Time Series Data Analytics: Clustering-Based Anomaly Detection Techniques for Quality Control in Semiconductor Manufacturing

by

Oumaïma Makhlouk

Diplôme d'ingénieur, Arts et Métiers ParisTech (2018)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of
Master of Engineering in Advanced Manufacturing and Design

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2018

© 2018 Oumaïma Makhlouk. All rights reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created

Author..... Signature redacted

Department of Mechanical Engineering
August 15, 2018

Certified by..... Signature redacted

Duane S. Boning
Clarence J. LeBel Professor, Electrical Engineering and Computer Science
Thesis Supervisor

Certified by..... Signature redacted

David E. Hardt
Ralph E. and Eloise F. Cross Professor, Mechanical Engineering
Thesis Reader

Accepted by..... Signature redacted

Rohan Abeyaratne
Quentin Berg Professor of Mechanics &
Chair, Committee of Graduate Students

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
OCT 02 2018
LIBRARIES
ARCHIVES

Time Series Data Analytics: Clustering-Based Anomaly Detection Techniques for Quality Control in Semiconductor Manufacturing

by

Oumaïma Makhlouk

Submitted to the Department of Mechanical Engineering
on August 19, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Advanced Manufacturing and Design

Abstract

Optimizing their manufacturing systems and processes whilst ensuring a low production cost is important for Analog Devices, Inc. (ADI). Therefore, detecting anomalies on production lines and alerting on out-of-control processes is crucial. Although Statistical Process Control (SPC) methods have been implemented in the past and have proven to be efficient, the company seeks improvements using machine learning. The Machine Health Project is one of the data analytics-based projects under way at ADI to implement such improvements. Anomaly detection techniques can be effective in improving the quality control on semiconductor production lines. Sets of data collected from semiconductor manufacturing machines, such as a plasma etcher, can be analyzed to control the fabrication process and test the efficiency of machine learning algorithms. This thesis focuses on cluster analysis for outlier detection, and provides a univariate strategy to find potential anomalous behaviors in the data when a given parameter is known as relevant. If a more thorough analysis of the data is needed, a multivariate clustering analysis can also be computed. In addition, decomposition-based algorithms are presented. These rely on techniques such as the STL and SAX representations of time series, and provide a visual computation of time series discords. In this thesis, these methods are implemented, and their results are compared. Recommendations are provided as to how to best utilize the outputs of these outlier detection algorithms.

Thesis Supervisor: Duane S. Boning
Title: Clarence J. LeBel Professor, Electrical Engineering and Computer Science

Thesis Reader: David E. Hardt
Title: Ralph E. and Eloise F. Cross Professor, Mechanical Engineering

This page was intentionally left blank.

Acknowledgments

I would like to take this opportunity to express my very great appreciation to the people who contributed, directly or indirectly, to making this project a reality and a success.

This journey would not have been possible without the support and love of my dear parents and little brothers. Thank you for encouraging me to follow my dreams and teaching me values such as determination and hard work. Nothing would have been possible without you. I would like to express my deepest gratitude to my lovely grandma, Fatima-Sofia, who left us this past May. The courage and dedication of my grandma, who was illiterate but the wisest person I have ever met, has always inspired me. I wish to dedicate this work to her.

I would like to thank Prof. Duane Boning for his guidance through the project and his valuable advice. I also want to offer my special thanks to Jose Pacheco for putting us in contact with Analog Devices, Inc. It was a great opportunity to gain industry experience.

I would like to express my very great appreciation to Mr. Jack Dillon, Mr. Ken Flanders, Mr. Leslie Green and Mr. Charles Mathy for their valuable and constructive suggestions during the planning and development of this research work. Their willingness to give their time so generously has been very much appreciated. My special thanks are extended to every person from Analog Devices, Inc. for their support.

Last but not least, I would like to thank my teammates Tiankai Chen and Han He for the time we spent together, their help, the fruitful discussions and brainstorming sessions we had. Thank you to all my friends who made my journey at MIT a memorable one. And to my friends in France and Morocco, thank you for your thoughts and support throughout this project.

This page was intentionally left blank.

Table of contents

List of Figures

This page was intentionally left blank.

List of Tables

This page was intentionally left blank.

Chapter 1

Introduction

In this chapter, the thesis project is introduced. The scope and objectives as well as the motivation for this project are presented. Finally, the organization of the thesis is provided.

1.1 Problem statement

Analog Devices, Inc. (ADI) is an American multinational semiconductor company that specializes in data conversion and signal processing technology. The company manufactures analog, mixed-signal and digital signal processing (DSP) integrated circuits (ICs) used in virtually all types of electronic equipment. The company was founded in 1965 by Ray Stata and Matthew Lorber and currently operates in 23 countries. ADI has fabrication plants located in Massachusetts, United States and in Limerick, Ireland. These fabrication plants primarly supply low volume high value products, while high volume and low value products are outsourced to contract manufacturers.

In order to reduced the cost of fabrication, ADI faces pressure to optimize their manufacturing system and processes. Although ADI has relevant results when implementing Statistical Process Control (SPC) on their production line, they are looking at the feasibility of machine learning (ML) algorithms to further improve their system.

One effort to supply this goal is the Machine Health Project that focuses on optimizing the way data is collected from their processes and analyzed. This system hopes to provide relevant timely alerts on out-of-control processes and anomalies and enables immediate response to the problem. The project aims at improving machine reliability, productivity and quality, while reducing the costs.

1.2 Project scope

The project presented through this thesis is part of a larger one conducted by a team of three MIT students. Its aim is to optimize the health monitoring model for the machines and processes in the Wilmington Fabrication Plant (Massachusetts).

The objective of the project is twofold: to implement the data analytics and evaluate the analytical techniques used, for anomaly detection in semiconductor fabrication processes. In this team effort, evaluation of ML algorithms for both limits monitoring and anomaly detection. The ML algorithms considered are included in the Microsoft Azure Machine Learning Suite. Figure 1-1 shows an overview of this project and functions of the company's manufacturing and assembly facilities.

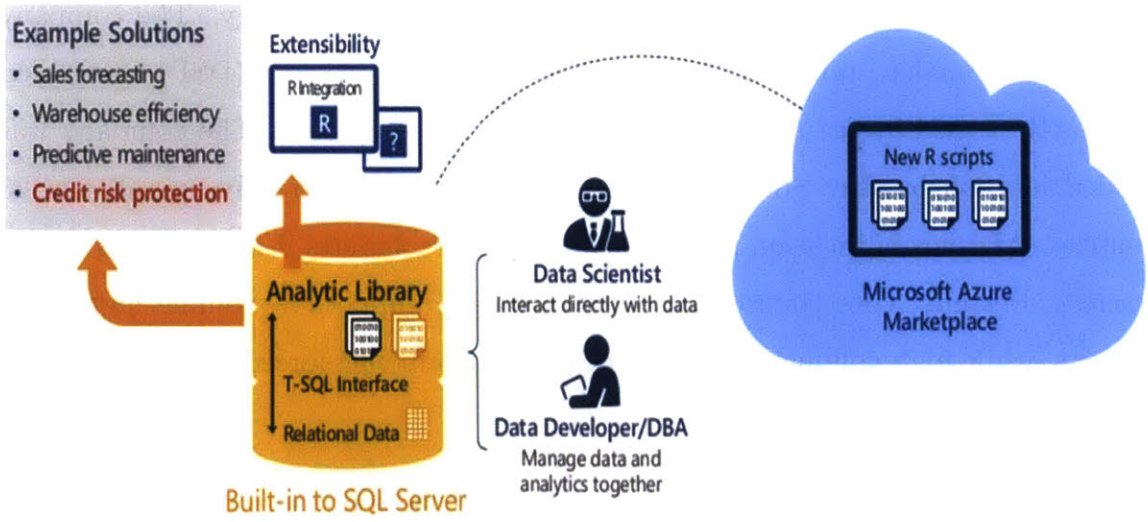

Figure 1-1: Built-in Analytics using R [1]

The project develops analytics using the R software within the SQL server. The data has to be prepared carefully before ML algorithms can be used. In the long-term (beyond the scope of the project), ADI is also looking to implement "real time" analytics in their manufacturing facilities. Therefore, some of the best practices for implementation are determined and deployed. A platform has been created in order to analyze the data which consists of a Preprocessing Toolbox, a Data Visualization Toolbox and a Data Analytics Toolbox. Figure 1-2 below shows the workflow of this project.

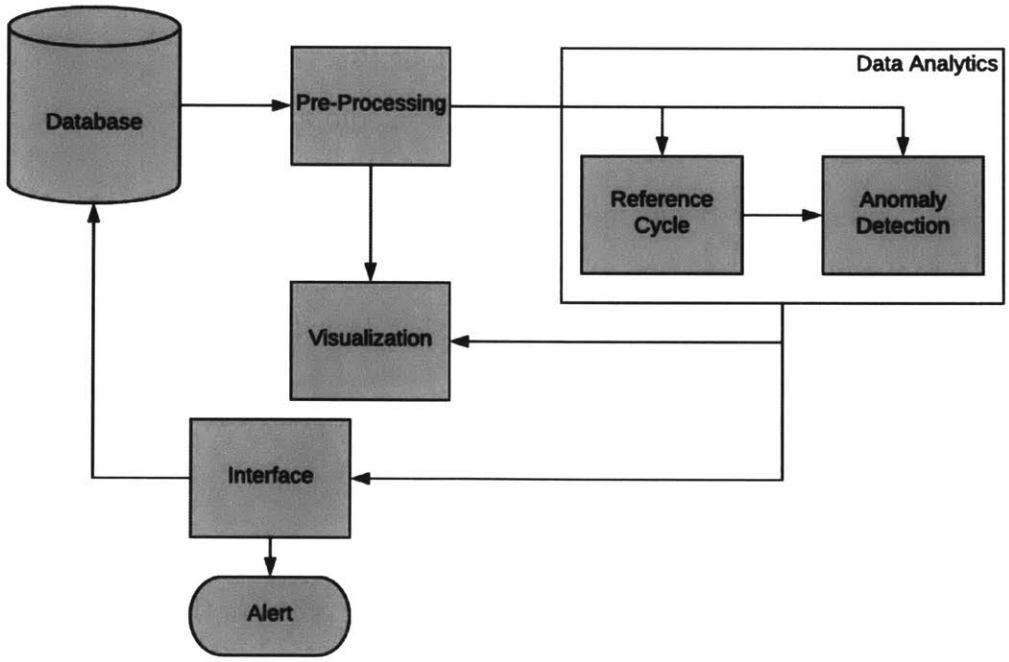

Figure 1-2: Workflow of the project

1.3 Objective

The main objective of this thesis project is to implement a model in order to:

- Evaluate anomaly detection algorithms
- Evaluate machine learning algorithms
- Determine best practices for implementing "real time" analytics

The proposed model will contribute to improving the quality control process of the manufacturing of wafers. The project focuses on one particular machine, the plasma etcher, as the case study to explore and evaluate ML anomaly detection methods. However, the approaches implemented in this project are meant to be broadly applicable to other semiconductor fabrication equipment and processes.

This thesis focuses on the deployment of cluster analysis algorithms for anomaly detection. More detailed information about building the reference cycle can be found in Han He's thesis [2]. Tiankai Chen's thesis [3] has focused on developing neural networks to detect outliers.

1.4 Thesis organization

This thesis is structured into five chapters. Chapter 1 presents the theory behind all the cluster analysis methods chosen as part of this project. Chapter 2 provides the results of the implementation of the clustering algorithms. Chapter 3 suggests additional anomaly detection techniques based on time series decomposition. Finally, Chapter 5 contains conclusions and a summary of the objectives achieved through this thesis project.

Chapter 2

Cluster Analysis

In this chapter, various cluster analysis methods are presented. The concept of clustering is first introduced, followed by presentation of similarity measures used to create clusters. Two major types of clustering algorithms are then discussed, partitional methods and hierarchical clustering. Finally, fuzzy clustering is also presented.

2.1 Clustering

Unsupervised learning problems deal with finding a structure in unlabeled data. Cluster analysis consists of a set of unsupervised learning methods aiming at determining subgroups or clusters of observations within a data set [4]. Such an analysis is based on the information found in the data that describes the objects and their relationships. Unlike classification analysis, clustering requires determining the number and composition of the groups. It is used in a variety of fields, including biology, statistics and pattern recognition. The aim of clustering is to group similar objects whilst the groups are different from one another, such that the clusters present high internal homogeneity in addition to high external heterogeneity. The greater the similarity within a group and dissimilarity between the groups, the higher the quality of the cluster analysis.

Multiple types of clustering and clusters have been developed. It is important to make a relevant choice for both the type of clustering and type of clusters when performing cluster analysis. Hierarchical and partitional clustering are the most commonly used ones. **Partitional clustering** consists of dividing the set of data objects into mutually disjoint partitions. The clusters are therefore distinct from one another. **Hierarchical clustering** consists of permitting clusters to be nested, meaning that clusters can also be part of subgroups; clusters are thus organized as a tree.

Cluster analysis can be a useful exploratory data analysis tool in discovering outliers in a dataset. An outlier is defined as an anomalous observation, one that significantly deviates from the rest of the data. Identifying outliers in a dataset is a precious source of information on the untypical behavior of a set of observations.

2.2 Similarity metrics

In this section, key similarity metrics used for clustering are presented. Distance measures are first explained, followed by correlation distance measures.

2.2.1 Distance measures

The most important feature of cluster analysis is the fact that data points within a group need to be similar whilst the clusters are unrelated to one another. The more distinct the cluster, the higher the clustering quality is considered to be. Other clustering requirements include the ability to deal with a wide variety of attributes, noise or outliers. The similarity between two observations corresponds to the degree of correspondence among objects based on all their characteristics. It is typically defined by dissimilarity or distance measures between different observations [5].

There are various methods for measuring distances for clustering. Three of the main distance measures are the Euclidean, the Manhattan and the Minkowski metrics.

Euclidean

The Euclidean distance is the most commonly used metric. Formally, it is also known as the l_2 norm. The distance $d_{Euclidean}$ (x'_i,x'_j) between two p-dimensional points x'_i=$(x_{i1},...,x_{ip})$ and x'_j=$(x_{j1},...,x_{jp})$ in Euclidean space is defined by:

$$d_{Euclidean} = \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2} \tag{2.1}$$

Manhattan

The Manhattan distance is another commonly used metric. Formally, it is also known as the l_1 norm. The distance $d_{Manhattan}$ (x'_i, x'_j) between two p-dimensional points x'_i=$(x_{i1},...,x_{ip})$ and x'_j=$(x_{j1},...,x_{jp})$ is defined by:

$$d_{Manhattan} = \sum_{k=1}^{p}|x_{ik} - x_{jk}| \tag{2.2}$$

Minkowski

The Minkowski distance metric can be used for a rectilinear configuration. It is also known as the l_r norm. The Euclidean (r = 2) and the Manhattan (r = 1) distances are both special cases of this distance. The Minkowski distance $d_{Minkowski}(x'_i, x'_j)$ between two p-dimensional points x'_i=$(x_{i1},...,x_{ip})$ and x'_j=$(x_{j1},...,x_{jp})$ is defined by:

$$d_{Minkowski} = (\sum_{k=1}^{p}|x_{ik} - x_{jk}|^r)^{1/r} \tag{2.3}$$

2.2.2 Correlation distance measures

Other dissimilarity measures exist, including correlation-based distances. Through this type of distances, two objects are considered as being similar if their features are highly correlated, even if other distance metrics like the Euclidean one might find those objects as being far apart. These include the Pearson and the Eisen cosine correlation distances.

Pearson correlation distance

The distance $d_{Pearson}\ (x'_i, x'_j)$ between two p-dimensional points x'_i=$(x_{i1}, ..., x_{ip})$ and x'_j=$(x_{j1}, ..., x_{jp})$ is defined by:

$$d_{Pearson} = 1 - \frac{|\sum_{k=1}^{p}(x_{ik} - \bar{x_i})(x_{jk} - \bar{x_j})|}{\sqrt{\sum_{k=1}^{p}(x_{ik} - \bar{x_i})^2(x_{jk} - \bar{x_j})^2}} \tag{2.4}$$

Eisen cosine correlation distance

The Eisen cosine correlation distance is a special case of Pearson correlation distance where the mean of the points is zero. This distance $d_{Eisencosine}\ (x'_i, x'_j)$ between two p-dimensional points x'_i=$(x_{i1}, ..., x_{ip})$ and x'_j=$(x_{j1}, ..., x_{jp})$ is defined by:

$$d_{Eisencosine} = 1 - \frac{|\sum_{k=1}^{p} x_{ik}x_{jk}|}{\sqrt{\sum_{k=1}^{p} x_{ik}^2 \sum_{k=1}^{p} x_{jk}^2}} \tag{2.5}$$

The choice of distance measures has a paramount influence on the quality of the clustering. While the Euclidean distance is one of the most commonly used, the choice of distance metrics depends on the type of the data and the problem to solve. The value of these distance metrics depends on the scale on which measurements are made. This is the reason why the variables of interest are often standardized. In fact, such a scaling of the data is relevant, especially for multivariate analysis. It aims at making the variables comparable when they are measured in different scales.

2.3 Partitional clustering

Partitional clustering [6] consists of dividing the data into k different clusters. The number k has to be determined beforehand. According to Sarda-Espinosa (2017), partitional clustering is an optimization procedure that minimizes the distance between the observations within the cluster and maximizes the distance between the clusters. However, finding the optimal solution to this problem is complex. Several heuristics have been developed in order to approximate such a solution. Those include **k-means**, **k-medoids** as well as its variation for large data sets, **CLARA**. In this section, the basic idea behind each of these methods are presented.

2.3.1 k-means algorithm

Developed by MacQueen in 1967 [7], k-means [8] is an unsupervised machine learning technique that falls under the category of partitional clustering. It is one of the most commonly used clustering approaches. The goal of k-means is to construct k groups or clusters of greatest possible distinction within a given data set. Through an iterative process, each data point is assigned to one of those k groups based on its similar features to the other data points of the cluster. Such a similarity is measured based on distance metrics, such as the Euclidean, Manhattan or Minkowski distance. Given that k-means is a prototype-based algorithm, it defines a prototype in terms of a centroid, which is typically the mean of a group of data points. Therefore, the cluster is represented by this "centroid", or mean of the points in the cluster. The centroid is thus not an actual data point.

As it is an optimization problem, the objective of k-means clustering is to minimize the total intra-cluster variance or the squared error function J. Given a set of k clusters $(C_1, C_2, ..., C_k)$ the k-means objective function would be determined by equation (2.6):

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{2.6}$$

where :

- μ_i represents the mean of the i-th cluster C_i with $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} |x|$
- $\|x - \mu_i\|^2$ is a chosen distance measure between a data point and the center of the i-th cluster, which is an indicator of the distance of the n data points considered from their respective center of one of the k cluster centers or means.

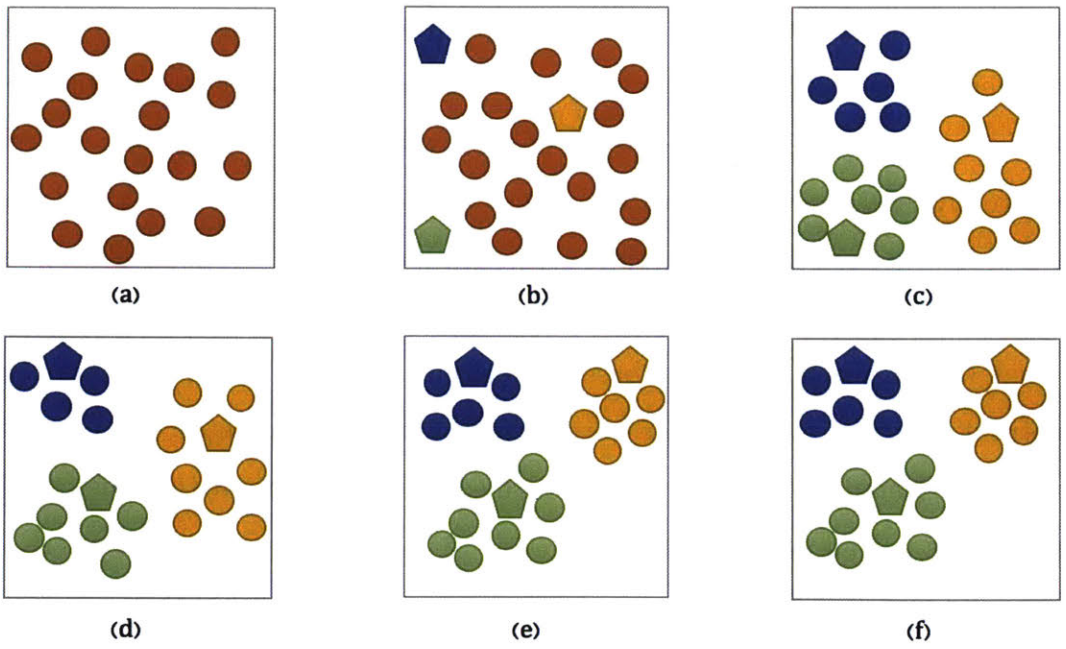

Figure 2-1: Principle behind the k-means clustering algorithm

(a) Original data set. (b) Initial cluster centers. (c) to (f) Three iterations of the k-means algorithm, during which each training example is assigned to its closest cluster centroid. The algorithm converges when no data point is moved to another cluster.

Algorithm 1 is the pseudo-code of the k-means clustering algorithm [9].

Algorithm 1. Pseudo-code for the k-means algorithm [9]

1: **Decide on a value k**
2: **Initialize k cluster centers** $\mu_1, \ldots, \mu_k$
3: **while stopping condition are not met do**
4: **for** $i = 1 : N$ **do**
5: $c_i := arg\,min_l d(x_i, \mu_l)$
6: **end for**
7: **for** $j = 1 : k$ **do**
8: $\mu_j := \frac{\sum_{i=1}^{N} 1\{c_{i=j}\} x_i}{\sum_{i=1}^{N} 1\{c_{i=j}\}}$
9: **end for**
10: **end while**
11: **return** $c_1, \ldots, c_N$ **and** $\mu_1, \ldots, \mu_k$

The first step in the algorithm consists of choosing the number of clusters k (**step 1**). This is a crucial and complex step in the algorithm which is explained in detail in the next section. The following step is to initialize k cluster centers (**step 2**). In order to do so, one technique to use is the Forgy method that was developed by Celebi et al. [9]. It randomly selects k observations from the data set and labels them as the initial centers. The third step is a while-loop (**step 3**) that runs until the algorithm converges. A widely-used stopping criterion for the k-means algorithm is the convergence of the algorithm. In addition, the k-means algorithm is NP-hard in general Euclidean space even for two clusters, hence the approximate solution algorithms as described above are generally used.

The algorithm converges when no change is noticed, either because no observations are assigned a different cluster center anymore (**step 5**) or because the variance did not improve by at least a pre-specified margin. The first for-loop (**steps 4-6**) describes the fact that each observation x_i is assigned to a label c_i indicating the cluster whose center μ_i has minimum distance to the observation. The distance is to be chosen prior to running the algorithm but is calculated internally.

Once the observations are assigned to a cluster, the second for-loop (**steps 7-9**) helps determining the new cluster centers μ_j by taking the average of all observations in a given cluster. Once the algorithm converges, it can return multiple outputs, among which the final cluster assignments c_1,...,c_N and the cluster centers μ_1, ..., μ_k.

The first for-loop requires a O(Nkn) calculation time, with N the number of n-dimensional data points and k the number of clusters. The second for-loop has a time complexity of O(Nk). Since both for-loops are repeated in the while-loop for an undetermined number of iterations I, the complexity of the algorithm is linear and of O($INkn$).

2.3.2 k-medoids algorithm

The k-medoids algorithm [10] is a clustering method that is related to k-means. While its aim is also to partition the data set, unlike k-means, k-medoids defines a prototype in terms of a "medoid". A medoid is by definition an actual data point and corresponds to the point that best represents most of the points of the cluster. The average dissimilarity between a medoid and the observations within the cluster it represents is thus minimal. Similarly, this algorithm is an NP-hard optimization problem: partitions can be determined through different heuristics. The most popular heuristic for k-medoids is the Partitioning Around Medoids (PAM) algorithm developed by Kaufman and Rousseeuw [11]. It is based on the search for k partitions of n objects and depends on a relevant choice of k medoids in a dataset. Each observation is then assigned to the medoid it is closest to. Figure 2-2 illustrates the clustering process using the k-medoids algorithm.

Given a set of k clusters $(C_1, C_2, ..., C_k)$ the k-medoids objective function consists of minimizing the total intra-cluster variance or the squared error function (2.7):

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - c_i\|^2 \tag{2.7}$$

where :

- c_i represents the center of the i-th cluster C_i, that is to say the centroid that minimizes the sum above.
- $\|x - c_i\|^2$ is a chosen distance measure between a data point and the center of the i-th cluster, which is an indicator of the distance of the n data points considered from their respective center of one of the k cluster centers or medoids.

