Cluster Analysis for Panel Data of Import and Export in EU Countries

Karel Horák

May 2013

Radomír Černoch, MSc.

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Acknowledgement

I would like to thank my supervisor Radomír Černoch, MSc. for guiding me through the problematics as well as for the time he spent with me consulting the matter.

I would like to express my thanks to doc. Ing. Vladimír Benáček CSc. (Institute of Economic Studies at Charles University in Prague) for providing me the dataset that was used as the main object of study in this work.

Declaration

I declare that I worked out the presented thesis independently and I quoted all used sources of information in accord with Methodical instructions about ethical principles for writing academic thesis.

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 23. května 2013

Podpis autora práce
Czech Technical University in Prague  
Faculty of Electrical Engineering  
Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Karel Horák
Study programme: Open Informatics
Specialisation: Computer and Information Science
Title of Bachelor Project: Cluster Analysis for Panel Data of Import and Export in EU Countries

Guidelines:

1. Find the relevant literature about clustering of time series.
2. Propose a data mining work-flow to analyse panel data of import and export in European countries.
3. Implement the chosen method and evaluate results.

Bibliography/Sources:

Bachelor Project Supervisor: Radomír Černoch, MSc.

Valid until: the end of the winter semester of academic year 2013/2014

[Signatures]

prof. Ing. Vladimír Mařík, DrSc.  
Head of Department

prof. Ing. Pavel Ripka, CSc.  
Dean

Prague, January 10, 2013
ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Karel Horák

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Shluková analýza časových řad pro panelová data importu a exportu zemí EU

Pokyny pro vypracování:

1. Vyhledejte aktuální literaturu k problému shlukové analýzy časových řad.
3. Zvolenou metodu implementujte a zhodnotěte její výsledky.

Seznam odborné literatury:

Vedoucí bakalářské práce: Radomír Černoch, MSc.

Platnost zadání: do konce zimního semestru 2013/2014

V Praze dne 10. 1. 2013

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan
Abstract

Velké množství informací o světě je popsáno prostřednictvím časových řad. Práce s časovými řadami nás však může postavit před nečekané problémy — pro některé z nich navrhneme v této práci možné řešení. Značná část práce je věnována problému časových posunů.

Hlavním úkolem této práce je najít shluky světových ekonomik na základě jejich obchodního styku se Spolkovou Republikou Německo. Časové řady, které zkoumáme, popisují objem německého exportu a importu do jednotlivých zemí. Na základě podobnosti těchto časových řad budeme usuzovat o podobnosti ekonomik dotyčných zemí.

Pro řešení tohoto problému navrhneme několik algoritmů. Při jejich výběru bylo dbáno na jejich jednoduchost — to umožní ekonomům najít interpretaci výsledků bez toho, aby bylo třeba pochopit komplikované pozadí algoritmu.

Klíčová slova

časové řady, shluková analýza, posuny v časových řadách, data mining, filtrování posunů, k-means, k-medoids, Fuzzy c-means, panelová data
Abstract

A lot of knowledge about the real world could be discovered from the time series. This work describes some of the problems one may encounter while dealing with time series and some solutions are proposed. The problem of time series lag is analyzed in detail.

The goal of this thesis is to find clusters of economics with respect to their trade relationship with Germany. The time series in the dataset describe the German export and import to and from individual countries on a yearly basis. The similarity in shape of these time series is used to decide about the similarity of the economics described by them.

Multiple algorithms are suggested to deal with this task — the chosen algorithms tend to be simple and straightforward which allows economists to find a reasonable interpretation of the results without the need to understand any underlying magic.

Keywords
time series, time series lags, data mining, cluster analysis, lag postprocessing, k-means, k-medoids, Fuzzy c-means, panel data
## Contents

1. Mining the time series .......................... 1  
   1.1. Dataset ...................................... 2

2. Time series representation .................... 3  
   2.1. Frequency analysis ........................... 3  
   2.2. Model parameters ............................ 3  
   2.3. Raw data representations .................... 3

3. Compressing time series ....................... 4  
   3.1. Piecewise aggregate approximation ............ 4  
   3.2. Symbolic aggregate approximation ............. 5  
   3.3. Lower-bounding property ....................... 5  
   3.4. SAX performance ............................ 6

4. Time series lags ................................ 7  
   4.1. Correlation .................................. 8  
   4.2. Relaxed Euclidean distance .................... 8  
   4.3. Dynamic time warping .......................... 9

5. Postprocessing lags .............................. 11  
   5.1. Graph interpretation of the problem ........... 11  
   5.2. Optimization criteria .......................... 12  
   5.3. Complexity of the restrictions ................. 12  
   5.4. Solution .................................... 14  
   5.5. Problems of the method ....................... 16  
   5.6. Summary .................................... 16

6. Clustering ...................................... 17  
   6.1. k-means ..................................... 17  
   6.2. k-medoids .................................... 17  
   6.3. Fuzzy c-means ................................ 18  
   6.4. Spectral clustering ............................ 18

7. Experimental results ........................... 20  
   7.1. Compatibility chart ............................ 20  
   7.2. Choosing the right number of clusters .......... 20  
   7.3. k-means clustering ........................... 21  
   7.4. Clustering with lag ............................ 22  
   7.5. Impact of SAX ................................ 24  
   7.6. Fuzzy clustering ................................ 24  
   7.7. Spectral clustering ............................ 25

8. Conclusion ..................................... 27

Appendices

A. Contents of the CD ....................... 28  
   A.1. Contents ................................. 28
List of Figures

1. Piecewise Aggregate Approximation in action ........................................... 5
2. Event model illustration .............................................................................. 7
3. It is necessary to normalize even the overlapping subsequences ............... 9
4. Warping path illustration .......................................................................... 10
5. Aligning time series on a single time axis ............................................... 11
6. Graph interpretation example ................................................................... 12
7. Path vector illustration .............................................................................. 13
8. General solution intuition .......................................................................... 15
9. Solution of the example from figure 6 ....................................................... 15
10. Average distance per lag observed ........................................................... 21
11. Elbow graph for k-Means clustering ......................................................... 21
12. k-means clustering for $k = 5$................................................................... 22
13. k-means clustering for $k = 14$ ................................................................. 22
14. Elbow graph for relaxed Euclidean distance k-medoids clustering ......... 22
15. k-medoids clustering using Euclidean distance ($k = 4$) .......................... 23
16. k-medoids clustering using relaxed Euclidean distance ($k = 4$) .......... 23
17. k-medoids clustering using Derivative Dynamic Time Warping ($k = 4$) . 23
18. Elbow graph k-medoids clustering using SAX distance (12 symbols) .... 24
19. k-medoids clustering using SAX distance ($k = 4, 12$ symbols) .......... 24
20. Fuzzy c-means clustering ($k = 5, m = 1.625$) ....................................... 25
21. Spectral clustering (5 clusters, mutual 70-NN similarity graph) ............. 26
List of Tables

1. Part of the dataset .................................................. 2
2. Method compatibility chart ......................................... 20
3. k-means clustering for $k = 5$ (see figure 12) .................. 31
4. k-means clustering for $k = 14$ (see figure 13) .............. 32
5. k-medoids clustering using Euclidean distance for $k = 4$ (see figure 15) .................................................. 33
6. k-medoids clustering using relaxed Euclidean distance for $k = 4$ (see figure 16) .................................................. 33
7. k-medoids clustering using Derivative DTW with $k = 4$ (see figure 17) .................................................. 34
8. k-medoids clustering using SAX distance ($k = 4$, 12 symbols — see figure 19) .................................................. 34
9. Spectral clustering (5 clusters — see figure 21) .............. 35
1. Mining the time series

A lot of real world data is in the form of sequences of consecutive measurements — from the daily exchange rate, the progress of temperature up to light curves of the stars. Even the genetic code does have a sequential structure. All in all it makes the study of time series particularly interesting.\[1\]

One may immediately think that it would be very helpful to be able to predict future values of a time series knowing only the values seen so far. Predicting future values is actually fairly challenging as it is needed to take all the underlying processes into account (which is usually impossible). We are then forced to be satisfied with an approximate model of a time series — which might work quite well in some situations but may be failing in other cases.

