Dot Plots for Time Series Analysis

Dragomir Yankov, Eamonn Keogh, Stefano Lonardi
Dept. of Computer Science & Eng.
University of California
Riverside, USA
{dyankov,eamonn,stelo}@cs.ucr.edu

Ada Waichee Fu
Dept. of Computer Science & Eng.
The Chinese University of Hong Kong
Shatin, N.T. Hong Kong.
adafu@cse.cuhk.edu.hk

Abstract

Since their introduction in the seventies by Gibbs and McIntyre, dot plots have proved to be a powerful and intuitive technique for visual sequence analysis and mining. Their main domain of application is the field of bioinformatics where they are frequently used by researchers in order to elucidate genomic sequence similarities and alignment. However, this useful technique has remained comparatively constrained to domains where the data has an inherent discrete structure (i.e., text).

In this paper we demonstrate how dot plots can be used for the analysis and mining of real-valued time series. We design a tool that creates highly descriptive dot plots which allow one to easily detect similarities, anomalies, reverse similarities, and periodicities as well as changes in the frequencies of repetitions. As the underlying algorithm scales well with the input size, we also show the feasibility of the plots for on-line data monitoring.

1. Introduction

The "diagonal match" or the "diagram method", as its founders Gibbs and McIntyre [11] initially called the dot plot method, was intended as a simple alternative for detecting similarities in amino acid sequences. Table 1 demonstrates the idea behind the plots: a dot is placed at position (i, j) in the collision matrix M_{mn}, for two strings of size m and n respectively, if their letters on the corresponding positions match.

Table 1. The collision matrix for strings `atgat` and `atgtag`

	a	t	g	t	a	g
a	•				•	
t		•		•		
g			•			•
a	•				•	
t		•		•		

The three interesting patterns that could be observed on the created plots are *matches*, *reverses* and *gaps*. A match is represented by a diagonal line in the collision matrix, e.g., the diagonal for the prefixes of the two sequences above (`atg`, `atg`). The reverses are captured by diagonals perpendicular to the main diagonal (`atg`, `gta`). The gaps imply a mismatch between the sequences compared and could be of particular interest if we look for anomalies in the data. For example if we compare two very similar chromosomes the plot is likely to have a long diagonal and the gaps in this diagonal would focus our attention on the regions that contain mutations.

Later works (see, e.g., [20]) suggested further refinements that would allow the detection of partial homologies between the sequences. The filters made the plots a lot more descriptive and lead to their vast popularization among the bioinformatics community. Yet this simple but very powerful technique remains restricted to domains where the data is represented by strings over a finite alphabet. Some examples of such domains outside the scope of bioinformatics are code and text similarity exploration [6], bilingual text translations [7], detecting hypertext link structure [2], etc.

A natural question that arises is how can we meaningfully apply the dot plots on real value time series data. One possible solution, called *recurrence plots*, was proposed by Eckmann et al [10] (see Figure 1-*left*). Their goal was to design a tool that would allow for the easier detection of patterns in the recurrent behavior of dynamical systems. The high dimensional phase space of the systems could be displayed on the two dimensional recurrence plot based on the Heaviside function:

$$M_{ij} = H(r(x_i) - \parallel x_i - x_j \parallel),\ i, j \in 1 \ldots m \quad (1)$$

where x_i are the states of the system, and $r(x_i)$ is the radius of the hypersphere around point x_i. As of today, a lot of modifications to recurrence plots have been suggested mainly in the choice of the radius and the expression on

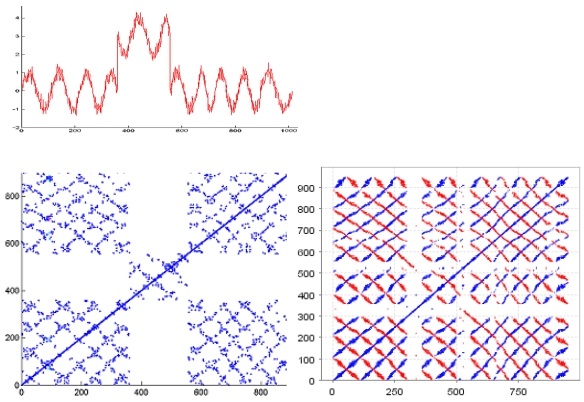


Figure 1. A sequence is compared to itself. *Top:* **the input time series.** *Left:* **a recurrence plot corresponding to the time series.** *Right:* **the time series dot plot generated with our tool**

which the Heaviside function is computed. The modifications usually target the detection of specific properties as oscillations, stationarity, heavy fluctuations etc, but the main shortcomings with recurrence plots still remain. Namely, the generated plots contain a significant amount of background noise similarly to the unfiltered dot plots. This is due to the fact that they use a local point based criteria for placing the dots rather than considering whole subsequences which also makes them not very suitable for handling *approximate* as well as time warped motifs. Solutions like the *thresholded recurrence plots* have been proposed to decrease the noise and strengthen the signal of the pattern. However, we believe that the approach of analyzing the data point-wise is not flexible enough for the purpose of time series analysis.

In this work we break the time series into smaller subsequences which we compare in search of possible motifs, anomalies or trends. Our approach is based on the algorithm for probabilistic discovery of *approximately repeated* subsequences, that some of us introduced in [5]. The algorithm first obtains a lower dimensional representation of the subsequences to be compared, and then uses a set of hash functions to project them into different classes of equivalence. The choice of the algorithm was determined by several requirements that we imposed in order to design practically useful dot plots. Ideally, we wanted to have a method that is (1) robust to noise, (2) invariant to value and time shifts, (3) invariant to a certain amount of time warping, and (4) efficiently computable.

