FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Clustering von Zeitreihen

Ein Überblick über Anwendungen von Zeitreihen-Clustering

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Georg Heiler
Matrikelnummer 1225063

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Peter Filzmoser

Wien, 25. Juni 2015

Georg Heiler	Peter Filzmoser

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Clustering time-series

An overview about different application contexts of time-series clustering

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Business & Information Systems Engineering

by

Georg Heiler
Registration Number 1225063

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Peter Filzmoser

Vienna, 25th June, 2015

Georg Heiler	Peter Filzmoser

Erklärung zur Verfassung der Arbeit

Georg Heiler
Nußbaumerstraße 24, 83278 Traunstein, Germany

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 25. Juni 2015

Georg Heiler

Acknowledgements

I want to sincerely thank Peter Filzmoser for giving me the opportunity to carry out this thesis. He helped me to remember to focus on the key topics.

I want to thank the research team for offering the possibility to write this thesis with the team at info-VIS (http://www.cvast.tuwien.ac.at/node/537). I am grateful for their effort, support and advice that helped this work to succeed.

I also want to thank my whole family and especially my parents who always encourage me in my actions and offer support.

Finally I want to thank my girlfriend who kindly improved my spelling in the thesis so gratefully.

Abstract

Time-series are becoming more and more important in the digitized industry 4.0. From forecasting of sales to increase the profit in retail industry, to real time streamed analysis for fraud detection, intrusion-detection, to medical applications e.g. combination of different time series (ECG, Blood, ...) for improvement of diagnoses, to applications in stock market. This work presents an overview of different application contexts of time-series clustering as a very hands-on, tutorial-like approach. Clustering time-series is often used to gain insight into the generating mechanism of the data in order to predict future values.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Overview

Clustering means identifying structure in an unlabeled set of data by segmenting it into groups of homogeneous values. As (**WarrenLiao2005**) explains, clustering can be used for any kind of input, but as they conclude "the bulk of clustering analyses has been performed on static data". There, static means that the values of features are constant over time. Usually it is possible to associate one object with a single cluster. Similar to clustering static data the choice of algorithm "depends on the type of data available as well as the particular purpose" (**WarrenLiao2005**). Several types of data can be differentiated:

- discrete: discrete-valued values (integers) are analyzed
- real-valued: e.g. sensor output which outputs non-discrete values
- uniformly sampled: the sample rate of the data is constant e.g. stock market
- non-uniformly sampled: variable sample rate e.g bank account transactions. For most clustering applications non-uniformly sampled data has to be normalized /converted into uniform data
- univariate: only one feature
- multivariate: multiple features
- length of the time-series: the length of the series may be equal or not. This has an impact on how a clustering algorithm works.

Generally speaking, there are three types of clustering algorithms: raw-data based (either frequency or time domain), feature based and model based (**Rani2012**). Time-series data

becomes more and more prevalent in a digitized industry, as more data is measured and analyzed in a time context.

From forecasting of sales to increase the profit in retail industry, to real time streamed analysis for fraud detection, intrusion-detection, to medical applications e.g. combination of different time series (ECG, Blood, ...) for improvement of diagnoses, to applications in stock market.

A lot of interesting problems for time-series mining are shown in Figure 1.1. What they

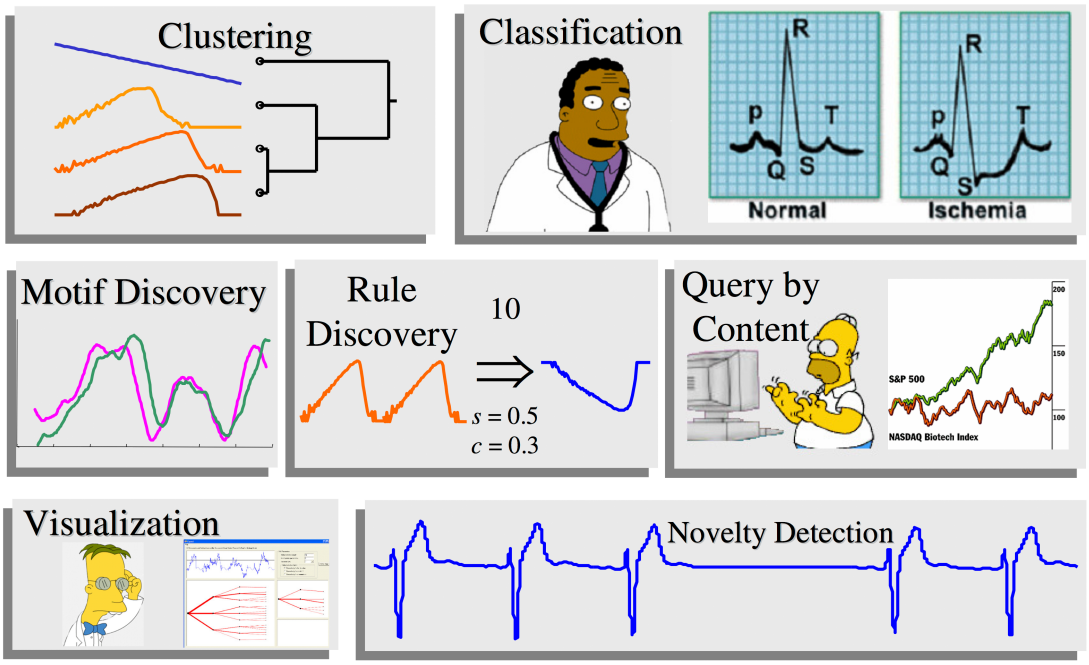

Figure 1.1: Similarity matching as the basis of a lot of time-series mining problems (**Keogh2006a**).

have in common is most of them require matching of similarity. Thus, clustering clustering plays a key part in a variety of problems related to time-series.

This work presents an overview of the different applications of clustering time-series as a very hands-on, tutorial-like approach. Clustering whole time-series as well as subsequences will be explained.

In general there are two use cases for clustering: stand-alone to get insight into the distribution of the data. Secondly it may be used for preprocessing the data to analyze the result with other algorithms (**clustering101**). As such clustering time-series is often used to gain insight into the generating mechanism of the data in order to predict future values (**Martinez-Alvarez2011**).

Major problems regarding time series data mining are:

- high dimensionality

- temporal order
- noise & missing values
- different sampling rates
- subjectivity of similarity

Keogh2006a concludes that three important issues are involved here:

- data representation. Efficient processing of time-series whilst retaining its essential characteristics
- similarity measurement. How can an intuitive measure of distance be formulated?
- indexing method. Structuring of a large set of time-series for fast querying

All three are key components of time-series mining systems. However, indexing will be out of scope of this work.

When approaching a problem of time-series mining, especially for prediction but also important for clustering, one has to decide whether a "black-box" solution such as a domain-independent pattern recognition system or a system based on domain knowledge should be used. **Keogh2006a** explains the difference as follows:

- "Black Box: Predict tomorrow's electricity demand, given only the last ten years electricity demand."
- "White Box: Predict tomorrow's electricity demand, given the last ten years electricity demand and the weather report, and the fact that the world cup final is on and ..."

It is apparent that a "black-box" system would have many advantages especially regarding usability, however in the recent years there have been only few attempts to solve this complex problem as **Keogh2006a** claims. Surprisingly, **noDomainKnowledgeNeeded** can show how successful Data Science can be sourced out to many people without domain knowledge. Competitions like Kaggle and KDD have proven that human 'black-box' systems can function very well.

1.2 General Thoughts

There are some general purpose clustering algorithms ready to use available even for clustering time-series. The only thing they need as an input is a reasonable distance matrix. There are some special dissimilarity measures for time-series. First we would like to introduce several ways of representing time-series, then explain dissimilarity measures which are the core part of a lot of clustering algorithms. We will focus on distance measures which are mostly used for time-series. Then we will explain some general, not necessarily time-series related clustering

approaches. The main part explains how to apply these for clustering time-series in several ways.

For enabling simpler and quicker processing, time-series usually have to be represented in a different format than CSV files.

1.2.1 Representation of time-series

Time-series are basically high dimensional data. Dealing directly with such complex data in a raw and uncompressed format is extremely expensive. Both in terms of storage coast and processing power. Therefore developing representation techniques which allow to reduce dimensionality of time series, while still preserving fundamental characteristics of the data is highly desirable. To enable fast processing it may be favorable to have different types of representation, depending on how the data is accessed (disk, tape or RAM).

Micro representation

The main idea is to compress the size of the data in order to fit in memory, but still retain the important features. An overview of the techniques is given in 6.1. Often, segmentation is used in order to summarize a time-series whilst still retaining its dominant features. Some of these approaches for representation are not adaptive to the data like Piecewise Linear Approximation (PLA) (**Shatkay1996**) or Piecewise Aggregate Approximation (PAA) (**YiB2000**; **Keogh2001b**). Wavelets (**generaltimeseries**) are the more modern approach of the Discrete Fourier Transformation (DFT) (**Bartolini2005**). Discrete Wavelet Transformation (DWT) introduced by **Shatkay1996** perform a scale-wise decomposition of time-series in such a way that only few coefficients are needed to represent the whole series. The original series is replaced by its wavelet approximation. As shown in Figure 1.2 only few Haar wavelets are needed to represent the time-series. The Haar wavelet is very popular and simple to use. Contrary to DFT, localized wavelets are used for the representation of the data and they capture both frequency and information about the location. One more point in favor of DWT is the optimal compression algorithm proposed by **Zhang2006** Efficient dimensionality reduction and preservation of as much original data as possible is automatically balanced. **generaltimeseries** tested on 65 datasets and all of these methods are the same on average. He concludes that in fact for 80% of the datasets the approaches are all within 10% of each other.

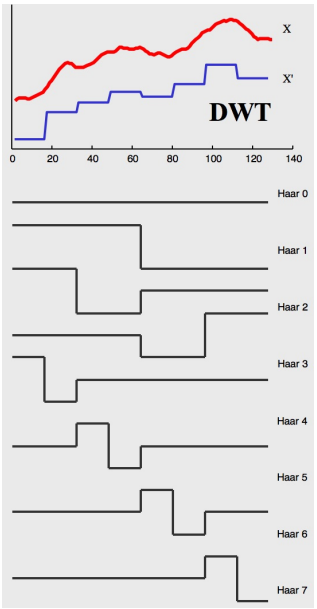

Figure 1.2: DWT only needs very few coefficients to represent a time-series(**generaltimeseries**)[4]

Symbolic Aggregate Approximation (SAX) (**Lin2003**)tries to symbolize the series. It is based on the same approach as PAA. The idea is to transform the search for similarity into a problem of finding subsequences in symbolic sequence data. **Shieh2008** improved this method to support fast indexing. It

is called indexed Symbolic Aggregate Approximation (iSAX) and allows faster querying of time-series.

Loïc Dutrieux https://github.com/dutri001/bfastApp offers an interesting application to explore the segmentation if irregular time-series interactively.

Macro representation

Recently, more and more time-series data was collected and the term *temporal database* was shaped. The idea is that the database system in itself especially supports the storage of time-series. It promotes a multi-faceted view of time.

A simple use-case for such a database would be the storage of addresses. For some, storing only the current address is enough. Nevertheless, a lot of organizations need to track historical data. A temporal database would allow for these applications and support special queries to cope with these concepts of time. In SQL:2011 some basic temporal functionality was standardized. However, this standard is not yet widely adopted. Commonly used databases like Postgres or Oracle often only implement only parts of this standard (**temporalDBMakro**). For especially long time-series (**tsdatabases**) recommends using Apache HBase or MapR-DB and explains why classical RDBMS are not sufficient.

Nearly every clustering algorithm requires a measurement of the distance. Depending on the type of data a specific measure will be more appropriate than another.

CHAPTER 2

Dissimilarity measures for time-series

The function used for computing the distance is a key component of the clustering algorithm. It is not only important for the sole purpose of clustering but also critical for classification and regression of time-series. As shown in Figure 2.1, similarity is subjective. Thus it is important to find some *objective* and consistent means of similarity or dissimilarity between time-series. A similarity measure should provide the following attributes (**Esling2012**):

Figure 2.1: Similarity can be confusing (**Keogh2006a**).

- consistent with human intuition
- recognition of similar objects even though they are not identical
- emphasis on salient features on local and global scale
- universally usable, meaning no restrictions for time series are assumed
- abstraction from noise and invariant to a set of transformations

In the literature several types of transformations are known: scaling, warping, additive noise and added outliers at random positions. The dissimilarity measure should be robust to combinations of these types of transformation. There are different possibilities to structure these measures. One approach is to differentiate four categories to measure dissimilarity:

- shape based

- edit based
- feature based
- structure based

Another - more mathematical - approach is to structure dissimilarity measures based on whether they are metric like Euclidean or non-metric like dynamic time warping (Dynamic Time Warping (DTW)).