If we manage to discover that there is an important cycle lasting 5 years in the time series, we can expect that the same trend will not be kept throughout whole this period. We do not get the information how exactly this series will look like in the future, but still this knowledge brings us a useful information that allows us to adjust our decisions — eg. one will not buy stocks when he gets to know that their value was rising for several years and this season is almost over.

In the pattern recognition field, it might be often enough to classify an unseen time series into one of the previously chosen classes. In case we have a sequence describing the past state of some system, we may attempt to decide whether the system is in the safe state or not — and take appropriate measures.

However the situation changes significantly if we are not dealing with a single sequence and we are trying to discover facts about the whole datasets of time series. A possible option how to deal with a dataset of time series is to cluster it based on the similarity and dissimilarity of the time series contained in it — which will be investigated in this work.

The study of time series brings several challenges on scene that are not usually seen in conventional data mining. It is necessary to take appropriate steps in order to make the data mining process flawless.

Many interesting time series are having a large number of samples. While building a dataset out of these sequences, it may quickly become fairly difficult to store the sequences and the operations with such a dataset may become slow and memory intensive. Moreover the conventional algorithms may be highly inefficient when dealing with such a highly dimensional data. It is therefore often necessary to find an alternate way how to represent the original time series so that these issues are diminished or eliminated completely.

Another question arises as it is often necessary to find out a reasonable distance between two time series. It turns out that it is not actually good to treat sequences as aligned vectors — and defining the distance of sequences as the Euclidean metric of two such vectors. Looking at the graph of two time series where one of them is a slightly delayed version of the second one, human will immediately state that these two sequences are similar. However looking only at the Euclidean distance of these two time series vectors, everything could look differently.
1.1. Dataset

These two problems will be addressed in next few chapters. Several possible representations of time series will be introduced (especially the SAX method allowing us to convert time series into symbolic representation) and we will discuss the problem of lag — how to detect it, how to use it to build better distance functions and we will briefly touch the novel method of postprocessing these lag measurements.

In order to make the further text and the theory cleaner, let us introduce a few assumptions:

- All time series in the dataset are describing the same period of time and they were sampled using the same sampling rate (therefore also the number of samples $L$ every sequence has is the same)
- No time series is constant (i.e. the variance is defined for every sequence in the dataset)
- All time series are normalized

The first assumption not only simplifies the text, but it is also very reasonable as it does not make much of sense to deal with irrelevant sequences. If we were given series that does not fulfil this assumption, we should try to unify them — and if this is not possible, the problem might well be impossible to resolve.

1.1. Dataset

The dataset contains time series describing German trade with world countries between years 1995 and 2009. Apart from the time series of German import and export, it contains many time series describing the economics of German trade partners — both on measurable basis (e.g. GDP) and soft basis (business freedom, education etc.).

We will focus only on those time series that are related to German import and export. Let us take a look at a small subset of this dataset:

<table>
<thead>
<tr>
<th>Country</th>
<th>Year</th>
<th>Xijt</th>
<th>Mij</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD ANDORRA</td>
<td>1995</td>
<td>33.334</td>
<td>1</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>1996</td>
<td>31.818</td>
<td>0</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>1997</td>
<td>34.948</td>
<td>0</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>1998</td>
<td>30.426</td>
<td>0</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>1999</td>
<td>27.530</td>
<td>1</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>2000</td>
<td>29.635</td>
<td>1</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>2001</td>
<td>36.689</td>
<td>4</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>2002</td>
<td>36.663</td>
<td>3</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>2003</td>
<td>29.526</td>
<td>2</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>2004</td>
<td>33.100</td>
<td>3</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>2005</td>
<td>36.863</td>
<td>10</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>2006</td>
<td>36.867</td>
<td>10</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>2007</td>
<td>31.720</td>
<td>9</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>2008</td>
<td>38.507</td>
<td>15</td>
</tr>
<tr>
<td>AD ANDORRA</td>
<td>2009</td>
<td>34.308</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1. Part of the dataset

The Xijt time series represents the amount of German export in the particular country (expressed in its nominal value in Euros) — this time it is Andorra. The Mij time series does the same for the amount of goods imported in Germany.
2. Time series representation

Choosing the best representation of the original object is a challenging problem in all data mining tasks. While we often deal with objects we know quite well (and we can therefore figure out the right set of features for extraction), the time series are much more inscrutable. There are many features that cry out to be used but it is uneasy to pick the right ones.

2.1. Frequency analysis

One of the common methods to construct a feature vector of a sequence is the usage of frequency analysis. One possible approach makes use of Fourier transform that allows us to discover important frequencies in the time series — part of this information is then used to construct the new representation of the time series.

2.2. Model parameters

Creating a model of a sequence is another option how to extract a set of coefficients describing a time series in an automated way. Hidden Markov Models (HMMs) are commonly used to perform this task for stationary series. For the time series where the usage of HMMs can not be justified, the ARMA like models (autoregressive models with moving average) might be an option.

2.3. Raw data representations

Both of the representations mentioned above are slightly tricky. They are using sophisticated techniques — but careless usage of these methods may hide important facts about the time series.

Therefore it is valid to use the original sequences as they are and figure out a representation that is close to the original one, but the size issue gets insignificant. The next chapter will present the way how to reduce the number of samples by using the Piecewise Aggregate Approximation (PAA) and furthermore how to reduce the amount of space required to store a single sample by using the Symbolic aggregate approximation (SAX).
3. Compressing time series

A time series could be compressed in two ways. We can aim to reduce the number of samples the sequence has — by averaging segments of the time series by a single number which is exactly what Piecewise aggregate approximation does. The second way how to compress the time series relies on substituting the real numbers by a finite set of symbols we can store in an efficient way.

3.1. Piecewise aggregate approximation

Given a time series \(c_1, \ldots, c_n\) of length \(n\), the Piecewise aggregate approximation (PAA)\(^3\) allows us to build an approximate series \(\bar{c}_1, \ldots, \bar{c}_w\) of length \(w\), where \(w < n\). This method could be described by the following formula:

\[
\bar{c}_i = \frac{w}{n} \sum_{j=\frac{i-1}{w+1}}^{\frac{i}{w}} c_j
\]

This ugly looking formula has actually a very simple and straightforward meaning. It divides the original sequence into \(w\) segments of the same length and computes the average of each segment. These averages are then used as values of the new sequence.

Such an approach would work only if \(w\) divides \(n\). For a real application of such a method, we definitely need to be able to work out a solution for the cases when \(w\) does not divide \(n\).

Let us convert the original time series into a continuous signal. For the simplicity’s sake, we will assume that this signal will be formed as a combination of box functions. Once we get a new value, this value will be kept until it gets replaced by another one.

\[
f(x) = \begin{cases} a_i & \text{if } i - 1 \leq x < i \\ 0 & \text{otherwise} \end{cases}
\]

We can then rewrite the original formula with an integral:

\[
\bar{c}_i = \frac{w}{n} \int_{\frac{i-1}{w}}^{\frac{i}{w}} f(x)dx
\]

It is once again nothing more than averaging segments of the original time series and using this averages as a new (compressed) time series. The only difference here is that not every sample contributes to the average with the same weight. The samples on the edge of each interval split their weights according to how much this sample is included in each interval.

The important note is that the resulting PAA approximate need not be normalized even though the original time series was. One should not therefore forget about normalizing this approximate.
3.2. Symbolic aggregate approximation

The Symbolic aggregate approximation (SAX)[3] goes even further. It takes the Piecewise aggregate approximate and converts each element into a symbol. A symbol is an element from a previously chosen finite domain. The size of this finite domain $a$ (the number of symbols) is one of the parameters of the SAX method.

