This is an expanded version of the ICDM 2003 paper. We welcome feedback on this work.

Clustering of Time Series Subsequences is Meaningless: Implications for Previous and Future Research

Eamonn Keogh Jessica Lin Wagner Truppel

Computer Science & Engineering Department
University of California - Riverside
Riverside, CA 92521

{eamonn, jessica, wagner}@cs.ucr.edu

Abstract

Time series data is perhaps the most frequently encountered type of data examined by the data mining community. Clustering is perhaps the most frequently used data mining algorithm, being useful in it's own right as an exploratory technique, and also as a subroutine in more complex data mining algorithms such as rule discovery, indexing, summarization, anomaly detection, and classification. Given these two facts, it is hardly surprising that time series clustering has attracted much attention. The data to be clustered can be in one of two formats: many individual time series, or a single time series, from which individual time series are extracted with a sliding window. Given the recent explosion of interest in streaming data and online algorithms, the latter case has received much attention.

In this work we make a surprising claim. Clustering of streaming time series is completely meaningless. More concretely, clusters extracted from streaming time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random. While this constraint can be intuitively demonstrated with a simple illustration and is simple to prove, it has never appeared in the literature.

We can justify calling our claim surprising, since it invalidates the contribution of dozens of previously published papers. We will justify our claim with a theorem, illustrative examples, and a comprehensive set of experiments on reimplementations of previous work. Although the primary contribution of our work is to draw attention to the fact that an apparent solution to an important problem is incorrect and should no longer be used, we also introduce a novel method which, based on the concept of time series motifs, is able to meaningfully cluster some streaming time series datasets.

Keywords

Time Series, Data Mining, Clustering, Rule Discovery

1. Introduction

Time series data is perhaps the most commonly encountered kind of data explored by data miners [26, 35]. Clustering is perhaps the most frequently used data mining algorithm [14], being useful in it's own right as an exploratory technique, and as a subroutine in more complex data mining algorithms [3, 5]. Given these two facts, it is hardly surprising that time series data mining has attracted an extraordinary amount of attention [3, 7, 8, 9, 11, 12, 15, 16, 17, 18, 20, 21, 24, 25, 27, 28, 29, 30, 31, 32, 33, 36, 38, 40, 42, 45]. The work in this area can be broadly classified into two categories:

- **Whole Clustering**: The notion of clustering here is similar to that of conventional clustering of discrete objects. Given a set of individual time series data, the objective is to group similar time series into the same cluster.
- **Subsequence Clustering**: Given a single time series, individual time series (subsequences) are extracted with a sliding window. Clustering is then performed on the extracted time series.

Subsequence clustering is commonly used as a subroutine in many other algorithms, including rule discovery [9, 11, 15, 16, 17, 20, 21, 30, 32, 36, 42, 45], indexing [27, 33], classification [7, 8], prediction [37, 40], and anomaly detection [45]. For clarity, we will refer to this type of clustering as STS (Subsequence Time Series) clustering.

In this work we make a surprising claim. Clustering streaming time series is meaningless! More concretely, clusters extracted from streaming time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random.

Since we use the word "*meaningless*" many times in this paper, we will take the time to define this term. All useful algorithms (with the sole exception of random number generators) produce output that depends on the input. For example, a decision tree learner will yield very different outputs on, say, a credit worthiness domain, a drug classification domain, and a music domain. We call an algorithm "*meaningless*" if the output is independent of the input. As we prove in this paper, the output of STS clustering does not depend on input, and is therefore meaningless.

Our claim is surprising since it calls into question the contributions of dozens of papers. In fact, the existence of so much work based on STS clustering offers an obvious counter argument to our claim. It could be argued: "*Since many papers have been published which use time series subsequence clustering as a subroutine, and these papers produced successful results, time series subsequence clustering must be a meaningful operation.*"

We strongly feel that this is not the case. We believe that in all such cases the results are consistent with what one would expect from random cluster centers. We recognize that this is a strong assertion, so we will demonstrate our claim by reimplementing the most successful (i.e. the most referenced) examples of such work, and showing with exhaustive experiments that these contributions inherit the property of meaningless results from the STS clustering subroutine.

The rest of this paper is organized as follows. In Section 2 we will review the necessary background material on time series and clustering, then briefly review the body of research that uses STS clustering. In Section 3 we will show that STS clustering is meaningless with a series of simple intuitive experiments; then in Section 4 we will explain *why* STS clustering cannot produce useful results. In Section 5 we show that the many algorithms that use STS clustering as a subroutine produce results indistinguishable from random clusters. Since the main contribution of this paper may be considered "negative," we conclude in Section 6 with the demonstration of a simple algorithm that *can* find clusters in at least some trivial streaming datasets. This algorithm is not presented as the best way to find clusters in streaming time series; it is simply offered as an existence proof that such an algorithm exists, and to pave the way for future research.

2. Background Material

In order to frame our contribution in the proper context we begin with a review of the necessary background material.

2.1 Notation and Definitions

We begin with a definition of our data type of interest, time series:

Definition 1. *Time Series*: A time series $T = t_1,...,t_m$ is an ordered set of m real-valued variables.

Data miners are typically not interested in any of the global properties of a time series; rather, data miners confine their interest to subsections of the time series, called subsequences.

Definition 2. *Subsequence*: Given a time series T of length m, a subsequence C_p of T is a sampling of length $w < m$ of contiguous positions from T, that is, $C = t_p,...,t_{p+w-1}$ for $1 \le p \le m - w + 1$.

In this work we are interested in the case where all the subsequences are extracted, and then clustered. This is achieved by use of a sliding window.

Definition 3. *Sliding Windows*: Given a time series T of length m, and a user-defined subsequence length of w, a matrix S of all possible subsequences can be built by "sliding a window" across T and placing subsequence C_p in the p^{th} row of S. The size of matrix S is $(m - w + 1)$ by w.

Figure 1 summarizes all the above definitions and notations.

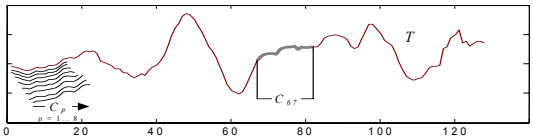


Figure 1. An illustration of the notation introduced in this section: a *time series T* of length 128, a *subsequence* of length w = 16, beginning at datapoint 67, and the first 8 subsequences extracted by a *sliding window*.

Note that while S contains exactly the same information as T, it requires significantly more storage space. This is typically not a problem, since, as we shall see in the next section, the limiting factor tends to be the CPU time for clustering.

2.2 Background on Clustering

One of the most widely used clustering approaches is hierarchical clustering, due to the great visualization power it offers [26, 29]. Hierarchical clustering produces a nested hierarchy of similar groups of objects, according to a pairwise distance matrix of the objects. One of the advantages of this method is its generality, since the user does not need to provide any parameters such as the number of clusters. However, its application is limited to only small datasets, due to its quadratic computational complexity. Table 1 outlines the basic hierarchical clustering algorithm.