In the following section we will give a brief introduction about the different types of measures.

In the following section $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{Y} = (y_1, y_2, \ldots, y_n)$ will be time-series of equal length.

2.1 Shape based distance

Shape based distance measures compare the overall shape of the data.

2.1.1 L_p norms

A classic and well understood distance is the euclidean distance, in general known as L_p norms (*Minkowski Distance*). As (**Keogh2003a**) points out, it is even one of the most widely used measures. Humans abstract the aforementioned types of transformation intuitively when observing characteristics of a time-series. The Euclidean distance is simple but can not provide such a high level of abstraction. A further downside is that L_p norm based measures are rigid metrics. This may be good for some applications but time-series of different lengths are not supported. However, as (**Shieh2008**) have shown, L_p based measures "provide advantages in the case of very big data sets, as there is a larger probability that an almost exact match exists in the database".

$$d_{L_P}(\mathbf{X}, \mathbf{Y}) = \sqrt[p]{\sum_{k=1}^{N}(x_k - y_k)^{\frac{1}{p}}}$$

In case $p = 1$ it is named Manhattan distance, in case $p = 2$ it is called Euclidean ($d_E = d_{L_2}$) and in the generalized case of $1 < p < \infty$ it is called Minkowski distance.

2.1.2 Short time-series distance (STS)

The Short Time-Series Distance (STS) proposed by **Moller-Levet2003** is a good choice for short and irregularly sampled time series. They define STS as:

$$d_{STS}(\mathbf{X}, \mathbf{Y}) = \sqrt{\sum_{k=1}^{N}\left(\frac{y_{k+1} - y_k}{t_{k+1} - t_k} - \frac{x_{k+1} - x_k}{t'_{k+1} - t'_k}\right)}$$

As mentioned already $\mathbf{X}$ and $\mathbf{Y}$ are two time-series, where t and t' are their respective temporal indexes and t_k and t'_k refer to a certain position in these indices. Both series must be of equal length. Even though the temporal indexes may start at different positions, their increments must be equal.

2.1.3 DISSIM distance

Another measure for irregular time series is the DISSIM distance. It was introduced by **Frentzos2007** to overcome the weaknesses of traditional work in similarity query processing. Earlier work either focused only on the time dimension of trajectories, or only considered trajectories of the same sampling rate. It did not combine both.

It is based on the definite integral of the Euclidean distance between time-series. To avoid the expense of calculating the integral, they proposed an approximation:

$$Dissim\_approx(\mathbf{X}, \mathbf{Y}) = \sum_{k=1}^{N-1} \left(d_{E_{x_{t_k}, y_{t_k}}} + d_{E_{x_{t_{k+1}}, y_{t_{k+1}}}} \right) * (t_{k+1} - t_k)$$

Again $\mathbf{X}$ and $\mathbf{Y}$ are two time-series, D_{x_t,y_t} is the Euclidean distance between the series at the point t out of the global time index $T = \{t_1, \ldots, t_N\}$.

2.1.4 Dynamic time warping (DTW)

DTW has been designed to overcome shortcomings of the Euclidean Distance. It was introduced by **Berndt1994** to compare different speech patterns and is highly popular nowadays. It not only supports the comparison of series of different lengths but is capable of dealing with a lot of transformations like warping and shifting.

As shown in Figure 2.2, unlike the L_p norms where the time axis is fixed and the sequences are aligned one to one, DTW enables a non-linear alignment of the sequences. DTW tries to find the best alignment of the series. The optimal path is defined as the shortest warping path in the distance matrix (D). The Euclidean distance between a pair of points in time x_k, y_k defines D. **WarrenLiao2005** explain three restrictions which bound this optimization problem. A start at $D(0,0)$ and an end at $D(N,M)$ is forced by the boundary. Where N is the last index of $\mathbf{X}$ and M is the last index of $\mathbf{Y}$. The $Rest(?)$ function takes the not yet processed time-series as the argument.

$$dtw(\mathbf{X}, \mathbf{Y}) = \begin{cases} 0, & \text{if } M = N = 0 \\ \inf & \text{if } M = 0 \text{ or } N = 0 \\ d(x_0, y_0) + minDTW(Rest(\mathbf{X}), Rest(\mathbf{Y})), & \\ \{DTW(Rest(\mathbf{X}), \mathbf{Y}), DTW(\mathbf{X}, Rest(\mathbf{Y}))\} & \text{otherwise} \end{cases}$$

Calculating DTW is rather expensive. **Keogh2005** developed a measure which is based on DTW. However, it utilizes lower bounds via the *Sakoe-Chiba* band which make it a lot more efficient. Instead of $O(n^2)$ only $O(n)$ is needed for computation.

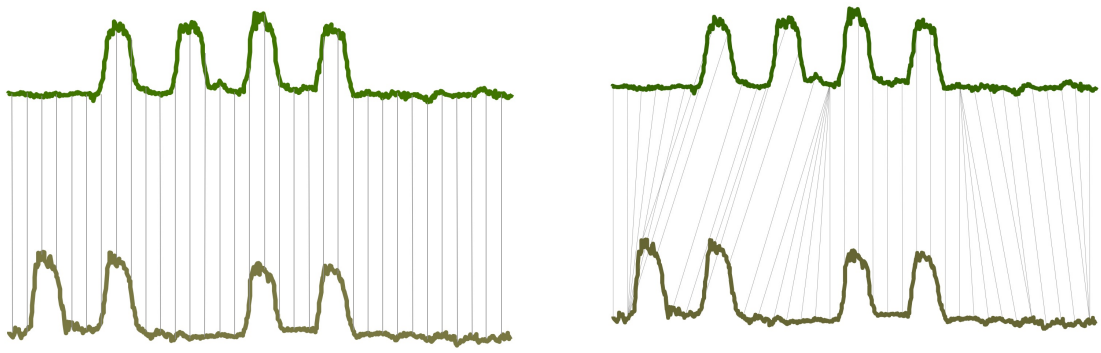

Figure 2.2: Euclidean vs. dynamic time warping. (**cs14**).

2.2 Edit-based distance

Originally, *Edit distance* was used to measure the difference between two sequences of strings. The idea is to transform the initial time-series with a minimal number of insertions, deletions and transformations into the other. Compensation for outliers is achieved by allowing gaps in the matching. The following distance measures are classified as common according to **Esling2012**; **Ding2008**

2.2.1 Longest common sub-sequence (LCSS)

Longest Common Subsequence (LCSS) is a prominent example for an edit based measure. **Vlachos2002** proved that LCSS is more robust than DTW under noisy conditions if the threshold settings are chosen wisely. The calculation of LCSS is based on the following recursion:

$$LCSS(X,Y) = \begin{cases} 0, & \text{if } M = 0 \text{ or} \\ N = 0 & \\ LCSS(Rest(X), Rest(Y)) + 1, & \text{if } |x_0 - y_0| \leq \epsilon \\ \max\{LCSS(Rest(X), Y), LCSS(X, Rest(Y))\} & \text{otherwise} \end{cases}$$

Again $\mathbf{X}$ and $\mathbf{Y}$ are two time-series and the $Rest(?)$ Function takes the not yet processed time-series as the argument. As before N is the last index of $\mathbf{X}$ and M is the last index of $\mathbf{Y}$. Usually such a recursion is solved via dynamic programming.

Working with real numbers, there is a problem to find the exact matching point in floating point arithmetics. Several adaptions have been proposed in literature to cope with this problem.

2.2.2 Edit-distance for real sequences (EDR)

A common method in computer science for dealing with real (=possibly infinitely long) numbers in the limited memory of a computer is the e-comparison. This means two points

are considered equal if the distance is $<$ than a supplied e. Otherwise, the distance itself is considered. Edit Distance for Real Sequences (EDR) utilizes such a comparison. In contrast to LCSS penalties are assigned by EDR (**Chen2005**) depending on the length of the gaps.

2.2.3 Edit-distance with real penalty (ERP)

Edit Distance with Real penalty (ERP) by **Chen2004** tries to combine DTW with EDR in order to have a distance function which supports local time shifting but still is a metric in the mathematical sense. With some improvements it delivers superb pruning performance and delivers a noticeable speedup when searching through large time-series databases.

Several extensions of edit based distance have been proposed by **Marteau2009**; **Fuad2008**; **Chhieng2007**

2.3 Feature-based distance

Instead of calculating the similarity of time-series based on raw values, feature based distances extract feature vectors and calculate the similarity between these. The advantage is reduced processing power especially for long time series as only the characteristic features are classified and less time is wasted dealing with noise. Pearson's correlation and Cross correlation are commonly used to select the most important periods of a series. The metrics presented above deal directly in the time domain. The frequency domain offers some interesting alternatives: A Fourier coefficients based distance can be defined by the Euclidean distance of the first n Fourier coefficients. Features can be extracted using different types of correlation, DWT or simply by extracting recurrent motifs of a series.

2.3.1 Correlation-based distances

A simple example for a correlation based distance metric is the *Pearson's Correlation* factor.

$$COR(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{k=1}^{N} \left(x_k - \overline{X}\right)\left(y_k - \overline{Y}\right)}{\sqrt{\sum_{k=1}^{N} \left(x_k - \overline{X}\right)} \sqrt{\sum_{k=1}^{N} \left(y_k - \overline{Y}\right)}}$$

$\overline{X}$ and $\overline{Y}$ are average values of the time-series $\mathbf{X}$ and $\mathbf{Y}$.

Golay1998 use Pearson's Correlation as a building block for a k-means clustering as follows:

$$d_{COR}(\mathbf{X}, \mathbf{Y}) = \sqrt{2(1 - COR(\mathbf{X}, \mathbf{Y}))}$$

Others like **Galeano2000** use an autocorrelation based distance function.

2.3.2 Periodogram-based distances

A simple approach for a spectral dissimilarity measure is the periodogram-based distance proposed by (**Caiado2006**). It is simply defined as the euclidean distance of the first

periodiograms of the time-series. $\mathbf{X}$ and $\mathbf{Y}$ are two time-series of length N. Let $I_X(\lambda_g) = N^{-1}\left|\sum_{k=1}^{N} x_k e^{-i\lambda_g k}\right|^2$ and $I_Y(\lambda_g) = N^{-1}\left|\sum_{k=1}^{N} y_k e^{-i\lambda_g k}\right|^2$ be the periodograms of $\mathbf{X}$ and $\mathbf{Y}$ at frequencies $\lambda_g = \frac{2\pi g}{N}$, with $g = 1, \ldots, n$ with $n = \frac{N-1}{2}$. Then the Euclidean distance between them can easily be calculated as:

$$d_P(\mathbf{X}, \mathbf{Y}) = \frac{1}{n}\sqrt{\sum_{g=1}^{n} (I_X(\lambda_g) - I_Y(\lambda_g))^2}$$

In case only the correlation structure is of interest normalization of the periodogram should be considered. Variance correlates to the spectrum value of corresponding frequencies. Thus using the logarithm of normalized periodograms can make sense.

Thompson2010 builds on this and proposes a measure based on cumulative version of the periodograms. He argues that integrated periodograms have several advantages.

2.3.3 Motif-based distances

Similar to other feature-based dissimilarity measures certain features are extracted & feature vectors compared. Instead of selecting features similar subsets (called motifs) are selected. This does not directly result in a rigid measure of dissimilarity. However clustering these motifs for example using them as the initialization of a k-means clustering algorithm (**Phu2011**) is possible. **Lin2010** describes how motifs are discovered using sub-series joins. **Vahdatpour2009** deals with motif discovery in a multi-dimensional environment where motifs have temporal, length and frequency variations. **Chiu2003** explains how to discover motifs in a probabilistic way. An example of motifs is depicted in Figure 2.3.

