Time Series Data Mining: A Unifying View

Eamonn Keogh
University of California, Riverside
Riverside, California, USA
eamonn@cs.ucr.edu

ABSTRACT

Time series data are ubiquitous; large volumes of such data are routinely created in scientific, industrial, entertainment, medical and biological domains. Examples include ECG data, gait analysis, stock market quotes, machine health telemetry, search engine throughput volumes etc. VLDB has traditionally been home to much of the community's best research on time series, with three to eight papers on time series appearing in the conference each year.

What do we want to do with such time series? Everything! Classification, clustering, joins, anomaly detection, motif discovery, similarity search, visualization, summarization, compression, segmentation, rule discovery etc. Rather than a deep dive in just one of these subtopics, in this tutorial I will show a surprisingly small set of high-level representations, definitions, distance measures and primitives can be combined to solve the first 90 to 99.9% of the problems listed above. The tutorial will be illustrated with numerous real-world examples created just for this tutorial, including examples from robotics, wearables, medical telemetry, astronomy, and (especially) animal behavior. Moreover, all sample datasets and code snippets will be released so that the tutorial attendees (and later, *readers*) can first reproduce the results demonstrated, before attempting similar analysis on their data.

PVLDB Reference Format:
Eamonn Keogh. Time Series Data Mining: A Unifying View. PVLDB, 16(12): 3861 - 3863, 2023.
doi:10.14778/3611540.3611570

1 INTRODUCTION

Time series data are ubiquitous; large volumes of such data are routinely created in scientific, industrial, entertainment, medical and biological domains. VLDB has traditionally been a home to much of the community's research on time series, with three to eight papers on time series appearing in the conference each year [4][28][29][30].

There are many tasks of interest that we may wish to perform with time series, including: classification [1][11][24][26], clustering, joins [1][2][14][15], anomaly detection [1][23][27], motif discovery [1][5][6][9][10][13][17][18][20][21], similarity search [11][32], visualization [3][25], summarization [12], compression [12], segmentation [8], data labeling [15], evolving pattern discovery [7][24], and rule discovery [4].

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611570

Clearly, any one of those tasks could be the subject of a tutorial. However, this tutorial has a different ambition, and makes a strong and concrete claim. Given just two simple tools, and perhaps a handful of lines of minor supporting code, you can solve somewhere between 90 to 99.9% of all these problems. The fact that such a small number of atomic ideas can be combined to solve so many diverse problems may surprise some, but it is an increasingly prevalent idea in the data mining literature. The two ideas in question are:

- The MASS algorithm [32].
- The Matrix Profile [1][2][10].

Both of these elements are now mature enough to be considered "command line" elements in most common data analytic environments (for concreteness, the tutorial will show MATLAB® snippets). I will show that using these tools with a single command line query, it is often possible outperform very complex methods on a dozen important time series processing tasks.

2 CORE PRIMATIVES FOR TIME SERIES

The MASS algorithm has been shown to be optimally fast, assuming only that the FFT algorithm is optimally fast. However, the situation is less clear for the Matrix Profile. If you only want the maximum value from the Matrix Profile, i.e., the time series *discords*, then DAMP provides an ultra scalable to find them [23].

The situation is less clear if you wish this report the time series *motifs*. A handful of papers claim to report faster methods [33], but they are all based on a naïve misunderstanding or misrepresentation.

Consider the following two time series, which are at extremes of difficulty for motif discovery. First we have *R*, an complete random vector, and *E*. The time series *E* is simply R, into which we have placed two identical copies of some pattern. It is easy to see that finding the closest pair of subsequences in *R* is an intrinsically $O(|R|^2)$ problem, whereas finding the closest pair of subsequences in *E* is just $O(|E|)$ problem.

It is possible to propose algorithms that are apparently fast, if the data searched almost E, that is to say, strong motifs in data that is otherwise not well conserved. However, such algorithms fail dramatically for even slightly harder problems (by fail, we mean take much longer than brute force search). In the tutorial we will show examples of this, but consider this simple example.