Table 1: An outline of hierarchical clustering.

Algorithm Hierarchical Clustering	
1.	Calculate the distance between all objects. Store the results in a distance matrix.
2.	Search through the distance matrix and find the two most similar clusters/objects.
3.	Join the two clusters/objects to produce a cluster that now has at least 2 objects.
4.	Update the matrix by calculating the distances between this new cluster and all other clusters.
5.	Repeat step 2 until all cases are in one cluster.

A faster method to perform clustering is k-means [5]. The basic intuition behind k-means (and a more general class of clustering algorithms known as iterative refinement algorithms) is shown in Table 2:

Table 2: An outline of the k-means algorithm.

Algorithm *k-means*	
1.	Decide on a value for k.
2.	Initialize the k cluster centers (randomly, if necessary).
3.	Decide the class memberships of the N objects by assigning them to the nearest cluster center.
4.	Re-estimate the k cluster centers, by assuming the memberships found above are correct.
5.	If none of the N objects changed membership in the last iteration, exit. Otherwise goto 3.

The k-means algorithm for N objects has a complexity of $O(kNrD)$, with k the number of clusters specified by the user, r the number of iterations until convergence, and D the dimensionality of time series (in the case of STS clustering, D is the length of the sliding window, w). While the algorithm is perhaps the most commonly used clustering algorithm in the literature, it does have several shortcomings, including the fact that the number of clusters must be specified in advance [5, 14].

It is well understood that some types of high dimensional clustering may be meaningless. As noted by [1, 4], in high dimensions the very concept of nearest neighbor has little meaning, because the ratio of the distance to the nearest neighbor over the distance to the average neighbor rapidly approaches one as the dimensionality increases. However, time series, while often having high dimensionality, typically have a low intrinsic dimensionality [25], and can therefore be meaningful candidates for clustering.

2.3 Background on Time Series Data Mining

The last decade has seen an extraordinary interest in mining time series data, with at least one thousand papers on the subject [26]. Tasks addressed by the researchers include segmentation, indexing, clustering, classification, anomaly detection, rule discovery, and summarization.

Of the above, a significant fraction use streaming time series clustering as a subroutine. Below we will enumerate some representative examples.

- There has been much work on finding association rules in time series [9, 11, 15, 16, 20, 21, 30, 32, 26, 42, 45]. Virtually all work is based on the classic paper of Das et. al. that uses STS clustering to convert real valued time series into symbolic values, which can then be manipulated by classic rule finding algorithms [9].
- The problem of anomaly detection in time series has been generalized to include the detection of surprising or interesting patterns (which are not necessarily anomalies). There are many approaches to this problem, including several based on STS clustering [45].
- Indexing of time series is an important problem that has attracted the attention of dozens of researchers. Several of the proposed techniques make use of STS clustering [27, 33].
- Several techniques for classifying time series make use of STS clustering to preprocess the data before passing to a standard classification technique such as a decision tree [7, 8].
- Clustering of streaming time series has also been proposed as a knowledge discovery tool in its own right. Researchers have suggested various techniques to speed up the clustering [11].

The above is just a small fraction of the work in the area, more extensive surveys may be found in [24, 35].

3. Demonstrations of the Meaninglessness of STS Clustering

In this section we will demonstrate the meaninglessness of STS clustering. In order to demonstrate that this meaninglessness is a product of the way the data is obtained by sliding windows, and not some quirk of the clustering algorithm, we will also do whole clustering as a control [12, 31].

3.1 K-means Clustering

Because k-means is a heuristic, hill-climbing algorithm, the cluster centers found may not be optimal [14]. That is, the algorithm is guaranteed to converge on a local, but not necessarily global optimum. The choices of the initial centers affect the quality of results. One technique to mitigate this problem is to do multiple restarts, and choose the best set of clusters [5]. An obvious question to ask is how much variability in the shapes of cluster centers we get between multiple runs. We can measure this variability with the following equation:

- Let $A=(\overline{a}_1,\overline{a}_2,...,\overline{a}_k)$ be the cluster centers derived from one run of k-means.
- Let $B=(\overline{b}_1,\overline{b}_2,...,\overline{b}_k)$ be the cluster centers derived from a different run of k-means.
- Let $dist(\overline{a}_i,\overline{a}_j)$ be the distance between two cluster centers, measured with Euclidean distance.

Then the distance between two sets of clusters can be defined as:

$$cluster\_distance(A,B)\equiv\sum_{i=1}^{k}\min\left[dist(\overline{a}_i,\overline{b}_j)\right] \quad ,1\le j\le k \qquad (1)$$

The simple intuition behind the equation is that each individual cluster center in *A* should map on to its closest counterpart in *B*, and the sum of all such distances tells us how similar two sets of clusters are.

An important observation is that we can use this measure not only to compare two sets of clusters derived for the same dataset, but also two sets of clusters which have been derived from *different* data sources. Given this fact, we propose a simple experiment.

We performed 3 random restarts of k-means on a stock market data set, and saved the 3 resulting sets of cluster centers into set **X**. We also performed 3 random restarts on random walk dataset, saving the 3 resulting sets of cluster centers into set **Y**.

We then measured the average cluster distance (as defined in equation 1), between each set of cluster centers in **X**, to each other set of cluster centers in **X.** We call this number *within_set_X_distance*. We also measured the average cluster distance between each set of cluster centers in **X**, to cluster centers in **Y;** we call this number *between_set_X_and_Y_distance*.

We can use these two numbers to create a fraction:

$$\text{clustering meaningfulness}(X,Y)\equiv\frac{within\_set\_X\_distance}{between\_set\_X\_and\_Y\_distance} \qquad (2)$$

We can justify calling this number "*clustering meaningfulness*" since it clearly measures just that. If the clustering algorithm is returning the same or similar sets of clusters despite different initial seeds, the numerator should be close to zero. In contrast, there is no reason why the clusters from two completely different, unrelated datasets to be similar. Therefore, we should expect the denominator to be relatively large. So overall we should expect that the value of *clustering meaningfulness*(**X**,**Y**) should be close to zero when **X** and **Y** are sets of cluster centers derived from different datasets.

As a control, we performed the exact same experiment, on the same data, but using subsequences that were randomly extracted, rather than extracted by a sliding window. We call this whole clustering.

Since it might be argued that any results obtained were the consequence of a particular combination of *k* and *w*, we tried the cross product of *k* = {3, 5, 7, 11} and *w* = {8, 16, 32}. For every combination of parameters we repeated the entire process 100 times, and averaged the results. Figure 2 shows the results.