Figure 1-*right* illustrates the plot generated by our algorithm when applied to a synthetic time series. The input time series is a simple sum of a sine wave with a square wave and some additional noise. Our dot plot (right) clearly emphasizes several important features of the time series that could be missed on the original recurrence plot (left). First, the continuous diagonal lines in our dot plot clearly identify the repeated patterns (due to the periodicity of the sine wave). Second, the two pairs of perpendicular white lines clearly show the boundaries of the central shifted region. At the same time the diagonal lines inside the central region show that the shifted segment is another repeated pattern. Third, the frequency change in the sine wave is illustrated by the curvature of the diagonals. Finally, our dot plot clearly shows the points that belong to forward (blue) and reverse (red) matches.

The rest of the paper is organized as follows. In Section 2 we describe the random projection algorithm and give a new estimate on the number of iterations necessary for the algorithm to identify the time series motifs on the dot plot. The application of the tool for the purpose of anomaly detection and pattern finding is demonstrated on several real-world examples in Section 3. In Section 4 we examine how the "invariance to time warping" can be detected by employing ideas from the field of time series segmentation.

2 Methods

To build the time series dot plots we proceed by first running a sliding window along the series which we want to compare. For every window position we discretize the corresponding subsequence. Then we show that the problem of computing the collision matrix for the time series dot plots can be reduced to the problem of finding motifs in the discretized data. Discovering motifs is an essential step in many time series mining tasks, namely, time series clustering and classification [15], novelty detection [9], robot planning [21], mining association rules in time series [8], among others. Here, motifs are discovered by a probabilistic approach based on random projections.

2.1 Problem definition

We follow the notation and the concepts introduced by Lin et al. [18] and Chiu et al. [5]. In those papers, the concept of a *match*, *trivial match* and a *motif* is given for the case of a single time series. Those definitions can easily be extended to the case in which we are comparing two (and possibly more) series.

Definition 1. Match: *Given a positive real number R (called* range*), a subsequence P of a time series T_1, and subsequence Q of a time series T_2, we say that Q* matches *P if the distance D between them satisfies the inequality $D(P, Q) \leq R$.*

If the subsequence Q_j starting at position j in T_1 matches P, then it is very likely that the subsequence Q_{j+1} starting at position $j + 1$ also matches P. If this is the case,

then we call Q_2 a *trivial* match. As it is not desirable to clutter the dot plot with dots that corresponds to the same matching, we try to exclude them. A motif can be defined as any non trivial match. The problem of building dot plots for time series data could easily be reduced to the problem of searching for all possible motifs between two time series.

Definition 2. Time Series Dot Plot: *Let T_1 and T_2 be two time series of length m and n respectively. A* time series dot plot *of (T_1, T_2) is a $m \times n$ binary matrix A defined as follows. Let P be a subsequence in T_1 starting at position i and Q be a subsequence in T_2 starting at position j, then*

$$A[i,j] = \begin{cases} 1 & \textit{if } P \textit{ is a non trivial match of } Q \\ 0 & \textit{otherwise} \end{cases}$$

In the rest of the paper we will demonstrate examples of data sets from different application areas, which support our argument that this definition of the time series dot plots leads to highly descriptive and intuitive visual representation.

Before we leave this section, it is worth pointing out that so far we have not imposed any restriction on the length of the subsequences P and Q that we compare. That length could be specified to any number between one and the length of the time series according the preference of the domain expert. When the length is equal to one the dot plot turns into a recurrence plot. In Section 4 we show that in the cases in which we cannot decide the size of the sliding window (i.e., the subsequence length) or when the best length varies, a dynamically-changing window length could also be used.

2.2 Discretizing the Time Series

The first step of our dot plot building procedure is the discretization of the time series. It serves several important purposes. First, it provides us with a lower dimensional presentation that reduces the effect of the noise in the raw time series data and at the same time preserves its main properties. Second, it gives us a natural string presentation that will be used in the subsequent step by the PROJECTION algorithm. We choose the *Symbolic Aggregate approXimation* (SAX) [19] for our discretization step. Here we briefly discuss the general idea behind the SAX procedure. For a more detailed description we refer the reader to [19].

SAX accepts as input parameters the time series $P = p_1, p_2, \ldots, p_n$ that we want to discretize, the desired length w of the symbolic representation and the size $|\Sigma|$ of the alphabet to be used. First P is normalized to have a mean of zero and standard deviation one. Then, P is split into equal-length segments and the averages on all segments are computed producing $\bar{P} = \bar{p}_1, \bar{p}_2, \ldots, \bar{p}_w$ where

$$\bar{p}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} p_j \qquad (2)$$

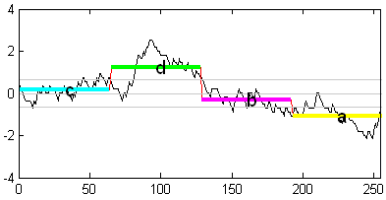


Figure 2. The symbolic representation of a time series obtained through SAX. The alphabet is $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$ and the word length is w=4.

This lower dimensional representation $\bar{P}$ is known as *Piecewise Aggregate Approximation* (PAA) [16]. The PAA representation is fast to compute and is demonstrated to be competitive with other more sophisticated techniques [14].

