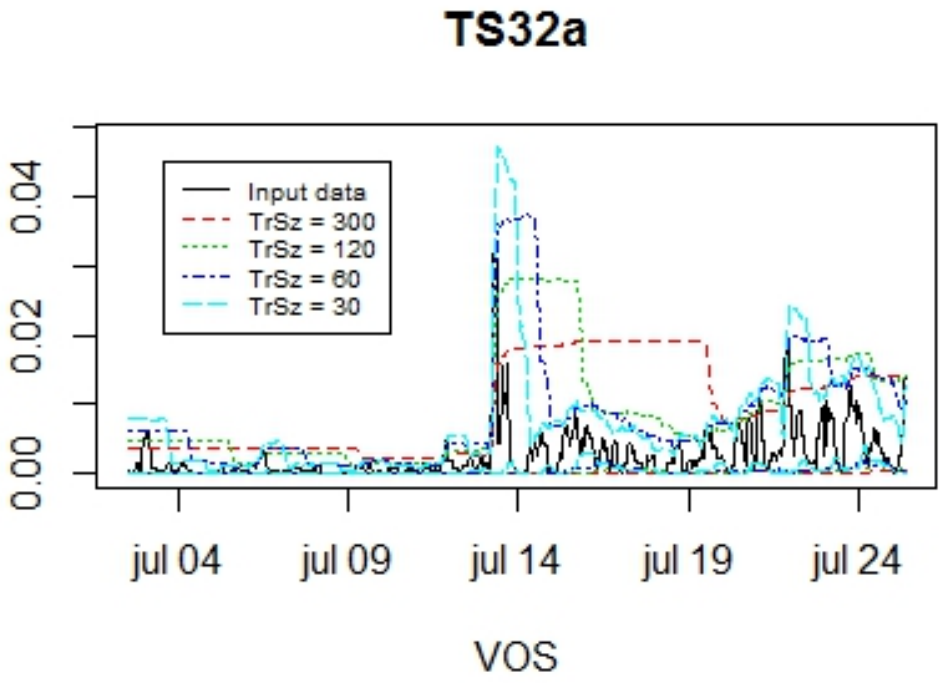

Figure 4.10: Control limits for different training sizes for time series 3.2.a for the feature variance of segment.

4.2.4 Impact of k

The parameter k is used to determine the control limits, where the control limits are given by $\pm k\hat{\sigma}$ and $\hat{\sigma}$ is the estimated standard deviation. For the variance, which is assumed to be χ^2 distributed, the control limits are determined by finding the limits where a sample has probability p to lie outside of the limits. The choice of k and p is a trade-off between false positives and false negatives. A common practice in the Shewhart diagram is to use $k = 3$ which corresponds to $p = 0.05$. Figure 4.11 shows how the number of detected anomalies depends on the choice of k for each of the time series. A desired result is to have a small difference in the number of found anomalies for small changes in k.

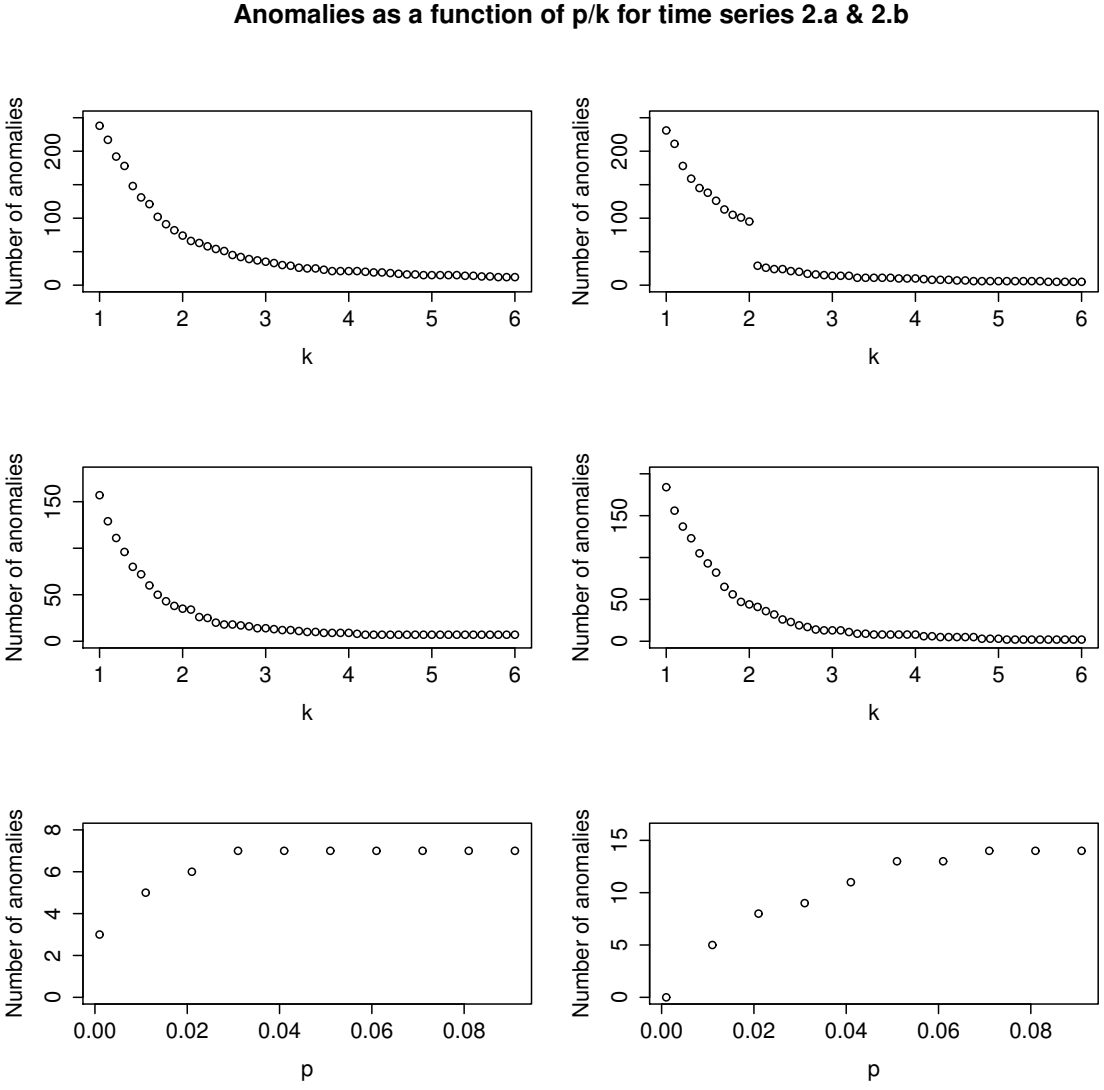

Figure 4.11: The number of anomalies detected from raw values (top), DTPP (middle) and VOS (bottom) as a function of parameter k that influences the control limits for time series 2.a (left) and 2.b (right). The time series is aggregated over 6 minutes.

To further analyse how the choice of k affects the detection of anomalies, the same method as before is applied together with information of when the suggested anomalies from Figure 4.5 are not longer detected, shown in Figure 4.12. Note that a larger value of k could be used to reduce the number of detected anomalies without missing the suggested anomalies.

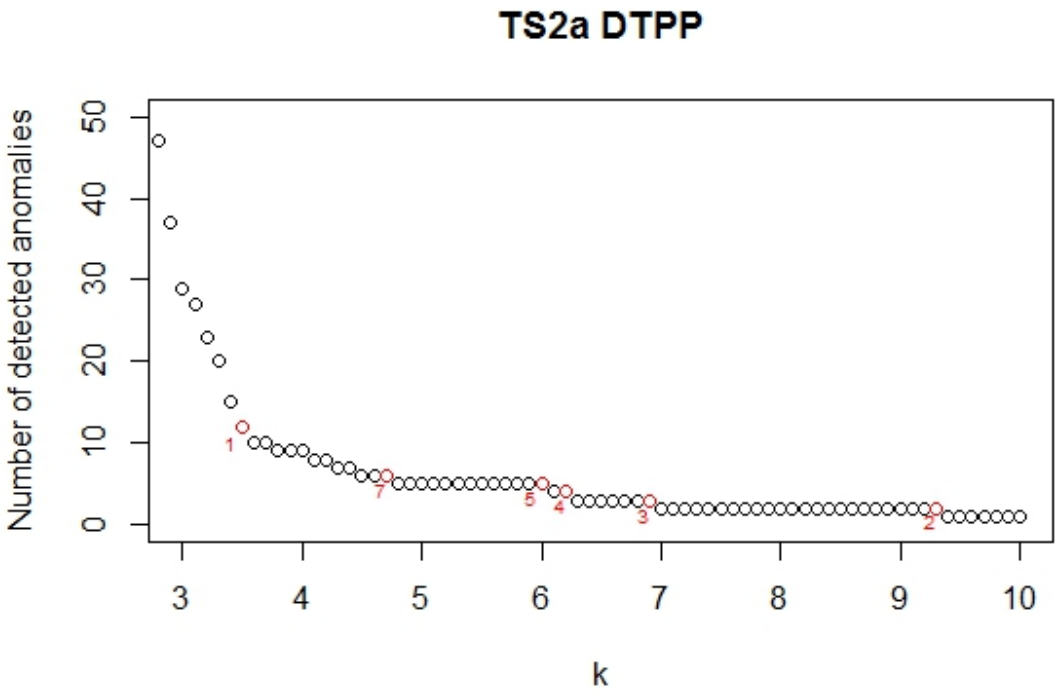

Figure 4.12: The number of anomalies detected for feature difference to previous point of time series 2.a. The time series is aggregated over 6 minutes. The breaking point where the suggested anomalies, Figure 4.5, are not longer detected are marked in red with the corresponding number next to it.

4.2.5 Normality assumption

The anomaly detection algorithm relies on the assumption that the input data, or at least the subsequences that the parameters are estimated from, is normally or χ^2 distributed. Q-Q plots is used to evaluate the normality assumption. Figure 4.13 shows Q-Q plots of two representative subsequences of the feature DTPP for time series 2.a. The figure also includes a Q-Q plot for the same feature over the entire time series. The figure shows that some subsequences can be considered normally distributed while others do not. Over all, the Q-Q plot for the entire time series shows that data is heavy tailed. Q-Q plots for the other time series are found in appendix E

