


Finding Local Groupings of Time Series

Zed Lee ¹, Marco Trincavelli², and Panagiotis Papapetrou¹

¹ Stockholm University
{zed.lee, panagiotis}@dsv.su.se
² H&M Group
marco.trincavelli@hm.com

Abstract. Collections of time series can be grouped over time both globally, over their whole time span, as well as locally, over several common time ranges, depending on the similarity patterns they share. In addition, local groupings can be persistent over time, defining associations of local groupings. In this paper, we introduce **Z-Grouping**, a novel framework for finding local groupings and their associations. Our solution converts time series to a set of event label channels by applying a temporal abstraction function and finds local groupings of maximized time span and time series instance members. A grouping-instance matrix structure is also exploited to detect associations of contiguous local groupings sharing common member instances. Finally, the validity of each local grouping is assessed against predefined global groupings. We demonstrate the ability of **Z-Grouping** to find local groupings without size constraints on time ranges on a synthetic dataset, three real-world datasets, and 128 UCR datasets, against four competitors.

Keywords: Local groupings, temporal abstractions, time series

1 Introduction

Groupings of time series can be found in several application domains where multiple time series instances are collected and monitored. These groupings comprise sets of time series of high similarity over a time period (e.g., in terms of concurrently similar values or trends). Such groupings can span the whole time series length, defining *global* groupings, or shorter time periods, defining *local* groupings. Furthermore, some instances may persist being grouped together in consecutive local groupings over a longer time period, possibly separated by short time gaps, hence forming *associations* of local groupings. For example, Fig. 1-left shows two time series with four consecutive local groupings (red-blue-red-blue), forming an *association* (green box). Moreover, Fig. 1-right shows six associations of local groupings each depicted by a different color, and each containing several local groupings (bold color) separated by short time gaps (light color). Note that all local groupings and associations have different lengths and member instances. Such local groupings and associations can be of high utility in various application domains, including retail monitoring [11] or stock price prediction [10]. More concretely:

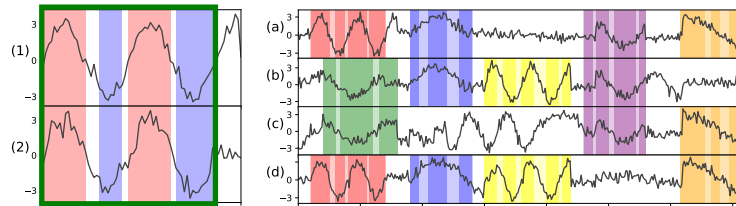


Fig. 1: An example of locally similar time series with four local groupings (left) and six associations (right) each containing several local groupings separated by short time gaps indicated by a lighter color.

- **Retail monitoring:** Product sales may follow different patterns over time, while a particular set of products can show a common local trend, e.g., high sales over the Christmas week, forming a “high sales” local grouping. After Christmas some sales in this grouping may drop, while some may still be maintained. This results in the “high sales” grouping to continue expanding over time and a new “low sales” grouping to be formed. If multiple local sales trends are shared by the same set of products, e.g., high sales over Christmas (1st local trend), followed by low sales in Feb-March (2nd local trend) and high sales over Easter (3rd local trend), they form an *association*.
- **Investing portfolio management:** Sets of stocks exhibit similar fluctuations from time to time. Local groupings represent stocks with similar fluctuation trends over fixed time periods. Moreover, at some time point some stocks from a local grouping may start showing different fluctuations and are, hence, placed into another grouping. Finally, *associations* are formed by growth stocks or sectors following consistently common fluctuation patterns, e.g., rising and dropping concurrently over the same time periods, and can be used for improving the portfolio over time or suggesting new portfolios.

In this paper, we study two problems: given a set of time series instances, we want to identify (1) local groupings of high similarity with maximized time span and number of member instances, (2) associations of maximum number of common local groupings and number of member instances. One solution is clustering (e.g., kmeans under the Euclidean distance) over a fixed-length sliding window. Nonetheless, such approach has two limitations: (1) noise or outliers can distort distance values, (2) as local groupings continuously evolve based on the similarity in each grouping, some instances in one grouping may become more similar to instances in another grouping, hence swapping groupings. As a result, detecting local groupings becomes even harder since groupings change over time both in length (i.e., time duration) and size (i.e., number of member instances).

One way to mitigate the first limitation is to resort to *temporal abstractions*, widely used to for time series summarization [19, 14]. A temporal abstraction compresses a time series by converting its original values into a set of *event labels* that are no longer sensitive to noise or outliers, and are no longer distorted by minor fluctuations, thus they tend to favor the formation of patterns over time.

For the second limitation, longest common prefix mining [18] can be applied for finding local groupings of varying length. However, it ignores the occurrences of smaller ones and may eventually miss many local groupings. Furthermore, it considers only the ordering of events, hence failing to localize these patterns in time. An alternative way is semigeometric tiling [9], a technique for finding local patterns in binary matrices spanning different ranges without constraining the number of neighbors or a time range. However, it only handles binary matrices and does not include any principled strategy to handle real (original time series) or categorical (abstraction) values. One way to apply semigeometric tiling is to binarize the original time series values by converting them to 1 if the original value is greater than, e.g., the mean of the time series values. Nonetheless, such solution is unable to capture all granularity levels in the data and the formed groupings will be sparse. In addition, this solution cannot identify associations.

1.1 Related work

There have been several attempts to cluster time series by employing different distance functions [13, 6], exploiting temporal features of high utility (e.g., shapelets [15], temporal abstractions [20]), or hybrid solutions [3]. However, they focus only on finding global groupings [2]. Subsequence clustering finds clusters of subsequences within a single time series, but not across different time series [22]. The problem of finding patterns from subsets of a dataset has been addressed in diverse ways including segmentation [8], bi-clustering [5]. However, all previous attempts have focused on finding common patterns between data examples without placing any constraints to the ordering of the features (i.e., both data rows and columns can be re-ordered), making it infeasible to be applied to time series. Column-coherent bi-clustering [12] clusters time series keeping their column order, but the problem formulation and its solutions are still not free from placing specific constraints on the column order. Although semigeometric tiling [9] imposes column order to the problem, it assumes a binary data representation and suffers from generating small tiles that are not practically useful for identifying local groupings of maximum time span. Longest common prefix mining [18] can be applied to extracting temporal patterns of time series; nonetheless, it is not directly applicable to our problem setup as it fails to detect concurrent patterns, since it only focuses on the longest patterns per time series, hence missing many local patterns shared by time series instances. Maximum correlation algorithms [16, 17] find the local segment with the maximum correlation between pairs of time series under a minimum subsequence length threshold. This setup is orthogonal to ours since we are interested in finding all local groupings with the maximum number of instances and time span under a similarity measure. In contrast to maximum correlation algorithms, local maximum correlation algorithms [21] identify the longest gapped time interval between two time series with the maximum correlation. This outputs the pair of regions with the highest correlation across the two time series. This problem differs from the problem studied in this paper, since we aim to find groupings of time series subsequences with high similarity over the same time period. Time

series motif discovery [4] aims to find motifs, i.e., repetitive patterns in one time series. However, this paper focuses on detecting different time ranges where high similarity of time series instances can be detected, thus creating a grouping that is not repetitive throughout time, unlike motifs. Applying motif discovery across time series instances over a fixed-length sliding window would identify some local groupings, but it would fail to find local groupings of variable time ranges.