The key idea behind the SAX is to find breakpoints $\beta_0, \ldots, \beta_a$ where $\beta_0 = -\infty$ and $\beta_a = \infty$, such that probability $P[\beta_{i-1} \leq X < \beta_i]$ is the same for all $i \in 1, \ldots, a$ — and this probability is therefore equal to $1/a$. We can then introduce a mapping $f(x) = i$ if $\beta_{i-1} \leq x < \beta_i$. The new sequence $\hat{a}_1, \ldots, \hat{a}_w$ will be an approximation of $a_i$ using the mapping $f$.

$$\hat{a}_i = f(\pi_i)$$

The entropy of the symbol distribution is then maximal as the distribution is uniform which allows us to store the symbolic samples efficiently using $\log_2 a$ bits.

In the original work the authors assume that the normalized time series has approximately normal distribution (with zero mean and unitary variance). This allows us to construct the breakpoints easily using the quantiles of the Gaussian.

3.3. Lower-bounding property

Both of these representations have a favourable property — we can define a distance on them such that this distance lower bounds the Euclidean distance defined on the original sequence. This property is actually very important for the latter use — a lot of algorithms are working well if the distance is optimistic.

It is fairly easy to define such a distance for the PAA — we just need to scale the Euclidean distance of two PAA sequences[3]:

$$\sqrt{\frac{L}{w} \sum_{i=1}^w (\pi_i - \bar{b}_i)^2} \geq \sqrt{\frac{L}{w} \sum_{i=1}^w (\pi_i - \bar{b}_i)^2} = d_{PAA}(a, b)$$

To define such a distance for the SAX sequence, we should first investigate the distance of a pair of symbols. We can define this distance of two symbols $a$ and $b$ as the
distance of inner bounds of corresponding intervals[^3]:

\[
d(a, b) = \begin{cases} 
0 & \text{if } |a - b| \leq 1 \\
\beta_{\max(a,b)} - \beta_{\min(a,b)} & \text{otherwise}
\end{cases}
\]

As this distance \(d\) on the single symbols level lower bounds the difference between original real values from the PAA, we can simply build a time series distance that lower bounds the PAA distance (and therefore the Euclidean distance as well):

\[
\sqrt{\frac{L}{w} \sum_{i=1}^{w} (\hat{a}_i - \hat{b}_i)^2} \geq \sqrt{\frac{L}{w} \sum_{i=1}^{w} d(\hat{a}_i, \hat{b}_i)^2} = d_{SAX}(a, b)
\]

### 3.4. SAX performance

Experiments have shown that the optimism about the true distance may in some situations outperform the Euclidean distance. This slightly surprising fact has turned our attention to this method of representing the time series.

The possible explanation of this behaviour is the fact that SAX actually works a bit as a noise reduction filter. By reducing the amount of information contained in the data we expect the noise to be removed. Many algorithms are sensitive to noise and this way of preprocessing may help them achieve better results.

However it is important to say that SAX is not faultless. If the approximation is not done carefully, one may end up with a very bad approximation and the performance may deteriorate.
4. Time series lags

It was already mentioned that it does not make much of sense to look on two sequences as on two vectors of corresponding coefficients. To understand the problem of lags we can think about an event model that is underlying to these time series.

Assume that there is a series of events happening in our hypothetical world. Different time series may react to these events with different magnitude of reaction as well as with different reaction times. It is a question whether the difference in the reaction time should actually matter in the case of time series similarity. We can state that the time series are similar, if they react to the same set of events in a similar way — and we do not care about the time at which this happen.

One may become quite confused by this event based modeling. Let us introduce a simple world from a fairy tale. Two companies are customers of a single factory. The value of their stocks increases by $25\%$ on a normal day. However if they get to know that the factory is closed due to a strike, the value of their stocks is cropped to half of the original value instead. There are no modern technologies and it takes a while until the manager gets the information whether the employees of the factory are working. We can illustrate the impact of this simple event model on the stock values by following figure:

![Figure 2. Event model illustration](image)

The construction of such a model that would allow us to perform such an analysis is an extremely uneasy task — and it is very probable that we will not be able to construct such a model properly. We should therefore pay attention to data-based models and lag estimation methods.
4.1. Correlation

Correlation\[^4\] is a statistical method for determining the degree of dependence between two sets of pairwise measurements. The time series are containing some kind of measurements — but the involvement of the time introduces the time lag and one sample may correspond to a sample seen later on in the second sequence.

It is therefore a vital idea to deal with different alignments of two time series in question and compute the correlation of the overlapping part\[^5\]:

\[
c(k) = \sum_{1 \leq i \leq L} a_i b_{i+k}
\]

The values of \(k\) that makes \(c(k)\) reach one of its peaks are a good candidates for being the optimal lag between two time series we study.

This computation runs in quadratic time. The tempting property of this method is the fact that it can be speeded up by using Fast Fourier Transform which reduces the complexity to \(n \log n\).

4.2. Relaxed Euclidean distance

Probably the most straightforward way of getting the information about the lag is to relax the Euclidean distance. The goal is to find such an alignment of the time series that minimizes the Euclidean distance of the overlapping segments. In order to make this method reasonable, we must take few problematic parts into account:

- When we are dealing with the overlapping segments, we have to normalize these segments independently.
- It does not make any sense to allow overlapping segments of length 1. Actually it would not be even possible to normalize such segments.
- Overlapping segments are having different lengths. In order to be able to compare the distances of segments of various lengths, we have to rescale them — such that the number of elements in the summation tends to be the same.

The first problem could be demonstrated by the figure 3. Major part of the first time series has low values after normalization (due to the very first sample). The second time series does not have any such sample and its values are greater in general. As we pick such a segment that the first sample of the first series is not contained in it, the scale of this sequence needs to be adjusted once again so that it could be compared with the second time series. Actually these two time series are identical if we get rid of that outlying sample after proper normalization.

Considering these things we could build a formula similar to the one from the previous section.

\[
d(k) = \begin{cases} 
\sqrt{\frac{L}{L-k}} d_E(a_{1}^{L-k}, b_{1}^{L}) & \text{if } k \geq 0 \\
\sqrt{\frac{L}{L+k}} d_E(a_{1}^{k}, b_{1}^{L}) & \text{if } k < 0 
\end{cases}
\]

where \(b_a\) is a normalized subsequence of the original time series from index \(a\) to \(b\) and \(d_E\) is Euclidean distance of two sequences without taking the lag into account. Once again we can find optimal \(k\) — in this case that one that minimizes \(d(k)\).

The need of normalization introduces further performance overhead that makes this method suitable for short time series only. As our dataset contains sequences of length 15, this method is fully viable.
4.3. Dynamic time warping

The Dynamic time warping (DTW)\cite{6} is somewhat different from the methods introduced above. It does not give us a single number denoting the lag between two time series, however it gives us more flexibility in return.

This algorithm actually steps closer to the event model introduced in the beginning of this chapter. We have said that time series aren’t forced to react on all events with the same reaction times. This may cause a time series to be locally accelerating and decelerating in comparison to the second time series.

To understand what problem does this algorithm solve, we need to investigate the term of warping path. In the beginning, both time series are at their very first sample. In each step we can decide between three alternatives — advancing both time series or advancing just one of them. We expect to end up pointing to the last sample in both series. The warping path describes these decisions.