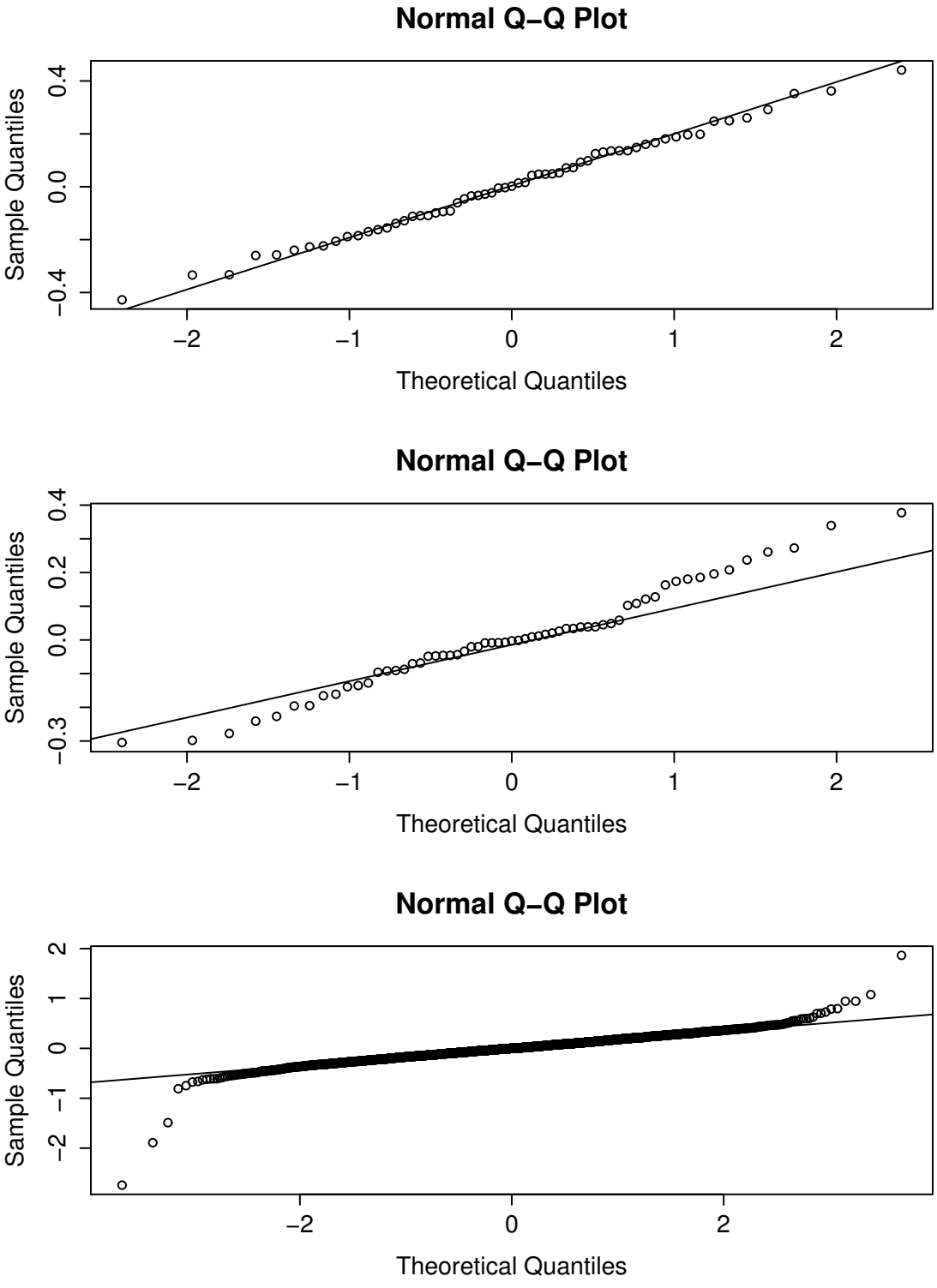

Figure 4.13: Q-Q plots of the feature difference to previous point of two representative subsequences (top and middle) and the entire period (bottom) for time series 2.a .

4.3 Change in slope of the trend

In this section the results from the implementation of the algorithm for detecting changes in the trend is presented. The algorithm is evaluated with different choices of the parameters for time series 1 and 2 since they show a change in the trend. The algorithm is applied to time series 3.1 for evaluation of the performance on a time series without a change in trend, Figure 4.16. In figures 4.14 - 4.15 we see that the number of segments and thus the reported changes in trend decreases as the training size and α increases.

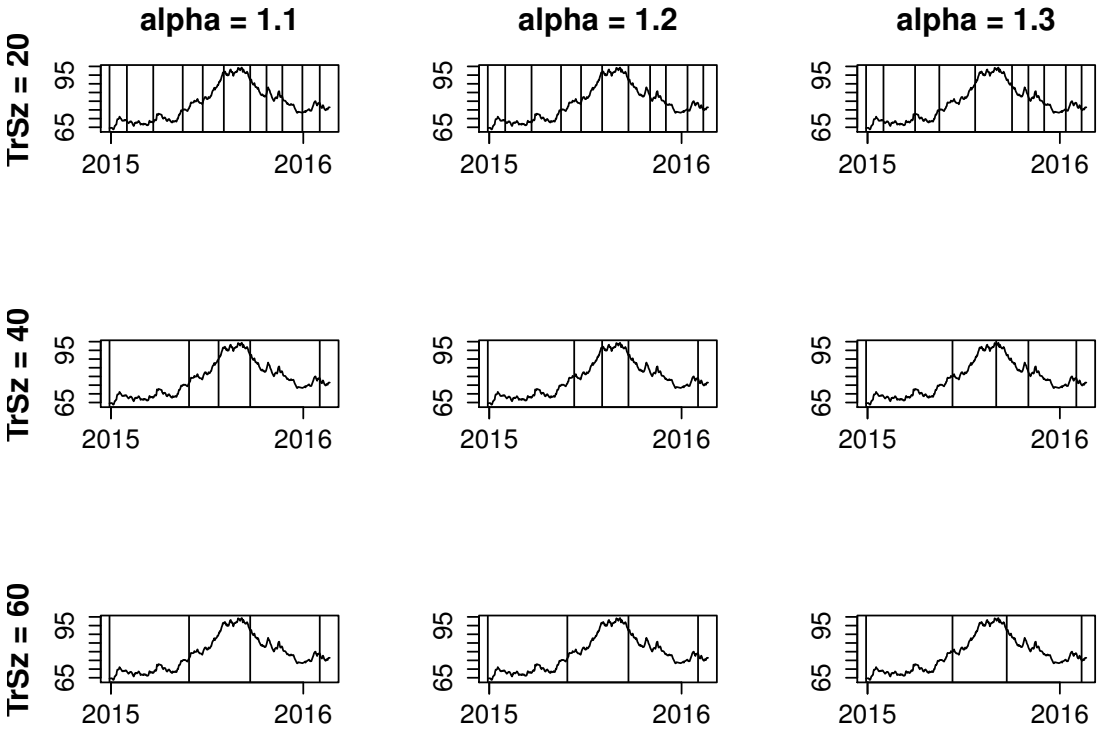

Figure 4.14: Result from detecting changes in trend for different values of training size and α for time series 1. The vertical lines indicate where a new segment is created.

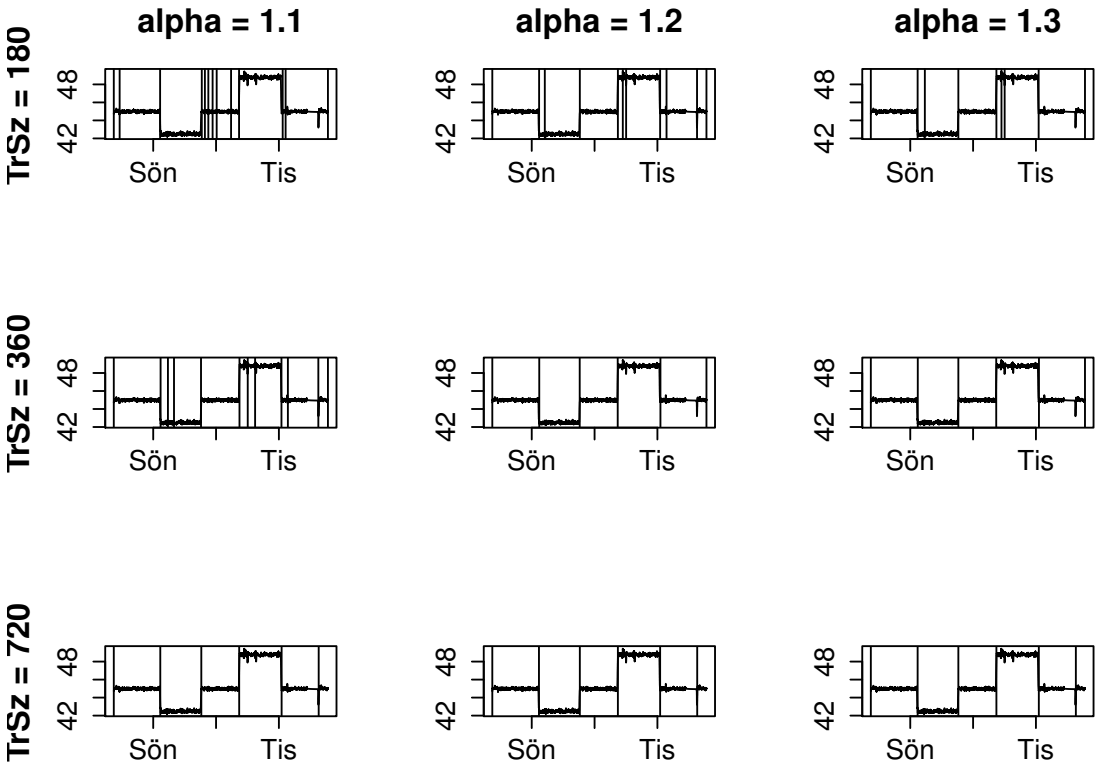

Figure 4.15: Result from detecting changes in trend for different values of training size and α for time series 2. The vertical lines indicate where a new segment is created.

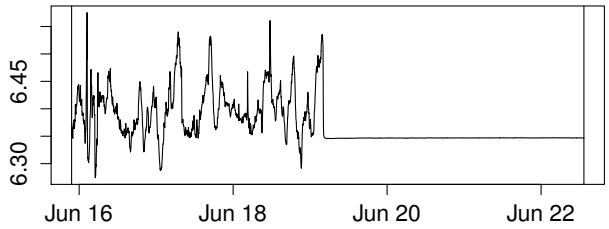

Figure 4.16: Result from detecting changes in trend for time series 3.1 with training size of 2000 (≈5.5h) and $\alpha = 1.3$. The vertical lines indicate where a new segment is created.

4.4 Time complexity

An important aspect of the developed algorithm is the time complexity. If the algorithm should be able to run in an online environment with streaming data of

Table 4.12: Time complexity per sample of anomaly detection algorithm, where n is the time frame for aggregation, l is the length of the sequence that the variance is calculated over and $TrSz$ is the training size for estimating the mean and standard deviation for the control limits.

Step	Time complexity
Eliminating trend	$O(1)$
Aggregation	$O(n)$
Extraction of features	$O(l)$
Detect anomaly	$O(1)$
Update control limits	$O(TrSz)$
Total	$O(\max(n, l, TrSz))$

Table 4.13: Time complexity per sample of algorithm for detecting changes in the trend, where $TrSz$ is the training size, max_length is the maximum length of the subsequence.

Step	Time complexity
Calculate trend	$O(1)$
Fit a line in the least square sense and calculate max_error	$O(TrSz)$
Add l points and fit linear trend	$O(max\_length)$
Detect change in trend	$O(1)$
Total	$O(max\_length)$

a sampling frequency of at least 0.1Hz the time complexity must be low so that the execution is finished before the next sample arrives. In addition, the algorithm should be able to analyse upwards of 20 000 time series simultaneously.
Table 4.12 outlines the time complexity for the anomaly detection algorithm.
Table 4.13 outlines the time complexity for the algorithm that detects changes in the trend.

5
Discussion

This chapter discusses the results of the project. The chapter begins with evaluating to what extent the aims of the thesis were fulfilled. This is followed by evaluations of the performances of the anomaly detection algorithm and the trend change detection algorithm. The chapter ends with ideas for improvements for future work.

5.1 Evaluation of the goals of the thesis

The anomaly detection algorithm was developed to be applicable to any arbitrary time series of the pulp and paper production process. This generality results in limitations when it comes to defining patterns of anomalies and determining the parameters of the algorithms.

A difficulty during the project has been that there is no straightforward way to define patterns that in general separates abnormal behaviour from normal, due to the fact that what is considered normal and abnormal depends on the underlying process. A certain pattern in one time series may be considered anomalous while the same pattern in another time series may be considered normal.

Determining the parameters of the algorithm becomes challenging when implementing the algorithm in an arbitrary setting, since suitable choices of the parameters change with the specific time series due to the reasons mentioned above. This created the need to develop a method that adaptively chooses the values of the parameters. This introduces another source of errors apart from the performance of the algorithm itself. As a contrast, if the algorithm was to be implemented for a single time series, the parameters could be chosen by hand to give optimal performance. However, if the method that adaptively determines values for the parameters is successful the algorithm will be powerful as it can be applied to a variety of time series. The results of the method are promising but further evaluation is needed.

Another precondition for the project was that the algorithm should only use historical data to distinguish between normal and abnormal events. This has the advantage of not needing labelled data. The detection of anomalies may however be more accurate by training on labelled data, but extracting labelled training data is not feasible in the target application due to the vast amount of work required.

The implemented method using a statistical approach for anomaly detection has the advantage of being time efficient. It can also provide a measure of the degree of anomaly together with confidence interval for the detected anomaly. The statistical approach is also suited for the input data, since it is numerical and patterns can be extracted as a quantitative measure. In addition it is reasonable to believe that the data follows a normal distribution. This assumption is supported by the Q-Q plot shown in Figure 4.13. Other detection techniques, such as clustering techniques, may be beneficial with other preprocessing techniques of the data.

Furthermore, the algorithm was developed with the aim of being user friendly. The algorithm is considered user friendly if the presentation of the anomalies is clear and it is easy to use, e.g. there are few parameters for the user to tune. The algorithm is implemented such that it performs adequately with predetermined parameters as defined in the project. At this stage there is no user interface implemented. The design of the future user interface will have a great impact on the user experience.