1.2 Contributions

To the best of our knowledge, none of the existing formulations and solutions is directly applicable and comparable to our problem, since we aim to find all maximized time spans of groupings of locally similar time series without a specific time range as a constraint. Our main contributions include:

- **Novelty.** We propose **Z-Grouping**, an effective algorithm for finding *local groupings* of time series with high similarity and their *associations* in four steps by: (a) exploiting the notion of *temporal abstraction*, (b) generating local groupings based on the abstraction labels, (c) generating associations using a *grouping-instance matrix*, (d) validating the local groupings and associations against predefined global groupings.
- **Effectiveness.** We benchmark the effectiveness of **Z-Grouping** against four competitors on one synthetic and three real-world datasets. Effectiveness is measured in terms of the ability of extracted local groupings to identify highly similar local regions in unseen instances. **Z-Grouping** achieves lower mean squared error (MSE) up to 59.2% compared to the four competitors on our synthetic dataset, and up to 44.3% on three real-world datasets.
- **Generalizability.** We additionally benchmark **Z-Grouping** on 128 UCR time series datasets to demonstrate its ability to find valid local groupings with lower errors than the four competitors.

2 Problem Definition

Let $\mathbf{t} = \{t_1, \dots, t_m\}$ be a *time series instance* of length $|\mathbf{t}| = m$. A *time series collection* $\mathcal{T} = \{\mathbf{t}_1, \dots, \mathbf{t}_n\}$ is a collection of $|\mathcal{T}| = n$ time series instances.

Definition 1. (Event sequence) An event sequence $\mathbf{e} = \{e_1, \dots, e_l\}$ of length $|\mathbf{e}| = l$ is a collection of event labels, with $\forall i \leq n, e_i \in \Sigma$, where Σ is a set of discrete event labels of size λ .

Definition 2. (Temporal abstraction) A temporal abstraction of a time series instance \mathbf{t} is an event sequence \mathbf{e} obtained by applying a mapping function f to \mathbf{t} , such that $\mathbf{e} = f(\mathbf{t})$, with $\mathbf{e} \in \Sigma^{|\mathbf{e}|}$ and $|\mathbf{t}| = |\mathbf{e}|$.

Function f can be any time series summarization technique, such as discrete Fourier transform (DFT) [20] or symbolic aggregate approximation (SAX) [14]. For example, given $\mathbf{t} = \{1, 3, 3, 4, 5\}$, f converts \mathbf{t} to an event sequence $\mathbf{e} = \{a, b, b, c, c\}$ with $\lambda = 3$. Applying temporal abstraction to a time series collection \mathcal{T} results in an $n \times m$ event sequence matrix $M = \{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ where $\mathbf{e}_i = f(\mathbf{t}_i)$.

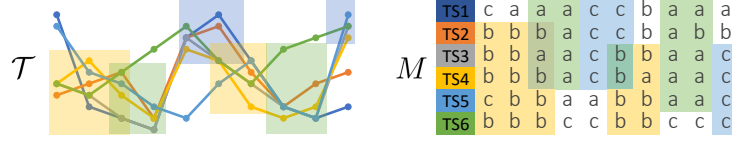


Fig. 2: An example of a time series collection \mathcal{T} of size 6 (left) and its event sequence matrix M (right) with $\lambda = 3$ with six candidates for local groupings.

Definition 3. (Local grouping) A local grouping $\rho = \{\mathbf{r}_\rho, \mathcal{T}_\rho\}$ is defined by a time range $\mathbf{r}_\rho = [r_s, r_e]$ and a subset of time series instances $\mathcal{T}_\rho \in \mathcal{T}$ that are similar over time range \mathbf{r}_ρ .

The time series instances in \mathcal{T}_ρ are called *member instances* of ρ , where $|\rho| = (\mathbf{r}_\rho.r_e - \mathbf{r}_\rho.r_s + 1) \times |\mathcal{T}_\rho|$ is the *size* of ρ . A set of local groupings is denoted as $\mathcal{R} = \{\rho_1, \dots, \rho_{|\mathcal{R}|}\}$.

Example. Fig. 2 depicts a time series collection \mathcal{T} with $|\mathcal{T}| = 6$, and an event sequence matrix M abstracting \mathcal{T} . In M we define six local groupings (colored areas) of sizes $|5 \times 3|$, $|4 \times 2|$, $|4 \times 2|$, $|4 \times 2|$, $|5 \times 2|$, and $|4 \times 1|$.

Definition 4. (Association) An association $\gamma = \{\mathbf{r}_\gamma, \mathcal{T}_\gamma, \mathcal{R}_\gamma\}$ is a merger of local groupings \mathcal{R}_γ with a time range spanning all $\rho \in \mathcal{R}_\gamma$, while containing the *intersection* of the member instances, i.e.:

$$\mathbf{r}_\gamma = \{\min(\mathbf{r}_\rho.r_s : \rho \in \mathcal{R}_\gamma), \max(\mathbf{r}_\rho.r_e : \rho \in \mathcal{R}_\gamma)\}, \mathcal{T}_\gamma = \{\mathcal{T}_{\rho_1} \cap \mathcal{T}_{\rho_2} \cap \dots \cap \mathcal{T}_{\rho_k}\}.$$

The *cardinality* of an association $|\gamma|$ is the number of comprised local groupings. The purpose of an association is to identify time series instances with maximum commonalities over time, by sharing many adjacent local groupings not easily detected globally, due to their negligible time spans or noise and outliers.

Based on the above definitions, we formulate our two problems.