Once all $\bar{p}_i$ are computed they are quantized into $|\Sigma|$ intervals in such a way that approximately the same number of them fall into each interval. This guarantees equiprobable symbols in the final representation of the sequence, which is crucial for the PROJECTION algorithm. Finally SAX assigns the same letter to all $\bar{p}_i$ that belong to the same interval. Figure 2 illustrates the discretization of a time series using the SAX algorithm.

The experiments in [14] show that words of length 4-16 usually capture quite accurately the shape of the series and produce a precise estimate for the distances between the sequences. This is essential again for the PROJECTION algorithm as we will see in the following section.

2.3 Probabilistic Discovery of Time Series Motifs

We have shown that building the time series dot plots could be reduced to the problem of discovering time series motifs. In this section we focus our attention on motif discovery algorithms that satisfy the initial constraints for invariance, efficiency and robustness to noise.

In [5] the authors describe a method for probabilistic discovery of time series motifs, which turns out to meet all our requirements. The algorithm discovers approximate motifs of a specified length w, which occur frequently with up to $d < w$ mismatches. These motifs are called (w, d)-motifs. The d mismatches can be thought as "don't care" positions in the motif. These latter positions are of an essential utility in time series analysis as they allow robust handling of noise, scaling and translations. They also allow to ignore regions which according to the domain expert may be inessential for the inter-sequence comparisons [1, 15].

As the probabilistic motif discovery algorithm [5] is based on Buhler and Tompa's PROJECTION algorithm [3], we start by briefly explaining the planted motif problem and the idea of PROJECTION.

2.3.1 PROJECTION and the Planted Motif Problem

The algorithm that Buhler and Tompa proposed was intended as a computationally efficient alternative for solving Pevzner and Sze's planted (w, d)-motif problem [22].

Planted (w, d)-motif problem. INPUT: integers w and d, and t sequences $\{x_1, \ldots, x_t\}$ of length n generated at random, where symbols in the alphabet are equiprobable. Each sequence is planted with exactly one occurrence of an unknown motif y of length w. Each occurrence differs in exactly d positions from y. QUESTION: Given the $\{x_1, \ldots, x_t\}$, w and d find the unknown motif y.

The "challenge problem" that Pevzner and Sze proposed was the planted $(15, 4)$-motif problem with t=20 sequences of length n=600 over the four letter DNA alphabet. This problem turns to be very hard computationally as one planted occurrence corresponds to several potential motifs. As a consequence, even if we guessed the correct positions of all planted occurrences it may still not be enough to reconstruct the motif y.

Buhler and Tompa showed that the $(15, 4)$- as well as a set of other difficult problems like the $(14, 4)$-, $(16, 5)$- and $(18, 6)$-motif problems, could be successfully approached by using a probabilistic scheme. Their algorithm belongs to the family of *locality-preserving hashing* methods. A property of these methods is that they hash multidimensional points that are close to each other to values that are also likely to be close to each other in a lower dimensionality space [13]. It is important to mention however, that these methods perform well when the number of dimensions is comparatively small, e.g. between ten and twenty.

PROJECTION applies a set of hash functions to all substrings of length w. Each function splits the set of substrings into a number of classes of equivalence. When all functions are applied if in some of those classes there is more than a predefined number of strings hashed, then with a very high probability these strings correspond to the planted motif.

2.3.2 Projection for Time Series Dot Plots

Once we have the symbolized sequences we need to find out which of them correspond to approximate motifs. To illustrate the procedure consider the following example. Assume that we compare two time series T_1 and T_2 of length m=1024, using a sliding window of size n=128, word length w=4 and alphabet size $|\Sigma|$=3 (see Figure 3).

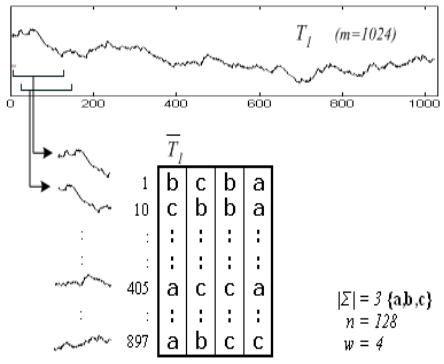


Figure 3. Time series T_1 and its symbolic representation $\bar{T}_1$. Time series length m=1024, sliding window n=128, word size w=4, alphabet size $|\Sigma|$=3

We obtain similar string table, as the one on Figure 3, for the second time series too (if we compare a series with itself, only one table is necessary). Obviously we do not need to keep all consecutive elements that symbolize to the same string. In our example, the second entry in the table corresponds to the tenth sequence, which means that sequences one to nine all map to the string `bcba`. This optimization is simply the run-length encoding compression.

Suppose that we would like to discover $(4, 0)$- and $(4, 1)$-motifs, i.e. motifs that allow up to d=1 don't care positions. If we project $k < w - d$ positions, it is very likely to obtain the same projection value for strings between which the Hamming distance is less or equal to d.

On Figure 4-*top* we have selected k=2 random positions and projected the strings on those positions (in the example the second and the fourth position were selected). All strings with the same projection are hashed to the same bin. We also keep a flag indicating to which time series the elements belong. Finally we scan the bins and all elements from the first time series are paired with all elements from the second series. For all pairs the counters in the corresponding cells in the collision matrix are set to one (see Figure 4-*Bottom*).

The described procedure is repeated m times and in each iterations new random projecting positions are selected and the counters in the collision matrix are being set or increased. After the last iteration a threshold s is applied, filtering all positions in the matrix that have a counter larger or equal to s. Finally a dot is placed on the corresponding locations of the dot plot.