Moreover, the algorithm should be able to run in an online environment, which demands that the algorithm is time efficient. As of this project the possibility to run the algorithm in an online environment has not been tested in a real-life environment. However, a theoretical investigation on the time complexity was done. It shows that both the anomaly detection algorithm and the algorithm for detecting changes in the trend are linear in input size which is a desirable result. However, this may still cause restrictions on possible values of the input parameters. For the algorithm that detects changes in the trend the time complexity is $O(TrSz + n * l)$ where $TrSz$ is the training size, l is the number of points added in each round and n is the number of repetitions. If no change in trend is reported for many iterations n, $TrSz + n * l$ will become very large. To lower the time complexity a maximum number of iterations n can be set. After these n iterations a new segment can be created without reporting a change in trend.

Due to time constraints the part of the aim concerning writing detected anomalies to a database was not implemented. However, there is no inherent limitation to doing this in the algorithm, and it should be possible by formatting the output data correctly. Storing the anomalies in a database would have the advantage of being able to further study eventual relationships between the anomalies. Such findings may in turn give valuable information about the dynamics of the process itself.

In conclusion, the anomaly detection algorithm is able to find deviating patters captured by the extracted features. It was also shown that the trend detecting algorithm was successful. The implemented algorithms meet the aims except for writing the detected anomalies to a database, which is not implemented as of yet. In addition, whether the detection was performed with a reasonably low frequency of false positives and false negatives could not be confirmed since there is no labelled test data at this point. There were more points detected as anomalous than expected by chance under the assumption of normally distributed data. This suggests that either there were many anomalous points in the time series or that the normality

assumption does not hold true. This may be further investigated using labelled test data. Labelled test data would also be useful to evaluate the choice of parameters.

5.2 Performance of the anomaly detection algorithm

It is clear from tables 4.5 - 4.10 that it is possible to detect anomalies with the proposed anomaly detection algorithm. However, the lack of labelled test data has limited the development and evaluation of the algorithm. Given labelled data it would be possible to find more specific features defining anomalies. It would also be possible to evaluate the influence of the parameters on the anomaly detection algorithm more accurately. However, from the produced results it is possible to study how the parameters time frame n for aggregation, training size and k for control limit affects the anomaly detection in general.

- **Impact of aggregation** Tables 4.5 - 4.10 show that the investigated aggregations in general rather masks than clarifies patterns for large number of points n used for aggregation. However, as seen in tables 4.5 and 4.6 the aggregation may also both clarify and mask critical patterns. A way to avoid this scenario must be found to increase the performance of the algorithm. The autocorrelation function was utilised to find a suitable time frame n for aggregation. From the results of tables 4.5 - 4.10 it is hard to conclude if these are good indications of n due to two reasons. Firstly, by the lack of labelled test data it is impossible to say if the aggregation masks or clarifies patterns. Secondly, the results are also dependent on the choices of the other parameters. However, the results indicates that the investigated values of n are reasonable to use for aggregation as they approach the optimal point for decreasing the number of points while still not masking interesting patterns. We see that as in general more suggested anomalies are found for the smaller time frame for aggregation.
- **Impact of training size** The choice of training size is a trade off between quickly adapting to new patterns and certainty of the estimated parameters. Table 4.11 gives an idea of how the estimated parameters are distributed for different training sizes. As expected the standard deviation of the estimated variables decreases with an increase in training size. Furthermore, figures 4.8-4.10 show the behaviour of the control limits as a function of the training size. A small training size better adapts when there are gradual changes in the trend of the time series. However, for small training sizes the influence of singular extreme values is large. If there is no reason to believe that new patterns will arise in the time series, a larger training size is preferred. When calculating control limits for the variance it seems preferable to use a larger training size to account for the extreme values. A possible improvement in this area would be to perform some kind of outlier removal before updating the control limits. There is also the possibility to investigate the potential of developing a "smart" method that adapts the training size to these prerequisites.
- **Impact of k/p** The parameter k determines the control limits together with the estimated standard deviation. In effect, k influences the border that sep-

arates normal points from anomalous. It is not possible to quantitatively determine the choice of k as we have no labelled test data. If such information would be available k could be evaluated by analysing the number of false positives and false negatives. However, we get an indication on what k/p that is suitable from Figure 4.11. For a suitable k/p it is preferable to have a small difference in the number of found anomalies with small deviations in k. This connects to the estimation of the standard deviation. If the estimation is far from the actual standard deviation it is equivalent to alter the value of k/p. It is preferable if the algorithm is robust with respect to such deviations. Especially the result in Figure 4.12 suggests an increased value of k to reduce the total number of detected anomalies while still detecting the suggested anomalies. This is an example on how labelled data could come in use to tune parameters.

5.3 Performance of trend change detection

This method is a first attempt to detect changes in the trend. From figures 4.14 and 4.15 it is clear that this method works well with suitable choices of α and $TrSz$ by visual inspection of the data. A weakness of the algorithm is that the initial subsequence may capture a trend change, for example a quality shift, and in such case initialise the algorithm with bad choices of parameters. The algorithm could thus need a method to reduce the probability of such scenario. To further improve the performance of the trend change detection algorithm, a better method could be used to calculate the trend. Currently the calculated trend is ragged, as seen in figures 4.14 and 4.15, but a smooth and close to piecewise linear trend would be favourable. A better performing trend change detection would result in a more stable detection with fewer false positives.

5.4 Future work

The following list presents a few possible areas of improvements to further improve the algorithm and realise the aims of the project as described in section 1.1.

- **Selection of Data** Three out of the fourteen provided time series with different characteristics were selected as representatives for evaluating the anomaly detection algorithm. These time series capture a variety of features that the time series in a pulp and paper industry possess. These features include different sampling frequencies, adjusted and non-adjusted processes, laboratory and sensor values, processes with one, multiple or no target value among others. However, as there are way more time series collected in a pulp and paper production a more comprehensive study of these might be necessary to capture all properties of the targeted time series.
- **Labelled test data** A limitation of this project is the lack of any labelled data and a possible next step would be testing the algorithm on labelled data. As stated earlier, it is not possible to label all time series in a pulp and paper industry, but labelling a few time series would probably improve the result for

more time series than the labelled ones. Testing the algorithm on labelled data would allow for better verification as to whether the algorithm detects all the known anomalies or not. By using labelled data it would also be possible to extract more specific features that separates anomalous instances from normal ones. In addition, using a labelled data set, machine learning techniques could be applied to tune parameters.

- **Improvement of the algorithm** From the current state of the algorithm, the greatest improvement is probably to refine the features that distinguish normal instances from abnormal ones. A good way to do this would be by either using labelled data or by applying prior process knowledge. A feature with the only purpose to find anomalies of a specific type of time series is still valuable if it does not cause false positives in other time series. Even if the feature causes false positives it could be used only for the time series where it is relevant. There is also the possibility to use other analysing tools, such as analysing the time series in the frequency domain using spectral analysis.
- **Implementation and evaluation of online performance** One of the main aspects of the anomaly detection algorithm was its ability to run in an online environment. This was only tested theoretically and the next step would be to implement the algorithm in a real world scenario to test if it would be realisable.
- **Design of user interface** The anomaly detection algorithm was implemented with the intention to be user friendly. This was accomplished by implementing methods to automatically set the values of the parameters. An important part for the final application to be user friendly is the design of the user interface. This entails presenting the detected anomalies in an informative way and detailing the conditions under which the anomalies were detected. The design of such a user interface would be a major project in itself.
- **Writing anomalies to a database** As mentioned this part of the aim was not completed. This relatively straightforward step would form the basis for further analysis of the behaviour of the found anomalies.

6
Conclusion

The developed anomaly detection algorithm was shown to be able to detect anomalies. In fact, most of the beforehand suggested anomalies of the selected time series were detected for suitable choices of the parameters. Since there is no labelled test data given, the algorithm cannot be evaluated in detail, i.e. the true number of false positives and false negatives is unknown. Labelled data would therefore be valuable for future development of the algorithm.

The proposed algorithm to detect changes in the trend gives desirable results for the considered time series with suitable choices of the parameters. However, as this method is a first attempt to detect changes in the trend further investigation is needed to conclude its performance.

Theoretically, both the anomaly detection algorithm and the algorithm to detect changes in the trend could run in an online environment. That is since their time complexities are estimated to be linear in input size. However, it remains to be tested if it is possible to run the algorithms in a real-life situation.

Bibliography

[1] *Industrie 4.0: Smart Manufacturing for the Future*. Requested 2016-06-14. 2014. URL: http://www.gtai.de/GTAI/Content/EN/Invest/_SharedDocs/Downloads/GTAI/Brochures/Industries/industrie4.0-smart-manufacturing-for-the-future-en.pdf.

[2] Kenneth W. Kemp. *The efficient use of quality control data*. English. Vol. 4. Oxford: Clarendon, 2001. ISBN: 9780198536741;0198536747;

[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3 (July 2009), 15:1–15:58. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882.

[4] Eamonn Keogh et al. *Segmenting Time Series: A Survey and Novel Approach*. World Scientific, 2004. ISBN: 9789812382900.

[5] Victoria J. Hodge and Jim Austin. "A Survey of Outlier Detection Methodologies". English. In: *Artificial Intelligence Review* 22.2 (2004), pp. 85–126.

[6] M. Markou. "Novelty detection: a review—part 2: neural network based approaches". English. In: *Signal Processing* 83.12 (2003), pp. 2499–2521.

[7] Animesh Patcha and Jung-Min Park. "An overview of anomaly detection techniques: Existing solutions and latest technological trends". English. In: *Computer Networks* 51.12 (2007), pp. 3448–3470.

[8] Buse M. Ozyildirim and Mutlu Avci. "Generalized classifier neural network". English. In: *Neural Networks* 39 (2013), pp. 18–26.

[9] Xueying Jiang et al. "Application of Improved SOM Neural Network in Anomaly Detection". English. In: *Physics Procedia* 33 (2012), pp. 1093–1099.

[10] Lida Rashidi, Sattar Hashemi, and Ali Hamzeh. "Anomaly Detection in Categorical Datasets Using Bayesian Networks". English. In: vol. 7003. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. Chap. 2, pp. 610–619. ISBN: 0302-9743.

[11] Maryamsadat Hejazi and Yashwant P. Singh. "One-class support vector machines approach to anomaly detection". English. In: *Applied Artificial Intelligence* 27.5 (2013), pp. 351–366.

[12] N. Duffield et al. "Rule-Based Anomaly Detection on IP Flows". English. In: 2009, pp. 424–432. ISBN: 0743-166X.

[13] Eamonn Keogh et al. "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases". English. In: *Knowledge and Information Systems* 3.3 (2001), pp. 263–286.

[14] *autocorrelation*. English. 2014.

[15] Jon Kleinberg et al. *Algorithm design*. English. First;International; Harlow, Essex: Pearson, 2014;2013; ISBN: 1292037040;9781292037042;9781292023946;1292023945;

A

Appendix A

The normal distribution $N(\mu, \sigma^2)$ is a symmetric distribution that depends on two parameters, the mean μ, where $\mu \in [-\infty, \infty]$ and the standard deviation σ, where $\sigma > 0$. The normal distribution and has probability density function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}.$$

If $X_1, X_2, ..., X_n$ is a sample, drawn or believed to have been drawn from a normal distribution $N(\mu, \sigma^2)$, the parameters μ and σ can be estimated from the sample by

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

$$\hat{\sigma} = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \hat{\mu})^2$$

Now $\hat{\mu}$ and $\hat{\sigma}$ are random variables. The estimated mean $\hat{\mu}$ follows the t-distribution as $\frac{\hat{\mu}-\mu}{s_{\hat{\mu}}} \sim t_{n-1}$ where $s_{\hat{\mu}}$ is the estimated variance of $\hat{\mu}$. A $100(1-\alpha)$ confidence interval of μ is thus $\hat{\mu} \pm t_{n-1}(\alpha/2) \cdot s_{\hat{\mu}}$. Similarly the estimated variance, S^2, follows the χ^2 distribution as $\frac{(n-1)S^2}{\sigma^2} \sim \chi^2_{n-1}$. This gives an exact $100(1-\alpha)$ confidence interval of σ^2 as $\left[\frac{(n-1)s^2}{\chi^2_{n-1}(\alpha/2)}, \frac{(n-1)s^2}{\chi^2_{n-1}(1-\alpha/2)}\right]$.
The χ^2 distribution has probability density function

$$f(x) = \frac{1}{2^{\frac{k}{2}}\Gamma\left(\frac{k}{2}\right)} x^{\frac{k}{2}-1} e^{-\frac{x}{2}},$$

where k is a positive natural number usually referred to as the degrees of freedom.