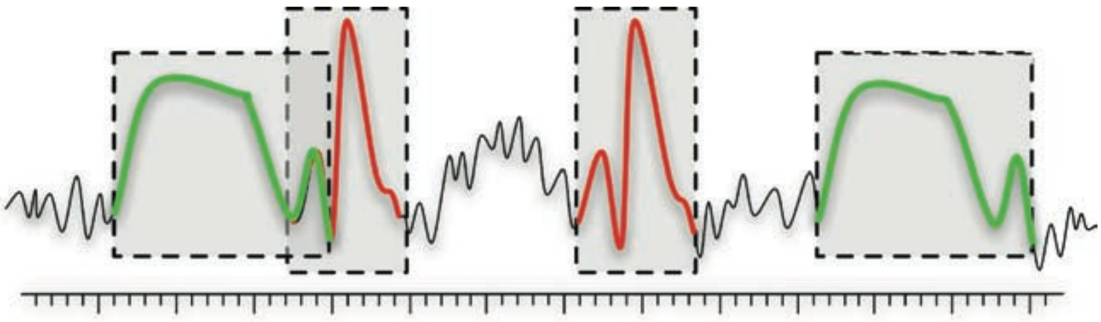

Figure 2.3: Motifs are recurring patterns in a series. (**Esling2012**)

2.4 Model-based distance measures

Previous methods are not well suited for extremely long time-series. Model based similarity tries to find higher-level structures on a bigger, more global scale. Prior knowledge about the

process can be used for improving the measure of similarity. Different types of parametric temporal models can be used. Hidden Markow Models (HMM) or Autoregressive Moving Average (ARMA) processes are common. The best result is achieved when the model fits to the underlying process. Thus, there are a lot of possibilities to parametrize these models. Some Hidden Markow Model (HMM)'s (**Ghassempour2014**) deal with clustering multivariate time-series, others like **Wang2011**; **Strelioff2014** try to use pattern based Hidden Markov Model (pHMM) in order to utilize patterns for finding the semantics of time series. **Mesot2006** combines AR processes with HMM into a Switching autoregressive moving average Hidden Markov Model (SAR-HMM). They propose a Bayesian approach to prevent over-fitting the model in order to compensate for issues faced by AR processes. Other common model based distances are: Piccolo (**Piccolo1990**), Maharaj (**Maharaj2000**) & Cepestral (**Kalpakis2001**) -based distances.

2.4.1 Piccolo distance

Piccolo1990 describes a distance measurement which is based on the Euclidean distance between the $AR(\infty)$ operators. He states that the autoregressive part of the stationary ARIMA process contains most parts of useful information about the processes structure. This method requires the time-series to be stationary and not contain seasonality.

2.4.2 Maharaj distance

Maharaj2000 defined a distance similar to Piccolo distance. However, here noise is used.

$$d_{MAH}(\mathbf{X}, \mathbf{Y}) = \sqrt{N} \left(\mathbf{\Pi}'_X - \mathbf{\Pi}'_Y \right)^{\mathsf{T}} \mathbf{V}^{-1} \left(\mathbf{\Pi}'_X - \mathbf{\Pi}'_Y \right)$$

$\mathbf{\Pi}'_X$ and $\mathbf{\Pi}'_Y$ are $AR(p)$ parameter estimations of $\mathbf{X}$ and $\mathbf{Y}$ which are time-series and $AR(p) = \sum_{k=1}^{p} a + \phi_k x_{N-k} + u_N$. a and ϕ are two unknown parameters and u_N is white noise. The p is the same as the one selected by Piccolo's distance. $\mathbf{V}$ is an estimation for the variance of the white noise. The main difference between Piccolo and Maharaj distance is that Maharaj takes this variance into consideration.

2.5 Prediction-based approaches

All the other dissimilarity measures mentioned before were focused on the past. However in many applications the main interest of clustering relies directly on clustering the predictions. An example is a sustainable development project or any situation where it is important to reach target values on pre-specified future time periods. It is useful to instead of clustering the past deal with clusters for future values **tsclust Alonso2006** propose to measure dissimilarity as a comparison forecast densities. The variability of predictions is fully taken into account however their model only supports $AR(1)$ processes. **Vilar2010** generalizes this approach. **Lee** deals with clustering time-series based on the forecast distributions using Kullback-Leibler divergence.

2.6 Compression-based approaches

Many data-mining algorithms require a lot of parameters to be set. This means it is a time consuming process setting these parameters. But the even bigger problem is that these parameters have to be chosen in a sensible way. Sometimes only time consuming experiments lead to the right parameters. **compression** proposes a compression based dissimilarity measure which is simpler and easier to use and can compete and sometimes outperform classic approaches. The proposed algorithm is named CDM and based on Kolmogorov complexity A downside is that this approach can only deal with high dimensional data.

2.7 Comparison of distance measures

Choosing the adequate measure of distance is important as the efficiency of a clustering algorithm highly depends on a measure of distance which fits best to the nature of the analyzed data. For short time series visual methods or shape based methods are meaningful. If the data is very specific, expert knowledge can be used and model-based approaches might be better suited. Whereas shape-based techniques deal mainly with local comparisons and work well with short series, structure-based approaches aim to compare the underlying dependence structure and can cope with long time-series better. Feature-based approaches are best used when periodicities are the main focus.

generaltimeseries states that dynamic time warping (DTW) is the best shape based distance available for short time series so far. For long time-series he recommends two different approaches depending on prior knowledge. If nothing is known about the data he recommends compression based distance, if there is some prior knowledge he would try to leverage it for feature extraction

As shown in Figure 2.4, it can be clearly seen how the choice between shape- and structure-based similarity influences the clustering result. If similarity is measured based on geometric similarity, P1 and P2 are forming a mixed cluster. If the underlying correlation structure is used to identify proximity, P1 and P3 move closer together.

As **Esling2012** point out, the accuracy of the similarity distance chosen still has to be evaluated even when following the aforementioned rules. **Keogh2003a** have proposed a time-series data mining benchmark to cope with this problem and to help to better and more reproducible research. **Rooyen** tries to improve the theoretical understanding of measuring the performance of feature learning.

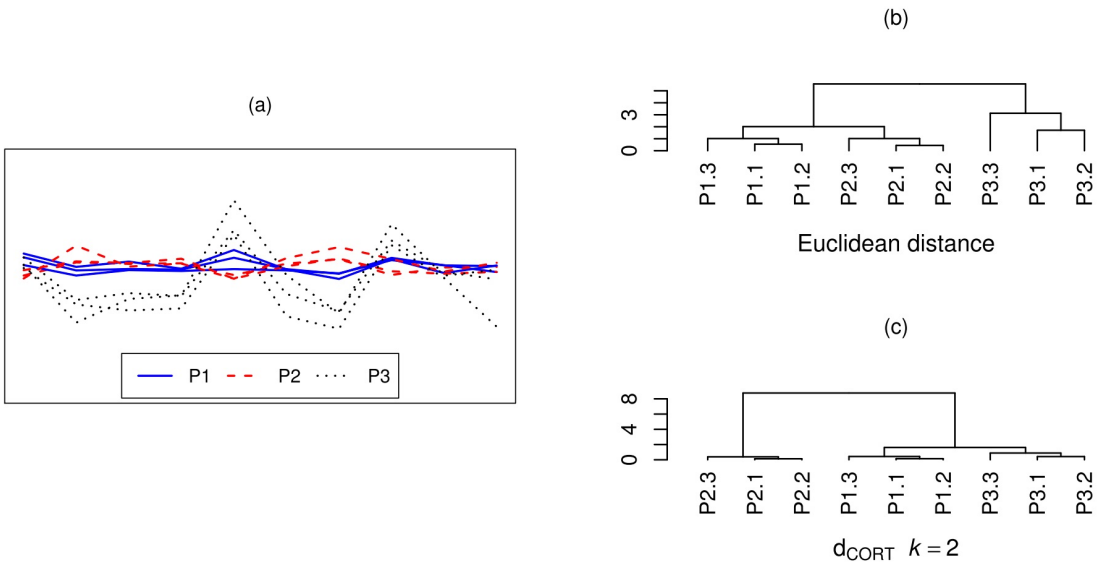

Figure 2.4: a) shows several time-series generated from patterns p1, p2 and p3. The Dendograms depict the difference of choosing a b) shape-based distance or a c) structure based distance for clustering (**tsclust**).

CHAPTER 3

Clustering algorithms

Clustering is a task which belongs to unsupervised learning and thus a sub-category of machine-learning. In contrast to classification where labels are already supplied, for cluster analysis machines have to learn on their own. In general there are two use cases for clustering: stand-alone to get insight into the distribution of the data or as a preprocessing step for other algorithms (**clustering101**).

Generally speaking, there are three types of clustering algorithms: raw-data based (either frequency or time domain), feature based and model based. **Rani2012** Depending on the technique used **Han2012** differentiate partitioning, hierarchical, density-based, grid, model-based and probabilistic and generating methods clustering techniques. **WarrenLiao2005** applied three of them for time-series.

For real world clustering scenarios often demand the integration of several clustering methods and thus cannot easily be classified to one category uniquely (**Han2012**).

3.1 Distance-based partitioning methods

Most algorithms are distance-based, in fact distance based clustering has been most extensively studied as a branch of statistics (**Han2012**).

Partitioning approaches like k-means (**Vlachos2003**), k-medians (**Har-peled2004**), k-medoids (**Kalpakis2001**) and hierarchical algorithms like agglomerative or divisive methods like **Rodrigues2004** are sub categories of distance based clustering algorithms.

Both distance based clustering methods are fairly generic and can be used if a fitting distance matrix is provided. Often such algorithms are already implemented in your tool of choice.

Most of the distance functions explained earlier in Section 3 output such a matrix and thus can be utilized to make use of these generic algorithms in a time-series context.

The interested reader will find a lot more distance metrics in the literature. But **Mori2012**; **tsclust** already implemented a lot of them ready to be used in further research projects.

3.1.1 Partitioning methods

Partitioning methods often require the number of clusters to be specified up front, like k-means. In general it is the simplest clustering method. Partitioning methods provide a one-level grouping of the data. This means each of the k groups must contain at least one object. This method tries to improve the clustering progressively, like k-means (**MacQueen1967**). **Jain2010** provides a good overview about modern clustering algorithms. Other methods are PAM, CLARA and CLARANS. Figure 3.1 explains the main differences between partitioning and hierarchical methods visually.

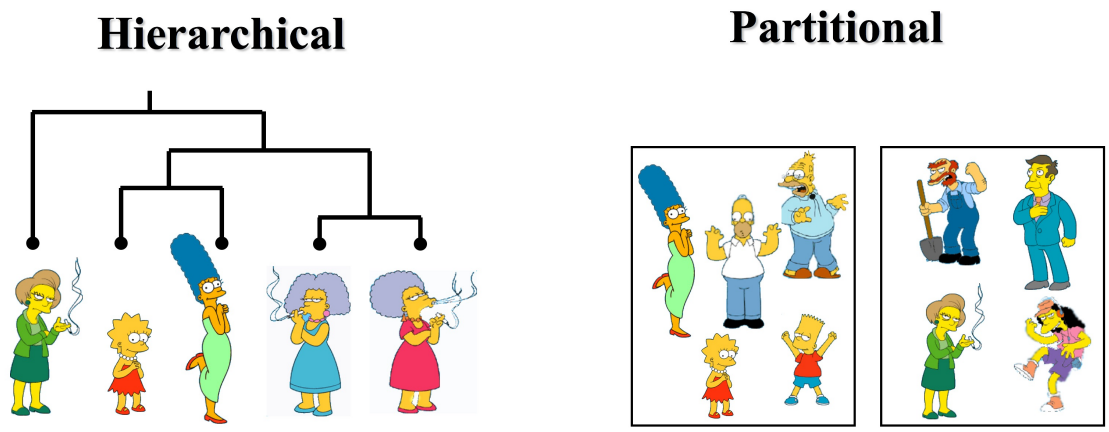

Figure 3.1: The difference of partitioning and hierarchical methods is clearly depicted. Hierarchical methods support a multi-level composition (**imagesEamon**).

3.1.2 Hierarchical methods

Hierarchical methods provide a multi-level composition of the data. This method was introduced by **Zhang1996** as BIRCH to bypass computational limitations to allow for clustering data which would exceed the working memory. It can be either agglomerative (bottom-up) or divisive (top-down). Figure 3.1 explains the agglomerative approach visually. It can be seen how the clusters grow and subsequent levels of similarity are added. A downside of using hierarchical clustering is that when a merge or split has been performed it can never be undone. This may lead to a reduction in computation time, however might not result in the globally optimal solution.

