Abdullah Mueen

Eamonn Keogh

Time Series Data Mining Using the Matrix Profile: A Unifying View of Motif Discovery, Anomaly Detection, Segmentation, Classification, Clustering and Similarity Joins

KDD2017

We will start at 8:25am to allow stragglers to find the room

To get these slides in PPT or PDF, go to www.cs.ucr.edu/~eamonn/MatrixProfile.html

This tutorial is based on work by:

Chin-Chia Michael Yeh, **Yan Zhu**, **Abdullah Mueen**

Nurjahan Begum, Yifei Ding, Hoang Anh Dau

Diego Furtado Silva, Liudmila Ulanova, Eamonn Keogh

Zachary Zimmerman, Nader S. Senobari, Philip Brisk

Shaghayegh Gharghabi, Kaveh Kamgar

..and others that have inspired us, forgive any omissions.

This tutorial is funded by:

- NSF IIS-1161997 II
- NSF IIS 1510741
- NSF 544969
- CNS 1544969
- SHF-1527127
- AFRL FA9453-17-C-0024

Any errors or controversial statements are due solely to Mueen and Keogh

Disclaimer:

Time series is an inherently *visual* domain, and we exploit that fact in this tutorial.

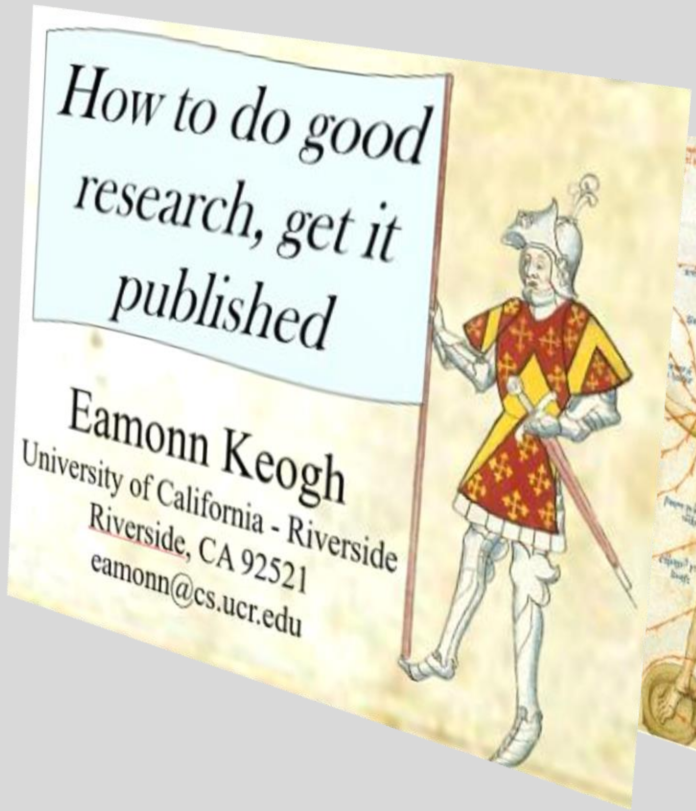
We therefore keep formal notations and proofs to an absolute minimum.

If you want them, you can read the relevant papers [a]

All the datasets used in tutorial are freely available, all experiments are reproducible.

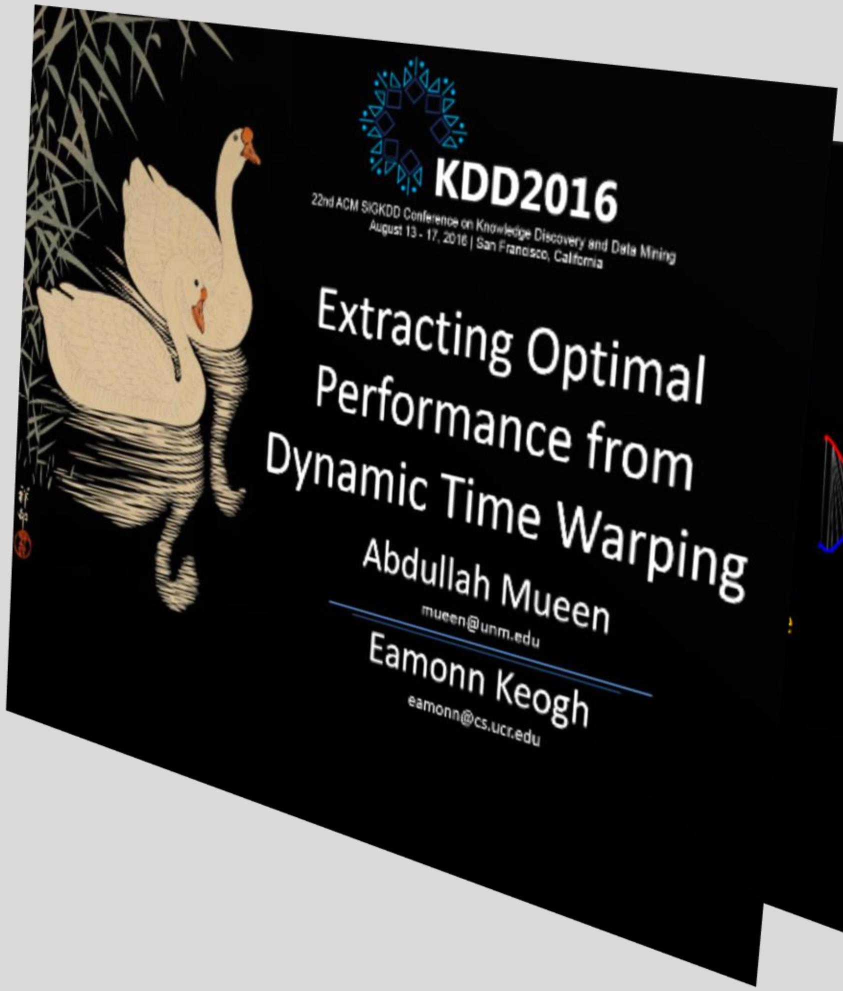
[a] www.cs.ucr.edu/~eamonn/MatrixProfile.html

If you enjoy this tutorial, please check out our other tutorials..



How to do good research, get it published

Eamonn Keogh
University of California - Riverside
Riverside, CA 92521
eamonn@cs.ucr.edu

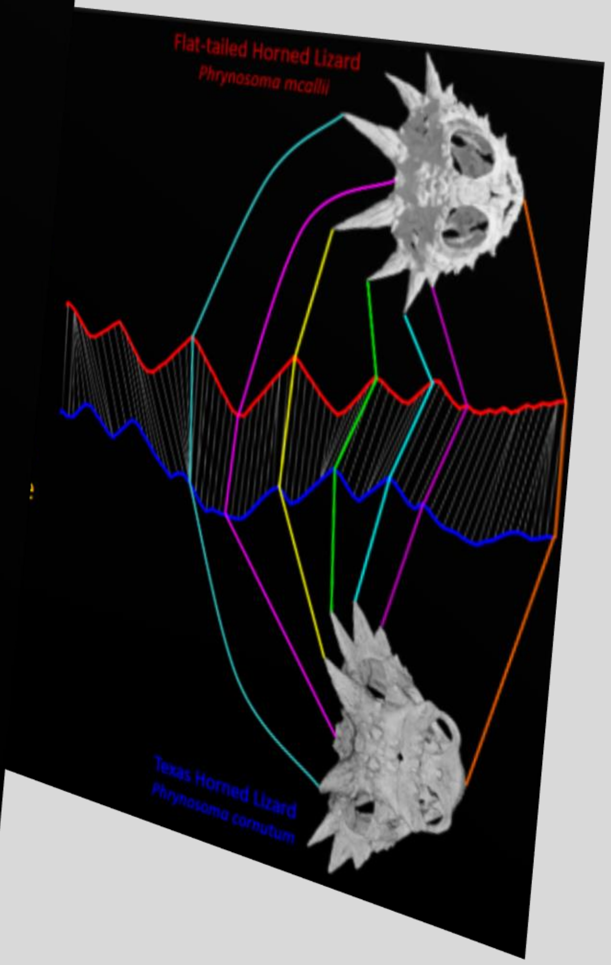


KDD2016
22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining
August 13 - 17, 2016 | San Francisco, California

Extracting Optimal Performance from Dynamic Time Warping

Abdullah Mueen
mueen@unm.edu

Eamonn Keogh
eamonn@cs.ucr.edu



www.cs.ucr.edu/~eamonn/public/SDM_How_to_do_Research_Keogh.pdf

www.cs.unm.edu/~mueen/DTW.pdf

Outline

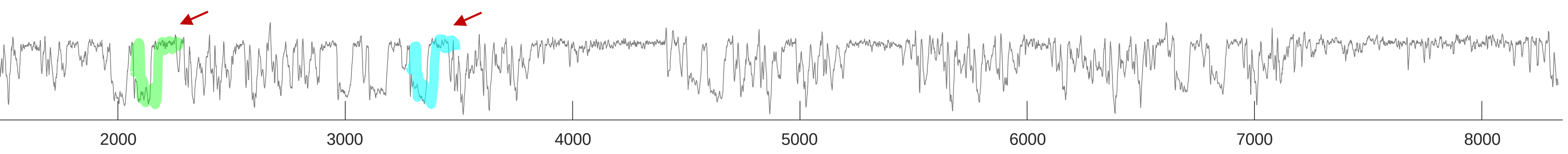
Act 1

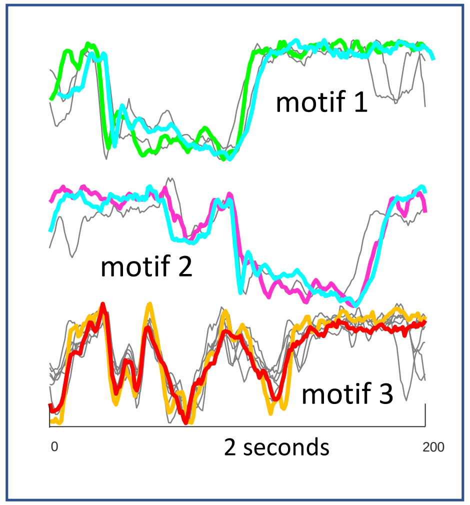
- Our Fundamental Assumption
- What is the (MP) Matrix Profile?
- Properties of the MP
- Developing a Visual Intuition for MP
- Basic Algorithms
  - MP Motif Discovery
  - MP Time Series Chains
  - MP Anomaly Discovery
  - MP Joins (self and AB)
  - MP Semantic Segmentation
- From Domain Agnostic to Domain Aware: The Annotation Vector (A simple way to use domain knowledge to adjust your results)
- The "*Matrix Profile and ten lines of code is all you need*" philosophy.
- Break

Act 2

- Background on time series mining
  - Similarity Measures
  - Normalization
- Distance Profile
  - Definition and Trivial Approach
  - Just-in-time Normalization
  - The MASS Algorithm
  - Weighted Distance Profile
  - Distance Profile with Gaps
- Matrix Profile
  - STAMP
  - STOMP
  - GPU-STOMP
- Open problems to solve

Fundamental Assumption: *Conservation is Key*





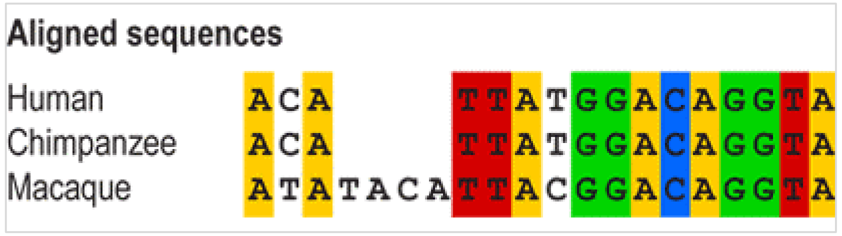
If a pattern is *conserved*, there must be some mechanism that conserves it. This is true in linguistics, music, genetics, literature, religions….

Much of our work asks *what* is conserved in time series, *when* is it conserved, and *why* was an expected conservation not observed…

For discrete strings, *conserved* is easy to define, for example $papa = {*}a{*}a$. For *time series* it requires a distance function, here we will use Euclidean Distance.

* Bengali: Bābā	* Norwegian : papa
* Mandarin : baba	* Spanish : papá
* Polish : tata	* Swahili : baba
* Swahili : baba	* English : papa
* Turkish : baba	* Hindi : papa
* Xhosa: -tata	* Indonesian : bapa

en.wikipedia.org/wiki/Mama_and_papa



Aligned sequences

```
Human       ACA    TTATGGACAGGTA
Chimpanzee  ACA    TTATGGACAGGTA
Macaque     ATATACATTACGGACAGGTA
```

What is the Matrix Profile?

- The Matrix Profile (MP) is a data structure that annotates a time series.
- **Key Claim:** Given the MP, most time series data mining problems are trivial or easy!
- We will show about ten problems that are trivial given the MP, including motif discovery, density estimation, anomaly detection, rule discovery, joins, segmentation, clustering etc. However, *you* can use the MP to solve *your* problems, or to solve a problem listed above, but in a different way, tailored to your interests/domain.

What is the Matrix Profile?

- The Matrix Profile (MP) is a data structure that annotates a time series.
- **Key Claim:** Given the MP, most time series data mining problems are trivial or easy!
- We will show about ten problems that are trivial given the MP, including motif discovery, density estimation, anomaly detection, rule discovery, joins, segmentation, clustering etc. However, *you* can use the MP to solve *your* problems, or to solve a problem listed above, but in a different way, tailored to your interests/domain.
- **Key Insight**: The MP profile has many highly desirable properties, and any algorithm you build on top of it, will inherit those properties.
  - Say you use the MP to create: *An Algorithm to Segment Sleep States*
  - Then, *for free*, you have also created: *An Anytime Algorithm to Segment Sleep States*
    An Online Algorithm to Segment Sleep States
    A Parallelizable Algorithm to Segment Sleep States
    A GPU Accelerated Algorithm to Segment Sleep States
    An Algorithm to Segment Sleep States with Missing Data
    etc.

The Highly Desirable Properties of the Matrix Profile I

- It is **exact**: For motif discovery, discord discovery, time series joins etc., the Matrix Profile based methods provide no false positives or false dismissals.
- It is **simple** and **parameter-free**: In contrast, the more general algorithms in this space that typically require building and tuning spatial access methods and/or hash functions.
- It is **space efficient**: Matrix Profile construction algorithms requires an inconsequential space overhead, just linear in the time series length with a small constant factor, allowing massive datasets to be processed in main memory (for most data mining, *disk is death*).
- It allows **anytime algorithms**: While exact MP algorithms are extremely scalable, for extremely large datasets we can compute the Matrix Profile in an anytime fashion, allowing ultra-fast approximate solutions and real-time data interaction.
- It is **incrementally maintainable**: Having computed the Matrix Profile for a dataset, we can incrementally update it very efficiently. In many domains this means we can effectively maintain exact joins/motifs/discords on streaming data forever.

The Highly Desirable Properties of the Matrix Profile II

- It can **leverage hardware**: Matrix Profile construction is embarrassingly parallelizable, both on multicore processors, GPUs, distributed systems etc.
- It is **free of the curse of dimensionality**: That is to say, It has *time complexity that is constant in subsequence length*: This is a very unusual and desirable property; virtually all existing algorithms in the time series scale poorly as the subsequence length grows.
- It can be constructed in **deterministic time**: Almost all algorithms for time series data mining can take radically different times to finish on two (even slightly) different datasets. In contrast, given *only* the length of the time series, we can precisely predict in advance how long it will take to compute the Matrix Profile. (this allows resource planning)
- It can handle **missing data**: Even in the presence of missing data, we can provide answers which are guaranteed to have no false negatives.
- Finally, and subjectively: **Simplicity and Intuitiveness:** Seeing the world through the MP lens often invites/suggests simple and elegant solutions.

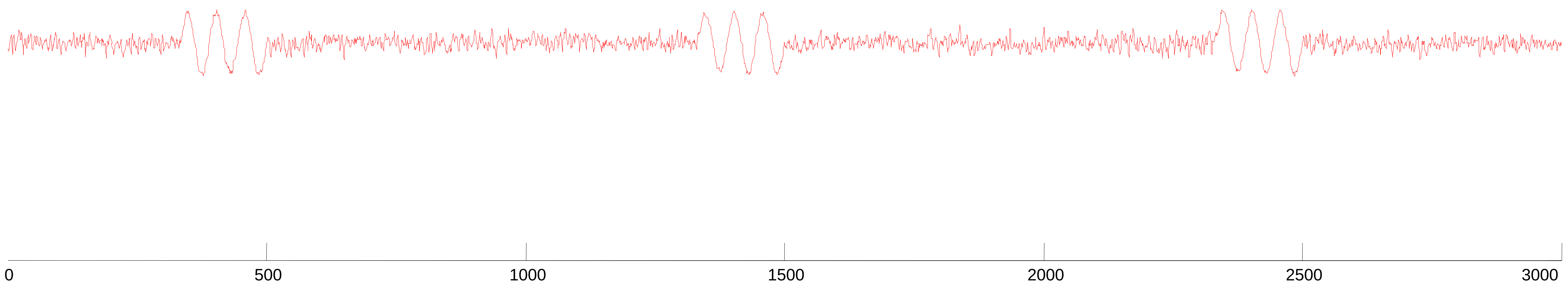
Developing a Visual Intuition for the Matrix Profile

In the following slides we are going to develop a visual intuition for the matrix profile, *without* regard to how we obtain it.