B

Appendix B

Algorithms and their time complexities are evaluated and compared using asymptotic upper bounds for the worst-case time complexity. Let $T(n)$ be a function of the worst case running time of an algorithm with input size n. If $T(n) \leq c \cdot f(n)$ is true for all $n \geq n_0$ where $n_0 > 0$ and $c > 0$ is a constant, then $T(n)$ is said to be $O(f(n))$. In other words $T(n)$ is asymptotically upper bounded by f. The time complexities of algorithms can be classified using this notation. [15] For example, linear algorithms are algorithms with $T(n) = O(n)$, quadratic $T(n) = O(n^2)$ etc. Table B.1 gives an idea of the running time for different $f(n)$ and n.

Table B.1: The running times of algorithms with different time complexities with $c = 1$ for a processor performing a million instructions per second. The times are rounded upwards.[15]

	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	36 years	$> 10^{25}$ years
$n = 10^3$	< 1 sec	< 1 sec	1 sec	18 min	10^{17} years	$> 10^{25}$ years
$n = 10^4$	< 1 sec	< 1 sec	2 min	12 days	$> 10^{25}$ years	$> 10^{25}$ years
$n = 10^6$	1 sec	20 sec	12 days	10^4 years	$> 10^{25}$ years	$> 10^{25}$ years

C

Appendix C

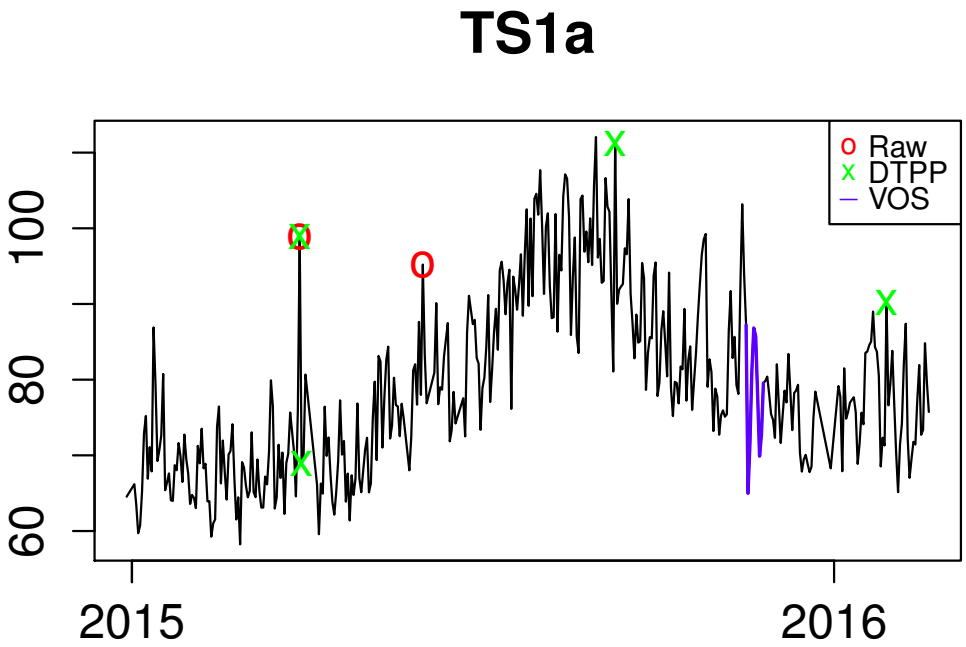

Figure C.1: Detected anomalies for time series 1.a with time frame for aggregation of 1 day.

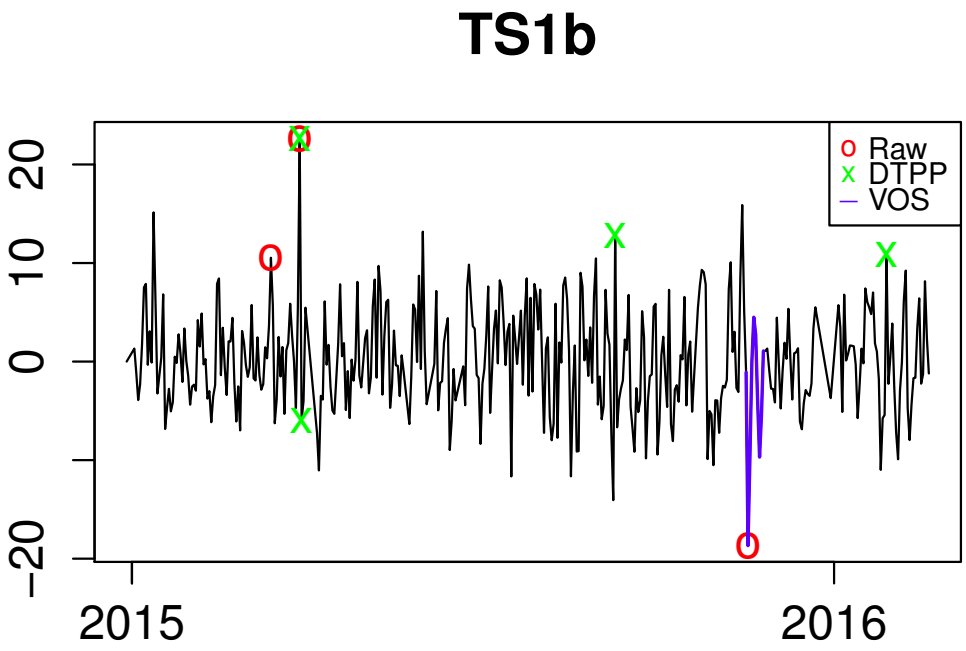

Figure C.2: Detected anomalies for time series 1.b with time frame for aggregation of 1 day

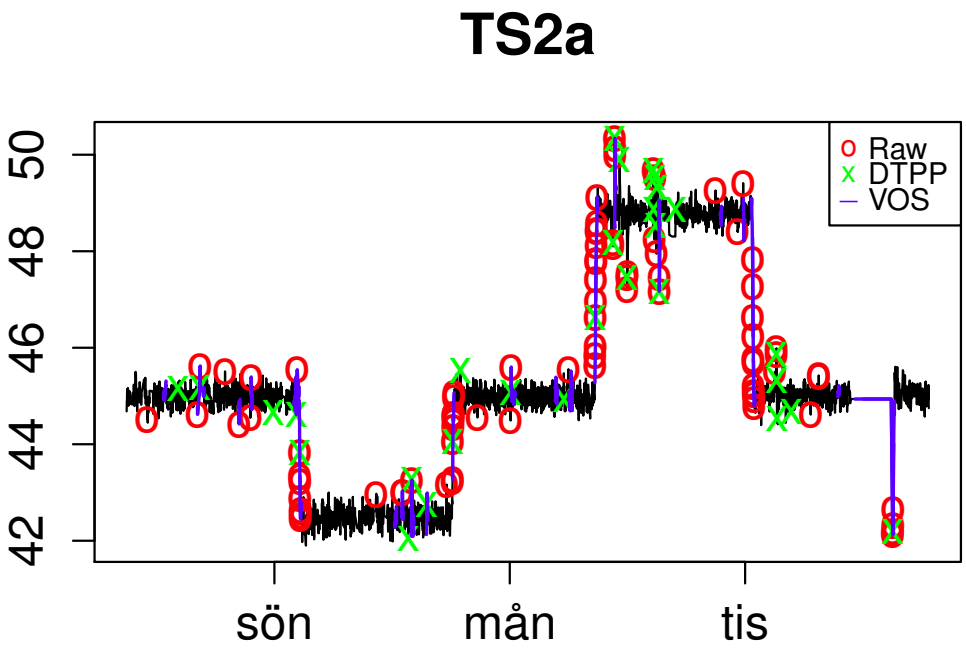

Figure C.3: Detected anomalies for time series 2.a with time frame for aggregation of 1 minute.

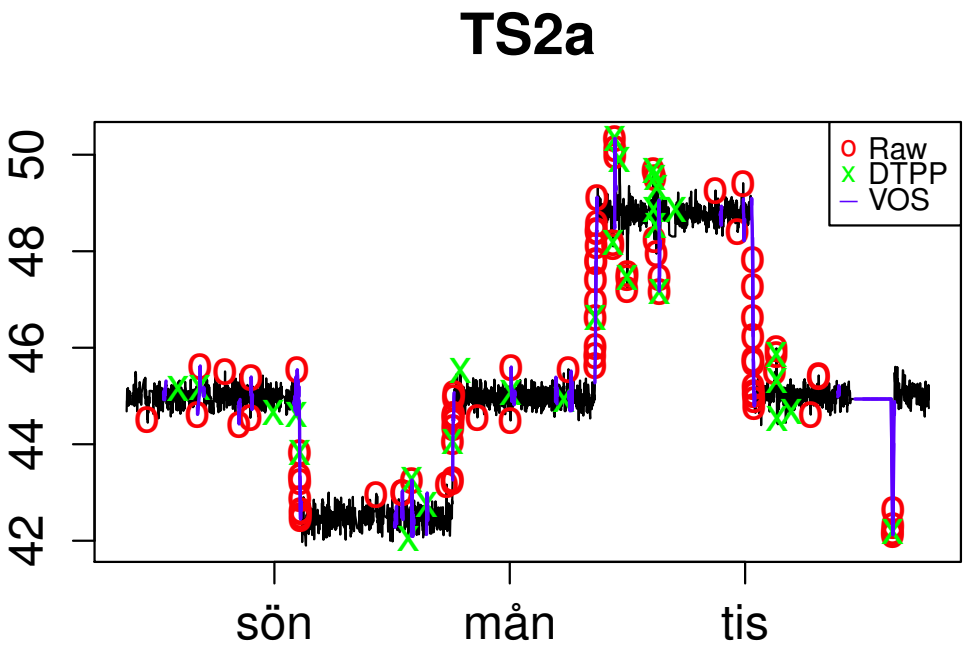

Figure C.4: Detected anomalies for time series 2.b with time frame for aggregation of 1 minute.

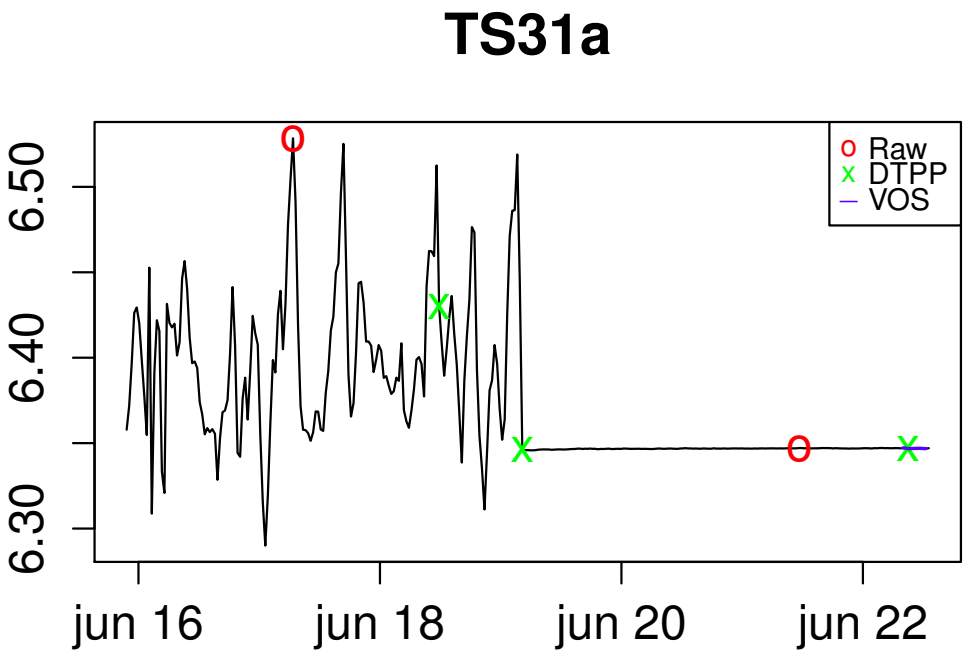

Figure C.5: Detected anomalies for time series 3.1.a with time frame for aggregation of 30 minutes.