2.3.3 Estimating the Number of Iterations

Separating the motifs into bins requires just one pass through the time series on each iteration. If the time series length is n, pairing the positions in the worst case could

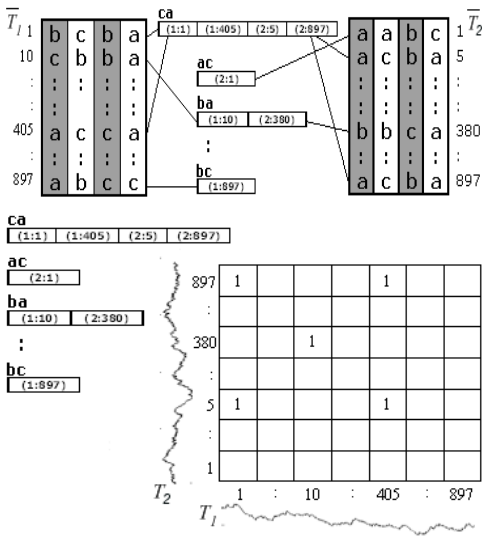


Figure 4. *Top:* **projection of the two time series and the corresponding bins.** *Bottom:* **elements from different time series that were hashed to the same bin are paired.**

have time complexity $O(n^2)$. For practical data sets, however, the number of pairs is clearly sub-quadratic. More precisely, the space and time required by the algorithm are respectively $O(|M|)$ and $O(m|M|)$, where $|M|$ is the number of elements in the sparse collision matrix and m is the number of iterations. Buhler and Tompa showed that, provided we know the other parameters, the required iterations for results within the 95% confidence threshold, could easily be estimated. Unfortunately this number may be very large, which increases unacceptably the time for building the plots. For example, if we use word size w=16, projection size k=7 and d=3 don't care symbols, the iterations to be performed would be m=132. For the same settings if we allow for d=5 don't care positions the iterations increase to m=3599.

Raphael et al. [24] suggested an optimization to PROJECTION in which rather than selecting k positions uniformly at random, the positions are selected in a way that would allow to sample the space of the projections more efficiently. This improvement is demonstrated to decrease the number of iterations required by PROJECTION to converge. However, we noticed that even with uniformly selected positions, the plots are very descriptive with far less iterations than those estimated by PROJECTION. Usually an order of ten iterations is sufficient to capture the information structure in the data. The reason for this is that the condition of detecting occurrences with *exactly* d differences is a bit over-restrictive for the task of time series motif finding. Instead, if the more relaxed condition of finding motifs of *up to* d differences is considered, then the estimated iterations are close to the values that lead to correctly defined plots. Another important feature to notice is that the projection size k is very essential for both the number of iterations and for filtering the results. Smaller values of k would obviously lead to smaller number of iterations, but would also make it harder to distinguish between occurrences with up to d and more than d differences. Based on these observations, we tried to design some guidelines for the following optimization problem. Find out k-s that are sufficient to filter the occurrences, and estimate for those projection sizes the minimum number of iterations required to discover the motifs with up to d differences.

Consider two pairs of sequences corresponding to positions (i_1, j_1) and (i_2, j_2). Let the sequences from the first pair have up to d differences and the sequences from the second one have more than d differences. Let also X_1 and X_2 are the numbers in the collision matrix at positions (i_1, j_1) and (i_2, j_2) after m iterations. X_1 and X_2 are binomially distributed with probability mass function equal to the probability of projecting the corresponding sequences to the same value in a single iteration. To compute this probability we first compute the probability that two strings have exactly d differences.

$$p_d = \binom{w}{d} \left(1 - \frac{1}{a}\right)^d \left(\frac{1}{a}\right)^{w-d} \tag{3}$$

where a is the size of the alphabet. The probability that two strings have up to d differences is

$$p_{all} = \sum_{i=0}^{d} p_i \tag{4}$$

The probability that two strings with d differences project to the same value is

$$\hat{p}_d = \frac{\binom{w-d}{k}}{\binom{w}{k}} \tag{5}$$

Finally, the probability that we will hash the sequences at position (i_1, j_1) to the same value is

$$p1 = \sum_{i=0}^{d} \frac{p_i}{p_{all}} \hat{p}_i \tag{6}$$

Similarly the probability mass function for X_2 is

$$p2 = \sum_{i=d+1}^{w-k} \frac{p_i}{1 - p_{all}} \hat{p}_i \tag{7}$$

Now if μ_x and σ_x are the mean and the standard deviation of X, we can express our problem with the system of inequalities:

$$\frac{\mu_{x_1} - \mu_{x_2}}{\sigma_{x_1} + \sigma_{x_2}} \geq \alpha_1 \tag{8}$$

$$\mu_{x_1} - \alpha_2 \sigma_{x_1} \geq 0 \tag{9}$$

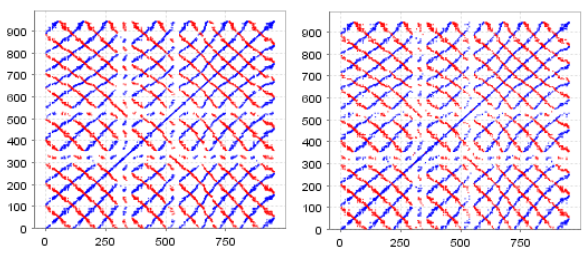


Figure 5. Dot plots for the synthetic time series from Figure 1. *Left:* **132 iterations have been used as estimated by** PROJECTION**.** *Right:* **15 iterations have been used. For both plots w=16, k=7, d=3**

Inequality 8 has the same structure as the standard t-test and gives us the statistical significance of the difference between the two distributions. We use it to identify the projection size and the number of iterations required in order to filter out motifs with more than d differences. The second inequality verifies that the number of iterations which we perform produces at least one hit in the collision matrix for most motifs with up to d differences. In our experiments we found out that values of α_1 greater than 1 usually separate well the two classes, and $1 \leq \alpha_2 \leq 2$ allows the detection of most of the existing motifs.