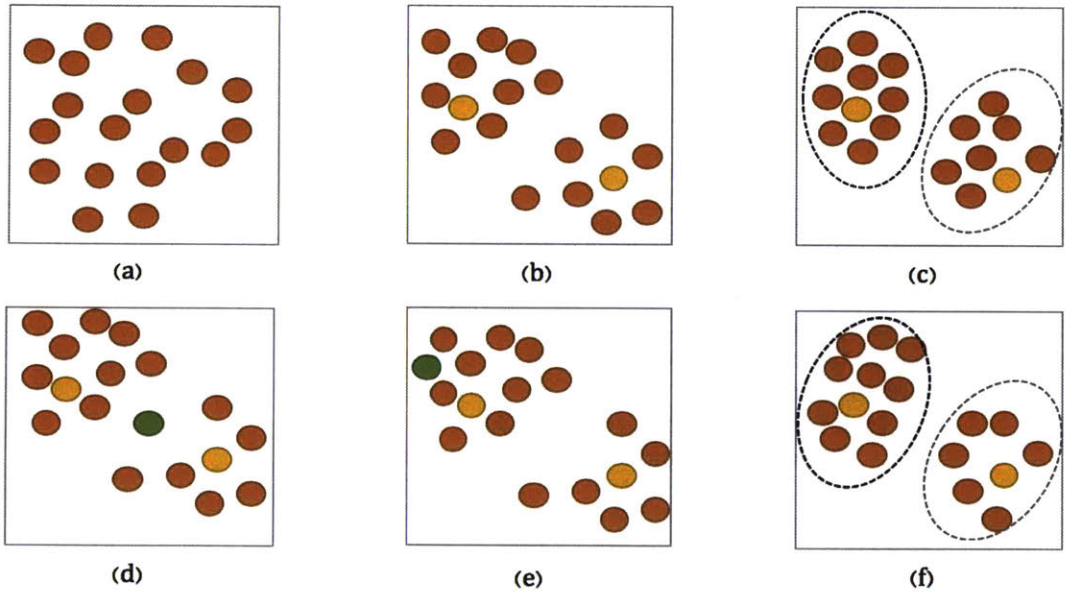

Figure 2-2: Principle behind the k-medoids clustering algorithm

(a) Original data set. (b) Choice of k = 2 initial medoids. (c) Each observation is assigned to its nearest medoid (yellow data point). (d) Random selection of a non-medoid object (green observation). After computing the total cost of swapping, if the quality is improved, the point is swapped from one cluster to another. (e) to (f) Process iterated as long as the quality is improved and stopped if no change is observed.

Algorithm 2 [11] presents the pseudo-code for the Partitioning Around Medoids (PAM) algorithm. PAM has two phases. The first is the building phase (**steps 1 to 2**), where a collection of k objects are selected similarly to those in the initialization part of the k-means method. Those objects are the initial medoids. Each data point is assigned to its closest medoid through an iterative process. The second phase consists of swapping the set of objects. For each pair of medoid and non-medoid objects (**steps 4 to 5**), the total swapping cost is determined (**steps 3 to 11**).

If this cost is indeed minimal, the medoid and non-medoid data points are swapped (**steps 6 to 7**). The non-selected objects are associated with the most similar representative object. The procedure is repeated until the quality of the clustering is consequently improved. Unlike k-means, k-medoids converges regardless of the distance metrics used. Because the medoids are real data points, only the distances between the observations are needed to run the algorithm.

Algorithm 2. Pseudo-code for the k-medoids algorithm [11]

```
1: Decide on a value k
2: Randomly choose k observations as the initial medoids
3: while no change in the centroid assignment do
4:     for each medoid c do
5:         for each non-medoid observation o do
6:             if swapping c and o improves the solution then
7:                 Swap c and o
8:             end if
9:         end for
10:    end for
11: end while
12: return the k medoids and the cluster assignment
```

At every iterative step, for all k medoids, there are n-k potential swaps. Thus, the swapping cost for one swap is O(n-k). Each iteration has a time complexity of $O(k(N-k)^2)$. Given I iterations, the complexity of the whole algorithm is therefore $O(Ik(N-k)^2)$.

2.3.3 CLARA algorithm

The PAM heuristic for the k-medoids algorithm is not well-suited for large data sets. It is indeed not easily scalable. This is the reason why a sampling-based method called Clustering LARge Applications (CLARA) was implemented. CLARA [12] draws a sample from the large dataset and applies the PAM algorithm to determine an optimal set of medoids. For each sample, the objective function is minimized. It corresponds to the total intra-cluster variance or the squared error function, as defined by equation (2.7).

CLARA repeats the sampling and clustering procedures a given number of times until the minimal cost is obtained through the iterations. The corresponding clustering is then selected as the final clustering result. Because CLARA is based upon a sampling method, the quality of the clustering results highly depends on the size of the chosen samples. If the sample size is small, the efficiency of CLARA comes at the cost of the quality of the clustering result.

Figure 2-3 shows the process behind the CLARA algorithm. A number of z samples are drawn from the whole data set and PAM is applied on each one of them. The best clustering result is the one that is finally returned.

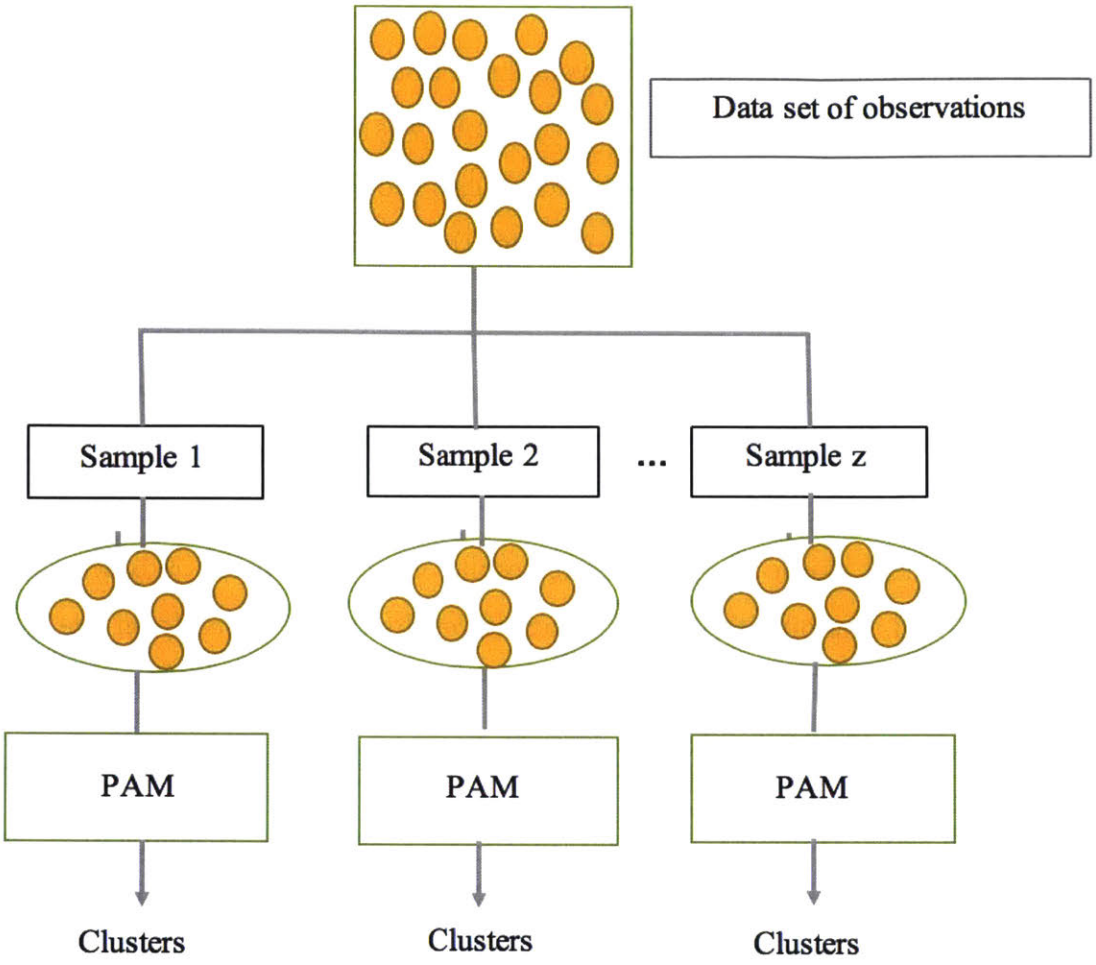

Figure 2-3: Principle behind the CLARA algorithm

Algorithm 3 [12] presents the pseudo-code of the CLARA algorithm. The goal of the CLARA algorithm is to apply PAM to each one of the S random samples of observations of size z drawn from the data set (**steps 1 to 3**). The PAM algorithm is computed on each of those subsets and k medoids are chosen within this subset (**step 4**). Once PAM is run on a sample, the remaining N-z observations are assigned to the medoid they are the closest to (**step 5**). The total sum of distances between all the observations and their closest medoid are then calculated (**step 6**).

If it is a minimized value, the current assignment is considered to be the best clustering result (**steps 7 to 9**). In total, there are therefore S different clustering configurations and the best of those is returned by CLARA (**step 11**).

Algorithm 3. Pseudo-code for the CLARA algorithm [12]

```
1: Decide on values for S, z and k
2: for s = 1: S do
3:     Draw z random observations from the data set to create a sample
4:     Apply PAM on the sample
5:     Assign the N-z observations that are not in the sample to their closest medoid
6:     Calculate the total sum of distances between all observations and their closest medoid
7:     if total sum of distances is lowest found so far then
8:         Update the best clustering assignment to be the current assignment
9:     end if
10: end for
11: return the best clustering assignment
```

Considering z the size of the sample, k the number of clusters and n the number of objects, the complexity of every iteration is $O(kz^2 + k(n-k))$. The number of iterations is the number of samples, that is to say S. Consequently, the time complexity of the algorithm is $O(kSz^2 + kS(n-k))$ and choosing the right values for S, m and k is paramount. Low values allow faster computational time but at the cost of a decrease in the quality of the clustering.

2.3.4 Summary

Partitioning-based clustering algorithms minimize a given clustering criterion by moving the data points within clusters through an iterative procedure. This iterative optimization problem starts with an initial partition. The data points are assigned to the clusters until the convergence of the algorithm is reached.

Tables 2-1 and 2-2 summarize the advantages and drawbacks of all the partitioning clustering algorithms presented in this chapter.

Table 2.1: Advantages of the partitioning clustering methods

Clustering methods	Advantages
k-means	Fast Robust Easy to implement
k-medoids (PAM)	More robust than k-means Efficient for small data sets
CLARA	Well-suited for larger data sets

Table 2.2: Drawbacks of the partitioning clustering methods

Clustering methods	Drawbacks
k-means	Need to specify the number of clusters k Initial choice of the clusters centers Sensitivity of the data to scalability Only applicable if the mean of a cluster is defined Not suitable for clusters of very different sizes
k-medoids (PAM)	Need to specify the number of clusters k Not good scalability for large data sets Costlier than the k-means method
CLARA	Efficiency dependent on the sample size If the sampling is biased, no good clustering

2.4 Hierarchical clustering

Hierarchical cluster analysis [13] aims at building a hierarchy of clusters. The resulting cluster tree or *dendrogram* is a multilevel structure that represents the identified groups or clusters in the data set. There are two hierarchical clustering methods:

- **Agglomerative**: known as a "bottom-up" approach and sometimes called "Agglomerative Nesting" (AGNES). Every data point starts in its own cluster of one (or leaf) and new clusters are defined by iteratively **merging** the different observations while moving up the hierarchy until there is just one big cluster (or root).

- **Divisive**: known as a "top-down" approach and is also referred to as a "Divisive Analysis" (DIANA). All observations start in the same cluster and they are iteratively **split** into multiple clusters while moving down the hierarchy.

One important feature is that agglomerative clustering [14] is well-suited to identify small clusters while divisive clustering [15] is good at determining large clusters. These two approaches are done in a greedy manner. In fact, those algorithms aim at making local optimal choices at each stage in order to find a global optimum. Even if a global optimum is sometimes not found, the trade-off between the quality of the local optimum and the computational time justifies the use of such heuristics. Figure 2-4 shows the difference between the agglomerative and divisive hierarchical algorithms and illustrate how the dendrogram is built for both approaches.

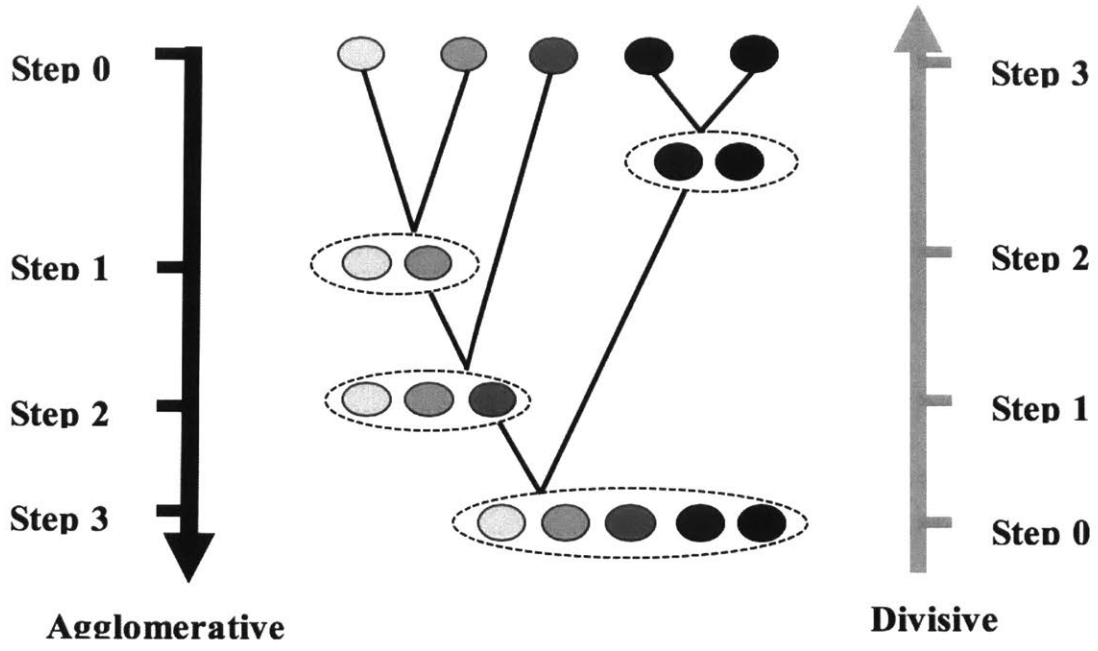

Figure 2-4: Principle behind the agglomerative and divisive clustering algorithms

Each circle represents an observation in the data set.

Various cluster agglomeration methods have been developed in order to measure the dissimilarities between two clusters of observations. They are based on the implementation of a linkage criterion [16]. Some of those approaches of distance measurement, named d, between two clusters of observations A and B include:

- **Maximum or complete linkage clustering:** the distance between two clusters is the *largest* value of the dissimilarities between all the possible combination of an element in cluster 1 and one in cluster 2. This distance is defined by:

$$max\left\{d(a,b) : a \in A, b \in B\right\} \tag{2.8}$$

- **Minimum or single linkage clustering:** the distance between two clusters is the *smallest* value of the dissimilarities between all the possible combination of an element in cluster 1 and one in cluster 2. This distance is defined by:

$$min\{d(a,b) : a \in A, b \in B\} \tag{2.9}$$

- **Mean or average linkage clustering:** the distance between two clusters is the *average* value of the dissimilarities between all the possible combination of an element in cluster 1 and one in cluster 2. This distance is defined by:

$$\frac{1}{|A|\,|B|}\sum_{a\in A}\sum_{b\in B} d(a,b) \tag{2.10}$$

- **Centroid linkage clustering:** the distance between two clusters corresponds to the dissimilarity between the centroid of the cluster 1 and that of cluster 2. This distance is defined by:

$$d(c_S, c_T) \tag{2.11}$$

  where c_S and c_T are the centroids of the clusters S and T

- **Wards minimum variance method:** the total within-cluster variance is minimized so that two clusters with minimum between-cluster distance are merged at each step.The squared Euclidean distance between points is defined by:

$$d(a \in A, b \in B) = \|a - b\|^2 \tag{2.12}$$

Though the single-link clustering is optimal, it has a high computational time compared to the other methods.

The visualization output of the hierarchical clustering algorithms is a dendrogram. It is also referred to as a tree and shows the similarity of grouped clusters. Every level k corresponds to the step of clustering the data points with n-k+1 clusters. If the samples are in the same cluster at level k, they do not change the cluster they belong to in higher levels.

2.4.1 Agglomerative clustering

Agglomerative clustering consists of merging very similar data points and iteratively building larger clusters from a combination of smaller clusters. It enables to produce a family of clusters that are illustrated thanks to dendrogram.

The algorithm, as shown in Algorithm 4 [14], follows a bottom-up approach that relies on an active set of clusters of observations. At each stage of the algorithm, two clusters are chosen to be merged. The active set starts as being empty (**step 2**). Through an iterative process, all the data points are initially defined as their own cluster (**steps 3 to 4**). The active set of clusters is progressively used to build the dendrogram by successively merging the clusters (**step 6**). Until the active set only contains one data point (**step 7**), a pair of two mergeable groups with best distance in the active set is chosen (**step 8**). Each element of the pair is removed from the active set (**step 9**) and added back as a union and new cluster to that same active set (**step 10**). This union is also added to the clustering tree (**step 11**). A dendrogram is the highlighted visualization output of the algorithm that has a time complexity of $O(n^3)$.

Algorithm 4. Pseudo-code for the Agglomerative algorithm [14]

1: **Input:** Data set of observations $\{x_n\}_{n=1}^{N}$
2: Å $\leftarrow \emptyset$
3: **for** n = 1 : N **do**
4: Å $\leftarrow$ Å $\cup \{\{x_n\}\}$
5: **end for**
6: T $\leftarrow$ Å
7: **while** |Å| > 1 **do**
8: $\mathcal{G}_1^*, \mathcal{G}_2^* \leftarrow \arg_{\mathcal{G}_1, \mathcal{G}_2 \in \text{Å}}$ min distance$(\mathcal{G}_1, \mathcal{G}_2)$
9: Å $\leftarrow$ (Å $\setminus \{\mathcal{G}_1^*\} \setminus \{\mathcal{G}_2^*\}$
10: Å $\leftarrow$ (Å $\cup \{\mathcal{G}_1^* \cup \mathcal{G}_2^*\}$
11: T $\leftarrow$ (T $\cup \{\mathcal{G}_1^* \cup \mathcal{G}_2^*\}$
12: **end while**
13: **Output:** Tree T

2.4.2 Divisive clustering

Divisive clustering [15] is a top-down procedure that starts with all the data in the same cluster. That data is then successively divided into subgroups. It thus has an opposite process to that of the agglomerative hierarchical algorithm.

The algorithm, as shown in Algorithm 5 [15], takes a set of clusters of observations as an input. It aims at applying the function SUBDIVIDE to the whole data set (**step 16**). This function (**steps 3 to 14**) consists of dividing the data into small clusters (**step 4**) through an iterative process. Through a loop over the resulting partitions (**step 6**), if the chosen point of the group is a singleton (step 7), it is added to the set of sets (**step 8**). Otherwise, if it is a group (**step 9**), it is subdivided into small clusters (**step 10**) that are added to the set of sets that corresponds to the final output (**step 13**). The algorithm has a time complexity of $O(n^3)$.

Algorithm 5. Pseudo-code for the Divisive algorithm [15]

```
1: Input: Data set of observations {x_n}_{n=1}^N
2:
3: function SUBDIVIDE(G, K)
4:         {H_k}_{k=1}^K ← CLUSTER(G, K)
5:         S ← ∅
6:       for k = 1 : K do
7:         if |H_k| = 1 then
8:           S ← S ∪ {H_k}
9:         else
10:            S ← S ∪ SUBDIVIDE{H_k, K}
11:        end if
12:    end for
13:      Return: S
14: end function
15:
16: Output: SUBDIVIDE({x_n}_{n=1}^N, K)
```

2.4.3 Summary

Hierarchical-based clustering algorithms move the data points within clusters through an iterative procedure that can be bottom-up (agglomerative) or top-down (divisive). It presents some advantages over partitioning algorithms, such as the fact that there is no need to specify the number of clusters beforehand. However, it is also expensive computationally.

Table 2-3 summarizes the advantages and drawbacks of both the agglomerative and divisive hierarchical clustering algorithms presented in this chapter.

Table 2.3: Advantages and Drawbacks of the hierarchical clustering methods

Advantages	Drawbacks
No information on the number of clusters required Easy implementation No sensitivity to the choice of distance metric	High time complexity Once a cluster is formed, it is not possible to go back and undo it Sensitivity to the type of distance matrix: - Large clusters - Difficulty to handle different sizes of clusters - Noise Difficulty to identify the clusters on the dendrogram

2.5 Fuzzy clustering: fuzzy C-means algorithm

Developed by Dunn [17], fuzzy clustering has been labeled as a soft cluster analysis procedure. Each element has a given probability of being in each cluster that are built. In fact, unlike other clustering methods, every element of the clustered data set does not belong to one cluster only but to a set of clusters. Each data point has a set of membership coefficients, with every coefficient corresponding to the probability or degree to which the point belongs to a certain cluster. If this point is close to the center of a given cluster, it belongs to that cluster to a higher degree than other points situated in the edge of the cluster. The degree to which a data point is in a cluster is a value between 0 and 1. The fuzzy C-means (FCM) algorithm [17] is among the most widely used fuzzy clustering methods. The center of a cluster is determined by calculating the mean of all the points of the cluster, weighted by the degree of a point of belonging to the cluster.

Considering a data set that is divided into two clusters C_1, C_2, the membership coefficient of each point is represented for both algorithms. For the k-means algorithm, a data point is part of either the cluster C_1 (membership coefficient of 1) or the cluster C_2 (membership coefficient 0) and to one cluster only. On the other hand, for the fuzzy c-means algorithm, a data point can belong to one (membership coefficient of either 0 or 1) or both clusters (membership coefficient of a value between 0 and 1).

The fuzzy C-means algorithm is an optimization problem. A N-dimensional data set Y = $\{y_1, y_2, , y_N\}$ can be fuzzy c-partitioned. Given c number of clusters, with $2 \leq c \leq N$, m the weighting exponent, with m 1, the resulting matrix U = $\{u_i(y_k)\}$ = $\{u_{ik}\}$ that represents the partition Y_i is defined by:

$$u_{ik} = \begin{cases} 1, \text{if} \quad y_k \in Y_i \\ 0, \quad \text{otherwise} \end{cases}$$

$$\text{J} = \sum_{i=1}^{N} u_{ik} > 0 \text{ for all i}$$

$$\text{J} = \sum_{i=1}^{N} u_{ik} = 1 \text{for all k} \quad (2.13)$$

In order to identify the optimal fuzzy c-partitions for a given data, a clustering criterion that has been developed consists of minimizing the generalized least-squared errors function:

$$J(U, \nu) = \sum_{k=1}^{N} \sum_{i=1}^{c} u_{ik}^{m} \|y_k - \nu_i\|^2 \quad (2.14)$$

- The degree of membership of y_i in the j^{th} cluster is defined by :

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \frac{\|y_i - \nu_j\|}{\|y_i - \nu_k\|}^{\frac{2}{m-1}}} \quad (2.15)$$

- $\nu = (\nu_1, \nu_2, , \nu_c)$ represents the vector of centers where the center of the i^{th} cluster can be determined by:

$$\nu_j = \frac{\sum_{i=1}^{N} u_{ij}^m y_i}{\sum_{i=1}^{N} u_{ij}^m} \tag{2.16}$$

- $\nu_i = (\nu_{i1}, \nu_{i2}, , \nu_{ic})$ represents the center of the cluster i
- $\|y_i - \nu_j\|^2$ is a chosen distance measure between a data point and the cluster center

The algorithm, as shown in Algorithm 6 [17], starts with an initialization of the fuzzy c-partition (**step 2**) randomly. Through an iterative process, the cluster centers are progressively updated (**steps 3 to 4**). At the k^{th} step, the center vectors are obtained and the membership matrix is then updated (**step 5**). Those steps are repeated until the input threshold value is reached.

Algorithm 7. Pseudo-code for the k-means algorithm [17]

1: Input: data Y, centroid matrix υ, weighted exponent of fuzzy membership m, threshold ε used as stopping criterion
2: Randomly initialize the fuzzy partition U $^{(0)}$
3: while $\max_{ij} \left\| u_{ij}^{(k)} - u_{ij}^{(k+1)} \right\| > \varepsilon$ **do**
4: Calculate the cluster centers υ with U $^{(k)}$
5: Update the membership matrix U $^{(k+1)}$
6: return the vector of centers υ

Given n the number of points in the data set, N the number of n-dimensional data points, c the number of clusters and I the number of iterations needed to reach the threshold, the time complexity of the fuzzy c-means algorithms is of $O(nNc^2I)$.

Table 2-4 summarizes the advantages and drawbacks of the fuzzy C-means algorithm.