3.2 Density-based methods

Most clustering methods can only recognize spherical clusters, as they are based on the distance between objects (**Han2012**). Density-based clustering can grow a cluster as long as the number of objects in the neighborhood is above a certain threshold. Thus density-based clustering easily deals with noise, filters out outliers and can recognize clusters of

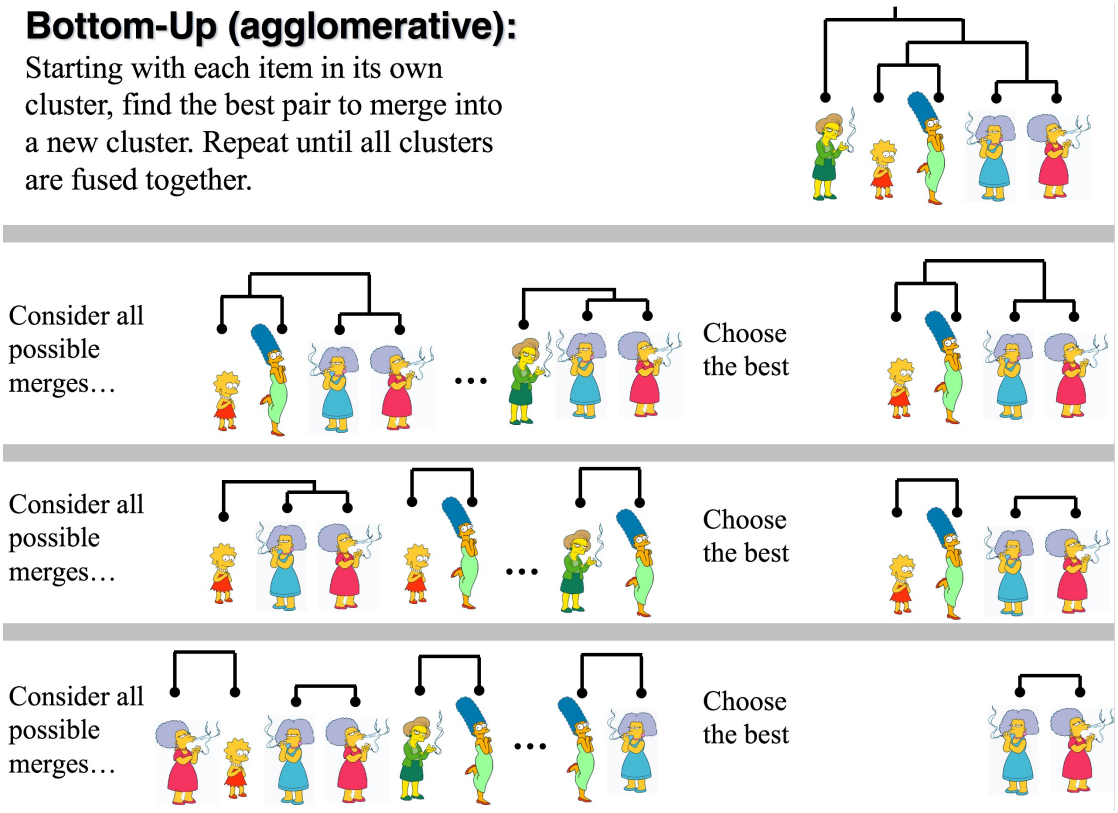

Figure 3.2: Agglomerative clustering explained (**imagesEamon**).

arbitrary shape. Examples for such methods are DBSCAN (**Ester1996**; **Chakraborty2011**) and AUTOCLUST (**Estivill-Castro2001**). It builds on DBSCAN but includes dense and thin bridges which connect clusters.

3.3 Grid-based methods

Instead of dealing with raw values, grid-based methods quantize the objects in a grid-like space. The main advantage of grid-based methods is that the time needed to perform clustering is independent of the amount of data but only depends on the size of the grid. As a result, grid-based methods are often integrated with other methods (**Han2012**). A good example is Self Organizing Maps (SOM) (**Fu2001**).

3.4 Probabilistic and generating methods

Probabilistic and generating methods are often used in conjunction with fuzzy clustering (**Han2012**) where one object may be grouped in multiple clusters. Gaussian mixture models (GMM) (**Gorur2010**; **gaussModel**) and expectation maximization (EM) (**Xiong2002**) are

good examples for such algorithms. Mixture models assume that observations are drawn from one of several components and thus can infer the parameters of these. EM tries to find the maximum likelihood estimations in mixture models. Sometimes (**courersaClusterAnalysis6**) probabilistic methods are put as a sub-category of model-based methods as probabilistic model-based clustering methods.

Iterative progress of a GMM is demonstrated in Figure 3.3.

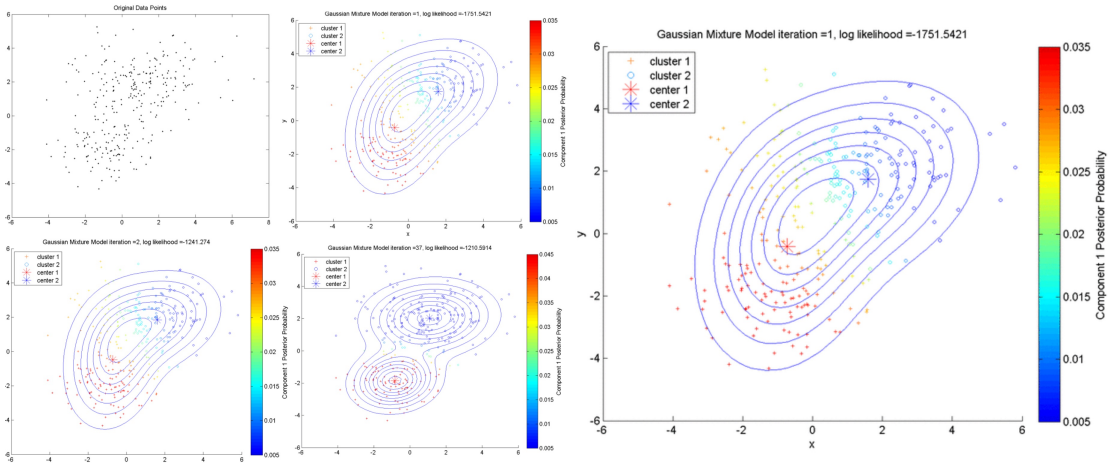

Figure 3.3: Execution of the EM algorithm for a 2D dataset (**courersaClusterAnalysis6**).

3.5 Model-based methods

Model based similarity tries to find higher-level structures on a more global scale. Apriori knowledge about the process can be used for improving the dissimilarity measure. HMM (**Ghassempour2014**), SAR-HMM (**Mesot2006**), utilizing model based distances (as explained in section 2.4) and SVM (**Alvarez2010**) are common for model-based clustering.

Wang2011; **Strelioff2014**; **semanticsInMultipleTimeSeries** go one step further and try to use variations of HMM in order to find the semantics of time series.

After fitting, the model dissimilarity can be measured between these. Often model based approaches assume certain regularity conditions, e.g. the time-series have to be stationary and the underlying processes must be linear.

3.6 Comparison of clustering algorithms

Clustering can be performed on raw data or in a derived space created using features or models. Such a derived space may be formed either in the time or frequency domain (**Fokianos2012**; **Wellens2009**). The approach which fits best to the specific clustering task has to be chosen. Fairly generic clustering algorithms like distance based clustering algorithms may be used either on raw data or in derived space depending on the dissimilarity measure and the input.

However these can mostly deal only with spherical clusters. Density based methods are more flexible as arbitrary clusters can be identified. Grid-based methods are independent of the input. Their runtime only depends on the size of the grid.

There is no objectively "correct" clustering algorithm (**Keogh2006a**). Similarity can be confusing and thus is in the eye of the beholder. Choosing the most appropriate clustering algorithm for a particular problem needs to be conducted experimentally. As **superLearner** shows, it is fairly reasonable to combine the results of several algorithms. This might especially be a good idea as the clustering task will be performed in presence of corrupt data. **Rooyen2015** provides a framework to deal with corrupted data.

The concepts of the proposed algorithms are now applied to time-series data.

CHAPTER 4

Explanation of the different application contexts of time-series clustering

Most problems regarding time series require a matching of similarity. Thus, clustering is the basis of a variety of problems related to time-series. Depending on the need different approaches for clustering have been defined. Clustering might consider the similarity of time-series or be focused on finding repeating patterns. The idea is to identify natural groups in the data set. The basic problem of clustering is to identify the correct number of clusters. This holds true for non-time-series based clustering algorithms as well. As already mentioned in the introduction most techniques assume static data. Therefore, time series need a special treatment. As **agronomyTSClustering** concludes, it is common to consider a time series T of length N as a single data instance with N attributes. And use the possibility to apply the classical techniques. A different approach is to modify classical clustering techniques to support handling time-series data. Several approaches are outlined in the next section in a tutorial hands-on like manner.

4.1 Preparation

In general the data will have to be cleaned, e.g. corrput records will have to be identified and then corrected or removed. But time series require further processing: time-series recorded at different scales need to be normalized to be comparable (**Rakthanmanon2012**). Figure 4.1 shows the effect of normalization. As **tsclust** conclude it is especially important for distance based clustering algorithms to include a normalization step to prevent common misunderstandings.

Depending on the approach chosen for clustering e.g. model based clustering there may be further preparation tasks involved. For example model based distance measures usually

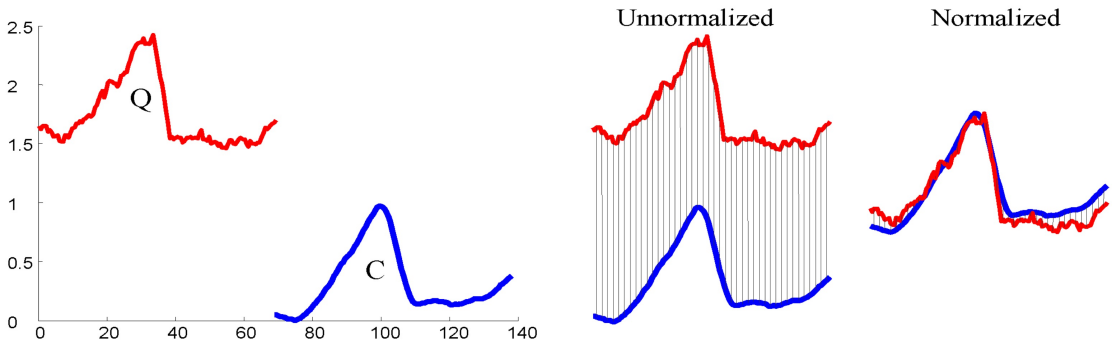

Figure 4.1: Unnormalized data greatly overstates the subjective dissimilarity distance (**Ratanamahatana2010**).

assume certain regularity conditions, like stationary time-series with a linear process.

Time series usually are a sequence of equally spaced data points. However there a use cases when such a regularity condition is not fulfilled as for example with credit card transactions: sometimes there is one transaction per day, sometimes none and sometimes there are multiple. Such a series is referred to as irregular. As a lot fewer approaches exist to deal with irregular time-series, converting it into a regular series should be considered. Sometimes, interpolation like calculating the mean between between two values and filling the gaps, is possible. But such an approach may not be applicable for certain types of data. Consider the example from above: credit card transactions are discrete and they either do happen or not.

4.2 Clustering a set of whole time-series

A lot of time-series clustering algorithms deal with clustering whole series in a set. Identifying selling patterns or finding stocks that behave similar are good examples for grouping whole time-series.

4.2.1 Examples

Simple hierarchical and agglomerative clustering

A simple example for hierarchical clustering depicts how easy it is to replace a clustering algorithm if the distance matrix fits well for the data to be analyzed. In this case dynamic time warping (DTW) is used.

A classic time-series mining dataset is the control charts dataset developed by Alcock and Manolopoulos (1999). It contains 600 examples of control charts synthetically generated. The data can be separated into six different classes: Normal, Cyclic, Increasing trend, Decreasing trend, Upward shift or Downward shift. Some randomly chosen time-series from this set (see Section 6.4) can be utilized to show how simple hierarchical clustering can be performed.

