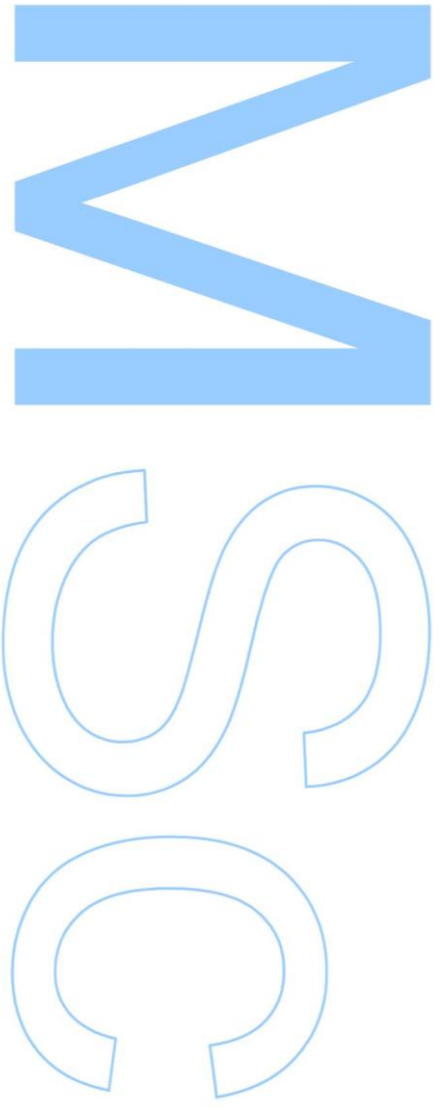
Data Mining for Sensor Intelligence: Change Point Detection and Clustering

Fábio Henrique da Silva Pereira
Mestrado em Ciência de Dados
Departamento de Ciência de Computadores
2020

Orientador
Alípio Mário Guedes Jorge, Professor Associado, Faculdade de Ciências da Universidade do Porto

Coorientador
Inês de Castro Dutra, Professor Auxiliar, Faculdade de Ciências da Universidade do Porto



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Abstract

In a growing world of sensors, the importance of reducing their failures keeps increasing, particularly in sensors such as fire detection ones, in which a failure may provoke a useless mobilization of firefighting teams or, in a worst case scenario, not trigger an alarm when there is a fire, possibly resulting in many deaths.

To better understand the nature of fire sensor data and to further the cause of reducing their failures, we experiment with a new approach: divide the time series captured by the sensors in segments with different distributions and then proceed to cluster them through a shape similarity distance function.

This approach gave us a new way of thinking about the nature of the time series and to better understand what may cause the different shapes. Through that approach, we were able to identify multiple predominant shapes present in the data, to describe them and understand their nature, and to look at the characteristics of each cluster, in order to understand why the clusters exist in the first place.

Resumo

Num mundo em que há cada vez mais sensores, a importância de reduzir as falhas deles também aumenta constantemente, particularmente nos sensores de deteção de incêndio, em que uma falha pode significar a mobilização inútil de equipas de bombeiros, ou no pior dos casos, não ser detetado um incêndio, possivelmente causando várias mortes.

Para perceber melhor a natureza dos dados de sensores de incêndio e avançar a causa de redução das falhas deles, experimentamos uma nova abordagem: dividir as séries temporais capturadas pelos sensores em segmentos com diferentes distribuições e depois proceder ao *clustering* desses segmentos através de uma função de semelhança de formas que esses segmentos tomam.

Esta abordagem dá-nos uma nova forma de pensar sobre a natureza das séries temporais e a dá-nos a possibilidade de perceber melhor o que pode causar essas diferentes formas presentes nas séries. Através dela, conseguimos obter várias formas predominantes nos dados, descrevê-las e perceber a sua natureza, assim como olhar para as diferentes caraterísticas dos *clusters*, de forma a perceber porque é que esses clusters existem.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In a growing technological world [2, 3], our lives are increasingly dependent on technology. This technology doesn't include only the ones we use directly, like our cellphones or laptops. It also includes a much bigger world of devices that, without our noticing, make our lives easier and more secure – sensors. Sensors are devices built to capture data and transmit it (in real-time or not), so that a user or some other device can make better informed decisions [5].

Every sensor, as any electronic device, is prone to failures. Those failures may result in a misreading of the data, making the data unreliable for the very decisions it's supposed to help make. Unfortunately, making a decision based on false data can have a disastrous impact [4]. Take the example of fire detection sensors: a misreading of the data can lead to the useless mobilization of firefighting teams by detecting a non-existent fire or in the worst case scenario, many deaths, by not detecting an occurring fire.

From that, the problem of reducing such failures arises, as one needs to identify them to avoid making any ill-informed decisions. The detection of devices' failures falls into the category of Anomaly Detection (AD), a broader category that, in simple terms, detects abnormalities in data.

There is plenty of sensor data available that may be used in this work. Unfortunately, most of that data is independent of alarms. I.e., the data has no identification of when an alarm was triggered. Even if we choose a source that only represents data in which

an alarm was triggered, we still would not know which of those alarms were false or true alarms.

Despite that, and still in the pursuit of false alarm reduction, we will try to understand if there are recurrent patterns in the data, and try to gather valuable knowledge about these patterns.

With that goal in mind, we will study regimes in the sensor data. A regime is a contiguous sequence in a time series that follows the same distribution of values. A sequence is considered to be a new regime when it deviates from the distribution of values that come before it. Then, we will cluster the regimes in order to understand whether the regimes have well defined clusters in terms of shapes and what predominant shapes these clusters have. Finally, we will study these clusters to try to gather valuable information about their regimes and, subsequently, about the nature of the data. Finding such clusters would be of uttermost interest and would provide us with valuable knowledge about the data, because then, we would have evidence that the data is not homogeneously distributed but rather there is some type of variables in the originating mechanism that dictates the shapes that the data takes.

1.2 Objectives

We will try to answer the following set of questions:

- Can we verify the existence of regimes in any important variable of our data?
- Are there clusters of shapes in these regimes?
- Can we describe the shapes of the clusters of regimes?
- Do shapes have a correlation with the dates that the regimes were recorded?
- Is there some interesting pattern on what regimes come before a regime from a certain cluster? And after?

In order to assess the veracity of the following hypotheses:

1. Regimes exist,
2. Regimes have well defined clusters of predominant shapes,
3. The clusters and their shapes can provide valuable information about the data.

1.3 Thesis' layout

In this section, we summarize the layout of this thesis. In Chapter 2, an introduction to the theoretical foundations needed to understand the rest of this work is given. Next, in Chapter 3, we give the stage to works done by other authors that relate to this paper, from the general goal to the specific tasks that will be done. Chapter 4 focuses on the data and its transformation (from time series to regimes) in order to proceed to clustering in Chapter 5. Finally, in Chapter 6, we give conclusions and final remarks about this thesis, as well as future work that might be done to improve it.

Chapter 2

Theoretical foundations

In this chapter, the reader will be introduced to key concepts and methods that are going to be used throughout this work. As we have described in chapter 1, our aim is to understand and characterize sensor signals. Changes of behavior, either permanent or temporary, are often an indication of sensor failure or of the detection of a relevant event. In the first section, Change Point Detection (CPD), a technique that allows us to identify segments of the time series that may correspond to such behavior changes or different regimes. In the second and last section, we introduce clustering, and more precisely, clustering in time series, that will serve the purpose of understanding how and why the regimes' shapes cluster together and analyze the clusters' characteristics.

2.1 Change Point Detection

Change point detection (CPD) is a type of procedure that aims to find whether and where in a time series the data generating model changes, i.e., it aims to find different states in the data. In this section and work, we will only address and use offline CPD.

A change point is a transition from one state to another. A state, or as we refer to it in this work, a regime, is a time period in which the nature of the recording device has some specific characteristics that don't change during that period of time. Those characteristics are not particular to the device itself, but rather to anything that might have an impact on the measures of the device, such as the environment. For instance, a thermometer will record low temperatures in the winter and record high temperatures in the summer. These would be considered two different states in

the underlying mechanism (environment) that generates the temperature data.

Note that CPD might be applied to a complex system composed by multiple devices, in which case the change point detection would refer to the system as a whole and not in each device in separate.

The most common types of changes searched for in CPD are the following:

- Mean change,
- Variance change,
- And co-variance change.

Any data characteristic that is able to change might influence a new state of the underlying model.

Formally, CPD is all about finding the best possible segmentation τ of the data that minimizes a criterion $V(\tau, y)$. This criterion encodes the sum of costs that measure the goodness of fit of each segment to a specific model. This means that CPD will try to find segments in which the data is consistent in a distribution-wise perspective, in accordance to the cost function. More precisely, the criterion is defined as:

$$V(\tau, y) = \sum_{k=0}^{K} c(y_{t_k..t_{k+1}})$$

2.1.1 Types of CPD problems

There are two types of problems in CPD[41]:

- Type 1: known number of changes. This problem has a fixed number of changes that the CPD method must find and so, the criterion is restrained to that fixed number of changes:

$$\underset{|\tau|=K}{min} V(\tau)\ ^{1}$$

- Type 2: unknown number of changes. In this type of problem, the criterion must have a penalization factor that increases with the number of segments:

$$\underset{\tau}{min} V(\tau) + pen(\tau)$$

[1] $V(\tau) \Leftrightarrow V(\tau, y)$

2.1.2 CPD methods composition

There are three components that constitute a CPD method:

- Cost function – measures how heterogeneous a segment is. I.e., the cost is expected to be low if the data in the segment is homogeneous, and high if not. The cost function encodes what types of changes will be detected.
- Search method – responsible for iterating segments until it finds the best set of them.
- Constraint – responsible for maintaining coherence between the real number of changes and the number of changes detected, or in the case of a type 2 problem, of limiting the number of changes to a reasonable value.

2.1.2.1 Cost functions

The cost function, as already stated above, encode what types of changes will be detected – mean, variance, or other data characteristics.

Cost functions are divided in two groups:

- Parametric models – these are cost functions that make assumptions about the data distribution.
- Non-parametric models – in contrast to parametric models, non-parametric are close to being assumption-free (the only assumption is that the distributions are continuous).

In Figure 2.1, we show an account of the most popular cost functions.

2.1.2.2 Search methods

The search method is responsible for solving the CPD optimization problem. Search methods are also divided in two main groups:

- Optimal detection – finds the exact solution. Optimal detection methods include *Opt* [11], which aims to solve type 1 problems and *Pelt* (Pruned Exact Linear

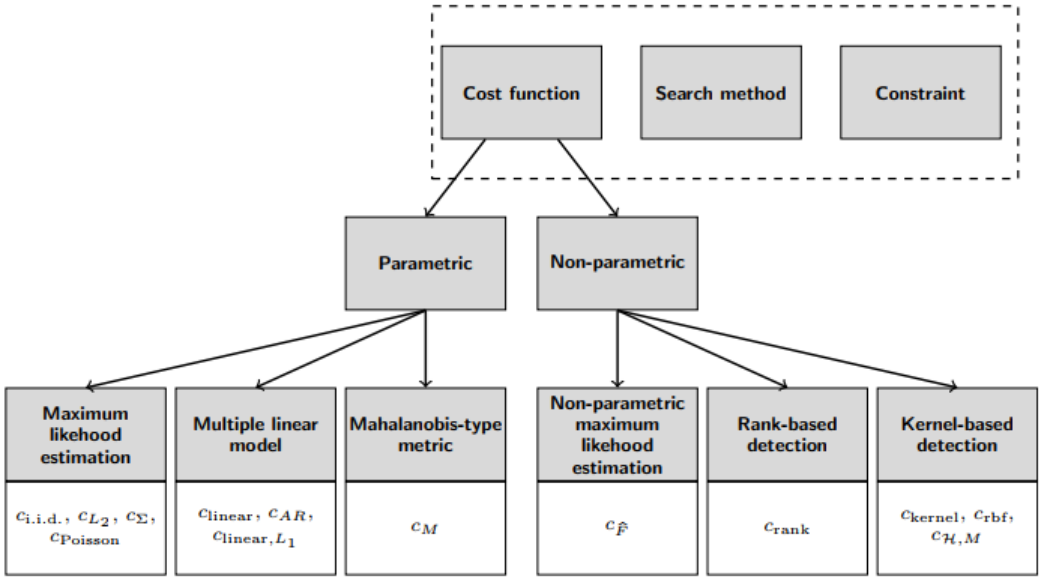


Figure 2.1: Most popular cost functions. Image taken from [41].

Time) [25] which aims to solve type 2 problems. These methods have a high computational complexity.

- Approximate detection – finds the approximate solution. These are used when optimal detection methods are too computationally expensive for the task at hand. Examples of these methods are window-based methods, binary segmentation and bottom-up segmentation. These methods, of finding an approximation of the optimal solution, are not as accurate as the optimal methods, despite being faster.

2.1.2.3 Constraint and penalty

Finally, there is also the need for a constraint in type 1 problems, which will make the CPD method find a fixed number of regimes. For type 2 problems, since we don't know for certain how many regimes there are, we use a penalty rather than a constraint. The main idea behind this is: the higher the number of segments, the higher the penalty value will be. This penalty mechanism will result in a balance between the goodness of fit of the partitions and the number of total partitions.

The most popular penalties are linear ones [25], which means that the higher the number of segments, the higher the penalty, in a linear fashion.

For a more in-depth study about penalties and CPD methods in general, the reader

should refer to surveys on the matter, such as [41, 34, 8].

2.1.3 Assumption of normality

For some techniques, such as parametric cost functions in CPD, there are some requirements that the data must meet. In the CPD case, we are interested in whether the data follows a normal distribution [41]. In an ideal setting, we would know what distribution the data comes from. Unfortunately, most of real world problems are not ideal and that means that we have to make assumptions about the data distribution.

To assess if a data set follows a normal distribution, one can resort to visualization methods, such as QQ plot, box plot, normal probability plot and simple histograms.