Problem 1. (Detecting local groupings maximizing the time span and the number of member instances) Given \mathcal{T} and a threshold $\theta \in \mathbb{R}$, find \mathcal{R} with maximum instance size, maintaining the internal pairwise distance between the raw times series members below θ . That is, for each $\rho = \{\mathbf{r}_\rho, \mathcal{T}_\rho\} \in \mathcal{R}$:

$$\max |\rho|, \text{ s.t. } \forall \mathbf{t}_i, \mathbf{t}_j \in \mathcal{T}_\rho : \text{dist}(\mathbf{t}_i[\mathbf{r}_\rho.r_s : \mathbf{r}_\rho.r_e], \mathbf{t}_j[\mathbf{r}_\rho.r_s : \mathbf{r}_\rho.r_e]) \leq \theta.$$

Problem 2. (Detecting associations maximizing the number of common local groupings and member instances) Given \mathcal{T} and a threshold $\theta' \in \mathbb{R}$, find the set of associations $\Gamma = \{\gamma_1, \dots, \gamma_{|\Gamma|}\}$ of maximum cardinality and number of instance members, keeping the internal pairwise distance between the raw instance members below θ' . That is, for each $\gamma = \{\mathbf{r}_\gamma, \mathcal{T}_\gamma, \mathcal{R}_\gamma\} \in \Gamma$:

$$\max |\gamma| \times |\mathcal{T}_\gamma|, \text{ s.t. } \forall \mathbf{t}_i, \mathbf{t}_j \in \mathcal{T}_\gamma : \text{dist}(\mathbf{t}_i[\mathbf{r}_\gamma.r_s : \mathbf{r}_\gamma.r_e], \mathbf{t}_j[\mathbf{r}_\gamma.r_s : \mathbf{r}_\gamma.r_e]) \leq \theta'.$$

Constraining time to a specific range makes it difficult to spot local groupings that can be wider or narrower than the specified range. An exhaustive search with any distance function to optimize *Problems 1, 2* is computationally prohibitive.

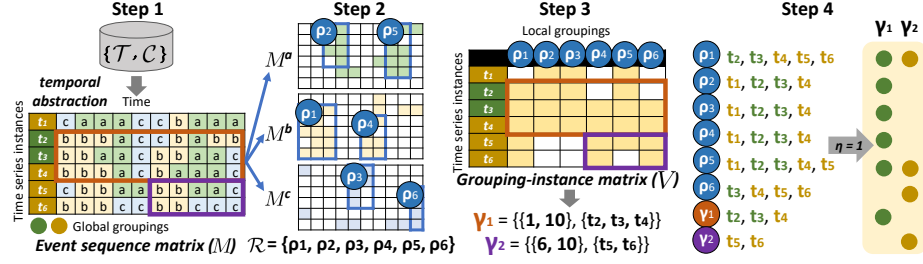


Fig. 3: An example of the four steps of Z-Grouping.

Algorithm 1: Z-Grouping

Input : \mathcal{T} , λ , \mathcal{G} : global groupings, α : purity, η : global grouping density
Result: \mathcal{R} : local groupings, Γ : associations, \mathcal{Z} : validity matrix

- 1 $\mathcal{R} \leftarrow \{\}$, $M \leftarrow \text{SAX}(\mathcal{T}, \lambda)$
- 2 **for** $M^k \in \text{generateChannels}(M, \lambda)$ **do**
- 3 **for** $\rho \in \text{generateLocalGroupingCandidates}(M^k)$ **do**
- 4 **if** $\frac{\sum_{t \in \mathcal{T}_\rho} (M^k[t, r_\rho \cdot r_s : r_\rho \cdot r_e])}{\sum_{t \in \mathcal{T}_\rho} |M^k[t, r_\rho \cdot r_s : r_\rho \cdot r_e]|} \geq \alpha$ **then** $\mathcal{R} \leftarrow \mathcal{R} \cup \rho$
- 5 $\Gamma \leftarrow \text{createAssociations}(\mathcal{T}, \mathcal{R}, \alpha)$
- 6 $\mathcal{Z} \leftarrow \text{validateGroupings}(\mathcal{R}, \mathcal{G}, \Gamma, \eta)$
- 7 **return** $\mathcal{R}, \Gamma, \mathcal{Z}$

3 The Z-Grouping Algorithm

Z-Grouping is a four-step algorithm for solving *Problems 1, 2*. The first step converts a time series collection into an event sequence matrix by applying a temporal abstraction function. The second step generates local groupings on the abstractions (*Problem 1*), while the third step identifies associations of local groupings using a *grouping-instance matrix* (*Problem 2*). Finally, local groupings and associations are validated against predefined global groupings. These steps, also outlined in Fig. 3 and Alg. 1, are described below.

3.1 Event sequence matrix generation

This step converts a collection \mathcal{T} of n time series of length m into an *event sequence matrix* M by applying a temporal abstraction function f . Without loss of generality, we employ SAX as our abstraction function, but different abstraction techniques can also be applied. SAX is applied to each time series instance $t_i \in \mathcal{T}$ resulting in an $n \times m$ event sequence matrix M , such that

$$M = \{\mathbf{e}_i \mid \forall i \leq n, \mathbf{e}_i = \text{SAX}(t_i, \lambda)\},$$

and λ corresponds to the event label size parameter of SAX (Alg. 1, line 1). Next, we split M into λ subsets, which we refer to as *event label channels*,

where each channel M^k corresponds to the k^{th} event label in Σ and records its occurrence in the form of binary values, indicating whether an event is assigned with the abstraction label of that channel (line 2); i.e., $\forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\} (M_{ij} = k \rightarrow M_{ij}^k = 1) \wedge (M_{ij} \neq k \rightarrow M_{ij}^k = 0)$. Note that the channels do not contain duplicate values, hence always satisfying $M_{ij}^k \neq M_{ij}^{k'}$ if $k \neq k'$.

Example. An example of this transformation is depicted in Fig. 3 (Step 1), where we obtain matrix M with event labels $\{a, b, c\}$, i.e., $\lambda = 3$. Next, we divide M into three event label channels $\{M^1, M^2, M^3\}$, one per event label.

3.2 Local grouping generation

The second step reduces *Problem 1* to finding λ separate sets of local groupings, one set per channel M^k . Since by definition the active events in each channel (indicated by 1s) share the same temporal abstraction label, they also fall in the same value range in their original representation. This implies that each channel can be used to extract local groupings, creating subsets of the channel by selecting time series instances and a time range. We apply semigeometric tiling [9] to each channel to generate candidates for local groupings (Alg. 1, line 3). The algorithm employs a priority queue to store the counts of active labels for every combination of time ranges $\mathbf{r} = \{r_s, r_e\}$ in each channel M^k . It then iteratively selects the time range with the maximum count from the priority queue and adds rows from the one with the highest number of active labels until a given threshold α is satisfied. α controls *label purity* of each subset, which we define as the proportion of active events in the subset of the channel (see Eq. 1).