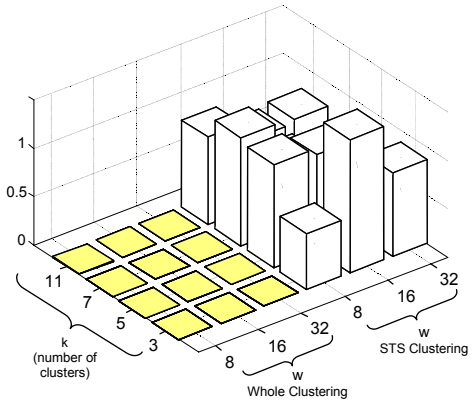


Figure 2. A comparison of the clustering meaningfulness for whole clustering, and STS clustering, using k-means with a variety of parameters. The two datasets used were Standard and Poor's 500 Index closing values and random walk data.

The results are astonishing. The cluster centers found by STS clustering on any particular run of k-means on stock market dataset are not significantly more similar to each other than they are to cluster centers taken from random walk data! In other words, if we were asked to perform clustering on a particular stock market dataset, we could reuse an old clustering obtained from random walk data, and no one could tell the difference!

We reemphasize here that the difference in the results for STS clustering and whole clustering in this experiment (and all experiments in this work) are due exclusively to the feature extraction step. In particular, both are being tested on the same dataset, with the same parameters of w and k, using the same algorithm.

We also note that the exact definition of *clustering meaningfulness* is not important to our results. In our definition, each cluster center in A maps onto its closest match in B. It is possible therefore that two or more cluster centers from A map to one center in B, and some clusters in B have no match. However we tried other variants of this definition, including pairwise matching, minimum matching and maximum matching, together with dozens of other measurements of clustering quality suggested in the literature [14]; it simply makes no significant difference to the results.

3.2 Hierarchical Clustering

The previous section suggests that k-means clustering of STS time series does not produce meaningful results, at least for stock market data. An obvious question to ask is, is this true for STS with other clustering algorithms? We will answer the question for hierarchical clustering here.

Hierarchical clustering, unlike k-means, is a deterministic algorithm. So we can't reuse the experimental methodology from the previous section exactly, however, we can do something very similar.

First we note that hierarchical clustering can be converted into a partitional clustering, by cutting the first k links [29]. Figure 3 illustrates the idea. The resultant time series in each of the k subtrees can then be merged into single cluster prototypes. When performing hierarchical clustering, one has to make a choice about how to define the distance between two clusters, this choice is called the linkage method (cf. line 3 of Table 1).

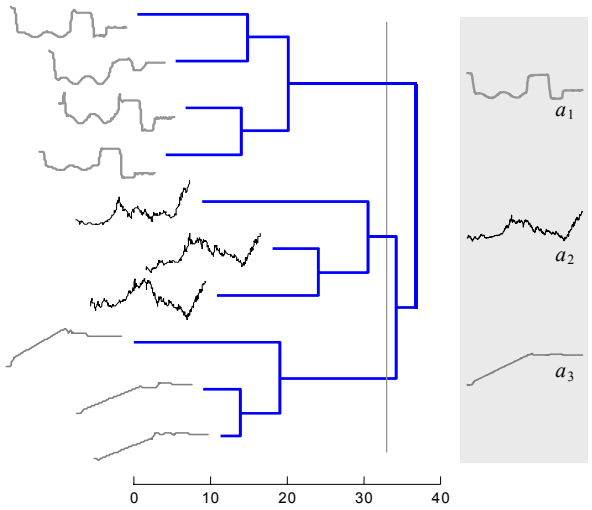


Figure 3. A hierarchical clustering of ten time series. The clustering can be converted to a k partitional clustering by "sliding" a cutting line until it intersects k lines of the dendrograms, then averaging the time series in the k subtrees to form k cluster centers (gray panel).

Three popular choices are complete linkage, average linkage and Ward's method [14]. We can use all three methods for the stock market dataset, and place the resulting cluster centers into set X. We can do the same for random walk data and place the resulting cluster centers into set Y. Having done this, we can extend the measure of clustering meaningfulness in Eq. 2 to hierarchical clustering, and run a similar experiment as in the last section, but using hierarchical clustering. The results of this experiment are shown in Figure 4.

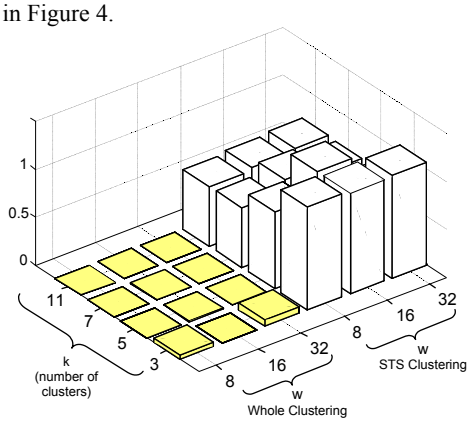


Figure 4. A comparison of the clustering meaningfulness for whole clustering and STS clustering using hierarchical clustering with a variety of parameters. The two datasets used were Standard and Poor's 500 Index closing values and random walk data.

Once again, the results are astonishing. While it is well understood that the choice of linkage method can have minor effects on the clustering found, the results above tell us that when doing STS clustering, the choice of linkage method has as much effect as the choice of dataset! Another way of looking at the results is as follows. If we were asked to perform hierarchical clustering on a particular dataset, but we did not have to report which linkage method we used, we could reuse an old random

walk clustering and no one could tell the difference without rerunning the clustering for every possible linkage method.

3.3 Other Datasets and Algorithms

The results in the two previous sections are extraordinary, but are they the consequence of some properties of stock market data, or as we claim, a property of the sliding window feature extraction? The latter is the case, which we can simply demonstrate. We visually inspected the UCR archive of time series datasets for the two time series datasets that appear the least alike [23]. The best two candidates we discovered are shown in Figure 5.

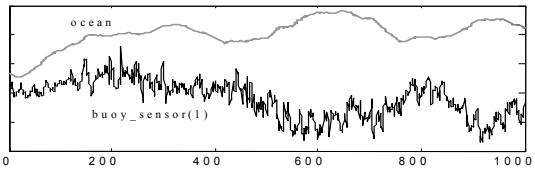


Figure 5. Two subjectively very dissimilar time series from the UCR archive. Only the first 1,000 datapoints are shown. The two time series have very different properties of stationarity, noise, periodicity, symmetry, autocorrelation etc.

We repeated the experiment of Section 3.2, using these two datasets in place of the stock market data and the random walk data. The results are shown in Figure 6.

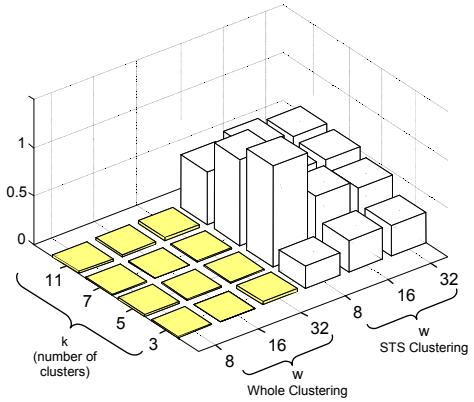


Figure 6. A comparison of the clustering meaningfulness for whole clustering, and STS clustering, using k-means with a variety of parameters. The two datasets used were buoy_sensor(1) and ocean.