One can also make use of statistical tests, such as Chi-Squared test, Kolmogorov-Smirnov Goodness of Fit test [23], and many more.

2.2 Clustering

Clustering, a type of unsupervised learning, serves the ultimate purpose of discovering hidden data structures. This is done by partitioning data into groups in which its instances are similar, and then study what the instances from each group have in common.

There are multiple types of clustering algorithms:

- Hierarchical,
- Squared-error based,
- Mixture Densities based,
- Graph Theory based,
- Fuzzy,
- And many others.

The most important concept in clustering, used in most of the algorithms, is the similarity between instances. The similarity measures how identical two instances are,

according to some function, such as the Euclidean distance, Minkowski distance, and others.

Then, after having a matrix of the distances between the instances, a clustering algorithm uses that matrix to discover the clusters, by grouping instances that are close together. Examples of such algorithms are DBSCAN [16], OPTICS [9], k-Medoids [36]. There are also other clustering algorithms that do not work with a distance matrix, such as KMeans [29]. For more in this subject, refer to [43].

After grouping the data into clusters, it might also be useful to derive prototypes – i.e., instances that adequately represent each one of the clusters.

In the next section, we see how clustering time series differs from the typical clustering.

2.2.1 Clustering time series

One of the main problems that arise in time series clustering is the incompatibility of the usual distance functions with time series. Distance functions such as the Euclidean distance and others can not be used in a straightforward fashion, as in many cases it would be impossible to apply such functions (because time series may have different lengths, for instance). Even if it was possible to apply such distance functions, they would have unwarranted consequences in the resulting clusters.

First, one must consider whether all the time series are equal in length. If not, an elastic distance measure is required. An elastic distance measure is one that works on time series with different lengths. The main idea behind elastic distance measures is that it contracts (or expands) a time series in order to compare it to another time series of different length.

Then, the choice of the distance function and/or model will depend on what characteristics of the time series we are most interested in:

- Similarity in time – addresses correlation between time series. Here, classic distances such as Euclidean (or some variant of it) are used.
- Similarity in shape – takes into account the shapes present in two time series, regardless of the time at which a particular shape happens. In this case, a distance measure of elastic nature is of extreme importance; it consider only the shapes and not to the time step at which they happen.

- Similarity in structure – this type is identical to similarity in shape, but for longer time series. In this case, one would use a Hidden Markov Model or an ARMA process, and only then measure the distance between the parameters of the models, rather than the time series themselves.

In subsection 2.2.1.1, we briefly expand on the distance measure that will be used throughout this work.

In what relates to prototypes, there are multiple ways to get them [6]:

- The medoid sequence of the set, where the prototype of a cluster is defined by the sequence that minimizes the sum of squared distances to other objects within the cluster.
- The average sequence of the set, where a simple averaging of the time series of the cluster is done to obtain its prototype. Unfortunately, in the cases of time series with different lengths or when the similarity between sequences is based on their shapes, a simple averaging will not capture the actual average shape of the cluster. For these cases, more complex averaging methods are necessary, but most of the times such methods are avoided in advantage of simpler ones [6].
- The local search prototype, where a combination of the medoid and the averaging method happens.

2.2.1.1 Dynamic Time Warping

From the multiple distance measures existent, Dynamic Time Warping (DTW) [12] is one of the most popular. DTW is an elastic distance measurement that aims to measure the similarity in shape of two time series, regardless of their speed (or the difference thereof).

In figure 2.2, a comparison between Euclidean distance and DTW is shown, so the reader can have a better intuition of how DTW works.

2.2.2 Clustering evaluation

There are multiple metrics to evaluate clusters. From those, two will be used throughout this work: Silhouette score and Dunn's Index.

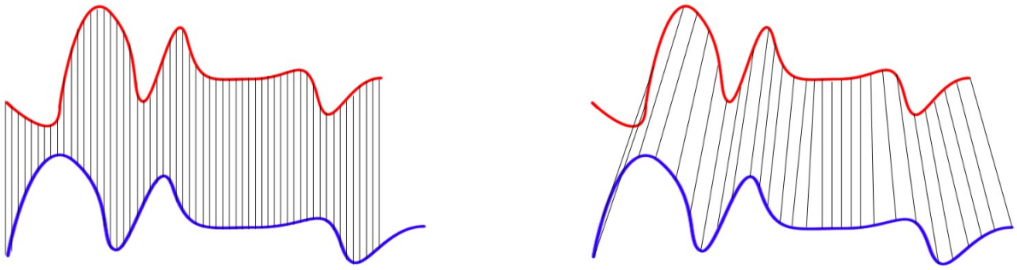


Figure 2.2: On the left, euclidean matching and on the right, DTW matching. Image taken from: [1]

Silhouette score is computed by first calculating the silhouette width for each instance i^{th}, which is a confidence indicator on the membership of the i^{th} instance in the cluster that it was assigned to. This confidence is measured by how easy it would be for this instance to be assigned to the closest cluster other than the one it was actually assigned to. After having each silhouette width, a silhouette score of each cluster is computed, by calculating the mean of the silhouette widths. Finally, we can get the global silhouette score by getting the mean of each cluster's silhouette score. Silhouette scores have a domain of $[-1, 1]$, where a value of 1 would mean that the instances were correctly assigned to the respective clusters; 0 would mean that the instances could be assigned to some other cluster; and -1 would mean that the instances were incorrectly assigned and should be assigned to some other cluster.

Dunn's Index main goal is to have a sense of how much the clusters overlap and how compact the clusters are. It is calculated through the inter-cluster distance and intra-cluster variance. This index takes the $[0, 1]$ domain where higher values mean that there is less overlapping, and therefore a better clustering.

Chapter 3

Related work

In this chapter, works that might be of relevance to this thesis are presented. First, we go through works on anomaly detection in sensor data. Then, we dive into works on CPD, statistical tests for normality assumption, and finally, clustering on time series.

3.1 Anomaly detection in sensor data

Multiple works have been done on this subjects, but many of these are are specific to a single type of sensors. For instance, in Fujimaki et al. [18], the authors develop a system to detect anomalies in spacecrafts using kernel feature space. The authors of Ahn et al. [7] also develop an anomaly detection procedure for spacecraft control systems, but with Deep Generative Models. There are some works that address anomaly detection of time series in an unsupervised setting, without a particular task in mind. Examples of that are Munir et al. [33], in which the authors use deep learning approaches such as Convolutional Neural Networks. Another instance of such work is Munir et al. [32], in which the authors fuse DL with statistical models.

There are also multiple surveys that are related to this matter. In Ball et al. [10], the authors write extensively about DL in remote sensing data, although not exclusively about anomaly detection, but rather all the problems that are related to the sensors' world. In Chalapathy et al. [14], the authors write about DL for AD for all type of data (from simple time series, to video). In both of these surveys, several references to AD in sensor data can be found.

3.2 Change Point Detection

In the last decade, many works on CPD were done. Most of the recent works use the *Pelt* optimization method [13, 21, 19], to be able to solve both type 1 and type 2 CPD problems. The only exceptions, in recent years, are Lung-Yut-Fong et al. [28], in which the authors use the *Opt* method and a rank-based cost function and Frick et al. [17], in which the authors use *BinSeg*, an approximate optimization method to solve both CPD problem types.

In what relates to CPD optimization methods, *Opt* and *Pelt* were introduced in 1958 [11] and 2012 [25], respectively. The first of these was first introduced in a non-related field of research, and only applied to CPD a few decades later.

There are also multiple surveys about this subject. In the most recent one, Truong et al. [41] provide an overview of offline CPD methods, showcasing most of the optimization methods (optimal and approximated) and many cost functions. The authors also write about estimating the number of changes and most appropriate penalty. Furthermore, they implement some CPD methods in a Python package they called *ruptures*, which was useful in the present work. Niu et al. [34] and Shannalee et al. [8] also wrote extensive surveys on this topic.

3.3 Statistical tests for normality assumption

Two of the most popular statistical tests are the Chi-squared test and the Kolmogorov-Smirnov (KS) Goodness of Fit test [30]. Unfortunately, these two tests are not adequate for most problems. The Chi-squared test requires the sample to be small (up to only 20 samples), while the KS test leads to very conservative statistics, i.e. p-values strongly biased upward when an estimation of the mean and variance is needed [39].

Hubert W. Lilliefors [27] made a correction to deal with this upward bias of the KS test. Later on, this was further updated by Dallal and Wilkinson [15] and Stephens [40]. Despite these corrections, the KS test doesn't seem to be as powerful as other statistical tests, like Shapiro-Wilk (SW) [38] and Jarque-Bera (JB) [22][1].

[1]This means that SW and JB tests will make less Type II errors than the KS and Lilliefors tests, i.e., the less powerful tests (KS and Lilliefors) will incorrectly fail to reject the null hypothesis more often than their counterparts (SW and JB).

3.4 Clustering time series

Most of the existing works related to clustering in time series use existing clustering procedures, such as partitioning, hierarchical, grid-based, model-based, density-based and multi-step algorithms. In what relates to hierarchical clustering, according to Aghabozorgi et al. [6], such methods are weak in the quality of the resulting clusters, and for that reason, these methods are usually paired with another algorithm as a hybrid clustering approach. Furthermore, there are multiple works that aim to improve the quality of the hierarchical procedure, such as Chamaleon[24], CURE[20] and BIRCH[44].

The authors combine these procedures with a distance measure of their choosing – one that will reflect the warranted characteristics of the time series. There are multiple distance measures for different purposes. A few examples of such distances are: Euclidean, correlation-based, Dynamic Time Warping (DTW)[12], Longest Common Subsequence (LCSS)[42], Minimal Variance Matching (MVM)[26] and many others. The reader should refer to Aghabozorgi et al. [6] for more about this subject.

There are multiple works that combine an already existing clustering procedure with a distance measure. For instance, in Oates et al. [35], the authors combine agglomerative clustering and DTW to cluster the experiences of an autonomous agent.

An implementation of DTW in Python – *dtaidistance* [31] – will be used throughout this work.

Chapter 4

Identifying regimes

As stated in the first chapter, our goal is to improve the understanding of our data by identifying its regimes and by characterizing them through clustering. With that, we hope to gain valuable information about the data.

In this chapter, the work done on identifying the regimes is explained. First, we introduce the data and what part of it will be used. Then, we pre-process it, and follow through with an analysis and a brief outlier analysis. Next, CPD is applied, resulting in the regimes, which, in the next chapter, we will cluster and analyse the results and their characteristics, which hopefully will bring valuable knowledge about the data at hand.

4.1 Original data

The data that we will be using was provided by Bosch, came from systems of fire detection, and amounted to close to 60 gigabytes of disk space. There was plenty of more data to use other than these 60 gigabytes, but we chose to use this particular sample of it. The reason behind this decision is that we think that this amount of data will be enough to reach our goal (while a tinier sample would not), and at the same time it will not require as much computational resources as of hundreds of gigabytes would. The same method could likely be applied to more data with the same success.

In this data set, composed by more than 250 million instances and 60 attributes, there are 12 systems[1] from a single customer. Each row represents an observation of multiple

[1] A system is an infrastructure containing several devices.

variables of a single device at a given time. These observations date from 2016-09-23 to 2020-03-21. There are many types of sensors in this data set: points, couplers, modules and panels, each with its particular task. Each of these device groups may also have different types (e.g., there are multiple types of modules). These types can be thought of as a hierarchy, points being the most individual sensors, which are responsible for capturing specific environmental variables, and the panels being the most high-level device, that are responsible for monitoring the totality of the sensors (points, couplers and modules) that are assigned to it. Each type of sensor also has multiple models – points for instance, can be the model FAP-O420, FAP-O425, FAH-T420, etc. and each one of these models may capture different variables. The devices capture data from environmental variables (optical, temperature, pollution, etc.) to electrical variables and more.

In this data set, there is a very high amount of missing values. The reason for this is that, as stated before, there are dozens of different models of devices, and each of them is made to record some specific variables. For instance, it doesn't make sense for a module[2] to record environmental values. For that reason, the data set is very sparse.

We chose to work only with one of the most important types of sensors and their most important variable – FAP-O420 and opt1[3]. We chose this path because we wanted homogeneity in the data, rather than having multiple problems to address. Also, this set of device/variable is one of the few that provided sufficient data to analyse.

4.2 Pre-processing

The data processing consisted of transforming the raw data into sets of contiguous time series for each sensor. In Figure 4.1, we show the pre-processing pipeline done.

For a better understanding of the flowchart, we briefly expand on some of the sub-routines:

1. Date conversion – the date field is a timestamp composed by the date (year, month and day) and time (hour, minute and second) at which the observation was recorded. As we are only interested in the date, we remove the time from

[2]A module is a device that is responsible for monitoring the health of a loop of sensors.
[3]Most common optical sensor and environmental variable, respectively.

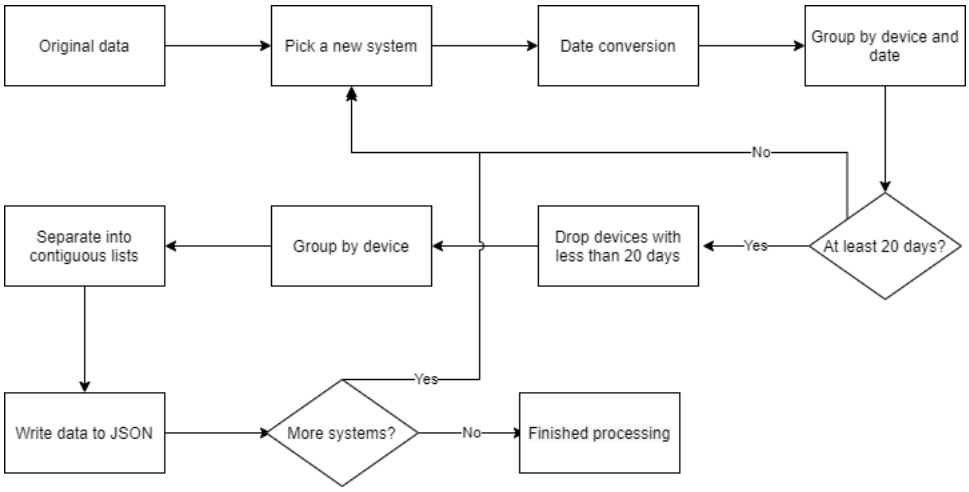


Figure 4.1: Pre-processing flowchart.

the values to get date-only timestamps. With this step, we will be able to group by the date at which the observations were recorded.

