Scale and Time Independent Clustering of Time Series Data

Florian Steinwidder[1,2][a], Istvan Szilagyi[3][b], Eva Eggeling[2][c] and Torsten Ullrich[1,2][d]

[1] *Institute of Computer Graphics and Knowledge Visualization, Graz University of Technology, Graz, Austria*

[2] *Fraunhofer Austria Research GmbH, Graz, Austria*

[3] *Medical University Graz, Graz, Austria*

Keywords: Time Series Analysis, Cluster Analysis, Visualization Toolkits.

Abstract: The analysis of time series, and in particular the identification of similar time series within a large set of time series, is an important part of visual analytics. This paper describes extensions of tree-based index structures to find self-similarities within sets of time series. It also describes filters that extend existing algorithms to better fit real-world, error-prone, incomplete data. The ability of time series clustering to detect common errors in real data is also described. These main contributions are illustrated with real data and the findings and lessons learned are summarised.

1 INTRODUCTION

The analysis of data in which a temporal component is an essential aspect is called time series analysis. These data are recorded over time and are assumed to have an internal time-dependent structure that should be taken into account when building models. The application areas are numerous and continue to grow with digitisation and the associated increase in data collection of all areas of society and industry is leading to a greater need for data analysis in general. In the case of monitoring, time series analysis provides the essential tools for forecasting. Despite the variety of applications in very different domains, the mathematical toolbox remains the same (Ott and Longnecker, 2015). In addition to predicting future values, finding similar time series in a collection of data, i.e. clustering time series (Aghabozorgi et al., 2015), (Hennig et al., 2015), is an important task in exploratory data analysis and in preparation for model building (Hochheiser and Shneiderman, 2003), (Neamtu et al., 2016).

In this work the focus is on time series clustering and the task to identify similar time series in a large data collection.

[a] https://orcid.org/0009-0004-5337-1195
[b] https://orcid.org/0000-0001-7542-3911
[c] https://orcid.org/0000-0001-6703-2865
[d] https://orcid.org/0000-0002-7866-9762

2 RELATED WORK

A *time series* T is a sequence of pairs (x_i, y_i), which consists of a time component x_i and an arbitrary component y_i. The time component can be continuous $x_i \in \mathbb{R}$ or discrete $x_i \in \mathbb{Z}$ (if the absolute timing is less important than the relative timing, the data set may be indexed with (semi-) positive integers $x_i \in \mathbb{N}$). If the context describes the timing implicitly, e.g. by a regular sampling in fixed intervals, the time component may be omitted.

The identification of similar time series is called “twin subsequence search”. In detail, the problem is to find subsequences S in a larger time series T that are similar to a query sequence Q. The subsequences have the same length as the query sequence and their similarity is defined by a distance metric.

The naive approach to finding similar subsequences in a time series is to use a sweepline scan, moving a sliding window of the same length as the query sequence Q along the time series and comparing at each time stamp the distance between Q and the subsequence currently covered by the sliding window. Obviously, this approach is not efficient for time series consisting of many time stamps. Instead, index-based methods can be used to search for twin subsequences, which have the advantage of being more efficient and scalable.

Steinwidder, F., Szilagyi, I., Eggeling, E. and Ullrich, T.
Scale and Time Independent Clustering of Time Series Data.
DOI: 10.5220/0012377000003660

In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2024) - Volume 1: GRAPP, HUCAPP and IVAPP*, pages 583-592
ISBN: 978-989-758-679-8; ISSN: 2184-4321

2.1 KV-Index

The key-value index (KV index) exploits the property of data locality, which states that the values of successive time stamps are often close to each other (Wu et al., 2019). Thus, adjacent sliding windows will have similar mean values. The index is constructed over all subsequences of a predefined length that can be extracted from an input time series. A subsequence is represented as a pair (p,μ), where the first entry denotes its starting position and the second entry denotes the mean value over the time stamps covered. These pairs are used to construct an inverted index data structure with ordered rows. The key of each row represents a range of mean values, while the corresponding value is a list of starting positions of subsequences whose mean values fall within the range given by the key.

The KV index can be used to search for twin subsequences in the following way: two subsequences S and S' of the same length are twins if the Euclidean distance between them is less than a predefined threshold ε. It follows that the means of two twin subsequences cannot differ by more than this threshold. For a given query sequence with mean μ_q, potential twin subsequences are those within a list with key $[\mu_{min}, \mu_{max}]$, such that

$$\mu_{min} - \varepsilon \leq \mu_q \leq \mu_{max} + \varepsilon.$$

Each candidate obtained must be verified by calculating its actual distance to the query sequence before it can be called a twin subsequence.

2.2 iSAX

The indexable symbolic aggregate approximation (*i*SAX) is used for similarity search between *z*-normalised time series (Shieh and Keogh, 2008). *Z*-normalisation (Goldin and Kanellakis, 1995) ensures, that all elements of the input vector are transformed into an output vector whose mean is 0 and whose standard deviation is 1. Specifically, the time series mean is subtracted from the original values and the difference is divided by the standard deviation value. According to most of the recent work on time series structural pattern mining, *z*-normalisation is an essential preprocessing step that allows a mining algorithm to focus on the structural similarities/dissimilarities rather than the amplitude-driven ones. *i*SAX is a tree-based structure that indexes time series based on their symbolic aggregate approximation (SAX). A time series is transformed into its symbolic representation by the following two steps:

- In the first step, the piecewise aggregate approximation (PAA) (Keogh et al., 2001) is applied to the time series. This involves dividing the time series into a predefined number m of segments and approximating each segment by its mean.
- In the second step, a discrete SAX symbol X is assigned to each mean using specified breakpoints pre-computed for *z*-normalised time series. This allows a time series to be represented as a sequence of m SAX symbols. This sequence is called a SAX word, and each SAX symbol X within the word corresponds to a range of mean values $[\mu_{X_{min}}, \mu_{X_{max}})$.