In our view, this experiment sounds the death knell for clustering of STS time series. If we cannot easily differentiate between the clusters from these two extremely different time series, then how could we possibly find meaningful clusters in any data?

In fact, the experiments shown in this section are just a tiny subset of the experiments we performed. We tested other clustering algorithms, including EM and SOMs [43]. We tested on 42 different datasets [24, 26]. We experimented with other measures of clustering quality [14]. We tried other variants of k-means, including different seeding algorithms. Although Euclidean distance is the most commonly used distance measure for time series data mining, we also tried other distance measures from the literature, including Manhattan, L_∞, Mahalanobis distance and dynamic time warping distance [12, 24, 31]. We tried various normalization techniques, including Z-normalization, 0-1 normalization, amplitude only normalization, offset only normalization, no normalization etc. In every case we are forced to the inescapable conclusion: whole clustering of time series is usually a meaningful thing to do, but sliding window time series clustering is *never* meaningful.

3.4 Why is STS Clustering Meaningless?

Before explaining why STS clustering is meaningless, it will be instructive to visualize the cluster centers produced by both whole clustering and STS clustering. We will demonstrate on the classic Cylinder-Bell-Funnel data [26]. This dataset consists of random instantiations of the eponymous patterns, with Gaussian noise added. While each time series is of length 128, the onset and duration of the shape is subject to random variability. Figure 7 shows one instance from each of the three clusters.

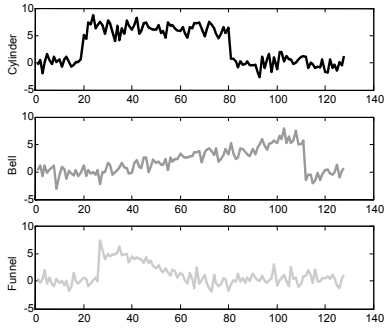


Figure 7. Examples of Cylinder, Bell, and Funnel patterns.

We generated a dataset of 30 instances of each pattern, and performed k-means clustering on it, with k = 3. The resulting cluster centers are show in Figure 8. As one might expect, all three clusters are successfully found. The final centers closely resemble the three different patterns in the dataset, although the sharp edges of the patterns have been somewhat "softened" by the averaging of many time series with some variability in the time axis.

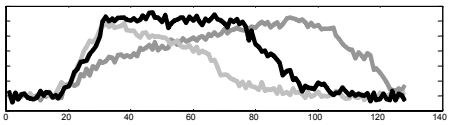


Figure 8. The three final centers found by k-means on the cylinder-bell-funnel dataset. The shapes of the centers are close approximation of the original patterns.

To compare the results of whole clustering to STS clustering, we took the 90 time series used above and concatenated them into one long time series. We then performed STS k-means clustering. To make it easy for the algorithm, we use the exact length of the patterns (w = 128) as the window length, and k = 3 as the number of desired clusters. The cluster centers are shown in Figure 9.

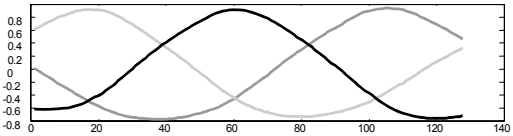


Figure 9. The three final centers found by subsequence clustering using the sliding window approach.

The results are extraordinarily unintuitive! The cluster centers look nothing like any of the patterns in the data; what's more, they appear to be perfect sine waves.

In fact, for $w << m$, we get approximate sine waves with STS clustering regardless of the clustering algorithm, the number of clusters, or the dataset used! Furthermore, although the sine waves are always exactly out of phase with each other by $1/k$ period, overall, their joint phase is arbitrary, and will change with every random restart of k-means.

This result completely explains the results from the last section. If sine waves appear as cluster centers for every dataset, then clearly it will be impossible to distinguish one dataset's clusters from another. Although we have now explained the inability of STS clustering to produce meaningful results, we have revealed a new question: why do we always get cluster centers with this special structure?

3.5 A Hidden Constraint

To explain the unintuitive results above, we must introduce a new fact.

> **Theorem 1**: For any time series dataset T with an overall trend of zero, if T is clustered using sliding windows, and $w << m$, then the mean of all the data (i.e. the special case of $k = 1$), will be an approximately constant vector.

In other words, if we run STS k-means on *any* dataset, with $k = 1$ (an unusual case, but perfectly legal), we will always end up with a horizontal line as the cluster center. The proof of this fact is straightforward but long, so we have elucidated it in a separate technical report [41]. Note that the requirement that the overall trend be zero can be removed, in which case, the $k = 1$ cluster center is still a straight line, but at some angle.

We content ourselves here with giving the intuition behind the proof, and offering a visual "proof" in Figure 10.

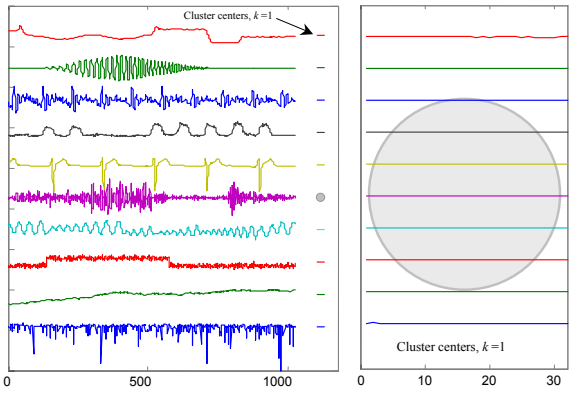


Figure 10: A visual "proof" of Theorem 1. Ten time series of vastly different properties of stationarity, noise, periodicity, symmetry, autocorrelation etc. The cluster centers for each time series, for $w = 32$, $k = 1$ are shown at right. Far right shows a zoom-in that shows just how close to a straight line the cluster centers are. While the objects have been shifted for clarity, they have *not* been rescaled in either axis; note the light gray circle in both graphs. The datasets used are, reading from top to bottom: Space Shuttle, Flutter, Speech, Power_Data, Koski_ecg, Earthquake, Chaotic, Cylinder, Random_Walk, and Balloon.

The intuition behind Theorem 1 is as follows. Imagine an arbitrary datapoint t_i somewhere in the time series T, such that $w \le i \le m - w + 1$. If the time series is much longer than the window size, then virtually all datapoints are of this type. What contribution does this datapoint make to the overall mean of the STS matrix S? As the sliding window passes by, the datapoint first appears as the rightmost value in the window, then it goes on to appear exactly once in *every* possible location within the sliding window. So the t_i datapoint contribution to the overall shape is the same everywhere and must be a horizontal line. Only those points at the very beginning and the very end of the time series avoid contributing their value to all w columns of S, but these are asymptotically irrelevant. The average of many horizontal lines is clearly just another horizontal line.