We are ignoring the elephant in the room; the MP *seems* to be much too slow to compute to be practical. We will address this in Part II of the tutorial.

Note that algorithms that use the MP do *not* require us to visualize the MP, but it happens that *just* visualizing the MP can be 99% of solving a problem.

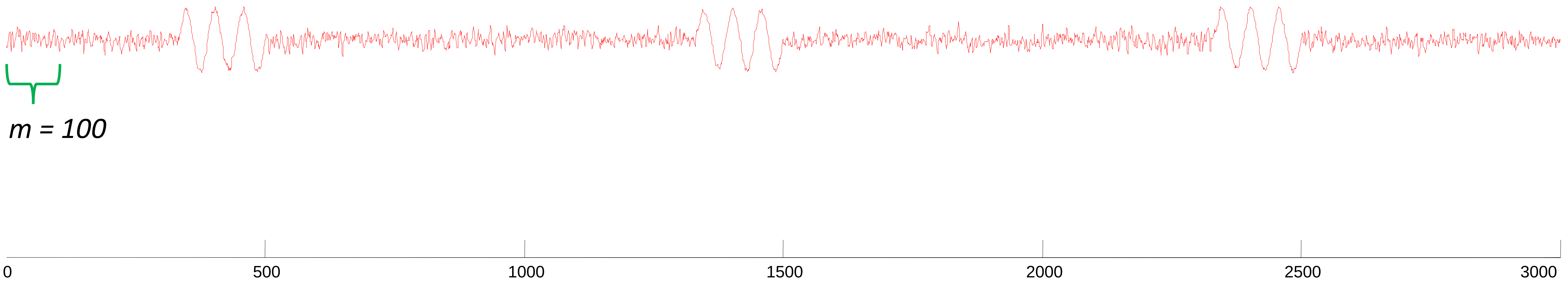
Intuition behind the Matrix Profile: Assume we have a time series *T*, lets start with a synthetic one...



$$|T| = n = 3{,}000$$

Note that for most time series data mining tasks, we are not interested in any *global* properties of the time series, we are only interested in small *local* subsequences, of this length, *m* $\underbrace{\qquad}$

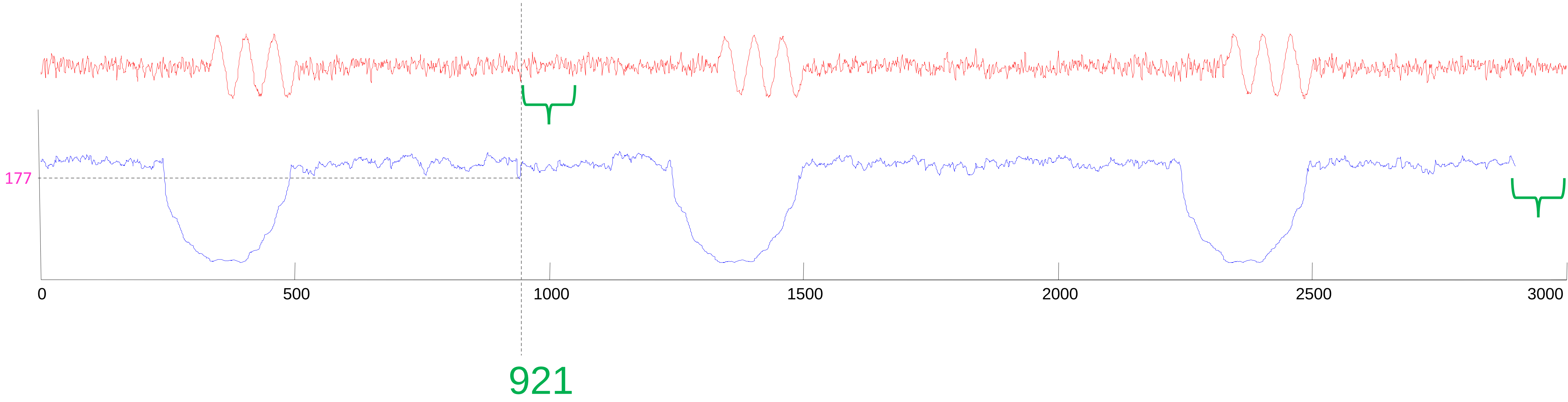
These subsequences might be about the length of individual heartbeats (for ECGs), individual days (for social media behavior), individual words (for speech analysis) etc



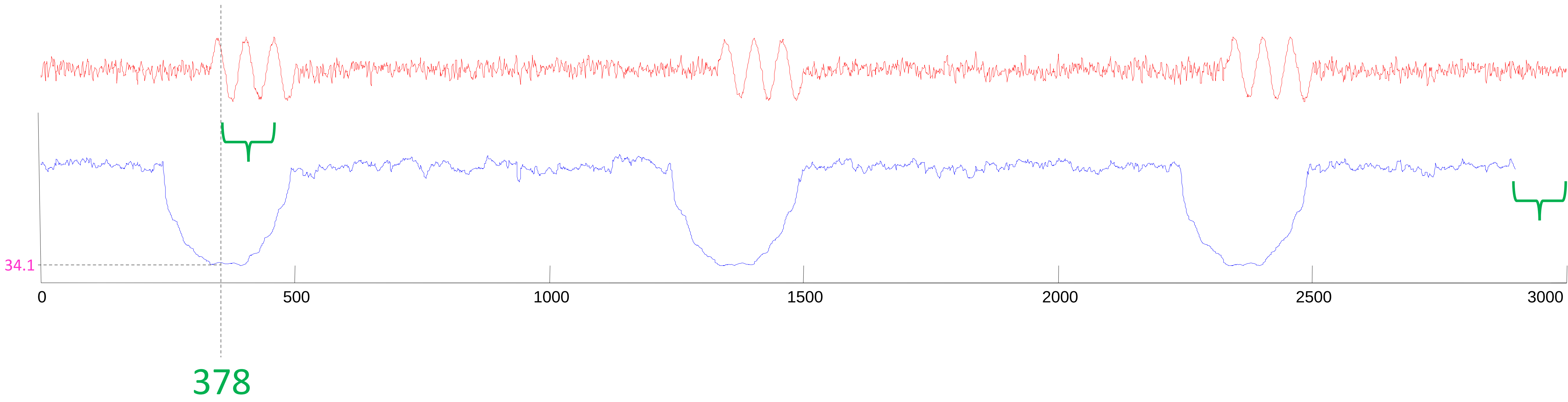
We can created a companion "time series", called a Matrix Profile or MP.

The matrix profile at the *i*th location records the distance of the subsequence in *T*, at the *i*th location, to its nearest neighbor under z-normalized Euclidean Distance.

For example, in the below, the subsequence starting at 921 happens to have a distance of 177.0 to its nearest neighbor (wherever it is).



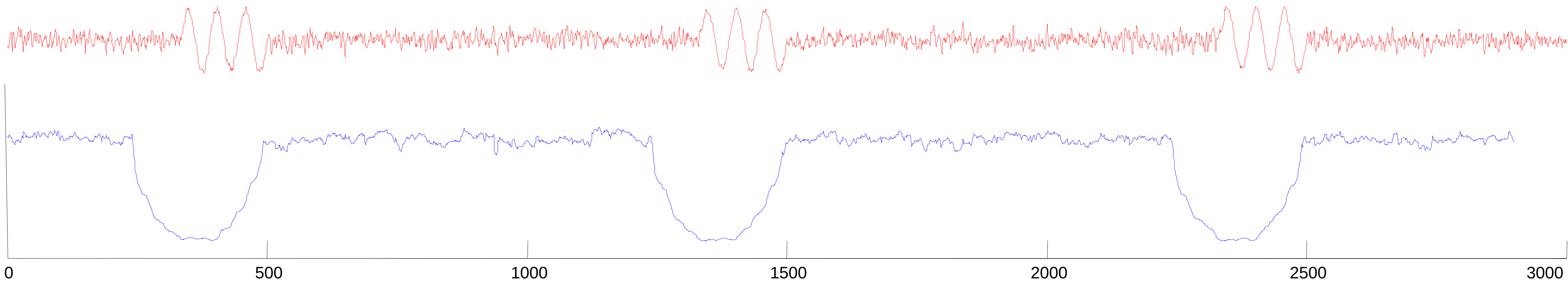
Another example. In the below, the subsequence starting at 378 happens to have a distance of 34.2 to its nearest neighbor (wherever it is).



For the rest of this tutorial….

The Matrix Profile is always shown in blue.

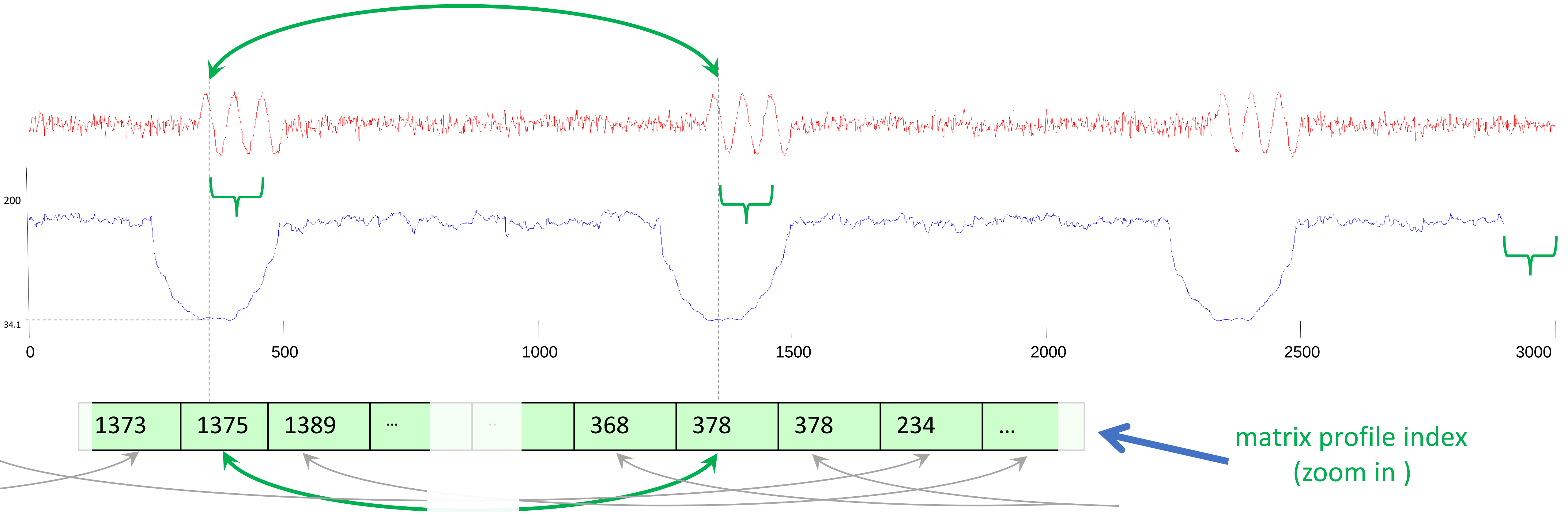
The real time series data, is generally shown in red.



We can create another companion sequence, called a matrix profile index.

The MPI contains integers that are used as pointers. As a practical matter, even 32-bits will let us have a MP of length 2,147,483,647, over two years of data at 60Hz. A 64-bit integer gives us ten billion years at 60Hz)

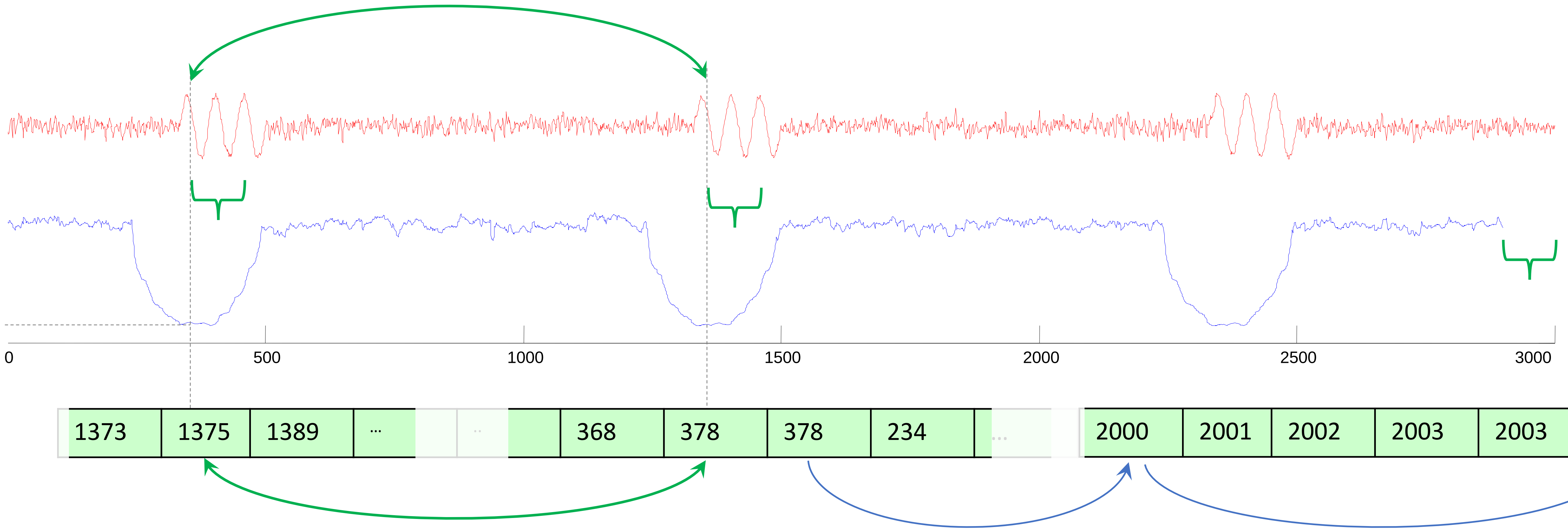
In the following slides we won't bother to show the matrix profile index, but be aware it exists, and it allows us to find the nearest neighbor to any subsequence in constant time.



Note that the pointers in the matrix profile index are not necessarily symmetric.

If **A** points to **B**, then **B** may or may not point to **A**

An interesting exception, the two smallest values in the MP must have the same value, and their pointers must be mutual. This is the classic *time series motif*.

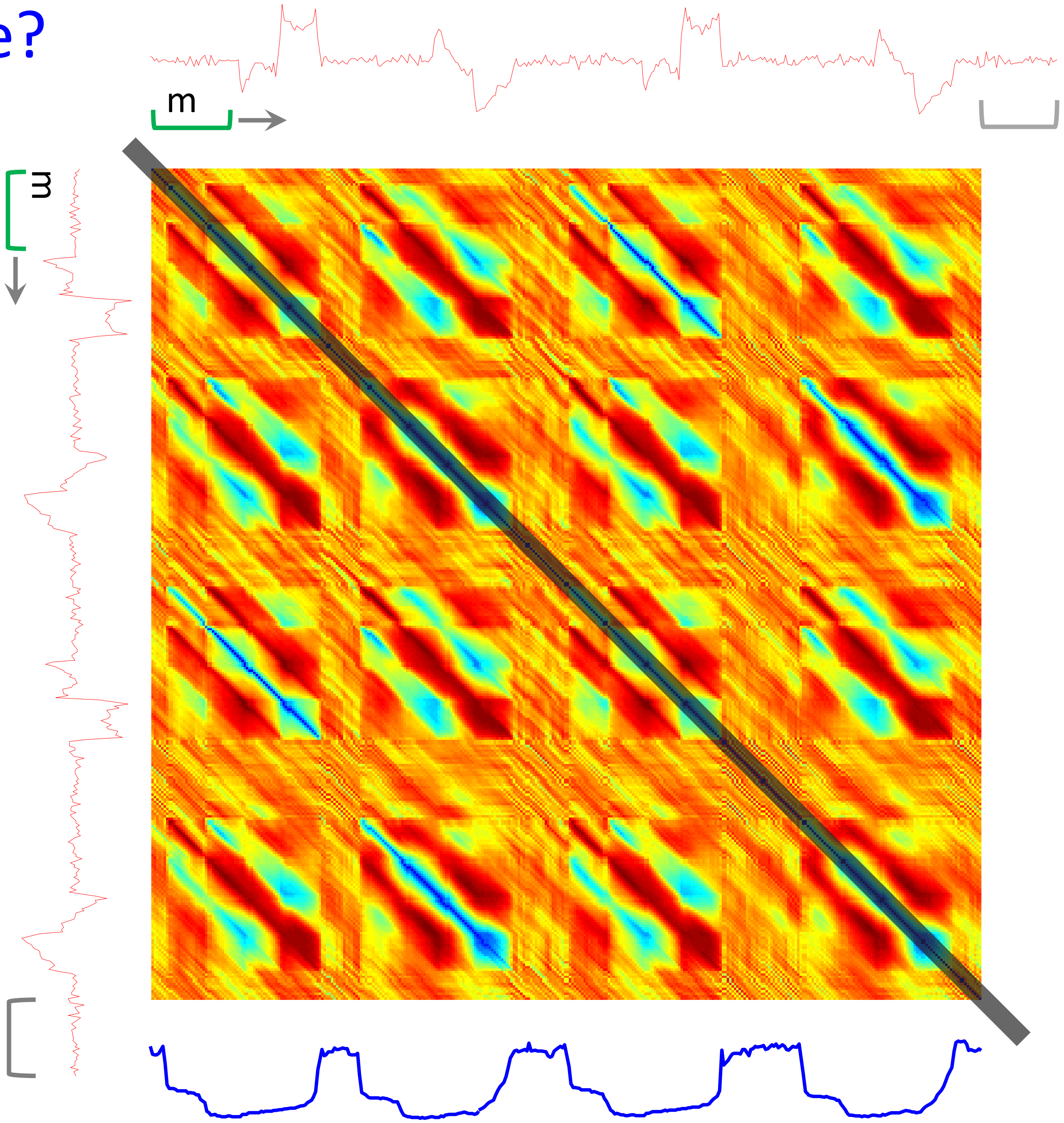


Why is it called the Matrix Profile?