**Definition 1** (Warping path). Warping path of time series \(\{a_i\}_{i=1}^m\) and \(\{b_i\}_{i=1}^n\) is a sequence of tuples \(\{(A_i, B_i)\}_{i=1}^K\) (\(A_i, B_i \in \mathbb{N}\)) such that:

- \(A_1 = B_1 = 1\) (i.e. both time series are at their beginning)
- \(A_K = m, B_K = n\) (both time series are processed at their entirety once we are ready)
- \(A_{i+1} = A_i + 1\) or \(B_{i+1} = B_i + 1\) for any index \(i\) (both could be true at the same time)
- \(A_{i+1} \geq A_i\) and \(B_{i+1} \geq B_i\) (i.e. it is forbidden to go back in time)
- \(A_{i+1} - A_i \leq 1\) and \(B_{i+1} - B_i \leq 1\) (i.e. it is forbidden to skip any sample)

The goal of the Dynamic time warping algorithm is to find out warping path that minimizes following criterion:

\[
\min_{w=\{(A_i, B_i)\}_{i=1}^K} \sqrt{\sum_{i=1}^K (a_{A_i} - b_{B_i})^2} / K
\]

The concept of warping path optimization is well illustrated in the following figure. Each of the orange lines represents one element of the warping path — these elements are chosen in such way that the endpoints are similar.

This problem might seem challenging, but in fact it can be solved in an efficient way by using dynamic programming — once we construct a \(m \times n\) matrix where the elements represents the squared distance between respective samples, the problem reduces to
4.3. Dynamic time warping

Warping path elements

Figure 4. Warping path illustration

finding a shortest path from the top left corner to the bottom right one. We solve this task by filling matrix \( W \in \mathbb{R}^{m \times n} \) using the following formula:\(^1\)

\[
\begin{align*}
w_{i,j} &= \min \{w_{i-1,j}, w_{i-1,j-1}, w_{i,j-1}\} + (a_i - b_j)^2
\end{align*}
\]

The traditional version of Dynamic time warping suffers from one issue. Once the time series do not differ only in the \( x \)-axis and there are also differences in the \( y \)-axis (e.g. one time series decreases more than the other one), this version of DTW algorithm tends to explain this behaviour in terms of singularities. One of the time series advances whereas the second one stalls — and the roles need to be switched afterwards. Such a warping path does not seem to be a good candidate. One of the methods that has been suggested is the derivative variant of this algorithm.\(^6\)

The distance in this case is not computed as the squared difference between respective samples. Instead we estimate the derivatives at the respective points in time and use squared difference of these estimates. This method diminishes the impact of slight \( y \)-axis deteriorations as the first derivative need not differ as much as the original values.

The authors of this method suggest to use three datapoints to estimate the derivative — and compute it by following formula:

\[
d_q(i) = \frac{(q_i - q_{i-1}) + (q_{i+1} - q_i)}{2}/2
\]

This does not make sense for the first and the last elements of the sequence — the neighboring value is used instead.

\(^1\)The \( K \) in the denominator would complicate the optimization process and is therefore ignored.
5. Postprocessing lags

All of the techniques mentioned previously suffer from one problem. They are exploring the lags only on the level of sequence pairs — and the information contained in the whole dataset remains hidden to them. We propose a novel method that attempts to fix this potential oversight.

Given a set of pairwise measurements, this method aims to make these measurements more accurate. It applies a transitive principle that requires that the lag of two time series is equal to the sum of lags of time series occurring in between.

Assume that we can find an ordering of time series contained in the dataset such that \( s_i \preceq_{\text{LAG}} s_j \) if time series \( s_i \) precedes \( s_j \) (or there is zero lag between these two sequences). We will denote the computed lag of these two sequences as \( \delta_{i,j} \). In the process of computation we have likely obtained also the distance of these two time series assuming the lag of \( \delta_{i,j} \) — we will denote this distance as \( \epsilon_{i,j} \).

It does make sense to spread all the time series along single time axis such that the overlaps are similar. We have quite a lot of information about the way how we have to work out this problem — we were given the pairwise measurements. However on the single time axis, we expect that once we know \( \delta_{i,j} \) and \( \delta_{j,k} \) for time series \( s_i \preceq s_j \preceq s_k \), the equation \( \delta_{i,k} = \delta_{i,j} + \delta_{j,k} \) should hold. But according to our measurements this constraint need not be satisfied.

\[
\begin{align*}
\delta_{s_1,s_2} & \quad \delta_{s_2,s_3} \\
\text{time} 
\end{align*}
\]

Figure 5. Aligning time series on a single time axis

The key idea behind this method relies on finding such \( \delta s \) such that this property holds. For better insight in the way how to solve this problem, let us investigate it from the graph point of view.

5.1. Graph interpretation of the problem

Let us construct a directed graph \( G \) as follows:

- Vertices \( V \) represents the time series in the dataset \( (s_1, \ldots, s_K) \).
- A directed edge \( e \in E \) connects two time series \( s_i \) and \( s_j \) if observed lag \( \delta_{i,j} \) between these two sequences is positive. (Note: If lag \( \delta_{i,j} \) is negative, lag \( \delta_{j,i} \) will be positive)
5.2. Optimization criteria

The graph currently contains only edges corresponding to non-zero lags. The zero lag edges were not inserted because we do not know their orientation. It could be easily fixed if we assume that the current version of the graph is acyclic (i.e. there is no cyclic precedence).

Let us build the topological ordering of the graph. The zero lag edges will be oriented in such way, that no cycle will be formed (that means that the edge will be pointing always to the time series that occurs later on in the ordering).

The graph $G$ we have just built is a fully connected directed acyclic graph. Each edge in this graph contains information coming from the lag detection algorithm:

- $\delta(s_i, s_j) = \delta_{i,j}$ — the observed lag of time series $s_i$ and $s_j$
- $\epsilon(s_i, s_j) = \epsilon_{i,j}$ — the distance of time series $s_i$ and $s_j$ (the criteria why the lag $\delta_{i,j}$ is the best choice)

We said that we are interested only in such lags that $\delta_{i,k} = \delta_{i,j} + \delta_{j,k}$. In the graph view of the problem, this is equal to the fact that all paths between two vertices accumulate the same sum of $\delta$s — we will denote accumulated lag along path $p$ as $\delta(p)$. This property will be referred to as the transitivity property.

To get an idea about the problem, let us take a look at the following example (figure 6). The transitivity principle in this case forces $\delta(e_2) = \delta(e_1) + \delta(e_3)$ which is not true in this case and needs fixing.

![Figure 6. Graph interpretation example](image)

5.3. Complexity of the restrictions

The number of distinct paths in such a graph is growing exponentially with the number of vertices. This may make us think that this problem is actually very hard to handle.
5.3. Complexity of the restrictions

However the situation is different and the problem is actually quite easy to handle.

**Definition 2** (Primitive edge). *Given a fully connected directed acyclic graph $G$, the primitive edge $e$ is an edge connecting two adjacent vertices in the topological ordering of $G$.*

It could be easily observed that there are $n - 1$ primitive edges in a $n$-vertex graph. The total number of edges in a fully connected graph is equal to $n(n − 1)/2$.

It could be observed that we have $n − 1$ degrees of freedom how to construct a graph that holds the transitivity property — the choice of $\delta$ for the primitive edges decides the values of all other edges. It will be more easy to see as we describe the problem using the linear algebra.

- $E = e_1, e_2, \ldots, e_l$ — the set of all edges in the graph
- $\delta = [\delta(e_1), \delta(e_2), \ldots, \delta(e_l)]^T$ — vector of measured lags (lag vector)
- $l = n(n − 1)/2$ — the number of edges in a graph

The lag vector allows us to come up with more understandable way of representing the accumulated lag $\delta(p)$ along an arbitrary path $p$.