For Example in R performing this task is a two-liner as only the distance matrix has to be passed to the algorithm. First the data samples from the synthetic control charts time-series are chosen. Figure 4.2 gives an overview about the randomly chosen series. The rest of the code is needed for preparation of the data.

```
sc=read.table("~/pathToData/sc.data.txt",
+ quote="\"", comment.char="")
n=2
s <- sample(1:100, n)
idx <- c(s, 100+s, 200+s, 300+s, 400+s, 500+s)
sample2 <- sc[idx,]
observedLabels <- c(rep(1,n), rep(2,n), rep(3,n),
+rep(4,n), rep(5,n), rep(6,n))

library(dtw)
library(cluster)

distMatrix <- dist(sample2, method="DTW")

hc <- hclust(distMatrix, method="average")
plot(hc, labels=observedLabels, main="")
```

Changing only a few lines instead of top down-hierarchical clustering an agglomerative approach can be pursued based on the same DTW distance matrix. Figure 4.3 shows the dendogram generated by hierarchical clustering and Figure 4.4 the dendogram created by the agglomerative approch. It can be clearly seen that the order of the dendograms is inverted.

```
ag <- agnes(distMatrix, method="average")
plot(ag, labels=observedLabels, main="")
```

4.2.2 HMM

Hidden Markov models are a popular choice of model-based algorithms used for sequential data-mining. They are based on the assumption that the system being modeled follows a Markov process. For a simple Markov model as depicted in Figure 4.5 the state is visible to the observer. This simple model contains two states A and E and the probabilities of the choices. In Hidden Markov models the state is hidden. However the output ins dependent on the state and thus the original states can be reconstructed. **Ghassempour2014a** formulate a simple example of a HMM in the context of health care. Such a "Health HMM" contains two hidden health states: sick and healthy as well as the probabilities for the transitions between these states. Instead of observering the states blood temperature and the amount of white blood cells is tracked. As the output is dependent on the hidden state we can infer whether the person was sick or not.

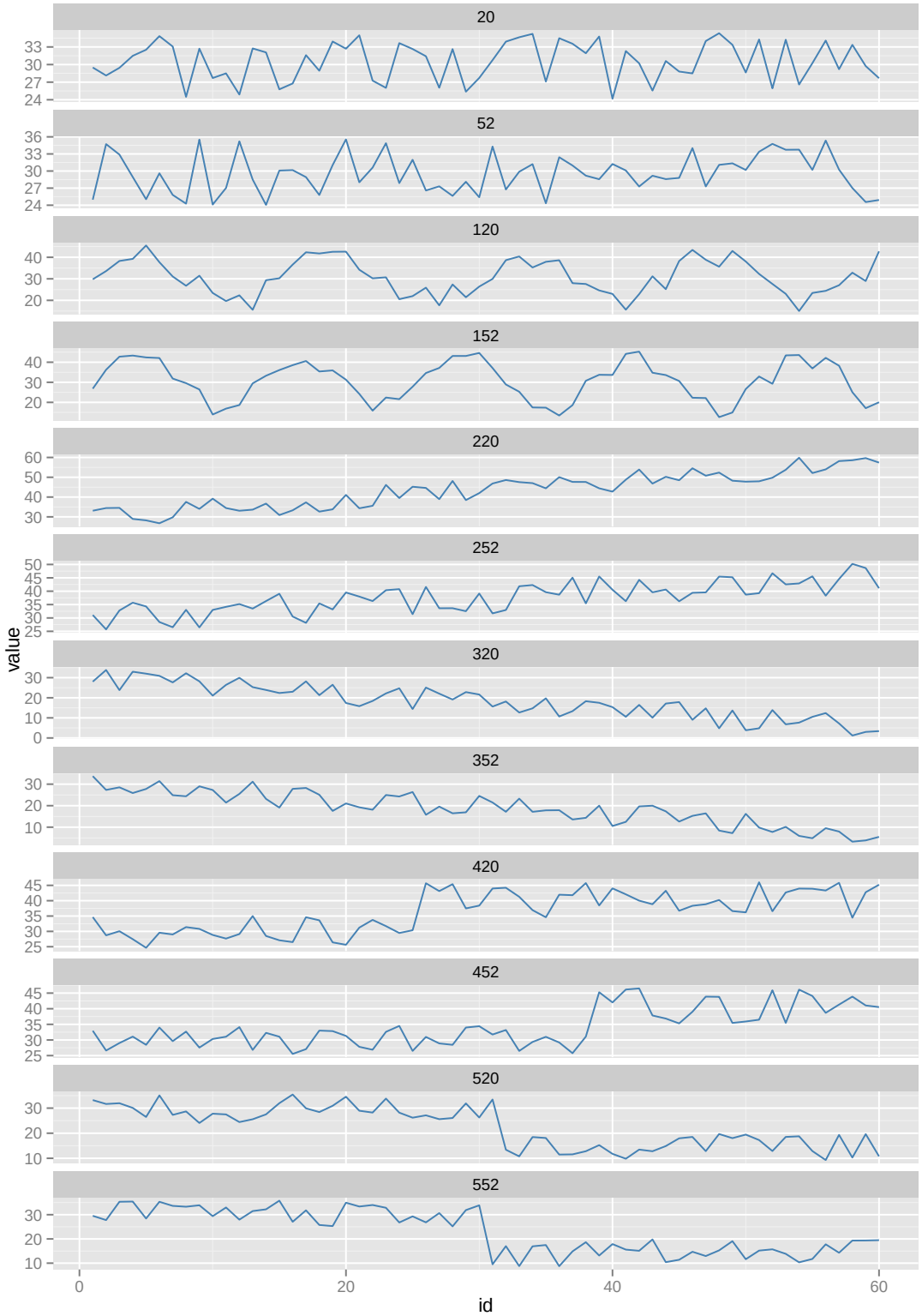

Figure 4.2: Overview about the 12 randomly sampled time-series from the synthetic control charts dataset

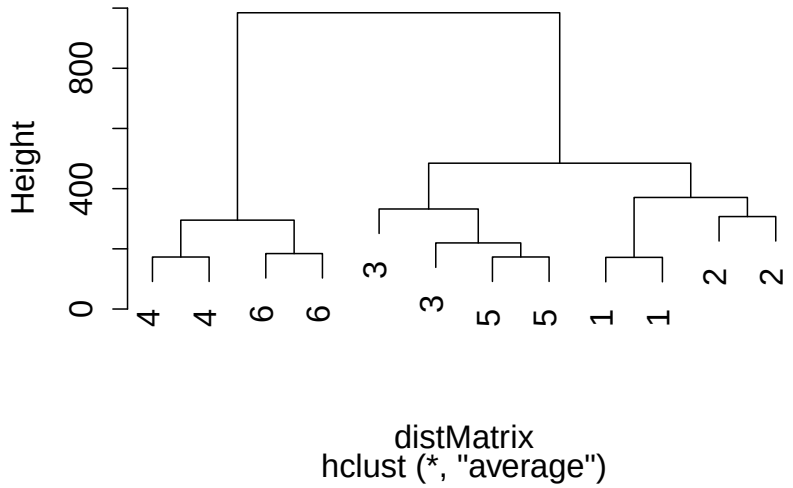

Figure 4.3: Hierarical clustering of synthetic control time-series data

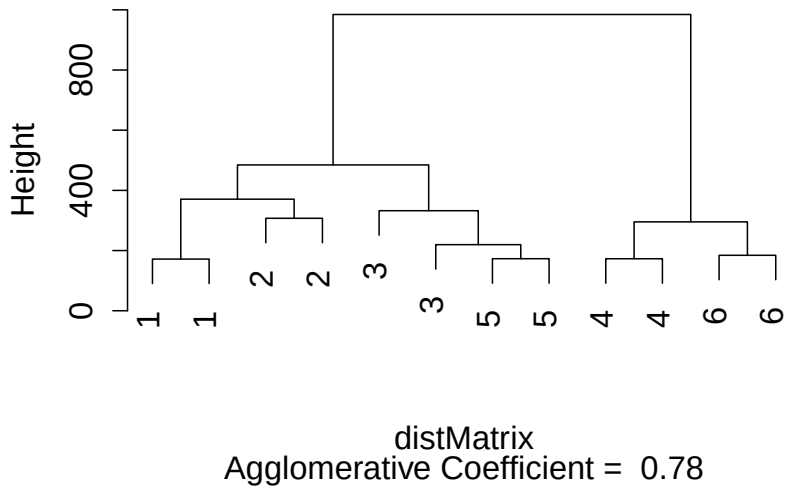

Figure 4.4: Agglomerative clustering of synthetic control time-series data

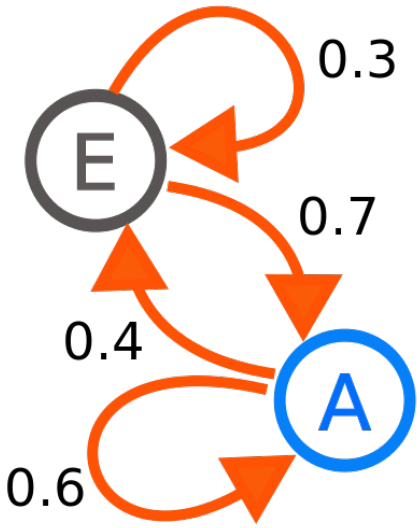

Figure 4.5: A very simple Markov Model containing two states (**markovSimple**).

HMM's are very popular in applications like speech recognition or other fields where the temporal character of the data is important. The interested read will find a lot of literature regarding HMM on the internet. **introHMM** provide a good introduction to HMM.

Example

Ghassempour2014a use a HMM to cluster multivariate time-series with categorical and continuous variables. They deal with representation of health trajectories of individuals. They try to keep their approach in a fairly simple way in order to make their results accessible to a wider audience which might not possess advanced statistical knowledge. As such they use already existing packages from R. For the implementation of HMM the *depmixS4* package using Expectation Maximization (EM) is used. They explored several algorithms but the best results were reached utilizing Partitioning Around Medoids (PAM) from the *cluster* package. To combine these building blocks a distance measure is needed. **Ghassempour2014a** propose the following distance: The idea is that the likelyhood $P(T|\lambda_i)$ is used as a probability density on the health trajectories. One good example from literature for a distance measure between probability densities is the Kullback-Leibler divergence measured between two densities $P(T|\lambda_i), P(T|\lambda_j)$ of the to models λ_i, λ_j.

$$d_{KL}(P(T|\lambda_i), P(T|\lambda_i)) \equiv \int dT P(T|\lambda_j) \log\left(\frac{P(T|\lambda_i)}{P(T|\lambda_j)}\right)$$

T denotes a multivariate time-series. This is fairly hard to compute. Therefore **Ghassempour2014a** suggest a compromise between a Markov Monte Carlo (MCMC) estimation strategy and a

one-point estimation. As such they replace the probability density with a discrete probability distribution.

4.2.3 Spectral clustering

Spectral analysis of time-series analyzes the variance over the frequency. In general it is very suitable for analysis of periodic signals corrupted by noise (**Cowpertwait2011**). Thus spectral analysis is well suited for time-series as **Wang2005a**; **Jebara2007** show.

yoshkkk introduced a general spectral distance:

$$d_w(\mathbf{X}, \mathbf{Y}) = \frac{1}{4\pi} \int_{-\pi}^{\pi} W\left(\frac{f_X(\lambda)}{f_Y(\lambda)}\right) d\lambda$$

$f_X(\lambda)$ and $f_Y(\lambda)$ are the spectral densities of the time-series $\mathbf{X}$ and $\mathbf{Y}$. $W(\cdot)$ is a function to satisfy regular conditions in order to keep distance-like attributes. As $f_X(\lambda)$ and $f_Y(\lambda)$ are unknown for practical applications they need to be estimated. **Fan1998**; **tsclust** show three possibilities:

- $d_{W(DLS)}$ uses local linear smoothers
- $d_{W(LS)}$ uses exponential transformation of locally smoothed log-radiograms
- $d_{W(LK)}$ but instead of using least squares like $d_{W(LS)}$ it uses maximum local likelihood criterion

Example

The R packages *TSclust* and *cluster* allow to perform spectral clustering easily.x

```
library(TSclust)
library(cluster)
df = as.data.frame(t(as.matrix(sample2)))
IP.dis <- diss(df, "INT.PER")
IP.hclus <- hclust(IP.dis)
plot(IP.hclus, labels=observedLabels, main="")
```

Figure 4.6 depicts the result of clustering the same set of synthetic control series used above. It can be seen that the result is fairly different from other approaches mentioned earlier. But still for most of the clusters it is true that $4, 6$ and $1, 3, 5$ are separated.