2. Group by device and date – a simple "group by" procedure is done here, where we get the mean of temperature observations of every device in each day.

3. Drop system/devices if it doesn't have a minimum of 20 different days – this is done to lessen the computational costs of the steps that follow. Note that this data would be dropped at the last step of this pipeline regardless.

4. Group by device – the goal here is to get a list of values, ordered by date, for each device.

5. Separate into contiguous time series – separate each device's list into multiple date-contiguous lists. Any list with less than 20 points is dropped.

Only time series which had at least 20 data points were considered. The reason for this is that we think this is the lowest reasonable series' length to work with in the next stages.

At the end of the pipeline, the data was written to JSON because this format seems to be the only one that correctly stores lists, something useful for us at this stage.

4.3 Data analysis

We will now delve into the analysis of our data after pre-processing, so that we have a better knowledge of it before moving on to CPD. The data processing procedure described in the last section culminated in 9900 contiguous time series. Two examples of such time series are shown in Figure 4.2. Notice that the series don't have the same time frame. This is due to different devices having observations at different times. Despite many of the series not having the same time frame, there will also be many of them that have.

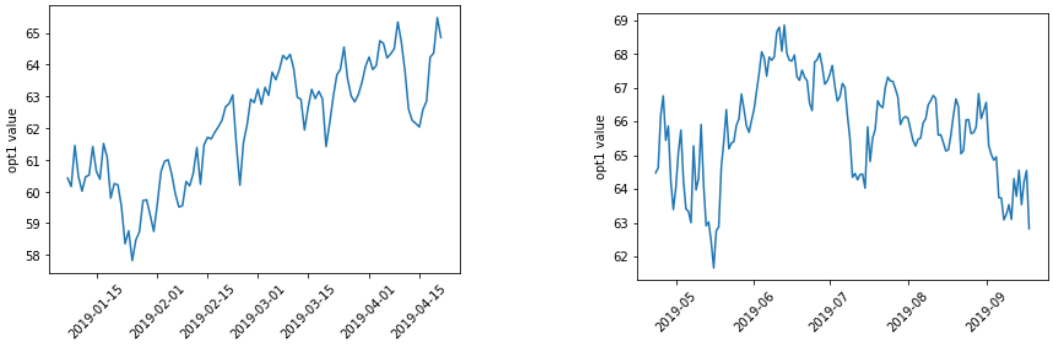


Figure 4.2: Two examples of the resulting time series.

To have an understanding of what values our time series takes along the time frame at which they were captured, in Figure 4.3 we show both the average time series and the region that contains all the values. One might think that different regimes can already be seen in this visualization. That would be a wrong assessment, as this visualization is an average of all the time series, which are heterogeneous and have different lengths, and so this can not be seen as regimes of the individual sensors. It's also worth to notice that there are multiple breaks in the series, where the value drastically goes up or down. Take for instance the upwards break that happens between May and September of 2019 of the average time series (on the left of Figure 4.3) – the reason for this event is that there are multiple time series with high $opt1$ values starting at that precise date, resulting in that steep increase in the average value.

In Figure 4.4, we show both the numerical distribution and the length distribution of the time series. We can see that most of the $opt1$ values are between 0 and 100, with some outliers going up to 300. In terms of length, its distribution is wide, ranging from 20 to close to 350 time steps, with an outlier surprassing the 400 time steps mark.

We also show, in Figure 4.5, observations' frequency throughout time. In the picture, it can be noted that there are multiple moments where the frequency bounces. The

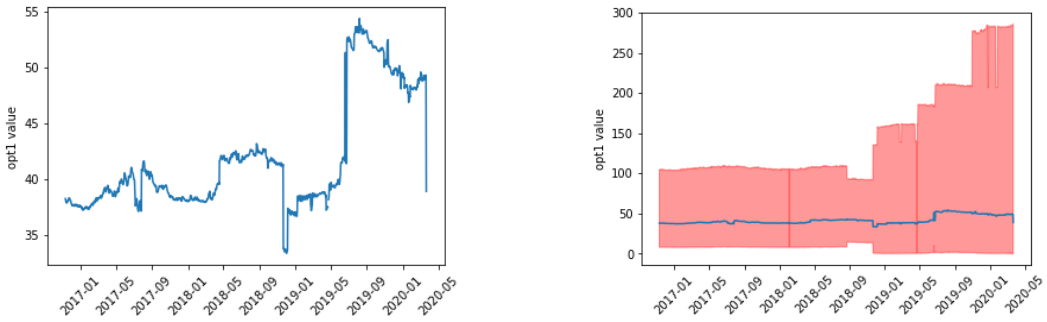


Figure 4.3: Average time series on the left. On the right, the same time series but with a red shadow representing the region in which data points are present.

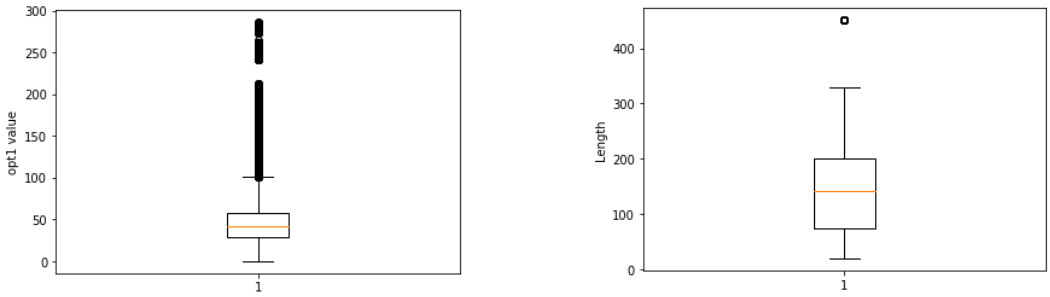


Figure 4.4: Distributions of the time series values and lengths, on the left and right, respectively.

reason for this is that there are many time series ending at a given date and then many starting right after that. The balance between how many finish and start in a given date dictates the dynamics of the bounce. This is the same reason that explains the steep increase in the average $opt1$ value of Figure 4.3. Furthermore, between May and September of 2019, we notice that there are multiple series ending and many starting right after. This indicates that the steep increase at that time, that we talked about above, is not only due to the appearance of new time series with high $opt1$ values, but also due to the end of many time series at that date (with lower $opt1$ values).

Finally, to start understanding the individual distribution of each time series, we present a heat map of $opt1$ values for 10 devices of a single system, in Figure 4.6. In this heat map, regimes are easily identified in multiple devices. I.e., in many rows of the heat map, there are patterns of colours, such as cooler and warmer regions in many of the series. This corroborates our first hypothesis (presented in subsection 1.2), that there indeed are regimes in our data. There are also some values which seem to be outliers, around the 164th day, in many of the devices (rows) presented.

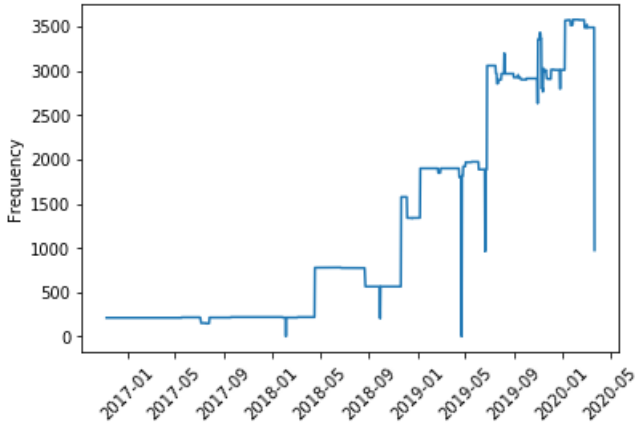


Figure 4.5: Frequency of dates in the time series.

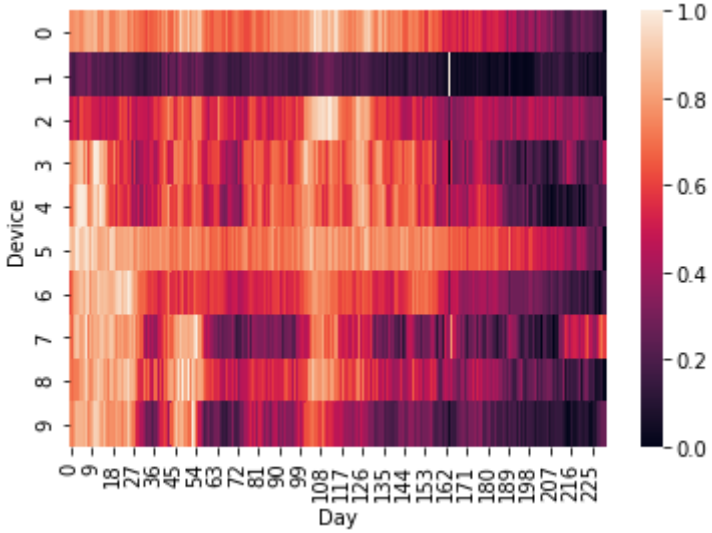


Figure 4.6: Heat map of 10 different devices of a single system for normalized *opt*1 values.

4.3.1 Will outliers be a problem?

Some outliers were detected in Figure 4.6, and for that reason, we will make a brief analysis of the outliers present in our data so that we can be sure they will not raise problems in a later stage of our work. In that same figure, despite multiple outliers being present, there is one that stands out the most – the one present in device 1. In Figure 4.7 we showcase the respective time series.

Furthermore, through z-score analysis of the data, we are able to find the rest of the series that have an outlier. In Figure 4.8, we show both the distribution of z-scores

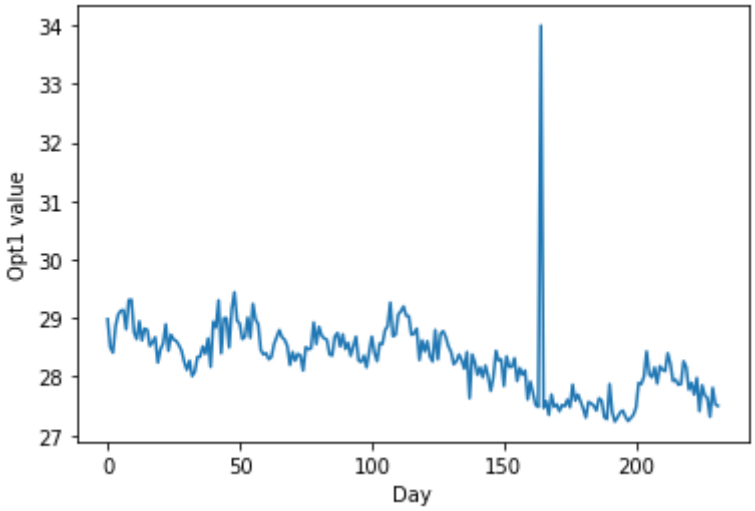


Figure 4.7: Time series of Device 1 of Figure 4.6

for all the data points of each time series and the distribution of the days that the outliers were registered. For the boxplot on the right, we only considered the 35 most extreme outliers (zscore greater than 8). With these illustrations, if we consider an outlier to have a z-score of greater than 3, then we have hundreds of time series with at least one outlier. Furthermore, the time steps at which the outliers happen seem to be well distributed through the lengths of the time series, falling close to the actual time series lengths' distribution (left-side of figure 4.4).

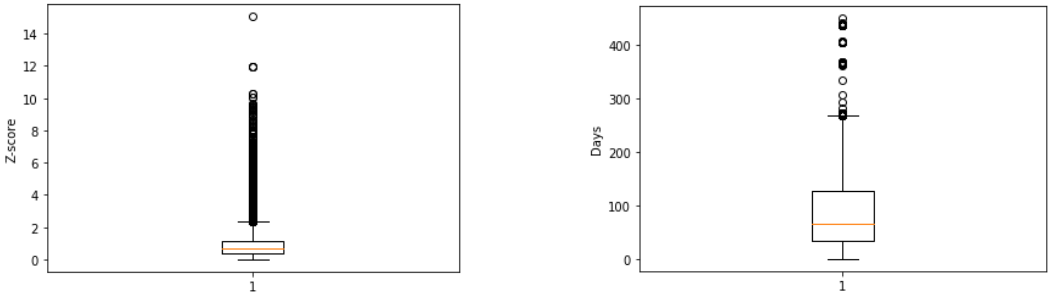


Figure 4.8: Distributions of the (absolute) z-scores and days at which an outlier was present, on the left and right, respectively.

We present two more examples of time series that have extreme outliers, in Figure 4.9. Most extreme outliers follow the pattern present in these two time series – the series are somewhat constant and then in a single day, the *opt*1 value grows a lot, to then come back to normalcy.