A *i*SAX index can be constructed from one or more SAX words. Each node in the index contains one SAX word and represents a subset of the SAX space. As already observed for the KV-Index, the difference between the mean values of two subsequences S and S' of length l and the SAX representations

$$SAX(S) = \{X_1, X_2, \dots, X_m\}$$

and

$$SAX(S') = \{X'_1, X'_2, \dots, X'_m\}$$

are bounded by a threshold ε, if S and S' are twins. Furthermore, if S and S' are twin subsequences with respect to ε, then any two time-aligned segments extracted from S and S' are also twins. Based on these two properties, two subsequences can be identified as twins if the difference between the mean values of each pair of SAX symbols X_i and X'_i in their SAX words is less than or equal to a predefined threshold.

Twin subsequence search can be performed for a given time series T and query sequence Q by first extracting all subsequences of length l from T and creating a *i*SAX index over these subsequences. The index is then traversed from top to bottom, checking at each node whether the node's SAX word satisfies the above properties when compared to the query's SAX word. If the difference between the mean values of the SAX symbols contained in the node and the SAX symbols of the query sequence is above the threshold, the node can be ignored and its subtree pruned. On the other hand, if the properties are met, the search continues until a leaf node is reached. Subsequences indexed within this leaf node are possible twin subsequences and are used for a final check.

2.3 TS-Index

The time series index (TS-Index) is a tree-based index dedicated to the twin subsequence search problem (Chatzigeorgakidis et al., 2021). Its leaf nodes store subsequences of the same length as the length of a given query sequence, while the nodes above summarise the contained subsequences. This allows the filter-verification algorithm to prune the search space, speeding up the search for twin subsequences within the TS index. The nodes summarise the contained subsequences using "Minimum Bounding Time Series" (MBTS), the core concept of the TS-Index. An MBTS consists of two sequences enclosing a set of time series. This pair of sequences $B^{\sqcap}$ and $B^{\sqcup}$ forms a tube, with the upper sequences running along the maximum values and the lower along the minimum values at each time stamp (see Figure 1).

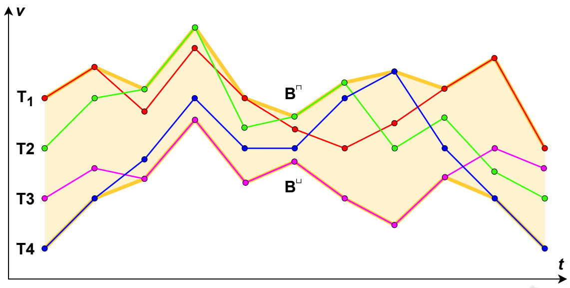

Figure 1: A set of four time series is enclosed by a "Minimum Bounding Time Series" (MBTS) using the minimum and maximum values at each time stamp. The result is a pair of time series with upper $B^{\sqcap}$ and lower $B^{\sqcup}$ limit. Although all time series are equally sampled in this example illustration, this is not a requirement. Time series with different sampling can also be combined to form an MBTS.

As mentioned above, the TS-Index has a tree data structure. Its internal nodes point to child nodes at the next level. Each leaf node points to the starting positions, relative to the input time series T, of its indexed subsequences. All leaf nodes must be on the same level. Furthermore, each internal node as well as each leaf node has an MBTS that encloses all indexed subsequences contained in the node. As the number of indexed subsequences per node decreases from the root node downwards, the MBTS also become narrower.

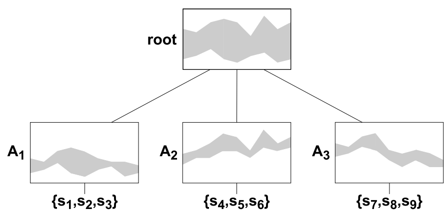

Figure 2: The TS-Index is a tree structure with a pair of boundary time series in each node.

The TS-Index construction requires an input time series T and a desired subsequence length l. The index is constructed by sequentially extracting all possible subsequences of l length from T and inserting them one by one into the TS-Index. As the construction follows a top-down approach, the tree must be traversed from the root node to its leaf nodes when inserting a subsequence S. At each level, the MBTS of the nodes within are compared with S. By repeatedly selecting the nodes with the smallest distance to S at each level, a leaf node is reached. Among all leaf nodes, this leaf node has the smallest distance between its MBTS and S. Therefore, the starting position of S can be added to the set to which this leaf node points. As subsequences are added to the TS-Index, a minimum-maximum criterion forces nodes to split in order to evenly distribute the number of children per node. The resulting tree structure is shown in Figure 2.

For a given query sequence Q, the twin subsequence search uses a filter-verification algorithm. As a prerequisite, the query sequence Q must be of the same length as the subsequences used to construct the TS-Index.