4.2.4 Permutation-based clustering

Similar to compression based approaches permutation distribution clustering is an alternative complexity-based procedure. The core part of this approach is a m-dimensional embedding which is sorted and permuted in order to measure divergence of patterns.

$$\chi_m^{'} \equiv \left\{\mathbf{X}_m^{'} = (X_t, X_{t+1}, \ldots, X_{t+m}), t = 1, \ldots, \mathbf{T} - m\right\}$$

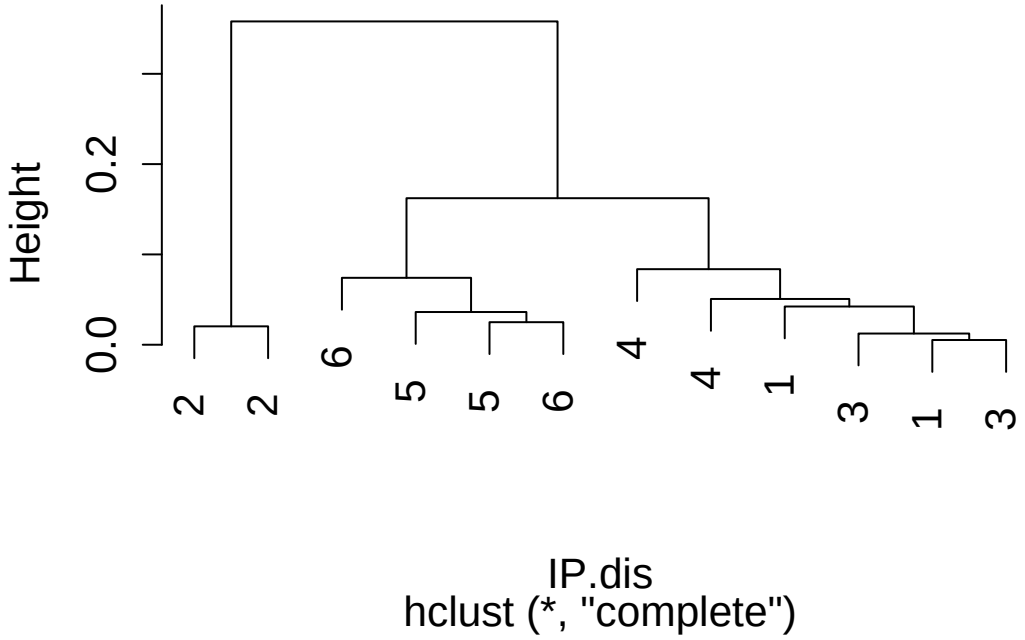

Figure 4.6: Clustering the same synthetic control series using a spectral approach.

A permutation $\Pi(\mathbf{X}_m^{'})$ is obtained for each $\mathbf{X}_m^{'} \in \chi_m^{'}$ by sorting $\mathbf{X}_m^{'}$, the so called codebook of $\mathbf{X}_m^{'}$. The complexity is characterized by the codebook of $\mathbf{X}_T$, where T refers to the time-index of the time-series $\mathbf{X}$. It is defined as the distribution of the permutations on $\chi_m^{'}, P(\mathbf{X}_T)$. Thus now dissimilarity can be defined as the discrepancies between the codebooks of $\mathbf{X}_T$ and $\mathbf{Y}_T$. As the choice of m is crucial to obtain a valuable clustering result a heuristic was presented by **cranpdc** and is available in the *pdc* package.

Example

Here we show how simple permutation-based clustering can be performed using the *pdc* package.

```
library(pdc)
plot(pdclust(df), cols=observedLabels, labels= observedLabels)
```

Figure 4.7 depicts the result of clustering the same set of synthetic control series used above. Similar to Figure 4.6 again the result differs compared to previous approaches. But here it is not true that $4, 6$ and $1, 3, 5$ are separated.

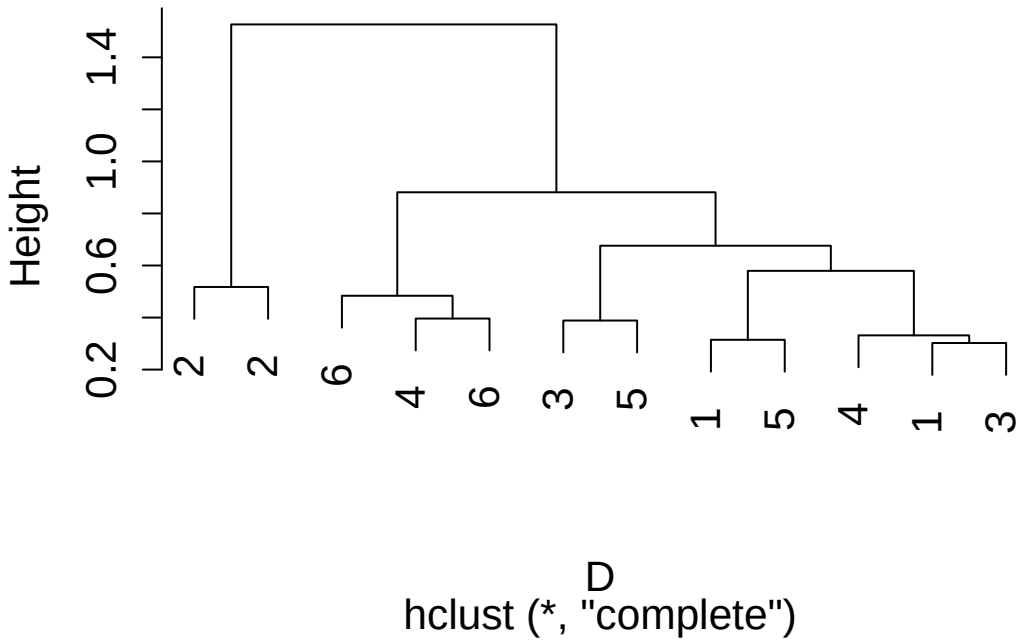

Figure 4.7: Clustering the same synthetic control series using permutation-based approach.

4.2.5 funFEM - functional mixture models

bikeSharing recently introduced a new algorithm for clustering of time-series. The algorithm allows for model based clustering of time-series or more general functional data and is called funFEM. Functional data (http://www.psych.mcgill.ca/misc/fda/) tries to describe the data as spline functions. funFEM is based on functional mixture models which allow for clustering in a discriminative functional subspace. Bike sharing is fairly popular in a lot of big cities. However the bikes have to be spread about a city evenly in order to ensure that a bike station is neither empty nor too full. Bike sharing systems (BSS) usually allow for real-time reports of bike stations, which often results in a collection of very large amounts of data. Thus automatic algorithms need to be used for the analysis. Clustering is often used for such purposes. **bikeSharing** explain that former research focused on classifying usage profiles of BSS. **Froehlich2008** were one of the first ones to analyze BSS data. However they only used traditional clustering approaches which cannot exploit the temporal dynamic of the data. Others like (**Lathia2012**; **Vogel2011**) are still limited to one BSS (one city) and do not make use of the functional nature of the data. Figure 6.2 and Figure 4.8 show the bike stations clustered by when they get bikes. The latter also depicts the clustering result in a spatial context. It can be clearly seen that the result differs on weekends and the location of

the bike station. Using funFEM presents the advantage to be parsimonious. Therefore it is

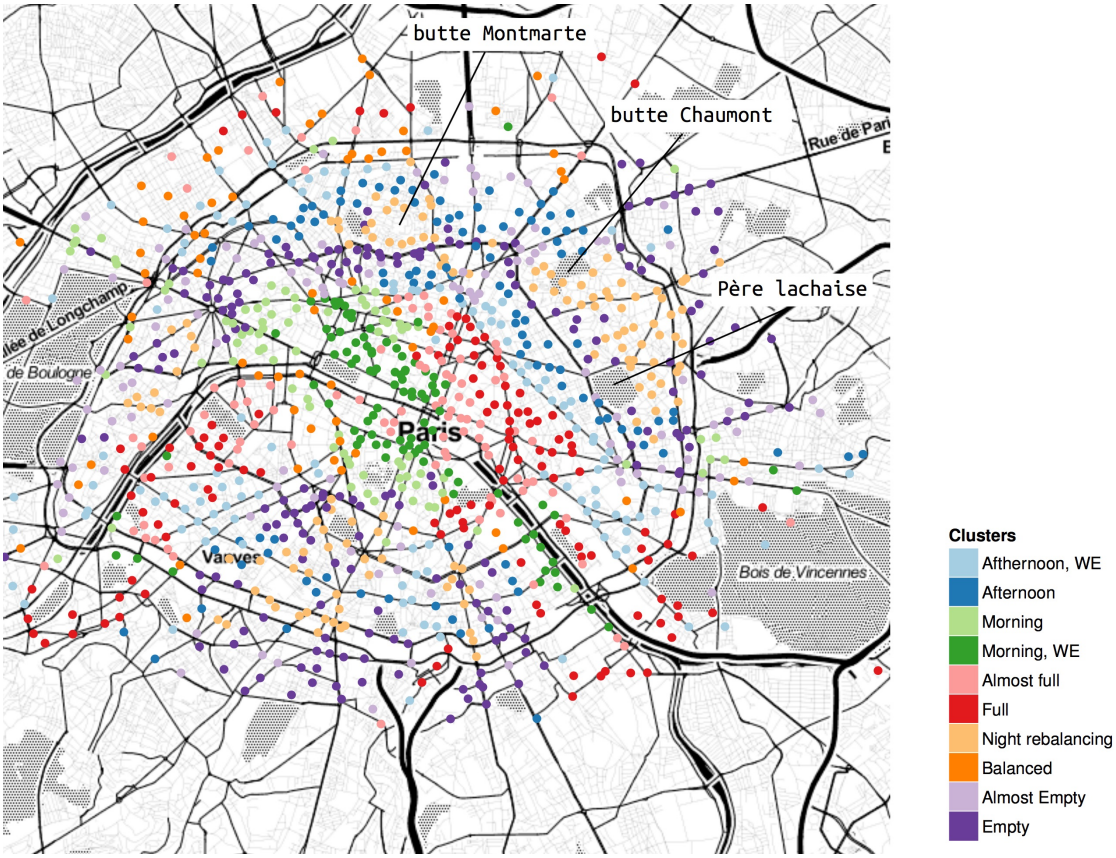

Figure 4.8: Map of the clustering results for Paris stations (**bikeSharing**).

well suited for handling long time series.

This algorithm can easily be explored in a simple example (**bikeSharing**):

```
library(funFEM)
basis <- create.bspline.basis(c(0, 365), nbasis=21, norder=4)
fdobj <- smooth.basis(day.5, CanadianWeather$dailyAv[,,"
    ↪ Temperature.C"], basis,
                fdnames=list("Day", "Station", "Deg C"))$fd
res = funFEM(fdobj,K=4)

par(mfrow=c(1,2))
plot(fdobj, col=res$cls, lwd=2, lty=1)
fdmeans = fdobj; fdmeans$coefs = t(res$prms$my)
plot(fdmeans, col=1:max(res$cls), lwd=2)
```

Figure 4.9 depicts the result of clustering the "Canadaian temperature" dataset.

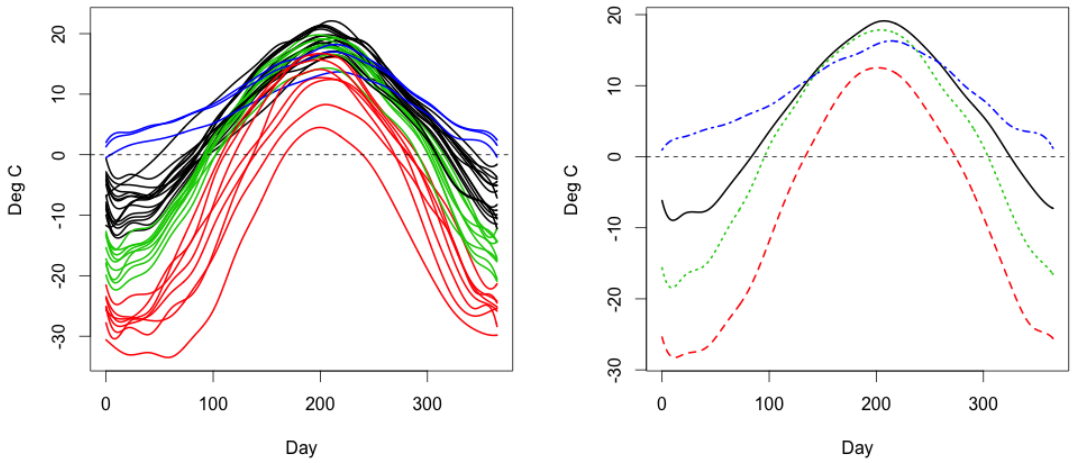

Figure 4.9: Clustering the well-known "Canadian temperature" data using funFEM (**bikeSharing**).

4.2.6 WEKA for time-series clustering

WEKA http://www.cs.waikato.ac.nz/ml/weka/ is an open sourced collection of machine learning algorithms which can be used either via a GUI or an API. It is developed by the University of Waikato in New Zealand. It is implemented in Java and fairly user friendly. Figure 4.10 shows how WEKA can be used to use clustering without programming.