On a problem in *E* of length 100,000, Attimo takes 1.8 seconds and SCRIMP++ [10] takes 19.5 seconds. While the former is only approximately correct and the latter is exact, this is a large time difference. However, suppose we consider a harder problem. For example, *R2*:

```
rng(2023); % set random seed
R2 = rand(1,100000);
R2(10001:10128) = TS(20001:20128)+ randn(1,128)/10;
```

In the last line we have inserted a weak motif in order to to have a known ground truth (see also *Time Series Sanity Checks* below). For this task, SCRIMP++ *again* takes 19.5 seconds, but Attimo takes 4.6 hours! In fact, as we will show in this tutorial, the apparent advantage of Attimo was illusionary. By exploiting the *anytime algorithm* properties of SCRIMP++ and the more general *contract algorithm* properties of the Matrix Profile (which takes a single line of code!), we find the motif in 0.1 seconds.

2.1 Time Series Sanity Checks

More than almost any other datatype, time series is an inherently visual data type. In this tutorial I will explain how we can exploit this fact to test hypotheses, perform sanity checks and convince a reader (an employer, a grant review panel, journal editors etc.) of a finding or viewpoint. For concreteness I will give a simple example.

It can be useful to transform a raw time series into a piecewise linear approximation (PLA), either as a form of data compression or for feature extraction. The task is often phrased as: Given a time series **T**, approximate it with *K* segments ($K << |\mathbf{T}|$), minimizing the RMSE. However, there a several essentially identical variants that minimize the maximum per segment error etc.

The optimal algorithm for this is known, but computationally untenable, therefore several fast approximate algorithms have been suggested, dating back at least thirty years.

How can we know if we have a good algorithm, if we cannot compute the optimal solution? There is in fact a simple "trick", we can create a toy dataset for which we know the optimal solution by out-of-band knowledge and test our algorithm on it. Because such toy datasets are necessarily special and a limited subset of the space of possible time series, success on them should only be taken as weak evidence that our algorithm is useful, but failure on them should be taken as *strong* evidence that our algorithm is worthless. In Figure 1 we create such an example.

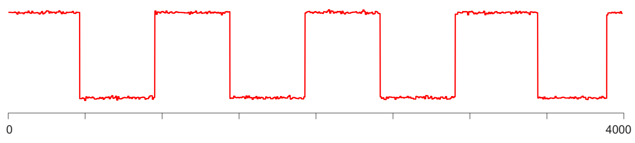


Figure 1: A toy dataset for which the optimal PLA for $K = 9$ and $K = 8$ is obvious.

The reader will appreciate that with $K = 9$, the optimal solution is that each constant region, including the short one at the right side), be approximated by its own segment. Moreover, doing this will give us a near zero RMSE. For $K = 8$, the reader will further appreciate that two of the above segments must be merged, and it is obvious that the last two segments can be merged to give the smallest overall RMSE.

In Figure 2 we segment the toy dataset with the Bottom-Up algorithm, which is at least thirty-five years old, but best known through its reference in [31].

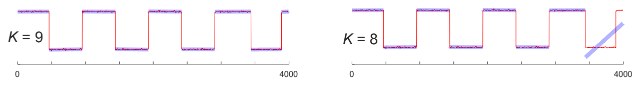


Figure 2: The toy dataset segmented with the Bottom-Up algorithm (a special case of SWAB), with $K = 9$ and $K = 8$.

At least in this case, the classic algorithm performs perfectly. By way of contrast, let us consider a more recently proposed algorithm, Sim-Piece [28]. With Sim-Piece you do not control the *K* directly, but by adjusting the error threshold parameter ε we can make it produce different *K* approximations. In Figure 3: The toy dataset segmented my Sim-Piece with $K = 9$ (ε =20%) and $K = 8$ (ε =47.5%).