Table 2.4: Advantages and drawbacks of the fuzzy clustering

Advantages	**Drawbacks**
Well-suited for overlapped data set Unlike other partitioning algorithms, a data point doesnt need to exclusively belong to one cluster only	Specification of the number of clusters needed Euclidean distance measures might unequally weight the underlying factors

Chapter 3

Cluster implementation in R

In this chapter, the cluster analysis algorithms presented in the previous chapter are implemented. The studied data is that collected from the plasma etcher. The aim of the analysis is to cluster the data points for the purpose of detecting anomalies in this manufacturing process. In order to illustrate the principles behind the clustering algorithms in a simple and visual way, a univariate analysis is conducted. In addition, a multivariate analysis is presented to successfully detect anomalies in large data sets.

3.1 Cluster validation

Cluster validation refers to the procedure of evaluating the quality of the results of clustering algorithms. Two cluster validation methods have been presented in this section:

- **Internal cluster validation:** uses the information of the clustering process internally in order to evaluate the quality of the clustering without any additional information.
- **Stability measures:**: evaluates the consistency of the clustering result by comparing this result with the clusters built.

In addition to these methods, the concept of confusion matrix is introduced and used to evaluate the quality of the clustering results.

3.1.1 Internal measures

Internal validation consists of comparing the clustering result with itself only. It relies on the three features of the cluster partitions:

- **Compactness:** refers to how close the elements within a cluster are. The lower the within-cluster variation is, the more compact the given cluster is.
- **Separation:** corresponds to how distinct the clusters from one another. In order to estimate the separation metric, the distances between cluster centers or the pairwise minimum distances between objects in different clusters can be calculated.
- **Connectivity:** consists of estimating how the elements are placed in the same cluster as the data points they are closest to in the data set. The connectivity varies from 0 to infinity and has to be minimized.

When conducting the internal clustering validation, both the compactness and separation are paramount. Two commonly used indices to determine the quality of clustering are the silhouette width and the Dunn index.

The Dunn index

Given two clusters C_k and $C_{k'}$ that contain the observations x_i^k and $x_i^{k'}$, the distance $d_{kk'}$ between those two clusters is determined by the distance between their closest points, such that:

$$min_{i \in C_k, j \in C_{k'}} \left\| x_i^{(k)} - x_j^{(k')} \right\| \tag{3.1}$$

The minimum of those distances is called d_{min}.

For every cluster C_k, D_k is the largest distance separating two distinct points within the cluster, also referred to as the diameter of the cluster:

$$max_{i \in C_k, j \in C_{k'}, i \neq j} \left\| x_i^{(k)} - x_j^{(k')} \right\| \tag{3.2}$$

The maximum of those distances is d_{max}.

The Dunn index D is defined as the ratio:

$$D = \frac{d_{min}}{d_{max}}. \tag{3.3}$$

If the data set clustering consists of compact and well-separated clusters, the diameter of the clusters is small and the distance between the clusters needs to be large since they have to be very distinct.

The Silhouette coefficient

The silhouette analysis shows the quality of the clustering and estimates the average distance between clusters. For each observation i of the data set, the silhouette width s_i is determined by:

$$s_i = \frac{(b_i - a_i)}{max(a_i, b_i)} \tag{3.4}$$

where:

- a_i is the average dissimilarity between the observation i and the other points of the cluster it belongs to.
- b_i is the dissimilarity between the observation i and its neighbor cluster, the one it is closest to but does not belong to.

The silhouette width varies from -1 to 1 and shows how similar the observation is to the other objects in its own cluster in comparison with those in the neighbor cluster. Observations with a large silhouette width (close to 1) are often very well clustered: they are similar to the ones in the cluster they belong to. In contrast, those with a small value are poorly clustered: those observations are close to not one but two distinct clusters. Reassessing whether that observation belongs to one or the other cluster might be needed.

3.1.2 Stability measures

Cluster stability measures include:

- **the average proportion of non-overlap (APN):** measures the average proportion of observations that are not in the same cluster by comparing the clustering of the whole data with that obtained by removing one column of the data set.
- **the average distance (AD):** corresponds to the average distance between the observations within a cluster for both the whole data and the data with one column removed.
- **the average distance between means (ADM):** measures the average distance between the centers of the clusters for observations in the same cluster for both the whole data and the data with one column removed.
- **the figure of merit (FOM):** measures the average intra-cluster variance of the column removed from the data set whilst clustering the data without the column removed.

All stability metrics vary between 0 and 1. The smaller the values, the higher the quality of the clustering.

3.1.3 Confusion matrix

A confusion matrix [18] aims at describing the performance of a model on a set of test data. It is a table that consists of information on the actual/expected results and the predicted ones. This matrix indeed reflects how good the observations illustrate the actual events and is shown in Table 3.1.

Table 3.1: Confusion Matrix

		Predicted	
		Yes	**No**
Actual	**Yes**	True Positives (TP)	False Negatives (FN)
	No	False Positives (FP)	True Negatives (TN)

True Positives (TP) correspond to the number of correct predictions of truly positive instances while True Negatives (TN) are the number of correct predictions of truly negative instances. False Positives (FN) are observations that are in fact negative when predicted as positive. False Negatives (FN) are observations labeled as positive but are predicted as being negative.

Based on the entries of the table, standard metrics can be defined:

- **the recall** or sensitivity is the proportion of positive observations correctly identified as such out of all the actually positive observations (whether correctly predicted as such or not) and defined by:

$$recall = \frac{TP}{TP + FN} \tag{3.5}$$

  Recall determines the accuracy of the outlier detection, with a value of 1 that corresponds to the detection of all anomalies.

- **the precision** is the proportion of the predicted positive observations that were indeed correctly identified as such out of all the observations labeled as positive (whether or not they actually are positive) and defined by:

$$precision = \frac{TP}{TP + FP} \tag{3.6}$$

The higher the precision, the better the performance of the algorithms is since the rate of false positive is low. A precision of 1 refers to the fact that there are no false positives.

- **the F1 score** is a weighted harmonic average of both the recall and the precision. It takes both false positives and false negatives into account and is useful when it comes to uneven class distributions. It is determined as:

$$F - 1score = \frac{2 * recall * precision}{recall + precision} \tag{3.7}$$

- **the accuracy** is the proportion of the total number of predictions that were correctly predicted among the total observations. It is defined by:

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{3.8}$$

The higher the accuracy, the better the performance of the algorithms.

3.2 Data preparation

In this section, the preparation of the data before applying the clustering algorithms is presented. In addition to the explanation of the pre-processing of the data, the Principal Component Analysis (PCA) is explained.

3.2.1 Pre-processing of the data

The data of interest is prepared for analysis using the Preprocessing Toolbox developed by the team. The pre-processed data consists of a list of data frames, each of which represents a cycle through 31 parameters. Each data frame has 31 columns for all the parameters.

The parameters extracted for analysis are the following: Time, Bot_Match_ LoadCap_Mon, Bot_Match_TuneCap_Mon, Bot_RF_RevPwr_In, Gas_01_Flow_Mon, Gas_02 _Flow_Mon, Gas_03_Flow_Mon, Gas_04_Flow_Mon, Gas_05_Flow _Mon, Gas_06_ Flow _Mon, Gas_07 _Flow_Mon, Gas_08_Flow_Mon, Gas_09_Flow_In, Gas_09_Pressure _Mon, PM_ RecipeStepNum, ProcChm_Bot_Elec_Temp_Mon, ProcChm_ESC_Voltage _In, ProcChm_EndPt_ChanC_In, ProcChm_EndPt_Step_Time, ProcChm_Mano_Pres _In, ProcChm_Pres_Vlv_Angle, ProcChm_Pressure_Mon, Gap_Position, ProcChm_Rcp_ Tuner_Tap, TCP_Elec_Temp_In, ProcChm_Vacuum_DI, ProcChm_At_ATM_DI, ProcChm _Mano _RefrnPres_In, Timer_23_Resettable_Value and Timer_24_Resettable_Value.

However, the most paramount parameters considered in this project are:

- **BOT_RF_RevPwr_In:** Radio Frequency (RF) power is used in order to ionize the gas generating a plasma in plasma etching. The RF power is part of a recipe for the particular etch. Forward power is delivered to the load and reflected power is reflected back to the supply. An ideal reflected power would be 0 watts but because of inefficiencies of the RF match network or the physics of the chamber, reflections occur which result in losses. This reflected power indicates the amount of how efficiently the power is being delivered to the chamber or the load.

- **ProcChm_Bot_Elec_Temp_Mon:** the temperature of the wafer during the processing of the wafer that sits on a chuck.
- **ProcChm_EndPt_ChanC_In:** a signal that is expressed in counts of the intensity of the plasma at a specific wavelength. Once the target material that is etched is completed, another layer is exposed which results in a change in the spectrum along with a change in amplitude at the endpoint signal, which is used to determine if the etching process is finished.

A "Visualization Toolbox" has been implemented in order to plot the parameters and cycles of interest using the R software. It consists of various functions that enable to plot specific parameters for specific cycles.

3.2.2 Principal Components Analysis (PCA)

Given an analysis which includes a substantial number of correlated variables, a **Principal Component Analysis (PCA)** [19] can be applied in order to reduce the dimensions in a large set of data. PCA, however, retains the trends and patterns of the data. The goals of the PCA are:

- extraction of the main features in a data set
- reduction of the dimensionality of the data
- simplification of the description of the data set
- analysis of the structure of the observation sand variables

It consists of transforming a number of correlated variables into a smaller number of uncorrelated variables. Those are known as principal components and are defined as the linear combinations of the original variables in the data set. The first of those components account for as much of the variability in the data as possible and the remaining variability is taken by the other components. For the purpose of visualizing the clustering results, PCA is applied and the results are plotted on graphs where the axes are defined as the two main principal components.

3.3 Univariate analysis

In this section, the univariate analysis is conducted in order to illustrate the various clustering methods presented. Only one parameter of interest is considered for this analysis. The ultimate aim in dividing the data into clusters is to detect anomalies in the production of wafers using the plasma etcher.

A paramount step in performing a cluster analysis is to prepare and pre-process the data. The quality of the clustering results indeed highly depends on this first step. For the purpose of the cluster analysis, given a number C of cycles studied through S number of observations, a data frame of dimensions $C \cdot S$ is created where every row represents a cycle, every column is an observation and a cell is an observation of the parameter for a given cycle. This structure is illustrated in Table 3.2. Before proceeding with the cluster analysis, a requirement is for all the constant columns (variance of zero) to be removed.

Table 3.2: Structure of the data frame for the univariate analysis

Cycle	**Parameter**
1	S observations of the parameter for cycle 1
2	S observations of the parameter for cycle 2
3	S observations of the parameter for cycle 3
...	...
C-3	S observations of the parameter for cycle C-3
C-2	S observations of the parameter for cycle C-2
C-1	S observations of the parameter for cycle C-1
C	S observations of the parameter for cycle C

3.3.1 Example

In this example of a univariate analysis, the chosen parameter is called the "ProcChm_EndPt_ChanC_In". A number of 667 observations of this parameter are collected for all 11 cycles.

In Figure 3-1, all 11 cycles (081743, 082317, 082849, 083423, 083957, 084619, 085152, 085724, 090255, 090826, 091459) are plotted against time for the parameter "ProcChm_EndPt_ChanC_In". An analysis of the plot by ADIs process engineers, as shown in Figure 3-1, underlines the fact that some of the cycles are anomalous and others highly anomalous since they present additional anomalous variations.

- The cycles 084619 and 091459 are **highly anomalous**.
- The cycles 082317, 082849, 083423, 083957, 085724 and 090255 are **anomalous**.
- The cycles 081743, 085152 and 090826 are **not anomalous**.

For the purpose of the cluster analysis, a data frame of dimensions 667 x 11 is created, as illustrated in Table 3.3, where every row represents a cycle, every column is an observation and a cell is an observation of the parameter for a given cycle.

Table 3.3: Structure of the data frame for the univariate analysis example

Cycle	Parameter : ProcChm_EndPt_ChanC_In
81743	667 observations of the parameter for cycle 81743
82317	667 observations of the parameter for cycle 82317
82849	667 observations of the parameter for cycle 82849
83423	667 observations of the parameter for cycle 83423
83957	667 observations of the parameter for cycle 83957
84619	667 observations of the parameter for cycle 84619
85152	667 observations of the parameter for cycle 85152
85724	667 observations of the parameter for cycle 85724
90255	667 observations of the parameter for cycle 90255
90826	667 observations of the parameter for cycle 90826
91459	667 observations of the parameter for cycle 91459

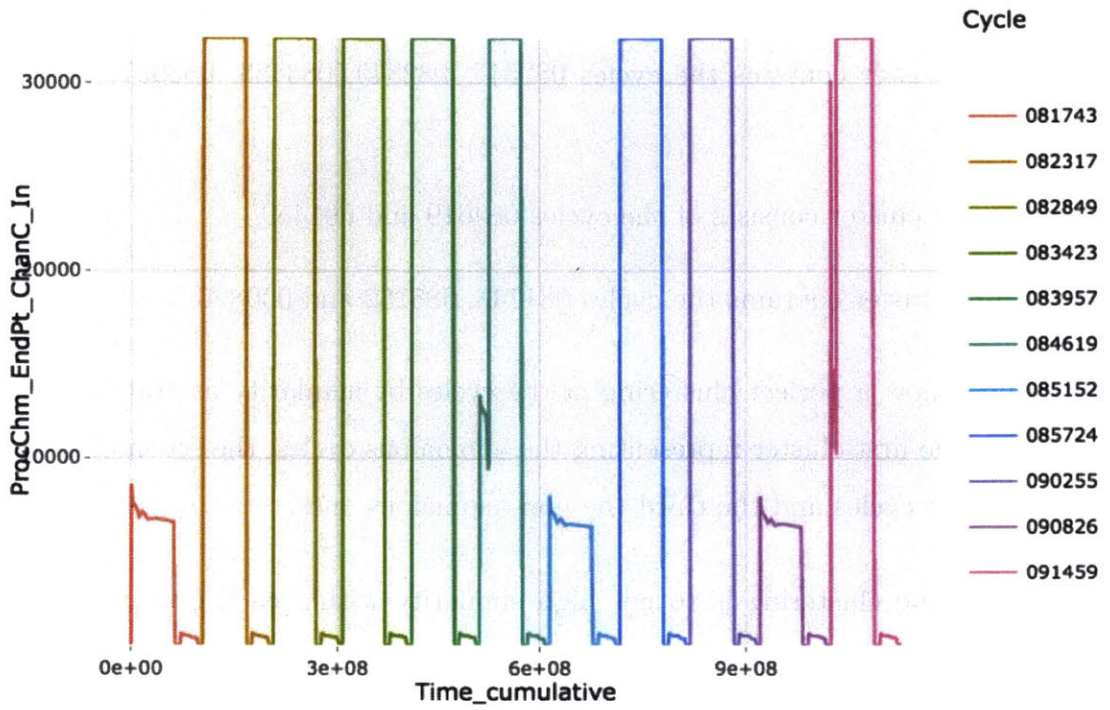

Figure 3-1: Plot of the parameter "ProcChm_EndPt_ChanC_I" for all 11 cycles

3.3.2 Partitioning clustering results

In this section, all the clustering algorithms presented in the previous chapter are implemented and the results are discussed.

K-means algorithm

Once the number of optimal clusters k is specified (in that case, $k = 3$), the algorithm is computed. It only takes **1.175 seconds** to run. The clustering results are also depicted visually after having applied the PCA, as illustrated in Figure 3-2.

The analysis results in three clusters of respective sizes of 6, 2 and 3:

- The first cluster contains the cycles 082317, 082849, 083423, 083957, 085724 and 090255
- The second cluster consists of the cycles 084619 and 091459
- The third cluster contains the cycles 081743, 085152 and 090826

The results show a perfect clustering of the cycles by similarity as stated in the example, with the first cluster representing the anomalous cycles, the second one the highly anomalous cycles and the third the non-anomalous ones.

The goal of the clustering is to get high similarity within each group and low similarity between each group. This means that the within-cluster variance needs to be low whilst the variance between the cluster needs to be high. The ratio of the latter by the first is to be maximized. In this example, the ratio is **96.8%** which indeed assesses the high quality of the clustering as shown by the results.

The internal measure cluster validation shows that the value of the Dunn index for this clustering result is of **1.29**. The silhouette widths of clusters 1, 2 and 3 are respectively, **0.81**, **0.32** and **0.97**. The average silhouette width is of **0.77** and thus is close to 1, which characterizes a very well clustered result.

All in all, the k-means implementation for a univariate analysis has proven to be highly efficient. Not only is the computational time low, the clusters are well-defined which implies an overall clustering of high quality.

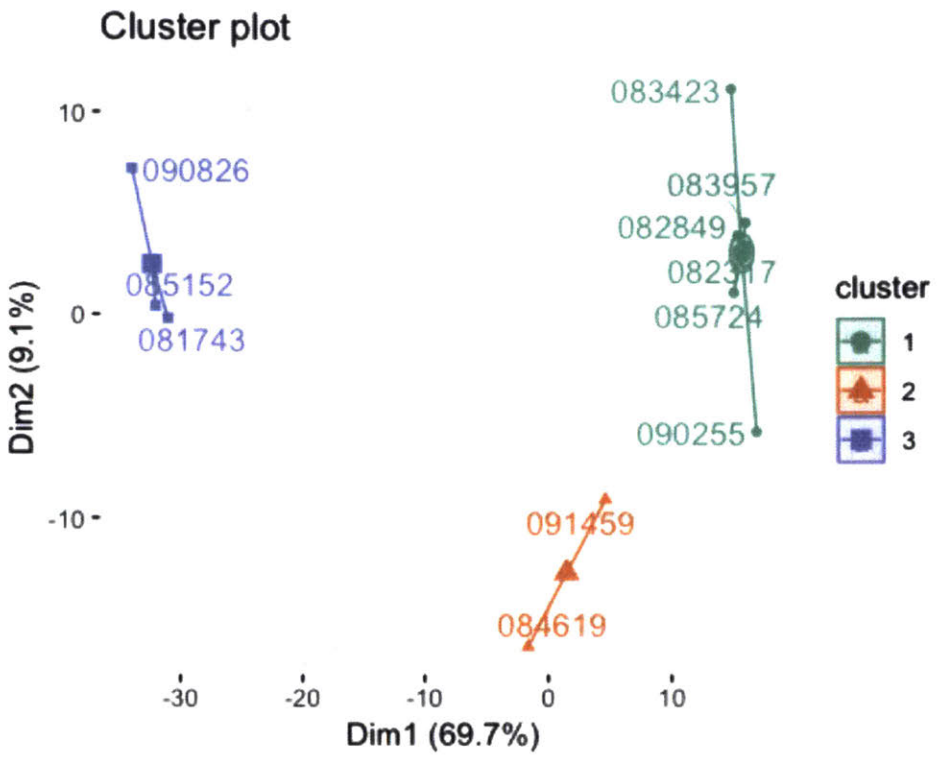

Figure 3-2: Scatter plot of the clusters from the univariate k-means algorithm

K-medoids algorithm

For the k-medoids algorithm to be computed, the number of clusters k has to be specified as $k = 3$. Its computational time is of **0.935 seconds**. It is thus faster than the k-means algorithm. The clustering result is computed visually in Figure 3-3 to show the built clusters.

The cluster analysis results are similar to the ones given by the k-means analysis. It consists of the same three clusters of respective sizes of 3, 6 and 2 as the ones obtained using the k-means algorithm. The clustering result is thus compliant with the analysis performed for k-means. The cycles are successfully clustered by similarity. The first cluster represents the non-anomalous cycles, the second one anomalous cycles, and the last contains the highly anomalous cycles.

The internal measure cluster validation shows that the Dunn index for this clustering result is **1.29**. The silhouette widths of clusters 1, 2 and 3 are respectively **0.97, 0.81** and **0.32**. The average silhouette width is again **0.77**, and thus close to 1. This also characterizes a very good clustering result, similarly to the k-means clustering result.

All in all, the k-medoids implementation for a univariate analysis has proven to be of high quality. It is also less computationally expensive than the k-means algorithm.

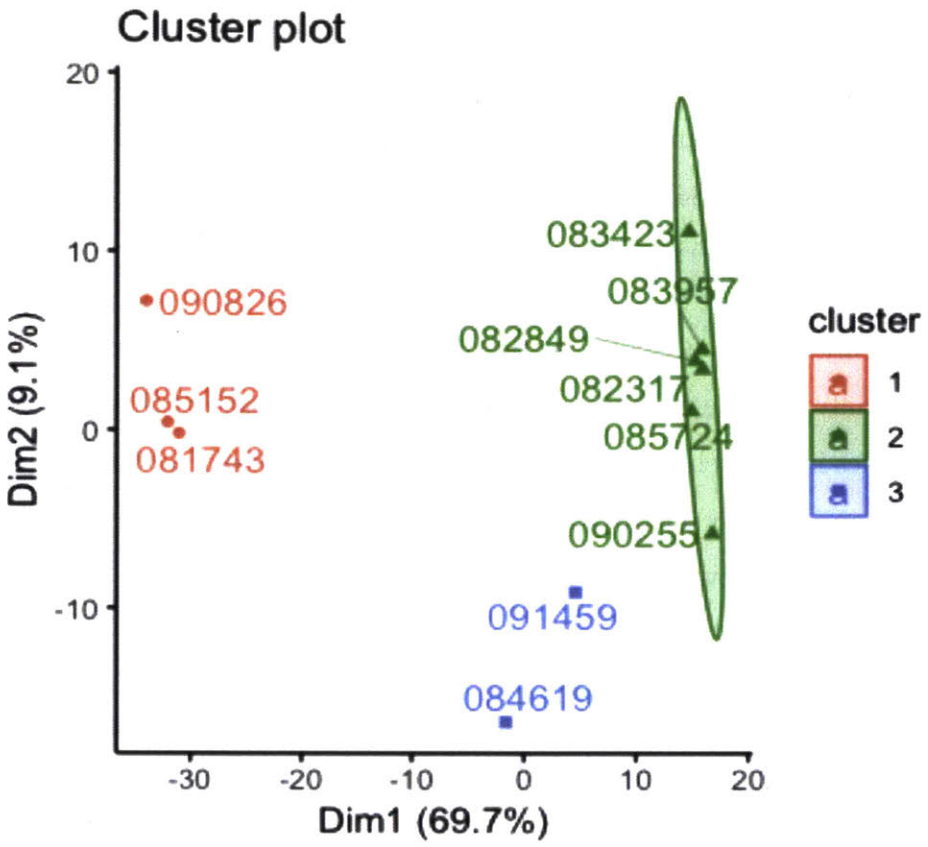

Figure 3-3: Scatter plot of the clusters from the univariate k-medoids algorithm

CLARA algorithm

The CLARA algorithm is a version of the PAM algorithm that is well-suited for large data sets. Since the analysis performed is for one parameter only, though the computational time of 0.884 slightly lower than that of the k-medoids, the output result is similar. The clustering result is computed visually to show the clusters built.

The cluster analysis results are exactly similar to the ones given by the k-medoids and k-means analysis. The clustering result once again consists of three clusters with the correct cycles in each cluster. Consequently, the Dunn index and silhouette widts of the clusters are similar to the previous ones. It is again of high quality and is computed visually, as shown in Figure 3-4.

All in all, the CLARA implementation for a univariate analysis has proven to be exactly similar to that of the k-medoids one, with a slightly lower computational time.

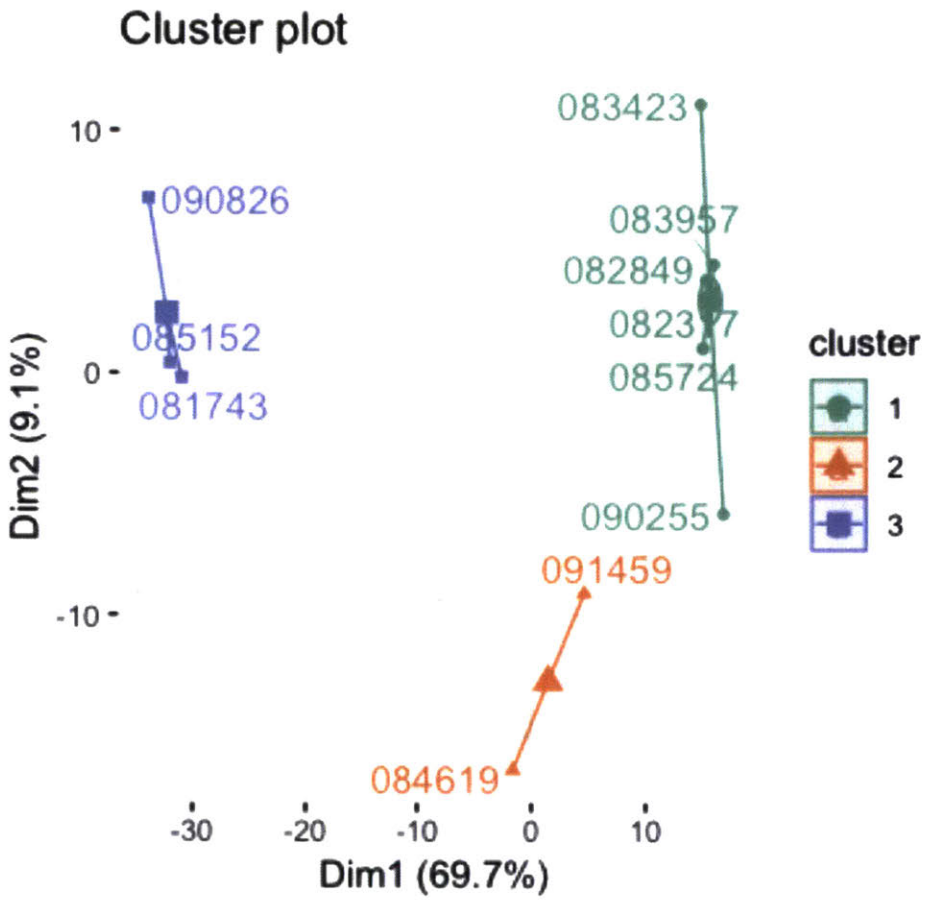

Figure 3-4: Scatter plot of the clusters from the univariate CLARA algorithm

3.3.3 Fuzzy clustering results

The Fuzzy C-means clustering algorithm is computed in **1.433 seconds**. The clustering result is illustrated in Figure 3-5 and shows similar clustering results to the other algorithms.