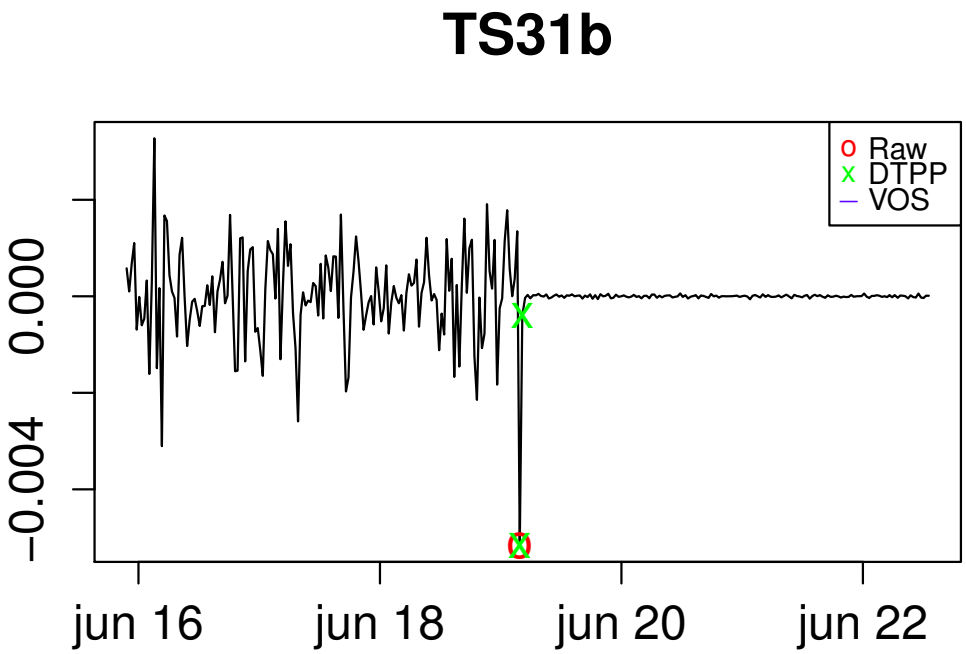

Figure C.6: Detected anomalies for time series 3.1.b with time frame for aggregation of 30 minutes.

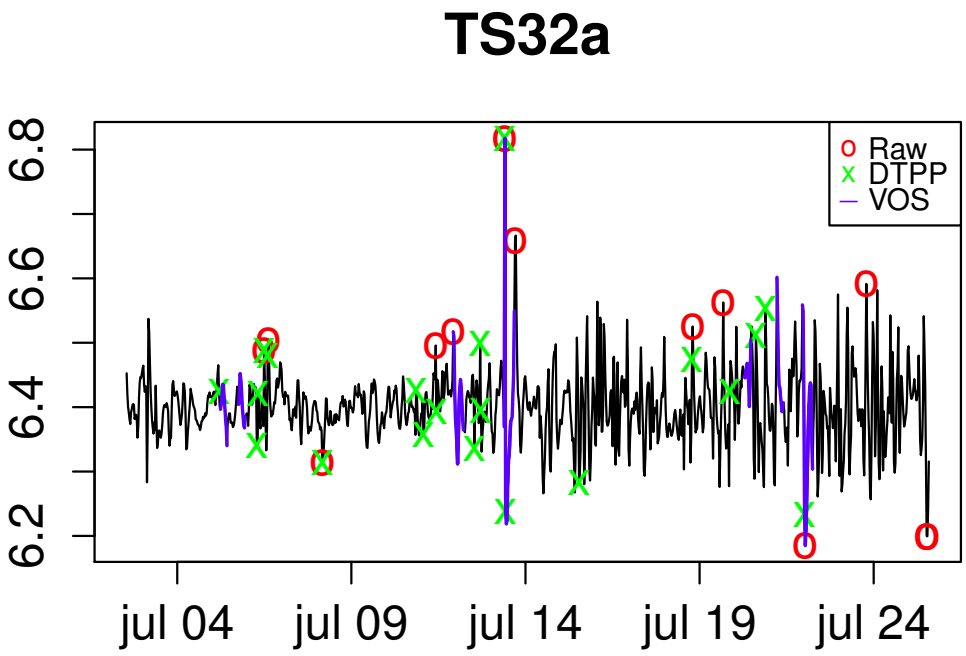

Figure C.7: Detected anomalies for time series 3.2.a with time frame for aggregation of 30 minutes.

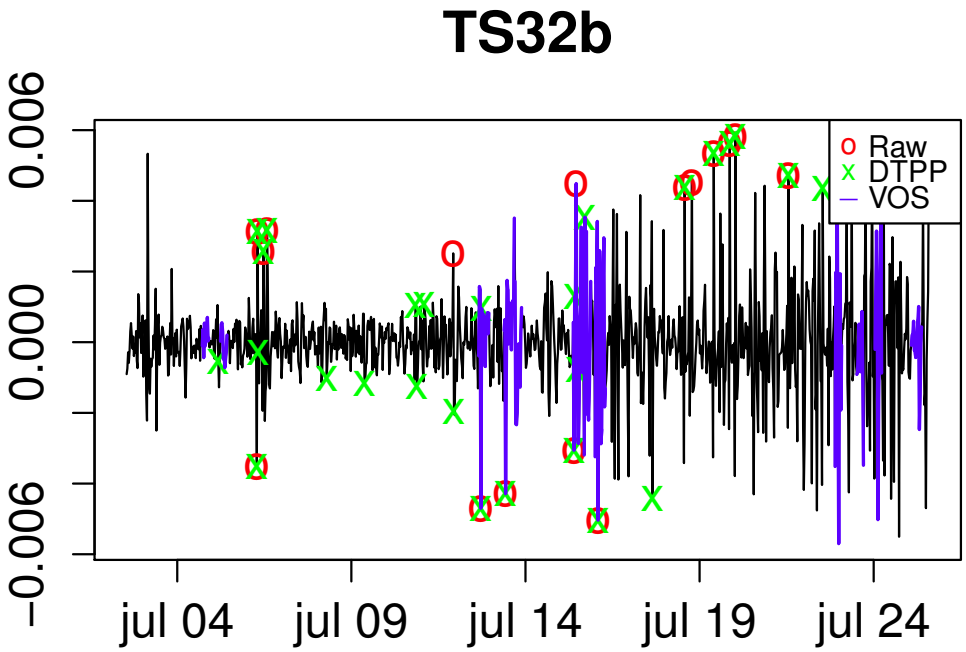

Figure C.8: Detected anomalies for time series 3.2.b with time frame for aggregation of 30 minutes.

D

Appendix D

This appendix gives figures that shows the number of detected anomalies as a function of k and p for time series 1 and 3.

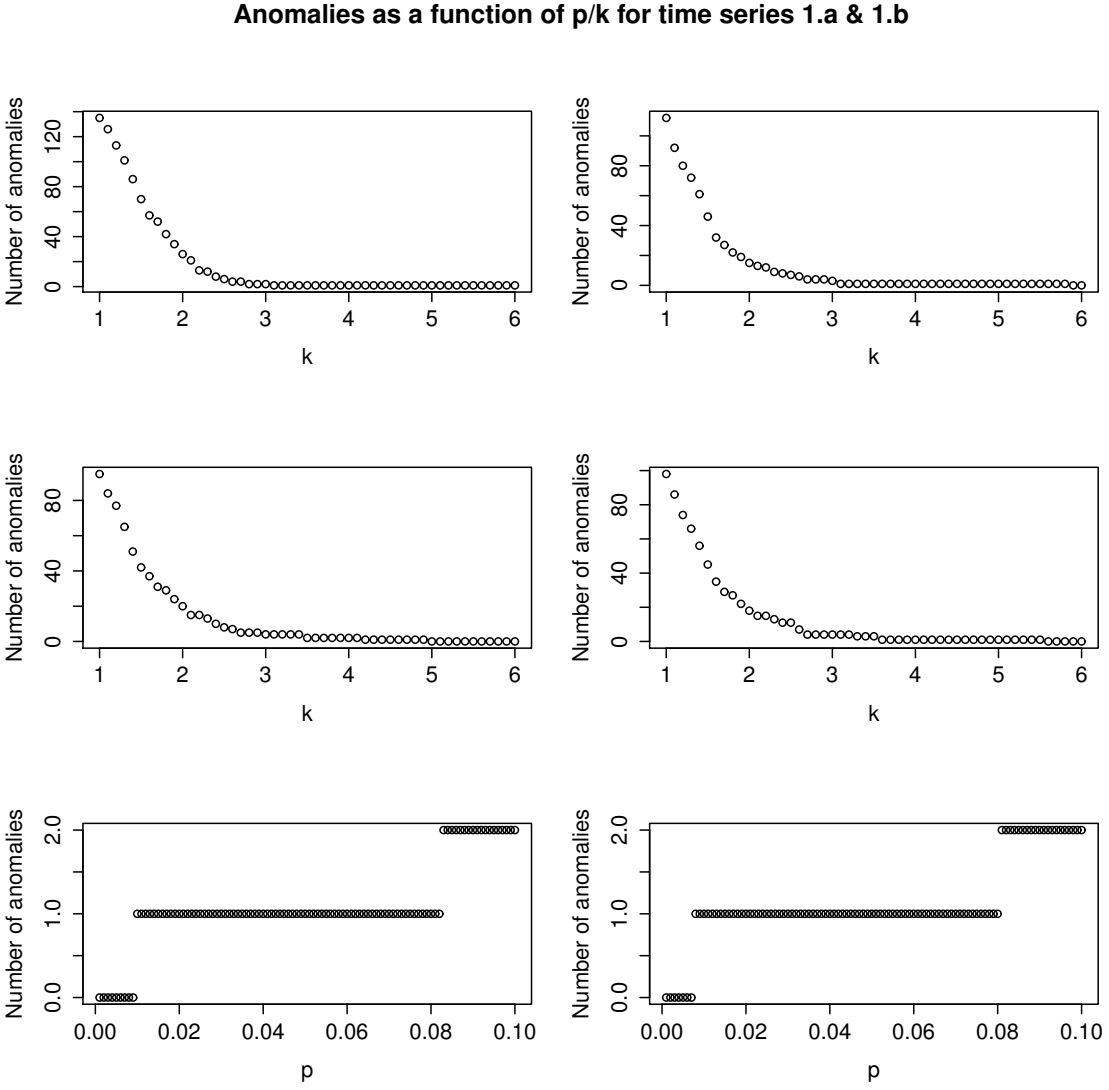

Figure D.1: The number of anomalies detected from raw values (top), DTPP (middle) and VOS (bottom) as a function of parameter k that influences the control limits for time series 1.a (left) and 1.b (right). The time series is aggregated over 1 day, i.e. the raw values are used.