A subsequence S is similar to the query sequence Q if the distance between the two sequences is less than or equal to a desired threshold ε. Each node N within the TS-Index contains an MBTS that bounds all subsequences indexed by N. Thus, once the filter-verification algorithm encounters a node where the distance between its MBTS and Q is greater than ε, the search space can be pruned, as the algorithm does not need to check for similar subsequences in N and N's child nodes. In this procedure, the algorithm *filters* leaf nodes that contain possible twin subsequences compared to Q. Then, in the *verification* phase, each subsequence S contained in the filtered leaf nodes is compared to Q.

3 METHOD

The extended method presented in this paper is based on the TS-Index data structure introduced by Chatzigeorgakidis et al. (Chatzigeorgakidis et al., 2021). It extends the "Twin Subsequence Search in Time Series" approach in several ways:

- tree balancing for better performance,
- exploration method using self-similarities,
- not-a-number handling, and
- uncertainty quantification.

3.1 Unbalanced Trees

The original implementation of TS-Index uses a minimum-maximum criterion during the tree construction to ensure an almost balanced tree for optimal run-time performance. The tree is constructed by inserting one time series at a time. The maximum capacity specifies the limit when a node has to be split into two new ones; the minimum capacity defines the number of child nodes a parent node must at least point to. When a node is split into two new nodes, its indexed subsequences are assigned to one of the two new nodes based on their similarity. The main assumption of this approach is a common uniform distribution of all time series with a positive expectation value of the distance between two series. If the distribution differs significantly, especially if several time series are the same, the advantages of the tree structure for speeding up the search are no longer effective and the disadvantages of the overhead of the acceleration structure prevail.

Therefore, our implementation does not use a minimum and maximum number of child nodes, but the distance measure as a separation rule. Similar (and especially identical) time series should be close together in the tree structure, ideally in the same node; very different time series should not be unnecessarily grouped together. This is particularly useful if identical series occur frequently in the data.

3.2 Self-Similarities Between all Time Series

Most data structures are optimized for a single search task consisting of a query time series Q and a set $\mathcal{T}$ of time series $\{T_1, T_2, \ldots, T_n\}$. The desired result is the element T_i with the smallest distances to Q – the so-called nearest neighbour of Q in $\mathcal{T}$.

In the context of exploratory time series analysis, the task is similar, but differs in one important aspect: the nearest neighbour query does not consist of a single time series Q, which may not even be an element of the set $\mathcal{T}$, but of all elements $\{T_1, T_2, \ldots, T_n\}$ of $\mathcal{T}$. This extended search for similar elements in a set inevitably leads to many "hits" – at least the number of elements in $\mathcal{T}$. When all elements of a set are compared with all elements of the same set, these "hits" occur on the diagonal:

$$\begin{array}{ccccc} & T_1 & T_2 & \cdots & T_n \\ T_1 & d(T_1,T_1) & d(T_1,T_2) & & d(T_1,T_n) \\ T_2 & d(T_2,T_1) & d(T_2,T_2) & & d(T_2,T_n) \\ \vdots & & & \ddots & \\ T_n & d(T_n,T_1) & d(T_n,T_2) & & d(T_n,T_n) \end{array}$$

These diagonal elements are usually not wanted in the result list of similarities and can be easily filtered out.

However, this problem does not only occur with identical subsequences of the same time series, but also with small temporal shifts in the start times of two subsequences of the same sequence. The question of whether these similarities are desirable in the list of results cannot be answered without context. While small temporal shifts should normally be filtered out, similarities with a large temporal offset indicate cyclical, seasonal or generally recurring patterns within a time series and are therefore desirable detection results. We have solved this problem with a new parameter Δ, which filters out self-similarities within the same time series by means of a minimum time distance threshold. This solution generalises the filtering to avoid the "diagonal self-similarity problem" described above.

3.3 Not-A-Number

Apart from the theory, time series are characterised by a multitude of potential problems: synchronisation problems, calibration problems, sensor failures, etc. Every real-world system therefore needs a strategy for dealing with these problems.

Our approach contains three filters that require a valid parameter range to be specified for each time series: Values that are below this parameter range are replaced by a user-defined value; values that exceed this parameter range are replaced by a second user-defined value; undefined values (not-a-number) are replaced by a third value.

The choice of these three replacement values should be adapted to the metric used; in particular, the error case of not-a-number propagation through the calculation should be taken into account (Monniaux, 2008).

3.4 Uncertainty and Inaccuracy

Real data, even if complete, is always subject to measurement error and imprecision (Fuller, 2006). This imprecision should be represented together with the data and also taken into account in the distance and comparison metrics (Wang and Bi, 2021). By consistently accounting for imprecision, a distance metric on time series becomes a statistical adjustment test (Zhang and Chen, 2018).

The approach we have chosen to cope with imprecision and uncertainty extends a time series $x_0, x_1, x_2, x_3, \ldots$ by quantifying uncertainty in terms of standard deviations $x_i \pm t_i$:

$$x_0 \pm t_0, x_1 \pm t_1, x_2 \pm t_2, x_3 \pm t_3, \ldots$$

In the visualisation, this inaccuracy can be reflected in a diffuse interval that is drawn semi-transparently as an offset (Bhatt et al., 2021). Figure 3 shows the visualisation of a time series including its standard deviations $x_i \pm t_i$.