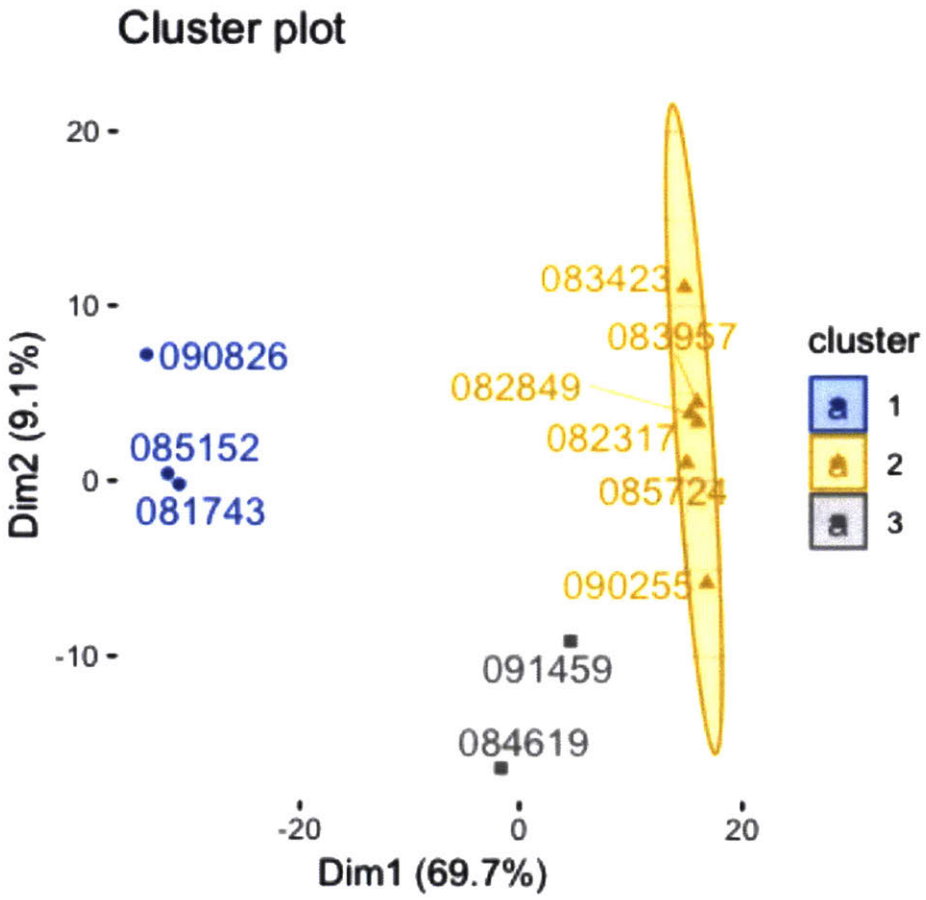

Figure 3-5: Scatter plot of the clusters from the univariate fuzzy C-means algorithm

However, unlike the previous approaches, the fuzzy C-means algorithm gives additional information on the quality of the clustering. It provides a membership matrix as an output. As shown in Table 3.4, this matrix contains the degree to which each observation belongs to a given cluster. The columns correspond to the clusters and the rows represent the cycles. According to this matrix, most of the cycles highly belong to one cluster only. In fact, the cycles 081743, 085152 and 090826 belong to cluster 1 as they are highly distinct from the others (non-anomalous). In addition, the cycles 082317, 082849, 083957, 085724 and 090255 (anomalous) belong to cluster 2 only. However, the remaining highly anomalous cycles 084619 and 091459 belong to the second cluster at a probability of 0.10 and 0.19 respectively as well as cluster 3 at a probability of, respectively, 0.83 and 0.72. In fact, those two cycles are very similar to one another but also to the highly anomalous cycles which explains why they belong to two clusters.

Table 3.4: Membership matrix of the univariate fuzzy clustering analysis

CYCLES	Cluster 1	Cluster 2	Cluster 3
081743	**0.98038702**	0.008359544	0.01125343
082317	0.02420116	**0.913271004**	0.06252784
082849	0.02871834	**0.896191247**	0.07509042
083423	0.04507747	*0.838186453*	0.11673608
083957	0.02470209	**0.911639039**	0.06365887
084619	0.07003824	0.104594974	*0.82536678*
085152	**0.98209405**	0.007636806	0.01026914
085724	0.03041087	**0.890341317**	0.07924781
090255	0.04655321	*0.835218650*	0.11822814
090826	**0.97999718**	0.008534448	0.01146838
091459	0.08607405	0.191954430	*0.72197152*

One of the outputs of the clustering algorithm is the Dunns partition coefficient *F(k)*, with k the number of clusters. It corresponds to the sum of all squared membership coefficients divided by the number of observations clustered. The normalized Dunns coefficient $\bar{F(k)}$ is defined by:

$$\bar{F(k)} = \frac{F(k) - \frac{1}{k}}{1 - \frac{1}{k}} \tag{3.9}$$

A low value (close to 0) of the normalized Dunns coefficient indicates a very fuzzy or weak clustering while a high value (close to 1) suggests a near-crisp or distinct clustering. In this example, the normalized Dunns coefficient is **0.71** and consequently characterizes a near-crisp clustering result. This means that the quality of the clustering is high.

All in all, the fuzzy C-means univariate analysis is indeed of high quality. It also gives an additional insight to how some elements might be similar to several clusters of data sets.

3.3.4 Hierarchical clustering results

Agglomerative algorithm

Several cluster agglomeration methods have been implemented in order to compare the dissimilarities between two clusters of observations. They are based on the implementation of the following linkage criteria: the complete, single, average, centroid linkage clustering as well as the Wards minimum variance method. The computational time of the algorithm is **1.912 seconds**.

The hierarchical agglomerative provides once again the same results and thus successfully clusters the cycles accordingly. These clusters are visually shown in Figure 3-6.

The dendrogram under the average linkage criterion is built, as shown in Figure 3-7. Once the dendrogram is built, it is important to verify that the distances in the tree accurately reflect those of the original distances. For this purpose, the correlation between the cophenetic distances (intergroup dissimilarity at which the two observations are combined into a cluster) and the original distance data is determined. The value of the calculated coefficient can range from -1 to 1. The results are summarized in Table 3.5 for all the linkage criteria evaluated.

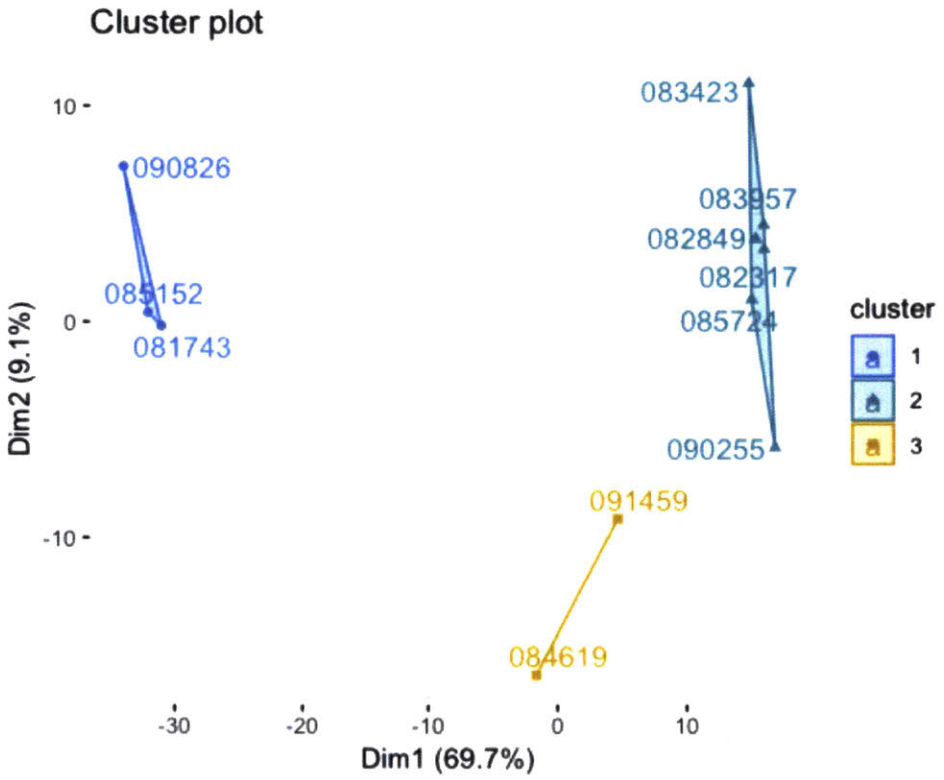

Figure 3-6: Scatter plot of the clusters from the univariate agglomerative algorithm

The higher the correlation coefficient, the better the tree represents the original data set. In this example, it seems that the average linkage clustering is the best linkage criterion, though all of the criteria perform highly. The dendrogram obtained under the average linkage clustering criteria is plotted in Figure 3-7.

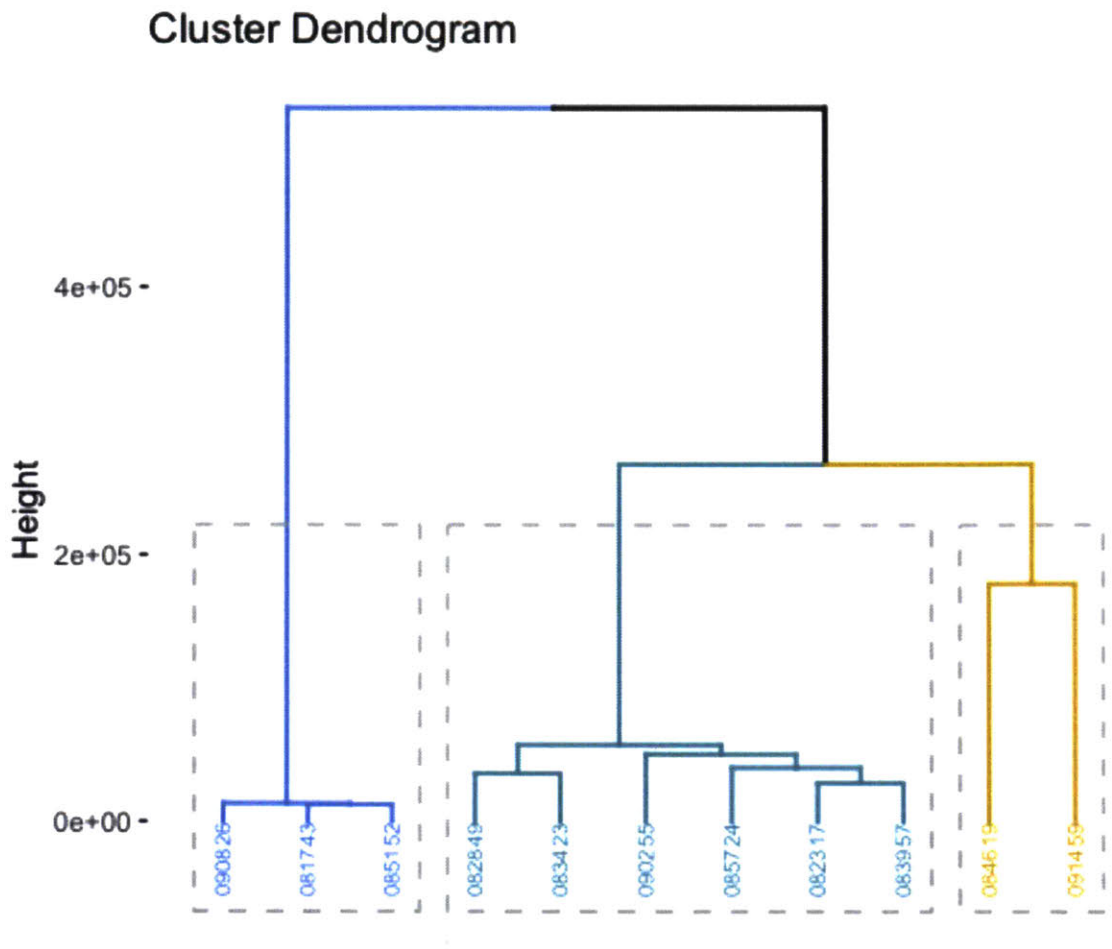

Figure 3-7: Agglomerative dendrogram under the average linkage clustering criteria

Table 3.5: Correlation coefficients for five linkage methods

Wards minimum variance	0.9705647
Complete linkage clustering	0.984001
Single linkage clustering	0.9833871
Average linkage clustering	0.9849523
Centroid linkage clustering	0.9828692

Divisive algorithm

The divisive algorithm has been implemented and has a computational time of **1.912 seconds**.

The divisive clustering result is slightly different from the agglomerative one. The data is clustered in three clusters of respective sizes of 3, 7 and 1:

- The first cluster contains the cycles 081743, 085152 and 090826
- The second cluster contains the cycles 082317, 082849, 083423, 083957, 085724, 084619 and 091459
- The third cluster consists of the cycle 090255 only

Therefore, an anomalous cycle, 090255, has been labeled as a highly anomalous one whilst the highly anomalous cycle 091459 has been considered as slightly anomalous. Given the values of the data for that parameter, there has thus be an error and two cycles have been swapped from one cluster to another. Figure 3-8 illustrates the result of the clustering by using a dendrogram under the average linkage criterion.

The divisive coefficient gives insight on how strong the clustering structure is. The closest it is to 1, the strongest the structure is. In that case, the coefficient is of 0.66, which shows a rather mediocre clustering.

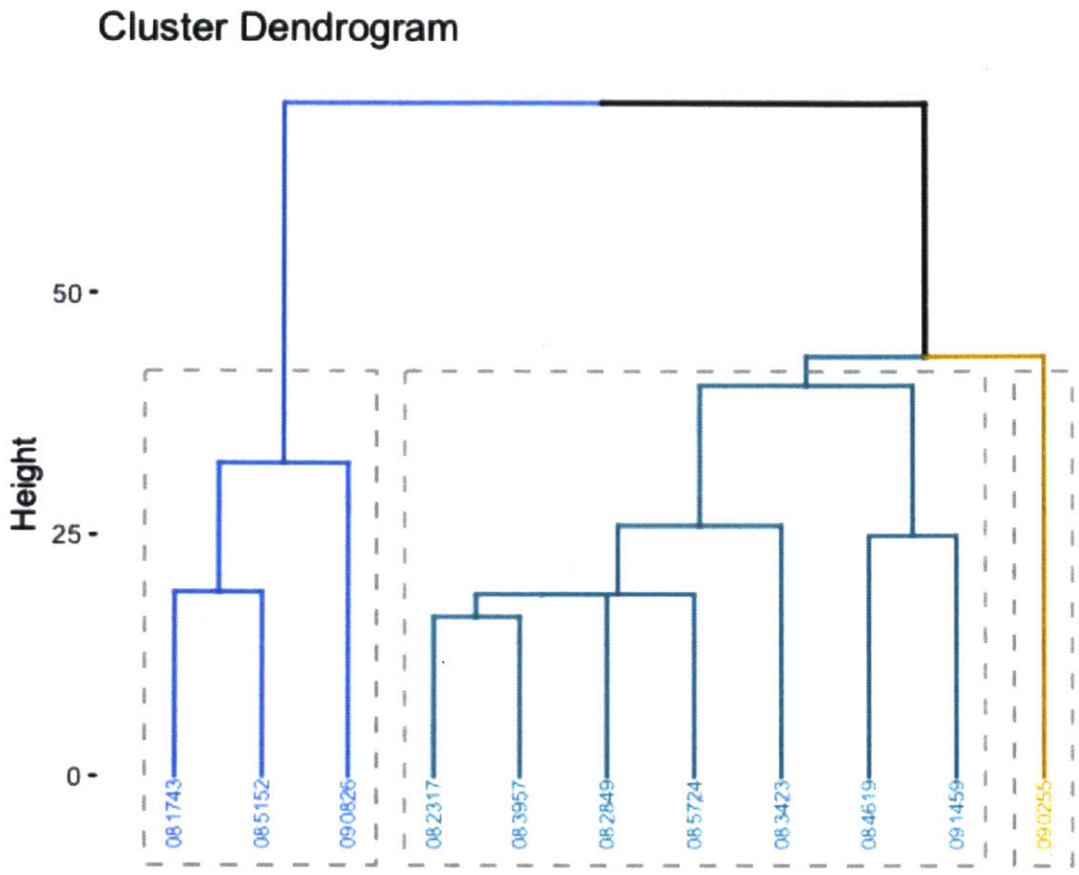

Figure 3-8: Divisive dendrogram under the average linkage clustering criteria

3.3.5 Discussion

The univariate analysis shows that all partitioning algorithms, as well as the fuzzy C-means and the agglomerative hierarchical clustering algorithms ensure a high clustering quality. The divisive hierarchical clustering algorithm has a lower clustering quality. In order to determine which among those algorithms is the best suited for the univariate analysis performed, a cluster internal analysis as well as a stability performance are implemented.

Cluster internal analysis

For all the clustering algorithms of interest, internal measures in combination with a range of cluster numbers are determined. Those include the connectivity, the Dunn Index and the silhouette width. The connectivity shows how well the elements in the clusters are connected. It is based on a supervised classification algorithm, the k-nearest neighbors. Both the Dunn Index and the silhouette width measure the compactness and the separation of the clusters. The higher these measures are, the higher the quality of the clustering is in terms of connectivity, compactness and separation. The performances of the partitioning, hierarchical and fuzzy clustering algorithms are evaluated for a range of number of clusters from two to four. Based on the analysis, the optimal algorithm to use is the k-means since it performs the best for connectivity, Dunn and silhouette measures.

Stability analysis

Stability measures are a special case of internal measures where the quality of the clustering result is determined by removing one potential cluster at a time. These metrics include the APN, AD, ADM and FOM. They are computed in order to assess the quality of the clustering. According to this analysis, the k-means algorithm is once again identified as a well-suited algorithm. Though it is a bit slower to compute than others, given that only a univariate analysis is computed on a rather small data set, this choice of algorithm is indeed appropriate for this particular example. The time complexity of the k-means algorithm is the smallest one.

Confusion matrix

The confusion matrix is computed in order to evaluate the performance of the algorithms computed for this univariate analysis. The recall, precision, F-1 score and accuracy are calculated and summarized in Table 3-6. All the algorithms perform highly, except for the divisive clustering algorithm which doesn't cluster the data perfectly well.

Table 3.6: Performance metrics of all the algorithms

Algorithm	Precision	Recall	F-1 score	Accuracy
k-means	1	1	1	1
k-medoids	1	1	1	1
CLARA	1	1	1	1
fuzzy C-means	1	1	1	1
agglomerative	1	1	1	1
divisive	0.750	1	0.857	0.909

3.4 Multivariate analysis

The purpose of this project is to detect anomalies in large multivariate sets of data that are collected on the production line. A univariate cluster analysis of a key parameter, one that is used by process engineers to understand potential production failures, can be helpful when detecting manufacturing outliers. However, a more thorough analysis to find anomalous cycles is crucial. In fact, the implementation of fast and efficient clustering methods for multivariate analysis is highly interesting for large data sets. The ultimate goal in dividing the data into clusters is to label or find the anomalous cycles in the production of wafers using the plasma etcher.

Once again, the first essential step in performing a cluster analysis is to prepare and pre-process the data. The quality of the clustering results indeed highly depends on that. This time, all the parameters are taken into account for the purpose of the cluster analysis. Given a number C of cycles studied through S number of observations for P parameters, a data frame of dimensions $C \cdot S \cdot P$ is created where every row represents a cycle, every parameter is represented by S columns for each cycle. Therefore, a cell corresponds to the observation of a given parameter for a given cycle. Before proceeding with the cluster analysis, a requirement is for all the constant columns (variance of zero) to be removed.

Table 3.7: Structure of the data frame for the multivariate analysis

Cycle	**Parameter 1**	**Parameter 2**	**...**	**Parameter P**
1	S observations of parameter 1 for cycle 1	S observations of parameter 2 for cycle 1		S observations of parameter 1 for the cycle P
2	S observations of parameter 1 for cycle 2	S observations of parameter 2 for cycle 2		S observations of parameter P for cycle 2
3	S observations of parameter 1 for cycle 3	S observations of parameter 2 for cycle 3		S observations of parameter P for cycle 3
...				
C-2	S observations of parameter 1 for cycle C-2	S observations of parameter 2 for cycle C-2		S observations of parameter P for cycle C-2
C-1	S observations of parameter 1 for cycle C-1	S observations of parameter 2 for cycle C-1		S observations of parameter P for cycle C-1
C	S observations of parameter 1 for cycle C	S observations of parameter 2 for cycle C		S observations of parameter P for cycle C

Since the clustering algorithms implemented are unsupervised, it is important to build a reference cycle, also referred to as a good cycle. By labeling that cycle, it is then possible to identify the cluster that represents the non-anomalous cycles (the reference cycle will be part of that cluster) and the ones that are anomalous.

The reference cycle is determined with the Dynamic Time Warping Barycenter Averaging (DBA) method. Dynamic Time Warping (DTW) aims at finding the optimal coupling between two feature sequences and determines their similarities by aligning their coordinates. Using DTW, two time series can be warped to match each other. DBA is an averaging method that iteratively refines an initial average sequence so that the sum of squared DTW distances from the average sequence to the set of sequences is minimized. At each refinement step of the time series, the DTW between each time series and their average sequence is computed so that associations between coordinates of the average sequence and the set of sequences are found.

Each coordinate of the average sequence is updated and defined as the barycenter of the coordinates associated to it. The ultimate goal of the DBA method is to compute all the barycenters to obtain the final time series average sequence. Once the reference cycle is defined by the DBA method, it is added in the data frame of interest along with all the cycles to study for the multivariate analysis. More information on this and related reference cycle identification methods can be found in Han He's thesis [3], which documents this element of the larger team effort exploring ML approaches for plasma etch anomaly detection.

For the purpose of the multivariate analysis, a reference cycle, conventionally labeled as a "good" cycle, one that encompasses the normal, non-anomalous properties of the data is built. For this purpose, the DBA method has been applied on a set of 200 cycles labelled as good by process engineers to create a reference cycle, that will indeed enable to properly determine which among the clusters are good or bad. Figures 3-9 and 3-10 are the plots of both parameters of interest Bot_RF_RevPwr_In and ProcChm_EndPt_ChanC_In for the reference cycle.