Despite having some extreme outliers present, we choose to do nothing about them,

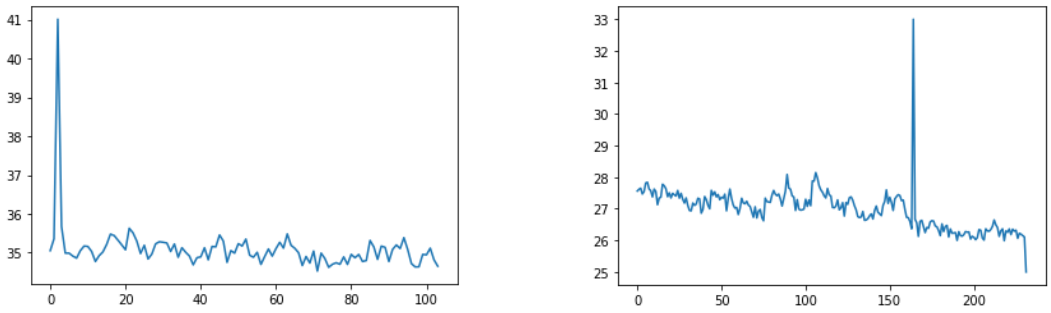


Figure 4.9: Two examples of time series that have an extreme outlier.

as they can bring valuable information to the regimes. E.g., two time series might be very identical due to the presence of outliers. Nonetheless, these outliers have to be kept in mind throughout this whole work, since we can not be sure about their impact at this stage.

4.4 Can we isolate the regimes?

In this section, we will dive into CPD in order to isolate each one of the regimes present in each one of the time series. First, we assess which assumptions can be made about our data – more precisely, if we can assume normality, – so that then we can choose which CPD methods to apply in our time series. Afterwards, we will ponder about which optimization method and cost function to use and then choose penalty values to combine with our methods. Finally, we will take the combination of our methods and penalties to do a preliminary analysis of the CPD results, to then proceed, in the next chapter, to the clustering of the resulting regimes.

4.4.1 Should we assume normality?

In some of the CPD methods, the data is required to be normally distributed. As we do not know the originating mechanism of our data, we do not have a priori information about the distribution of the data. Nonetheless, we can study the distribution in order to make or reject assumptions about it.

There are two main ways to assess if a sample is normally distributed – through visualization or statistical tests (see subsection 2.1.3). Because we are not working with a single distribution, but rather thousands of them, there is no easy way to

visualize whether each time series follows a normal distribution. For that reason, we have no choice other than use statistical tests.

In this particular problem, we have no knowledge of the mean and variance of our distributions, making the KS test inadequate. Furthermore, there are more powerful tests to assess the normality assumption, such as SW and JB (see subsection 3.3). In the remainder of this subsection, we will use the SW and JB tests to study the hypothesis that our time series follow normal distributions.

To assess whether we can assume normality, in Figure 4.10 we show the box plots for each of the statistical tests done – KS, SW and JB. Note that each test was done for each time series, and therefore each box plot represents the distribution of the p-values of the respective test for all the time series. Furthermore, the KS test was done just to have a sense of the discrepancy between it and the rest of the tests. Both the SW and JB tests result in a rejection of the null hypothesis (i.e., that a series follows a normal distribution) for 64% and 54%, respectively. On the contrary, the KS test had somewhat different results from his counterparts, resulting in a rejection of the null hypothesis for only 27% of the series.

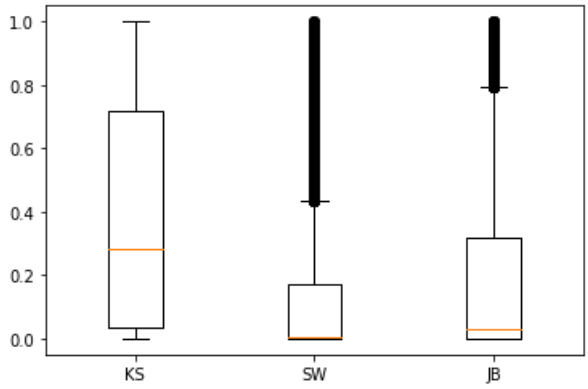


Figure 4.10: Boxplots for KS, SW and JB tests for the 9900 time series.

In order to assume normality, we would have to eliminate close to 50% of the time series present in our data set, which would result in a drastic reduction of the data. As we are not prepared to eliminate such a big portion of our data, we will not assume normality for any of the series and will rather continue with no assumptions about the underlying distributions.

4.4.2 What CPD method should we use?

In this problem, we have no prior knowledge about the number of partitions. We also don't have an immense amount of data. For those reasons, the best optimization method is *Pelt*, as it is the only one that provides exact solutions and works with an unknown number of partitions.

In what relates to the cost function, we don't have a choice other than to choose a kernel-based detection – more precisely, the Gaussian kernel. We have no choice because we are using the *Ruptures* package, and this package only provides this combination of optimization method and cost function for problems with unknown number of partitions and no assumptions about the data.

For those reasons, we are going to continue with a single combination of optimization method and cost function – *Pelt* and c_{rbf}.

4.4.3 What penalty is suitable for our time series?

The penalty has a great impact in the number of segments that the CPD method finds (see subsection 2.1.2.3). The number of partitions is of uttermost importance, since we do not want to find too many nor too few partitions. In Figure 4.11, the reader can get an insight about the impact of the penalty in the CPD procedure for a single time series of our data. Note that the vertical black lines are the change points detected by the CPD method. As the penalty grows, the number of change points detected decreases drastically.

Next, to study the impact of the penalty in the results of the CPD procedure in our data, we show, in Figure 4.12, how the number of change points detected for our 9900 time series varies, using the $Pelt + c_{rbf}$ method.

As expected, in the left side of the image, it can be seen that as the penalty gets bigger, fewer change points are found, until a convergence happens (number of minimum change points detected reached).

It can also be noticed, in the right side of the image, that the number of change points found is highly correlated with the length of the time series, i.e. the lengthiest time series, represented in red, has the highest mean of change points found. Furthermore, lengthier time series also seem to converge to the higher values of change points detected than shorter time series.

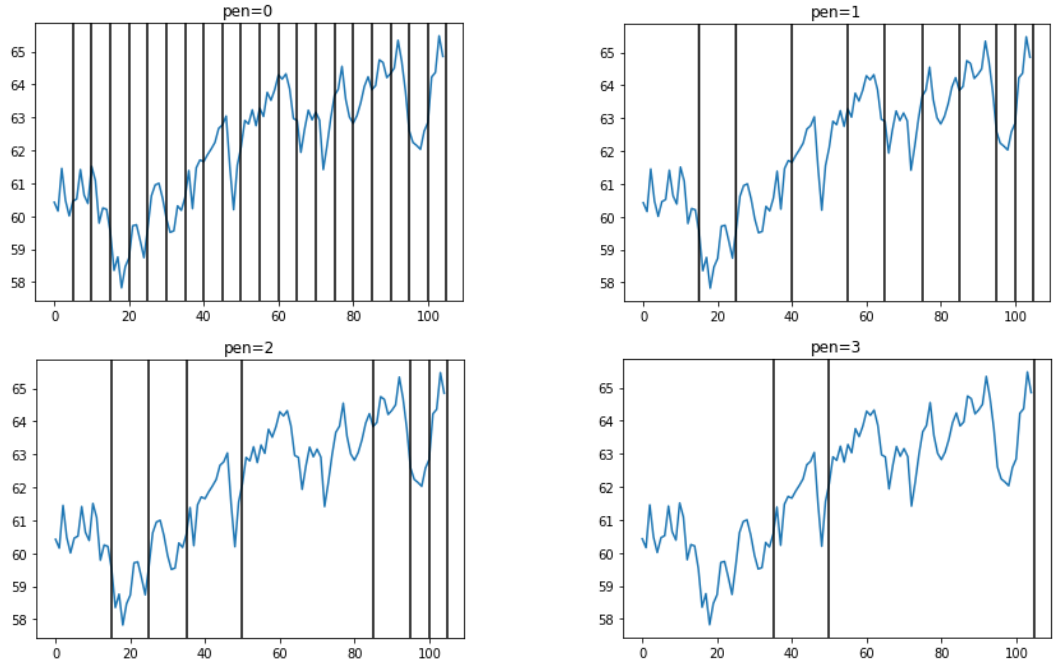


Figure 4.11: $Pelt + c_{rbf}$ CPD method results for a single time series, varying the penalty.

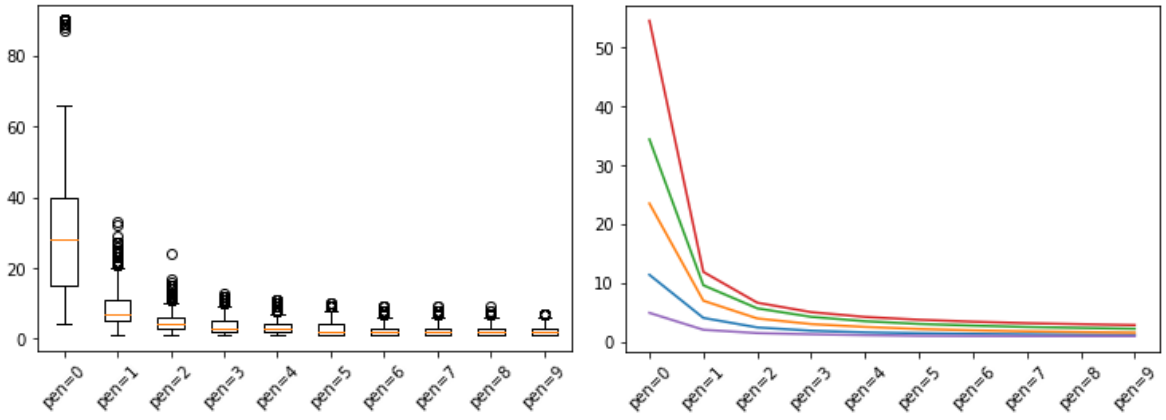


Figure 4.12: On the left, box plots of the number of change points found, by penalty. On the right, line plots of number of change points found, by penalty, grouped by length of series (red > green > orange > blue > purple).

At low penalty values, there is a drastic decrease of change points detected. This is due to the time series nature, which has a lot of tiny variations, and will in turn make the CPD procedure find a lot of change points when no penalty is used. This phenomena can also be seen in the first plot of Figure 4.12, where the tiniest change in the values produce a change point. This would be counter-productive for our goal,

as we want our regimes to have some useful meaning, and not encode all of the tiniest changes in our series.

Because we do not want either too many nor too few change points, we are choosing penalty values that we find to be the most balanced. Those values, in our opinion, are both 1 and 2 – those that come immediately after the "elbow" of figure 4.12. We will run the CPD procedure two times, using these two different penalty values. In the next subsection, a brief analysis of the results of such procedures follows.

4.4.4 CPD results

The CPD procedure resulted in 78175 and 45417 regimes, for $pen = 1$ and $pen = 2$, respectively. We will now proceed to briefly analyze these results.

In figure 4.13, the lengths' distribution for each set of regimes can be visualized. As expected, the lengths' distribution of the regimes from $pen = 2$ span a greater number of lengths and has a higher median, which means that it's more likely for a regime of $pen = 2$ to be longer than those of $pen = 1$.

Next, in figure 4.14 we visualize the mean and variance for each regime. The two figures are very similar, because a good portion of the regimes found in $pen = 1$ were also found in $pen = 2$, as expected. It is noticeable that there are a few regimes that stands higher than the rest in respect to variance. The regime with the highest variance can be seen in figure 4.15. The reader might ask why such regime was not cut around day 5, producing 2 regimes. The reason for this is that this depression around day 5, only spans 3 days, and the CPD procedure was set up to have a regimes with minimum of 5 time points, making it impossible to consider this depression a new regime.

In this chapter, we went from lengthy time series, derived from raw data, to regimes discovered through the CPD procedure. Next, we will dive into the clustering of these regimes, to try to find interesting patterns that may prove useful to better understanding the mechanics behind the data.

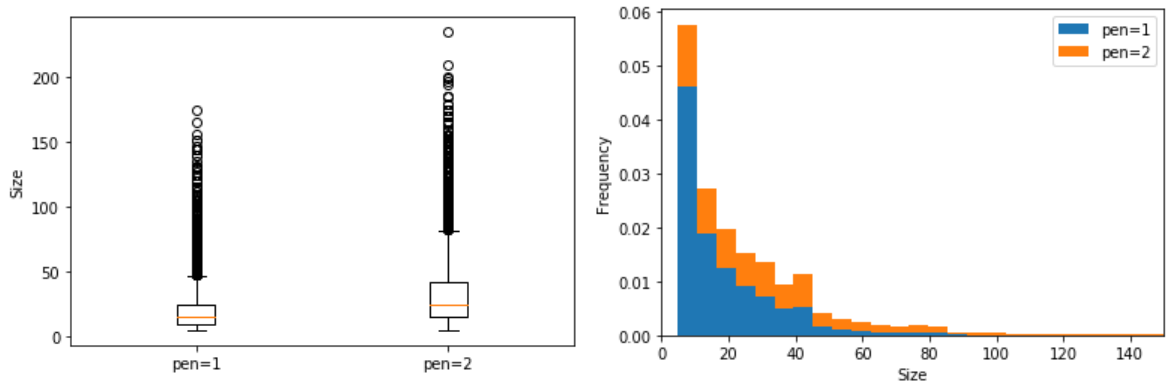


Figure 4.13: Box-plot and histogram of the lengths of regimes.

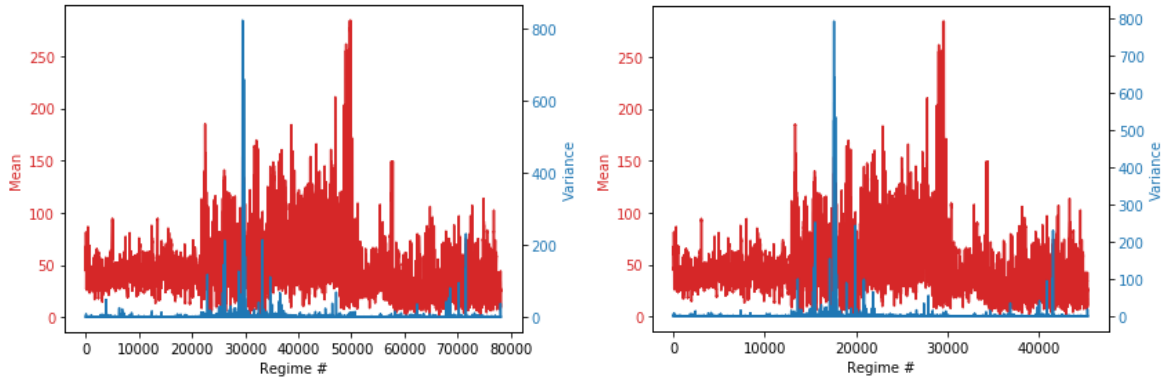


Figure 4.14: Mean and variance for each regime of $pen = 1$ and $pen = 2$, on the left and right, respectively.