One naïve way to compute it would be to construct a distance matrix of all pairs of subsequences of length *m*.

For each column, we could then “project” down the smallest (*non diagonal*) value to a vector, and that vector would be the Matrix Profile.

While in general we could never afford the memory to do this (4TB for just |T|= one million), for most applications the Matrix Profile is the *only* thing we need from the full matrix, and we can compute and store it very efficiently. (as we will see later)

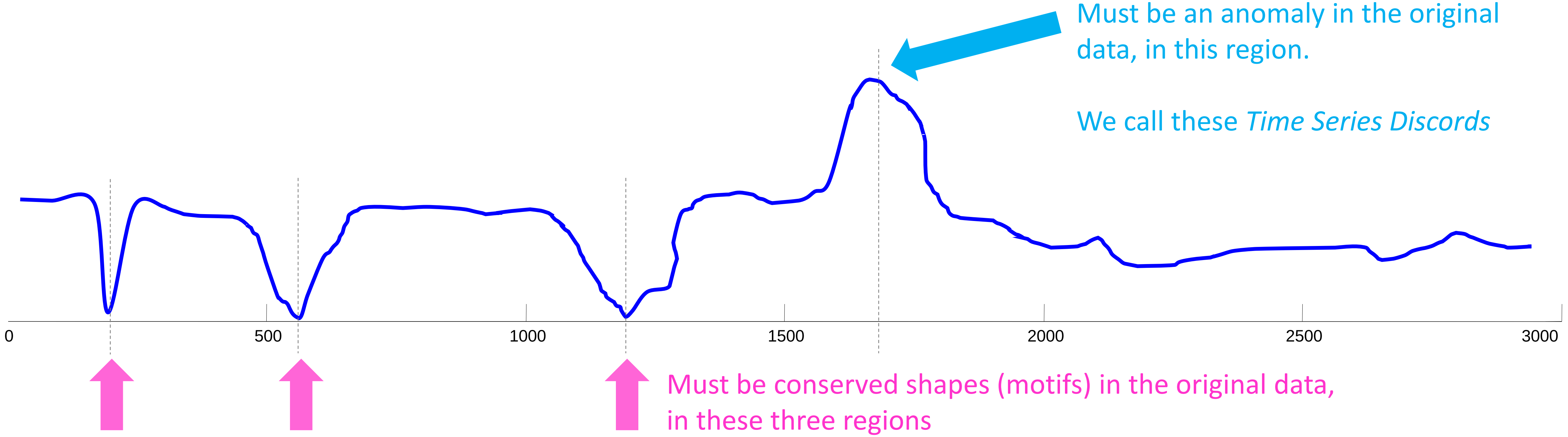


Key:
Small distances are blue
Large distances are red
Dark stripe is excluded

How to "read" a Matrix Profile

Where you see relatively low values, you know that the subsequence in the original time series must have (at least one) relatively similar subsequence elsewhere in the data (such regions are "motifs" or reoccurring patterns)

Where you see relatively high values, you know that the subsequence in the original time series must be unique in its shape (such areas are "discords" or anomalies).



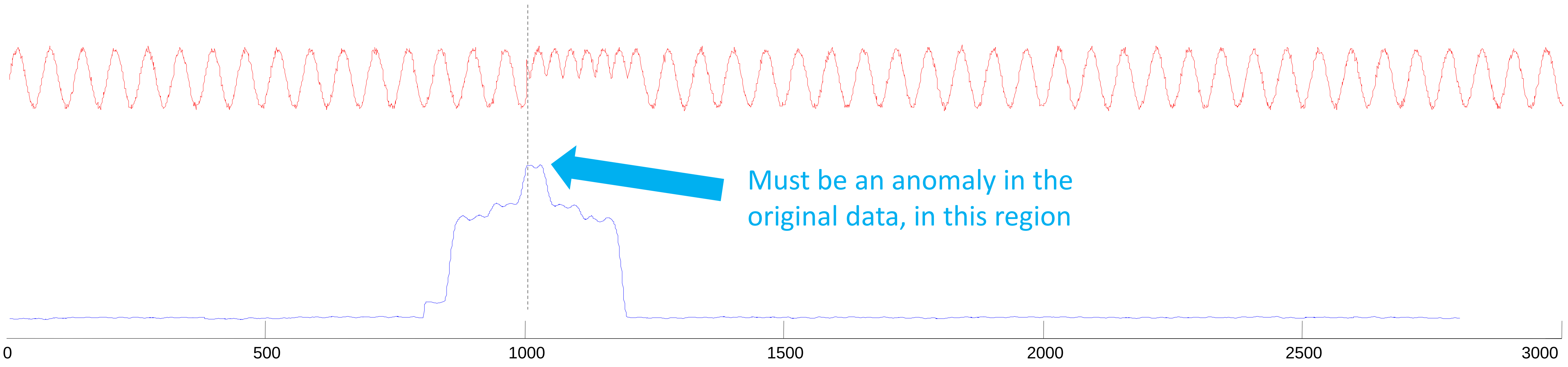
Must be an anomaly in the original data, in this region.

We call these *Time Series Discords*

Must be conserved shapes (motifs) in the original data, in these three regions

How to “read” a Matrix Profile: Synthetic Anomaly Example

Where you see relatively high values, you know that the subsequence in the original time series must be unique in its shape. In fact, the highest point is *exactly* the definition of Time Series Discord, perhaps the best anomaly detector for time series*

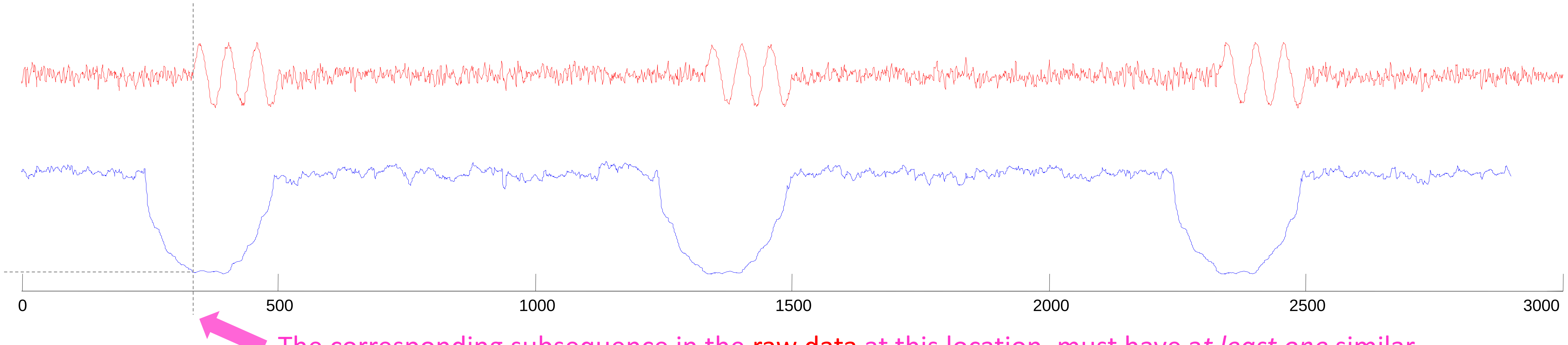


* Vipin Kumar performed an extensive empirical evaluation and noted that “..on 19 different publicly available data sets, comparing 9 different techniques (time series discords) is the best overall technique.”. V. Chandola, D. Cheboli, V. Kumar. Detecting Anomalies in a Time Series Database. UMN TR09-004

How to "read" a Matrix Profile: Synthetic Motif Example

Where you see relatively low values, you know that the subsequence in the original time series must have (at least one) relatively similar subsequence elsewhere in the data.

In fact, the lowest points must be a tieing pair, and correspond exactly to the classic definition of *time series motifs*.



The corresponding subsequence in the raw data at this location, must have at *least one* similar subsequence somewhere

How to “read” a Matrix Profile:

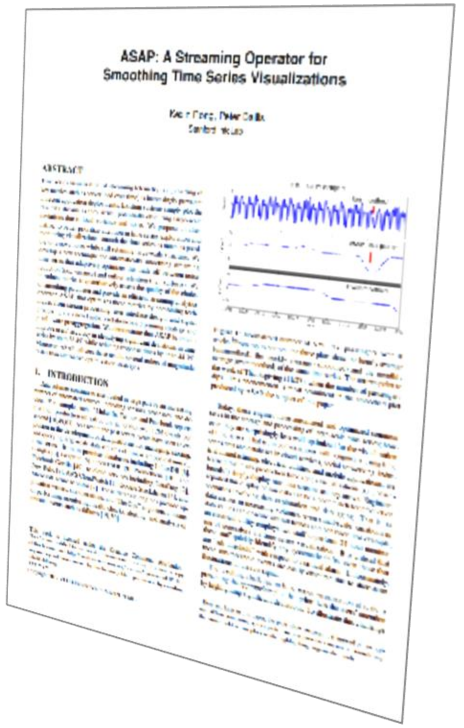
Now that we understand what a Matrix Profile is, and we have some practice interpreting them on *synthetic* data, let us spend the next five minutes to see some examples on *real* data.

Note that we will typically create algorithms that use the Matrix Profile, without actually having humans *look* at it.

Nevertheless, in many exploratory time series data mining tasks, just looking at the Matrix Profile can give us unexpected and actionable insights.

Ready to begin?

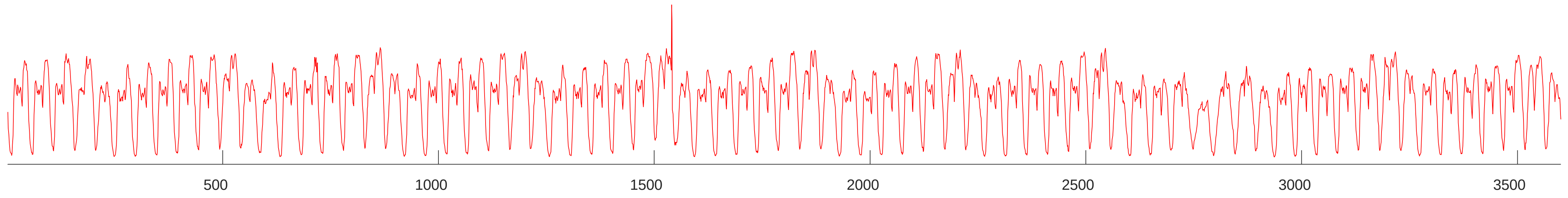
Taxi Example: Part I


ASAP: A Streaming Operator for Smoothing Time Series Visualizations

Given a long time series, where should you examine carefully?
The problem is called “Attention Prioritization”, a group at Stanford is working on this [a].
However we think that the Matrix Profile can be used for this, “for free”.

Below is the data, the hourly average of the number of NYC taxi passengers over 75 days in Fall of 2014.

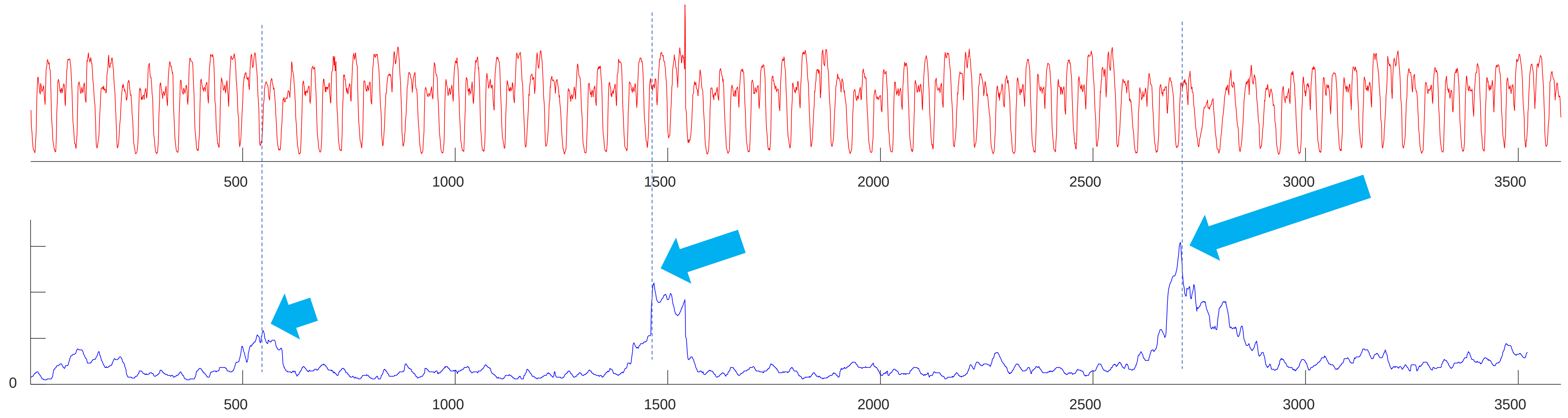
Lets compute the Matrix Profile for it, we choose a subsequence length corresponding to two days.... (next slide)



[a] http://futuredata.stanford.edu/ASAP/extended.pdf

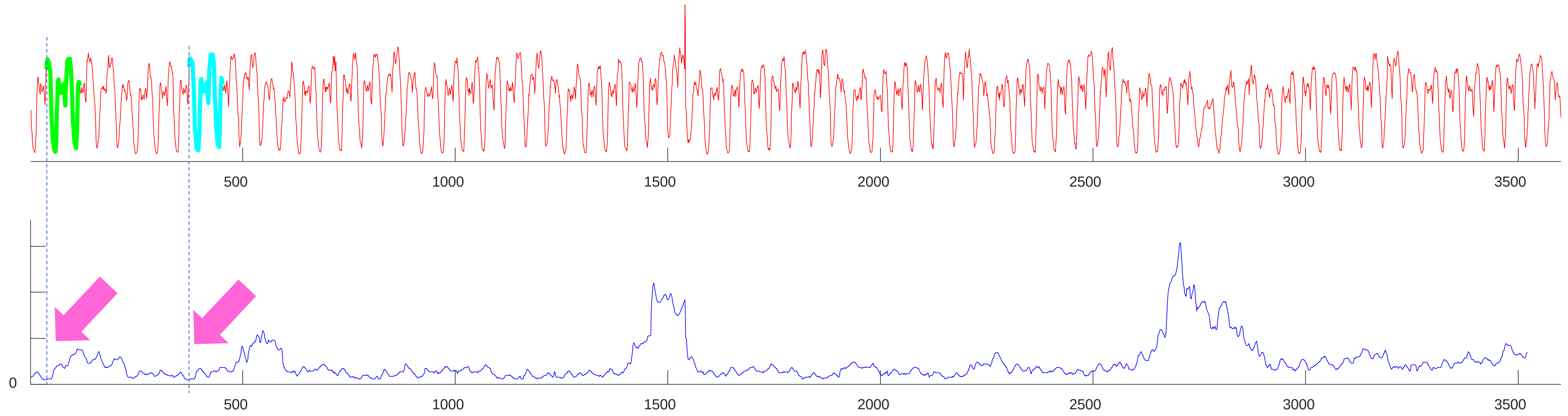
Taxi Example: Part II

- The highest value corresponds to Thanksgiving (the uniqueness of Thanksgiving was the only thing the Stanford Team noted)
- We find a secondary peak around Nov 6th, what could it be? Daylight Saving Time! The clock going backwards one hour, gives an *apparent* doubling of taxi load.
- We find a tertiary peak around Oct 13th, what could it be? Columbus Day! Columbus Day is largely ignored in much of America, but still a big deal in NY, with its large Italian American community.