**Definition 3** (Path vector). *Given a path $p$ in a fully connected directed acyclic graph, the path vector of path $p$ is a binary vector $a_p \in \mathbb{R}^l$ such that*

$$a_{p,i} = \begin{cases} 1 & \text{if edge } e_i \text{ is contained in path } p \\ 0 & \text{otherwise} \end{cases}$$

Path vector allows us to rewrite the accumulated lag $\delta(p)$ as the scalar product of the lag vector with the path vector:

$$\delta(p) = a_p^T \delta$$

This allows us to shift from unclear constraining equations to the difference of two such terms:

$$(a_p - a_q)^T \delta = 0, \quad p, q \in P_{i,j} \text{ for any two vertices } s_i, s_j$$

Putting all such equations in a matrix allows us to rewrite it in a matrix form

$$A \delta = 0$$

Let us illustrate the path vector on the example from figure 6. Using the path vector we can compute the accumulated delay of path $p$ in a concise way.

![Figure 7. Path vector illustration](image-url)
In a fully connected directed acyclic graph of three vertices, there is the only pair of paths connecting same two vertices — $p = e_2$ and $q = e_1e_3$. Therefore there will be the only one constraining equation $(a_p - a_q)^T \delta = 0$ and the matrix $A = [-1, 1, -1]$.

**Theorem 1** (On rank of matrix $A$). Matrix $A$ has $l - n + 1$ linearly independent rows. The dimension of the solution space is therefore $n - 1$.

**Proof.** Let us divide the proof in two parts — firstly we will prove that at least $l - n + 1$ linearly independent rows exist and then we will show that no other row could be put in this linearly independent set.

Let us form equations such that only one non-primitive edge is present:

$$\delta(s_i, s_j) = \sum_{k=i+1}^{j} \delta(s_{k-1}, s_k), \ i + 2 \leq j$$

We already know that there are $l = n(n - 1)/2$ edges in the graph and $n - 1$ of them are primitive. Therefore there are $l - n + 1$ equations of such form and all of them are linearly independent (as there is always different non-primitive edge used).

No more equation could be added to this set — if we allow usage of multiple non-primitive edges in an equation, these edges could be rewritten as sums of primitive edges using equations previously introduced, e.g.:

$$\delta(s_i, s_k) = \delta(s_i, s_j) + \delta(s_j, s_k) \ (\text{where } i \leq j \leq k)$$

$$= \sum_{l=i+1}^{j} \delta(s_{l-1}, s_l) + \sum_{l=j+1}^{k} \delta(s_{l-1}, s_l)$$

$$= \sum_{l=i+1}^{k} \delta(s_{l-1}, s_l)$$

The dimension of the solution space follows from the simple observation from linear algebra that the dimension of the solution space is the difference of number of variables ($l$) and number of linearly independent rows ($l - n + 1$) — therefore it is $n - 1$. 

Back to our example — we can see that there is just one equation constraining three variables. That means we can find $n - 1 = 2$ linearly independent solution of this equation.

### 5.4. Solution

In this section we will sketch how to find the solution space for any arbitrary number of vertices and how to use it to solve the original optimization task.

We know that we have to find $n - 1$ linearly independent solutions of the constraining equations. Recall that all the equations are in form of

$$\delta(s_i, s_j) = \sum_{k=i+1}^{j} \delta(s_{k-1}, s_k)$$

Let us investigate the solutions where only one primitive edge has a non-zero lag. It can be easily observed which other $\delta$s have to be adjusted — it will be those ones
where the chosen primitive edge is included in the respective equation (i.e. in its sum). Let us look at the lag variables in form of a triangle (figure 8) that illustrates the particular solution vector.

We know how to build a basis of the null space of \( \mathbf{A} \) (null \( \mathbf{A} \)). Let us construct matrix \( \tilde{\mathbf{A}} \) such that \( \text{rng} \tilde{\mathbf{A}} = \text{null} \mathbf{A} \) (i.e. the solution vectors are columns in the matrix \( \tilde{\mathbf{A}} \)). We know that \( \delta' \in \text{rng} \tilde{\mathbf{A}} \) — this comes from the constraining equations. We can therefore express \( \delta' \) as \( \delta' = \tilde{\mathbf{A}} \lambda \) (where \( \lambda \) is a coordinate vector). This allows us to rewrite the original optimization problem as follows:

\[
\min_{\lambda \in \mathbb{R}^{n-1}} (\mathbf{A} \lambda - \delta)^T \mathbf{E}^{-1} (\mathbf{A} \lambda - \delta)
\]

where \( \mathbf{E} \) is a diagonal matrix where \( E_{i,i} = \epsilon(e_i) \).

Using the first derivative of the optimization criteria (and looking for its null point) we find following solution for \( \lambda' \) (\( \cdot^+ \) denotes pseudoinverse of a matrix):

\[
\lambda^* = (\tilde{\mathbf{A}}^T \mathbf{E}^{-1} \tilde{\mathbf{A}})^+ \tilde{\mathbf{A}}^T \mathbf{E}^{-1} \delta
\]

We may use these coordinates to find the optimal solution for \( \delta' \) as:

\[
\delta'^* = \tilde{\mathbf{A}} (\tilde{\mathbf{A}}^T \mathbf{E}^{-1} \tilde{\mathbf{A}})^+ \tilde{\mathbf{A}}^T \mathbf{E}^{-1} \delta
\]

Let us apply this formula to solve the simple example from figure 6. One can check that the transitivity property is satisfied for \( \delta' \). We can observe that \( \delta(e_2) \) corresponding to the most erroneous edge was changed the most. Intuitively the measurement corresponding to the most distant pair of sequences is likely to be the least accurate.
5.5. Problems of the method

It remains to investigate the situations when it is impossible to use this method (or it requires special care at least).

The most problematic situation (which may happen quite frequently in the practice) is the case when the resulting graph is not acyclic. It is recommended to cluster the dataset in multiple closely related groups of time series prior to using this method — actually the cycles are often formed by edges with high distance and such an edge usually gets removed once we perform the clustering.

If clustering does not help, it remains up to the user to figure out how to get rid of cycles in the graph — here are just few possibilities:

- Identify cycles and then assume that all edges forming them have zero lag.
- Turn the orientation of some edges around and let their lag be negative.

Obviously another problem comes with the $1/\epsilon(e)$ term — we need the distance to be non-zero. However this problem could be resolved by using another similarity function than $1/\epsilon(e)$.

5.6. Summary

The usage of this lag postprocessing method is not limited on the case of time series lags. It can be used in any scenario where the transitivity property of some measurement is expected. We suppose that this method can be successfully used in numerous situations across the scientific fields.
6. Clustering

Data clustering is a discipline of finding clusters within the dataset such that the elements inside one cluster share similar properties whereas the objects that lie in different clusters are significantly dissimilar.

Many algorithms have been proposed for solving this task. In this work, we are interested in simple clustering algorithms. The output of this work has to be easily interpretable — and the choice of clustering algorithm may help out a lot with interpretability (i.e. if there is no “magic” inside it, one may trust the results better).

6.1. k-means

K-means\cite{7, 8} is probably the most popular clustering algorithm at all. It uses a simple and straightforward procedure that iteratively improves the criteria value until it reaches (local) optima.

The optimization criteria for this algorithm is following:

$$\sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

Set $C_i$ contains all datapoints assigned to the $i$-th cluster. All $C_i$ are pairwise disjoint. The k-Means algorithm therefore minimizes the intracluster variance.

The optimization is performed in two steps:

1. Given cluster centroids $\mu_1, \ldots, \mu_k$, create the sets $C_1, \ldots, C_k$. The assignment cannot be any better than assigning each datapoint to the cluster whose centroid is closest to it according to the Euclidean distance.

2. Given a set of datapoints in a cluster, find an optimal centroid — the intracluster variance is minimized if we set the new cluster centroid as the mean of all datapoints in the cluster.

These steps are performed until the cluster assignment is changing. Before the first iteration, $k$ items are picked randomly from the dataset and these items are used as the initial configuration of the cluster centroids.

The result of one run of this algorithm need not be the global optima. This is usually resolved by running this algorithm multiple times and choosing the result with the lowest criteria value.