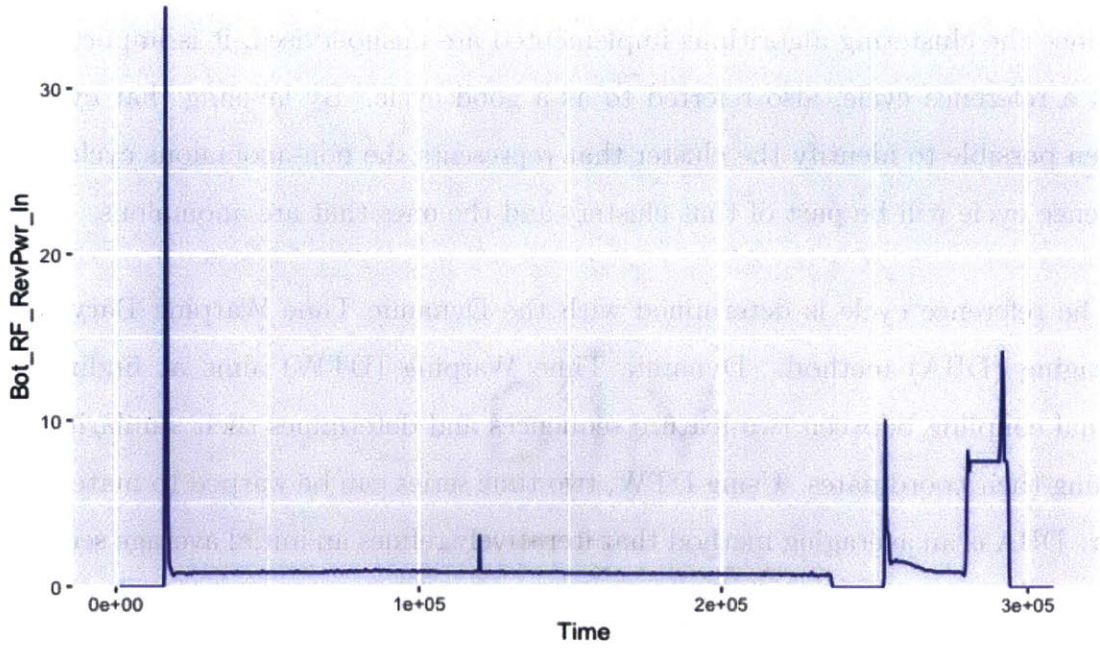

Figure 3-9: Plot of the parameter Bot_RF_RevPwr_In for the reference cycle

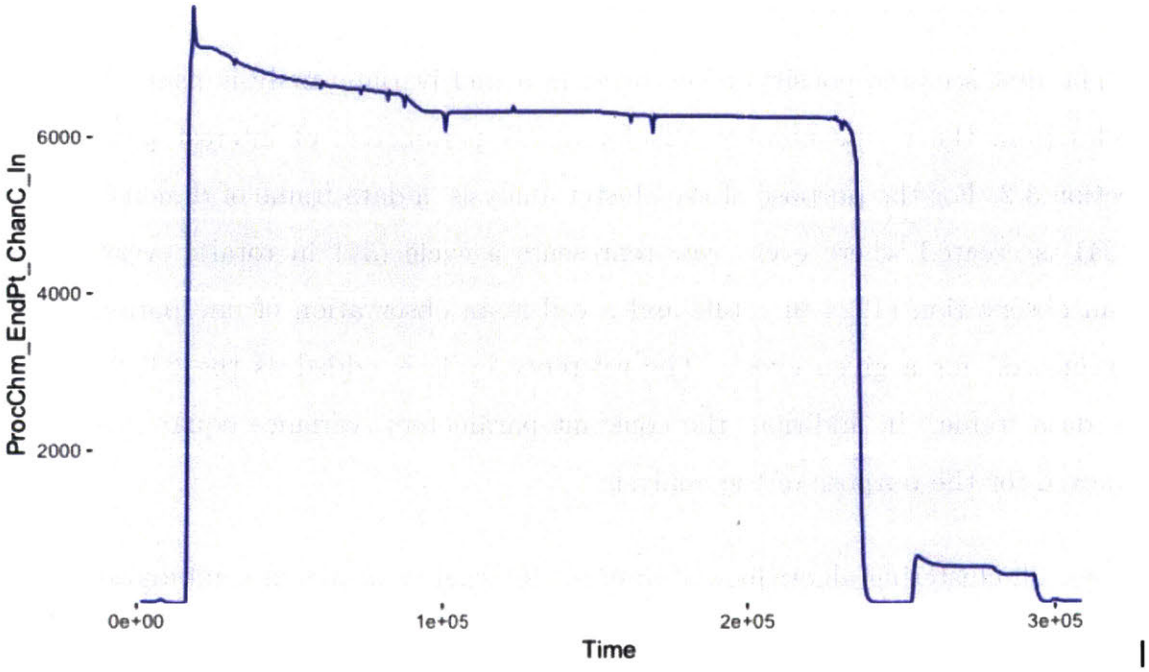

Figure 3-10: Plot of the parameter ProcChm_EndPt_ChanC_In for the reference cycle

In this section, two scenarios are taken into account in order to evaluate the performance of the clustering algorithms. The data for both scenarios are collected from the Machine OXLR7_LAMAL1 based on the plasma etching process for a period of occurrence from 2016-07-26 to 2016-08-02 for which there is a known problem.

3.4.1 Scenario 1: one recipe

The first scenario consists of performing a multivariate analysis using 341 wafer cycles from the recipe number 920 for all 31 parameters of interest presented in Section 3.2. For the purpose of the cluster analysis, a data frame of dimensions 1224 x 341 is created where every row represents a cycle (341 in total), every column is an observation (1224 in total) and a cell is an observation of one parameter (31 parameters) for a given cycle. The reference cycle is added as the 342^{nd} cycle in the data frame. In addition, the constant parameters (variance equal to zero) are removed for the purpose of the analysis.

For all clustering algorithms of interest, internal measures in combination with a range of cluster numbers are determined. The performances of those algorithms are evaluated for a range of clusters from 2 to 4. Those include the connectivity, the Dunn Index and the silhouette width. Based on the analysis of the values, the fuzzy clustering algorithms is excluded because it performs poorly and is computationally expensive. The best results are obtained for both the k-means (computational time of **39.7 seconds**) and the agglomerative (computational time of **42.6 seconds**) algorithms for 2 clusters.

Because those parameters are the most relevant in identifying potential anomalies, both Bot_RF_RevPwr_In (parameter 5) and ProcChm_EndPt_ChanC_In (parameter 19) are the ones plotted in order to show the cycles of interest in Figures 3-11 and 3-12.

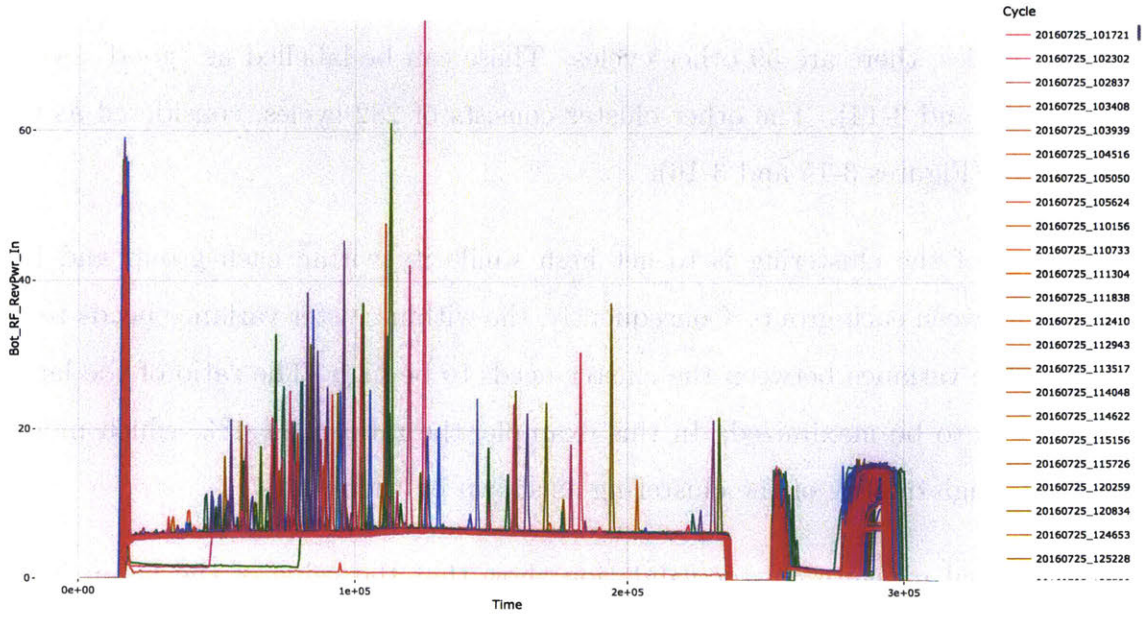

Figure 3-11: Plot of the parameter Bot_RF_RevPwr_In for all 341 cycles of recipe 920

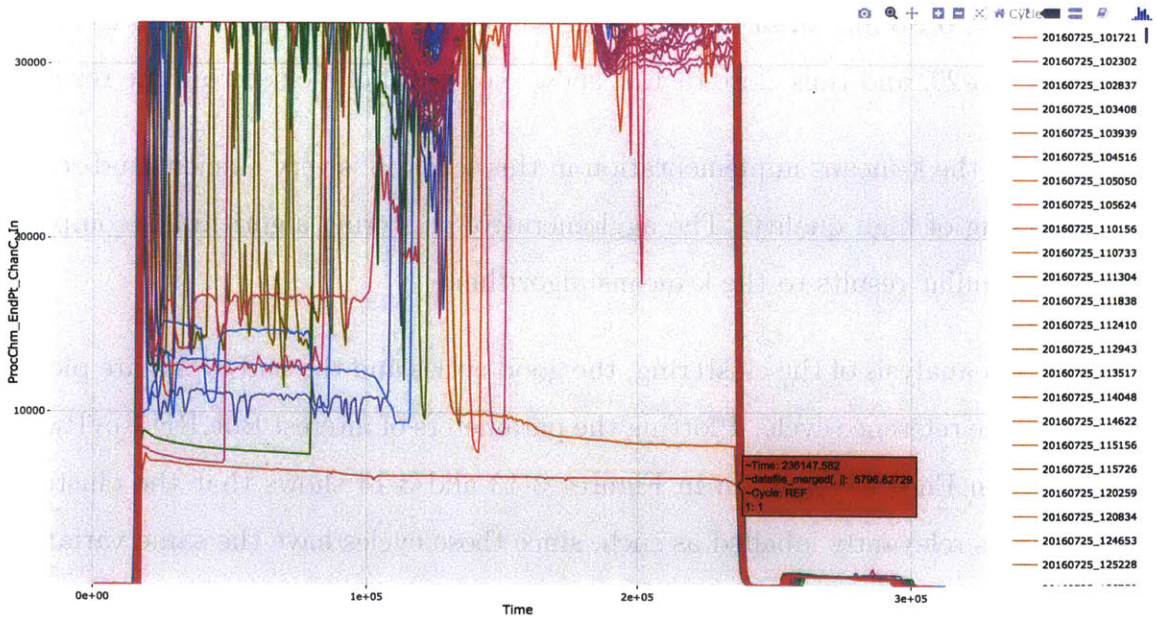

Figure 3-12: Plots of the parameter ProcChm_EndPt_ChanC_In for all 341 cycles of recipe 920

Once the number of optimal clusters k is specified (in that case, k = 2), the algorithm is computed. The clustering result is computed visually in Figures 3-13 to 3-16 for both parameters. In the cluster that contains the reference cycle, or cluster of "good" cycles, there are 59 other cycles. Those can be labelled as "good" cycles (Figures 3-13 and 3-14). The other cluster consists of 282 cycles, considered as the "bad" cycles (Figures 3-15 and 3-16).

The goal of the clustering is to get high similarity within each group and low similarity between each group. Consequently, the within-cluster variance needs to be low whilst the variance between the cluster needs to be high. The ratio of the latter by the first is to be maximized. In this example, the ratio is **91.4%**, which indeed assesses the high quality of the clustering as shown by the results.

The internal measure cluster validation show that the value of the Dunn index for this clustering result is of **0.83**, which is a rather good value and shows that the resulting clustering is good. In addition, the silhouette widths of clusters 1 and 2 are respectively of **0.85** and **0.93**. The average silhouette width being of **0.86** as shown in the figure 3-20, and thus close to 1, it shows how good of a clustering the result is.

All in all, the k-means implementation in this scenario is very efficient and results in a clustering of high quality. The agglomerative clustering algorithm is computed and shows similar results to the k-means algorithm.

For visual analysis of the clustering, the good cycles and the bad cycles are plotted along with the reference cycle. Plotting the parameters of interest Bot_RF_RevPwr_In and ProcChm_EndPt_ChanC_In in Figures 3-13 and 3-14 shows that the cluster of good cycles is relevantly labelled as such, since those cycles have the same variations as the reference cycle. On the contrary, the "bad cycles" are illustrated in Figures 3-15 and 3-16: they are highly different from the reference cycle. The amplitudes of all the bad cycles are indeed way higher than that of the reference cycle in the plots of both the ProcChm_EndPt_ChanC_In and Bot_RF_RevPwr_In parameters. There are also a lot of variations in the plot of the Bot_RF_RevPwr_In parameter.

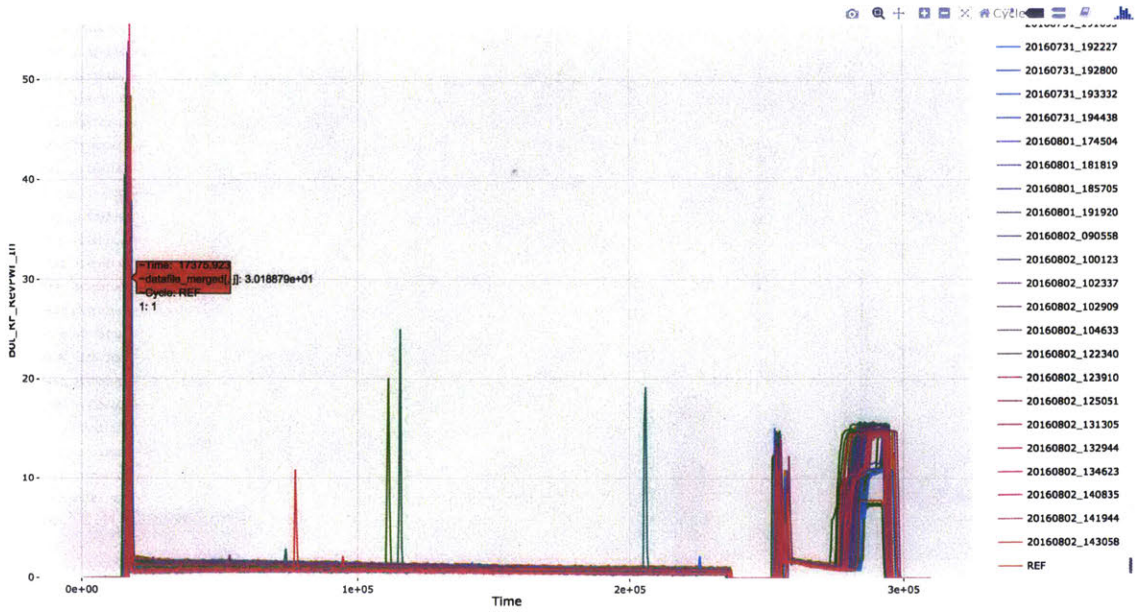

Figure 3-13: Plot of the parameter Bot_RF_RevPwr_In for the reference cycle (orange) and the good cycles

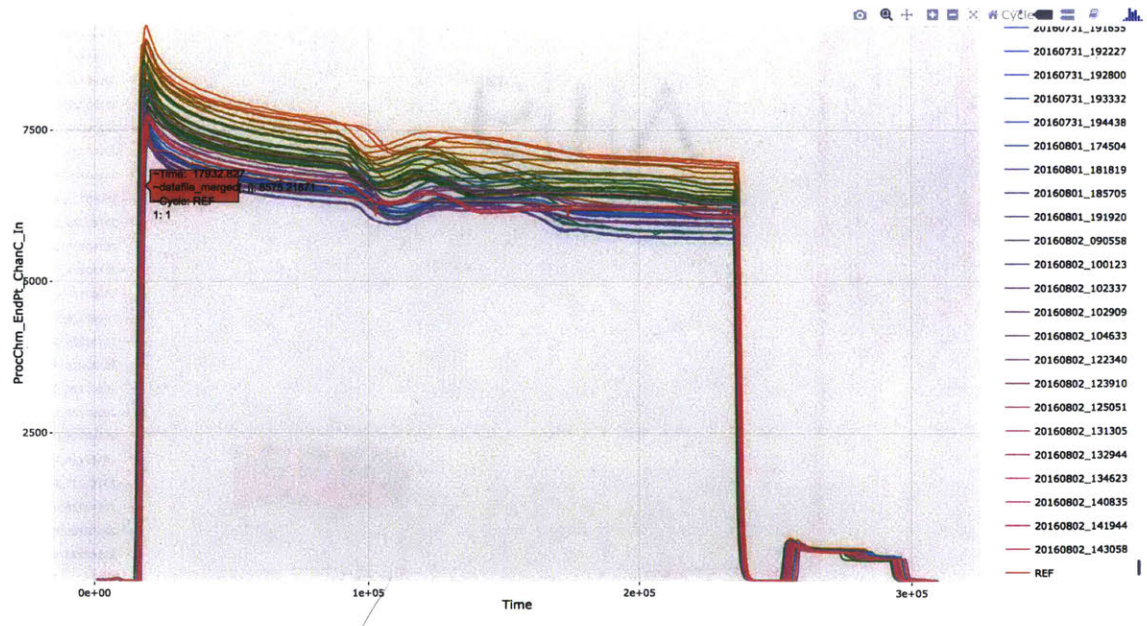

Figure 3-14: Plots of the parameter ProcChm_EndPt_ChanC_In for the reference cycle (orange) and the good cycles

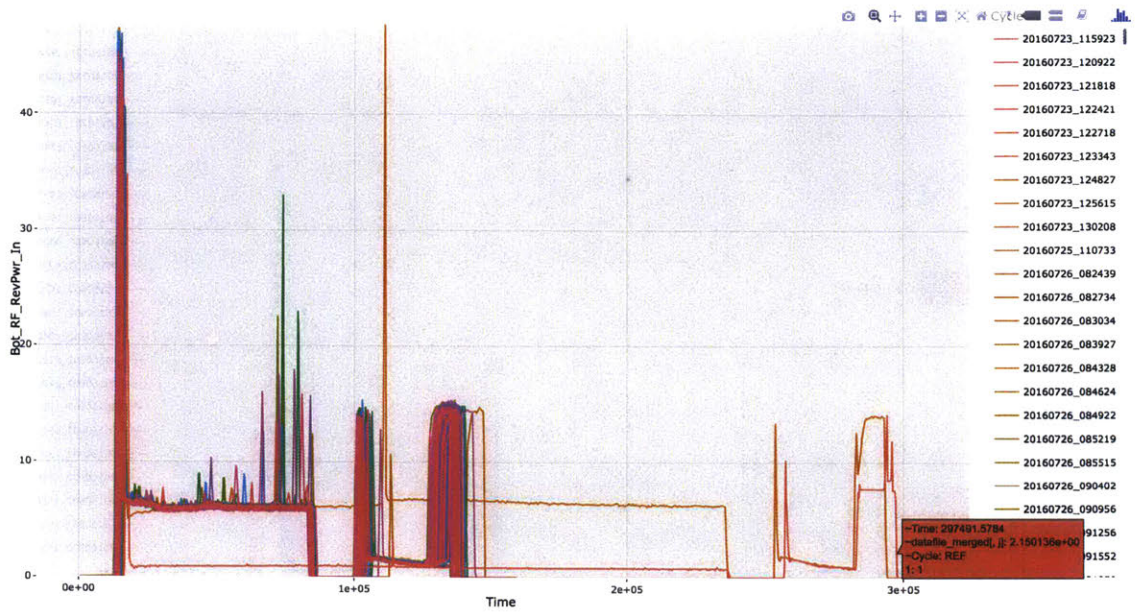

Figure 3-15: Plot of the parameter Bot_RF_RevPwr_In for the reference cycle (orange) and the bad cycles

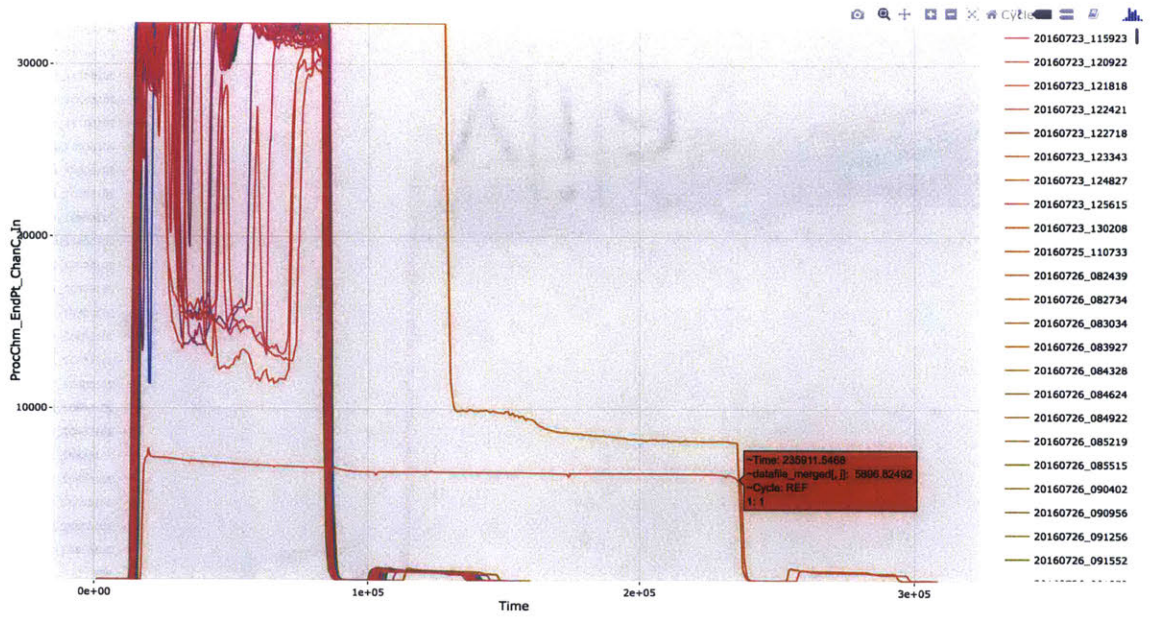

Figure 3-16: Plots of the parameter ProcChm_EndPt_ChanC_In for the reference cycle (orange) and the bad cycles

3.4.2 Scenario 2: two recipes

Similarly, the second scenario consists of performing a multivariate analysis based on two recipes. The purpose of this analysis is to determine whether the clustering algorithms can still correctly identifying the anomalies by considering different recipes at once. For this purpose, all 341 wafer cycles from the recipe number 920 and 370 wafer cycles from the recipe 945 are the basis of the analysis for all 31 parameters of interest presented in Section 3.2.

For the purpose of the cluster analysis, a data frame of dimensions 1224 x 711 is created where every row represents a cycle (711 in total), every column is an observation (1224 in total) and a cell is an observation of a parameter (31 parameters) for a given cycle. The reference cycle is added as the 712^{th} cycle in the data frame. In addition, the constant parameters (variance of zero) are removed for the purpose of the analysis. Figures 3-17 and 3-18 below are the plots of both parameters of interest for all cycles from both recipes 920 and 945.

Both the k-means and the agglomerative clustering algorithms are computed to identify the anomalies in the data that contains both recipes. The computational time of the k-means algorithm is of **210 seconds** and that of the agglomerative algorithm is **250 seconds**.

The k-means algorithm highlights three clusters of cycles: one of good cycles only with five false positives and two of anomalous cycles. As shown on Figures 3-19 and 3-20, the cluster of good cycles combines similar cycles to the reference cycle from both recipes 920 and 945. One cluster of anomalous cycles contains the bad cycles from recipe 920 alone (Figures 3-21 and 3-22) and another only contains bad cycles from recipe 945 (Figures 3-23 and 3-24). The reference cycle is labeled by an orange box.