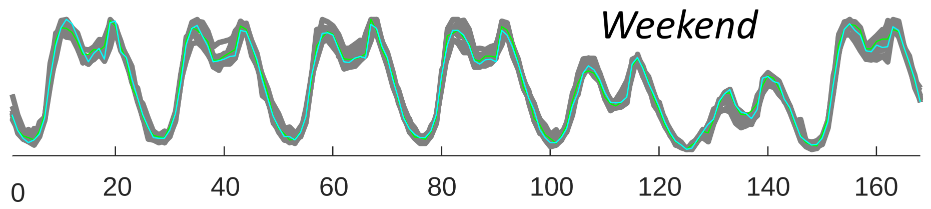
Taxi Example: Part III

- What about the lowest values? (the best motifs)
- They are *exactly* seven days apart, suggesting that in this dataset, there might be a periodicity of *seven* days, in addition to the more obvious periodicity of *one* day.



Italy Power Demand (1995 to 1998)

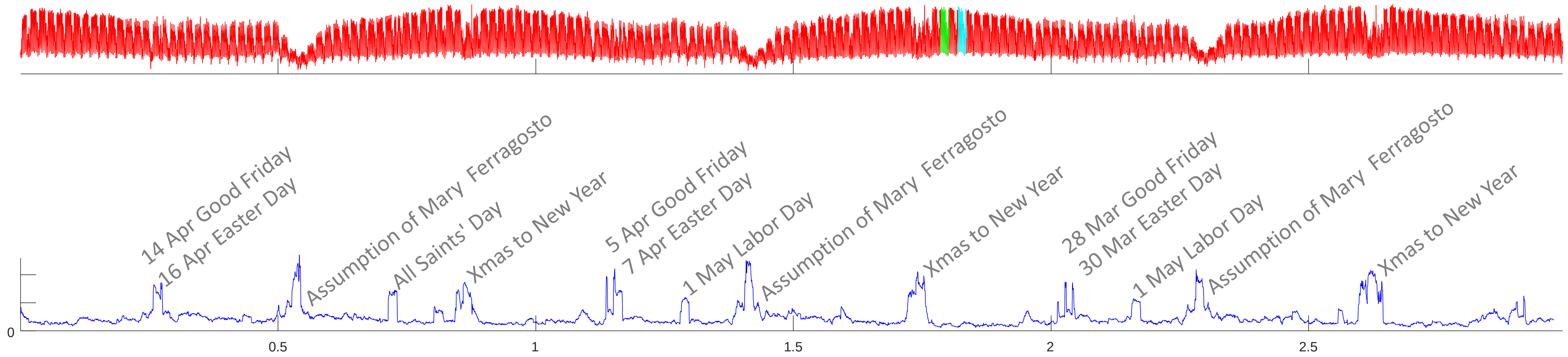
The top motif is a typical work week, starting from Tuesday

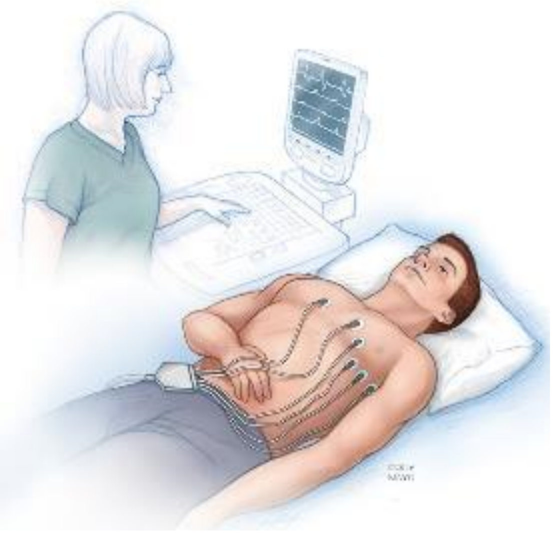


The Taxi example was easy to solve by manual inspection of the *raw* data, but with just an order of magnitude more data, the problem becomes much harder. Lets try a similar, but larger example, Italian Power Demand 1995 to 1998.

Note that the matrix profile is very low *on average*, most weeks are similar to the previous week (*persistence*) or the same week in a different year (*history*).

All the high values can be explained by Italian holidays, most of which fall on different days in consecutive years.



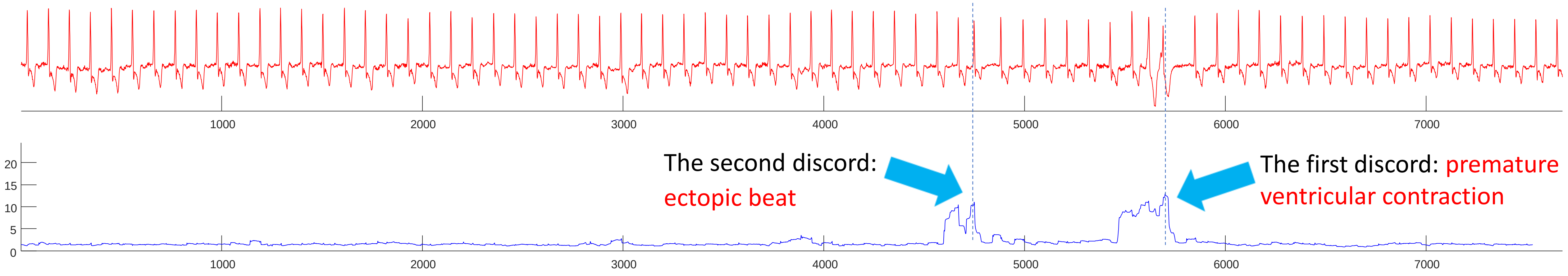


Electrocardiogram

(MIT-BIH Long-Term ECG Database)

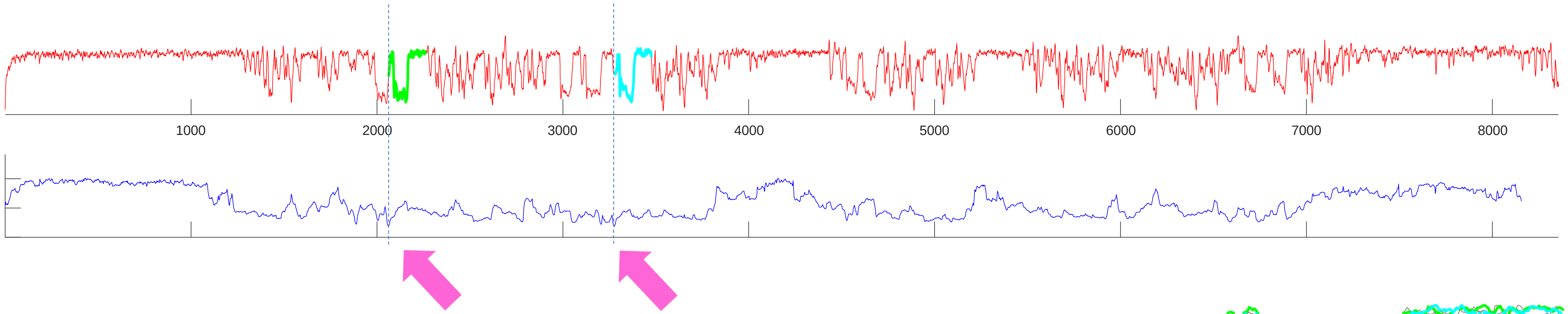
In this case there are two anomalies annotated by MIT cardiologists. The Matrix Profile clearly indicates them.

Here the subsequence length was set to 150, but we still find these anomalies if we *half* or *triple* that length.



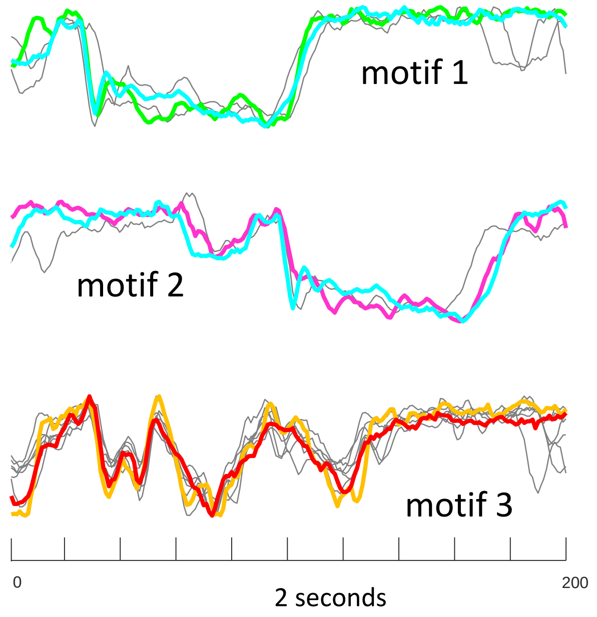
Zebra Finch

(Zebra Finch Vocalizations in MFCC, 100 day old male)

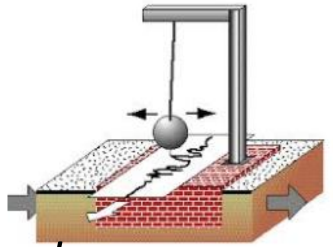


Motif discovery can often surprise you.

While it is clear that this time series is not random, we did not expect the motifs to be so well conserved or repeated so many times. There is evidence of a *vocabulary*, and maybe even a *grammar*…

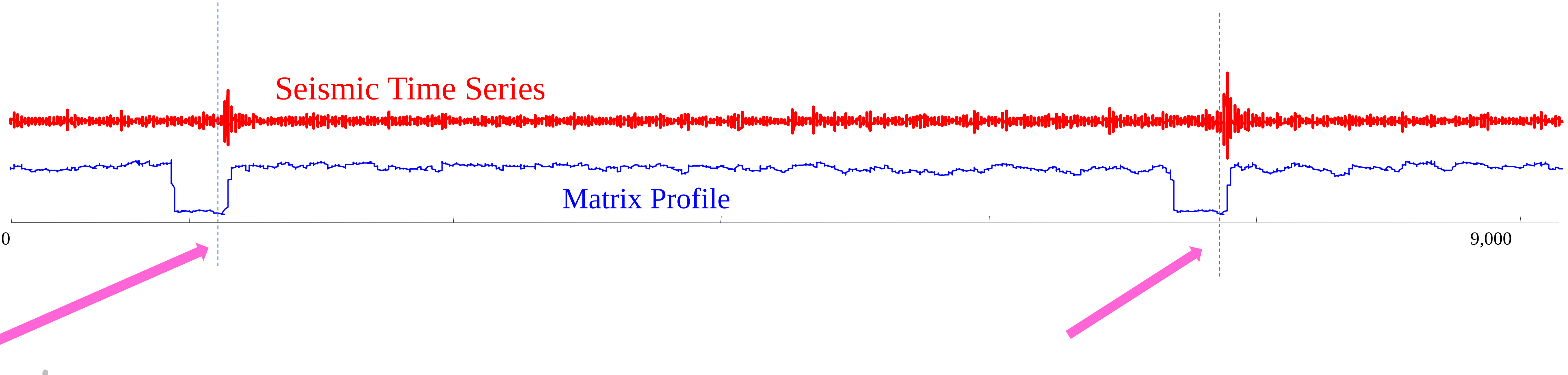


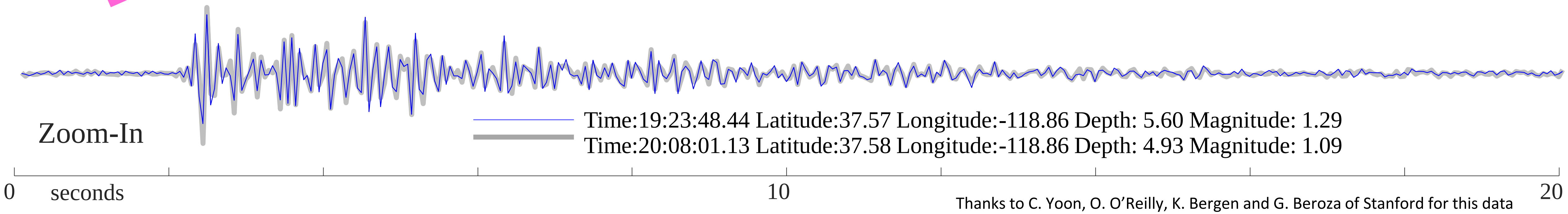
Seismology



If we see low values in the MP of a seismograph, it means there must have been a *repeated earthquake*.
Repeated earthquakes can happen *decades* apart.
Many fundamental problems seismology, including the discovery of foreshocks, aftershocks, triggered earthquakes, swarms, volcanic activity and induced seismicity, can be reduced to the discovery of these repeated patterns.

The corresponding subsequence in the raw data at this location, must have *at least one* similar earthquake somewhere

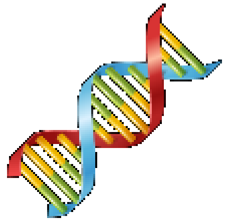




Thanks to C. Yoon, O. O'Reilly, K. Bergen and G. Beroza of Stanford for this data

Chimp DNA

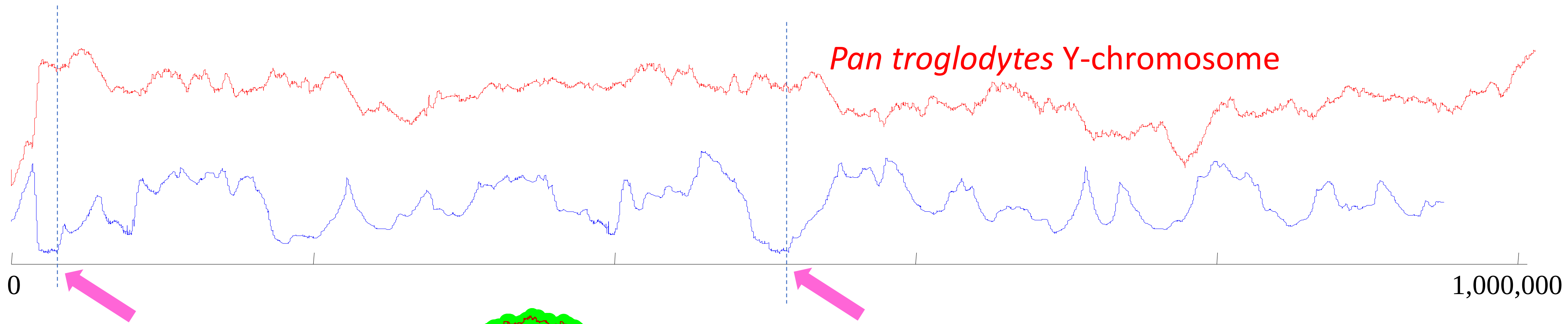
Y-chromosome converted to time series

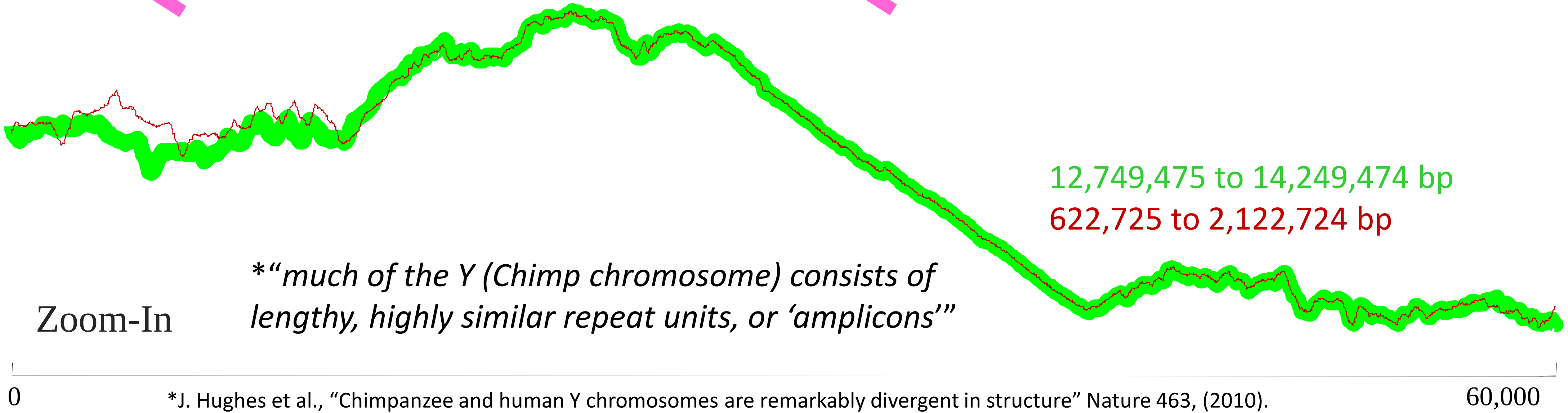


It is possible to convert DNA strings to real-valued time series, in a lossless fashion

```
$T_1 = 0$;
for i = 1 to length(chromosome)
    if $chromosome_i$ = A, then $T_{i+1} = T_i + 2$
    if $chromosome_i$ = G, then $T_{i+1} = T_i + 1$
    if $chromosome_i$ = C, then $T_{i+1} = T_i - 1$
    if $chromosome_i$ = T, then $T_{i+1} = T_i - 2$
end
```

Let us search the Chimp's DNA for repeated structure, of length 60,000…





*"*much of the Y (Chimp chromosome) consists of lengthy, highly similar repeat units, or 'amplicons'*"

*J. Hughes et al., "Chimpanzee and human Y chromosomes are remarkably divergent in structure" Nature 463, (2010).