6.2. k-medoids

Sometimes it is favourable that the cluster centroid is actually one of the datapoints from the original dataset — the k-medoids\cite{8} algorithm does exactly this. In some situations it might be even impossible to find new cluster centroid from the domain of all possible feature vectors.

The only difference between the k-means and k-medoids is in the way it chooses the new cluster centroid. K-means computes the mean of the cluster — the k-medoids
algorithm will go through all the datapoints in the cluster and finds the one minimizing the intracluster variance, i.e.:

\[ \mu_i = \arg \min_{x \in C_i} \sum_{x \in C_i} \| x - x_i \|^2 \]

This algorithm will be particularly good choice for us while clustering the dataset using some distance that takes the lag into account.

### 6.3. Fuzzy c-means

Fuzzy c-means\(^9, 10\) algorithm treats the clusters in a different way. Instead of finding a way how to divide the dataset in several disjoint clusters, the goal of this method is to find out degree of membership of each datapoint into each of the clusters. The output of this algorithm is therefore not a set of disjoint sets — this algorithm outputs a matrix \(U\) containing these degrees.

The optimization procedure is similar to k-means, the algorithms alternates between the assignment and cluster relocation steps — however the way how these steps are performed is somewhat different:

1. In the assignment step, the degrees of belonging are computed using the following formula:

\[ u_{ij} = \frac{1}{\sum_{l=1}^{k} \left( \frac{\| x_i - c_j \|}{\| x_i - c_l \|} \right)^{2(m-1)}} \]

where \(i\) denotes the index of the instance, \(j\) represents the index of the cluster and \(k\) is the number of clusters.

2. In the cluster relocation step, the new cluster centroids are computed as the weighted average of the datapoints with weights of the \(m\)-th power of the related degrees computed in the previous step:

\[ c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m} \]

Parameter \(m > 1\) controls the fuzziness of the resulting clustering. Larger values of \(m\) result in more fuzzy outcome. It is important to choose the value of \(m\) properly — large \(m\) may make the algorithm find degenerate clusters.

The algorithm is initialized using a randomly initialized matrix \(U\). It is not guaranteed once again that the algorithm will find globally optimal solution. The most straightforward attempt to resolve this problem is to try multiple random initializations and pick the solution minimizing following criterion (which is once again very similar to the one known from k-means):

\[ \sum_{j=1}^{k} \sum_{i=1}^{N} u_{ij}^m \| x_i - c_j \|^2 \]

### 6.4. Spectral clustering

Spectral clustering\(^11\) is a term for a whole group of clustering algorithms. These algorithms make use of the eigenvalues and eigenvectors of a similarity matrix (its
spectra) to reduce the dimensionality of input data before some conventional clustering algorithm comes in place.

We need to study a term of similarity graph. The similarity graph is a weighted undirected graph where the edges with higher weights correspond to more similar pairs of time series. This graph could be represented as a matrix $W \in \mathbb{R}^{n \times n}$ (where $n$ is the number of time series in the dataset). The element $w_{i,j}$ corresponds to the similarity of time series $s_i$ and $s_j$. Not all time series have to be connected by an edge — in this case they are treated as totally dissimilar (i.e. the respective $w_{i,j} = 0$).

In this work we will use the mutual $k$-nearest neighbors similarity graph. An edge $(s_i, s_j)$ is present in the graph if $s_i$ is amongst $k$ nearest neighbors of $s_j$ and vice versa. The weight of an edge $w_{i,j}$ is obtained by using Gaussian similarity function.

$$w_{i,j} = \exp\left(-\frac{\|s_i - s_j\|^2}{2\sigma^2}\right)$$

Once we have the graph ready we need to compute the graph Laplacian. The algorithm used in this work makes use of its unnormalized variant that is computed as $L = D - W$. The matrix $D$ is a diagonal matrix of degrees of the vertices in the graph.

To cluster the dataset using this algorithm we need to compute first $k$ eigenvectors $v_1, \ldots, v_k$ of the unnormalized graph Laplacian $L$ (where $k$ is the number of desired clusters). Let these vectors be columns of the matrix $V \in \mathbb{R}^{n \times k}$. The rows of this matrix correspond to the new representation of our original time series — and can be clustered by using the k-means algorithm.
7. Experimental results

We have implemented all of the algorithms sketched in the previous chapter — k-means, k-medoids, Fuzzy c-means and the spectral clustering algorithms. These algorithms were supplied by following representations:

- Vector based representation (Euclidean distance)
- Vector based lag-aware representation (Relaxed Euclidean distance)
- Derivative Dynamic Time Warping
- Symbolic Aggregate Approximate

This chapter presents the results obtained while clustering the time series of German export. The clustering results for import time series could be obtained by using the supplied program.

7.1. Compatibility chart

Not all of these representations are compatible with all of the algorithms. This compatibility is best expressed by following table (✓ means a technically possible combination that would need further implementation):

<table>
<thead>
<tr>
<th></th>
<th>k-Means</th>
<th>k-Medoids</th>
<th>c-Means</th>
<th>Spectral clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean distance</td>
<td>✓✓</td>
<td>✓</td>
<td>✓</td>
<td>✓✓</td>
</tr>
<tr>
<td>Relaxed Euclidean distance</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Derivative DTW</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>SAX</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓✓ (experimental)</td>
</tr>
</tbody>
</table>

Table 2. Method compatibility chart

We will study some of these combinations and discuss the features that might make them more appealing than others.

7.2. Choosing the right number of clusters

The choice of the right number of clusters is crucial for all of the algorithms in question. It is generally an uneasy task, but the performance is highly influenced by this decision.

We propose that an experienced economist checks the result and uses his knowledge to set the right number of clusters. It need not be a good choice to pick the number of clusters according to some of the heuristics — as it need not result in a clustering the economists want to see.

To get an intuition about the number of clusters, the elbow method will be used in this work. We will plot a graph of the optimization criteria depending on the choice of the cluster count. The goal is to choose such a number that adding another cluster will not improve the result (i.e. the criteria value) significantly.
This method is pretty straightforward, but it lacks stable statistical ground. For more sophisticated method to estimate the number of clusters based on the elbow method, see the article on gap statistic [12].

7.3. k-means clustering

The k-means clustering is available only for the Euclidean metric without taking the lag into account. This might seem as a significant oversight — but assuming that the world market is globalized, this need not produce bad results.

The following graph highlights this point. The pairs of time series were grouped according to the lag measured and the mean distance in each of these groups was plotted. We can see that as some lag is introduced the average distance significantly increases. This makes us think that the pairs of sequences where the lag is important should get split in two different clusters anyway.

![Figure 10. Average distance per lag observed](image)

This method would make sense even in the situation when neglecting the lag would be a real problem. It allows us to discover clusters of countries that reacts in the same way in the same time — which could be a useful information as well.

According to the elbow graph, it seems that we should choose around 5 clusters. The criteria is decreasing significantly until 5 whereas it continues to improve slower afterwards. However the choice of 5 clusters produces one large cluster containing majority of European, American and Asian countries — which need not be the result we want. The choice of larger values makes therefore sense as well.

![Figure 11. Elbow graph for k-Means clustering](image)
7.4. Clustering with lag

Once we take the lag into account, the k-means clustering algorithm is not available for us. This incompatibility is caused by the fact that it is uneasy to find relevant cluster centroid given a set of time series. Averaging them does not make much of sense as it does not take the lag in consideration. The k-medoids algorithm does not suffer from this issue and it will be therefore used for obtaining the results in this section.

For the sake of comparison we will use the same number of clusters regardless the distance measure used. This allows us to compare the results obtained without considering lag and using both the relaxed Euclidean distance and the Derivative Dynamic Time Warping. Using varying cluster count would make this comparison much more difficult.

If the lag is an important factor in the dataset, we expect that the output we get while using a lag-aware distance function would differ significantly from the one obtained while using a simple Euclidean metric. Otherwise we expect the clustering results to be more or less the same.