More concretely, we create optimal subsets by optimizing the trade-off between two scores: *recall* (i.e., the number of active labels in the subset divided by the total number of active labels in M^k) and α . The algorithm keeps expanding the size of each subset by maximizing the recall while satisfying a given constraint of the subset. Since for each M^k we have a binary problem setup, recall is submodular [9], thus the generated subsets are at least $1 - \frac{1}{e}$ times the optimal recall. A local grouping ρ is defined by a candidate channel subset with a time range \mathbf{r}_ρ and a subset of time series instances \mathcal{T}_ρ . Each ρ can be added to \mathcal{R} when the following condition on α is satisfied:

$$accept(\rho, M^k) = \frac{\sum_{\mathbf{t} \in \mathcal{T}_\rho} (M^k[\mathbf{t}, \mathbf{r}_\rho.r_s : \mathbf{r}_\rho.r_e])}{\sum_{\mathbf{t} \in \mathcal{T}_\rho} |M^k[\mathbf{t}, \mathbf{r}_\rho.r_s : \mathbf{r}_\rho.r_e]|} \geq \alpha \quad (1)$$

where $M^k[\mathbf{t}, \mathbf{r}_\rho.r_s : \mathbf{r}_\rho.r_e]$ corresponds to the subset of M^k that matches time series \mathbf{t} within \mathbf{r}_ρ , and *accept* is a function that computes the proportion of active labels in ρ on its corresponding channel M^k . We keep the continuous order of time, but we can include any time series instances to \mathcal{T}_ρ regardless of their order.

Next, we proceed by extracting a set of local groupings from each channel and store all local groupings in a list \mathcal{R} , which is used in the next steps (line 4).

Example. Fig. 3 (Step 2) shows six local grouping candidates with $\alpha = 0.75$ allowing 25% of impurity in each candidate. Using Eq. 1, ρ_1 , ρ_2 , and ρ_3 return $0.75 \geq \alpha$, so they are added to \mathcal{R} , while ρ_6 returns 1 and is also added.

Algorithm 2: createAssociations

Input : $\mathcal{T}, \mathcal{R}, \alpha$

- 1 $\Gamma = \{\}, V \leftarrow \mathbb{0} \|\mathcal{T}\| \times |\mathcal{R}|, \mathcal{R} \leftarrow \text{sort}(\mathcal{R}, \text{by}=\{\mathbf{r}_\rho.r_s : \forall \rho \in \mathcal{R}\})$
- 2 $\forall j \leq |\mathcal{R}| \forall i \leq |\mathcal{T}_{\rho_j}| (\mathbf{t}_i \in \mathcal{T}_{\rho_j} \rightarrow V_{ij} = 1) \wedge (\mathbf{t}_i \notin \mathcal{T}_{\rho_j} \rightarrow V_{ij} = 0)$
- 3 **for** $\bar{\gamma} = \{\mathbf{r}_{\bar{\gamma}}, \mathcal{T}_{\bar{\gamma}}\} \in \text{generateAssociationCandidates}(V)$ **do**
- 4 **if** $\frac{\sum_{t \in \mathcal{T}_{\bar{\gamma}}} (V[t, \mathbf{r}_{\bar{\gamma}}.r_s : \mathbf{r}_{\bar{\gamma}}.r_e])}{\sum_{t \in \mathcal{T}_{\bar{\gamma}}} |V[t, \mathbf{r}_{\bar{\gamma}}.r_s : \mathbf{r}_{\bar{\gamma}}.r_e]|} \geq \alpha$ **then**
- 5 $r_s^{min} \leftarrow \min(\{\mathbf{r}_{\rho_j}.r_s : \forall j \in \mathbf{r}_{\bar{\gamma}}\})$
- 6 $r_e^{max} \leftarrow \max(\{\mathbf{r}_{\rho_j}.r_e : \forall j \in \mathbf{r}_{\bar{\gamma}}\})$
- 7 $\Gamma \leftarrow \Gamma \cup \{ (r_s^{min}, r_e^{max}), \mathcal{T}_{\bar{\gamma}}, \{\rho_j : \forall j \in \mathbf{r}_{\bar{\gamma}}\} \}$
- 8 **return** Γ

3.3 Association generation

This step solves *Problem 2* by reducing it to the problem of associating adjacent local groupings represented by time-ordered local groupings. An association of local groupings is generated following Def. 4 requiring minimized internal pairwise distances between the time series instances, while expanding cardinality, i.e., the number of local groupings included in the association. Computing all pairwise distances of all possible instances over time would be computationally prohibitive. We solve this problem by defining a *grouping-instance matrix* V , i.e., a binary matrix of size $|\mathcal{T}| \times |\mathcal{R}|$ recording the memberships of time series instances to each local grouping, where the columns are sorted by start time of each local grouping to keep their temporal order (Alg. 2, lines 1-2).

Matrix V allows us to easily find consecutive local groupings even if they are not continuous (i.e., gaps between them are allowed) in time. We first generate local groupings of V as candidate associations, each denoted by $\bar{\gamma} = \{\mathbf{r}_{\bar{\gamma}}, \mathcal{T}_{\bar{\gamma}}\}$ (line 3). Using Eq. 1 with each candidate $\bar{\gamma}$, we accept the candidate if $\text{accept}(\bar{\gamma}, V) \geq \alpha$, by checking which instances belong to which local groupings, with α being the degree of proximity between local groupings in $\bar{\gamma}$ (line 4). Then we create an association γ by extracting the minimum start and maximum end time from the involved local groupings in the candidate $\bar{\gamma}$, alongside the time series instances sharing those groupings (lines 5-7). In this way, local groupings are merged into the association, if they are consecutive in V . Depending on α , non-consecutive but close local groupings in V can also be added. Since associations maintain the longest time span (Def. 4), time gaps are also covered by the associations. As the main goal of associations is to find contiguous groupings of locally similar time series instances, we only store the intersecting instances; thus an association contains fewer instances than merged local groupings.

Example. In Fig. 3 (Step 3), six local groupings are generated in Step 2. Using these groupings and their member instances, we create V , marking if the grouping contains the corresponding members. We find two associations γ_1 with range $[1, 10]$ and γ_2 with $[6, 10]$. Each association only contains the intersect-

ing instances, so γ_1 contains $\{\mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4\}$, excluding \mathbf{t}_5 and \mathbf{t}_6 . Associations can contain non-consecutive groupings based on α , e.g., ρ_3, ρ_5 in γ_1 , and ρ_4, ρ_6 in γ_2 .

3.4 Validation of local groupings

In this last step, we validate the obtained local groupings by assuming a set of global groupings on the same time series collection \mathcal{T} , obtained either by time series clustering or provided by a domain expert. Hence, each time series $\mathbf{t} \in \mathcal{T}$ can belong to a global grouping as well as to multiple local groupings. Our goal is to assess how related the global groupings are to local groupings and how can local groupings help us assess the similarity of these global groupings.