Just for comparison, if we use PROJECTION's estimate for w=16, d=3, k=7 as mentioned above we need to perform 132 iterations, while the new estimate suggest fifteen iterations. Figure 5 shows that the plots for both settings have exactly the same information structure.

2.3.4 On-Line Motif Discovery

The projection method exhibits the two essential properties of an on-line mining algorithm, namely good time *performance* and *updatability*. As we already mentioned the time complexity of the algorithm is $O(m|M|)$, where for realistic data sets M is very sparse, and m, as justified in the previous section, can be restricted to comparatively small values. The updatability property can also easily be achieved with the slight overhead of keeping all m hash tables that we have used so far to build the plot.

Assume that the user needs updates every l data points. Rather than recomputing the whole new matrix we could reuse the m hash tables and project the new l sequences on them. Of course we need to make sure that we have removed the elements corresponding to the first l time points that will now drop out of the plot. Now all we have to do is to find out the possible pairings between the already observed and the newly formed l sequences. Again we can come out with some theoretical cases when the new data will pair with all sequences so far, which will make the complexity of the update linear with respect to the time series length, but for realistic data sets the update will remain constant.

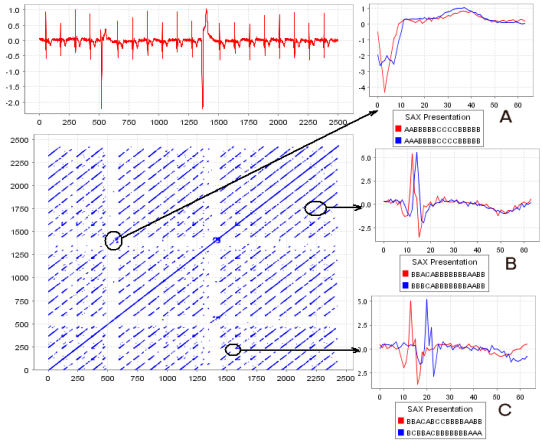


Figure 6. ECG data with V anomalies and the corresponding dot plot. *Right*: **detected and omitted motifs**

Having support for streaming data allows us to introduce the idea of applying the plots for monitoring purposes. Such an application could be of a great utility in different fields, e.g. monitoring cardio activities, seismological readings, observing the change in continuous natural phenomena as tides, winds, sun bursts, etc.

3 Experimental Results

We tested our tool on a number of datasets from different domains (e.g. medicine, industry, stock markets, natural phenomena, music), and with different characteristics. Some of the datasets are recurrent, some could be modeled as random walk, etc. We start by exploring the applicability of the tool for anomaly detection.

3.1 Dot Plots for Anomaly Detection

ECG data. Very often we search for anomalies in the recurrent behavior of a system. In these cases periodic occurrences of the pattern would be displayed as diagonals, and the anomaly in the occurrences will be manifested as a white cross on the plot.

Figure 6 shows the plot of an ECG dataset from the MIT-BIH Supraventricular Arrhythmia Database, part of the PhysioNet project [12] (We omit the reverse matches in the remaining plots, to make the regions of interest better observable). A word length w=16 and a maximum number of differences d=2 have been used. The two premature ventricular contractions could be identified on the dot plot by the corresponding white perpendicular lines. The anomaly

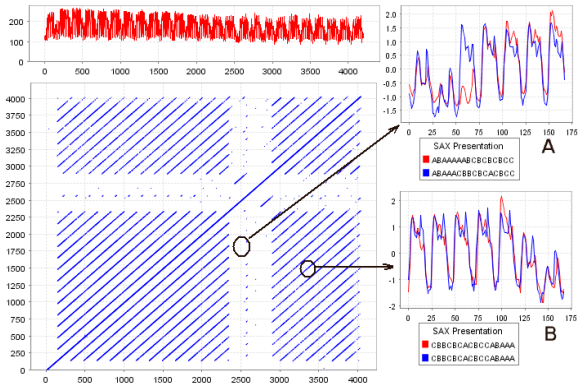


Figure 7. Power consumption in an Italian city during the first half of 1995

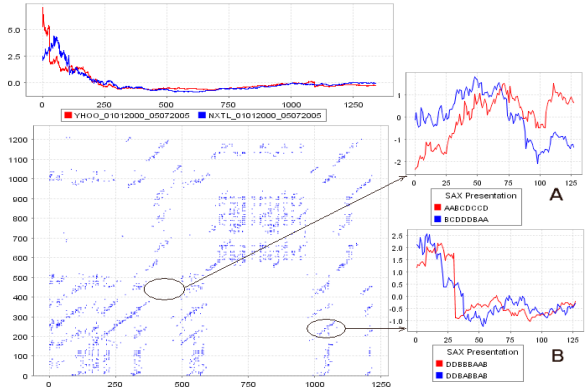


Figure 8. The daily stocks of Yahoo! and Nextel over a 5-year period. Though the time plots look very similar, the dot plot captures regions of anomalies(A) and patterns(B)

is comparatively easy to spot on the time plot too, so in this case the dot plot does not reveal much additional information. Still we include the dataset here as it demonstrates some interesting features of the tool.