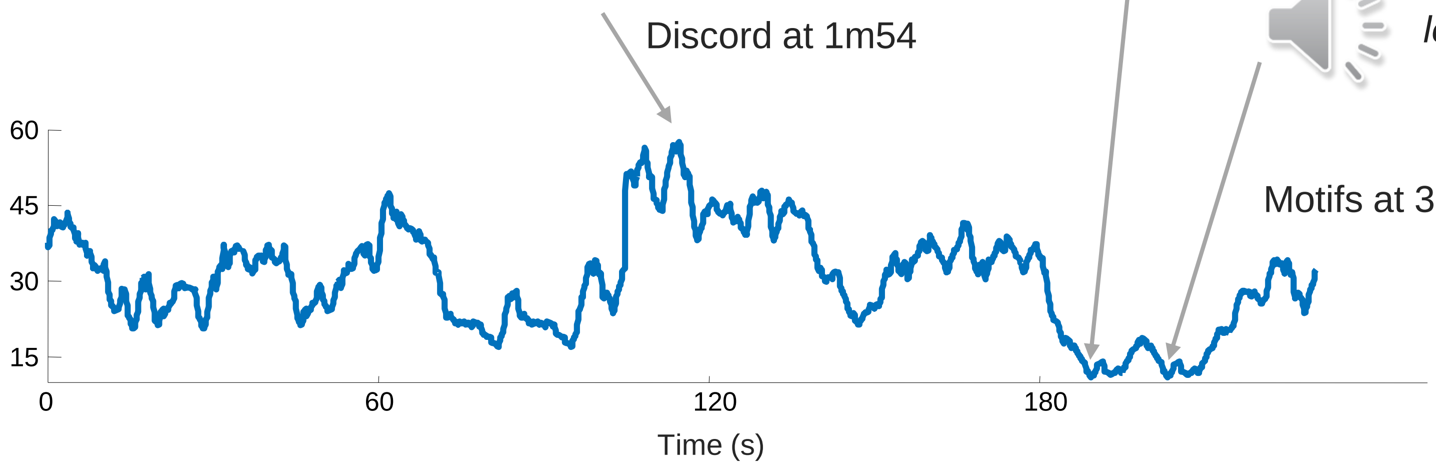
Music

While *motifs* usually point to chorus, *discords* point to bridges or solos

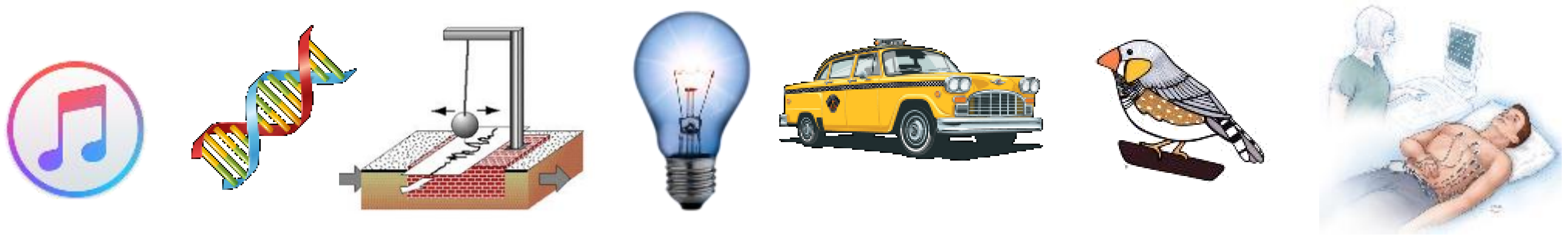
{instrumental bridge}

let it be, let it be, yeah let it be And there will be an answer, let it

let it be, let it be, yeah let it be And there will be an answer, let it



Summary



We could do this all day! We have applied the MP to *hundreds* of datasets.

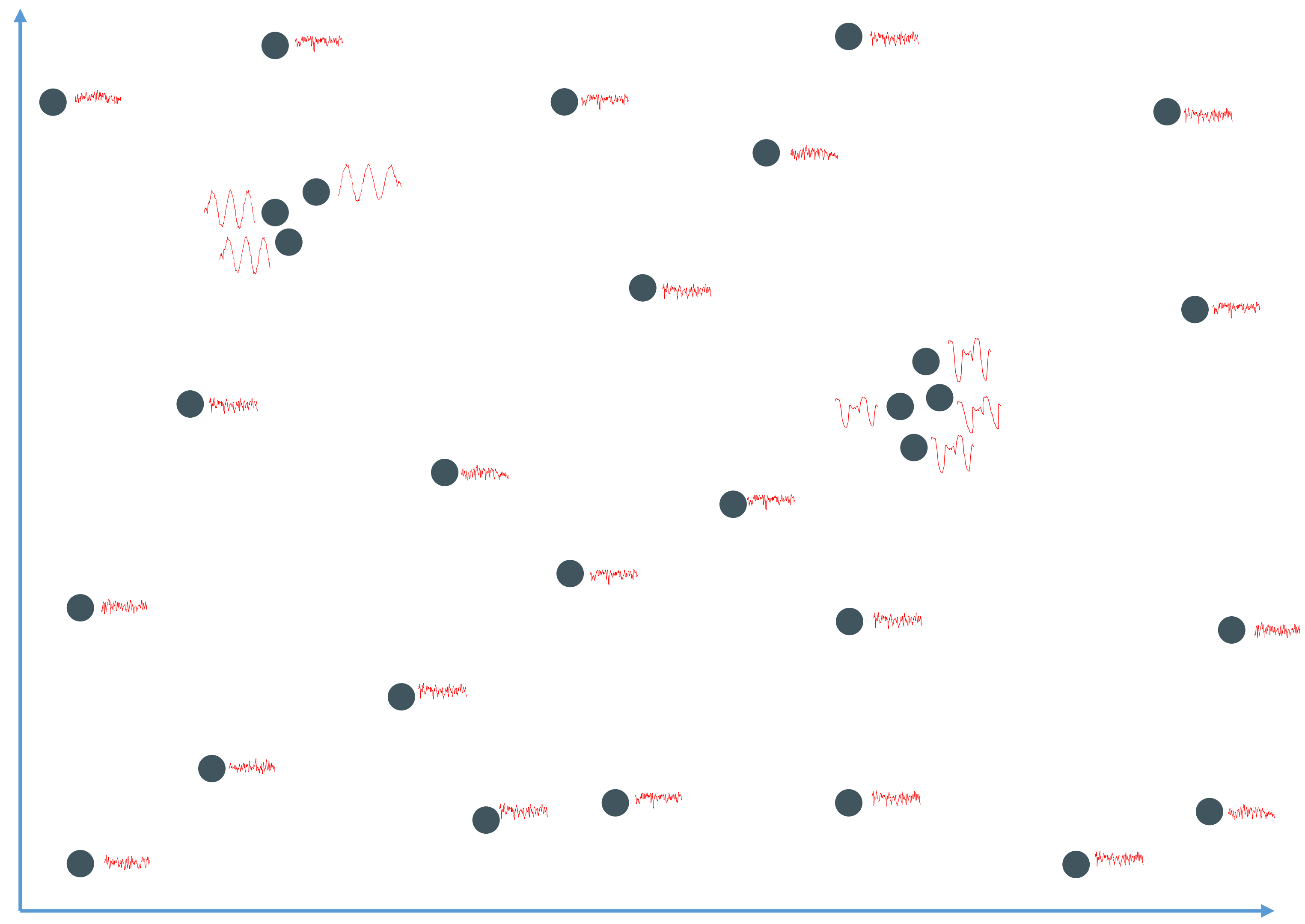
It is worth reminding ourselves of the following:

- The MP can find structure in taxi demand, seismology, DNA, power demand, heartbeats, music and bird vocalization data. However the MP does not "*know*" anything about this domains, it was not tweaked, tuned or adjusted in anyway.
- This domain-agnostic nature of MP is a *great* strength, you typically get very good results "out-of-the-box" no matter what your domain.

The following is also worth stating again:

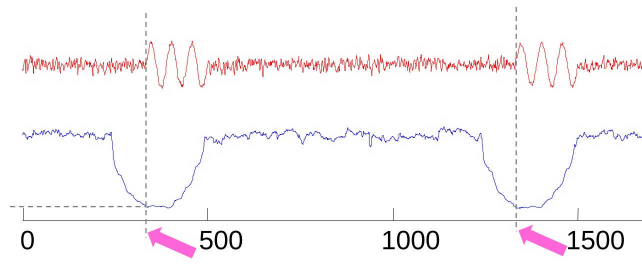
- We spent time *looking* at the MP to gain confidence that it is doing something useful. However, most of the time, only a higher-level algorithm will look at the MP, with no human intervention or inspection (we will make that clearer in the following slides).

A Minor Visual Mapping Trick



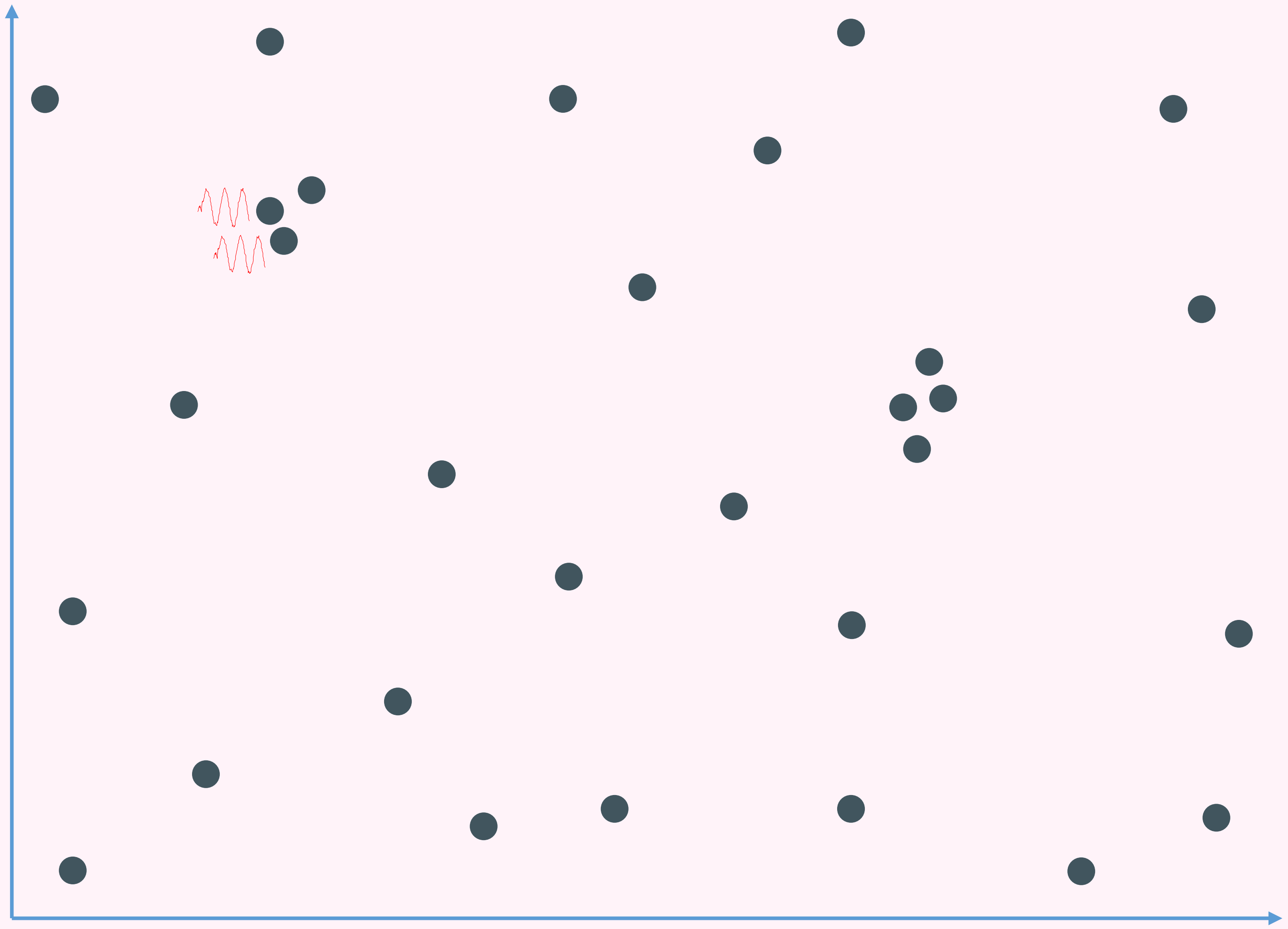
It is sometime useful to think of time series subsequences as points in m-dimensional space.

In this view, dense regions in the *m*-dimensional space correspond to regions of the time series that have a low corresponding MP



The Top-K Motifs I

Here we show a *sensible* way to extract the top-K motifs. However, there is nothing stopping you from inventing a different way. If you do, the MP will let you compute it in milliseconds.



We need a parameter R.

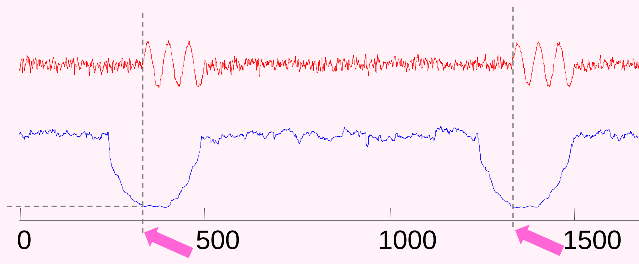
1 < R < (small number, say 3)

Lets make R = 2 for now.

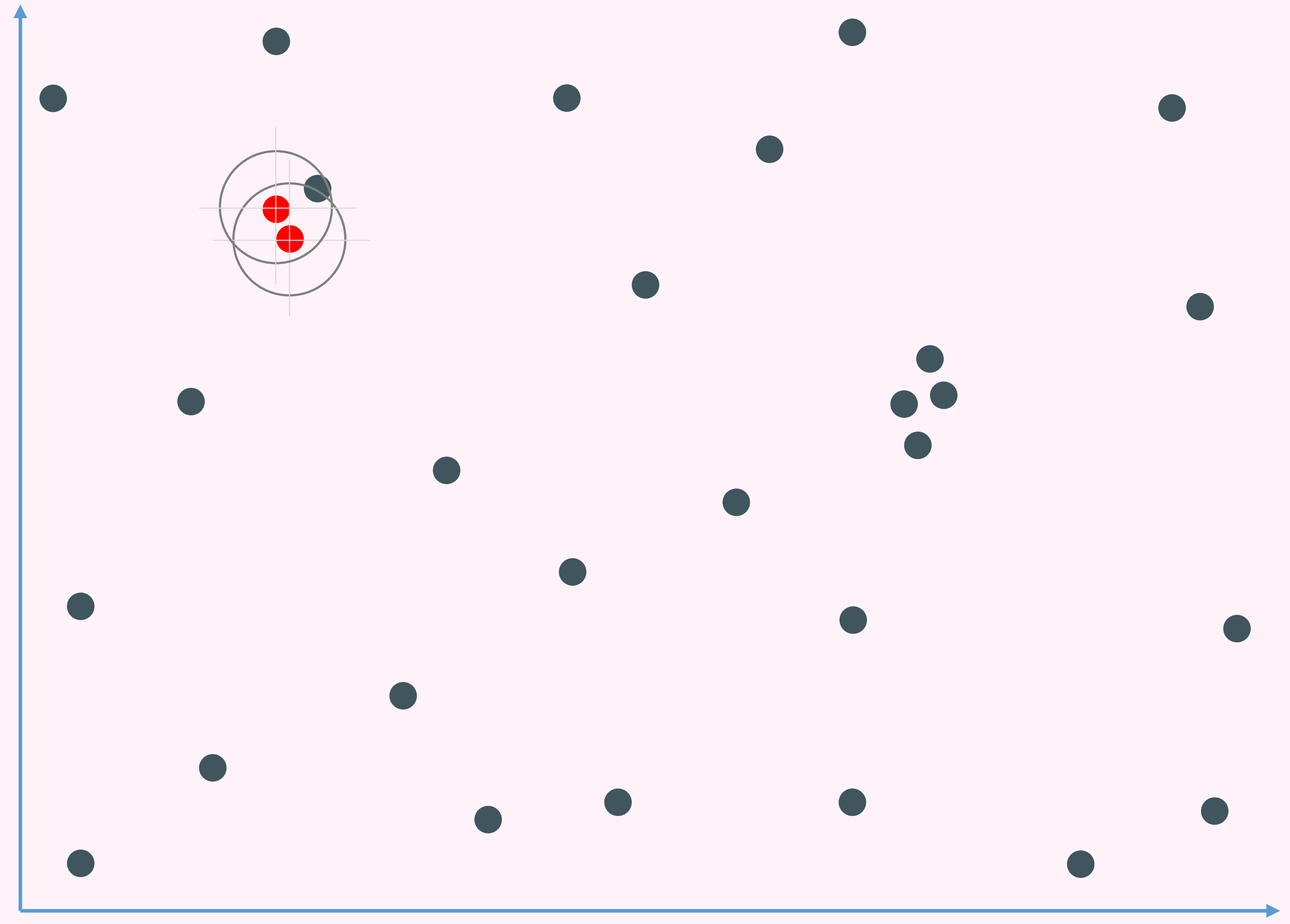
We begin by finding the nearest pair of points, the *motif pair*….