The implications of Theorem 1 become clearer when we consider the following well documented fact. For any dataset, the weighted (by cluster membership) average of k clusters must sum up to the global mean. The implication for STS clustering is profound. If we hope to discover k clusters in our dataset, we can only do so if the weighted average of these clusters happens to sum to a constant line! However, there is no reason why we should expect this to be true of *any* dataset, much less *every* dataset. This hidden constraint limits the utility of STS clustering to a vanishing small set of subspace of all datasets.

3.6 The Importance of Trivial Matches

There are further constraints on the types of datasets where STS clustering could possibly work. Consider a subsequence C_p that is a member of a cluster. If we examine the entire dataset for similar subsequences, we should typically expect to find the best matches to C_p to be the subsequences $\ldots, C_{p-2}, C_{p-1}, C_{p+1}, C_{p+2}, \ldots$ In other words, the best matches to any subsequence tend to be just slightly shifted versions of the subsequence. Figure 11 illustrates the idea, and Definition 4 states it more formally.

> **Definition 4**. *Trivial Match*: Given a subsequence C beginning at position p, a matching subsequence M beginning at q, and a distance R, we say that M is a *trivial match* to C of order R, if either $p = q$ or there does not exist a subsequence M' beginning at q' such that $D(C, M') > R$, and either $q < q' < p$ or $p < q' < q$.

The importance of trivial matches, in a different context, has been documented elsewhere [28]

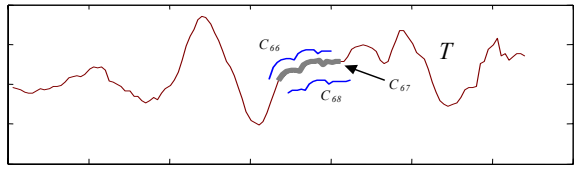


Figure 11: For almost any subsequence C in a time series, the closest matching subsequences are the subsequences immediately to the left and right of C.

An important observation is the fact that different subsequences can have vastly different numbers of trivial matches. In particular, smooth, slowly changing subsequences tend to have many trivial matches, whereas subsequences with rapidly changing features and/or noise tend to have very few trivial matches. Figure 12 illustrates the idea. The figure shows a time series that subjectively appears to have a cluster of 3 square waves. However, the bottom plot shows how many trivial matches each subsequence has. Note that the square waves have very few trivial matches, so all three taken together sit in a sparsely populated region of w-space. In contrast, consider the relatively smooth Gaussian bump centered at 125. The subsequences in the smooth ascent of this feature have more than 25 trivial matches, and thus

sit in a dense region of *w*-space; the same is true for the subsequences in the descent from the peak. So if clustering this dataset with k-means, $k = 2$, the two cluster centers will be irresistibly drawn to these two "shapes", simple ascending and descending lines.

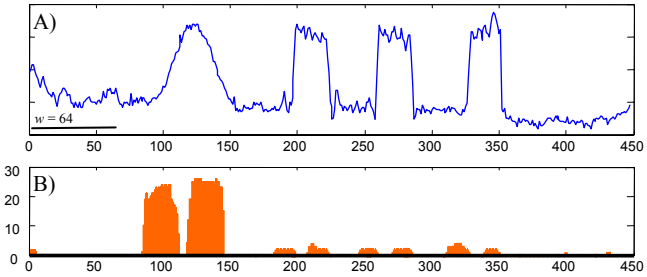


Figure 12: A) A time series *T* that subjectively appears to have a cluster of 3 noisy square waves. B) Here the i^{th} value is the number of trivial matches for the subsequence C_i in *T*, where $R = 1$, $w = 64$.

The importance of this observation for STS clustering is obvious. Imagine we have a time series where we subjectively see two clusters: equal numbers of a smooth slowing changing pattern, and a noisier pattern with many features. In *w*-dimensional space, the smooth pattern is surrounded by many trivial matches. This dense volume will appear to any clustering algorithm an extremely promising cluster center. In contrast, the highly featured, noisy pattern has very few trivial matches, and thus sits in a relatively sparse space, all but ignored by the clustering algorithm. Note that it is not possible to simply remove or "factor out" the trivial matches since there is no way to know beforehand the true patterns.

We have not yet fully explained why the cluster centers for STS clustering degenerate to sine waves (cf Figure 9). However, we have shown that for STS "clustering", algorithms do not really cluster the data. If not clustering, what are the algorithms doing? It is instructive to note that if we perform singular value decomposition on time series, we also get shapes that seem to approximate sine waves [25]. This suggests that STS clustering algorithms are simply returning a set of basis functions that can be added together in a weighted combination to approximate the original data.

An even more tantalizing piece of evidence exists. In the 1920's "data miners" were excited to find that by preprocessing their data with repeated smoothing, they could discover trading cycles. Their joy was shattered by a theorem by Evgeny Slutsky (1880-1948), who demonstrated that any noisy time series will converge to a sine wave after repeated applications of moving window smoothing [22]. While STS clustering is not exactly the same as repeated moving window smoothing, it is clearly highly related. For brevity we will defer future discussion of this point to future work.

3.7 Is there a Simple Fix?

Having gained an understanding of the fact that STS clustering is meaningless, and having developed an intuition as to why this is so, it is natural to ask if there is a simple modification to allow it to produce meaningful results. We asked this question, not just among ourselves, but also to dozens of time series clustering researchers with whom we shared our initial results. While we considered all suggestions, we discuss only the two most promising ones here.

The first idea is to increment the sliding window by more that one unit each time. In fact, this idea was suggest by [9], but only as a speed up mechanism. Unfortunately, this idea does not help. If the new step size *s* is much smaller than *w*, we still get the same empirical results. If *s* is approximately equal to, or larger than *w*, we are no longer doing subsequence clustering, but whole clustering. This is not useful, since the choice of the offset for the first window is a critical parameter, and choices that differ by just one timepoint can give arbitrarily different results.

The second idea is to set *k* to be some number much greater than the true number of clusters we expect to find, then do some post-processing to find the *real* clusters. Empirically, we could not make this idea work, even on the trivial dataset introduced at the beginning of this section. We found that even if *k* is extremely large, unless it is a significant fraction of *T*, we still get arbitrary sine waves as cluster centers. In addition, we note that the time complexity for k-means increases with *k*.

It is our belief that there is no simple solution to the problem of STS-clustering; the definition of the problem is itself intrinsically flawed.

3.8 Necessary Conditions for STS Clustering to Work

We conclude this section with a summary of the conditions that must be satisfied for STS clustering to be meaningful.

Assume that a time series contains *k* approximately or exactly repeated patterns of length *w*. Further assume that we happen to know *k* and *w* in advance. A necessary (but not necessarily sufficient) condition for a clustering algorithm to discover the *k* patterns is that the weighted mean of the patterns must sum to a horizontal line, and each of the *k* patterns must have approximately equal numbers of trivial matches.

It is obvious that the chances of both these conditions being met is essentially zero.