Having a dot line that intercepts the white band of the anomaly shows that the two anomalies are of the same type. Figure 6-*A* demonstrates that both anomalies were symbolized to strings that are within the predefined distance d=2. Another interesting fact is that the selected approach proves to be quite robust in handling small local time warps and fluctuations. This is due to the combined effect of the PAA representation and the allowed don't care positions (see Figure 6-*B*). Unfortunately, larger time warps are more difficult to capture (see Figure 6-*C*). Most of the interceptions in the lines are due to the fact that the data though recurrent does not have a fixed length period. Therefore using a fixed sliding window causes many of the motifs to be out of phase. A solution that mitigates this effect is to prefer shorter sliding window sizes over the larger ones. An even better solution of using a dynamically changing window will be presented in Section 4.

Powerplant data. As pointed above it is comparatively easy to spot V anomalies on the ECG time plots. However, if the time plot is too dense then spotting the anomalies could be really hard. In those settings the dot plot is of a real utility as the anomaly can usually be observed much easier on it. Figure 7 shows the plot for the hourly power consumption in an Italian city during the first six months of 1995.

There are two anomalies depicted by our tool. The white band in the very beginning of the plot corresponds to the New Year's week. The other band is due to several official holidays during the second half of April (Easter and Liberation Day) and the beginning of May (Labor Day). In both cases the power consumption pattern for the week differs from the general weekly pattern, e.g., Figure 7-*A* shows a week with unusually small power consumption on Monday. As the size of the period for this data set is constant (one week corresponding to 168 samples) the fixed sliding window performs very well, and the diagonals for the patterns are not intercepted as in the case of the heart beat data.

3.2 Dot Plots for Pattern Detection

Stock Market Data. We explored the daily stock quotes of two companies, Yahoo! and Nextel Corporation, over a period of five years (the data were obtained from Yahoo!Finance). As the time plots of the two series were very similar (see Figure 8), we expected a dot plot with well defined, though slightly scattered due to the high volume of noise, main diagonal.

The plot, however, revealed some really interesting patterns and anomalies that were difficult if not impossible to spot on the time plots. As we were interested in not so trivial long term similarities we decided to use a sliding window of 128 time points, which is comparatively equal to the half year quotes of the companies. We observed that though the time plots seemed similar, the main diagonal of the dot plot was intercepted at several regions. One such region is shown on Figure 8-*A*. When we looked at the data for this region, we found out that it corresponds to the second half of 2001 and the beginning of 2002. During this period after an unstable quarter for both companies Yahoo! recovered while Nextel's stock continued dropping steadily.

Also surprising were the many distinct diagonals, parallel to the main one, indicating similar stock movements for unaligned time frames. The diagonal on Figure 8-*B* for example corresponds to the first two quarters of 2001 for Nextel and the second and the third quarter of 2004 for Yahoo!. Both stocks dropped twice for this period though for

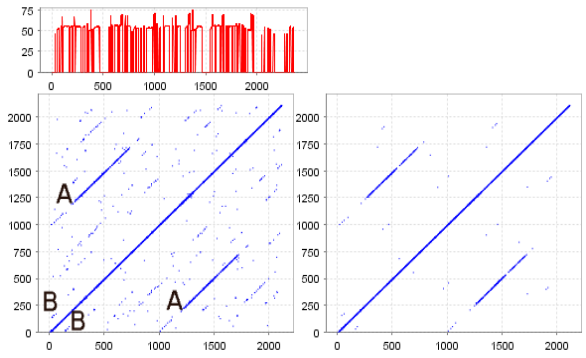


Figure 9. A 23sec sample of "Jingle Bells". Repetitions as the whole chorus (*A*) and *"jingle bells, jingle bells"* (*B*) could be observed on the dot plot. *Left:* Word of size w=14 is used. *Right:* Word of size w=16 is used.

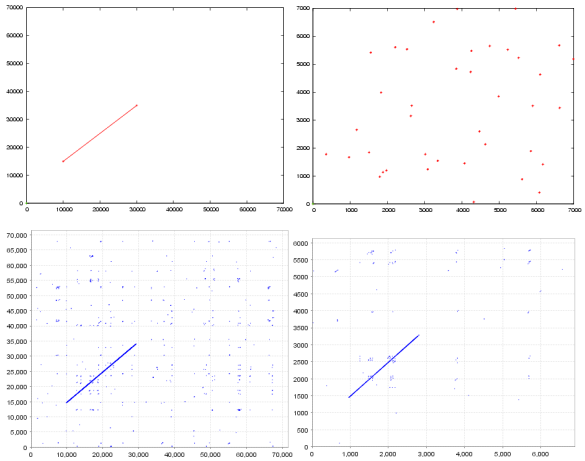


Figure 10. *Top*: The plots generated by MUMer for the original and the downsampled DNA strings. *Bottom*: The dot plots for the original and the downsampled X-projections.

different reasons, devaluation for Nextel and a 2:1 split for Yahoo!.

Audio data. Another interesting application of the tool is the visual mining of common tunes in audio streams.

We proceed by first applying *pitch extraction* on the data. Inspecting directly the pitch time series again does not reveal much information about the interesting patterns in the signal, see Figure 9-*top*. The figure demonstrates a short sample from "Jingle Bells". Regions marked with *A* on the dot plot clearly depict the repetition of the chorus, which comprised approximately one third of the sample. Using larger word sizes or reducing the number of allowed differences filters the noise on the plot even further but it also deteriorates the patterns that were not exactly identical or comparatively short like those from regions *B* (the repetition in the *"jingle bells, jingle bells"* tune).