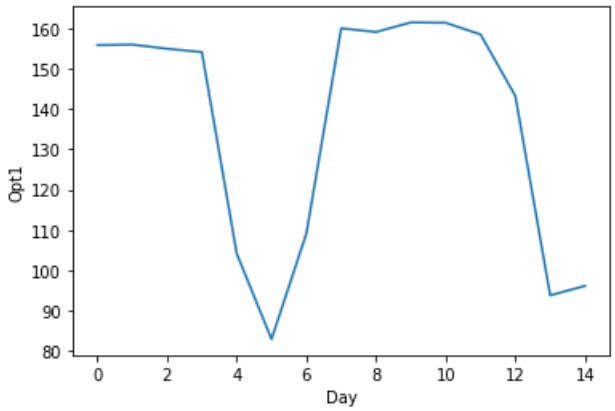


Figure 4.15: Regime with the most variance.

Chapter 5

Clustering regimes

In this chapter, we will work with the regimes found in the previous chapter. We will cluster them with the aim of trying to understand whether there are distinct groups of regimes with specific characteristics each, hoping to understand, through their characteristics, why these clusters exist in the first place and what makes them different. More precisely, we are looking for clusters of regimes according to their shape.

We will only use a sample of the regimes (the first 5 thousand of them). The reason for this is that working with all the regimes would not be feasible, as it would take substantially more time to deal with all the problems that it would raise (the clustering algorithms would need too much memory, to start with), particularly if we reach no satisfying result at the end. For now, this work serves to study if such methods will bring any useful knowledge about the data. In later stages, it may be adequate to expand this work to all the data available.

We should be aware that the results of this section could differ if another sample had been used. Note also that the results of the clustering for the regimes of $pen = 1$ were very similar to those of $pen = 2$. For that reason, we will only show the results of clustering in using the regimes obtained with $pen = 1$, and will only make the distinction when relevant.

In the remainder of this section, we will first discuss about the distance measure to be used and then visualize the resulting pair-wise distances. We will then talk about the possible clustering algorithms for this task and make some experiments with them. Next, we will find a suitable number of clusters to produce cluster prototypes and finally analyze the clusters and their predominant shapes, while trying to answer the

research questions made in the first chapter.

5.1 What distance measure fits our goal?

The distance measure to be used depends on whether the time series have the same length and what characteristic are we looking to encode with such measure. As seen in the last section, our time series differ in length. We have also mentioned that we are looking for time series similar in shape. An adequate option, in that case, is DTW (see subsection 2.2.1.1).

To visualize the pair-wise distances and to have a sense of how the regimes distribute themselves (distance-wise), we use both Muldi-Dimensional Scaling (MDS) and t-Distributed Stochastic Neighbor Embedding (t-SNE). Such visualizations are presented in figure 5.1. In none of these visualizations there seems to be any evidence of clusters. This likely means that we will not reach a satisfactory clustering result.

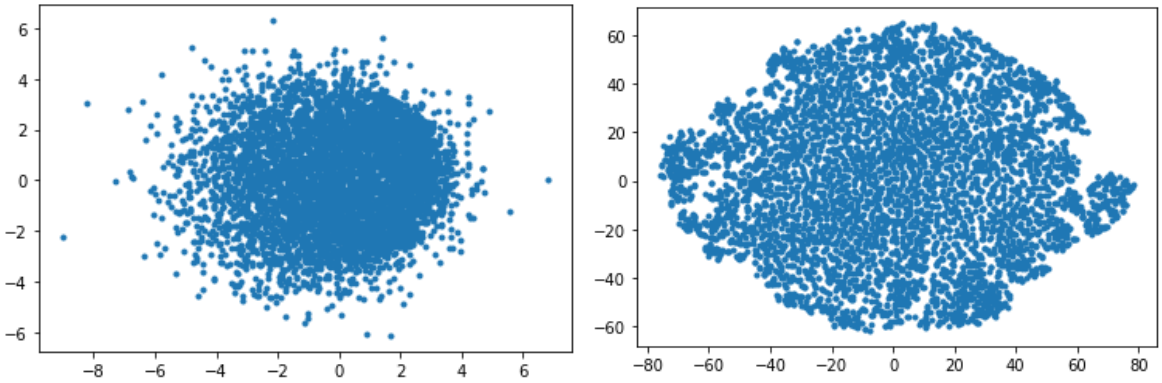


Figure 5.1: MDS and t-SNE visualization of the pair-wise DTW distances of the sample of regimes.

5.2 Clustering algorithms and preliminary results

After computing the pair-wise distance matrix, we can now apply a clustering algorithm to the matrix. There are many clustering algorithms that work with a precomputed distance matrix. We tested the following algorithms: DBSCAN, OPTICS, Affinity Propagation, K-Medoids and Agglomerative Hierarchical Clustering (AHC), all from the Scikit-Learn package [37].

All of the clustering algorithms produced very poor results:

- DBSCAN, with low ϵ, considered most of the time series to be noise instances, not belonging to any cluster. With higher ϵ, considered most of the time series belonging to the same cluster, and the rest as noise.
- OPTICS, resulted in most of the time series classified as noise, and then the few that weren't noise, were distributed throughout a high number of clusters with a low number of instances assigned to them.
- Affinity Propagation found multiple clusters with a somewhat satisfying number of instances in each of them. Unfortunately, looking to the Silhouette score and Dunn's index, one sees that the clusters are of a very low quality (both metrics close to 0).
- K-Medoids, having a parameter that defines the number of clusters to be found, it had no choice but to find that exact number of clusters. Regardless, the resulting clusters revealed to be very weak, as can be seen in figure 5.2 – independent of number of clusters chosen, the Silhouette score and Dunn's Index are very close to zero. Nevertheless, we show in figure 5.3, the resulting clusters, only for demonstration purposes.

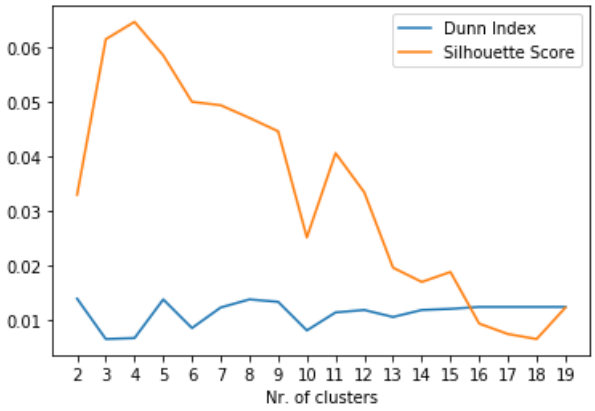


Figure 5.2: Silhouette score and Dunn's Index for K-Medoids, varying number of clusters.

The clustering algorithm that seemed to produce the best results was AHC, but this also fell short of the minimum required quality, as we are going to see now.

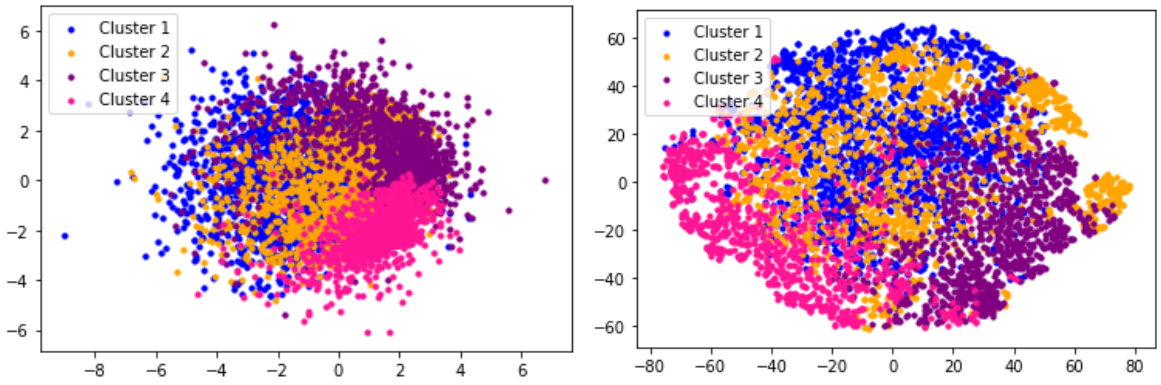


Figure 5.3: MDS and t-SNE visualization for the results of the K-Medoids clustering algorithm.

For pre-computed distance matrixes, Hierarchical Clustering offers 3 different methods to cluster the instances: complete or maximum linkage, average linkage and single linkage. All of these 3 methods were tried.

First, we show the respective dendograms for each of the methods, in figure 5.4. The dendograms show that the complete linkage method produced multiple groups of instances, with one of the groups (yellow), being the most predominant, followed by the brown and the red one. On the other hand, average linkage method resulted in a group completely dominating the rest, to the point that no other group can be seen in the dendogram. Single linkage dendogram is not shown as it produced a "maximum recursion depth exceeded" error while constructing the dendogram. Note that the dendogram colours do not match the colours of the clusters in the MDS and t-SNE visualizations.

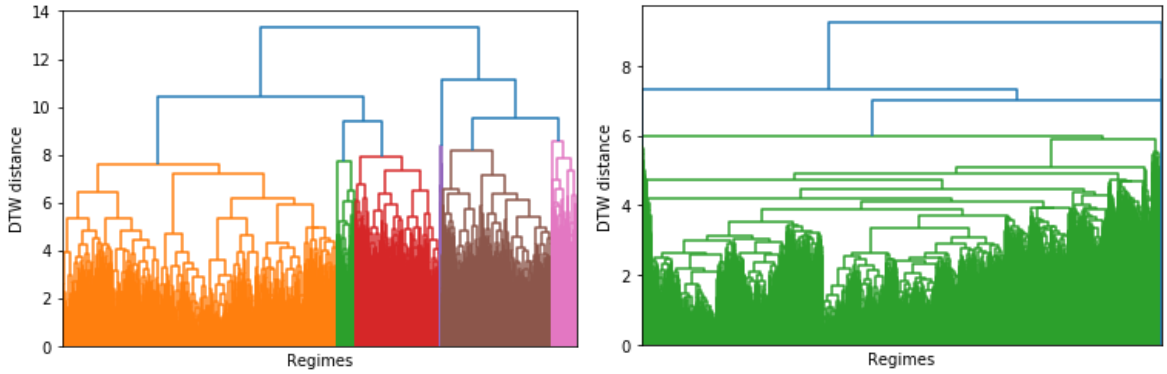


Figure 5.4: Dendograms for complete and average linkage methods, on the left and right, respectively.

Figure 5.5 shows, through the analysis of the Silhouette score and Dunn's index,

that single and average linkage with low number of clusters appear to be the most promising methods, reaching medium-high scores. Unfortunately, these good scores only happen because, as was saw in the dendogram of the average linkage in figure 5.4, the clustering algorithm assigns almost every instance to a single cluster, while the rest of the clusters, despite distant, have a very low number of instances assigned to them (in some cases, 1 or 2 instances). This is not useful for our task, and such clusters do not represent anything of worth.

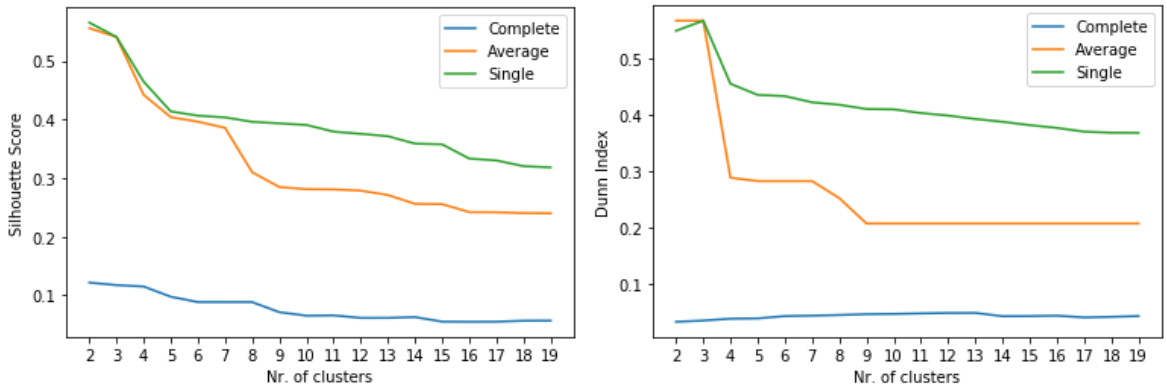


Figure 5.5: Silhouette score and Dunn's Index for each of the method, varying the number of clusters.

On the other hand, we have the complete linkage method, which do not result in good metrics, independently of the chosen number of clusters. The most satisfying number of clusters, by a tiny margin, seems to be 2 clusters, but again, the algorithm assigns only a few instances to the second cluster, which is not useful to the task at hand.

For comparison purposes, in figure 5.6, we show the scores for every clustering method that is not misleading, i.e., that does not have clusters with very few regimes. All of the methods resulted in very poor scores, with *OPTICS* standing out with a negative silhouette score.

Clustering, for these regimes, does not reach an acceptable quality, and therefore our second hypothesis presented in subsection 1.2 can not be validated for now. The rest of this section will be related to AHC with complete method linkage using 6 clusters. We chose 6 as the number of clusters because the silhouette score and Dunn's index decreases only by a tiny margin in low number of clusters, and we thought 6 to be a good number of clusters for visualization and exploration purposes.