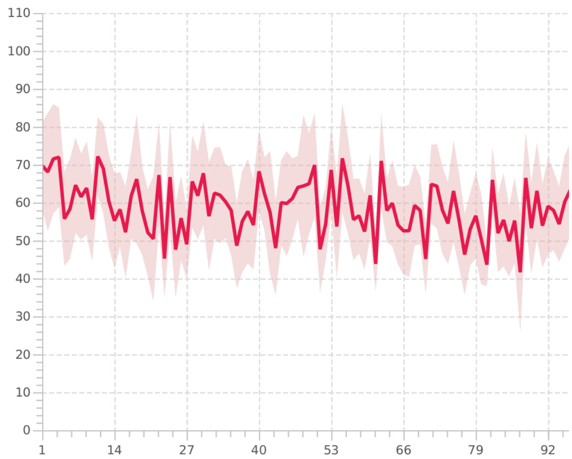

Figure 3: A time series of real data often contains measurement errors and imprecisions. One method of communicating these measurement errors and inaccuracies is to use semi-transparent intervals surrounding a time series.

3.5 System Implementation

The TS-Index data structure and the filter verification search algorithms are implemented in Java. In addition, a graphical user interface (GUI) is developed using JavaFX to facilitate user interaction and provide visualisations of the search results. The implementation of the system can be divided into two steps: the construction phase and the clustering process.

In the construction phase, each time series is split into all its possible subsequences using a sliding window. For each time series, a separate TS-Index is constructed by sequentially inserting the extracted subsequences. This approach is chosen over inserting all subsequences into a single TS-Index due to the improved efficiency it offers: This method allows us to take advantage of multithreading, starting separate threads for the creation of each time series' TS-Index. In contrast, inserting all subseries sequentially into a single large TS-Index would lead to a higher number of node splits and distance calculations, resulting in slower performance. The maximum node capacity parameter is set to $max = 5$, which strikes a good balance between minimising the time required for node splits and maximising the efficiency gained by early search space pruning.

For the cluster creation, query sequences covering a specified time span are extracted from each time series. The filter-verification algorithm is then applied to each query sequence to search for similar subsequences within the previously constructed TS-Indices. Similar to the construction phase, the identification of similar subseries is performed concurrently on separate threads for each query to further increase efficiency. A threshold value is used to quickly decide whether two time series are similar: The threshold is set to the expected value of the absolute difference between two stochastically independent, uniformly distributed random variables X, Y. Distances greater than this expected value $\mathrm{E}(|X - Y|)$ indicate a higher probability of random occurrence, which means that subsequences with such distances are not considered to be twin subsequences of the query.

For each query sequence, the twin subsequences identified by the filter verification algorithm are further sorted based on their Chebyshev distance to the query. If the number of identified subsequences exceeds the desired cluster size, the excess subsequences are removed. Notably, since the algorithm compares the query to all time series indices, the query sequence itself is included in the results due to its distance of 0, thus ensuring its presence within the cluster. Within a cluster, the subsequence with the largest Chebyshev distance determines the overall cluster distance, allowing the clusters to be ranked from most similar to least similar.

The GUI provides a visual interface for users to interact with the system. Through the GUI, users can select a metadata and time series attributes that will display the corresponding time series in a line graph. The graph provides information on observed values and standard deviations. In addition, by hovering over the time series, users can view different time series values at the same time. To determine the time span of queries for the clustering process, users can define a specific start and end date of interest. The GUI allows users to create clusters of a specified size containing subsequences from all time series, or to perform a faster twin subsequence search for a specific country and search term. The latter option also allows users to specify a similarity threshold.

4 EVALUATION

The presented system is primarily used for the exploration of one's own time series and is adapted to the specific needs and peculiarities of one's own data.

4.1 Medical Data Analysis

The use case of the application is the data exploration of several datasets, all based on the idea described in "Detecting influenza epidemics using search engine query data" (Ginsberg et al., 2009): people who notice symptoms of an illness first google for medical advice before consulting a doctor. Thus, days before an officially confirmed influenza epidemic (based on