Discrete data. Sometimes though the sequences are expressed over a finite alphabet it may still be beneficial to convert them into real value time series. Consider for example two very similar DNA strings that differ only in several *point mutations*. One or few base pairs have been inserted or deleted, causing identical frames from the two sequences to have different positions. Now if we want to downsample the strings, different base pairs from the homologous frames might be selected in the samples. Figure 10 illustrates the phenomenon.

In this example we have generated two random strings over the DNA alphabet and we have inserted a common pattern in both of them but at an odd and an even position respectively. Figure 10-*top* illustrates how MUMer [17] performs on the problem. While the pattern is detected on the original strings, on the downsampled versions it is not identified regardless of the size of the *mums* (Maximal Unique Matches) used. At the same time, if we convert the DNA string to a real value time series and apply our tool, we can clearly identify the patterns on both the original and the downsampled series. For the example only the X-projections from the 3D DNA representation suggested by [4] were used.

4 Dynamic Sliding Window

As already pointed out, using a fixed sliding window could lead to interceptions, implying anomalies in the recurrent states, while such anomalies may not really exist. The reason for this is that often two sequences will represent the same pattern but one of them will be a warped version of the other. Now we will demonstrate how this information could be included in the plot. To approach the problem we employ a technique in which the sliding window resizes dynamically according to the bounds of the recurrent patterns. Finding those bounds is not trivial and is closely related to the problem of segmenting time series data.

Here we do not try to solve the problem of segmentation as we consider it an orthogonal and tightly data and task dependent issue. Instead we simply demonstrate that if we select an appropriate, for our task and data, segmentation algorithm then the dynamic window technique would produce more accurate dot plots than the fixed window one. To check this assumption we applied several simple *change point* detection heuristics. In the first heuristics we select as change points the points that have higher than a predefined threshold deviation. A similar model, though with more complex evaluation, is used for example in [23] where a *dominant class* is identified first and at each time instant the deviation from this class is measured. The result is presented on Figure 11-*left*. The synthetic dataset on the figure

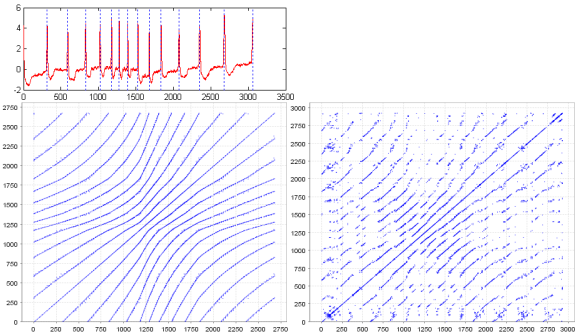


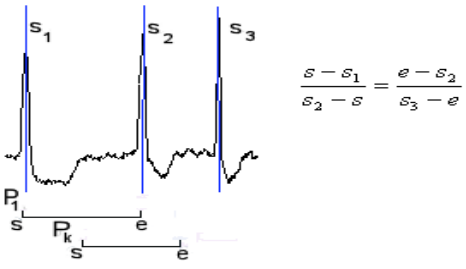
Figure 11. *Left*: **Dynamic sliding window plot for the synthetic ECG data set.** *Deviation* **heuristics is used for the segmentation.** *Right*: **The corresponding fixed sliding window plot**

was obtained from a real ECG time series by downsampling some of the heartbeat periods, and thus simulating a change in the frequency of the heartbeat rate.

The blue vertical lines on the time plot indicate the segment bounds selected by the heuristics. Once the bounds are identified the dynamic window algorithm proceeds as follows. Let $s_1, s_2, ..., s_k$ are the segment bounds for the time series. If s and e are the start and the end position of the dynamic sliding window, we begin by setting $s = s_1$ and $e = s_2$ (see Figure 12). At each time instant we move s one point to the right and then adjust e to preserve the ratio in which s divides the previous segment, i.e. if s divides the segment (s_i, s_{i+1}) in ratio $\frac{a}{b}$, then we make e to be the point that divides (s_{i+1}, s_{i+2}) as $\frac{a}{b}$ too. Finally, all of the subsequences though with a different length, we symbolize using the same word length.

On the synthetic heartbeat dot plots we can see that the dynamic sliding window keeps the similarity lines unintercepted as opposed to the fixed sliding window. The fact that the recurrent states are similar with a certain amount of time warping is also indicated by the curvature of the similarity lines.

The tide data set (Figure 13) we have segmented using



$$\frac{s - s_1}{s_2 - s} = \frac{e - s_2}{s_3 - e}$$

Figure 12. The dynamic window P_i augments and shrinks according to the segment bounds.

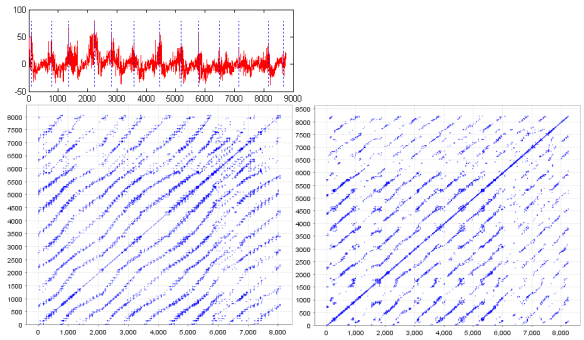


Figure 13. Larger period of the recurrent states leads to larger variability of the state sizes. *Left*:**Dynamic sliding window plot for the tide data set.** *Extremum* **heuristics is used for the segmentation.** *Right* **The corresponding fixed sliding window plot**

even simpler heuristics. For a predefined threshold l we mark as segment bounds the data points that are global maximums for intervals of length l. This approach could be suitable again when the recurrent states start or finish with distinct extremum values and when we would like to restrict the amount of allowed warping within certain bounds. For datasets where the average size of the recurrent states is comparatively small the dynamic sliding window technique usually does not preserve a lot more information compared to the fixed window one. However, if the average period size is larger (e.g. the tide time series), then larger variances in the state sizes are expected. For those cases the dynamic window technique creates far more accurate and descriptive plots.