4. A Case Study on Existing Work

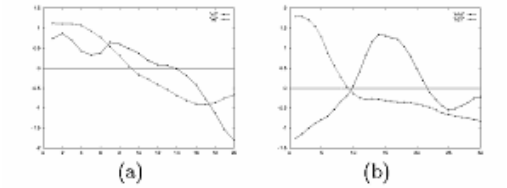
As we noted in the introduction, an obvious counter argument to our claim is the following. "*Since many papers have been published which use time series subsequence clustering as a subroutine, and these papers produce successful results, time series subsequence clustering must be a meaningful operation.*" To counter this argument, we have reimplemented the most influential such work, the Time Series Rule Finding algorithm of Das et. al. [9] (the algorithm is not named in the original work, we will call it TSRF here for brevity and clarity).

4.1 (Not) Finding Rules in Time Series

The algorithm begins by performing STS clustering. The centers of these clusters are then used as primitives that are fed into a slightly modified version of a classic association rule algorithm [2]. Finally the rules are ranked by their J-measure, an entropy based measure of their significance.

The rule finding algorithm found the rules shown in Figure 13 using 19 months of NASDAQ data. The high values of support, confidence and J-measure are offered as evidence of the significance of the rules. The rules are to be interpreted as follows. In Figure 13 (b) we see that "***if*** *stock rises then falls*

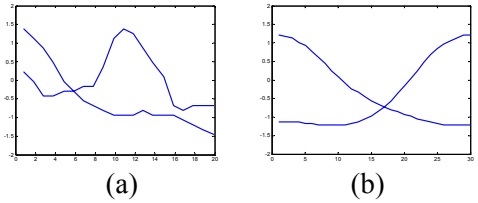
greatly, follow a smaller rise, ***then*** *we can expect to see within 20 time units, a pattern of rapid decrease followed by a leveling out*." [9].



w	d	Rule	Sup %	Conf %	J-Mea.	Fig
20	5.5	$7 \Rightarrow^{15} 8$	8.3	73.0	0.0036	(a)
30	5.5	$18 \Rightarrow^{20} 21$	1.3	62.7	0.0039	(b)

Figure 13: Above, two examples of "significant" rules found by Das et. al. (This is a capture of Figure 4 from their paper). Below, a table of the parameters they used and results they found.

What would happen if we used the TSRF algorithm to try to find rules in random walk data, using exactly the same parameters? Since no such rules should exist by definition, we should get radically different results[1]. Figure 14 shows one such experiment; the support, confidence and J-measure values are essentially the same as in Figure 13!



w	d	Rule	Sup %	Conf %	J-Mea	Fig
20	5.5	$11 \Rightarrow^{15} 3$	6.9	71.2	0.0042	(a)
30	5.5	$24 \Rightarrow^{20} 19$	2.1	74.7	0.0035	(b)

Figure 14: Above, two examples of "significant" rules found in *random walk* data using the techniques of Das et. al. Below, we used identical parameters and found near identical results.

This one experiment might have been an extraordinary coincidence; we might have created a random walk time series that happens to have some structure to it. Therefore, for every result shown in the original paper we ran 100 recreations using different random walk datasets, using quantum mechanically generated numbers to insure randomness [44]. In every case the results published cannot be distinguished from our results on random walk data.

The above experiment is troublesome, but perhaps there are simply no rules to be found in stock market. We devised a simple experiment in a dataset that does contain known rules. In particular we tested the algorithm on a normal healthy electrocardiogram. Here, there is an obvious rule that one heartbeat follows another. Surprisingly, even with much tweaking of the parameters, the TSRF algorithm cannot find this simple rule.

[1] Note that the shapes of the patterns in Figures 13 and 14 are only very approximately sinusoidal. This is because the time series are relatively short compared to the window length. When the experiments are repeated with longer time series, the shapes converge to pure sine waves.

The TSRF algorithm is based on the classic rule mining work of Agrawal et. al. [2]; the only difference is the STS step. Since the work of [2] has been carefully vindicated in 100's of experiments on both real and synthetic datasets, it seems reasonable to conclude that the STS clustering is at the heart of the problems with the TSRF algorithm.

These results may appear surprising, since they invalidate the claims of a highly referenced paper, and many of the dozens of extensions researchers have proposed [9, 11, 15, 16, 17, 20, 21, 30, 32, 36, 42, 45]. However, in retrospect, this result should not really be too surprising. Imagine that a researcher claims to have an algorithm that can differentiate between three types of Iris flowers (Setosa, Virginica and Versicolor) based on petal and sepal length and width [10]. This claim is not so extraordinary, given that it is well known that even amateur botanists and gardeners have this skill [6]. However, the paper in question is claiming to introduce an algorithm that can find rules in stock market time series. There is simply no evidence that any human can do this, in fact, the opposite is true: every indication suggests that the patterns much beloved by technical analysts such as the "calendar effect" are completely spurious [19, 39].

5. A Tentative Solution

The results presented in this paper thus far are somewhat downbeat. In this section we modify the tone by introducing an algorithm that *can* find clusters in some streaming time series. This algorithm is not presented as the best way to find clusters in streaming time series; for one thing, its time complexity is untenable for massive datasets. It is simply offered as an existence proof that such an algorithm exists, and to pave the way for future research.

Our algorithm is motivated by the two observations in Section 4 that attempting to cluster every subsequence produces an unrealistic constraint, and that considering trivial matches causes smooth, low-detail subsequences to form pseudo clusters.

We begin by considering motifs, a concept highly related to clusters. Motifs are overrepresented sequences in discrete strings, for example, in musical or DNA sequences [34]. Classic definitions of motifs require that the underling data be discrete, but in recent work the present authors have extended the definitions to real valued time series [28]. Figure 15 illustrates a visual intuition of a motif, and Definition 5 defines the concept more concretely.

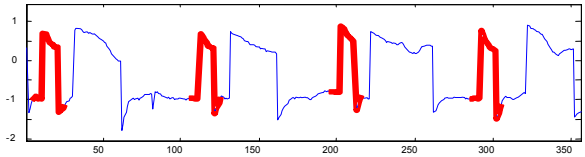


Figure 15: An example of a motif that occurs 4 times in a short section of the Winding(4) dataset.

Definition 5. *K-Motifs*: Given a time series T, a subsequence length n and a distance range R, the most significant motif in T (called *1-Motif*) is the subsequence C_1 that has the highest count of non-trivial matches. The K^{th} most significant motif in T (called *K-Motif*) is the subsequence C_K that has the highest count of non-trivial matches, and satisfies $D(C_K, C_i) > 2R$, for all $1 \leq i < K$.

Although motifs may be considered similar to clusters, there are several important differences, a few of which we enumerate here.

- When mining motifs, we must specify an additional parameter *R*.
- Assuming the distance *R* is defined as Euclidean, motifs always define circular regions in space, whereas clusters may have arbitrary shapes[2].
- Motifs generally define a small subset of the data, and not the entire dataset.
- The definition of motifs explicitly eliminates trivial matches.

Note that while the first two points appear as limitations, the last two points explicitly counter the two reasons that STS clustering cannot produce meaningful results.