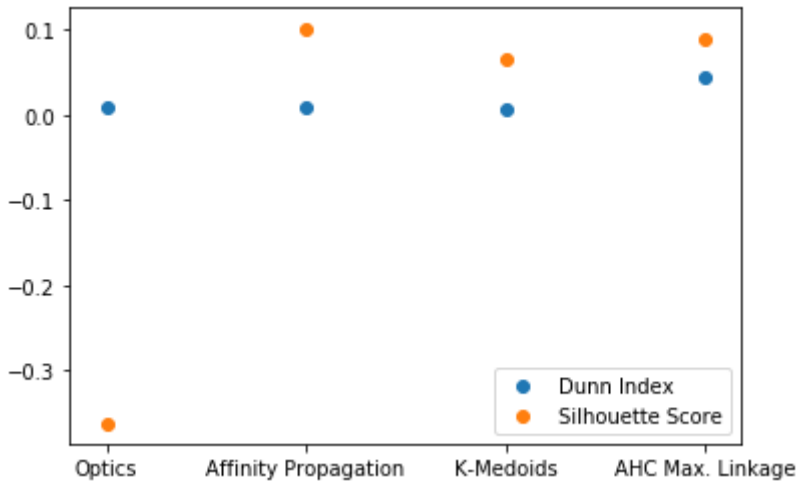


Figure 5.6: Silhouette score and Dunn's index for the clustering procedures that all clusters have a significant number of regimes.

5.3 Gathering knowledge from the resulting clusters

Despite the clusters being poor in quality, in this section, we are going to assess whether the clusters can give us any valuable information about the regimes at hand, proceeding to their in-depth analysis.

In table 5.1, it can be seen that, as expected from the first dendogram of figure 5.4, there's one cluster that dominates (cluster 4 with 2667 instances assigned to it), and then the clusters 2 and 5, with 1073 and 882 instances, respectively, that have a considerably higher number of elements than the other 3 clusters. The algorithm also produces a cluster with only 20 instances.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Nr. of points	233	1073	20	2667	822	185

Table 5.1: Table showing the count of instances per cluster, for the AHC with complete linkage results.

In figure 5.7, the most noticeable characteristic of either visualization, are the points assigned to the dominant clusters (cluster 4, 2 and 5, coloured as purple, blue and green, respectively). It is also clear that there is not much division between the clusters, as the low Silhouette score seen in figure 5.5 indicated.

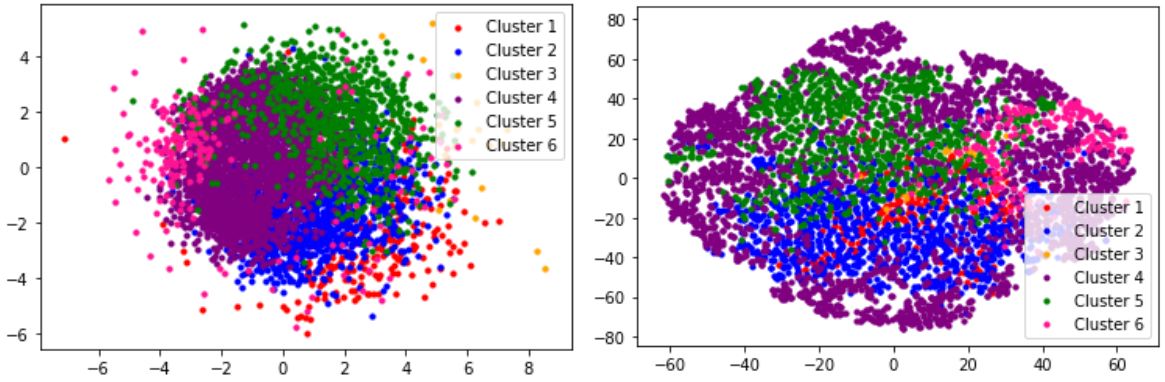


Figure 5.7: MDS and t-SNE visualizations for the AHC with complete linkage results.

With figure 5.8, we are trying to understand whether the regimes of the different clusters start and/or end at different dates. Unfortunately, one sees no relevant difference between the start and end dates of each cluster. Cluster 3 dates slightly deviates from the other clusters, but this is the cluster with 20 instances, so this deviation is probably due to the low number of instances, rather than a inherent difference in the cluster itself. Note that the first column in each image represents all the instances, regardless of cluster, for comparison purposes.

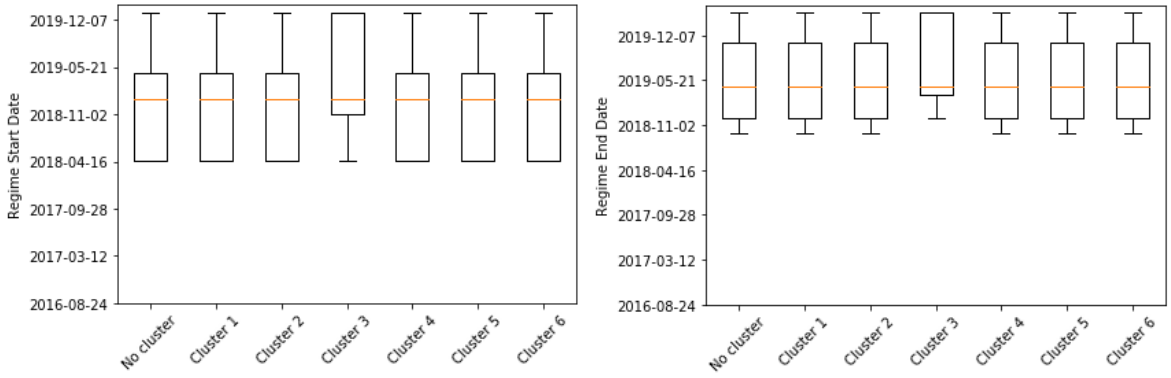


Figure 5.8: Boxplot distributions of start and end dates for each cluster of the AHC with complete linkage results.

In figure 5.9, we present the length and pair-wise distance distributions for each of the clusters. It seems like the regimes assigned to each cluster tend to be similar in respect to their lengths. This is probably due to the fact that the DTW distance tends to be lower for similar length regimes. In respect to pair-wise distance, these seem to have some correlation with how many points the cluster has assigned to it. For instance, the most frequent cluster (cluster 4), has the pair-wise distance distribution with most variance. This also means that it's the cluster that spans the most area, as

can be confirmed in figure 5.7. On the other hand, we have the less frequent cluster (cluster 3), that does not have the lowest variance of pair-wise distances, giving the podium of lowest variance to the 3rd less frequent cluster (cluster 1).

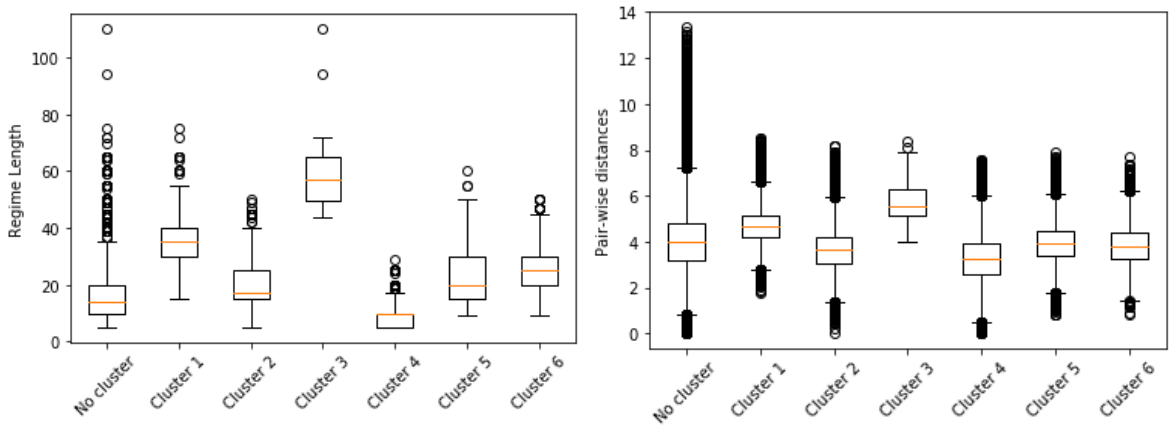


Figure 5.9: Boxplot distributions of lengths and pair-wise distances for each cluster of the AHC with complete linkage results.

To try to understand the patterns of precedence and succession for each of the clusters, we present figure 5.10. There, we can see that the distributions of the clusters of the regimes that precede and succeed each of the 6 clusters. These images represent what comes before and after (left and right hand sides, respectively) a given cluster (x-axis). E.g., from the left hand side of the figure we know that, independently of the cluster to which a regime belongs, the regime that comes before it will most likely belong to cluster 4 (purple). In this visualization, there are some points worth noticing:

- In the right hand side of the figure, it can be seen that after a regime from cluster 3, there will most likely come nothing (end of series). This was somewhat expected, since cluster 3 has the longest regimes.

- Cluster 3 also shows a preference of being succeeded by clusters 1 (red) and 4 (purple). As cluster 3 has the longest regimes, it was expected that if anything succeeds it, it would be a short regime (cluster 4 has the shortest regimes). However, cluster 1 is unexpected, since it has the second longest regimes in our set of clusters.

- Cluster 3 also deviates from the expected pattern of being preceded by cluster 2 (blue) more predominantly than the cluster 5 (green), a pattern that is seen in every other cluster.

- Cluster 4 (purple) is predominant almost everywhere, as expected, since it is the most frequent cluster.
- There is a clear pattern that is present in most of the clusters: first comes the purple with the highest probability of either succeeding or preceding a given cluster, then blue and finally green, while the rest of the clusters are always close to being insignificant. This goes hand in hand with the frequency of the clusters.

From the observations made above, it is evident that there are many deviations from the expected patterns in what relates to cluster 3. This is probably due to the low number of points belonging to that cluster.

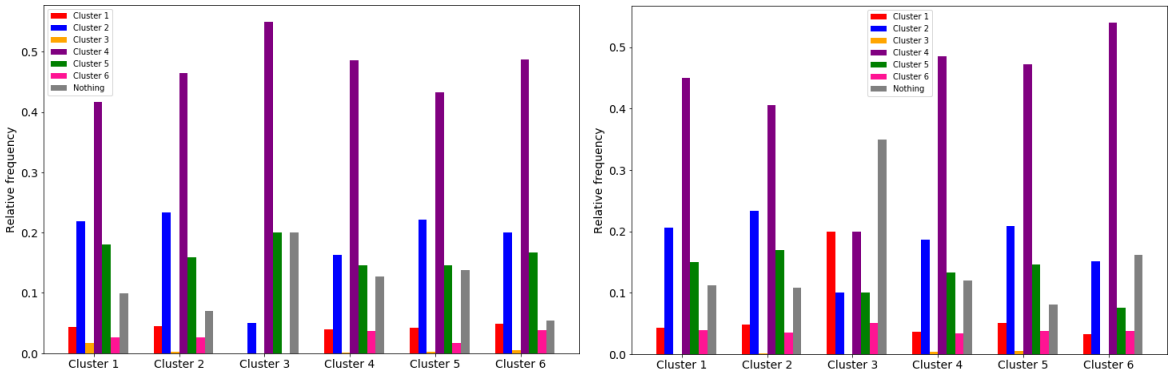


Figure 5.10: Precedence and succession of clusters for each cluster, on the left and right, respectively.

In figure 5.11, the positions of the regimes in their respective time series, grouped by cluster, are shown. The positions seem to be inversely proportional to the regimes length, visualized in figure 5.9. E.g., cluster 3 regimes' tend to be the longest, and so they come early in the time series they belong to, as can be seen in cluster 3 box plot of figure 5.11. This is expected, as very long time series will make the respective time series have less regimes. The rest of the distributions seem to be similar, as expected, due to the similarity of their lengths.

5.3.1 Cluster prototypes and their shapes

We considered the prototypes for each cluster to be the most centered instance of each cluster, i.e., the regime with the lowest sum of squared errors (DTW distance) when compared with the rest of the regimes assigned to the same cluster. In figures 5.12,

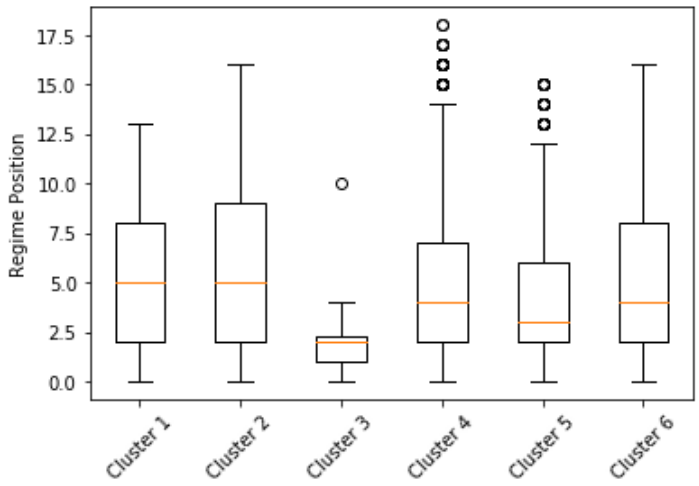


Figure 5.11: Boxplots of the regimes position in the time series they belong to, grouped by cluster.

5.13 and 5.14, the location of the prototypes and respective prototypes regimes are presented.