Let $\mathcal{G} = \{g_1, \dots, g_x\}$ be a list of global groupings and consider a local grouping ρ , with its member instances \mathcal{T}_ρ . For each $\mathbf{t} \in \mathcal{T}_\rho$, we can extract the corresponding global grouping g and its member instances \mathcal{T}_g , hence $\mathbf{t} \in \mathcal{T}_\rho \cap \mathcal{T}_g$. If the majority of \mathcal{T}_ρ also belong to \mathcal{T}_g , we can assume such global grouping follows a similarity pattern of ρ over the time span \mathbf{r}_ρ of the grouping. However, some instances in \mathcal{T}_ρ may belong to different global groupings, and all these groupings may not always follow the similarity pattern of ρ , as small number of global grouping instances can be included by chance. Therefore, it is important to check the validity of a local grouping for each global grouping having its members in \mathcal{T}_ρ . This is achieved by a *validity score* δ for g and ρ calculated as:

$$\delta_\rho^{g,\eta} = \lceil s_\rho \cdot s_g \cdot \eta \rceil, \text{ where } s_\rho = \frac{|\mathcal{T}_\rho|}{|\mathcal{T}|}, \quad s_g = |\mathcal{T}_g|.$$

A *global grouping density* parameter η controls the required proportion of the member instances of g in \mathcal{T}_ρ for ρ to be valid for g . If $\eta = 1$, ρ is valid when the proportion of $\mathcal{T}_g \cap \mathcal{T}_\rho$ in \mathcal{T}_ρ is equal to the proportion of \mathcal{T}_g in \mathcal{T} . For example, if $|\mathcal{T}| = 100$, $|\mathcal{T}_\rho| = 10$, and $|\mathcal{T}_g| = 90$, we need at least 9 members satisfying $\mathbf{t} \in \mathcal{T}_\rho \cap \mathcal{T}_g$ to accept ρ for g . If $\eta = 0$, we accept all $\{\rho : \forall \mathbf{t} \in \mathcal{T}_\rho, \exists \mathbf{t} \in \mathcal{T}_g\}$ since δ becomes 0. If $\eta = 2$, the same calculation of δ would require at least 18 members. Using δ , we create a *validity matrix* \mathcal{Z} of size $|\mathcal{G}| \times |\mathcal{R}|$ to record the validity between local and global groupings, where $\forall i \leq |\mathcal{G}| \forall j \leq |\mathcal{R}| (\sum_{\mathbf{t} \in \mathcal{T}_{\rho_j}} \llbracket \mathbf{t} \in \mathcal{T}_{g_i} \rrbracket \geq \delta_{\rho_j}^{g_i, \eta} \rightarrow \mathcal{Z}_{ij} = 1) \wedge (\sum_{\mathbf{t} \in \mathcal{T}_{\rho_j}} \llbracket \mathbf{t} \in \mathcal{T}_{g_i} \rrbracket < \delta_{\rho_j}^{g_i, \eta} \rightarrow \mathcal{Z}_{ij} = 0)$.

Example. Fig. 3 (Step 4) shows the valid groupings with $\eta = 1$. We first calculate δ for each $\rho_i \in \mathcal{R}$ and $g_j \in \mathcal{G}$. For groupings of size 4 ($\{\rho_2, \rho_3, \rho_4, \rho_6\}$), we need at least $\lceil \frac{4}{6} \cdot 2 \cdot 1 \rceil = 2$ instances of g_1 and $\lceil \frac{4}{6} \cdot 4 \cdot 1 \rceil = 3$ of g_2 to be valid. Since ρ_2, ρ_3, ρ_4 contain two instances of g_1 and two of g_2 , they are only valid for g_1 . On the other hand, ρ_6 has three of g_2 and one of g_1 so it is only valid for g_2 .

3.5 Complexity of Z-Grouping

Given an $n \times m$ time series collection \mathcal{T} and λ , the time complexity for creating the groupings for one event label follows the complexity of semigeometric tiling $O(m^2 n \log n)$ [9], leading to $O(\lambda m^2 n \log n)$ as the total complexity of Steps 1-2. For creating associations (Step 3), since, in the extreme case, the number of local

Table 1: Summary of the datasets used in this paper.

Dataset	$ \mathcal{T} $	$ \mathbf{t} $	$ \mathcal{G} $	avg($ \mathcal{T}_g $)	max($ \mathcal{T}_g $)	min($ \mathcal{T}_g $)
SYNTHETIC	1,000	365	20	50	50	50
GARMENT	3,963	365	50	79.26	282	29
STOCK	505	503	11	45.91	84	3
COVID	191	618	6	31.83	53	11

groupings can be the same as the count of all data points ($|\mathcal{R}| = mn$), the worst case complexity becomes $O(m^2n^3\log n)$. The validation (Step 4) takes $O(n^2m)$ since **Z-Grouping** checks every grouping and all time series instances for each grouping in the worst case. However, in practice, we can relax the complexity of Steps 1-3 by limiting the number of local groupings and associations **Z-Grouping** finds. For example, the complexity can be reduced to $O(kn\log n)$ by picking top k local groupings and associations, as they are chosen in descending order. Extremely small local groupings which have either few time points or few member instances may have no meaningful information and can be ignored. It may also be possible to give other constraints such as minimum length or size.

4 Experiments

4.1 Setup

Datasets We use three real-world datasets from (1) retail industry (**GARMENT**), (2) stock market (**STOCK**), and (3) COVID-19 epidemics (**COVID**). A summary of the properties of these datasets is provided in Table 1, while a detailed description and sources can be found in our repository [1]. We also generated a synthetic dataset (**SYNTHETIC**) for extensive parameter investigation. It contains 1,000 instances and 20 global groupings that resemble the presence of local similarity. Each global grouping comprises a sinusoidal pattern with a different frequency and amplitude. The number of inserted patterns is smaller than the number of global groupings, hence some of them can share the same pattern, which can be detected as local groupings. To simulate a realistic scenario, noise and outliers are imputed. We also tested on 128 UCR datasets [7], excluding the cases where at least one algorithm cannot find any valid groupings (17 cases) and where the dataset length is shorter than the smallest window size parameter (6 cases). Our datasets and code including synthetic data generator are available online [1].

Competitors While, to the best of our knowledge, there exists no direct competitor for the problem solved by **Z-Grouping**, we benchmark on the closest approach, i.e., semigeometric tiling and three additional baselines.

- **Semigeometric**: We employ a simple modification of semigeometric tiling [9]. First, we generate a binary matrix with its values set to 1 for the standardized time series values above the boundary found by SAX with $\lambda = 2$. Second, to make it directly comparable to **Z-Grouping**, we apply α (Eq. 1).

- **kmeans**: We divide each instance \mathbf{t} into r partitions of time range w , such that $w \cdot r = |\mathbf{t}|$. For each partition, we apply **kmeans**; the resulting k clusters per partition correspond to the local groupings.
- **kmeans-FLEX**: We define a flexible version of **kmeans** with a sliding window of width w and slide step $\frac{w}{2}$ as a brute-force solution to finding local groupings of multiple fixed window lengths. We repeat for different values of w and k , and for each window, clusters with an average silhouette above a cutoff threshold s are accepted as groupings.
- **kNN**: Using the same partitioning approach as **kmeans**, and given a global grouping (see Experiment protocol), for each partition, we identify k instances belonging to that global grouping. For each instance, we apply **kNN**, retrieving the k nearest instances under the Euclidean distance resulting in k^2 samples, which correspond to a local grouping.