We will figure out the right number of clusters for the relaxed Euclidean distance case. The elbow graph looks as follows.

![Elbow graph for relaxed Euclidean distance k-medoids clustering](image)

We can observe a similar graph to the one from the previous section. The value $k = 4$ seems to be a good candidate for the number of clusters in the dataset. Let us look at the results obtained by using different distance measures.
7.4. Clustering with lag

Figure 15. k-medoids clustering using Euclidean distance ($k = 4$)

Figure 16. k-medoids clustering using relaxed Euclidean distance ($k = 4$)

Figure 17. k-medoids clustering using Derivative Dynamic Time Warping ($k = 4$)

There is some difference between the clustering obtained by using raw Euclidean distance (figure 15) and the one we get while using the relaxed version of this distance measure (figure 16). However this difference is not that important which supports our theory about the degree of globalization of the current world. Usage of Derivative Dynamic Time Warping (figure 17) has however resulted in a significantly different outcome.
7.5. Impact of SAX

We have already discussed the possible positive impact of the SAX method on the outcome of the clustering. The dataset becomes less noisy — it is a question whether the resulting clustering will be less noisy as well.

Once again the first necessary step is to choose the right number of clusters. No surprising result comes out from the elbow graph (figure 18) and we could therefore choose to cluster into 4 clusters.

![Figure 18. Elbow graph k-medoids clustering using SAX distance (12 symbols)](image)

Looking at some geographical regions, it might seem that there is indeed less noise in the clustering. However it is questionable whether it is not an error from approximation.

![Figure 19. k-medoids clustering using SAX distance (k = 4, 12 symbols)](image)

7.6. Fuzzy clustering

We expect that the clusters in the economy need not be crisp. Some of the countries form relatively closed clusters, but some of them will definitely have trade relations with countries from multiple market clusters. This assumption takes us to the world of fuzzy clustering — we have chosen the Fuzzy c-means clustering algorithm to deal with this task.

The Fuzzy c-means algorithm needs two parameters to be tuned — the number of fuzzy clusters and the degree of fuzziness. It is not easy to choose both of these numbers correctly. It was decided that we will choose the number of clusters (we have some idea about it from the previous clustering results) and the fine tuning will be done on the fuzziness parameter $m$. The larger values of $m$ result in fuzzier results (which is what
we want). But using an extremely high value might result in degenerate clusters — and the result would be very poor. The number of clusters has been set to 5 — and number 1.625 seems like a reasonable choice for the fuzziness parameter.

The following set of images corresponds to the clusters we have found out by using this setting. The darker colors correspond to the countries with higher degree of membership into the cluster.

![Fuzzy c-means clustering](image)

**Figure 20.** Fuzzy c-means clustering ($k = 5$, $m = 1.625$)

### 7.7. Spectral clustering

The spectral clustering algorithm fails to accomplish one of our key goals — it is too complicated in order to be interpreted with ease. It is therefore used only for comparison with the results we have obtained so far.

For this reason, we have decided to use the same number of clusters we used for the k-means clustering ($k = 5$). The mutual $k$-nearest neighbors similarity graph with 70 neighbors was used.
We can observe that the clustering result is significantly different from those obtained using the simpler methods. The clustering using more complex methods (like spectral clustering) remains therefore a valid way how to proceed in the analysis of this dataset.
8. Conclusion

We have presented basic algorithms that are commonly used for cluster analysis and we have applied these algorithms on the problem of panel data of German export. The economists are supposed to find an interpretation for the cluster assignments we have found out in this work — and possibly adjust the clustering process (e.g. by choosing different number of clusters that would suit their theory better).

Our original idea was tightly connected with discovering lags of the time series — however as we proceeded with investigation of the dataset it turned out that the lag issue is not that significant. Despite this fact we have presented a novel method that aims to filter the lag measurements and attempts to make them more accurate. As we have turned our attention from the time series lags, this method has not been tested in practice yet.

Apart from the clustering results we have obtained in this work, we suggest to pay attention especially to the lag postprocessing method outlined in the chapter 5. This method is supposed to be useful in wide range of scenarios — but the real test case is still missing. Another research direction might make use of more sophisticated clustering methods (like spectral clustering) that were beyond the scope of this work.
Appendix A.

Contents of the CD

All the tools required to obtain the results presented in this work are contained on the CD — as well as the electronic version of this thesis.

A.1. Contents

- /Horak_Karel.pdf
  Electronic version of this document
- /Data_DE_XM-11-8-11.xlsx
  The dataset that was analyzed in this work
- /Code
  IntelliJ IDEA project used to generate the results
- /SpectralClustering
  Bachelor thesis of Ingo Bürk on spectral clustering topic\cite{28}
Appendix B.

Implementation notes

All the code that was implemented as a part of this thesis is located in the /Code directory on the CD.

B.1. Used libraries and software

- **Math Commons** library (http://commons.apache.org/proper/commons-math/) is used for computing quantiles of normal distribution.
- **Google Charts API** (https://developers.google.com/chart/interactive/docs/gallery/geochart) is used for visualizing the output.
- **Spectral Clustering** (part of bachelor thesis of Ingo Bürk[13]) is used to cluster the dataset using the spectral clustering methods.

B.2. Code overview

The code is divided in several package. We will go through the most important of them.

The clustering algorithms used in this work are contained in the algorithm.clustering package. Each of these algorithms is instantiated by its parameterless constructor — but it is necessary to provide the representation used for clustering (e.g. **new KMeans[Datapoint]**). These algorithms are configurable by the set of options using the bang (!) method (e.g. **new KMeans[Datapoint] ! NumClusters(4)**):

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KMeans</td>
<td>NumClusters(n: Int)</td>
<td>(sets the number of clusters)</td>
</tr>
<tr>
<td></td>
<td>NumRetries(n: Int)</td>
<td>(number of independent runs of the algorithm)</td>
</tr>
<tr>
<td>KMedoids</td>
<td>NumClusters(n: Int)</td>
<td>(sets the number of clusters)</td>
</tr>
<tr>
<td></td>
<td>NumRetries(n: Int)</td>
<td>(number of independent runs of the algorithm)</td>
</tr>
<tr>
<td>CMeans</td>
<td>NumClusters(n: Int)</td>
<td>(sets the number of clusters)</td>
</tr>
<tr>
<td></td>
<td>NumRetries(n: Int)</td>
<td>(number of independent runs of the algorithm)</td>
</tr>
<tr>
<td>CSV</td>
<td>NumClusters(n: Int)</td>
<td>(sets the number of clusters)</td>
</tr>
</tbody>
</table>

The **CSV** class does not represent the algorithm itself — it is only an interface to load the results from the Spectral Clustering.

Another important classes reside in the algorithm.clustering.representation package. These classes corresponds to the representations (and distance measures) introduced in this work (**DatapointEuclidean**, **DatapointRelaxedEuclidean**, **SAX**, **DatapointDDTW**). Instances of majority of these representations are obtained by calling the constructor of respective class (e.g. **new Datapoint("label", List(...)**)). The only exception from this principle is the **SAX.saxify** method exists to simplify the matter.
The `algorithm.sequence` package contains the `normalize` object whose application performs the sequence normalization.

In the `algorithm.clustering.analysis` the `elbow` object was used to obtain the values needed to construct an elbow graph of the clustering problem.

Finally the `io` package provides classes for dataset manipulation and the `visualization` package provides the `show` method that performs the clustering and displays the result.