(This the pair of subsequences corresponding to lowest pair of values in the MP)

Next slide…



The Top-K Motifs II

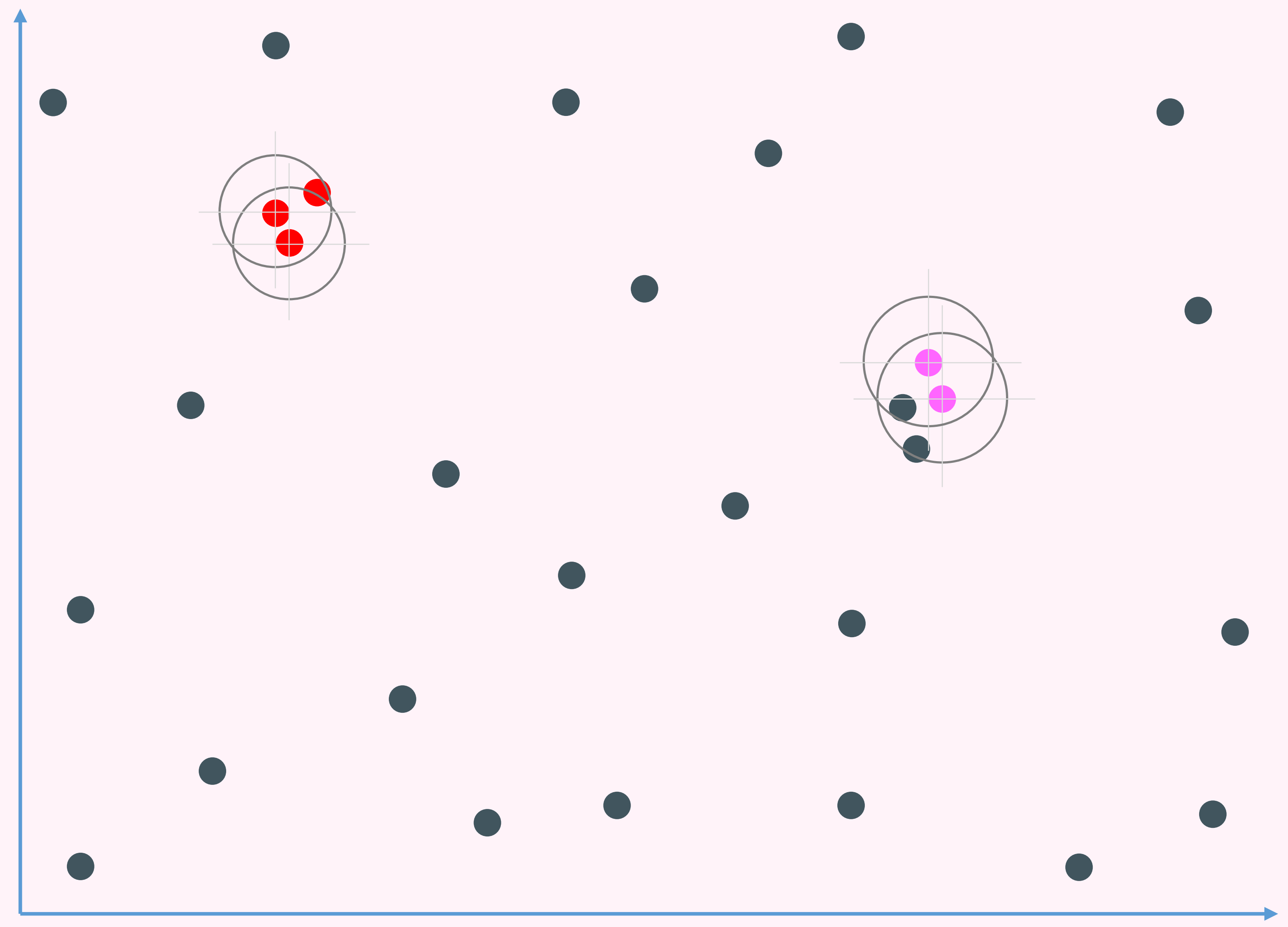


We find the nearest pair of points are D_1 apart.

Lets draw a circle, D_1 *times* R, around both points.

Any points that are within either of these circles, are added to this motif, in this case there is just one…
See next slide…

The Top-K Motifs III



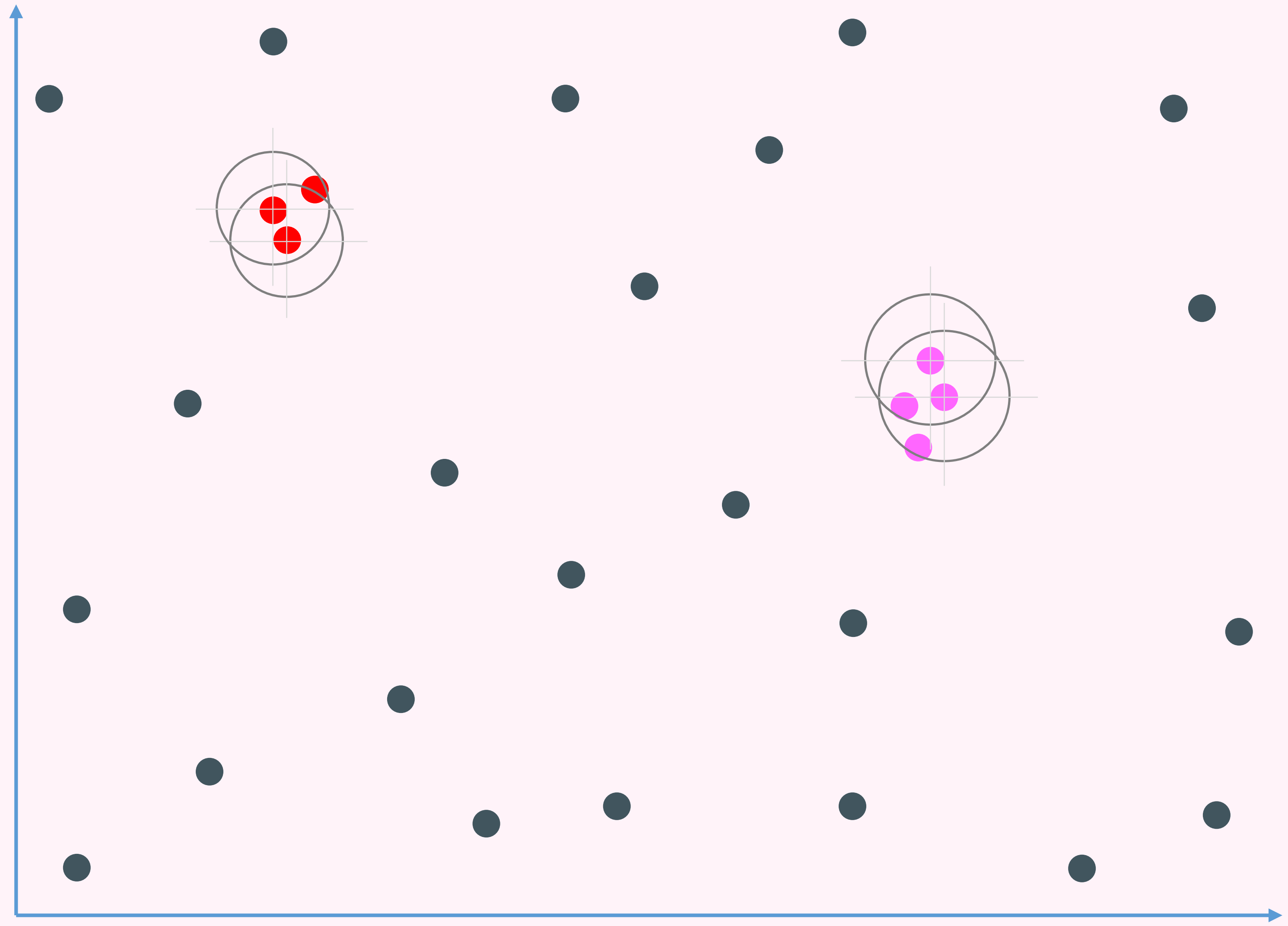
The Top-1 motif has three members, it is done.

Now lets find the Top-2 motif. We begin by finding the nearest pair of points, excluding anything from the top motif.

The nearest pair of points are D_2 apart.
Lets draw a circle D_1 *times* R, around both points.

Any points that are within either of these circles, is added to this motif, in this case there are two… See next slide…

The Top-K Motifs IIII



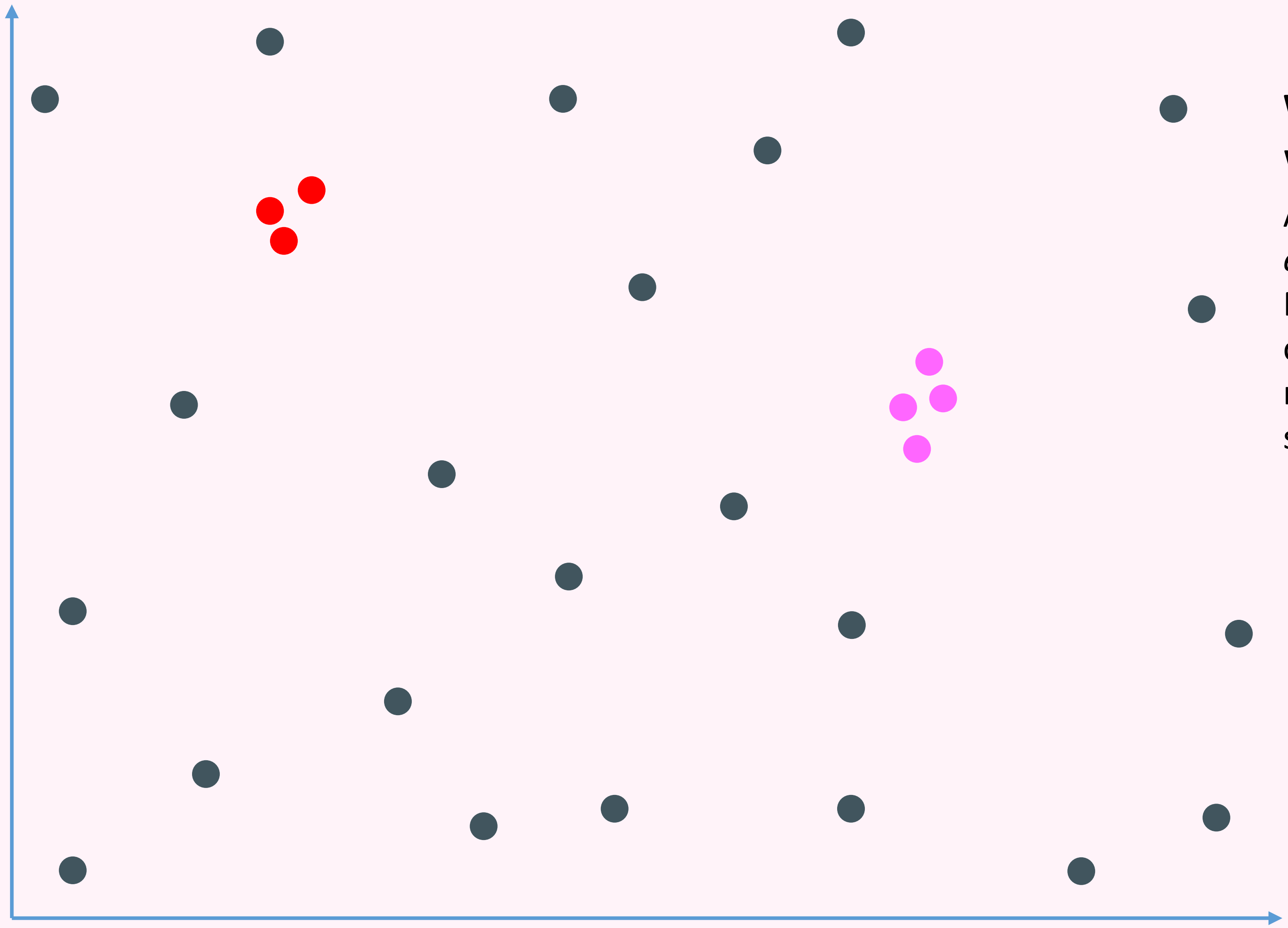
The Top-1 motif has three members, it is done.

Now lets find the Top-2 motif. We begin by finding the nearest pair of points, excluding anything from the top motif.

The nearest pair of points are D_2 apart.
Lets draw a circle D_1 *times* R, around both points.

Any points that are within either of these circles, are added to this motif, in this case there are two, for a total of four items in the Top-2 Motif

The Top-K Motifs V



We are done with the Top-2 Motif

Note that we will always have:

$D_1 < D_2 < D_3 \ldots$

When to stop? (what is K?)

We could use MDL etc.

As a practical matter, we can pull out all K, and use *eyeballing* to judge the quality of motifs.

For example, in the below, Motif 1 is stunningly well conserved, Motif 2 is somewhat conserved, Motif 3 may be getting close to random…. So here we would say we have a strong **Top-2** Motifs.

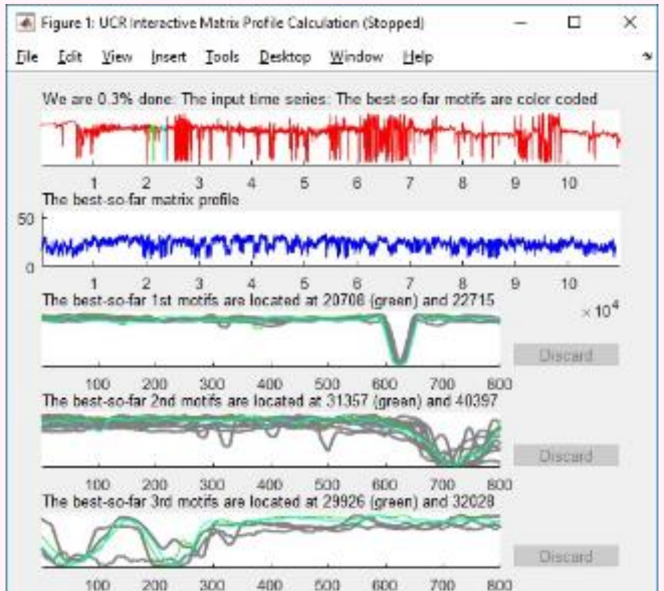


Figure 1: UCR Interactive Matrix Profile Calculation (Stopped)
File Edit View Insert Tools Desktop Window Help
Discard
Discard
Discard