4.3 Identification and grouping of patterns within a single time-series

Often it is very interesting to collect insights from pieces within a time-series. Clustering subsequences usually means slicing a single or multiple time-series into pieces and grouping these pieces into classes. These pieces can be obtained using feature extraction algorithms or constant segmentation. Some methods choose to slice in non-overlapping windows based on the periodical structure of the series. (**Hebrail2001**) In case this structure is weak or not present at all non-overlapping slicing may miss important structures. Thinking forward the next step would be to use overlapping structures. However (**Keogh2004**) drew the conclusion that this is meaningless. They propose a solution to this problem clustering motifs to prevent trivial matches. (**Liu2005**; **Fu2005**) tries to find motifs in a "meaningful" way to re-enable subsequence clustering. (**Goldin2006**) proposes a new distance measure for this purpose. It is based on the shape of the cluster.

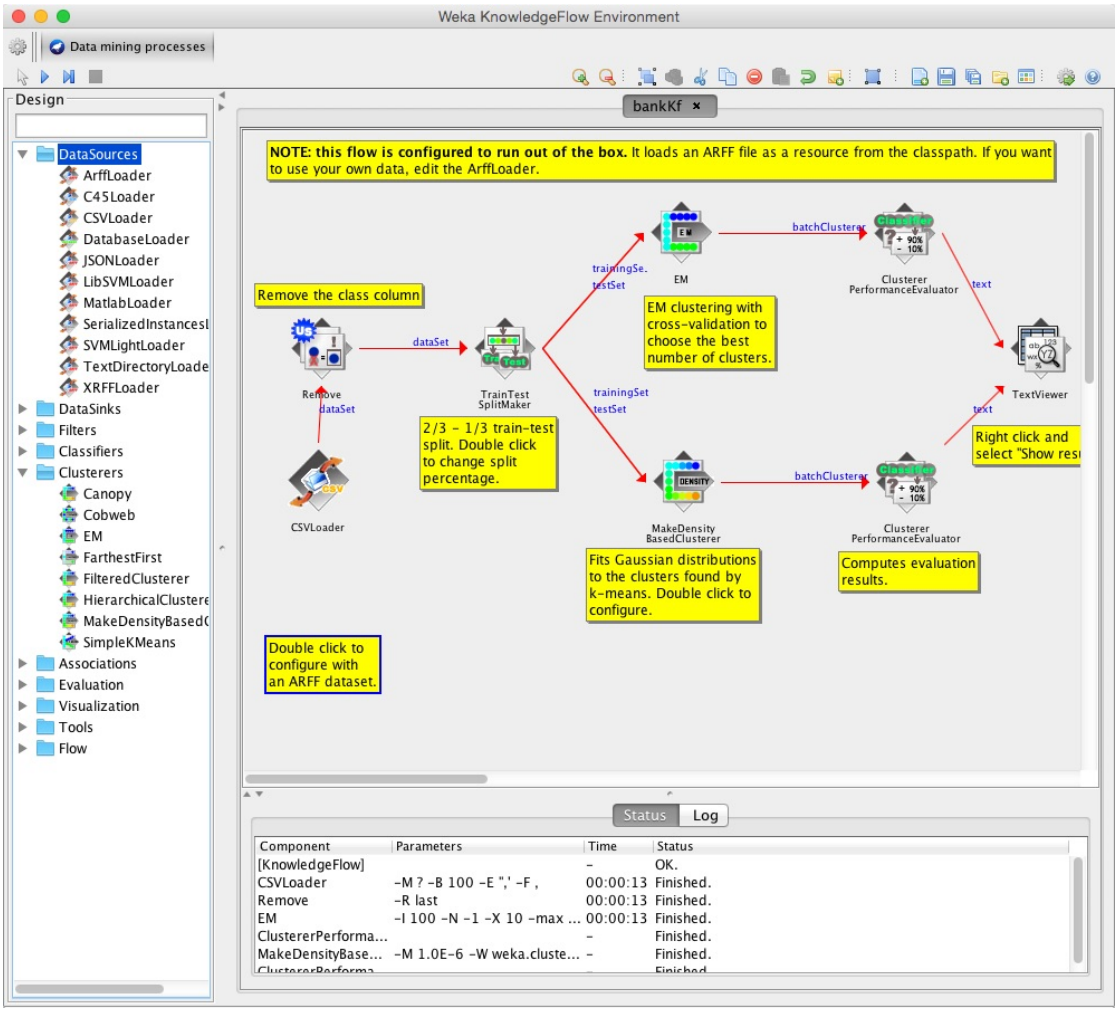

Figure 4.10: WEKA for clustering using EM and density based clustering

4.3.1 Examples visualizing sub-sequences

Grammarviz is a nice tool to visualize patterns in time-series. Figure 4.11 shows subsequences calculated from recurrent patterns identified in the AirPassengers dataset using Grammarviz.

4.4 Smart pattern detection and outlier analysis

Clustering and outlier detection are in a complementary relationship. (**Aggarawal2013**)

In contrast to subsequence matching, anomaly detection deals with identifying previously not known patterns. The idea is to recognize surprising patterns. In general a surprising behavior is defined as behavior which deviates from "normal" state however this context dependent and subjective. As a lot of anomaly detection algorithms rely on modeling normal

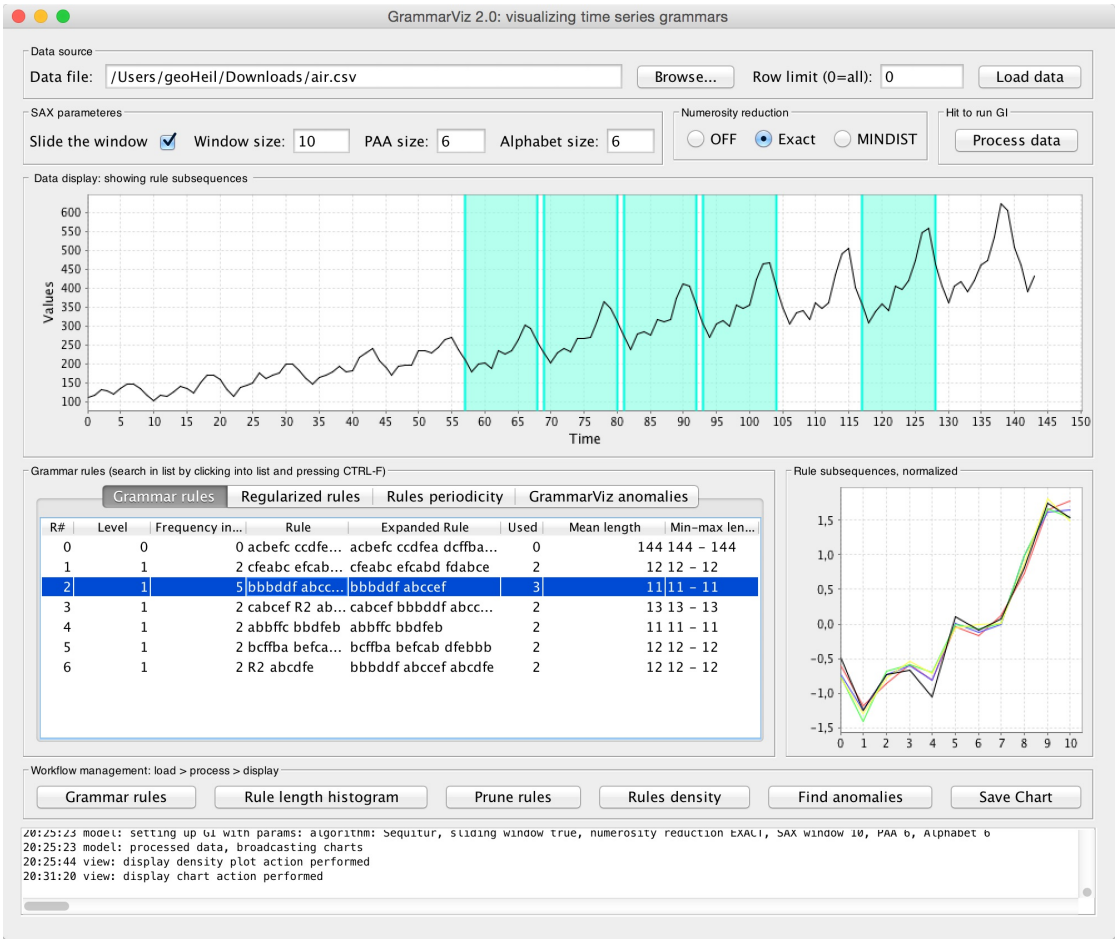

Figure 4.11: Recurrent patterns in the AirPassengers dataset

behavior with a set of typical shapes and predict dissimilar shapes motif discovery is a key part of such an algorithm. A motif is an important subsequence or pattern within a time-series. The definition from (**saxSeries**) is very suitable as they define an anomalous pattern as a pattern where the frequency of occurrences significantly differs from what would be expected.

The knowledge drawn from clustering the data can also be applied to classify outliers. Here values that are far away may be more interesting than common cases. A very common application of outlier detection is fraud detection in e-commerce transactions or recently in big-data security analytics.

Outlier detection can be performed in several ways. Proximity based models utilizing clustering are one basic and common way to detect outliers. (**Aggarawal2013**) **Thiprungsri2012** applies clustering to automate fraud filtering during an accounting audit.

4.5 Change point detection

Segmenting a time-series into pieces of (nearly) constant values can be used to drastically reduce noise, dimensionality and save storage space and processing power. As such it is a topic very related to the representation of time-series which was explained in Section 1.2.1. Change point detection is similar, but tries to gain analytical value through intelligent algorithms to detect structural changes in a system, this means identifying points in time where the trend changed. Several approaches exist for such a purpose. Clustering is one of them (**James**).

4.5.1 Example

The R package *ecp* allows for easy utilization of change point detection. The key advantages of using the *ecp* package are the nonparametric nature of the algorithms and the possibility to use them for multivariate data and multiple change points.

This code is used to detect the change points in the sample dataset of the synthetic time series which was introduced earlier.

```
library(ggplot2)
library(reshape2)
library(ecp)

synthetic_control.data <- read.table("/Users/geoHeil/Dropbox/6.
    ↪ Semester/BachelorThesis/rResearch/data/synthetic_control.
    ↪ data.txt", quote="\"", comment.char="")
n <- 2

s <- sample(1:100, n)
idx <- c(s, 100+s, 200+s, 300+s, 400+s, 500+s)
sample2 <- synthetic_control.data[idx,]
df = as.data.frame(t(as.matrix(sample2)))

#calculate the change points
changeP <- e.divisive(as.matrix(df[1]), k=8, R = 400, alpha =
    ↪ 2, min.size = 3)
changeP = changeP$estimates
changeP = changeP[-c(1,length(changeP))]

changePoints = data.frame(changeP,variable=colnames(df)[1])
for(series in 2:ncol(df)){
  changeP <- e.divisive(as.matrix(df[series]), k=8, R = 400,
      ↪ alpha = 2, min.size = 3)
  changeP = changeP$estimates
  changeP = changeP[-c(1,length(changeP))]
```

```
  changePoints = rbind(changePoints, data.frame(changeP,
    ↪ variable=colnames(df)[series]))
}

# plot
df$id = 1:nrow(df)
dfMelt <- reshape2::melt(df, id.vars = "id")
p = ggplot(dfMelt,aes(x=id,y=value))+geom_line(color = "
  ↪ steelblue")+ facet_grid(variable ~ ., scales = 'free_y')
p + geom_vline(aes(xintercept=changeP), data=changePoints,
  ↪ linetype='dashed', colour='darkgreen')
```

Figure 4.12 the result of change point detection using time-series clustering of the *ecp* package.

Sometimes it is important to get an overview about the data very quickly. Tools like Tableau allow for this. Tableau supports a decent R integration. Integration of R with tools like Tableau are possible to utilize R for change point detection. Such a setup allows for easy change point detection even if the end-user is not sufficiently fluent with R.