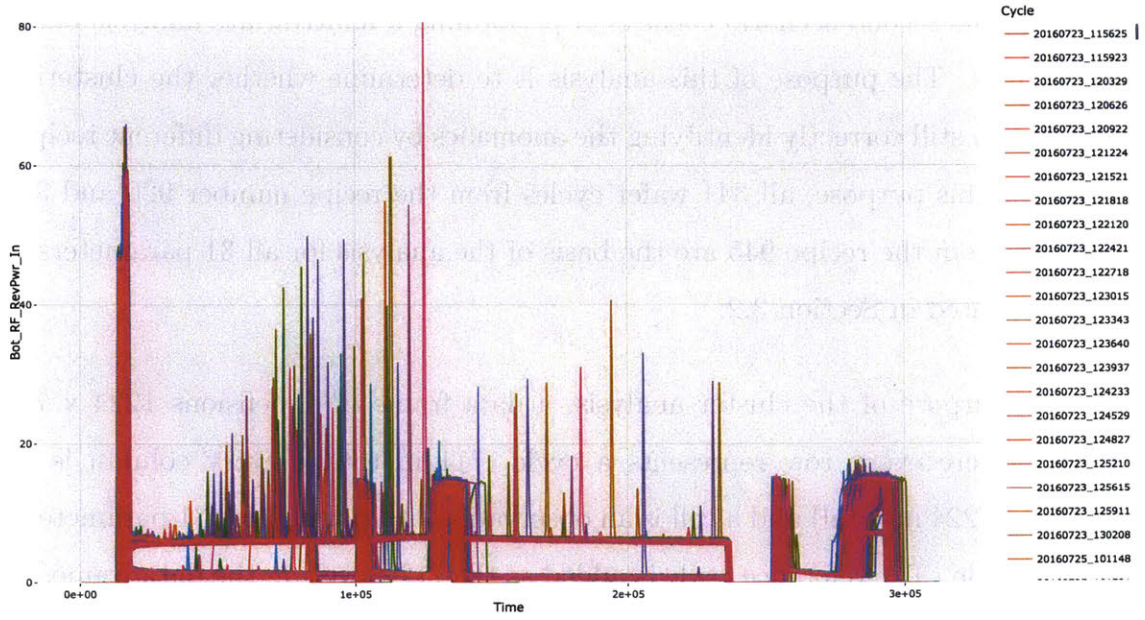

Figure 3-17: Plot of the parameter Bot_RF_RevPwr_In for all 711 cycles of recipes 920 and 945

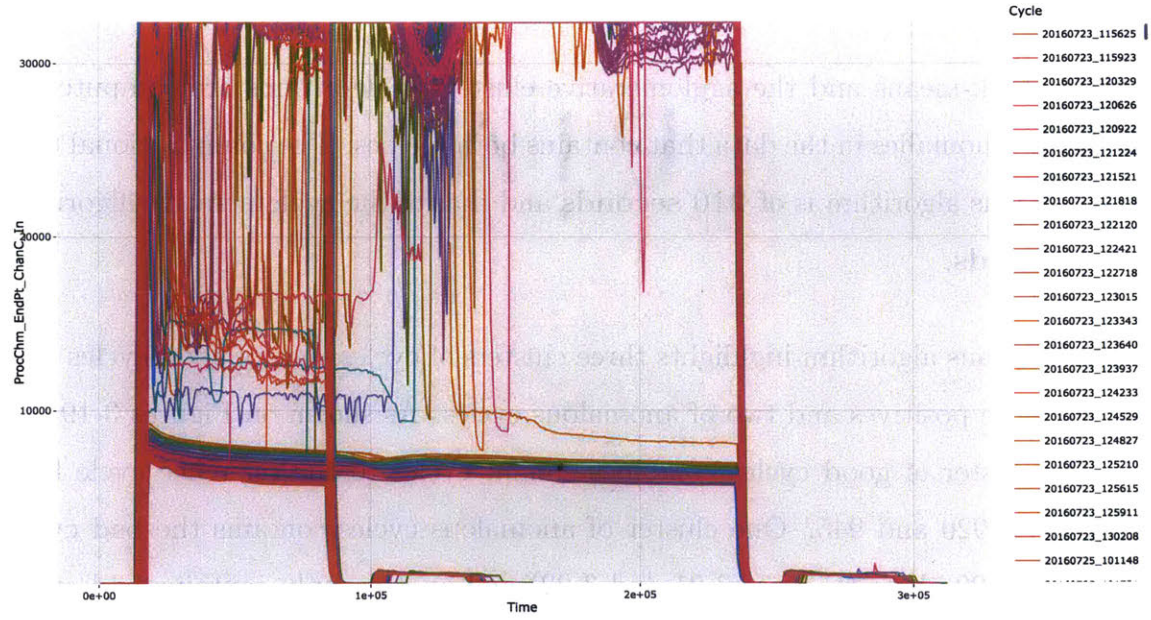

Figure 3-18: Plots of the parameter ProcChm_EndPt_ChanC_In for all 711 cycles of recipes 920 and 945

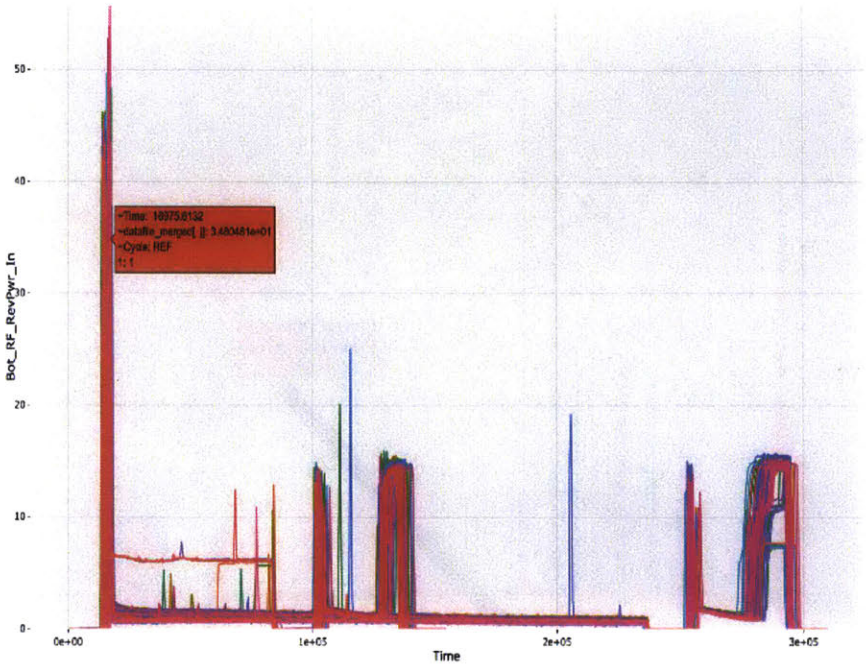

Figure 3-19: Parameter Bot_RF_RevPwr_In for the good cycles of recipes 920

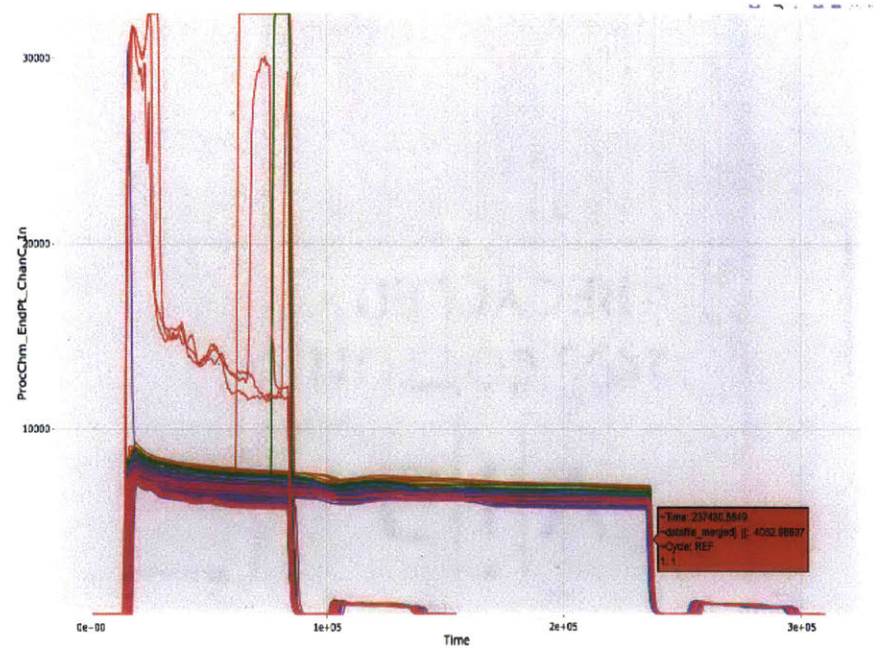

Figure 3-20: Parameter ProcChm_EndPt_ChanC_In for the good cycles of recipe 920

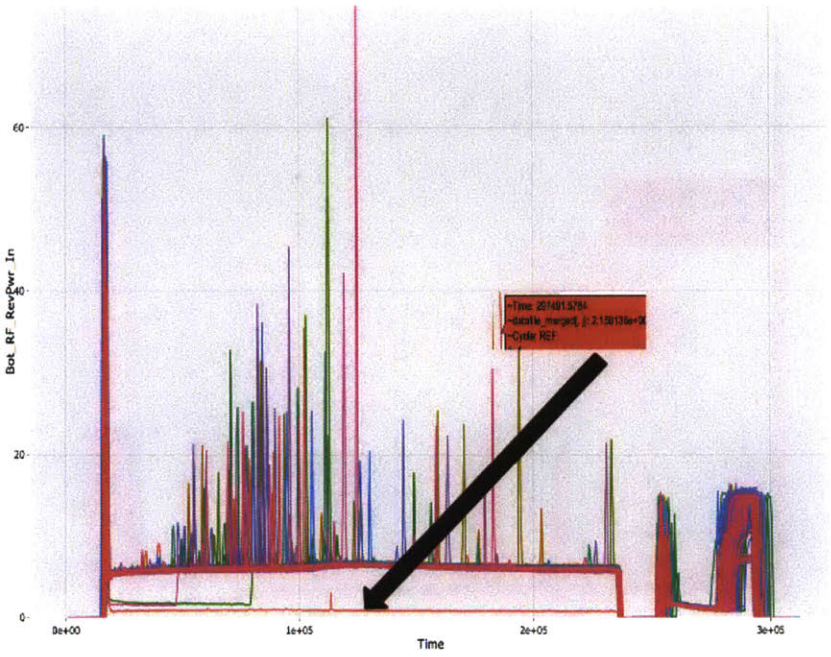

Figure 3-21: Parameter Bot_RF_RevPwr_In for the bad cycles of recipe 945

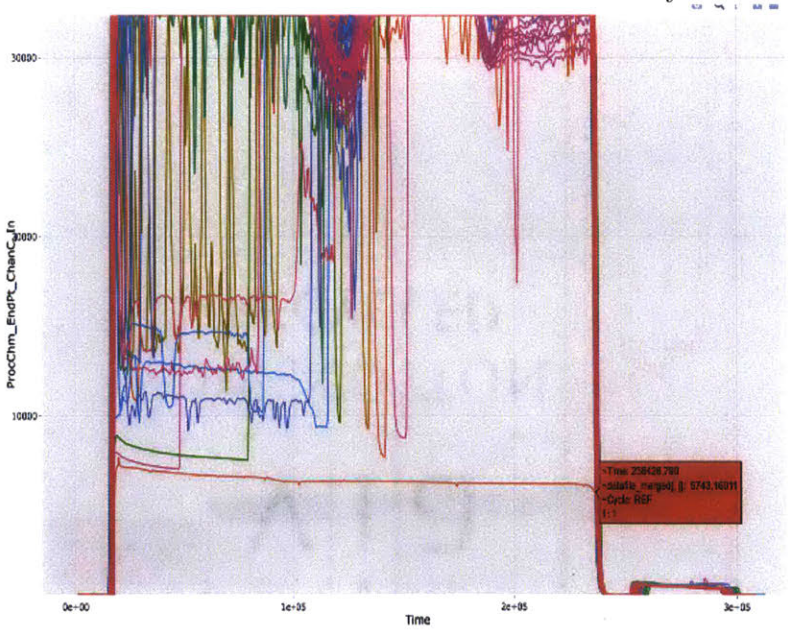

Figure 3-22: Parameter ProcChm_EndPt_ChanC_In for the bad cycles of recipe 945

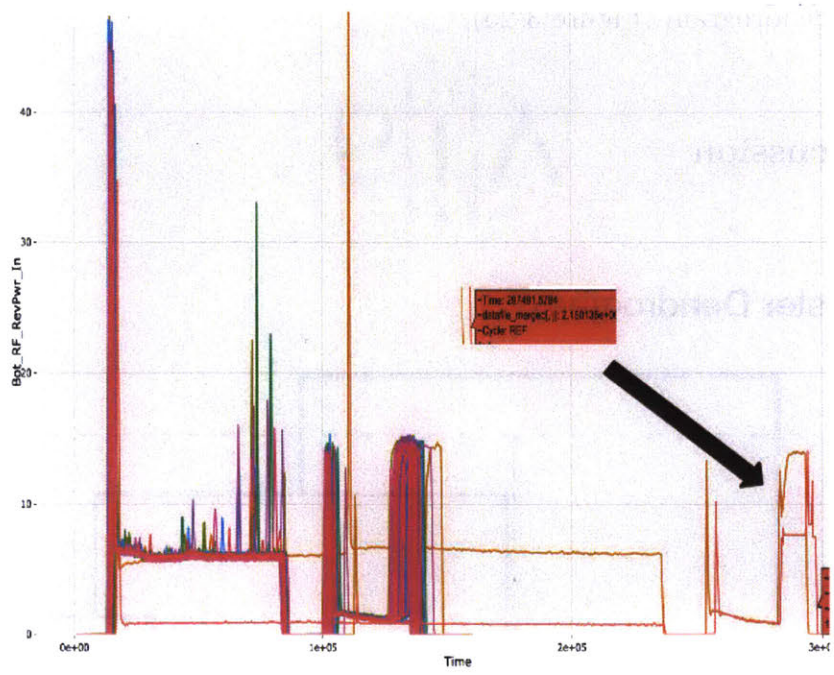

Figure 3-23: Parameter Bot_RF_RevPwr_In for the bad cycles of recipe 920

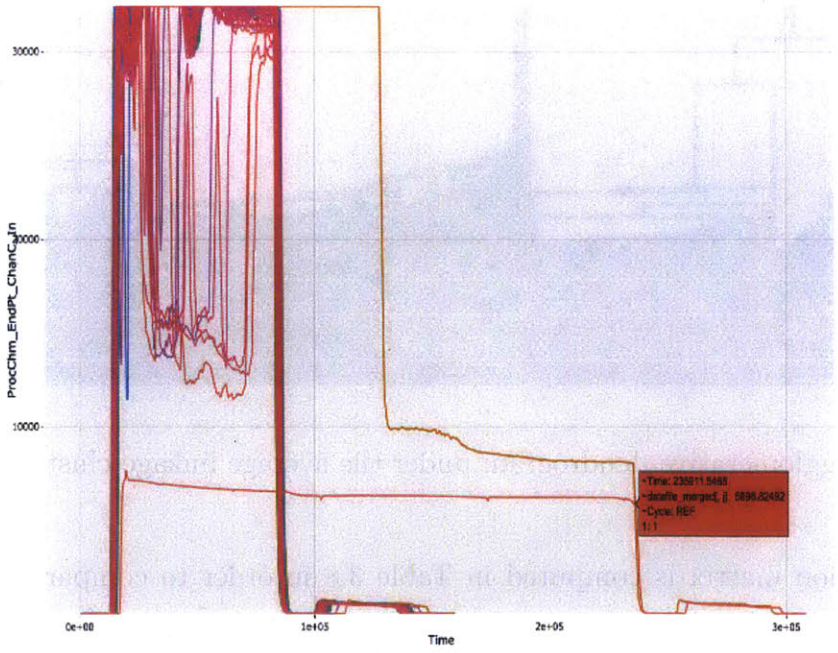

Figure 3-24: Paarameter ProcChm_EndPt_ChanC_In for the bad cycles of recipe 920

The agglomerative algorithm gives an additional analysis of the cycles since the clusters of bad cycles of both recipes 920 and 945 belong to a subgroup of clusters as shown on the dendrogram (Figure 3-25).

3.4.3 Discussion

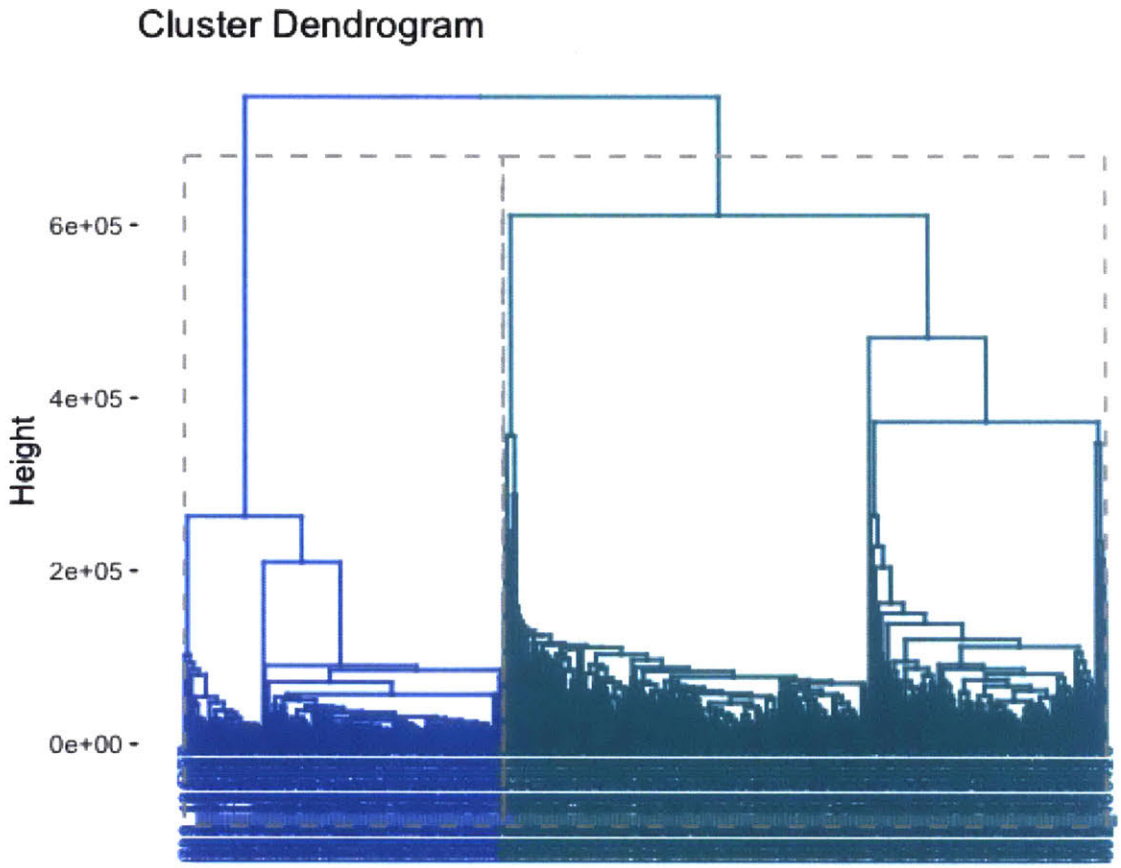

Figure 3-25: Agglomerative dendrogram under the average linkage clustering criteria

The confusion matrix is computed in Table 3.8 in order to compare the performances of the algorithms for both scenarios. Though both algorithms give similar clustering results in both scenarios, the k-means algorithm is faster than the agglomerative algorithm. Whilst scenario 1 has a higher accuracy than scenario 2, both experiments show that the clustering methods are indeed well-suited to identify good and bad cycles.

Table 3.8: Performance metrics for the multivariate analysis for both scenarios

Scenario	Algorithm	Precision	Recall	F-1 score	Accuracy
One recipe	k-means	1	1	1	1
	agglomerative	1	1	1	1
Two recipes	k-means	0.984	0.993	0.989	0.993
	agglomerative	0.984	0.993	0.989	0.993

3.5 Optimal number of clusters

An important feature of the quality clustering is to determine the optimal number of clusters. There are direct and statistical testing methods.

- **Direct methods:** a criterion needs to be optimized, such as the within cluster sum of squares or the average silhouette. The related methods are the *elbow* and *silhouette* techniques.

- **Statistical testing methods:** it consists of comparing evidence against null hypothesis and includes the *gap statistic* method among others.

  In this section, only the direct methods are presented using k-means as the clustering algorithm of reference. The computed methods are based on the example for the univariate analysis.

3.5.1 Elbow method

One of the main goals behind partitional clustering is to define clusters such that the total within-cluster sum of squares (WSS) is minimized. It measures how compact the clustering is. The total within-cluster variation $\mathrm{W}(C_k)$ can then be defined as the sum of squared Euclidean distances between items and the corresponding centroid as shown by equation (3.10).

$$W(C_k) = \sum_{x_i \in C_k} (x_i - \mu_k)^2 \tag{3.10}$$

where x_i designs a data point that belongs to the cluster C_k and μ_k is the mean value of the points that are assigned to the cluster C_k.

The total within-cluster sum of squares WSS can be defined by:

$$WSS = \sum_{k=1}^{N} W(C_k) = \sum_{k=1}^{N} \sum_{x_i \in C_k} (x_i - \mu_k)^2 \tag{3.11}$$

The elbow method consists of analyzing the total WSS as a function of the number of clusters. The chosen number of clusters is the one such that adding an additional cluster does not improve the total WSS. In doing so, the procedure is described as follow:

1. Compute the clustering algorithm (here, k-means) for a variety of number of clusters k
2. For each value of k, the total within-cluster sum of squares (WSS) needs to be calculated
3. The WSS as a function of the number of clusters k is plotted
4. The bend in the plot corresponds to the optimal value of the number of clusters

In order to determine the optimal number of clusters, the total within-cluster sum of squares against the number of clusters needs to be plotted. The example used is that of the univariate analysis described in Section 3.3. In that case, as shown in Figure 3-26 the bend in the plot is noticed for $k = 3$. This means the optimal number of clusters under the elbow method is 3. However, it could have also been possible to choose a number of clusters equal to 2 or 4 and thus the decision of the number of clusters, when done manually, can be complex. However, an automated algorithm was developed in R in order to determine the optimal number of clusters to use.

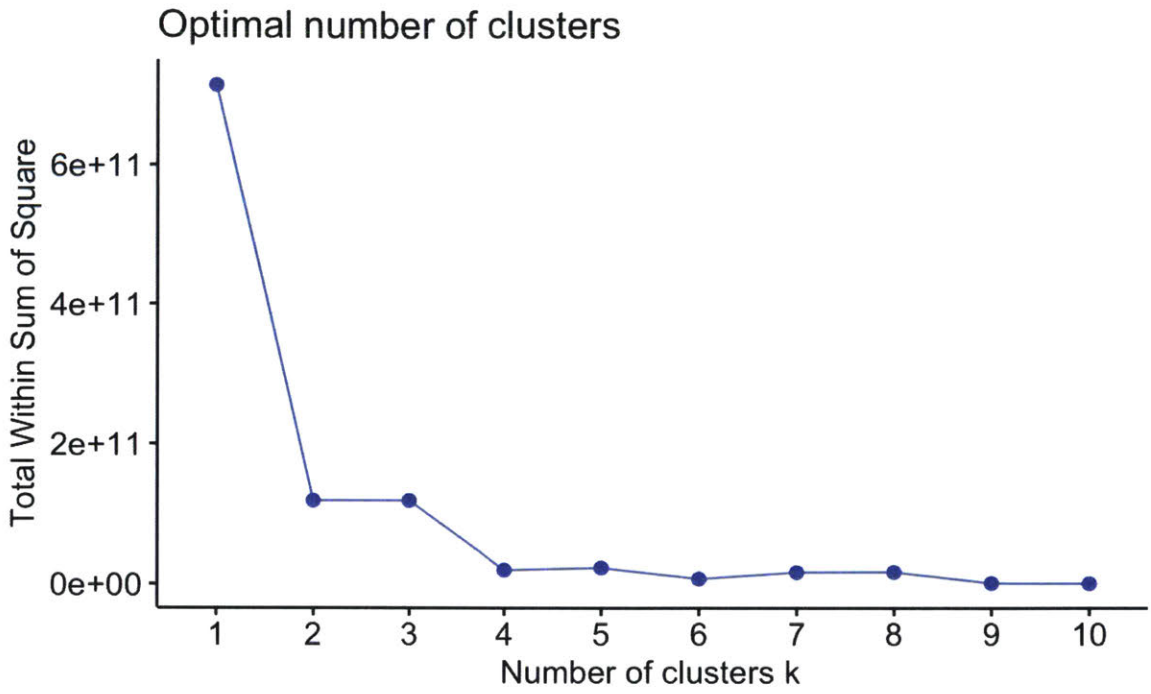

Figure 3-26: Plot of the WSS as a function of the number of clusters k

3.5.2 Silhouette method

The average silhouette method computes the average silhouette of observations for different values of k. The chosen optimal number of clusters is the one that maximizes the average silhouette. The silhouette analysis measures the quality of the clustering of the data by estimating the average distance between clusters. It therefore shows how close each point in a cluster is to neighboring clusters. For each observation i, the silhouette width s_i is calculated through these steps.

For each observation i, the average dissimilarity a_i between the observation i and all the other points of the cluster in which the observation is

For all the other clusters C to which the observation i does not belong, the average dissimilarity $d(i,C)$ of i to all observations of C. The dissimilarity between i and its neighbor cluster is the determined as the minimum value of all dissimilarities, that is to say $b_i = min_C d(i,C)$ The silhouette width S_i of the observation i is defined by $S_i = \frac{(b_i - a_i)}{max(a_i, b_i)}$

The observations with a large S_i (almost 1) are very well clustered whilst a small S_i (around 0) means that the observation is between two clusters. A negative S_i shows that the observation is in the wrong cluster.

The silhouette method consists of the following steps:

1. Compute the clustering algorithm (here, k-means) for a variety of number of clusters k
2. For each value of k, the average silhouette of observations needs to be calculated
3. The curve of average silhouette is plotted against the number of clusters k
4. The maximum in the plot corresponds to the optimal value of the number of clusters

In order to determine the optimal number of clusters, the average silhouette against the number of clusters needs to be plotted, as shown in Figure 3-27. The optimal number of clusters is the one for which the average silhouette width is maximized. In this example, the optimal number of clusters k is considered to be $k = 2$.