We are 0.3% done: The input time series: The best so far motifs are color coded

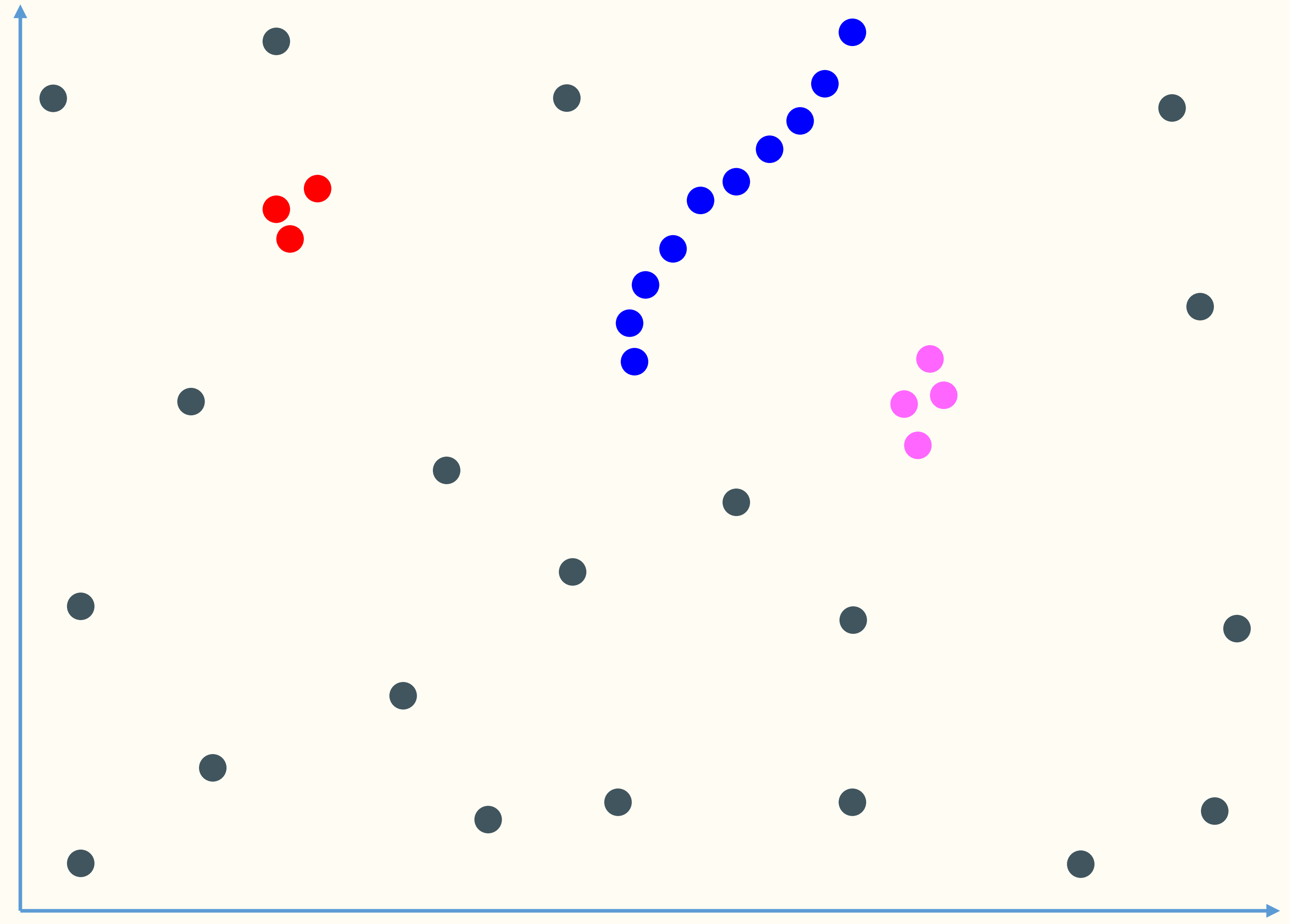
The best-so-far matrix profile

The best-so-far 1st motifs are located at 20708 (green) and 22715

The best-so-far 2nd motifs are located at 31357 (green) and 40397

The best-so-far 3rd motifs are located at 29926 (green) and 32028

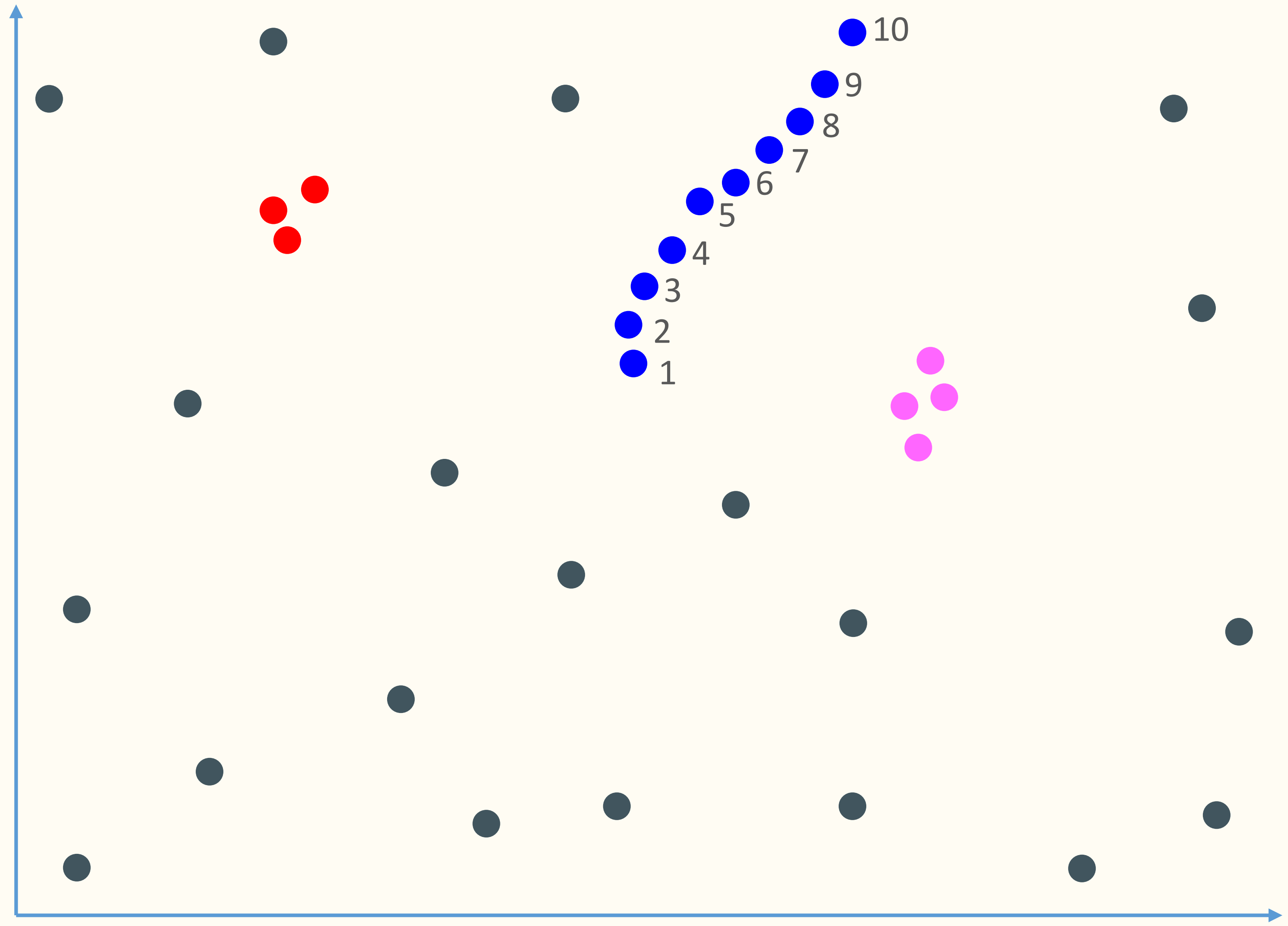
From Motifs to Time Series Chains



Take a look at the blue ‘subsequences”

They would not from a single motif (but perhaps they could form a *set* of motifs).

From Motifs to Time Series Chains



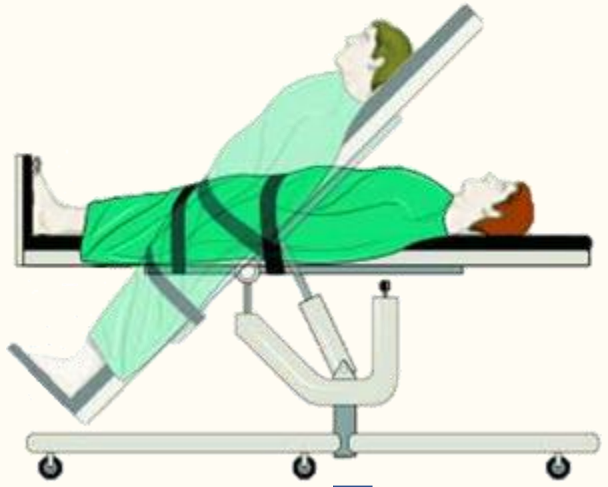
However, if we label them by *arrival* time, you can see that they are *drifting*, or *evolving* in time.

This is *actionable*, for example, where will the 11th item land? Surely just Northeast of the 10th item

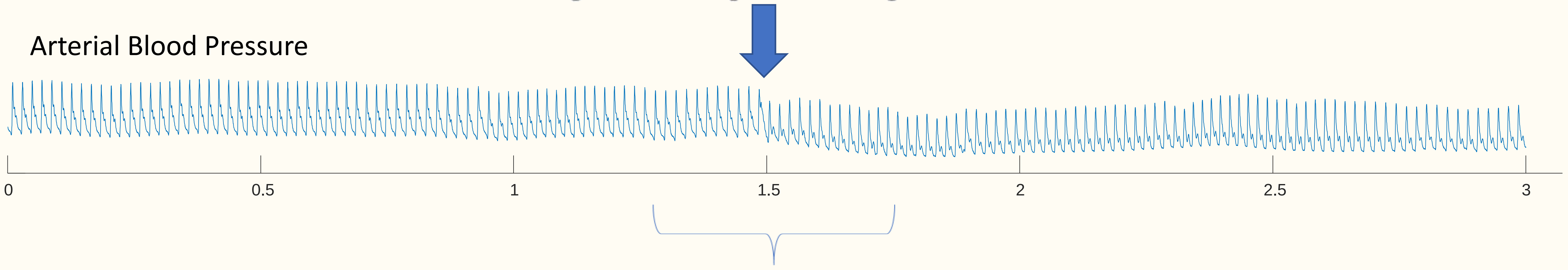
We call such pattern chains, with the first item as the *anchor*.

Do such patterns exist in the real world?

Can we find them?



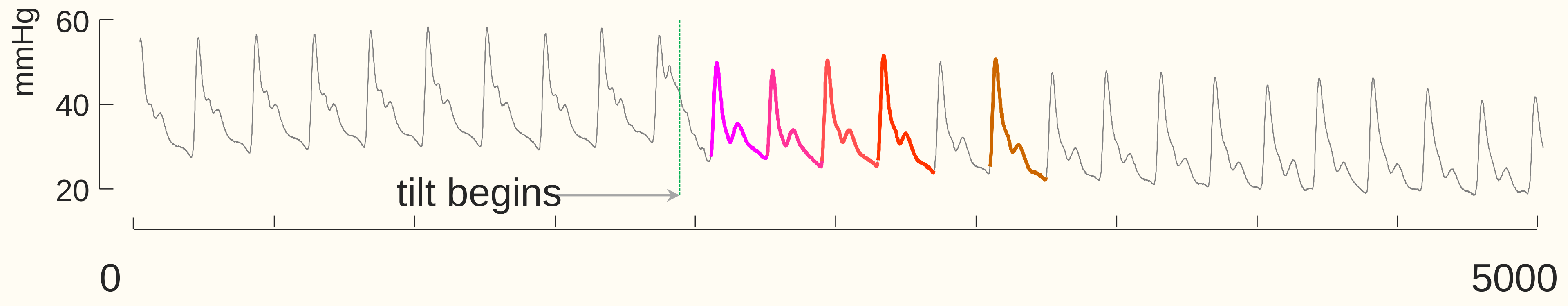
Arterial Blood Pressure



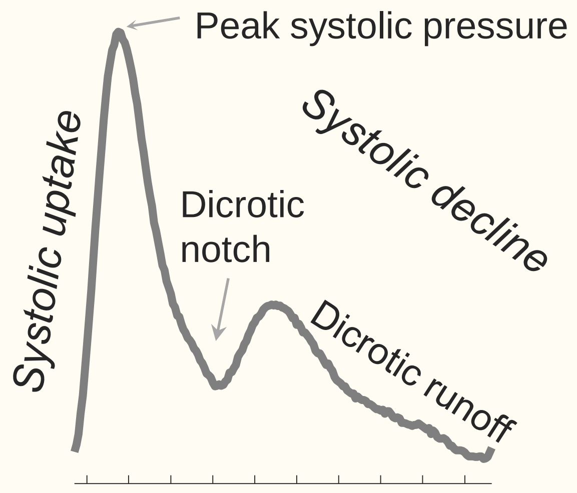
We will zoom-in to here in the next slide

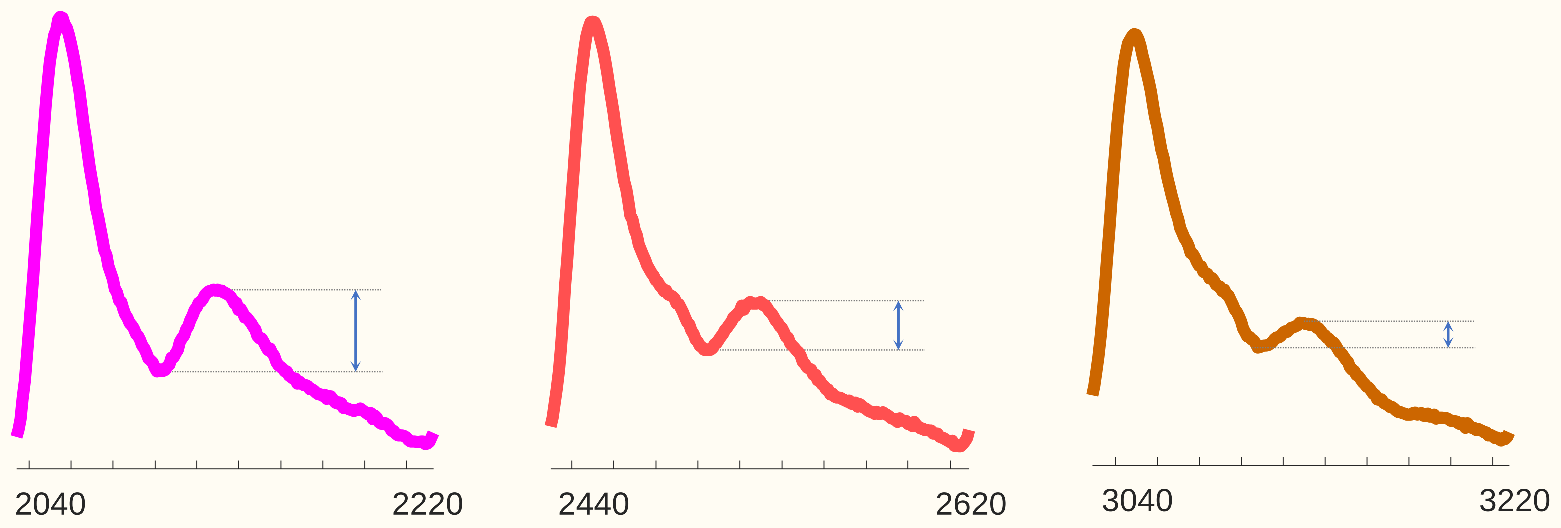
We ran time series chain discovery on the dataset. The only thing we tell it is the length of the subsequence to use (about one heartbeat long).

Zoom In



Ads the chain progresses, the depth of the dicrotic notch decreases….

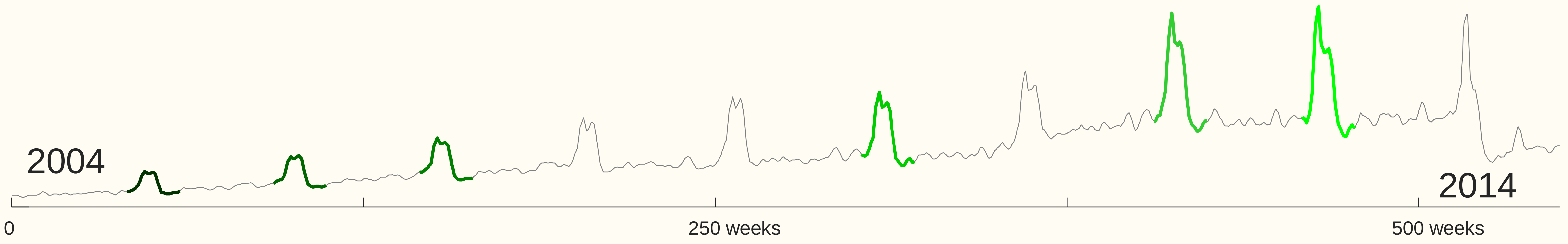


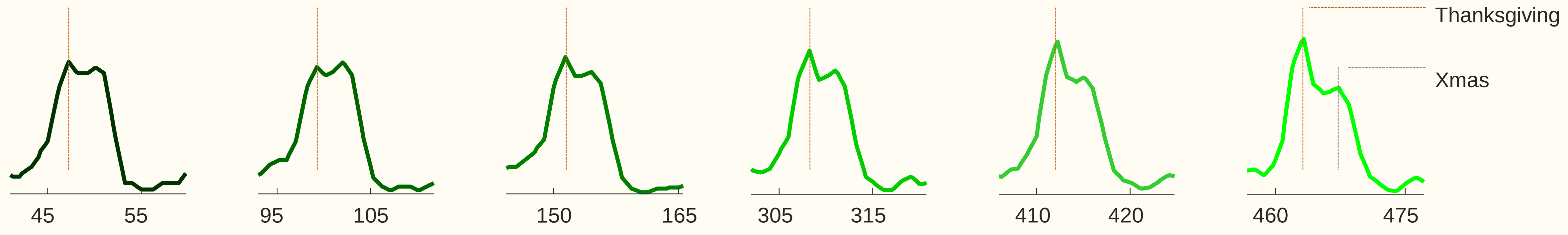


More Time Series Chains

We looked at the google query volume for *Kohl's*, an American retail chain.