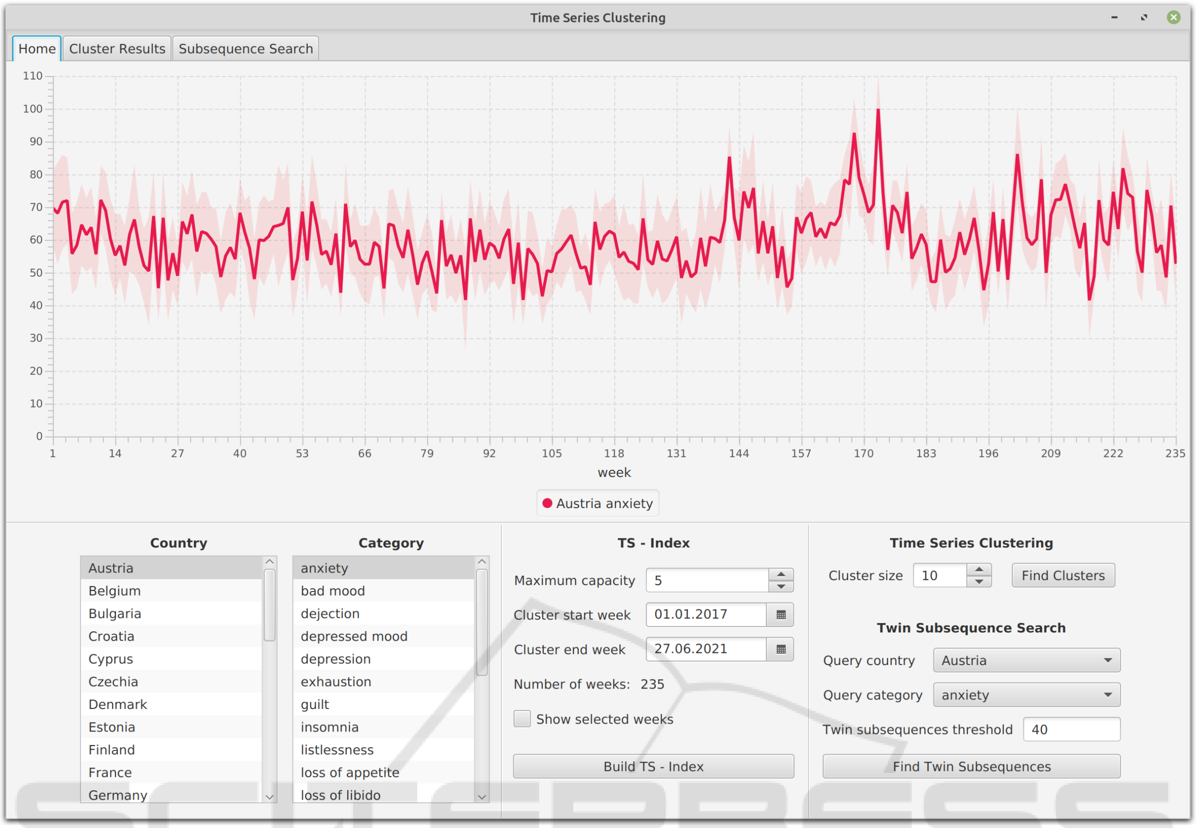

Figure 4: The first visual inspection of the time series input data has the goal to check completeness and plausibility. Furthermore, this is the starting point of the interactive visual analytics process.

diagnoses with pathological test results), the Google database already contains clusters of influenza-like symptoms as search terms.

Following this idea, the COVID-19 pandemic has been analysed. Each dataset contains a time series of one country on a search term translated into the local language and associated with a direct pandemic symptom or an indirect health impact. The task of exploratory data analysis is to look for similarities and differences between countries and to relate these to the number of cases and countermeasures (lockdown, etc.) taken by each country. In detail, the dataset consists of a list of countries. For each country it comprises

1. a time series of COVID-19 cases reported to/by the European Centre for Disease Prevention and Control (ECDC),
2. a time series of COVID-19 counter measures (closure of ed. institutions, stay-at-home orders, ...) reported to/by ECDC,
3. a translated list of direct pandemic symptoms (back pain, chest pain, ...) and indirect health impacts (anxiety, depression, ...) into the official language(s) of the corresponding country (based on the International Classification of Diseases, ICD-10), and
4. a time series of Internet query frequencies for each entry of the translated list as reported by Google Trends.

The starting point of the analysis is transformation and normalisation of the individual time series (Keim et al., 2008); Google Trends does not provide absolute numbers of Internet queries, but only relative increases and decreases. This step is completely automatic; the starting point of the interactive visual analysis is the inspection of the data to visually check its completeness and plausibility. Figure 4 shows the graphical user interface (GUI) after loading and transforming the dataset at the starting point of interactive data exploration.

The visual analytics process aims to make the best use of huge amounts of information in a wide range of applications by combining the strengths of intelligent automatic data analysis with the visual perception and analysis skills of the human user (Kohlhammer et al., 2011). This interactive process combines data transformation, model building and model visu-