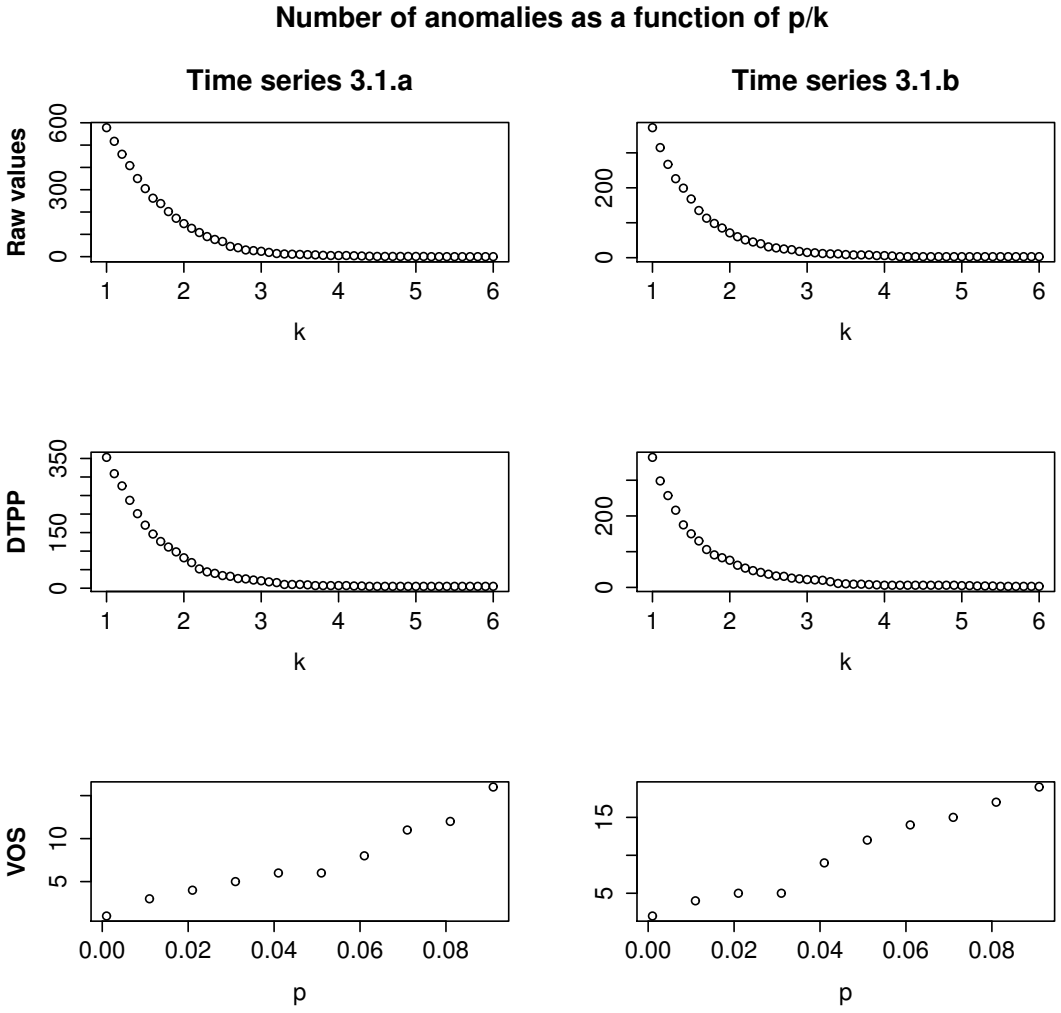

Figure D.2: The number of anomalies detected from raw values (top), DTPP (middle) and VOS (bottom) as a function of parameter k that influences the control limits for time series 3.1.a (left) and 3.1.b (right). The time series is aggregated over 1 hour.

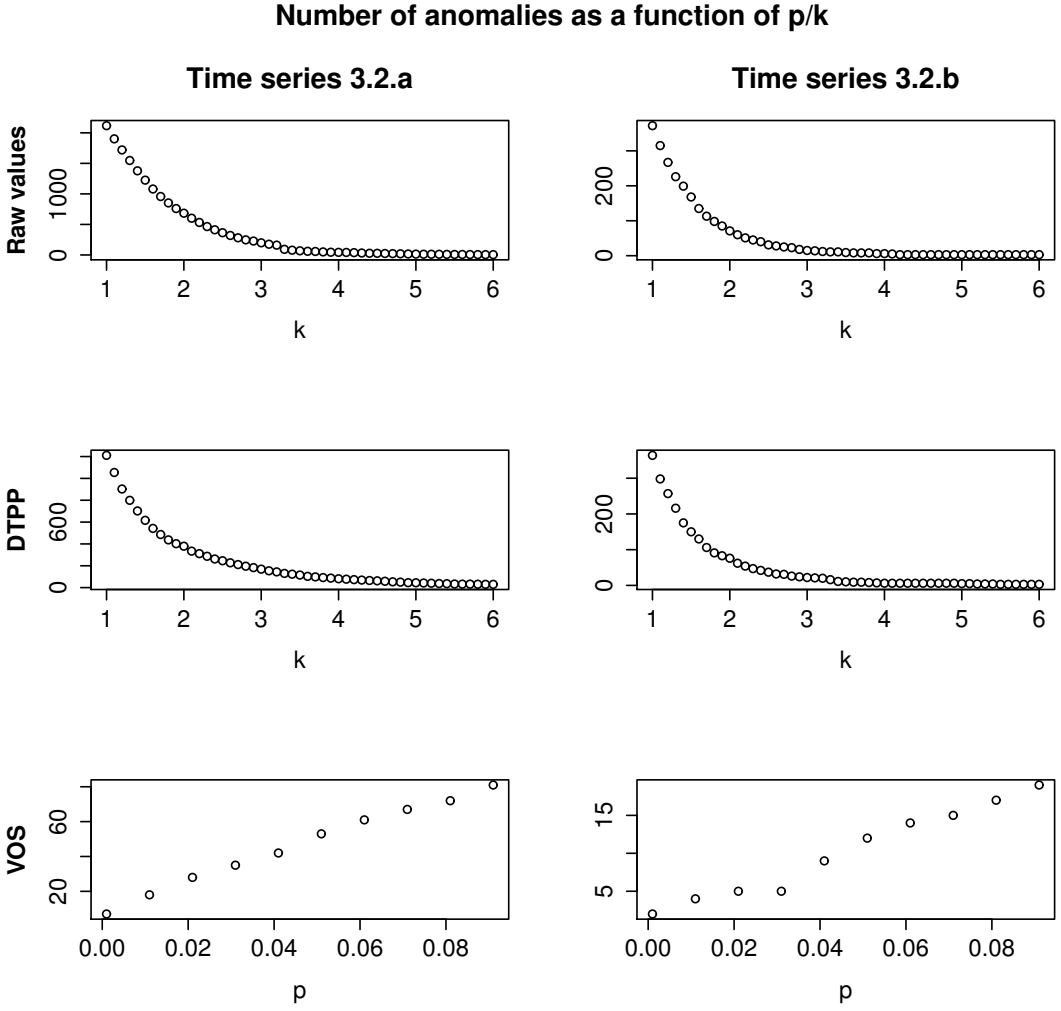

Figure D.3: The number of anomalies detected from raw values (top), DTPP (middle) and VOS (bottom) as a function of parameter k that influences the control limits for time series 3.2.a (left) and 3.2.b (right). The time series is aggregated over 1 hour.