We cannot simply run a *K*-motif detection algorithm in place of STS clustering, since a subset of the motifs discovered might really be a group that should be clustered together. For example, imagine a true cluster that sits in a hyper-ellipsoid. It might be approximated by 2 or 3 motifs that cover approximately the same volume. However, we could run a *K*-motif detection algorithm, with $K >> k$, to extract promising subsequences from the data, then use a classic clustering algorithm to cluster only these subsequences. This idea is formalized in Table 3.

Table 3: An outline of the motif-based-clustering algorithm.

Algorithm *motif-based-clustering*	
1.	Decide on a value for *k*.
2.	Discover the *K*-motifs in the data, for $K = k \times c$ (c is some constant, in the region of about 2 to 30)
3.	Run *k*-means, or *k* partitional hierarchical clustering, or any other clustering algorithm on the subsequences covered by *K*-motifs

Line two of the algorithm requires a call to a motif discovery algorithm; such an algorithm appears in [28].

5.1 Experimental Results

We have seen in Section 6 that the cluster centers returned by STS have been mistaken for meaningful clusters by many researchers. To eliminate the possibility of repeating this mistake, we will demonstrate the proposed algorithm on the dataset introduced in Section 4, which consists of the concatenation of 30 examples each of the Cylinder, Bell, Funnel shapes, in random order. We would like our clustering algorithm to be able to find clusters centers similar to the ones shown in Figure 8.

We ran the motif based clustering algorithm with $w = 128$, and $k = 3$, which is fair since we also gave these two correct parameters to all the algorithms above. We needed to specify the value of *R*, we did this by simply examining a fraction of our dataset, finding ten pairs of subsequences we found to be similar, measuring the Euclidean distance between these pairs, and averaging the results. The cluster centers found are shown in Figure 16.

These results tell use that on at least some datasets, we can do meaningful streaming clustering. The fact that this is achieved by working with only a subset of the data, and explicitly excluding trivial matches, further supports our explanations in Sections 4.1 and 4.2, of why STS clustering is meaningless.

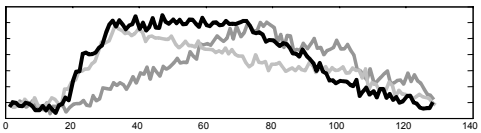


Figure 16: The cluster centers found by the motif-based-clustering algorithm on the concatenated Cylinder-Bell-Funnel dataset. Note the results are very similar to the prototype shapes shown in Figure 7, and the cluster centers found by the whole matching case, shown in Figure 8.

6. Conclusions

We have shown that a popular technique for data mining does not produce meaningful results. We have further explained the reasons why this is so.

Although our work may be viewed as negative, we have shown that a reformulation of the problem can allow clusters to be extracted from streaming time series. In future work we intend to consider several related questions; for example, whether or not the weaknesses of STS clustering described here have any implications for model-based, streaming clustering of time series, or streaming clustering of nominal data [13].

Acknowledgments: We gratefully acknowledge the following people who looked at an early draft of this paper and made useful comments and suggestions: Christos Faloutsos, Michalis Vlachos, Frank Höppner, Howard Hamilton, Daniel Barbara, Magnus Lie Hetland, Hongyuan Zha, Sergio Focardi, Xiaoming Jin, Shoji Hirano, Shusaku Tsumoto, Mark Last, and Zbigniew Struzik.

7. References

[1] Aggarwal, C., Hinneburg, A., & Keim, D. A. (2001). On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *proceedings of the 8th Int'l Conference on Database Theory*. London, UK, Jan 4-6. pp 420-434.

[2] Agrawal, R., Imielinski, T. & Swami, A. (1993). Mining Association Rules Between Sets of Items in Large Databases. In *proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*. Washington, D.C., May 26-28. pp. 207-216.

[3] Bar-Joseph, Z., Gerber, G., Gifford, D., Jaakkola, T. & Simon, I. (2002). A New Approach to Analyzing Gene Expression Time Series Data. In *proceedings of the 6th Annual Int'l Conference on Research in Computational Molecular Biology*. Washington, D.C., Apr 18-21. pp 39-48.

[4] Beyer, K., Goldstein, J., Ramakrishnan, R. & Shaft, U. (1999). When is Nearest Neighbor Meaningful? In *proceedings of the 7th Int'l Conference on Database Theory*. Jerusalem, Israel, Jan 10-12. pp 217-235.

[5] Bradley, P. S. & Fayyad, U.M. (1998). Refining Initial Points for K--Means Clustering. In *proceedings of the 15th Int'l Conference on Machine Learning*. Madison, WI, July 24-27. pp. 91-99.

[6] British Iris Society, Species Group Staff. (1997). A Guide to Species Irises: Their Identification and Cultivation. *Cambridge University Press*. March, 1997.

[7] Cotofrei, P. (2002). Statistical Temporal Rules. In *proceedings of the 15th Conference on Computational Statistics* - Short Communications and Posters. Berlin, Germany, Aug 24-28.

[2] It is true that k-means favors circular clusters, but more generally, clustering algorithms can define arbitrary spaces.

[8] Cotofrei, P. & Stoffel, K (2002). Classification Rules + Time = Temporal Rules. In *proceedings of the 2002 Int'l Conference on Computational Science*. Amsterdan, Netherlands, Apr 21-24. pp 572-581.

[9] Das, G., Lin, K., Mannila, H., Renganathan, G. & Smyth, P. (1998). Rule Discovery from Time Series. In *proceedings of the 4th Int'l Conference on Knowledge Discovery and Data Mining*. New York, NY, Aug 27-31. pp 16-22.

[10] Fisher, R. A. (1936). The Use of Multiple Measures in Taxonomic Problems. *Annals of Eugenics.* Vol. 7, No. 2, pp 179-188.

[11] Fu, T. C., Chung, F. L., Ng, V. & Luk, R. (2001). Pattern Discovery from Stock Time Series Using Self-Organizing Maps. *Workshop Notes of KDD2001 Workshop on Temporal Data Mining*. San Francisco, CA, Aug 26-29. pp 27-37.

[12] Gavrilov, M., Anguelov, D., Indyk, P. & Motwani, R. (2000). Mining the Stock Market: Which Measure is Best? In *proceedings of the 6th ACM Int'l Conference on Knowledge Discovery and Data Mining*. Boston, MA, Aug 20-23. pp 487-496.

[13] Guha, S. Mishra, N. Motwani, R. & O'Callaghan, L (2000). Clustering Data Streams. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science.* Redondo Beach, CA. Nov 12-14. pp. 359-366.

[14] Halkidi, M., Batistakis, Y. & Vazirgiannis, M. (2001). On Clustering Validation Techniques. *Journal of Intelligent Information Systems* (JIIS), Vol. 17, No. 2-3. pp. 107-145.

[15] Harms, S. K., Deogun, J. & Tadesse, T. (2002). Discovering Sequential Association Rules with Constraints and Time Lags in Multiple Sequences. In *proceedings of the 13th Int'l Symposium on Methodologies for Intelligent Systems*. Lyon, France, June 27-29. pp 432-441.