Note that **kNN**, **kmeans**, and **kmeans-FLEX** are also tested on the SAX abstracted space with $\lambda = 5$ to directly compare to **Z-Grouping** running on the abstracted space to detect maximized time spans. The results on the raw time series are in favor of the competitors since they can find more accurate neighbors but still only find fixed-length groupings. Also, as we are after local similarity synced in time, we do not explore elastic measures such as dynamic time warping.

Experiment protocol Assuming a set of predefined global groupings \mathcal{G} on a given time series collection \mathcal{T} , we divide \mathcal{T} into a training set \mathcal{T}^{train} and a test set \mathcal{T}^{test} , and create local groupings on \mathcal{T}^{train} . Our goal is to investigate if the local groupings detected by each algorithm can identify potential local similarity on unseen instances. More concretely, for each unseen sample $\mathbf{t}_i \in \mathcal{T}^{test}$, we retrieve its global grouping \bar{g} . Our assumption is that the values of \mathbf{t}_i are unseen, so the only information available for choosing valid local groupings is \bar{g} . For example, this simulates the scenario where we have a new product ready for market and we would like to identify potential local similarity patterns of its upcoming sales with existing products. In this case, \bar{g} corresponds to a predefined product type and its features (e.g., color, size, material type).

Hence, for each unseen sample $\mathbf{t}_i \in \mathcal{T}^{test}$ and its corresponding global grouping \bar{g} , our evaluation is as follows. For **kmeans**, we choose the cluster with the highest number of instances of \bar{g} for each time window. For **kNN** we employ the k^2 chosen samples. For **Z-Grouping** and **Semigeometric**, we choose the groupings based on δ (see Sec. 3.4). Next, for each ρ , we extract all global groupings $\{g : \forall \mathbf{t}_j \in \mathcal{T}_\rho, \exists \mathbf{t}_j \in \mathcal{T}_g\}$ except for the target global grouping \bar{g} . Then we calculate the errors (MSE and mean absolute error (MAE)) over the active time range \mathbf{r}_ρ between the test time series $\mathbf{t}_i[\mathbf{r}_{\rho.r_s} : \mathbf{r}_{\rho.r_e}]$ and $\{\mathbf{t}_j[\mathbf{r}_{\rho.r_s} : \mathbf{r}_{\rho.r_e}] \mid \forall \mathbf{t}_j \in \mathcal{T}^{train} \wedge \mathbf{t}_j \notin \mathcal{T}_{\bar{g}}\}$, i.e., all the instances in the local grouping that belong to the chosen global groupings. This way we explore if these global groupings show local similarity and benchmark the robustness of each algorithm to randomness and noise. For **Z-Grouping**, **Semigeometric**, and **kmeans-FLEX**, we report *coverage*, i.e., the fraction of time series covered by the groupings. **kmeans** and **kNN** have 100% coverage since they always find similar instances based on the distance

Table 2: Average test errors of the algorithms on SYNTHETIC (CV: Coverage (%)).

α	Semigeometric									Z-Grouping ($\lambda = 3$)									
	$\eta = 1$			$\eta = 1.5$			$\eta = 2$			$\eta = 1$			$\eta = 1.5$			$\eta = 2$			
	MSE	MAE	CV	MSE	MAE	CV	MSE	MAE	CV	MSE	MAE	CV	MSE	MAE	CV	MSE	MAE	CV	
0.8	1.30	0.79	72	1.28	0.78	41	1.24	0.74	15	1.22	0.75	88	1.08	0.68	55	1.09	0.68	29	
0.9	1.18	0.73	61	1.16	0.72	44	1.14	0.72	21	1.08	0.69	68	1.09	0.69	43	0.97	0.63	30	
1.0	1.11	0.71	40	1.16	0.72	22	1.09	0.68	7	0.97	0.73	40	0.95	0.62	24	0.89	0.61	15	
α	Z-Grouping ($\lambda = 5$)									Z-Grouping ($\lambda = 10$)									
0.8	1.00	0.65	45	0.93	0.62	27	0.85	0.57	15	0.88	0.56	20	0.88	0.56	12	0.78	0.52	10	
0.9	0.95	0.60	30	0.94	0.59	20	0.87	0.55	10	0.84	0.56	14	0.86	0.55	10	0.81	0.52	5	
1.0	0.87	0.57	21	0.87	0.56	15	0.77	0.52	8	0.73	0.51	8	0.89	0.56	5	0.97	0.61	2	
w	kNN-SAX									kNN									
	$k = 3$			$k = 5$			$k = 10$			$k = 3$			$k = 5$			$k = 10$			
30	1.30	0.77	-	1.41	0.83	-	1.59	0.92	-	1.31	0.77	-	1.40	0.83	-	1.58	0.91	-	
60	1.23	0.74	-	1.34	0.80	-	1.54	0.90	-	1.22	0.73	-	1.33	0.80	-	1.53	0.90	-	
180	1.12	0.68	-	1.22	0.72	-	1.40	0.82	-	1.10	0.67	-	1.20	0.72	-	1.40	0.83	-	
w	kmeans-SAX									kmeans									
30	1.49	0.88	-	1.51	0.89	-	1.51	0.88	-	1.51	0.89	-	1.53	0.90	-	1.53	0.90	-	
60	1.59	0.93	-	1.59	0.93	-	1.58	0.92	-	1.60	0.93	-	1.60	0.93	-	1.59	0.93	-	
180	1.57	0.91	-	1.56	0.91	-	1.55	0.90	-	1.58	0.92	-	1.58	0.91	-	1.57	0.91	-	
kmeans-FLEX-SAX				s = 0.1	1.44			0.84	100	kmeans-FLEX				s = 0.1	1.45			0.85	100
				s = 0.2	1.48			0.85	100		s = 0.2	1.47			0.85	100			
				s = 0.3	1.56			0.93	100		s = 0.3	1.53			0.89	100			
				s = 0.4	1.79			1.01	36		s = 0.4	1.79			1.01	36			

function. The ability of Z-Grouping can be shown by lower errors than the competitors since this confirms Z-Grouping can find groupings of different lengths showing better local similarity than fixed-size clusters.

4.2 Results

Results on the synthetic dataset For Z-Grouping, we test $\alpha = \{0.8, 0.9, 1\}$, $\lambda = \{3, 5, 10\}$, and $\eta = \{1, 1.5, 2\}$. For Semigeometric, we apply the same α and η . For kmeans and kNN, we test different time ranges of $w = \{30, 60, 180\}$ and $k = \{3, 5, 10\}$. For kmeans-FLEX, we apply a silhouette cutoff from 0.1 until it fails to detect any valid groupings. All results are 10-fold cross-validated.