E

Appendix E

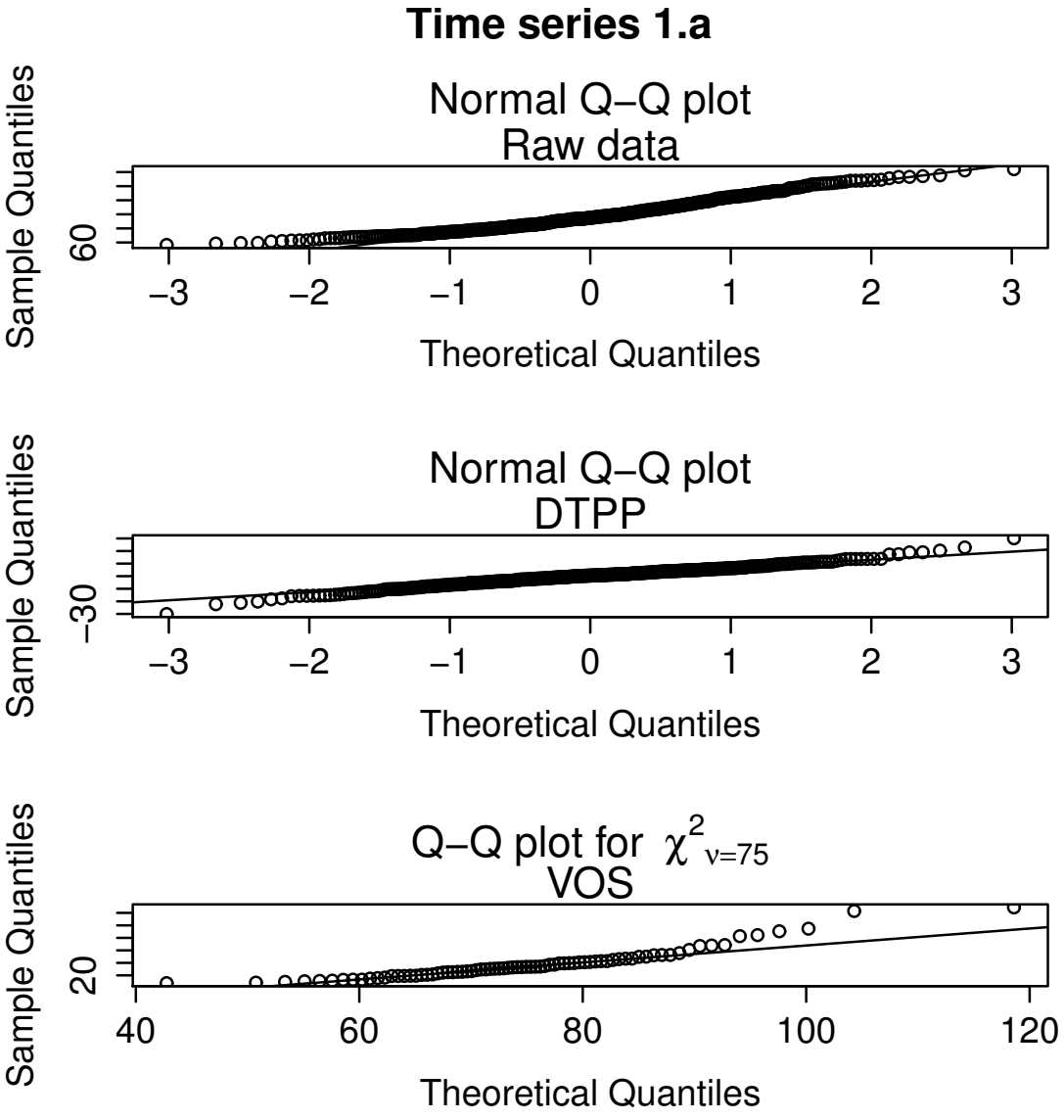

Figure E.1: Q-Q plot of time series 1.a

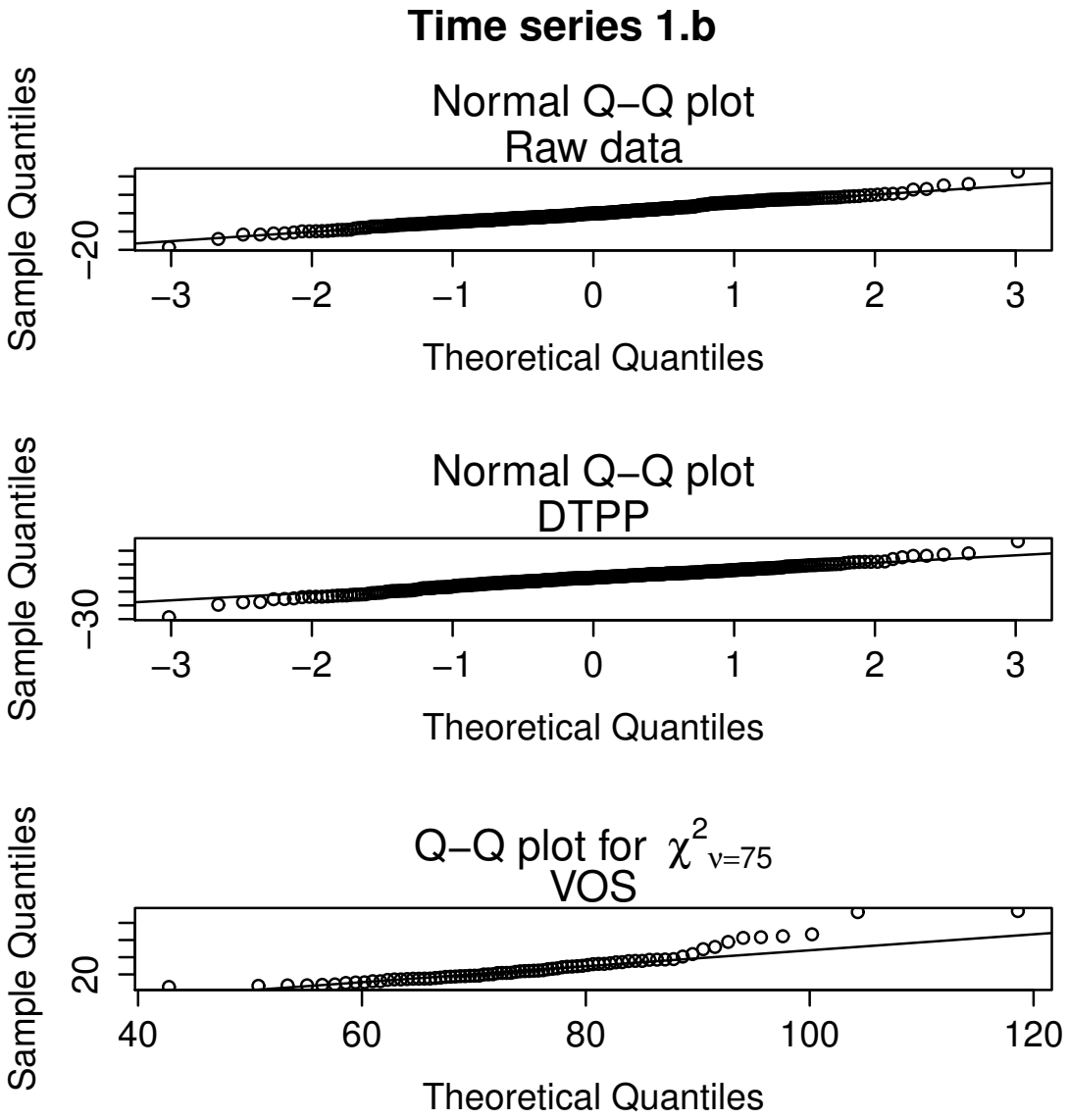

Figure E.2: Q-Q plot of time series 1.b

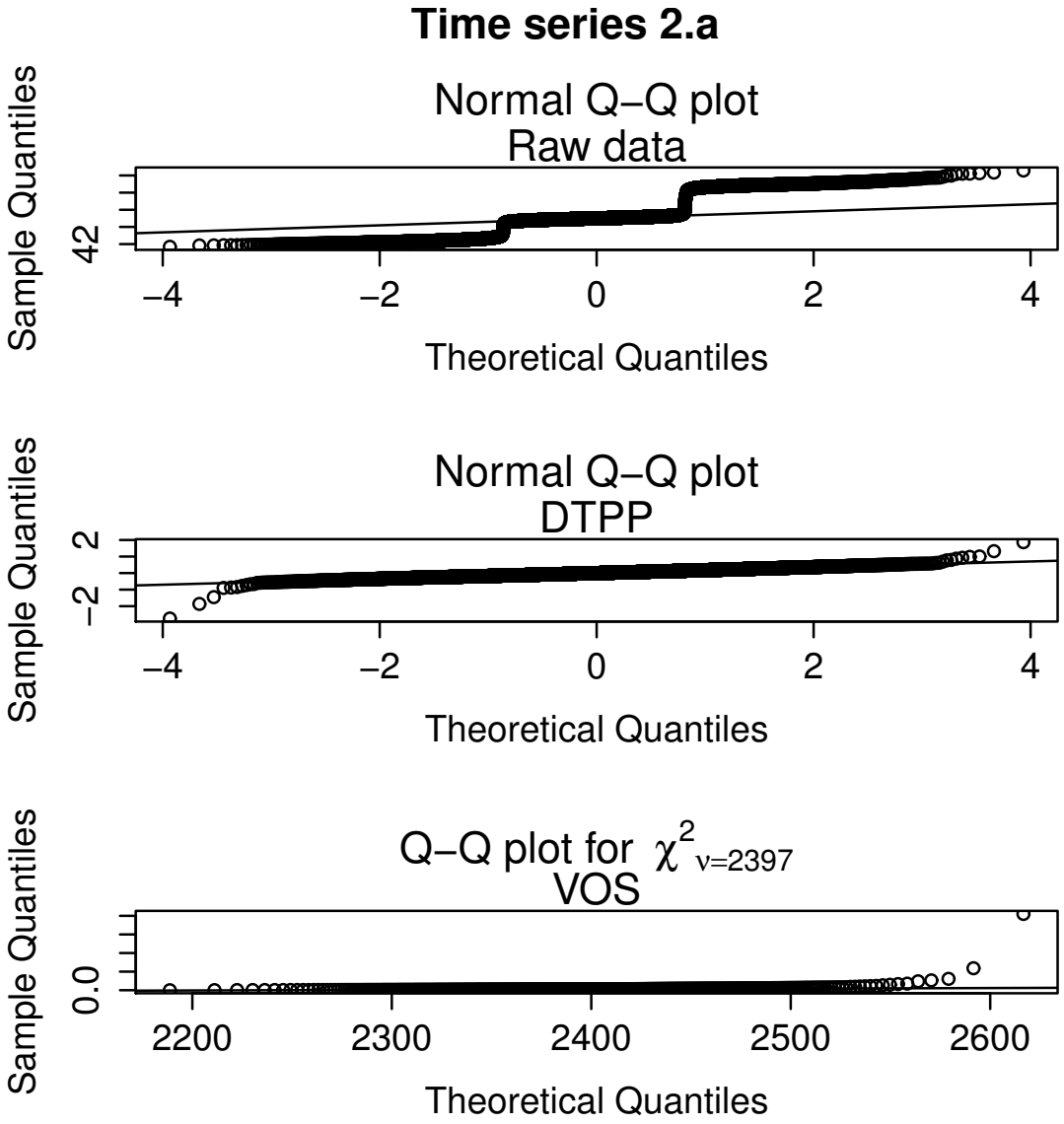

Figure E.3: Q-Q plot of time series 2.a

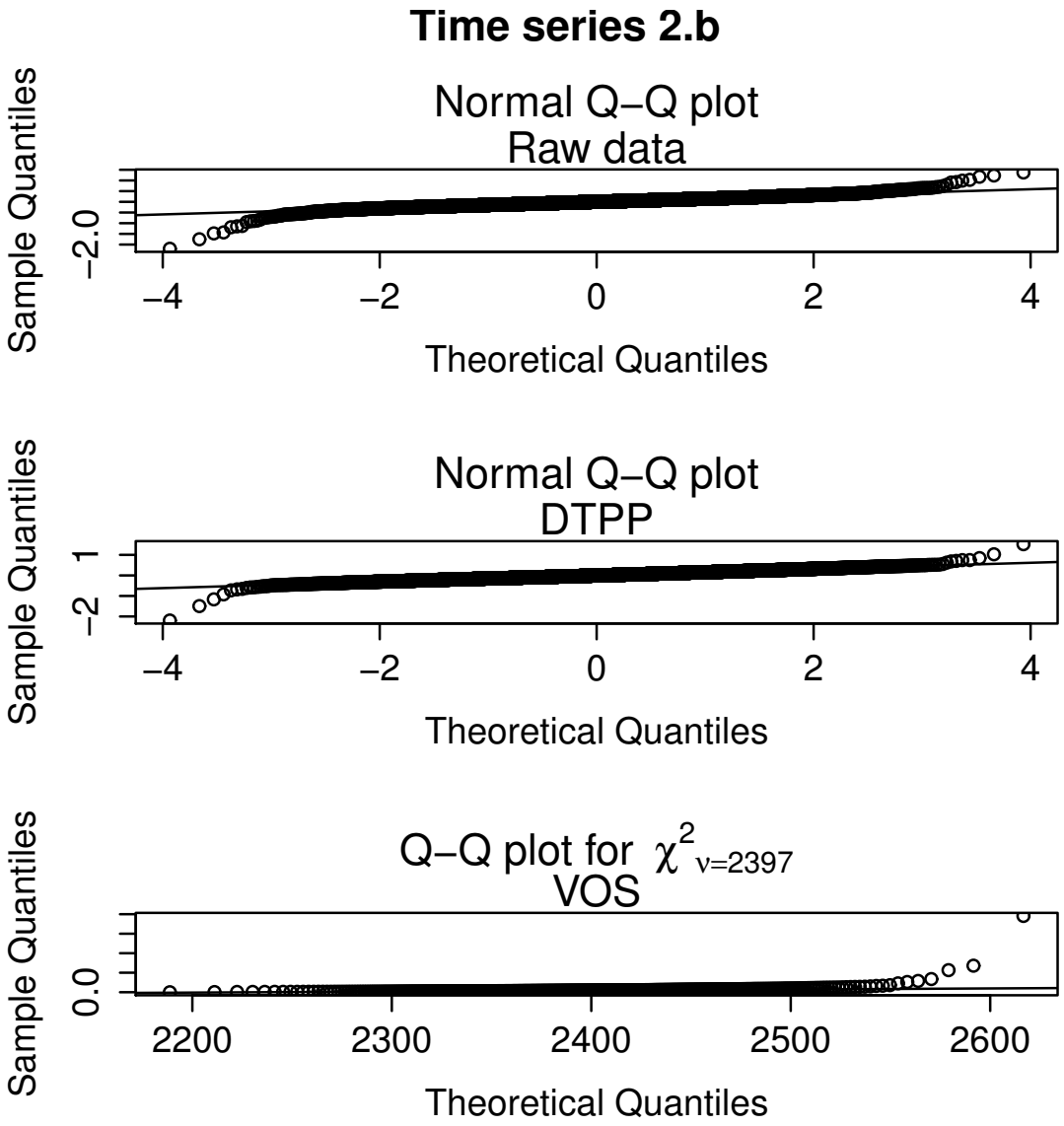

Figure E.4: Q-Q plot of time series 2.b