5 Conclusion

We studied the problem of building dot plots for real value time series data. We reduced this problem to the problem of discovering motifs in time series. Based on this observation, we developed an efficient dot plotting tool which exploits the random projection algorithm to solve the approximate time series motif finding.

The usefulness of the tool was illustrated in several tasks, mainly in anomaly detection and pattern recognition on a number of data sets from different areas and with different characteristics. The scalability and the updatability of the algorithm allowed us to apply it for live monitoring of time series data. Finally, we described a dynamic sliding window approach based on time series segmentation. In this approach the window changes adaptively allowing us to create more descriptive dot plots for recurrent time series data when the size of the recurrent states varies significantly.

References

[1] Agrawal, R., Lin, K. I., Sawhney, H. S., Shim, K., "Fast similarity search in the presence of noise, scaling, and translation in time-series Databases", 21st VLDB conference, 1995

[2] Bernstein, M., Bolter, J. D., Joice, M., Mylonas, E., "Architecture for volatile hypertext", 3rd annual ACM conference on Hypertext, 1991

[3] Buhler, J., Tompa, M., "Finding motifs using random projections", Jour. of Comp. Biology, 9:225–242, 2002

[4] Chang, H.T., Lo, N.W., Lu, W.C., Kuo, C.J., "Visualization and comparison of DNA sequences by use of three-dimensional trajectories", First Asia-Pacific bioinformatics conference on Bioinformatics, 2003

[5] Chiu, B., Keogh, E., Lonardi, S., "Probabilistic discovery of time series motifs", 9th ACM SIGKDD international conference on Knowledge discovery and data mining, 2003

[6] Church, K. W., Helfman, J. I., "Dotplot: a program for exploring self-similarity in millions of lines of text and code", American Statistical Association, Institute for Mathematical Statistics and Interface Foundations of North America, 2(2):153–174, 1993

[7] Church, K. W., "Char_align: a program for aligning parallel texts at the character level", 31st Annual Meeting of the Association for Computational Linguistics, 1993

[8] Das, G., Lin, K., Mannila, H., Renganathan, G., Smyth, P., "Rule discovery from time series", 4th International conference on Knowledge discovery and data mining, 1998

[9] Dasgupta, D., Forrest, S., "Novelty detection in time series data using ideas from immunology", 5th International Conference on Intelligent Systems, 1996

[10] Eckmann, J. P., Kamphorst, S. O., Rulle, D., "Recurrence plots of dynamical systems", Europhys. Lett., 4(9):973–977, 1987

[11] Gibbs, A. J., McIntyre, G. A., "The diagram, a method for comparing sequences", Eur. Jour. Biochem, 16(1):1–11, 1970.

[12] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals". Circulation 101(23):e215-e220, 2000

[13] Indyk, P., Motwani, R., Raghavan, P., Vempala, S., "Locality-preserving hashing in multidimensional spaces", 29th annual ACM symposium on Theory of computing, 1997

[14] Keogh, E., Kasetty, S., "On the need for time series data mining benchmarks: a survey and empirical demonstration", 8th ACM SIGKDD international conference on Knowledge discovery and data mining, 2002

[15] Keogh, E., Pazzani, M., "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback", 4th International conference on Knowledge discovery and data mining, 1998

[16] Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S., "Dimensionality reduction for fast similarity search in large time series databases", Jour. of Knowledge and Information Systems, 263–286, 2000

[17] Kurtz, S., Phillippy, A., Delcher, A.L., Smoot, M., Shumway, M., Antonescu, C., Salzberg, S.L., "Versatile and open software for comparing large genomes", Genome Biology, 5:R12, 2004

[18] Lin, J., Keogh, E., Lonardi, S., Patel, P., "Finding motifs in time series", 8th ACM SIGKDD international conference on Knowledge discovery and data mining, 2002

[19] Lin, J., Keogh, E., Lonardi, S., Chiu, B., "A symbolic representation of time series, with implications for streaming algorithms", 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, 2003.

[20] Maizel, J.V., Lenk, R.P.,"Enhanced graphic matrix analysis of nucleic acid and protein sequences", Proc. Natl. Acad. Sci. USA, 78(12):7665–7669, 1981

[21] Oates, T., Schmill, M. D., Cohen, P. R., "A method for clustering the experiences of a mobile robot that accords with human judgments", 17th National Conference on Artificial Intelligence, 2000

[22] Pevzner, P. A., Sze, S. H., "Combinatorial Approaches to Finding Subtle Signals in DNA Sequences", 8th International Conference on Intelligent Systems for Molecular Biology, 269–278, 2000

[23] Radhakrishnan, R., Xiong, Z., Divakaran, A., Memon, N., "Time series analysis and segmentation using eigenvectors for mining semantic audio label sequences", IEEE International Conference on Multimedia and Expo, 2004

[24] Raphael, B., Liu, L.T., Varghese, G., "A uniform projection method for motif discovery in in DNA sequences", IEEE/ACM Transactions on Computational Biology and Bioinformatics, 1:91–94, 2004