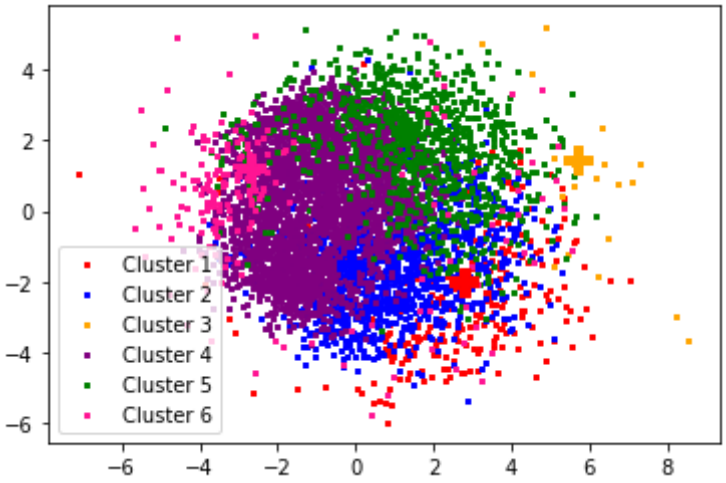


Figure 5.12: MDS visualization for the AHC with complete linkage results and respective prototypes localizations (represented by a plus signal).

Despite some of the prototypes (represented by crosses) being hidden by the rest of the points in figure 5.12, one can still notice that the prototypes have a fair distance between them. This is further verifiable in figure 5.14, where none of the prototypes are similar. The reader can have a better sense of the position of the prototypes, in each cluster, in figure 5.13.

To have a better understanding of the shapes inside of each cluster, in figures 5.15

and 5.16, we show the prototype and a sample of 9 regimes for each of the clusters, accompanied by where each of the instances is located. Through the analysis of figures 5.14, 5.15, 5.16, a description of each shape predominant characteristics can be made:

- In cluster 1, the regimes seem to start at high values and then have a succession of decreases and increases.
- In cluster 2, the regimes seem to start at high values, decrease, increase again, and at the end decrease more softly than at the beginning. Despite this shape seeming similar to that of cluster 1, the regimes of this cluster are shorter.
- Cluster 3 shape is clearly defined by the lengthiest regimes, which don't seem to have a proper shape other than being the longest.
- Cluster 4 seems to be defined by the shorter lengths.
- Cluster 5 shape seems to be mainly defined by 2 peaks – the regimes first start by increasing, reaching a peak, to then decrease and increase again, reaching another peak.
- Cluster 6 seems to be mostly defined by an increasing tail.

We can finally conclude that the shapes for this clustering method (AHC) seem to be mostly defined by:

1. Length of regime,
2. Number of peaks in regime,
3. Order of increase/decrease and where such increases and decreases happen in the regime.

5.4 Discussion on the clustering results

In this section, we will try to answer the questions made in the subsection 1.2 with the knowledge gathered in the previous section and provide some final remarks about the clustering.

First, we asked if there were clusters of shapes in the regimes. With these regimes, we can not say that there are clusters, since the obtained clusters are mostly overlapping

and have no apparent division between them. Nonetheless, there still may be useful information that can be gathered from these sections of regimes.

Second, we asked whether we could achieve a description of the shapes of the typical regime for each cluster. We did that in the end of the previous section, with some success. The only problem that we found here is that some clusters have a very high variance in shape, and many times a regime from a given cluster is not similar at all to the prototype of that same cluster.

Then, we asked whether there was some difference in the dates that the regimes happen from cluster to cluster. We saw in the last section that there is no difference from cluster to cluster.

Finally, we asked whether there were any interesting patterns on the precedence or succession among clusters. We found cluster 3 to be the most interesting, since it keeps deviating from the pattern present in the rest of the clusters. We are confident that this is due to the cluster 3 having the lengthiest regimes and also not having many points assigned to it. We believe that with a bigger sample, this cluster would be more identical to the other clusters, but still deviate from the usual pattern, for its usual length of regimes.

Unfortunately, the clusters were not well divided, and we believe that this was the main reason for not finding more interesting knowledge in the clusters. In our opinion, our clusters were not well divided because of the poor quality of the regimes used – the CPD procedure simply looks for different distribution in each of the time series, rather than looking for similar distributions that are present throughout all time series. Looking for different distributions that are present in many time series would provide more coherent regimes, and consequently, we believe, better clusters.

Despite the clusters not having the most quality, one could argue that the clusters provide us some useful knowledge about the data, since we are looking at regions of regimes, rather than the whole data, and therefore we have a better understanding of the different sections of the data.

We also noticed that the regimes' shapes inside each cluster are not very homogeneous. An increase in number of clusters would probably increase the homogeneity of the shapes, but this would likely decrease the quality of the clustering.

For the reasons mentioned, we cannot confidently validate our third hypothesis of subsection 1.2 – the clusters and their shapes can provide valuable information about the data, – although we believe that further work could indeed validate it. Despite that,

we are confident that this thesis is a step in the right direction of better understanding the data through a regimes/clustering approach, as we were close to promising results and even reached some interesting knowledge, such as the predominant shapes of the regimes.

In this chapter, we went from the regimes produced by CPD to clusters of regimes, using DTW as distance measure as input to AHC. Unfortunately, such clusters, did not properly separate the data at hand but still provided some valuable information about the data. In the end, we could not validate our second and third hypothesis of this thesis as we do not have sufficiently strong evidence to do so.

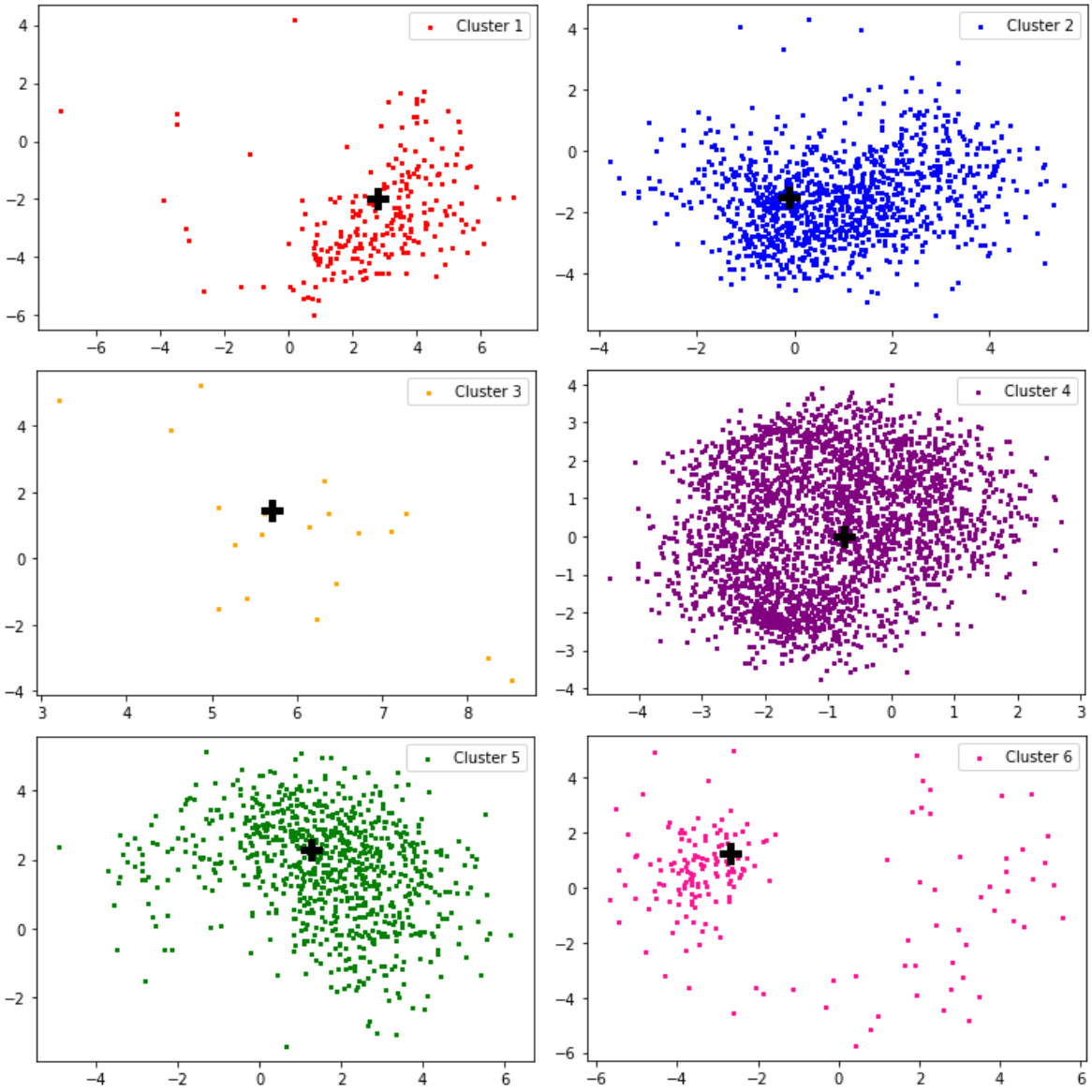


Figure 5.13: MDS representation of each cluster of the AHC with complete linkage results and respective prototype instances (represented by a plus signal).

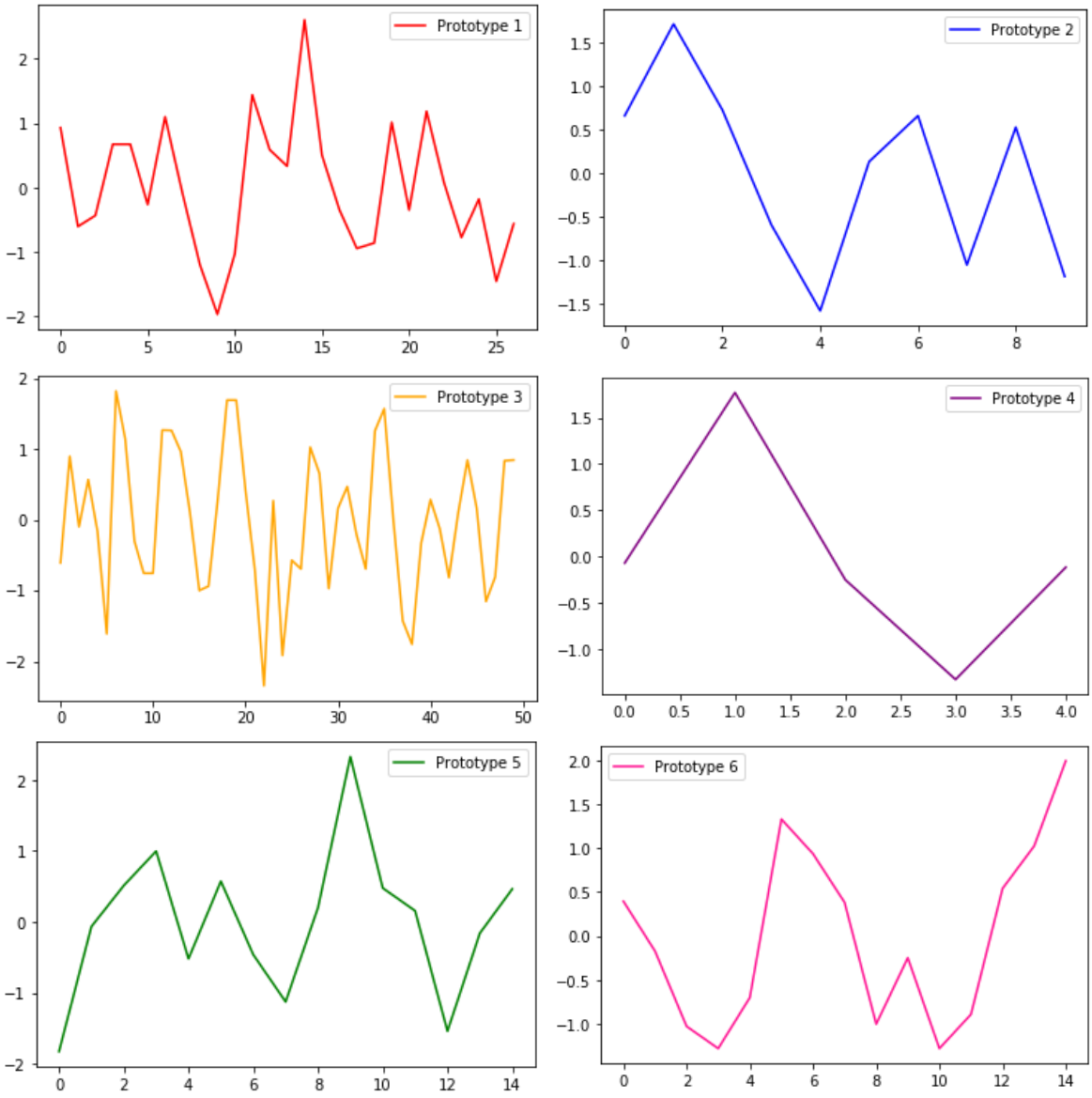


Figure 5.14: Regime prototypes of each of the clusters.