Table 2 shows the average test errors of Z-Grouping and its four competitors. Z-Grouping always succeeds in finding valid local groupings of low errors. Z-Grouping’s lowest MSE is at least 33.0% lower than the competitors’ lowest MSE (23.9% in MAE), and up to 59.2% lower (49.5% in MAE) than the worst score of the competitors. Semigeometric shows lower errors in general than kmeans and kNN, but it still has higher errors than Z-Grouping while covering smaller areas in the same parameter setting, as it suffers from its lack of representation power with a strong binary assumption, outperformed by Z-Grouping to a great extent up to 33.0% (25.0% in MAE). Except for two cases ($\alpha = \{0.8, 0.9\}, \eta = 1$), Semigeometric even fails to cover more than 50% of the

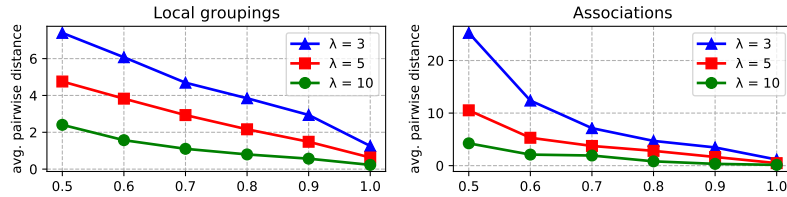


Fig. 4: Relation between α and the average pairwise distance of time series instances in local groupings (left) and associations (right) on **SYNTHETIC**.

dataset. This confirms the effectiveness of **Z-Grouping** in finding local groupings compared to its binary competitor; **Z-Grouping** with only one more abstraction label ($\lambda = 3$) achieves substantially better results than **Semigeometric** achieving from 6.0% to 18.3% lower MSEs while covering larger areas.

Since our synthetic data is designed to have clear local groupings, distance-based methods do not show noticeable differences in errors on both abstracted and original spaces. **kNN** achieves its best score with $\{w : 180, k : 3\}$ in both spaces, but its MSE is still 50.7% higher than **Z-Grouping**'s best error (31.4% in MAE). It achieves lower errors with smaller window sizes, as it is easier to identify local groupings within the window. **kmeans** does not show remarkable differences with various parameter settings; it is generally worse than its competitors. **Kmeans-FLEX** has its lowest MSE being only 3.4% lower than the lowest error of **kmeans** (4.6% in MAE) but 97.3% higher than the lowest error of **Z-Grouping** (64.7% in MAE). This means giving a few options for the length of local groupings can be worse than fixed-size search, as well as far worse than **Z-Grouping**'s ability to detect a maximized time range for local groupings. None of the competitors detect groupings of similar quality to **Z-Grouping**, which means an exhaustive search is required to find meaningful groupings, while **Z-Grouping** can find more flexible groupings with more valid local similarity.

Effect of the parameters. All three parameters (λ , α , and η) of **Z-Grouping** control the trade-off between coverage and error. Since the abstraction label size is directly related to the sparsity of the channels, higher λ can lead to lower coverage, making **Z-Grouping** difficult to find the same event labels adjacent to each other, while detecting better local groupings with lower errors. Higher α leads to purer groupings allowing a smaller number of different event labels. If we increase flexibility with small α , the error increases due to the formation of many impure groupings. Higher η requires more samples in the local grouping for validity, leading to smaller number of groupings. This results in lower error scores but reduces coverage too. Associations of the groupings help increase coverage by filling the gaps created by high values of the parameters. However, under the highest parameter values, the algorithm loses its ability to grow over a substantial area, only showing less than 10% coverage and the error also gets higher since there is no meaningful amount of data points to compare to.

Table 3: Best average test errors of the algorithms on three real-world datasets.

Datasets	Z-Grouping			Semigeometric			kNN-SAX			kNN		
	MSE	MAE	CV	MSE	MAE	CV	MSE	MAE	CV	MSE	MAE	CV
GARMENT	0.83	0.65	88	1.76	0.96	67	1.64	0.92	100	1.64	0.92	100
STOCK	0.99	0.74	77	1.49	0.84	70	1.21	0.83	100	1.20	0.83	100
COVID	0.84	0.49	40	2.17	0.92	74	1.37	0.70	100	1.37	0.71	100
Datasets	kmeans-SAX			kmeans			kmeans-FLEX			kmeans-FLEX-SAX		
GARMENT	1.65	0.90	100	1.67	0.92	100	1.49	0.87	100	1.51	0.92	100
STOCK	1.37	0.89	100	1.37	0.89	100	1.49	0.92	100	1.50	0.92	100
COVID	1.49	0.73	100	1.49	0.73	100	0.99	0.55	38	0.99	0.55	37

Relationship between θ, θ' and α . Z-Grouping solves *Problems 1, 2* by transforming θ, θ' to the purity parameter α while maximizing the same space, assuming that θ, θ' are dependent on our choice of α . Hence, it is important to validate the relation between θ and α . Fig. 4 shows the relation between α and the average pairwise distance of time series instances in the estimated groupings (left) and the associations (right) on SYNTHETIC. This confirms higher α leads to smaller θ , hence Z-Grouping approximately solves *Problems 1, 2*, while the actual thresholds (θ, θ') are dependent on α , and we maximize $|\rho_l|$ and $|\gamma_l| \times |\mathcal{T}_{\gamma_l}|$ keeping α . The same analysis on the real-world datasets is in the supplement (Sec. A).

Results on three real-world datasets Table 3 shows the average test errors of Z-Grouping and the competitors on three real-world datasets. We explore the same parameter settings as for the synthetic experiment. We report the best case in MSE covering more than 30% of the datasets, while Z-Grouping also gets lower errors with lower coverage. Overall, Z-Grouping achieves the best score on every dataset. On GARMENT and STOCK, Z-Grouping is a clear winner by having 44.3% lower MSE (25.2% in MAE) on GARMENT and 17.5% lower MSE (10.9% in MAE) on STOCK than the MSE under the best competitor, with at least 70% coverage. On COVID, Z-Grouping shows 15.2% lower MSE (11.0% in MAE) compared to the best of four competitors (kmeans-FLEX), but only covers 40% of the dataset as the COVID-19 patterns in one continent have not always been similar. It appears that the goal of finding consistent local similarity across continents has not been well met compared to the other two cases. Semigeometric performs the worst on every dataset, even outperformed by the baselines. Full results for each parameter setting are available in our repository [1].