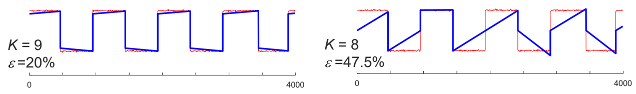
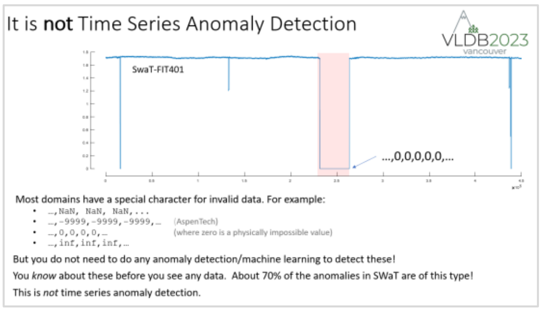


Figure 3: The toy dataset segmented my Sim-Piece with $K = 9$ (ε =20%) and $K = 8$ (ε =47.5%).

At a glance we see the value of visualizing results (amazingly, the original paper does not show a single visualization [28]).

There are many other examples where simply visualizing the time series would make us realize that there is a huge gap between the apparent and actual utility of an idea. For example, many papers on TSAD report apparently impressive results in tables, but never show examples of the anomalies discovered. In Figure 4 I show an example of a slide that discusses this issue. As the reader will appreciate, the "anomaly" discovered is trivial, and surely not worthy of a complex algorithm.



It is **not** Time Series Anomaly Detection

Most domains have a special character for invalid data. For example:
- …,NaN, NaN, NaN,...
- …,-9999,-9999,-9999,… (AspenTech)
- …,0,0,0,0,… (where zero is a physically impossible value)
- …,inf,inf,inf,…

But you do not need to do any anomaly detection/machine learning to detect these!

You *know* about these before you see any data. About 70% of the anomalies in SWaT are of this type!

This is *not* time series anomaly detection.

Figure 4: A sample slide from the tutorial that discusses TSAD.

The tutorial is replete with such examples of the power of *visualizing* time series, and every example will have all code, data, and provenance to let the attendees reproduce the findings.

ACKNOWLEDGMENTS

This work was supported by NSF 2103976 and gifts from MERL LABs, Amazon Research and Google Research. All opinions expressed are by the authors only. My brilliant students helped me to prepare some of the materials presented.

PRESENTER'S BIOGRAPHY

Dr. Keogh is a Distinguished Professor of Computer Science at the University of California. He is the inventor of many of the most commonly used time series data mining primitives, including PAA, the Matrix Profile, LB_{keogh}, SAX, Time Series Motifs and Time Series Shapelets. He has won numerous awards, including: The Bell Labs Bronze Prize 2021, the ACM SIGKDD 2022 Test of Time Paper Award, the 2021 IEEE ICDM Research Contributions Award, Two Google Faculty Awards and best paper awards at SIGKDD(2), SIGMOD, ICDM(2) and SDM.

REFERENCES

[1] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, Eamonn J. Keogh: Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. ICDM 2016: 1317-1322.

[2] Yan Zhu, Zachary Zimmerman, Nader Shakibay Senobari, Chin-Chia Michael Yeh, Gareth J. Funning, Abdullah Mueen, Philip Brisk, Eamonn J. Keogh: Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins. ICDM 2016: 739-748.

[3] Chin-Chia Michael Yeh, Helga Van Herle, Eamonn J. Keogh: Matrix Profile III: The Matrix Profile Allows Visualization of Salient Subsequences in Massive Time Series. ICDM 2016: 579-588.

[4] Chin-Chia Michael Yeh, Nickolas Kavantzas, Eamonn J. Keogh: Matrix Profile IV: Using Weakly Labeled Time Series to Predict Outcomes. Proc. VLDB Endow. 10(12): 1802-1812 (2017)

[5] Hoang Anh Dau, Eamonn J. Keogh: Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. KDD 2017: 125-134.