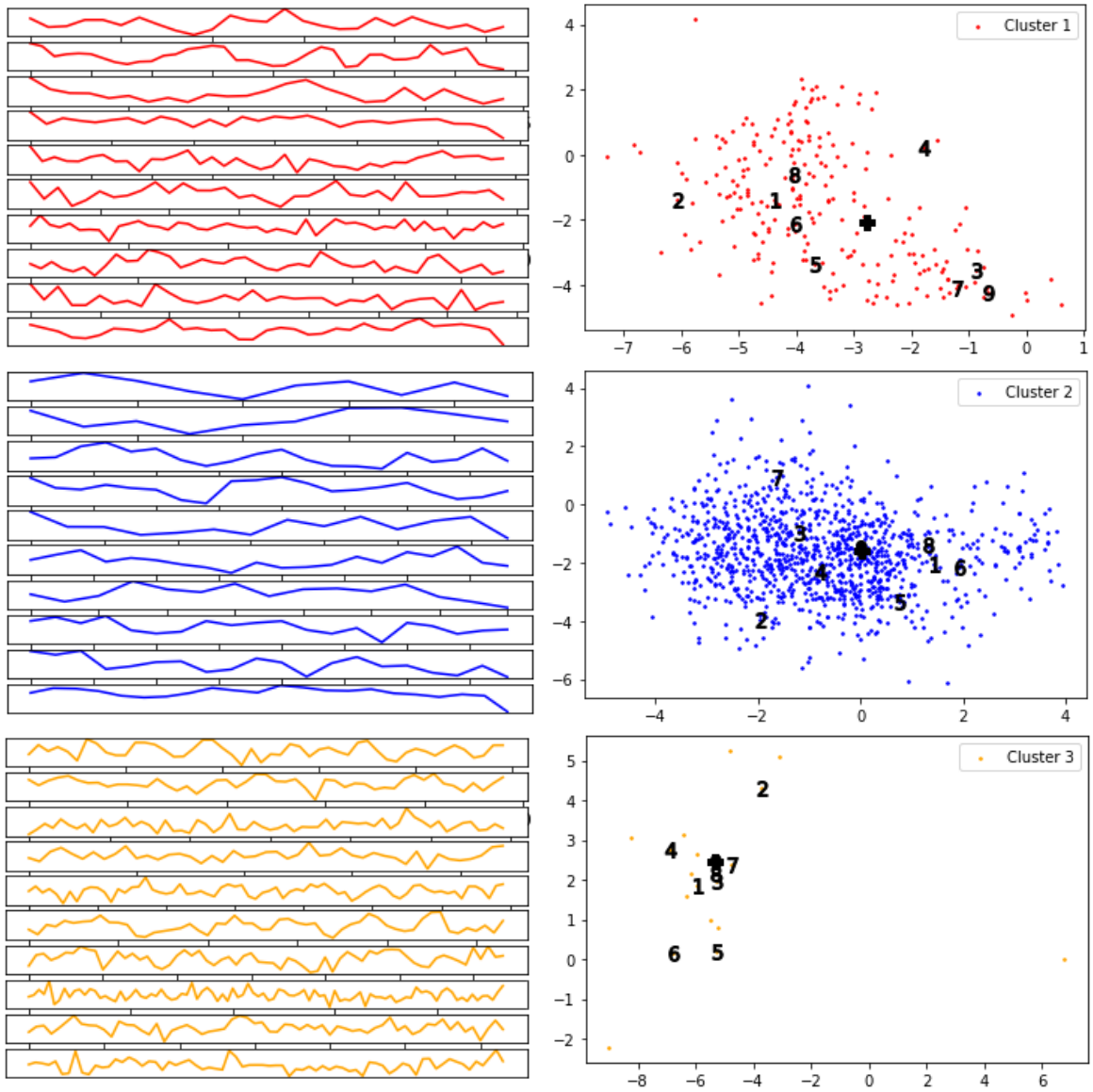


Figure 5.15: On the left, the prototypes (first series of of each plot on the left) and a sample of 9 regimes for each one of the first 3 clusters. On the right, their respective positions in the cluster.

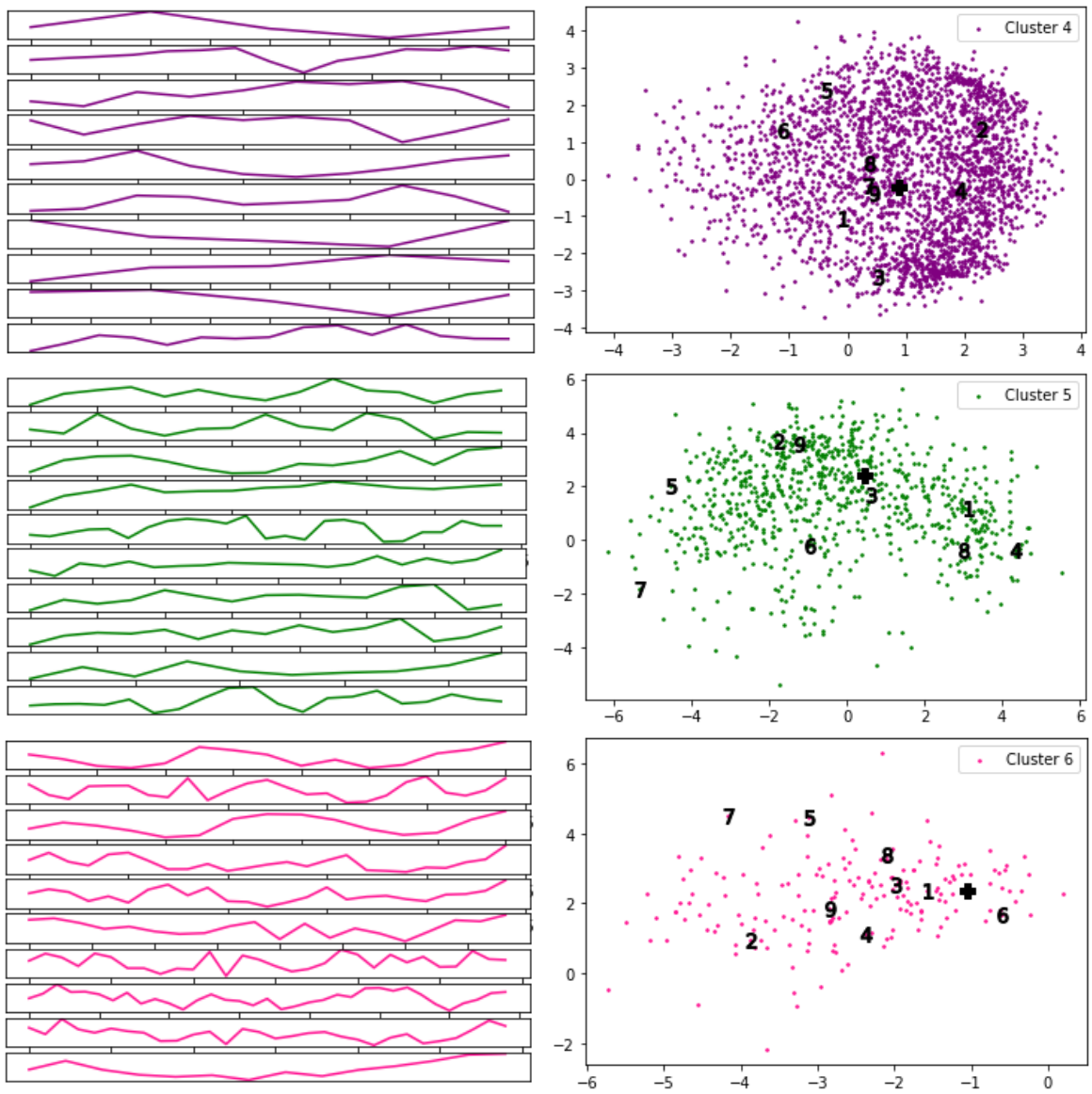


Figure 5.16: On the left, the prototypes (first series of of each plot on the left) and a sample of 9 regimes for each one of the last 3 clusters. On the right, their respective positions in the cluster.

Chapter 6

Conclusion

In this thesis, we aimed to better understand fire detecting sensors' data and, in a best case scenario, to reduce its false alarm rate. Right from the start, reducing the false alarm rate was a seemingly impossible task due to the nature of our data – we did not know whether the data we had, had any relation to alarms. Because of that, we decided to follow another path in order to, hopefully, gather valuable information about the data that would provide valuable insights into the originating mechanisms of the data: first, to divide our sensors' time series in regimes (sequences that follow different distributions), and then to cluster those regimes in the hope that there would be well defined clusters of them, providing us valuable information about the different clusters and why those clusters happen in the first place. The analysis of those clusters would provide us with the shapes of their prototypes (the most representative regime of each cluster), the dates at which the regimes of each clusters tend to happen, the precedence and succession of the regimes' clusters, and much more interesting knowledge.

For the detection of regimes, we used CPD, which is a method that divides a sequence into sub-sequences taking into account their distributions. Achieving tens of thousands of different regimes, we now were able to compare those regimes through the DTW distance function, which compares the shape of two time series. Calculating the distance of every pair of regimes, we now had a square matrix that could be used as input to multiple clustering methods. As seen through the MDS and t-SNE visualization of the distance matrix, clusters are not clearly identified. This was further confirmed by the clustering procedures. Every single one of them produced very poor results, with their resulting clusters overlapping consistently, achieving very poor silhouette scores and dunn's index.

In what relates to our hypotheses, we are confident to say that the first one was validated. The second and third one could not be validated in this work, since we achieved only poor clusters (i.e., data not well divided into clusters) and therefore those clusters do not provide us with the most faithful information about the data at hand. Nevertheless, we saw interesting knowledge emerging in the analysis of the clusters, and we believe that further work could prove useful in the pursuit of validating our hypotheses and gathering important knowledge of the data. Further work could include a rework of the definition of regimes – we do not think the CPD method used was the best for this task, as mentioned in the end of Chapter 5. We think that something like CPD but that maintains cohesiveness between all the time series would provide better results. Furthermore, deviating from the hypotheses of this thesis, one could also try to cluster the structure of the complete time series instead of the shapes of their regimes. That could also be useful to a better understanding of the data.

In this work, we dived into the uncharted fire sensors' data provided to us by Bosch. It served as a beginning of the exploration of the data and the regimes/clustering approach to gather valuable knowledge about the nature of it. We started with a non-processed dataset of dozens of gygabytes, and ended up with some interesting predominant shapes of regimes that are present throughout the time series of the sensors.

Appendix A

Acronyms

AD - Anomaly Detection

AHC - Agglomerative Hierarchical Clustering

CPD - Change Point Detection

DL - Deep Learning

DTW - Dynamic Time Warping

KS - Kologorov-Smirnov

JB - Jarque-Bera

SW - Shapiro-Wilk

Referências

[1] Euclidean vs dtw. https://commons.wikimedia.org/wiki/File:Euclidean$_v s_D TW.jpg. Accessed$: $2020 - 09 - 01$.

[2] Iot growth projections. https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/4d59e9b71480. Accessed: 2020-09-21.

[3] It industry trends analysis. https://www.comptia.org/content/research/it-industry-trends-analysis. Accessed: 2020-09-21.

[4] Poor quality data consequences. https://www.forbes.com/sites/forbespr/2017/05/31/poor-quality-data-imposes-costs-and-risks-on-businesses-says-new-forbes-insights-report/465daa7452b1 . Accessed: 2020-09-21.

[5] What is a sensor? https://www.electronicshub.org/different-types-sensors/. Accessed: 2020-09-21.

[6] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

[7] Hyojung Ahn, Dawoon Jung, and Han-Lim Choi. Deep generative models-based anomaly detection for spacecraft control systems. *Sensors*, 20(7):1991, 2020.

[8] Samaneh Aminikhanghahi and Diane J. Cook. A survey of methods for time series change point detection. *Knowl. Inf. Syst.*, 51(2):339–367, 2017.

[9] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[10] John E Ball, Derek T Anderson, and Chee Seng Chan. Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *Journal of Applied Remote Sensing*, 11(4):042609, 2017.

[11] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[12] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:, 1994.

[13] Souhil Chakar, E Lebarbier, Céline Lévy-Leduc, Stéphane Robin, et al. A robust approach for estimating change-points in the mean of an ar(1) process. *Bernoulli*, 23(2):1408–1447, 2017.

[14] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[15] Gerard E Dallal and Leland Wilkinson. An analytic approximation to the distribution of lilliefors's test statistic for normality. *The American Statistician*, 40(4):294–296, 1986.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[17] Klaus Frick, Axel Munk, and Hannes Sieling. Multiscale change point inference. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, pages 495–580, 2014.

[18] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 401–410, 2005.

[19] Damien Garreau, Sylvain Arlot, et al. Consistent change-point detection with kernels. *Electronic Journal of Statistics*, 12(2):4440–4486, 2018.

[20] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. *ACM Sigmod record*, 27(2):73–84, 1998.

[21] Kaylea Haynes, Idris A Eckley, and Paul Fearnhead. Computationally efficient changepoint detection for a range of penalties. *Journal of Computational and Graphical Statistics*, 26(1):134–143, 2017.

[22] Carlos M Jarque and Anil K Bera. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics letters*, 6(3):255–259, 1980.

[23] Frank J. Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[24] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[25] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[26] Longin Jan Latecki, Vasilis Megalooikonomou, Qiang Wang, Rolf Lakaemper, Chotirat Ann Ratanamahatana, and Eamonn Keogh. Elastic partial matching of time series. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 577–584. Springer, 2005.

[27] Hubert W Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American statistical Association*, 62(318):399–402, 1967.

[28] Alexandre Lung-Yut-Fong, Céline Lévy-Leduc, and Olivier Cappé. Homogeneity and change-point detection tests for multivariate data using rank statistics. *arXiv preprint arXiv:1107.1971*, 2011.

[29] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[30] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[31] Wannes Meert, Kilian Hendrickx, and Toon Van Craenendonck. https://github.com/wannesm/dtaidistance, August 2020.

[32] Mohsin Munir, Shoaib Ahmed Siddiqui, Muhammad Ali Chattha, Andreas Dengel, and Sheraz Ahmed. Fusead: unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models. *Sensors*, 19(11):2451, 2019.

[33] Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. Deepant: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, 7:1991–2005, 2018.

[34] Yue S Niu, Ning Hao, and Heping Zhang. Multiple change-point detection: A selective overview. *Statistical Science*, pages 611–623, 2016.

[35] Tim Oates, Matthew D Schmill, and Paul R Cohen. A method for clustering the experiences of a mobile robot that accords with human judgments. In *AAAI/IAAI*, pages 846–851, 2000.

[36] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[38] Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[39] Dag J Steinskog, Dag B Tjøstheim, and Nils G Kvamstø. A cautionary note on the use of the kolmogorov–smirnov test for normality. *Monthly Weather Review*, 135(3):1151–1157, 2007.

[40] Michael A Stephens. Edf statistics for goodness of fit and some comparisons. *Journal of the American statistical Association*, 69(347):730–737, 1974.

[41] Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Process.*, 167, 2020.

[42] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Proceedings 18th international conference on data engineering*, pages 673–684. IEEE, 2002.

[43] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[44] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996.