[16] Harms, S. K., Reichenbach, S. Goddard, S. E., Tadesse, T. & Waltman, W. J. (2002). Data Mining in a Geospatial Decision Support system for Drought Risk Management. In *proceedings of the 1st National Conference on Digital Government*. Los Angeles, CA, May 21-23. pp. 9-16.

[17] Hetland, M. L. & Sætrom, P. (2002). Temporal Rules Discovery Using Genetic Programming and Specialized Hardware. In *proceedings of the 4th Int'l Conference on Recent Advances in Soft Computing.* Nottingham, UK, Dec 12-13.

[18] Honda, R., Wang, S., Kikuchi, T. & Konishi, O. (2002). Mining of Moving Objects from Time-Series Images and its Application to Satellite Weather Imagery. *The Journal of Intelligent Information Systems*, Vol. 19, No. 1, pp. 79-93.

[19] Jensen, D. (2000). Data Snooping, Dredging and Fishing: The dark Side of Data Mining. SIGKDD99 panel report. *ACM SIGKDD Explorations*, Vol. 1, No. 2. pp. 52-54.

[20] Jin, X., Lu, Y. & Shi, C. (2002). Distribution Discovery: Local Analysis of Temporal Rules. In *proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Taipei, Taiwan, May 6-8. pp 469-480.

[21] Jin, X., Wang, L., Lu, Y. & Shi, C. (2002). Indexing and Mining of the Local Patterns in Sequence Database. In *proceedings of the 3rd International Conference on Intelligent Data Engineering and Automated Learning*. Manchester, UK, Aug 12-14. pp 68-73.

[22] Kendall, M. (1976) Time-Series. 2nd Edition. Charles Griffin and Company, Ltd., London.

[23] Keogh, E. (2002). The UCR Time Series Data Mining Archive [http://www.cs.ucr.edu/~eamonn/TSDMA/index.html]. Riverside CA. University of California - Computer Science & Engineering Department

[24] Keogh, E. (2002). Exact Indexing of Dynamic Time Warping. *In proceedings of the 28th International Conference on Very Large Data Bases*. Hong Kong, Aug 20-23. pp 406-417.

[25] Keogh, E. Chakrabarti, K. Pazzani, M & Mehrotra, S. (2001). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Journal of Knowledge and Information Systems. Vol. 3, No. 3, pp. 263-286.

[26] Keogh, E. & Kasetty, S. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *In proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. July 23 - 26, 2002. Edmonton, Alberta, Canada. pp 102-111.

[27] Li, C., Yu, P. S. & Castelli, V. (1998). MALM: A Framework for Mining Sequence Database at Multiple Abstraction Levels. In *proceedings of the 7th ACM CIKM Int'l Conference on Information and Knowledge Management*. Bethesda, MD, Nov 3-7. pp 267-272.

[28] Lin, J. Keogh, E. Patel, P. & Lonardi, S. (2002). Finding motifs in time series. In *the 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. July 23 - 26, 2002. Edmonton, Alberta, Canada.

[29] Mantegna., R. N. (1999). Hierarchical Structure in Financial Markets. *European. Physical Journal*. B11, pp. 193-197.

[30] Mori, T. & Uehara, K. (2001). Extraction of Primitive Motion and Discovery of Association Rules from Human Motion. In *proceedings of the 10th IEEE Int'l Workshop on Robot and Human Communication*, Bordeaux-Paris, France, Sept 18-21. pp 200-206.

[31] Oates, T. (1999). Identifying Distinctive Subsequences in Multivariate Time Series by Clustering. *In proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*. San Diego, CA, Aug 15-18. pp 322-326.

[32] Osaki, R., Shimada, M. & Uehara, K. (2000). A Motion Recognition Method by Using Primitive Motions, Arisawa, H. and Catarci, T. (eds.) *Advances in Visual Information Management, Visual Database Systems*, Kluwer Academic Pub. pp 117-127.

[33] Radhakrishnan, N., Wilson, J. D. & Loizou, P. C. (2000). An Alternate Partitioning Technique to Quantify the Regularity of Complex Time Series. *International Journal of Bifurcation and Chaos*, Vol. 10, No. 7. World Scientific Publishing. pp 1773-1779.

[34] Reinert, G., Schbath, S. & Waterman, M. S. (2000). Probabilistic and statistical properties of words: An overview. J. *Comput. Bio*., Vol. 7, pp 1-46.

[35] Roddick, J. F. & Spiliopoulou, M. (2002). A Survey of Temporal Knowledge Discovery Paradigms and Methods. *Transactions on Data Engineering.* Vol. 14, No. 4, pp 750-767.

[36] Sarker, B. K., Mori, T. & Uehara, K. (2002). Parallel Algorithms for Mining Association Rules in Time Series Data. CS24-2002-1 Tech report.

[37] Schittenkopf, C., Tino, P. & Dorffner, G. (2000). The Benefit of Information Reduction for Trading Strategies. *Report Series for Adaptive Information Systems and Management in Economics and Management Science*, July. Report #45.

[38] Steinback, M., Tan, P.N., Kumar, V., Klooster, S. & Potter, C. (2002). Temporal Data Mining for the Discovery and Analysis of Ocean Climate Indices. In the *2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, Alberta, Canada. July 23.

[39] Timmermann, A., Sullivan, R. & White, H. (1998). The Dangers of Data-Driven Inference: The Case of Calendar Effects in Stock Returns. *FMG Discussion Papers dp0304*, Financial Markets Group and ESRC.

[40] Tino, P., Schittenkopf, C. & Dorffner, G. (2000). Temporal Pattern Recognition in Noisy Non-stationary Time Series Based on Quantization into Symbolic Streams: Lessons Learned from Financial Volatility Trading. *Report Series for*

Adaptive Information Systems and Management in Economics and Management Science, July. Report #46.

[41] Truppel, Keogh, Lin (2003). A Hidden Constraint When Clustering Streaming Time Series. UCR Tech Report.

[42] Uehara, K. & Shimada, M. (2002). Extraction of Primitive Motion and Discovery of Association Rules from Human Motion Data. *Progress in Discovery Science 2002, Lecture Notes in Artificial Intelligence*, Vol. 2281. Springer-Verlag. pp 338-348.

[43] Van Laerhoven, K. (2001). Combining the Kohonen Self-Organizing Map and K-Means for On-line Classification of Sensor data. *Artificial Neural Networks*, Dorffner, G., Bischof, H. & Hornik, K. (Eds.), Vienna, Austria; Lecture Notes in Artificial Intelligence. Vol. 2130, Springer Verlag, pp.464-470.

[44] Walker, J. (2001). HotBits: Genuine Random Numbers Generated by Radioactive Decay. www.fourmilab.ch/hotbits/

[45] Yairi, T., Kato, Y. & Hori, K. (2001). Fault Detection by Mining Association Rules in House-keeping Data. In *proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*. Montreal, Canada, June 18-21.