[6] Chin-Chia Michael Yeh, Nickolas Kavantzas, Eamonn J. Keogh: Matrix Profile VI: Meaningful Multidimensional Motif Discovery. ICDM 2017: 565-574

[7] Yan Zhu, Makoto Imamura, Daniel Nikovski, Eamonn J. Keogh: Matrix Profile VII: Time Series Chains: A New Primitive for Time Series Data Mining (Best Student Paper Award). ICDM 2017: 695-704

[8] Shaghayegh Gharghabi, Yifei Ding, Chin-Chia Michael Yeh, Kaveh Kamgar, Liudmila Ulanova, Eamonn J. Keogh: Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. ICDM 2017: 117-126.

[9] Yan Zhu, Abdullah Mueen, Eamonn J. Keogh: Matrix Profile IX: Admissible Time Series Motif Discovery With Missing Data. IEEE Trans. Knowl. Data Eng. 33(6): 2616-2626 (2021)

[10] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, Eamonn J. Keogh: Matrix Profile XI: SCRIMP++: Time Series Motif Discovery at Interactive Speeds. ICDM 2018: 837-846.

[11] Shaghayegh Gharghabi, Shima Imani, Anthony J. Bagnall, Amirali Darvishzadeh, Eamonn J. Keogh: Matrix Profile XII: MPdist: A Novel Time Series Distance Measure to Allow Data Mining in More Challenging Scenarios. ICDM 2018: 965-970.

[12] Shima Imani, Frank Madrid, Wei Ding, Scott E. Crouter, Eamonn J. Keogh: Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining. ICBK 2018: 382-389.

[13] Zachary Zimmerman, Kaveh Kamgar, Nader Shakibay Senobari, Brian Crites, Gareth J. Funning, Philip Brisk, Eamonn J. Keogh: Matrix Profile XIV: Scaling Time Series Motif Discovery with GPUs to Break a Quintillion Pairwise Comparisons a Day and Beyond. SoCC 2019: 74-86

[14] Kaveh Kamgar, Shaghayegh Gharghabi, Eamonn J. Keogh:Matrix Profile XV: Exploiting Time Series Consensus Motifs to Find Structure in Time Series Sets. ICDM 2019: 1156-1161

[15] Frank Madrid, Shailendra Singh, Quentin Chesnais, Kerry Mauck, Eamonn J. Keogh: Matrix Profile XVI: Efficient and Effective Labeling of Massive Time Series Archives. DSAA 2019: 463-472

[16] Zachary Zimmerman, Nader Shakibay Senobari, Gareth J. Funning, Evangelos E. Papalexakis, Samet Oymak, Philip Brisk, Eamonn J. Keogh: Matrix Profile XVIII: Time Series Mining in the Face of Fast Moving Streams using a Learned Approximate Matrix Profile. ICDM 2019: 936-945

[17] Shima Imani, Eamonn J. Keogh: Matrix Profile XIX: Time Series Semantic Motifs: A New Primitive for Finding Higher-Level Structure in Time Series. ICDM 2019: 329-338

[18] Frank Madrid, Shima Imani, Ryan Mercer, Zachary Zimmerman, Nader Shakibay Senobari, Eamonn J. Keogh: Matrix Profile XX: Finding and Visualizing Time Series Motifs of All Lengths using the Matrix Profile. ICBK 2019: 175-182

[19] Makoto Imamura, Takaaki Nakamura, Eamonn J. Keogh: Matrix Profile XXI: A Geometric Approach to Time Series Chains Improves Robustness. KDD 2020: 1114-1122

[20] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Eamonn J. Keogh: Matrix Profile XVII: Indexing the Matrix Profile to Allow Arbitrary Range Queries. ICDE 2020: 1846-1849

[21] Sara Alaee, Kaveh Kamgar, Eamonn J. Keogh: Matrix Profile XXII: Exact Discovery of Time Series Motifs under DTW. ICDM 2020: 900-905.