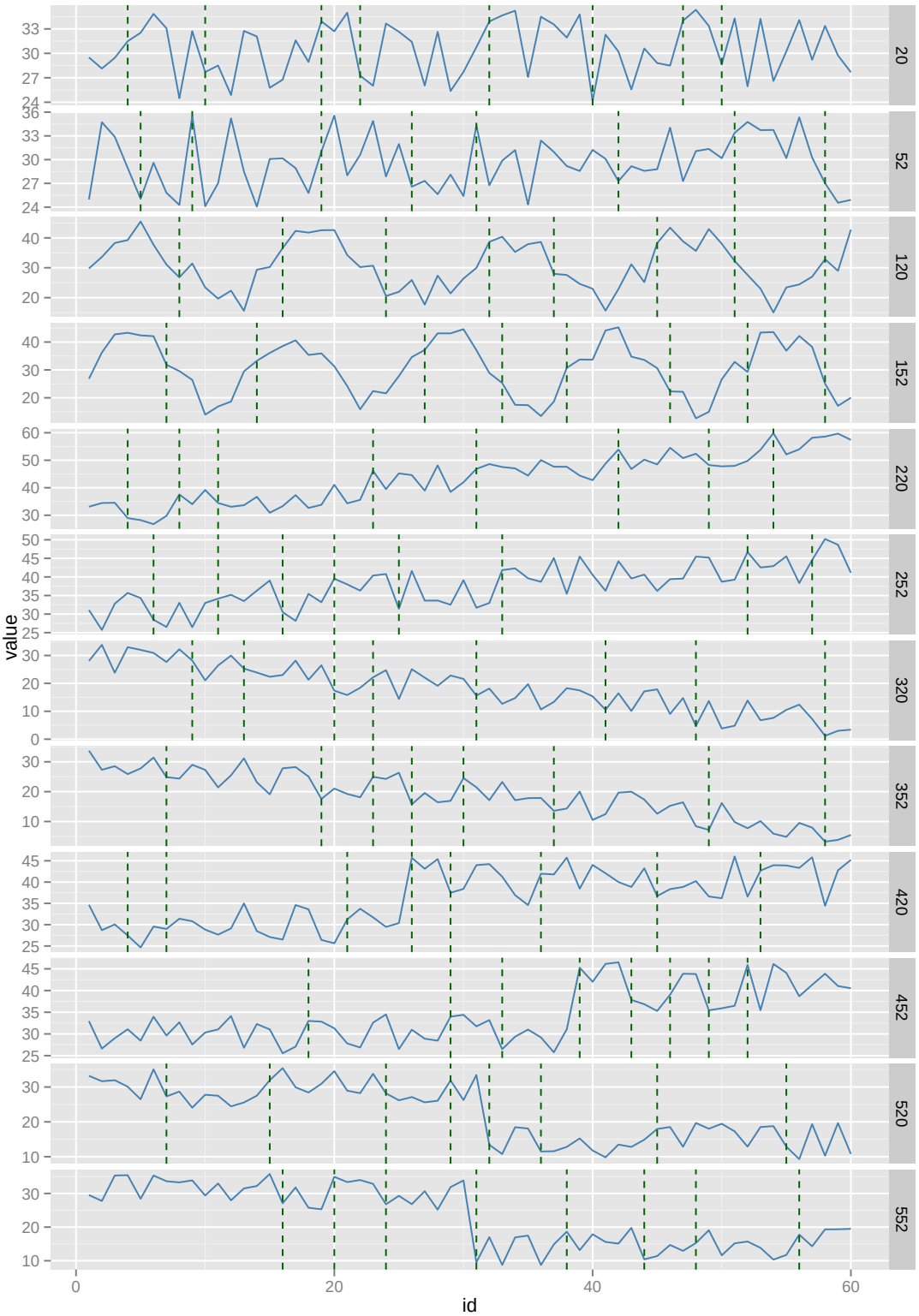

Figure 4.12: Compute change points for each of the sampled synthetic control charts.

CHAPTER 5

Summary and interesting problems

A broad variety of researchers has been interested in time series clustering. As the data are frequently high dimensional and may contain outliers, careful examination of the algorithm is important. We have shown that clustering is the basis of time-series data mining and has a lot of important applications. The need for efficient clustering of time-series will become more and more important as even more data is stored & thus analyzed in a time context. Even though some approaches for clustering time-series seem to be fairly complicated, we hope that we could show that there are some very hands-on and simple tools to solve them.

The following interesting problems for further research have been identified:

- Streaming time series and real-time analysis: these problems are especially interesting for topics like fraud analysis as "we witnessed an explosion of streaming technologies" (**Esling2012**) in recent years.
- Distributed time series clustering / mining are important for large organizations as the needs for collecting, storing and analyzing more data becomes apparent (big-data). As the amount of data grows, a monolithic system cannot deliver the needed performance. Therefore, a distributed system using distributed algorithms is needed as most algorithms today assume a centralized data set.
- An interesting approach for dealing with distributed data is shown by **Silva** As they note it is crucial to preserve privacy of sensitive data especially if the sources of data come from different organizations. Until now there has only been scarce research on privacy preserving time series mining.
- Location based data adds a whole new dimension to time series. There has been some research on spatio-temporal data in geology. However, only few tried to apply the spatial aspect of data to other fields of research.

- Understanding the semantics of a complex system means understanding its generating structure. First research has been conducted by **Wang2011** This is especially interesting as opposed to only finding isolated historic patterns. The focus is on explaining relationships between these patterns and understanding semantics.

- The concept of Hierarchical Time Series was proposed in **Hyndman2011** This can be really interesting for forecasting. There are already two R-packages: *hts* and *gtop*. The idea is that time-series may contain a hierarchical structure. Think about a bike shop. Several different types of bikes are sold. Each type contains further sub-categories. Observations of the bottom level are aggregated as observations of the level above.

- Selling analytics means converting it into a product (**research2business**).This means transforming some hacked together R-scripts into an automated computation system, as R's scripting environment is not optimally suited for an automated high performance environment. There are additional challenges, as selling a product means increasing usability. One of the major problems of time-series mining is the large number of parameters induced by the method. So the user is forced to perform a lot of fine-tuning. Automating the decision of finding the best model maybe via a combination of models like **superLearner** proposed could be the topic of further research.

CHAPTER 6

Appendix

6.1 Figures

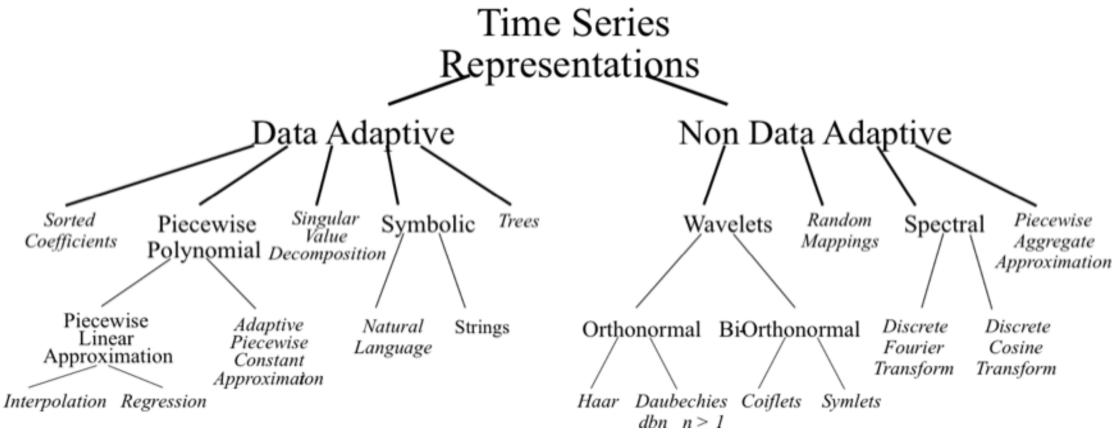

Figure 6.1: A hierarchy of all the various time series representations in the literature.. (**Debarr**)

6.2 Interesting R resources regarding clustering

- tsclust
- tsdist
- dse
- hts

- gtop
- pdc
- depmixS4
- cluster
- http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html
- https://www.researchgate.net/post/How_to_model_time-series_with_multiple_seasonalities#551109e6d767a6a1168b463c
- simulation of time-series http://t.co/1tW4G75hSs

6.3 Acquiring test data for time-series clustering

Model based time-series like ARIMA or GARCH can be easily simulated from different tools like R, matlab or mathematica. http://reference.wolfram.com/language/example/SimulateTimeSeriesData.html

6.4 Datasets

Here are some datasets for comparing time-series algorithms.

- http://www.cs.ucr.edu/~eamonn/time_series_data/
- http://cran.r-project.org/web/views/SpatioTemporal.html
- http://kdd.ics.uci.edu/databases/synthetic_control/synthetic_control.html

Generating sample data is possible. In case the problem is model based. The code snipped shows a simple ARIMA simulation.

```
ts.sim <- arima.sim(list(order = c(2,1,1), ar = 0.8), n
    ↪ = 256)
ts.plot(ts.sim)
```

An example which is a bit more complicated including seasonality can be found http://stats.stackexchange.com/questions/125946 here. ETS based:

```
fit <- ets(USAccDeaths)
plot(USAccDeaths,xlim=c(1973,1982))
lines(simulate(fit, 36),col="red")
```

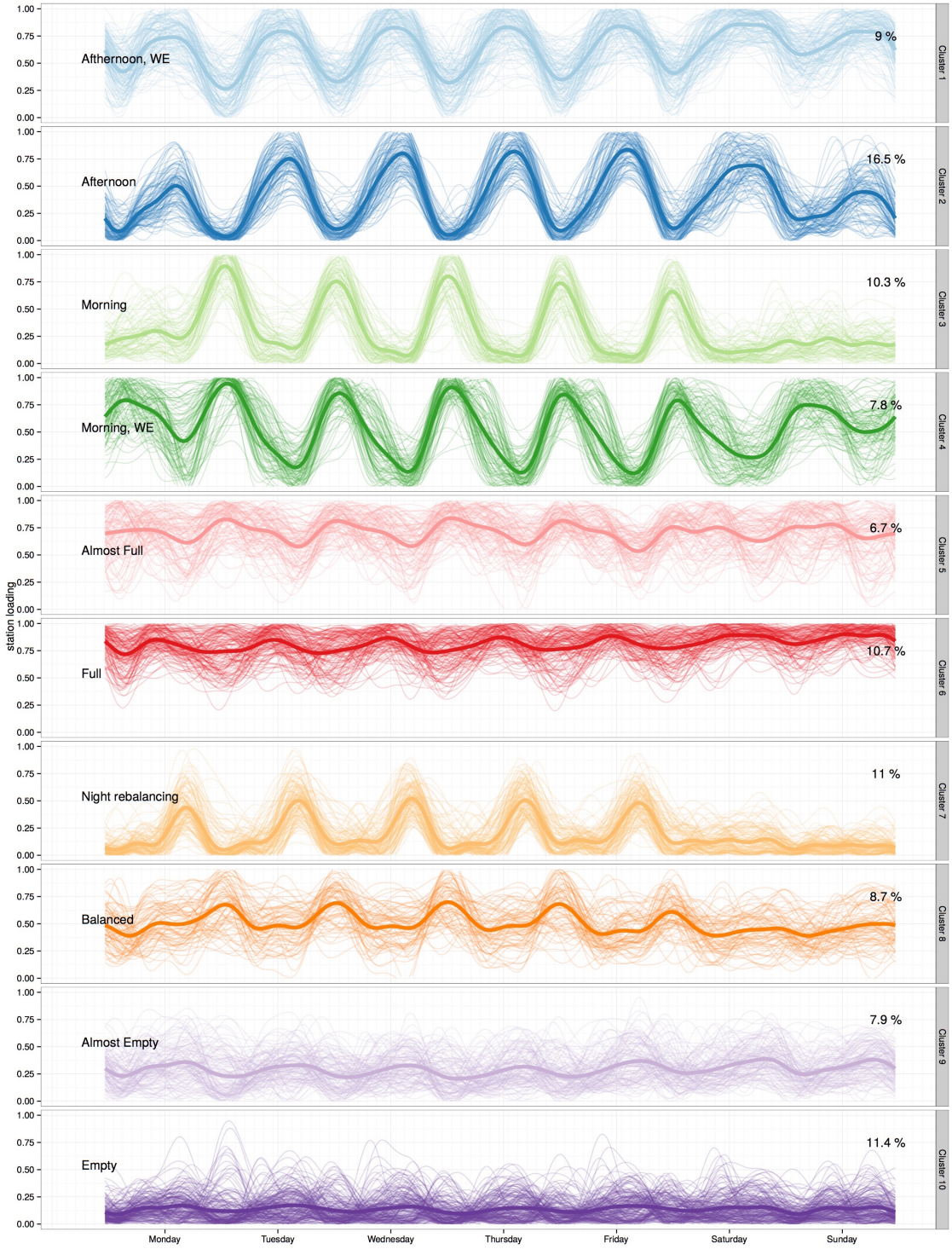

Figure 6.2: Cluster mean profiles (**bikeSharing**).