Results on the UCR datasets We test Z-Grouping and its competitors on 128 UCR datasets using the parameter settings yielding average performance in our synthetic experiments. Since the UCR datasets contain general cases and do not always have clear local similarity, the experiment shows a different perspective from our synthetic and three real-world data experiments as described

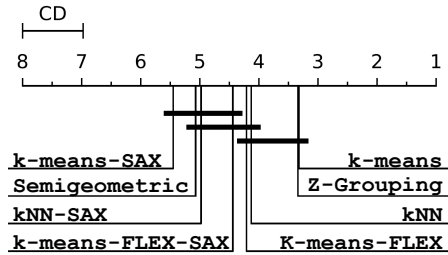


Fig. 5: Nemenyi post hoc test on the 128 UCR datasets.

Table 4: Average MSE rankings and wins/loses on the 128 UCR datasets.

Algorithms	Avg.rank	Win	Lose
Z-Grouping	3.34	48	9
Semigeometric	5.45	9	45
kNN	4.13	8	0
kNN-SAX	5.07	0	8
kmeans	3.32	20	1
kmeans-SAX	4.98	6	22
kmeans-FLEX	4.21	8	5
kmeans-FLEX-SAX	4.44	9	4

in Fig. 5 and Table 4. First, the solutions on the abstracted space show significantly worse results than the ones on the original space, while **Z-Grouping** still outperforms them. Second, **kmeans** performs well on the UCR datasets showing the lowest average rank in terms of MSE and MAE while it shows the worst performance in our synthetic experiment, due to some datasets entirely missing valid local groupings; this can be confirmed by **kmeans-FLEX** underperforming even though it is also the same distance-based solution.

Z-Grouping in itself requires temporal abstraction to maximize the time span of the local groupings and the associations given α , thus losing some of the original information. This might make the detection process harder when local groupings are not distinct in the dataset as some cases in the UCR datasets. However, **Z-Grouping** still succeeds in finding valid local groupings of variable time span from these cases. This can be confirmed by noticing that **Z-Grouping** wins in 48 cases and only loses in nine cases, while two competitors capable of searching for local groupings of varying lengths (**Semigeometric** and **kmeans-FLEX**) show significantly worse results. Our main state-of-the-art competitor (**Semigeometric**) loses in 45 cases, even underperforming the baselines and **Z-Grouping**.

While losing information due to SAX, **Z-Grouping** achieves statistically equivalent average rank to its competitors running on the original space, despite the fact that it is harder to identify local groupings in some of the UCR datasets. Moreover, this also means we show our effectiveness over the baselines, since the competitors are still limited to only finding the fixed area (i.e., window size) while **Z-Grouping** finds maximized length areas for the groupings with similar or lower errors. **Z-Grouping** loses (1) when there are completely no valid local groupings in the dataset, (2) when there are more than two different local similarities in the same period with enough support in one class, and (3) when the original values have only slight fluctuations, so the SAX space loses all this information and as a result distance-based clustering outperforms. Examples of losing cases can be found in the supplementary material (Sec. B).

5 Conclusion

We proposed **Z-Grouping**, a novel framework for detecting local groupings of locally similar time series and their associations. We benchmarked **Z-Grouping** on three real-world datasets and a synthetic dataset as well as 128 UCR datasets against four competitor methods. Our experiments showed that **Z-Grouping** could achieve lower error rates than its competitors while successfully retrieving local groupings without size constraints on time ranges, which is infeasible by using traditional methods. Future work includes exploring alternative temporal abstractions, applying global optimization to create the local groupings, studying multivariate time series, i.e., creating multidimensional groupings.

References

1. Z-Grouping repository, <https://github.com/zedshape/zgrouping/>
2. Aghabozorgi, S., Shirkhorshidi, A.S., Wah, T.Y.: Time-series clustering—a decade review. *ISJ* **53**, 16–38 (2015)
3. Aghabozorgi, S., Wah, T.Y.: Clustering of large time series datasets. *IDA* **18**(5), 793–817 (2014)
4. Alaei, S., Mercer, R., Kamgar, K., Keogh, E.: Time series motifs discovery under dtw allows more robust discovery of conserved structure. *DAMI* **35**(3), 863–910 (2021)
5. Cheng, Y., Church, G.: Biclustering of expression data. In: *ISMB*. vol. 8, pp. 93–103 (2000)
6. Cuturi, M., Blondel, M.: Soft-dtw: a differentiable loss function for time-series. In: *ICML*. pp. 894–903. PMLR (2017)
7. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The ucr time series archive. *JAS* **6**(6), 1293–1305 (2019)
8. Gionis, A., Mannila, H., Terzi, E.: Clustered segmentations. In: *TDM*. Citeseer (2004)
9. Henelius, A., Karlsson, I., Papapetrou, P., Ukkonen, A., Puolamäki, K.: Semigeometric tiling of event sequences. In: *ECML-PKDD*. pp. 329–344. Springer (2016)
10. Huang, C.F.: A hybrid stock selection model using genetic algorithms and support vector regression. *Applied Soft Computing* **12**(2), 807–818 (2012)
11. Jiang, Y., Liu, Y., Wang, H., Shang, J., Ding, S.: Online pricing with bundling and coupon discounts. *IJPR* **56**(5), 1773–1788 (2018)
12. Lee, J.H., Lee, Y.R., Jun, C.H.: A biclustering method for time series analysis. *IEMS* **9**(2), 131–140 (2010)
13. Li, H., Liu, J., Yang, Z., Liu, R.W., Wu, K., Wan, Y.: Adaptively constrained dynamic time warping for time series classification and clustering. *Information Sciences* **534**, 97–116 (2020)
14. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. *DMKD* **15**(2), 107–144 (2007)
15. Luo, L., Lv, S.: An accelerated u-shapelet time series clustering method with lsh. In: *Journal of Physics: Conference Series*. vol. 1631, pp. 12–77. IOP (2020)
16. Mueen, A., Hamooni, H., Estrada, T.: Time series join on subsequence correlation. In: *ICDM*. pp. 450–459. IEEE (2014)

17. Mueen, A., Nath, S., Liu, J.: Fast approximate correlation for massive time-series data. In: SIGMOD. pp. 171–182 (2010)
18. Raza, A., Kramer, S.: Accelerating pattern-based time series classification: a linear time and space string mining approach. KAIS **62**(3), 1113–1141 (2020)
19. Ruta, N., Sawada, N., McKeough, K., Behrisch, M., Beyer, J.: Sax navigator: Time series exploration through hierarchical clustering. In: VIS. pp. 236–240 (2019)
20. Schäfer, P.: The boss is concerned with time series classification in the presence of noise. DAMI **29**(6), 1505–1530 (2015)
21. Wu, J., Wang, Y., Wang, P., Pei, J., Wang, W.: Finding maximal significant linear representation between long time series. In: ICDM. pp. 1320–1325. IEEE (2018)
22. Zolhavarieh, S., Aghabozorgi, S., Teh, Y.W.: A review of subsequence time series clustering. The Scientific World Journal **2014** (2014)