[22] Ryan Mercer, Sara Alaee, Alireza Abdoli, Shailendra Singh, Amy C. Murillo, Eamonn J. Keogh: Matrix Profile XXIII: Contrast Profile: A Novel Time Series Primitive that Allows Real World Classification. ICDM 2021: 1240-1245

[23] Yue Lu, Renjie Wu, Abdullah Mueen, Maria A. Zuluaga, Eamonn J. Keogh: Matrix Profile XXIV: Scaling Time Series Anomaly Detection to Trillions of Datapoints and Ultra-fast Arriving Data Streams. KDD 2022: 1173-1182.

[24] Ryan Mercer, Eamonn J. Keogh: Matrix Profile XXV: Introducing Novelets: A Primitive that Allows Online Detection of Emerging Behaviors in Time Series. ICDM 2022: 338-347.

[25] Maryam Shahcheraghi, Ryan Mercer, João Manuel De Almeida Rodrigues, Audrey Der, Hugo Filipe Silveira Gamboa, Zachary Zimmerman, Eamonn J. Keogh: Matrix Profile XXVI: Mplots: Scaling Time Series Similarity Matrices to Massive Data. ICDM 2022: 1179-1184.

[26] Audrey Der, Chin-Chia Michael Yeh, Renjie Wu, Junpeng Wang, Yan Zheng, Zhongfang Zhuang, Liang Wang, Wei Zhang, Eamonn J. Keogh: Matrix Profile XXVII: A Novel Distance Measure for Comparing Long Time Series. ICKG 2022: 40-47.

[27] Sadaf Tafazoli and Eamonn Keogh: Matrix Profile XXVIII: Discovering Multi-Dimensional Time Series Anomalies with *K* of *N* Anomaly Detection. SIAM SDM 2023.

[28] Xenophon Kitsios, Panagiotis Liakos, Katia Papakonstantinopoulou, Yannis Kotidis: Sim-Piece: Highly Accurate Piecewise Linear Approximation through Similar Segment Merging. Proc. VLDB Endow. 16(8): 1910-1922 (2023).

[29] Sebastian Schmidl, Phillip Wenig, Thorsten Papenbrock: Anomaly Detection in Time Series: A Comprehensive Evaluation. Proc. VLDB Endow. 15(9): 1779-1797 (2022).

[30] Li Yan, Nerissa Xu, Guozhong Li, Sourav S. Bhowmick, Byron Choi, Jianliang Xu: SENSOR: Data-driven Construction of Sketch-based Visual Query Interfaces for Time Series Data. Proc. VLDB Endow. 15(12): 3650-3653 (2022).

[31] Eamonn J. Keogh, Selina Chu, David M. Hart, Michael J. Pazzani: An Online Algorithm for Segmenting Time Series. ICDM 2001: 289-296.

[32] Abdullah Mueen, Yan Zhu, Chin-Chia Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. The fastest similarity search algorithm for time series subsequences under euclidean distance. http://www.cs.unm. edu/~mueen/FastestSimilaritySearch.html

[33] Matteo Ceccarello, Johann Gamper: Fast and Scalable Mining of Time Series Motifs with Probabilistic Guarantees. Proc. VLDB Endow. 15(13): 3841-3853 (2022)

Appendix A: The Contract Algorithm Property of the Matrix Profile

For all the Matrix Profile (MP) algorithms, SCRIMP, STAMP, STOMP, DAMP etc., we have a very nice property, they are all their own approximation algorithms! Moreover, you can precisely tune the approximation to use *exactly* the amount of computational time you have available.

Let us see this with a case study. A paper [33] considers a time series **C19** of length $n = 26{,}415{,}043$ and a motif length $m = 18{,}000$, very daunting numbers. If this is an important dataset, we may be willing to wait a day or even a week for the results, but surely, we will not wait for years. How long will this take? We can predict how long it will take?

The time required is the square of the length of the time series, divided by some constant C. That constant depends on your machine but can easily be empirically determined.

$$\text{TimeRequired} = n^2/C$$

Which gives us:

$$C = n^2/\ \text{TimeRequired}$$

Note that the time is independent of m. Thus, we can choose how long we wish to spend on motif hunting.