For examples how to assemble these classes to a working application, see the `app` package.
Appendix C.
Clustering results

Following tables represent clustering results obtained in chapter 7. The country codes correspond to the ISO 3166-1 norm\textsuperscript{[14]}.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Size</th>
<th>Countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 0</td>
<td>75</td>
<td>MA, VN, KW, ML, BG, SG, AT, BD, TR, MY, RU, TN, JO, ES, AZ, SM, PK, NZ, US, SI, BH, HK, SD, RO, LU, FO, BW, IE, IR, IQ, BE, AU, SA, MD, HU, GT, TM, FI, NO, EE, LT, SY, CH, LV, CY, SE, KZ, BF, HR, BA, GB, FR, MT, RW, PT, KR, UA, LI, GA, YE, HN, IT, CZ, GR, ZA, XS, MX, SK, DZ, DK, NL, LA, CA, CN, TD</td>
</tr>
<tr>
<td>Cluster 1</td>
<td>45</td>
<td>AO, IN, EG, GQ, VE, MR, NA, MV, KY, KG, LY, MZ, BB, TZ, AF, CL, PL, NG, UG, CG, BI, AE, CO, ZM, KH, TJ, AM, BR, GN, CD, MG, GH, GY, BY, OM, KE, MK, QA, AL, ET, SN, PE, GE, SC, EC</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>16</td>
<td>ID, TH, CF, CI, MM, UY, ZW, PG, BO, MN, ER, AR, UZ, BN, PY, PA</td>
</tr>
<tr>
<td>Cluster 4</td>
<td>14</td>
<td>SV, HT, PH, VC, SZ, MU, MW, IS, LS, FJ, LK, CU, VI, BM</td>
</tr>
</tbody>
</table>

Table 3. k-means clustering for $k = 5$ (see figure 12)
<table>
<thead>
<tr>
<th>Cluster</th>
<th>Size</th>
<th>Countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>52</td>
<td>MA, VN, KW, BG, SG, AT, BD, TR, MY, RU, JO, AZ, PK, SI, BH, SD, RO, LU, IR, BE, AU, SA, MD, AF, CL, NO, EE, SY, PL, CH, CY, AE, KZ, BF, HR, BA, RW, KR, UA, LI, GA, YE, CZ, GR, ZA, XS, SK, DZ, DK, NL, CN, TD</td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>AO, IN, GQ, MV, KG, TZ, UG, BI, CO, ZM, TJ, GH, BY, OM, KE, MK, QA, LA, AL, ET, PE, GE, SC</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>ES, SM, NZ, HK, FO, IE, HU, FI, LT, LV, IS, SE, GB, FR, MT, PT, IT, MX, CA</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>EG, VE, TN, KY, BB, NG, CG, BR, GN, MG, GY, SN</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>MR, NA, BW, LY, MZ, IQ, TM, AM, CD, EC</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>NI, GD, US, MU, GT, PF, TG, JP, CR</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>ID, ZW, PG, BO, ER, UZ, BN, PY</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>CI, MM, UY, MN, AR, KH, PA</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>PH, SZ, LS, FJ, TT, HN, LK</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>TH, MH, AD, CF, NP, BS</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>IL, TW, AN, MW, AG, LR</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>KP, MO, JM, SL, NE, LB</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>ML, BZ, CM, GL, DO, BJ</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>SV, HT, VC, CU, VI, BM</td>
</tr>
</tbody>
</table>

Table 4. k-means clustering for $k = 14$ (see figure 13)
<table>
<thead>
<tr>
<th>Size</th>
<th>Countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
<td>AT, MA, VN, KW, ML, BG, SG, BD, TR, MY, RU, JO, ES, AZ, MR, PK, NZ, NA, SI, BH, SD, RO, LU, VC, FO, BW, MZ, SZ, IE, IR, IQ, BE, AU, TZ, SA, MD, AF, HU, TM, CL, FI, NO, EE, LT, SY, PL, CH, NG, LV, CY, SE, AE, KZ, BF, HR, BA, FR, LS, FJ, RW, KR, AM, TT, UA, LI, GN, CD, GA, YE, HN, IT, CZ, GR, ZA, BY, XS, SK, DZ, QA, CU, DK, VI, NL, LA, CN, AL, TD, EC</td>
</tr>
<tr>
<td>36</td>
<td>PE, AO, IN, EG, GQ, VE, TN, MV, CI, KY, KG, LY, BB, UY, MN, UG, MW, CG, BI, AR, CO, ZM, KH, TJ, BR, PA, MG, GH, GY, OM, KE, MK, ET, SN, GE, SC</td>
</tr>
<tr>
<td>12</td>
<td>UZ, ID, TH, HT, CF, MM, ZW, PG, BO, ER, BN, PY</td>
</tr>
</tbody>
</table>

**Table 5.** k-medoids clustering using Euclidean distance for \( k = 4 \) (see figure 15)

<table>
<thead>
<tr>
<th>Size</th>
<th>Countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>73</td>
<td>DK, MA, VN, KW, BG, SG, AT, GQ, MY, RU, TN, JO, ES, SM, SI, BH, SD, AD, RO, LU, FO, BW, CI, IE, BE, AU, SA, MD, AF, HU, BO, TM, NE, CL, NO, EE, LT, SY, CH, NG, LV, UG, IS, SE, AE, AR, HR, BA, GB, FR, KH, TJ, KR, AM, UA, LI, TG, CD, GA, YE, IT, CZ, GR, XS, OM, KE, SK, DZ, NL, CN, ET, TD, EC</td>
</tr>
<tr>
<td>23</td>
<td>GL, ID, SV, HT, MH, BZ, KP, TW, PH, JM, CM, AN, VC, CF, ZW, MN, ER, MW, BS, AG, BN, TT, VI</td>
</tr>
</tbody>
</table>

**Table 6.** k-medoids clustering using relaxed Euclidean distance for \( k = 4 \) (see figure 16)
<table>
<thead>
<tr>
<th>Cluster</th>
<th>Size</th>
<th>Countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 2</td>
<td>28</td>
<td>YE, VN, SV, GQ, MH, TW, PH, SM, HK, SD, AD, FO, CF, KG, NP, ZW, MU, LV, IS, BI, KZ, ZM, KH, CU, LA, BM, SC, LR</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>15</td>
<td>AF, EG, BD, VE, NA, KY, LY, MZ, BB, PG, CG, MG, CY, OM, SN</td>
</tr>
</tbody>
</table>

**Table 7.** k-medoids clustering using Derivative DTW with \( k = 4 \) (see figure 17)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Size</th>
<th>Countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>44</td>
<td>IT, KW, ML, SG, SV, TN, TW, ES, SM, NZ, US, BH, HK, SD, FO, IE, IQ, HU, GT, FI, LT, NG, LV, IS, SE, BA, GB, FR, PF, MT, PT, LI, TG, GN, HN, CZ, GR, ZA, MX, DK, CA, JP, CR, TD</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>15</td>
<td>UZ, ID, TH, HT, CI, MM, ZW, PG, BO, ER, AR, BN, KH, PY, GY</td>
</tr>
</tbody>
</table>

**Table 8.** k-medoids clustering using SAX distance \((k = 4, 12\) symbols — see figure 19)
| Cluster 1 | 35 | LR, EC, PE, CN, JP, BM, DK, CU, MK, KE, OM, LK, GH, CZ, YE, PA, TT, MT, AG, UZ, BS, SE, IS, PL, HU, MD, IQ, SD, BH, AZ, MH, RU, TR, BD, AO |
| Cluster 2 | 29 | CA, SK, MX, ZA, GR, BR, UA, KR, PT, HR, AE, NO, FI, SA, AU, IR, IE, Lu, Ro, HK, SI, TW, IL, MY, TH, SG, EG, ID, IN |
| Cluster 3 | 28 | QA, DZ, XS, BY, LI, CO, BA, AR, LB, KZ, CY, LV, NG, SY, LT, EE, CL, LY, NZ, PK, JO, PH, TN, VE, BG, KW, VN, MA |
| Cluster 4 | 9  | NL, IT, FR, GB, CH, BE, US, ES, AT |

Table 9. Spectral clustering (5 clusters — see figure 21)
Bibliography