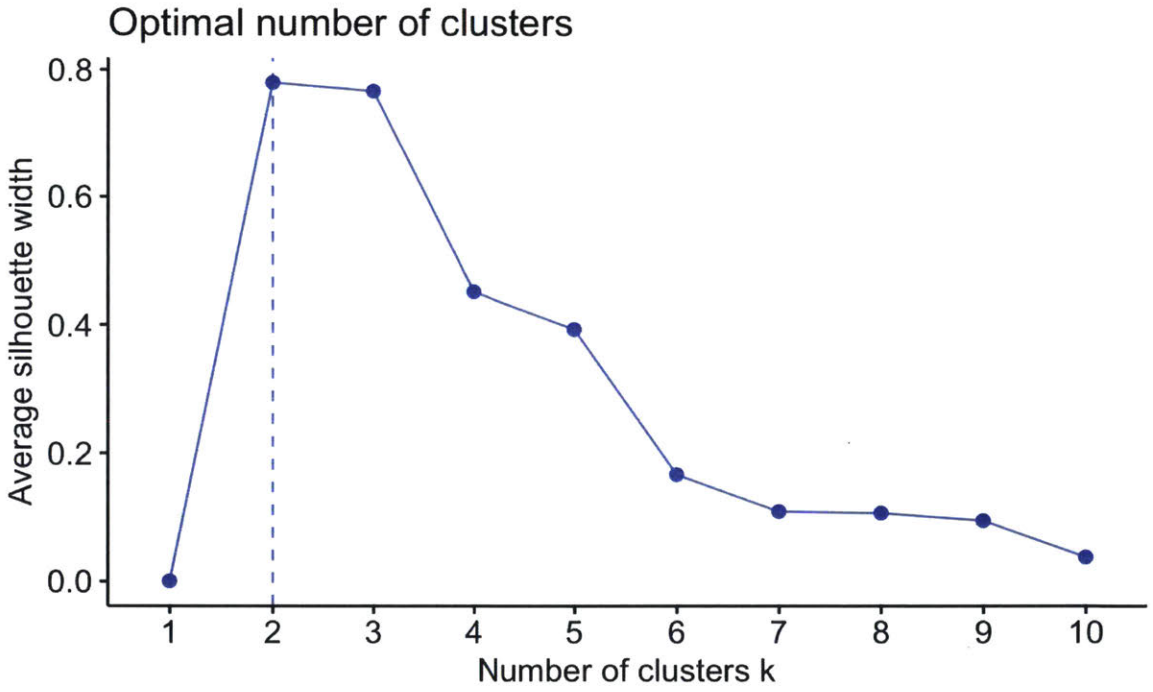

Figure 3-27: Plot of the average silhouette width as a function of the number of clusters k

This page was intentionally left blank.

Chapter 4

Decomposition-based anomaly detection techniques

Seasonal and Trend decomposition using Loess (STL) and Symbolic Aggregate approXimation (SAX) are decomposition-based methods that can help improve the quality of the clustering results. They can also be used as the basis for additional outlier detection algorithms, which are introduced in this chapter.

For the purpose of the analysis, the same univariate example as the one developed in chapter 3 is used in order to compare the different approaches. This univariate example is based on the analysis of the "ProcChm_EndPt_ChanC_In" parameter with five anomalous cycles and three non-anomalous ones.

4.1 STL decomposition

The Seasonal Trend Decomposition using Loess (STL) is a filtering algorithm that was developed by Cleveland et al. (1990) [20]. It aims at dividing up a time series into three components:

- trend: underlying trend of the data. It reflects the long-term progression of the time series and can illustrate either an increasing or a decreasing direction in the data. However, the trend doesnt need to be linear.
- seasonality: repetition of patterns with a fixed period of time. When the time series is influenced by seasonal factors, seasonality can occur over a fixed and known period.
- remainder: residuals of the original time series after both the trend and the seasonality are removed from it. It describes irregular influences.

Under the STL process, a time series Y_t can therefore be mathematically defined by:

$$Y_t = f(S_t, T_t, E_t) \tag{4.1}$$

where S_t, T_t, E_t are its seasonal, trend and remaining components at period t.

There are several models that can be used to represent a time series using the STL decomposition (additive, multiplicative). There are some major steps in the STL decomposition. The first consists of estimating the trend of data. For this purpose, the trend can be estimated with a smoothing procedure like the moving average or by modeling the trend with a regression equation. The second is to de-trend the series. The next step corresponds to the estimation of the seasonal factors based on the de-trended series. This depends on whether the data is evaluated on a yearly, quarterly, monthly or daily basis among other time scales. The final step is to determine the last component of the decomposition, that is to say the residuals or remainder.

The STL decomposition has several advantages, which include its simple design, flexibility in the amounts of variation in the trend and seasonal components as well as its fast computational time.

In order to illustrate the STL decomposition and the STL-based oulier detection, the same univariate time series example of the "ProcChm_EndPt_ChanC_In" parameter for the 11 cycles in section 3.3 is analyzed, resulting in the decomposition shown in Figure 4-1.

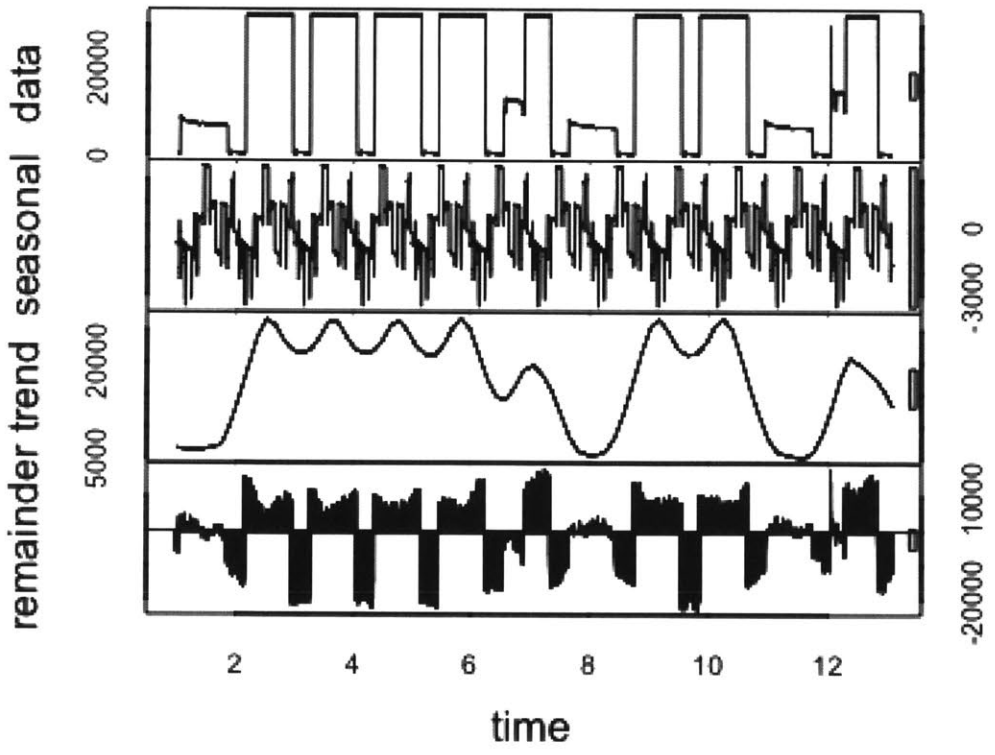

Figure 4-1: Plot of the STL Decomposition of the univariate time series of interest

Decomposing a time series using the STL method is the necessary first step in detecting anomalies in time series. Two methods for anomaly detection include the IQR and the GESD. The IQR method is faster whilst potentially not being as accurate. In contrast, the GESD method is better suited for outlier detection, but unlike the IQR method, it is an iterative process and therefore a bit slower.

4.1.1 InterQuartile Range (IQR) method

The IQR is a measure of statistical dispersion. It is based on dividing the data set into quartiles. It underlines how spread out the middle values are as well as how "far" some other points are. Given the 1st quartile Q_1 (25th percentile) and the 3rd quartile Q_3 (75th percentile) of the data, the IQR can be defined by:

$$IQR = Q_3 - Q_1 \tag{4.2}$$

A box-and-whisker plot uses quartiles in order to plot the shape of the data set. It represents the 1st and 3rd quartiles as well as the median (50th percentile). The IQR corresponds to the length of the box in the box-and-whisker plot, as shown in Figure 4-2. Figure 4-3 shows the boxplot of the univariate analysis of the data of interest.

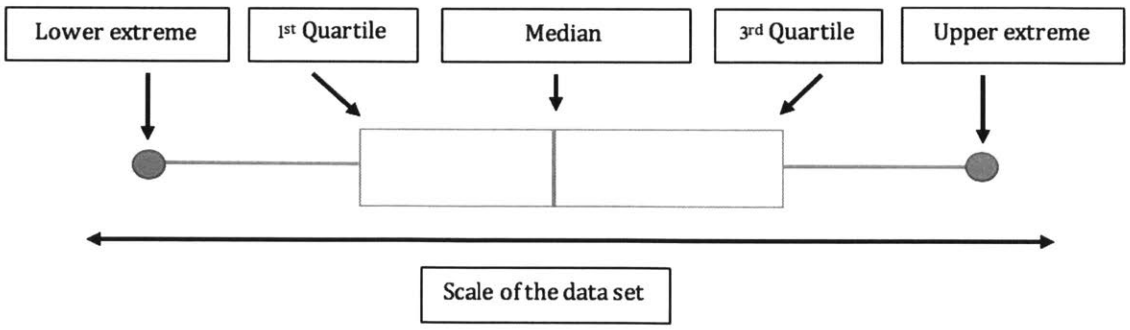

Figure 4-2: Illustration of a box-and-whisker plot

Some observations within the data set might fall outside the scope of the other observations. They are called outliers. Developed by John Tukey [21], pioneer in exploratory data analysis, the IQR is a robust method when it comes to labeling outliers in a set of data. Under the IQR method, an outlier is indeed defined as any data point that is below $Q_1 - 1.5IQR$ or that is above $Q_1 + 1.5IQR$. Those values are referred to as Tukey fences.

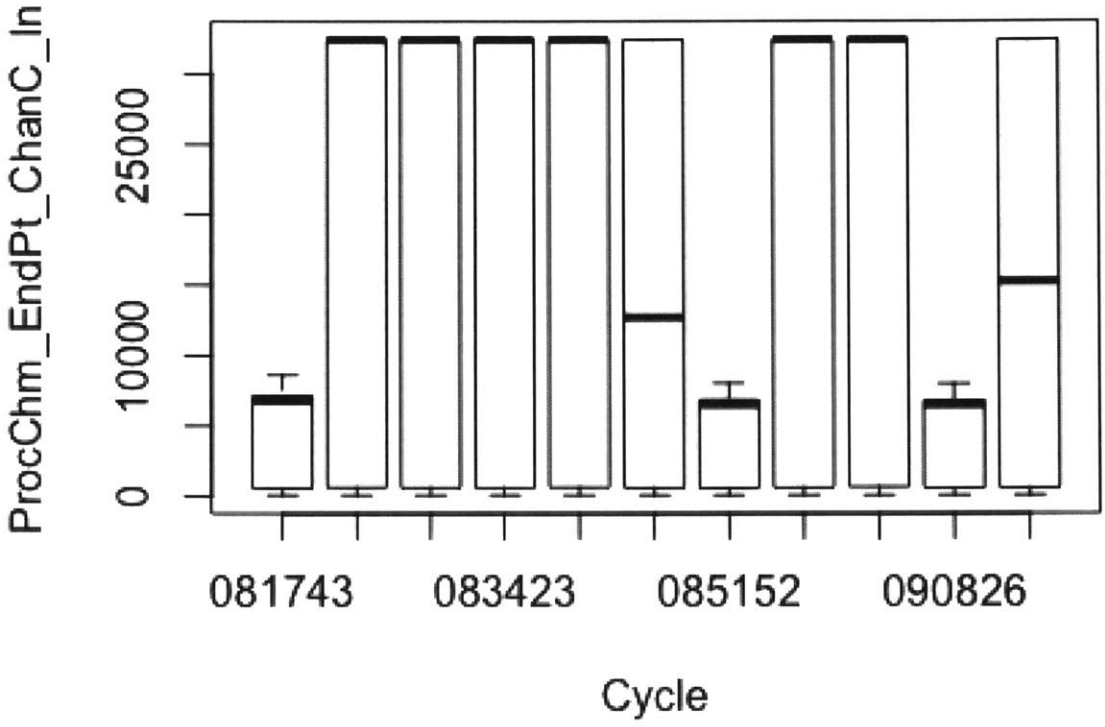

Figure 4-3: Plot of the box-and-whisker of the time series of interest

Using the definition of the IQR method, the outliers in the data set of interest have been identified as such on the plot of all 11 cycles for the parameter of interest. Figure 4-4 shows in red the anomalies detected using the Tukey fences for an alpha of 0.5. The results of this analysis label all the cycles as anomalous. A more thorough development of the IQR method for anomaly detection can help improve the analysis and better identify the real anomalies.

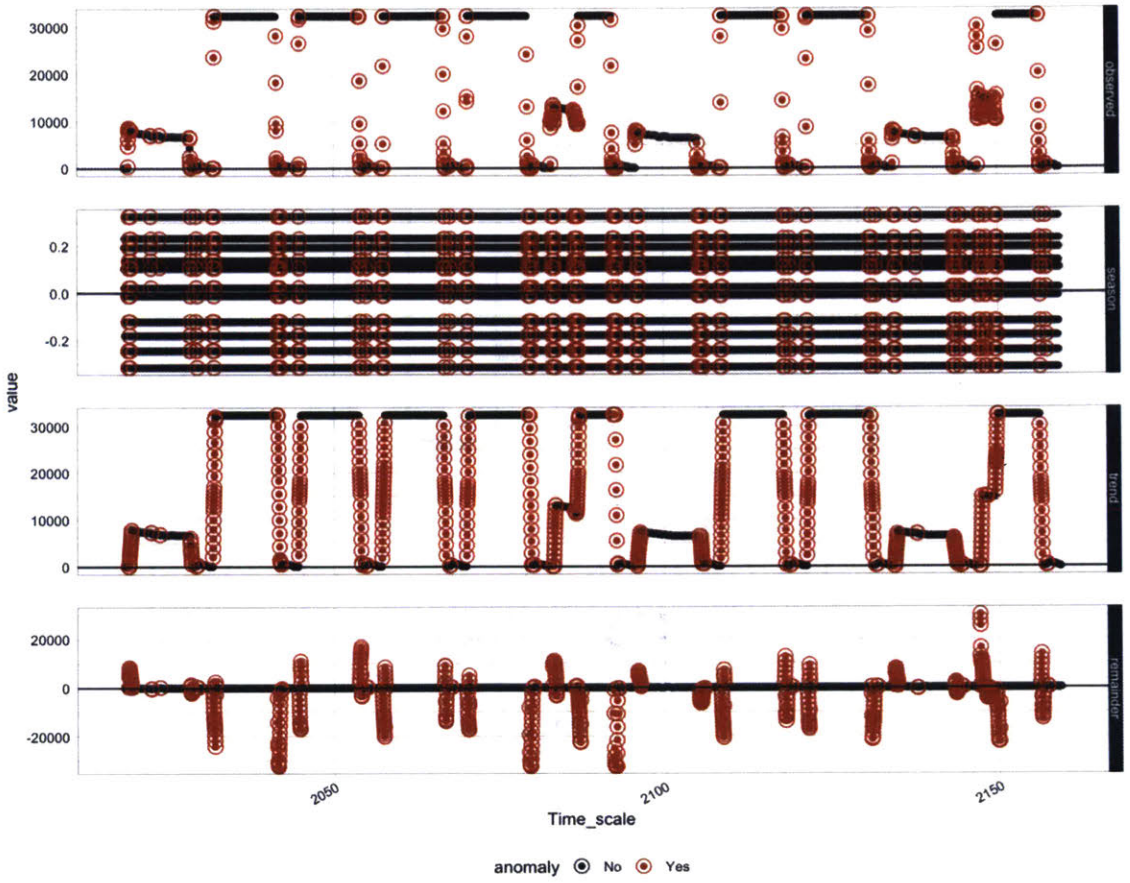

Figure 4-4: Plot of the anomalies detected using the STL decomposition with the IQR method for alpha = 0.5

4.1.2 Generalized Extreme Studentized Deviate (GESD) test

The Generalized Extreme Studentized Deviate Test (GESD), developed by Rosner (1983) [22], enables the detection of outliers in univariate data sets. Also known as the Grubbs test, it iteratively eliminates the outliers using a Students t-test comparing the test statistic to a critical value. It then recalculates the test statistic and critical value.

The test is defined by the following set of null and alternate hypotheses:

H_0: there are no outliers in the data set

H_1: there are at most a given number k of potential outliers in the data set

For the purpose of testing the data set D that contains n elements, k two-tailed Grubbs test statistics $G_1, G_2, ..., G_k$ are generated. These test statistics are determined by:

$$G_i = \frac{max_i |x_i - \bar{x}|}{s} \tag{4.3}$$

where x_i represents the i^{th} observation in the data and $\bar{x}$are respectively the sample mean and standard deviation.

In fact, the observation can either be greater than the mean, in which case the statistic would be $G_{max} = \frac{x_{max} - \bar{x}}{s}$ or less than the mean, with the statistic being $G_{min} = \frac{\bar{x} - x_{min}}{s}$ and where x_{max} and x_{min} are the extreme observation values. The observation that maximizes the value $|x_i - \bar{x}|$ is removed and the test statistic is computed with the remaining observations. The procedure is repeated until the k potential outliers are removed.

Using the GESD method, the outliers in the data set of interest have been identified as such on the plot of all 11 cycles for the parameter of interest. Figure 4-5 shows in red the anomalies detected using the GESD method for a value of alpha of 0.5. Similarly to the results from the IQR anomaly detection method, the results of this analysis label all the cycles as anomalous. A more thorough development of this method for anomaly detection can help improve the analysis.

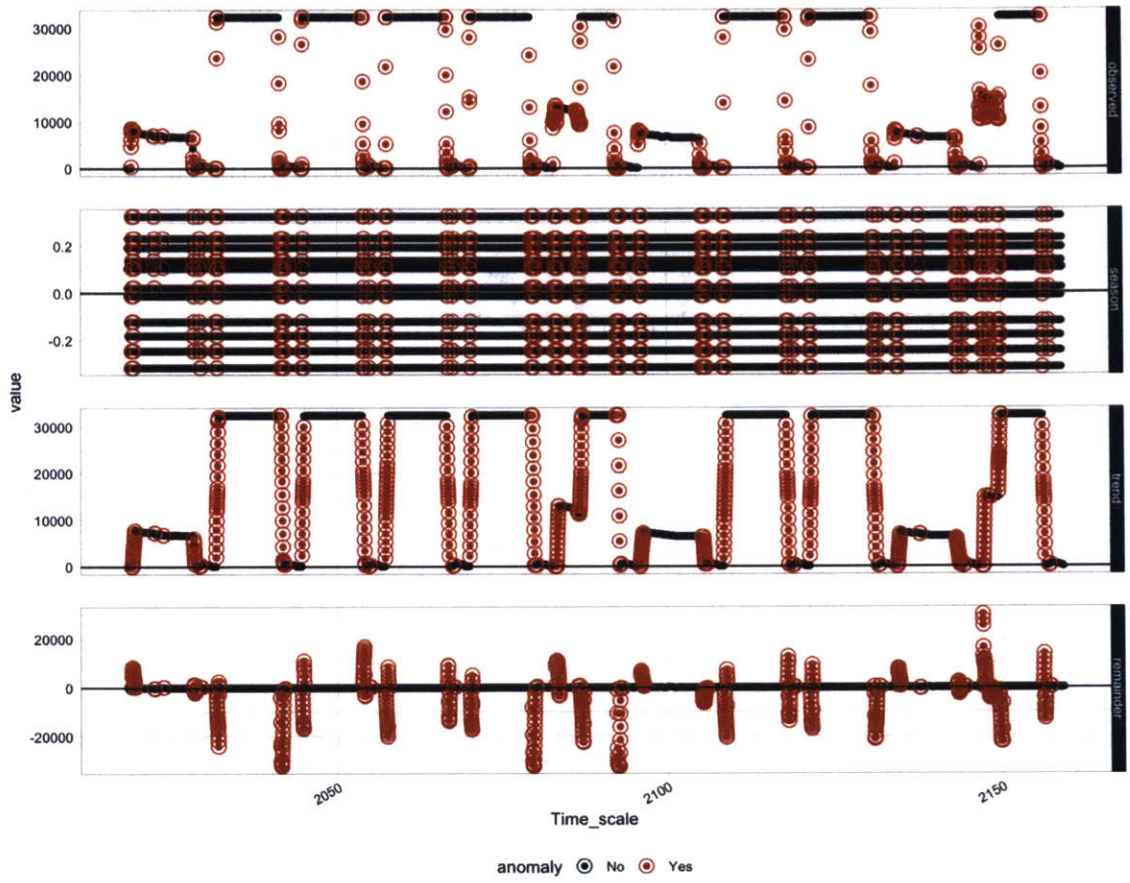

Figure 4-5: Plot of the anomalies detected using the STL decomposition with the GESD method for alpha = 0.5

4.2 SAX decomposition

Representations of time series, such as wavelets or Fourier transforms, present two major flaws. The first is that the dimensionalities of the original data and the symbolic representations must be similar. The second challenge is that the distance measures used for those symbolic representations are often dissimilar to the ones defined on the original data. Symbolic Aggregate ApproXimation (SAX) is a method that has been proposed by Lin et al.[23]. It is based on the representation of time series by symbolic strings. With SAX, a new symbolic representation was introduced: it overcomes the two challenges other symbolic representations face.

4.2.1 Transformation into a string

The algorithm for obtaining the SAX representation consists of two steps. The first step is to reduce the dimensions of the original time series using Piecewise Aggregate Approximation (PAA). The second step is to assign a letter or discrete label to every PAA element in order to symbolize the time series using string symbols. But before running the SAX algorithm, a pre-processing of the time series is needed. Since it is not possible to compare time series with different offsets and amplitudes, they need to be z-normalized beforehand.

Normalization of the time series

This normalization ensures that the data points that form the input vector are transformed into an output vector with a mean of 0 and the standard deviation close to 1. Given a data point x_i from the input vector that has a mean μ and standard deviation σ, a new data point is obtained using the z-normalization formula:

$$\frac{x_i - \mu}{\sigma} \tag{4.4}$$

This step is paramount since the amplitude of the data is no longer of interest and only the dissimilarities within it are considered.

Figure 4-6 is that of the z-normalized time series of the parameter "ProcChm_EndPt _ChanC_In" for all 11 cycles. All the observations have been normalized and consequently the resulting vector has a mean of 0 and a standard deviation of 1.

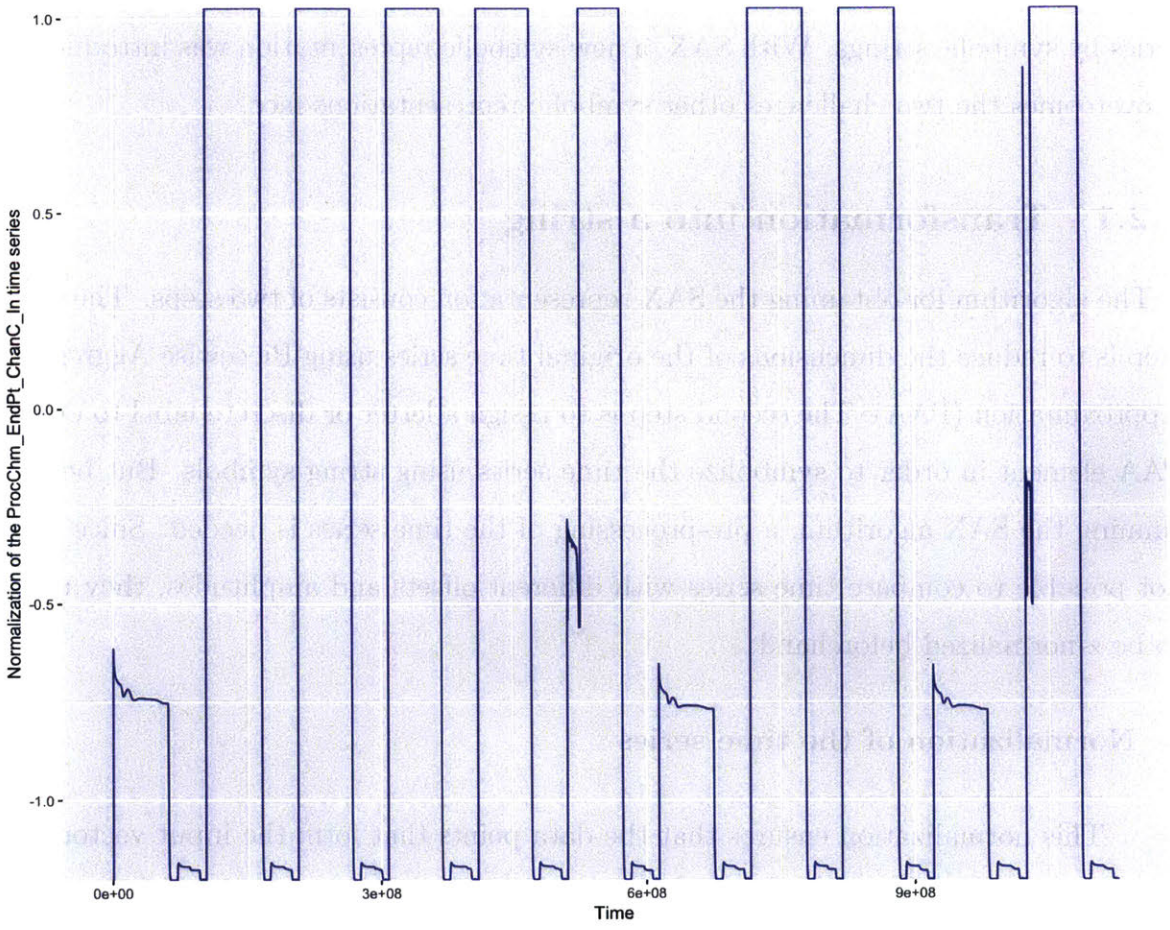

Figure 4-6: z-normalization of the time series of interest

Piecewise Aggregate Approximation (PAA)

The normalized time series then needs to be converted to a PAA representation. Through this dimension-reduction procedure, a time series C of length n is transformed into a vector of w equally-sized frames $\bar{C} = (\bar{c_1}, \bar{c_2}, ..., \bar{c_n})$, where $w << n$. In this vector, the j^{th} component is determined by:

$$\bar{c_j} = \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j \tag{4.5}$$

This vector corresponds to the data-reduced representation of the time series where every component is the mean value of the data in the considered w-dimensional frame.