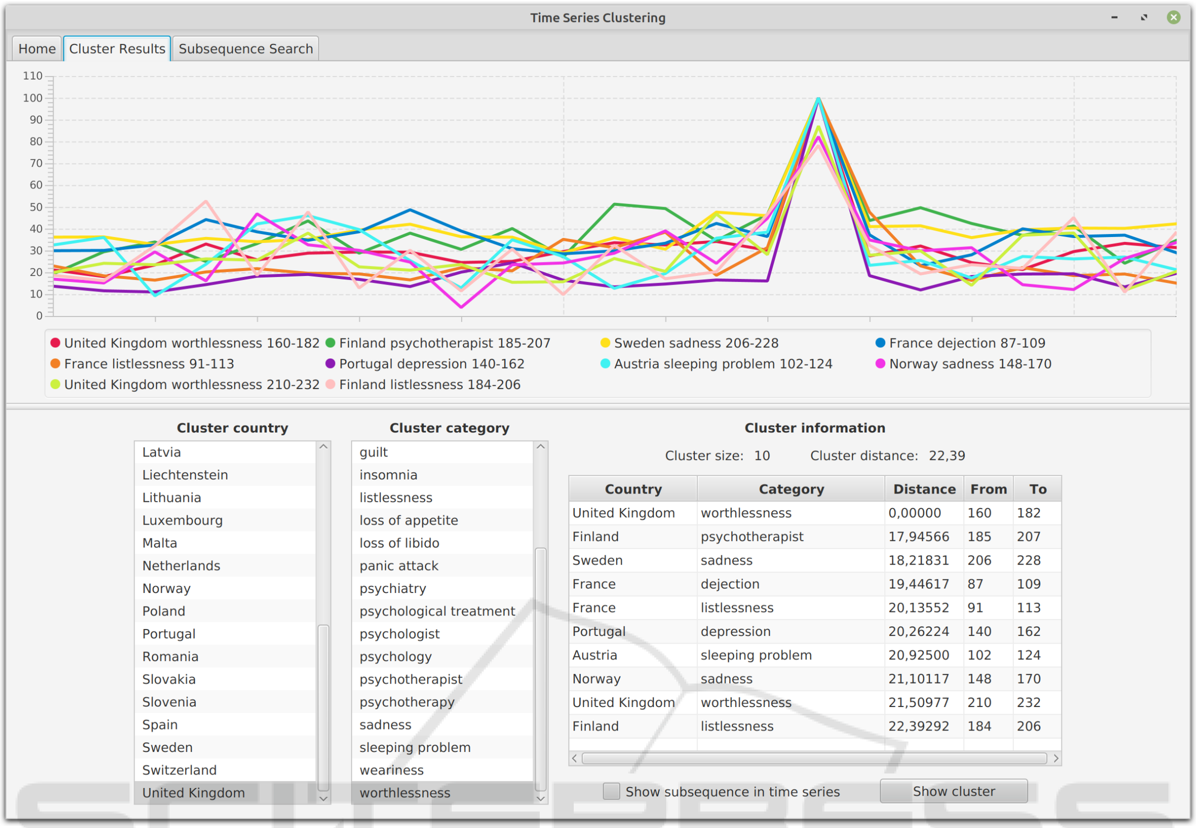

Figure 6: This screenshot shows a time series cluster centred on the time series "UK worthlessness 160–182"; the time series describes the frequency of internet search queries from the *UK* containing the term *worthlessness* over the period from the *160th* to the *182nd* week since 1 January 2017.

alisation with the aim of gaining knowledge in an iterative loop (see Figure 5).

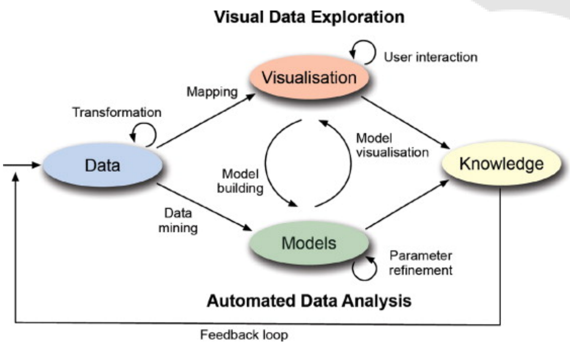

Figure 5: The visual analytics knowledge discovery process combines data transformation, model building and model visualisation in an iterative feedback loop (Keim et al., 2010). Image source: (Kohlhammer et al., 2011).

In this particular application, the modelling and visualisation is based on unsupervised clustering, which identifies the time series that are similar. The common features found were then correlated with the context (countermeasures) of the respective country to identify a general pattern. This pattern was then tested for plausibility and further refined by domain experts from medicine and psychology. Figure 6 shows one cluster of the clustering process applied to all time series data. The similarity of the individual time series within this cluster to each other is particularly striking from a visual point of view. This interactive process resulted in common patterns identified in the data; i.e., appropriate knowledge was extracted. However, these findings need to be confirmed. For this purpose, the extracted hypotheses were analysed with statistical tests and their significance was evaluated. The medical and psychological findings are published in "Google Trends for Pain Search Terms in the World's Most Populated Regions Before and After the First Recorded COVID-19 Case: Infodemiological Study" (Szilagyi et al., 2021) and "Impact of the pandemic and its containment measures in Europe upon aspects of affective impairments: a Google Trends informetrics study" (Szilagyi et al., 2023) and are not the primary concern of this paper, which focuses on the process, the tools used in the process, and the lessons learned.

4.2 Filter Results & Effects

The first step is to find similar time series in the frequencies of internet search queries from different countries. The results are not always as nice as shown in Figure 7.

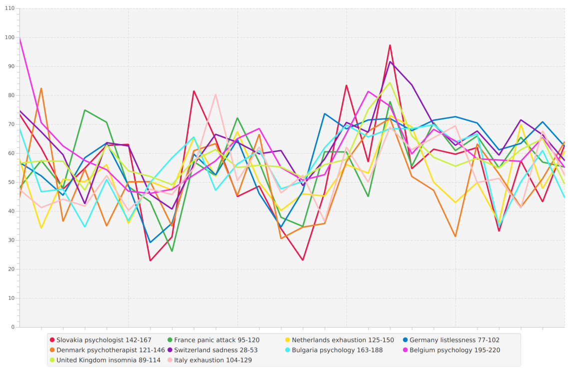

Figure 7: Different keywords from searches in different countries can show the same frequency over time.

However, some clusters from unsupervised cluster analysis stand out from the rest, forming a subset of time series that not only share similarities but also a common problem. If there are not enough queries from a country, Google Trends returns a corresponding flat line (complete or temporary). These zero lines, indicating data gaps, are obviously similar to each other and are therefore grouped into a cluster (see Figure 8). Identifying these clusters and excluding them from further analysis reduces the subsequent analysis effort and is therefore desirable.