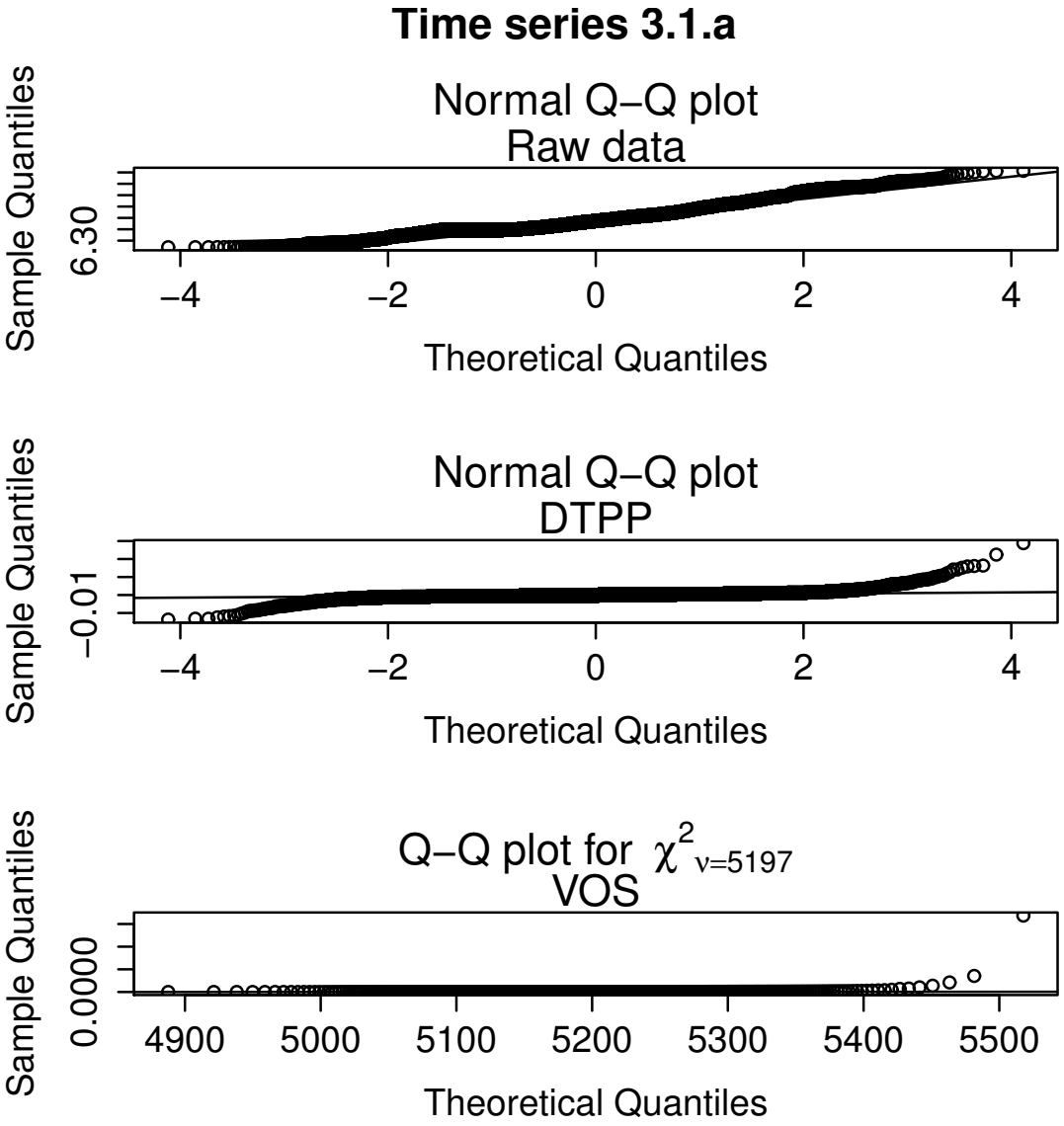

Figure E.5: Q-Q plot of time series 3.1.a

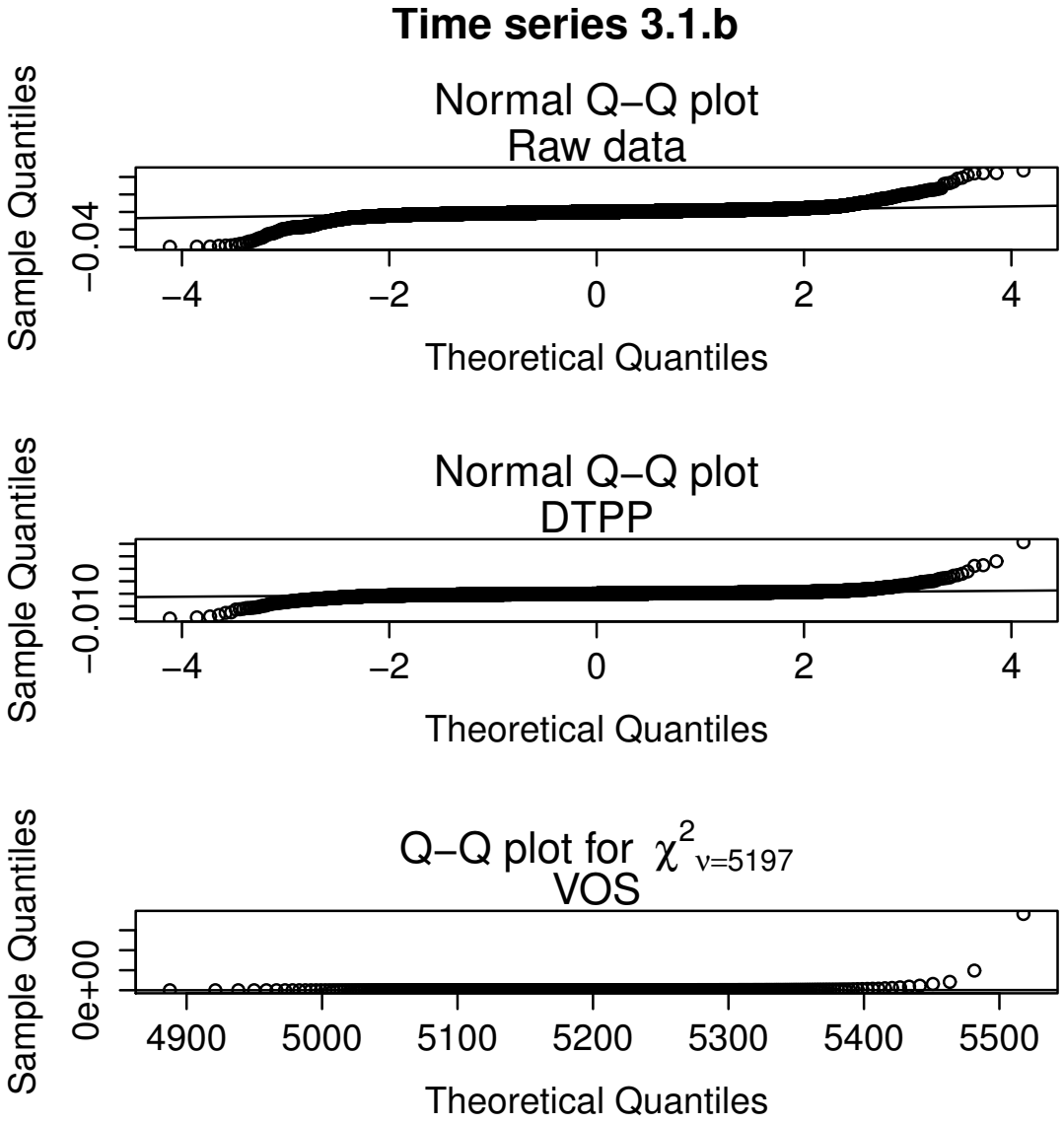

Figure E.6: Q-Q plot of time series 3.1.b

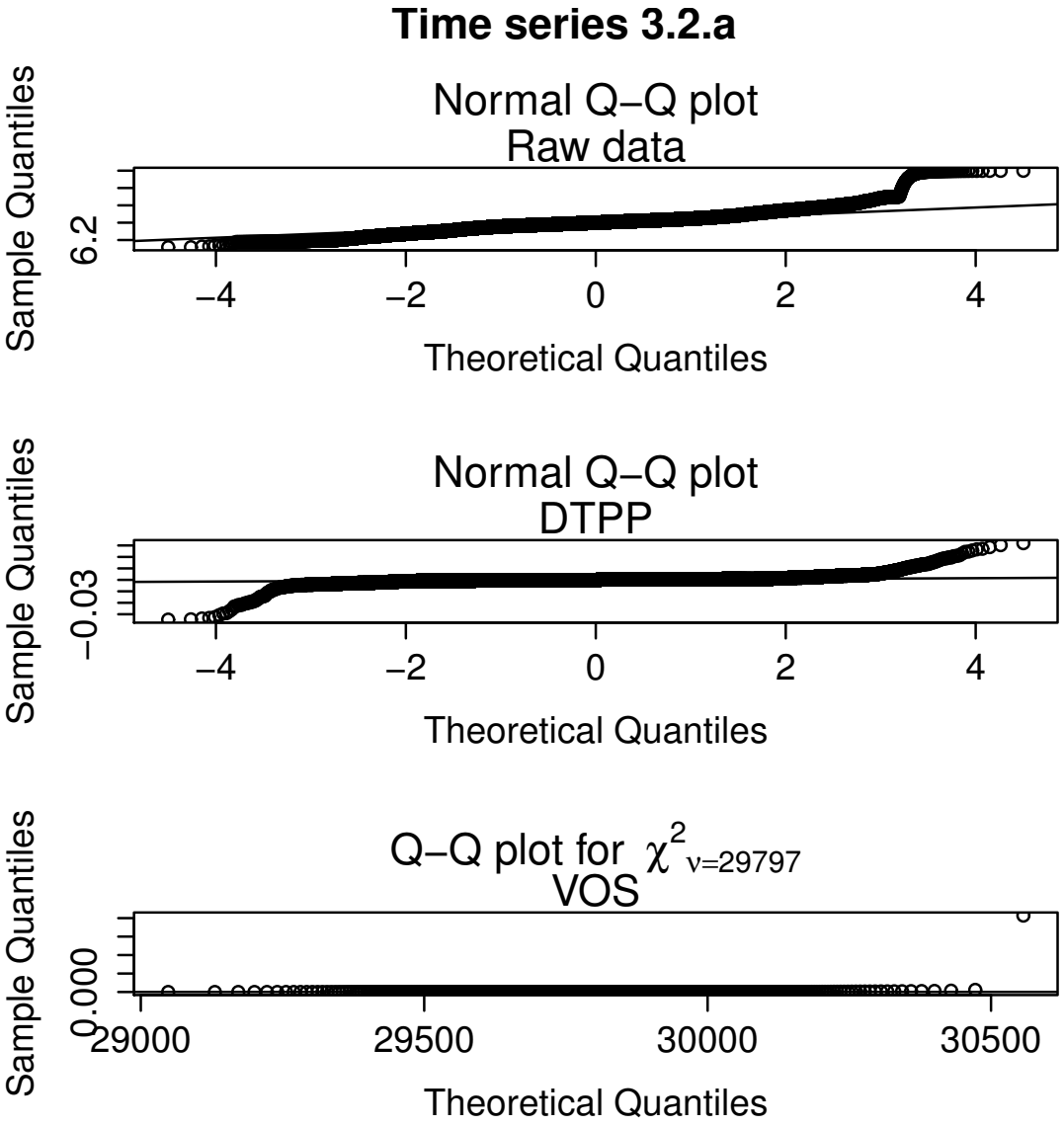

Figure E.7: Q-Q plot of time series 3.2.a

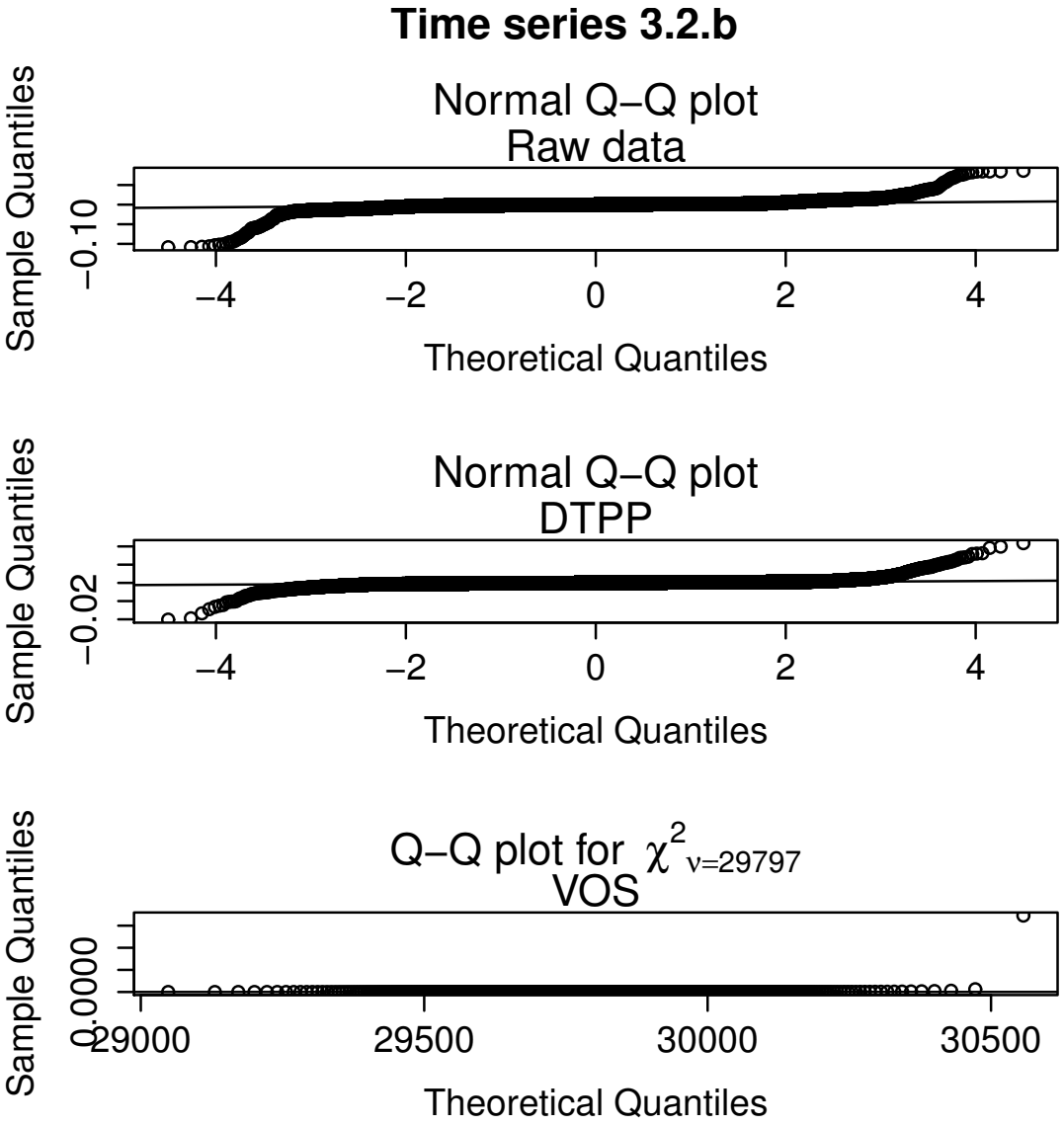

Figure E.8: Q-Q plot of time series 3.2.b