Figures 4-7 and 4-8 are those of the PAA representations of the z-normalized time series of the parameter "ProcChm_EndPt_ChanC_In" for all 11 cycles. It shows how different the representations are when reducing the dimensionality of 7337 by 25 or 100. In the latter, a reduction of 100 highly affects the data and does not enable to fully encompass its variations. The choice of by how much the dimensionality of the data needs to be reduced is thus paramount. In fact, the parameter w controls the length of the PAA representation and therefore the length of the SAX representation. If w has a high value, the PAA can be detailed whilst it is abstract when the value of w is small. Such a choice of the value of w depends on the application. In the example studied, reducing the dimensionality by 25 means the output PAA vector contains $w = \frac{7337}{25} = 294$ frames. Reducing the dimensionality by 100 results in obtaining a smaller value of $w = \frac{7337}{100} = 74$ frames. Consequently, a reduction of the dimensionality by 25 thus permits to obtain a finer PAA representation than when reducing it by 100.

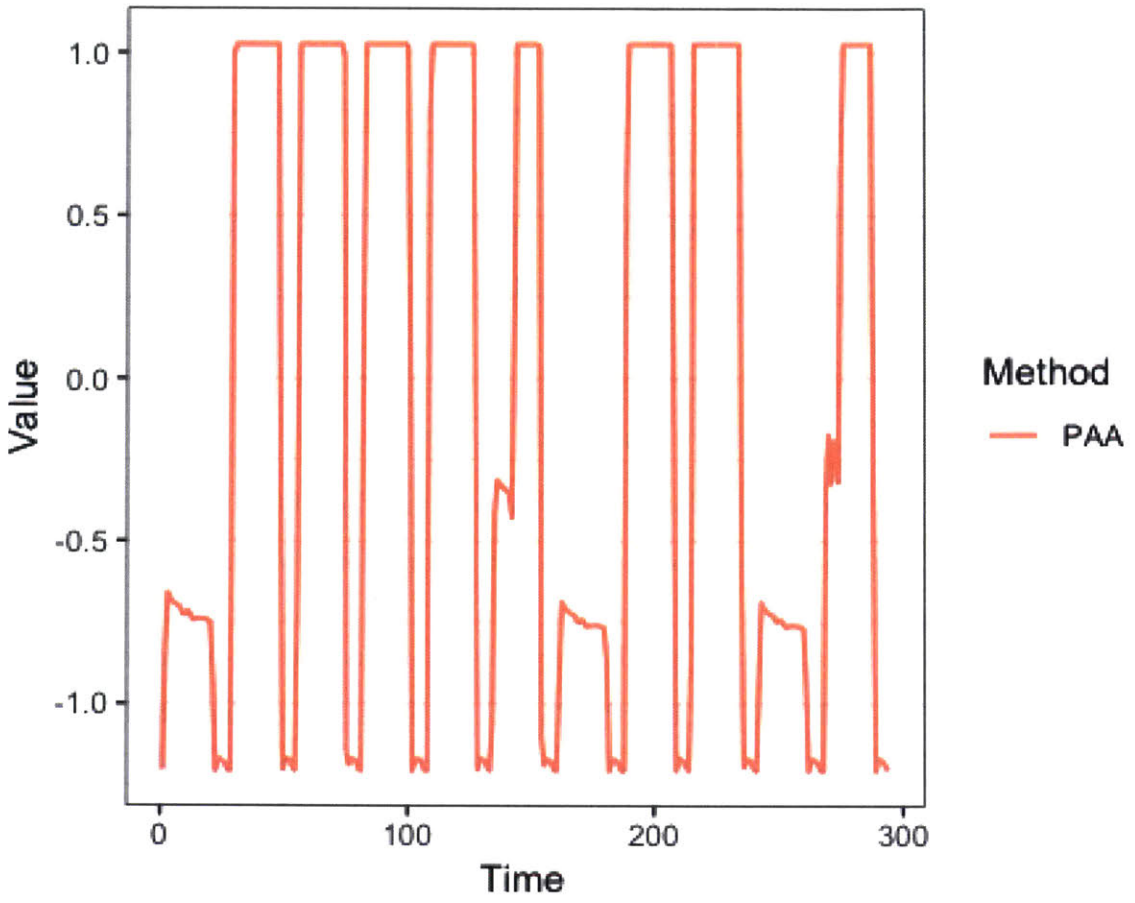

Figure 4-7: PAA representation of the time series by reducing dimensionality 100 times

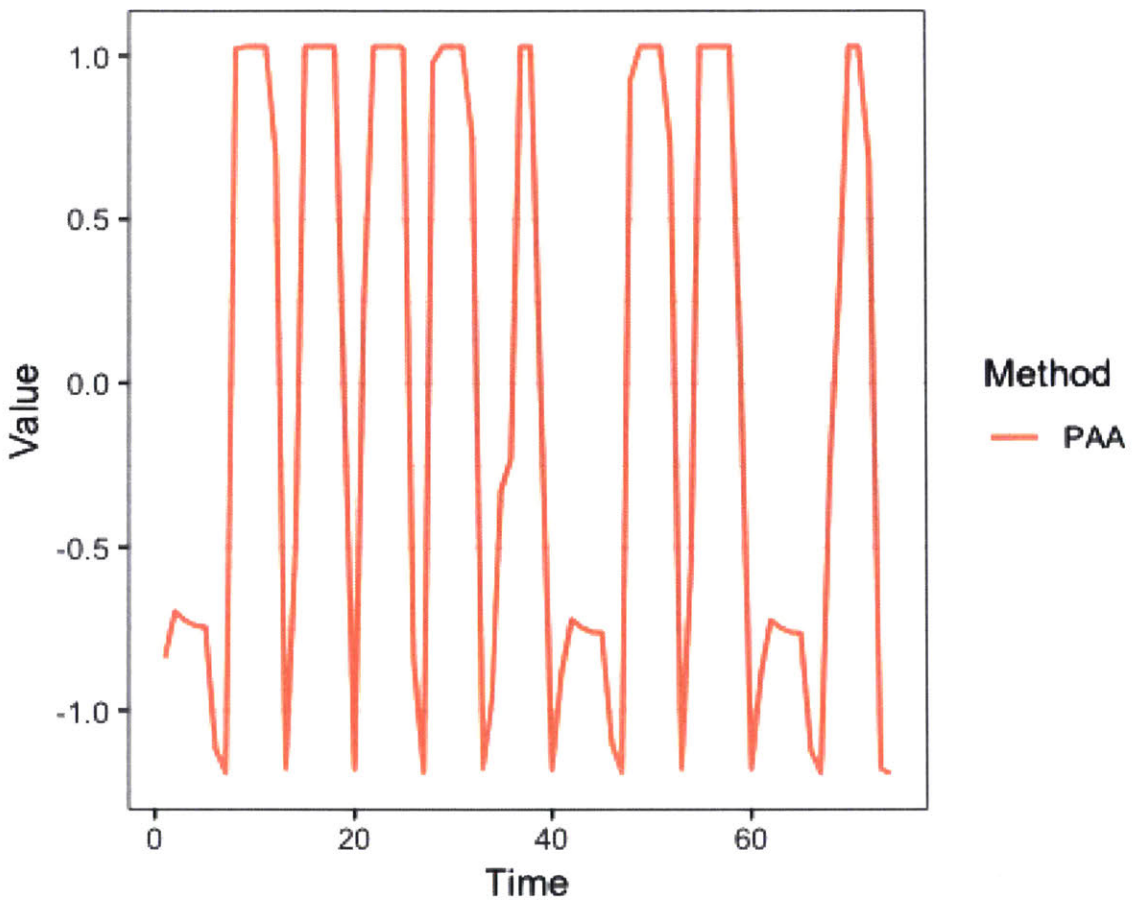

Figure 4-8: PAA representation of the time series by reducing dimensionality 100 times

The SAX representation was introduced in order to tackle two challenges that other time series representations faced: the dissimilarities of the dimensionalities and distance measures between the original data and the symbolic representation. Figure 4-9 enables to visually compare two methods, the Fourier Transform (DFT) and Wavelet Transform (DWT) with the PAA representation for the same univariate analysis of the time series of the parameter "ProcChm_EndPt_ChanC_In" for all 11 cycles. Despite z-normalizing the data, the amplitude of both the DWT and the DFT of the initial data show that the z-normalization has not been conserved through those transforms.

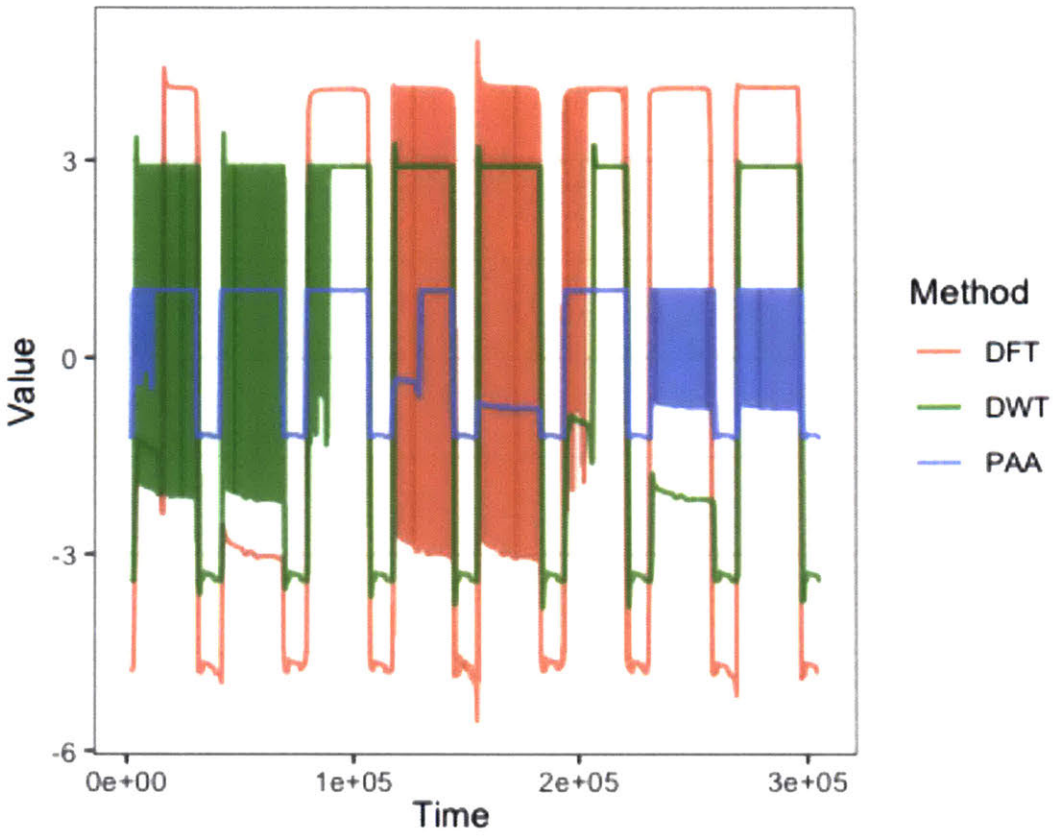

Figure 4-9: DFT, DWT and PAA representations of the time series DWT with level of 2^3, first 917 DFT coefficients extracted and inverted and classical PAA with a dimensionality reduction of 8

The discretization of the PAA representation of the time series into its SAX representation is to be implemented using an alphabet A of size $a > 2$. The values of the z-normalized time-series follow the Normal distribution. Based on those properties, a equal-sized areas under this Normal curve are picked using lookup tables for the cut lines coordinates. The list of coordinates of those lines is called a list of breakpoints. Such a list is defined by $B = (\beta_1, \beta_2, ..., \beta_{a-1})$ and divides the area under the normal curve into a equal areas.

Assigning a corresponding alphabet symbol $alpha_j$ for every interval $[\beta_{j-1}, \beta_j)$ enables to convert the vector of PAA coefficients $\bar{C}$ into a vector of strings $\hat{C}$ so that if the i^{th} component of $\hat{C}$ $\hat{c}_i \in [\beta_{j-1}, \beta_j)$, then $\hat{c}_i = \text{alpha}_j$.

Time series discords

Time series discords are subsequences of a time series that are consequently different from the rest of the time series subsequences. This concept has been introduced by Keogh et al. (2005) [24]. Those most unusual subsequences are used to detect anomalies in time series in a plethora of fields, including telemetry or medicine. Discords are well-adapted to anomaly detection given that only their length is required as a parameter to implement the related outlier detection algorithms. In order to find the greatest time series discords, several algorithms have been developed.

The Bruce-force Discord Discovery (BFDD) algorithm is one of these. It requires a time complexity of $O(n^2)$ to find discords. Another algorithm, the Heuristic Discord Discovery (HDD), was developed in order to reduce this high complexity whilst ensuring higher efficiency. The time series has to be discretized using the Symbolic Aggregate Approximation (SAX) technique into a symbolic string prior to applying the HDD algorithm. This pre-processing phase in addition to the HDD algorithm constitute the Heuristically Order Time series using SAX (HOT SAX) algorithm [25] and is faster than the BFDD.

4.2.2 HOT SAX algorithm

The brute force algorithm for finding the time series discords can be easily implemented. It relies on finding the distance to the nearest non-self-match for each possible subsequence of the time series. The discord is considered to be the subsequence with the greatest distance value. While the outer loop considers every possible subsequences, the inner loop enables to identify the candidates nearest non-self-match. Algorithm 7 [25] is the pseudo-code for the HDD version of the HOT SAX algorithm.

Algorithm 7. Pseudo-code for the HOT SAX algorithm [25]

```
1: Function [dist, loc] = Heuristic_Search(T, n, Outer, Inner)
2: best_so_far_dist = 0
3: best_so_far_loc = NaN
4: for each p in T ordered by heuristic Outer
5:     nearest_neighbor_dist = infinity
6:     for each q in T ordered by heuristic Inner
7:      if |p − q| ≥ n
8:       if dist(t_p, ..., t_{p+n−1}, t_q, ..., t_{q+n−1}) < best_so_far_dist
9:            break
10:      end if
11:      if dist(t_p, ..., t_{p+n−1}, t_q, ..., t_{q+n−1}) < nearest_neighbor_dist
12:           nearest_neighbor_dist = dist(t_p, ..., t_{p+n−1}, t_q, ..., t_{q+n−1})
13:      end if
14:     end for
15:     if nearest_neighbor_dist > best_so_far_dist
16:          best_so_far_dist = nearest_neighbor_dist
17:          best_so_far_loc = p
18:     end if
19: end for
20: return [best_so_far_dist, best_so_far_loc]
```

Figure 4-10 represents the anomalies found using the HOT SAX on the univariate time series of interest. Despite not labeling the cycles that are anomalous as such, given that most of the cycles are anomalous in this example, the algorithm is able to find most of the time discords in the time series.

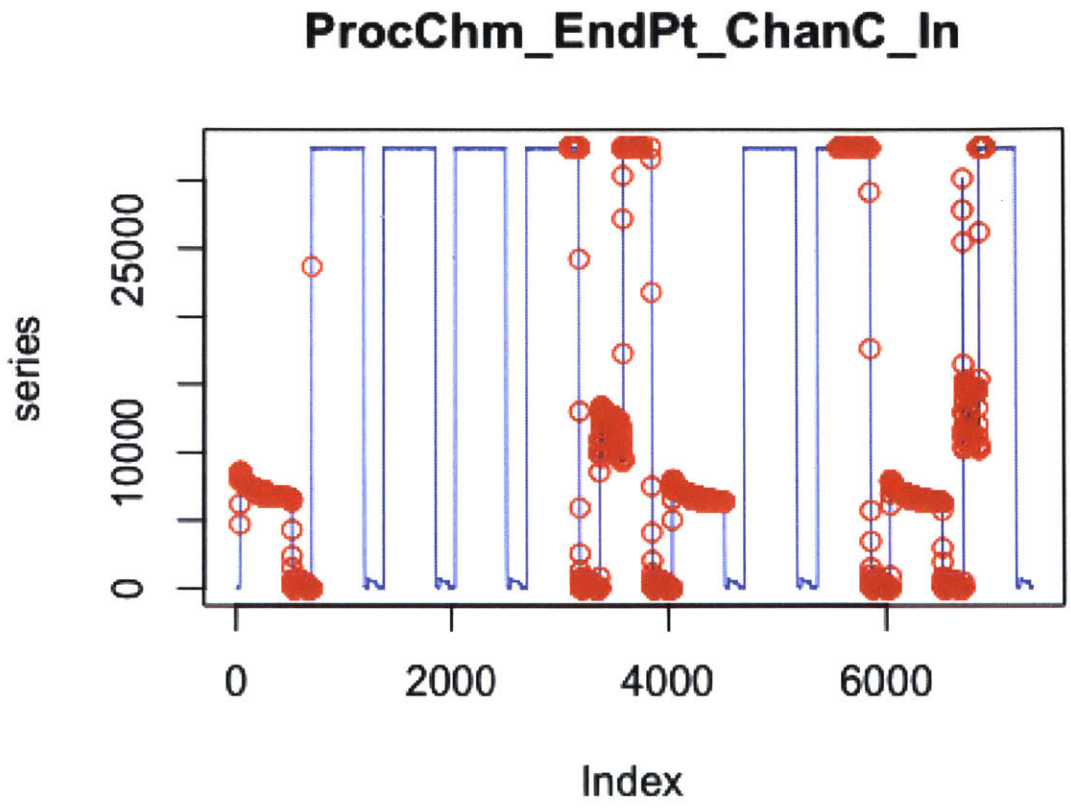

Figure 4-10: Plot of all the time series discords (red) found using a HOT SAX algorithm

This page was intentionally left blank.

Chapter 5

Conclusions and Future Work

This chapter is a conclusion of the conributions of this thesis. It captures the most important ideas of the anomaly detection techniques that have been implemented and discussed. In addition, it provides guidelines on further investigation to conduct in order to improve this work.

Clustering methods help identify potential anomalies in a data set of cycles of the same recipe or different recipes. Various algorithms have been developed throughout this thesis, including partitioning (k-means, k-medoids and CLARA) and hierarchical (agglomerative, divisive) clustering algorithms. Choosing the appropriate algorithm is crucial and depends on the application. If only one parameter of interest is deemed as being relevant to thoroughly encompass the characteristics of the data set, a univariate analysis can be performed in order to determine which cycles are to be labeled as "good" and those that are considered as being "bad". If a more thorough analysis is needed, one where multiple parameters are considered, a multivariate analysis is the best fit to cluster the data accordingly. In order to properly identify which cluster of cycles is good and which are bad, a reference cycle needs to be built using a data set of cycles labeled as good by process engineers. The reference cycle will be clustered along with the good cycles, thus making it possible to identify the cluster of good observations.

Extracting features from time series might improve the quality of the clustering. Decomposition-based methods enable such an advanced understanding of patterns in the data, in addition to detection of outliers. The STL decomposition relies on the analysis of the numerical components of the time series. While IQR is a measure of the statistical dispersion that labels outliers as data points that are outside predefined bounds, the GESD is a statistical test. The SAX method is a novel technique that relies on the symbolic representation of a time series.

Even though both the clustering and the decomposition-based anomaly detection methods have proven to be efficient, the results can be improved, especially those given by decomposition-based methods. In fact, once the data sets are clustered and the good and bad cycles are labeled as such, additional classification methods (SVM, k-NN, forest trees) can be implemented in order to delve more deeply into the analysis of the data. Given how fast and efficient the clustering and decomposition-based algorithms are in determining outliers, real-time analysis of the data collected on the plasma etcher could potentially be implemented. In addition, whilst the clustering algorithms aim at determining what cycles are anomalous, it would also be possible to compute algorithms that find anomalies at given time steps.

The team project can be optimized by combining all three individual research focuses: building a reference cycle [2] based on good cycles, then clustering the cycles according to whether or not they are anomalous can then help to train neural networks [3] towards a potential implementation of an automated anomaly detection method.

Bibliography

[1] J. Dillon, "MIT Presentation: Machine Health Project," p. 9, April, 2018.

[2] H. He, "Applications of Reference Cycle Building and k-Shape Clustering for Anomaly Detection in the Semiconductor Manufacturing Process," Master's thesis, MIT, Cambridge, MA, August, 2018.

[3] T. Chen, "Anomaly Detection in Semiconductor Manufacturing through Time Series Forecasting using Neural Networks," Master's thesis, MIT, Cambridge, MA, August, 2018.

[4] "Distance Metrics Overview," Web, https://home.deib.polimi.it/matteucc/Clustering/tutorial_html, September, 2003.

[5] "Introduction to Clustering," Web, http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering_Parameters/Distance_Metrics_Overview.htm, December, 2004.

[6] A. P. Reynolds, G. Richards, B. De La Iglesia, and V. J. Rayward-Smith, "Clustering Rules: A Comparison of Partitioning and Hierarchical Clustering Algorithms," *Journal of Mathematical Modelling and Algorithms*, vol. 5, no. 4, pp. 475–504, 2006.

[7] J. B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of Berkeley Symposium on Mathematical Statistics and Probability*, vol. 5, no. 1, pp. 281–297, 1967.

[8] A. Likas, N. Vlassis, and J. Verbeek, "The Global K-Means Clustering Algorithm.," *Pattern Recognition*, vol. 36, pp. 451–461, March, 2002.

[9] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A Comparative Study of Efficient Initialization Methods for the k-Means Clustering Algorithm," *Expert Systems with Applications*, vol. 40, no. 1, pp. 200–210, 2013.

[10] P. Arora, D. Deepali, and S. Varshney, "Analysis of K-Means and K-Medoids Algorithm for Big Data," *Physics Procedia*, vol. 78, pp. 507–512, December, 2015.

[11] L. Kaufman and P. J. Rousseeuw, "Finding Groups in Data: An Introduction to Cluster Analysis," *Wiley series in probability and mathematical statistics. John Wiley and Sons Inc.*, pp. 281–297, 1990.

[12] S. Vijayarani and P. Jothi, "An Efficient Clustering Algorithm for Outlier Detection in Data Streams," *International Journal of Computer Applications*, vol. 32, pp. 3657–3665, 2011.

[13] F. Murtagh and P. Contreras, "Algorithms for Hierarchical Clustering: An Overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[14] D. Mullner, "Modern Hierarchical, Agglomerative Clustering Algorithms," pp. 1–29, arXiv : 1109.2378, 2011.

[15] M. Roux, "A Comparative Study of Divisive and Agglomerative Hierarchical Clustering Algorithms," *Journal of Classification.*

[16] C. Shalizi, "Distances Between Clustering , Hierarchical Clustering," *Data Mining*, pp. 36–350, 2009.

[17] J. C. Bezdek, "FCM : The Fuzzy C-Means Clustering Algorithm," *Computers & Geosciences*, vol. 10, pp. 191–203, 1984.

[18] "Performance Metrics for Classification Problems in Machine Learning," Web, https://medium.com/greyatom/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b, November, 2017.

[19] A. Makiewicz and W. Ratajczak, "Principal Components Analysis," vol. 19, pp. 303–342, 1993.

[20] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A Seasonal-Trend Decomposition Procedure Based on Loess," *Journal of Official Statistics*, vol. 6, no. 1, p. 373, 1990.

[21] D. Hoaglin and J. Tukey, "Performance of Some Resistant Rules for Outlier Labeling," *Journal of the American Statistical Association*, vol. 81, no. 2, pp. 191–203, 1986.

[22] B. Rosner, "Percentage Points for a Generalized ESD Many-Outlier Procedure," *Technometrics*, vol. 2, no. 2, pp. 191–203, 1983.

[23] J. Lin, K. E., L. Wei, and S. Leonardi, "Experiencing SAX: a Novel Symbolic Representation of Time Series," *Springer Science*, vol. 15, pp. 191–203, 2007.

[24] E. Keogh, F. Bond, H. R., and J. Tilston, "Comparing Acceptance and Control-Based Coping Instructions on the Cold-Pressor Pain Experiences of Healthy Men and Women," *European Journal of Pain*, vol. 9, pp. 191–203, 2005.

[25] N. H. Kha and D. T. Anh, "From Cluster-Based Outlier Detection to Time Series Discord Discovery," *Proceeding Revised Selected Papers of the PAKDD 2015 Workshops on Trends and Applications in Knowledge Discovery and Data Mining*, pp. 16–28, 2015.