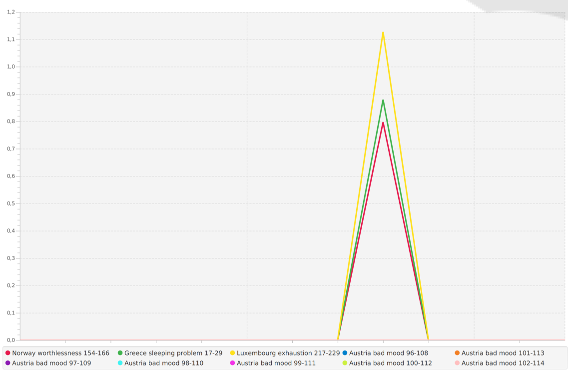

Figure 8: Different keywords from searches in different countries can show the same frequency over time.

An undesirable similarity is self-similarity. Figure 9 illustrates such a situation: the Figure shows the frequency distribution of the time series "Poland, panic attack, 196–221", i.e. the frequencies of search queries from Poland containing the keyword "panic attack" in the weeks 196 to 221 since 1.1.2017. This time series is of course similar to the time series "Poland, panic attack, 194–219", which is only offset by two weeks. Likewise, the cluster contains not only the time series that was two weeks earlier, but also the time series that was two weeks later and other variants with different time offsets.

This self-similarity is in the nature of time series. Any time series that is Lipschitz continuous has a bounded distance to a time-shifted instance of itself. This fact does not provide any new insight into the application domain of time series and is therefore not desirable.

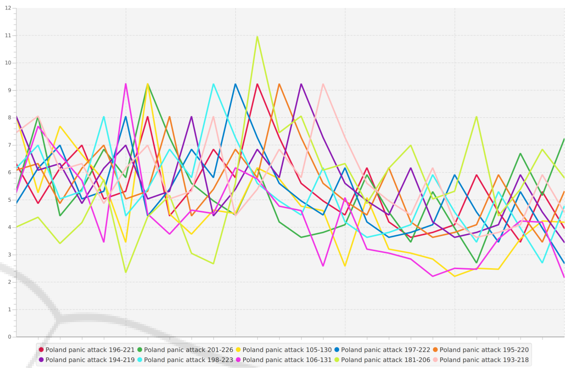

Figure 9: Most time series are similar to shifted versions of themselves. This fact provides no insight into the applied use case.

However, this "uninteresting" self-similarity should be excluded with caution in further analysis, as it becomes more "interesting" as the time shift increases. Figure 10 shows one such interesting shift, which prompted further investigation: Internet searches for sleep problems in the United Kingdom show an annual pattern. The corresponding time series with a high level of similarity

- "United Kingdom, insomnia, 39–60"
- "United Kingdom, insomnia, 91–112"
- "United Kingdom, insomnia, 143–164"

are shifted by exactly 52 weeks. This annual peak is a regularity that stimulates further analysis and research (outside of time series analysis and partly outside of medical or psychological domains of application). For this reason, the choice of the minimum distance between two instances of a time series that are shifted in time is critical and should not be made carelessly.

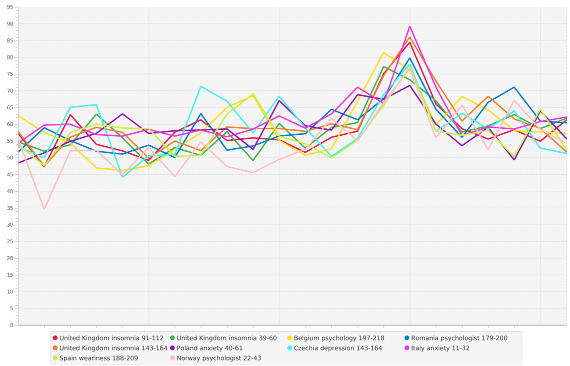

Figure 10: Seasonal and cyclical effects are self-similar. Detecting these properties within time series can provide new insights.

5 CONCLUSION

The presented time series analysis extends the state of the art by several improvements that have been implemented and tested on real data.

5.1 Contribution

Our contribution to the scientific community is the extension of the TS-Index approach to unbalanced trees in order to prioritise similarity over performance optimised tree structures – especially in visual analytics use cases with many similar elements, the similarity-based tree structure can be beneficial to analysis performance. Many identical elements can be quickly excluded when searching the tree.

The issue of self-similarity is also the subject of our second suggestion for improvement: self-similarity may be inherent in the time series or an indication of an external, application-specific factor. The detection of self-similarity is therefore both desirable and unnecessary noise in the analysis. The difference between the two possibilities is reflected in the time difference: the shorter the shift between two similar time series, the less interesting the similarity; large distances, on the other hand, indicate interesting patterns, especially seasonal or cyclical patterns. Appropriate filters, such as those we have implemented, offer the possibility of separation.

In addition to filtering out unwanted self-similarities, the efficient and effective handling of data problems (gaps, outliers, etc.) is a tedious but important issue. In theory, these cases are considered much less often than in practice. Here we have extended the existing algorithms with Not-a-Number and Out-Of-Range mapping filters. These allow to use of even short subsequences of valid time series and not to filter or discard them. Furthermore, the subsequent analysis steps do not need to consider special cases (NaN, PosInf, ...).

We have also added imprecision and uncertainty handling to the existing algorithms. Many time series (measurements, surveys, etc.) have precision information (range of variation, measurement precision, etc.) that usually goes unnoticed. In our implementation, this is taken into account throughout and is also displayed in the visualisations, if wanted.

5.2 Benefit

The lessons learned from this study are particularly valuable because the effects described are independent of the application domain and can occur in many different contexts.

The examples and the effects that occurred illustrate the problem of self-similarity and its ambiguous nature: self-similarities due to short shifts should be interpreted and filtered as noise; with increasing temporal distance, self-similarity gains importance and should be taken into account.

Another important benefit is the realisation that cluster analysis can also be used for data inspection and data transformation: similar errors often produce similar indications, an example of the feedback loop from knowledge back into data preparation.

ACKNOWLEDGMENT

The work was partially funded by the Austrian Research Promotion Agency (FFG) within the framework of the flagship project ICT of the Future PRESENT, grant FO999899544.

REFERENCES

Aghabozorgi, S., Shirkhorshidi, A. S., and Wah, T. Y. (2015). Time-series clustering – A decade review. *Information Systems*, 53:16–38.

Bhatt, U., Antorán, J., Zhang, Y., Liao, Q. V., Sattigeri, P., Fogliato, R., Melançon, G., Krishnan, R., Stanley, J., Tickoo, O., Nachman, L., Chunara, R., Srikumar, M., Weller, A., and Xiang, A. (2021). Uncertainty as a Form of Transparency: Measuring, Communicating, and Using Uncertainty. *AAAI/ACM Conference on AI, Ethics, and Society*, 4:401–413.

Chatzigeorgakidis, G., Skoutas, D., Patroumpas, K., Palpanas, T., Athanasiou, S., and Skiadopoulos, S. (2021). Twin Subsequence Search in Time Series. *International Conference on Extending Database Technology (EDBT)*, 24:475–480.

Fuller, W. A. (2006). *Measurement Error Models*. Wiley-Interscience, 1 edition.

Ginsberg, J., Mohebbi, M. H., Patel, R. S., Brammer, L., Smolinski, M. S., and Brilliant, L. (2009). Detecting Influenza Epidemics using Search Engine Query Data. *Nature*, 457:1012–1014.

Goldin, D. and Kanellakis, P. (1995). On similarity queries for time-series data: Constraint specification and implementation. *International Conference on Principles and Practice of Constraint Programming*, 976:137–153.

Hennig, C., Meila, M., Murtagh, F., and Rocci, R. (2015). *Handbook of Cluster Analysis*. Chapman and Hall/CRC, 1 edition.

Hochheiser, H. and Shneiderman, B. (2003). Interactive Exploration of Time Series Data. *The Craft of Information Visualization*, 1:313–315.

Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., and Melançon, G. (2008). Visual Analytics: Definition, Process, and Challenges. *Information Visualization (Lecture Notes in Computer Science)*, 4950:154–175.

Keim, D., Kohlhammer, J., Ellis, G., and Mansmann, F. (2010). *Mastering the Information Age Solving Problems with Visual Analytics*. CEurographics Association, 1 edition.

Keogh, E., Chakrabarti, K., Michael, P., and Mehrotra, S. (2001). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, 3:263–286.

Kohlhammer, J., Keim, D., Pohl, M., Santucci, G., and Andrienko, G. (2011). Solving Problems with Visual Analytics. *Procedia Computer Science*, 7:117–120.

Monniaux, D. (2008). The pitfalls of verifying floating-point computations. *ACM Transactions on Programming Languages and Systems*, 30:1–41.

Neamtu, R., Ahsan, R., Rundensteiner, E., and Sarkozy, G. (2016). Interactive Time Series Exploration Powered by the Marriage of Similarity Distances. *Very Large Data Base (VLDB) Endowment*, 10:169–180.

Ott, R. L. and Longnecker, M. T. (2015). *An Introduction to Statistical Methods and Data Analysis*. Cengage Learning, 7 edition.

Shieh, J. and Keogh, E. (2008). ISAX: Indexing and Mining Terabyte Sized Time Series. *International Conference on Knowledge Discovery and Data Mining*, 14:623–631.

Szilagyi, I. S., Eggeling, E., Bornemann-Cimenti, H., and Ullrich, T. (2023). Impact of the pandemic and its containment measures in Europe upon aspects of affective impairments: a Google Trends informetrics study. *Psychological Medicine*, online:1–13.

Szilagyi, I. S., Ullrich, T., Lang-Illievich, K., Klivinyi, C., Schittek, G. A., Simonis, H., and Bornemann-Cimenti, H. (2021). Google Trends for Pain Search Terms in the World's Most Populated Regions Before and After the First Recorded COVID-19 Case: Infodemiological Study. *Journal of Medical Internet Research*, 23:e27214.

Wang, W. and Bi, L. (2021). Research on strategies to improve model accuracy based on incomplete time series data. *Asian Conference on Artificial Intelligence Technology (ACAIT)*, 5:45–52.

Wu, J., Wang, P., Pan, N., Wang, C., Wang, W., and Wang, J. (2019). KV-Match: A Subsequence Matching Approach Supporting Normalization and Time Warping. *IEEE International Conference on Data Engineering (ICDE)*, 35:866–877.

Zhang, B. and Chen, R. (2018). Nonlinear Time Series Clustering Based on Kolmogorov-Smirnov 2D Statistic. *Journal of Classification*, 35:394–421.