The discovered chain shows that over the decade, the bump transitions from a smooth bump covering most of the period between thanksgiving and Xmas, to a more sharply focus bump centered on thanksgiving. This seems to reflect the growing importance of *Cyber Monday*, a marketing term for the Monday after Thanksgiving. The phrase was created by marketing companies to persuade people to shop online. The term made its debut on November 28th, 2005 in a press release entitled "*Cyber Monday Quickly Becoming One of the Biggest Online Shopping Days of the Year*" . Note that this date coincides with the first glimpse of the sharping peak in our chain.

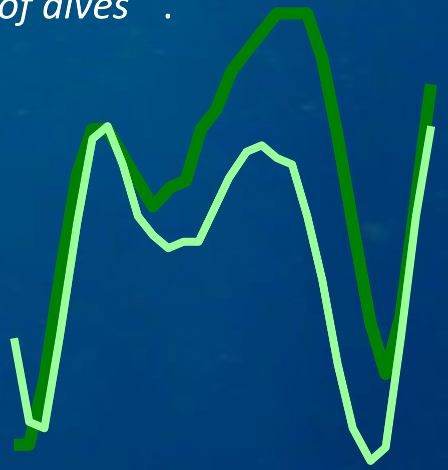


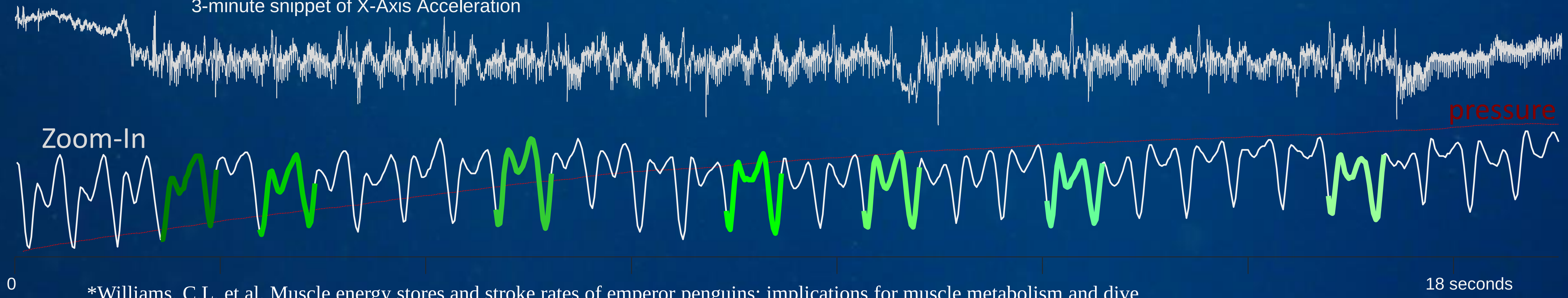


One Last Time Series Chain

Magellanic penguins regularly dive to depths of up to 50m to hunt prey. Penguins have typical body densities for a bird, but just before diving they take a very deep breath that makes them exceptionally buoyant. This positive buoyancy is difficult to overcome near the surface, but at depth, the compression of water pressure cancels it. In order to get to down to their hunting ground below sea level it is clear that "*locomotory muscle workload, varies significantly at the beginning of dives*"*.

The snippet of time series shown in does not suggest much of a change in *stroke-rate*, however penguins are able vary the thrust of their flapping by twisting their wings. The chains we discovered shows this dramatic and evolving sprint downwards leveling off to a comfortable cruise.





*Williams, C.L. et al. Muscle energy stores and stroke rates of emperor penguins: implications for muscle metabolism and dive performance. *Physiological and Biochemical Zoology*.*85.2*(2011):120-133

Photo by Paul J. Ponganis

- There are literally 100’s of time series anomaly detectors.
- However, many claim that *Time Series Discords* is among the best.

  *..on 19 different publicly available data sets, comparing 9 different techniques (time series **discords**) is the best overall technique among all techniques.* Vipin Kumar*

- This is good news for us, because if you compute the matrix profile, you have the discords “for free”. In fact, you have *all* the top K-discords, for any K.
- Why are discords so effective? (our subjective opinion)
  - They make no assumptions about the data (so no *wrong* assumptions).
  - They don’t need to learn a bunch of parameters, with no parameters to fit, it is hard to *over*fit.
- There is one pathological (but fixable) case where they don’t work (next slide)

Vipin Kumar
ACM SIGKDD
2012 Innovation
Award Winner

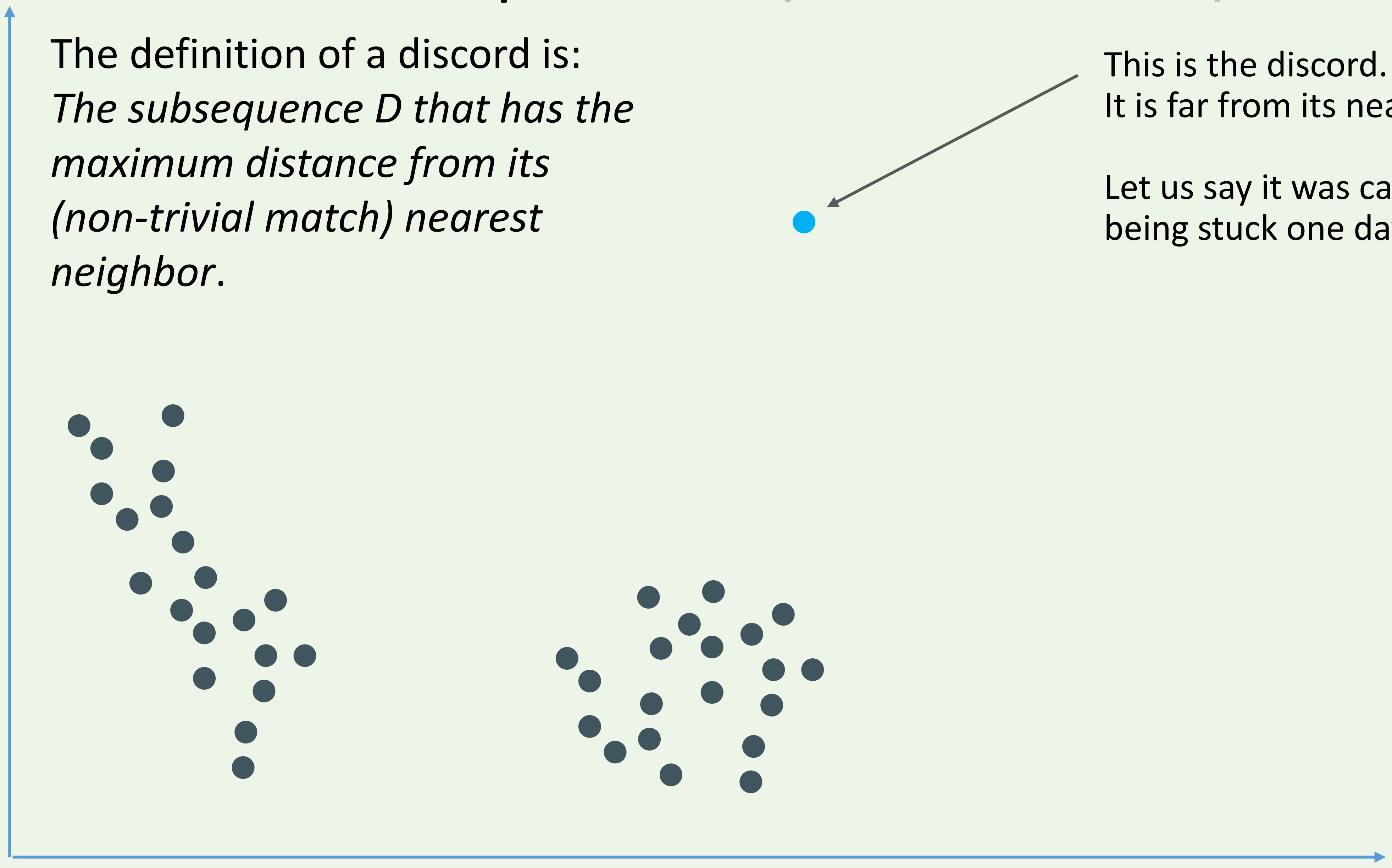
*https://www.cs.umn.edu/research/technical_reports/view/09-004

The twin freak problem (see next slide)

The definition of a discord is:
The subsequence D that has the maximum distance from its (non-trivial match) nearest neighbor.

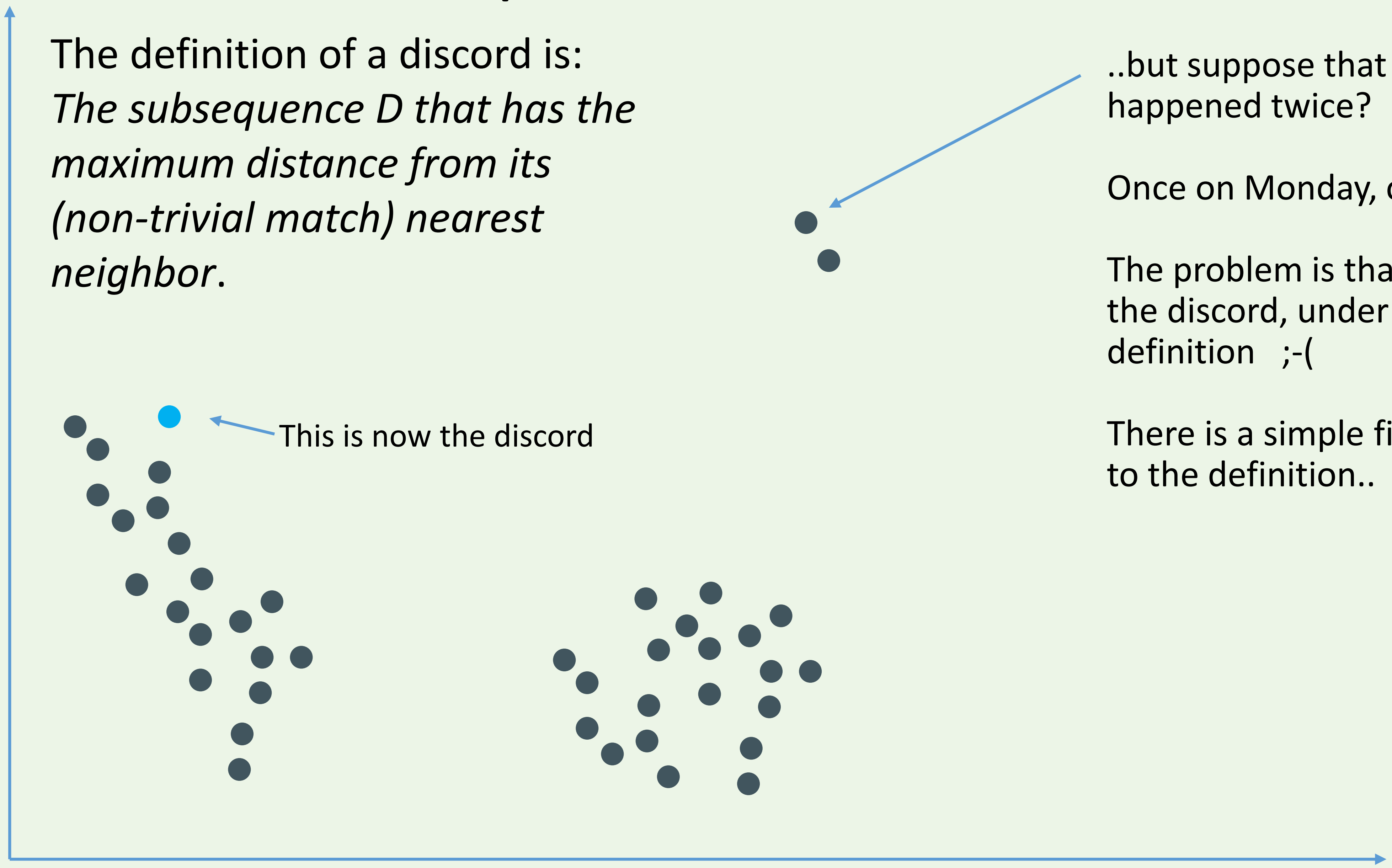
This is the discord.
It is far from its nearest neighbor

Let us say it was caused be a valve being stuck one day..



The twin freak problem

The definition of a discord is:
The subsequence D that has the maximum distance from its (non-trivial match) nearest neighbor.



..but suppose that the anomaly happened twice?

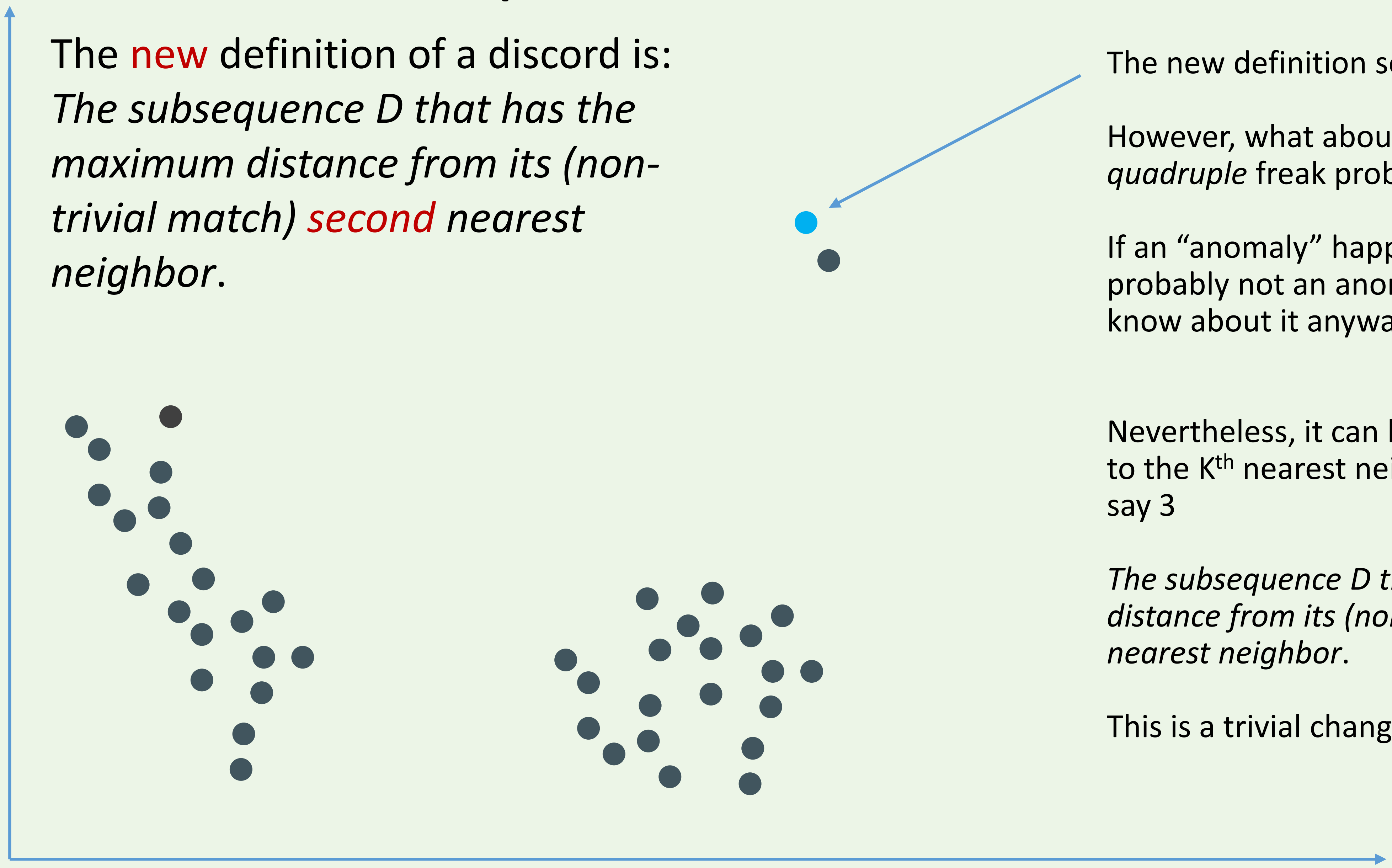
Once on Monday, once on Friday…

The problem is that it is no longer the discord, under our classic definition ;-(

There is a simple fix, a minor change to the definition..

The twin freak problem

The new definition of a discord is:
The subsequence D that has the maximum distance from its (non-trivial match) second nearest neighbor.



The new definition solves the problem.

However, what about the *triple* freak, or *quadruple* freak problem etc.…

If an “anomaly” happens many times, it is probably not an anomaly, and we probably know about it anyway.

Nevertheless, it can be useful to generalize to the Kth nearest neighbor, for a small K, say 3

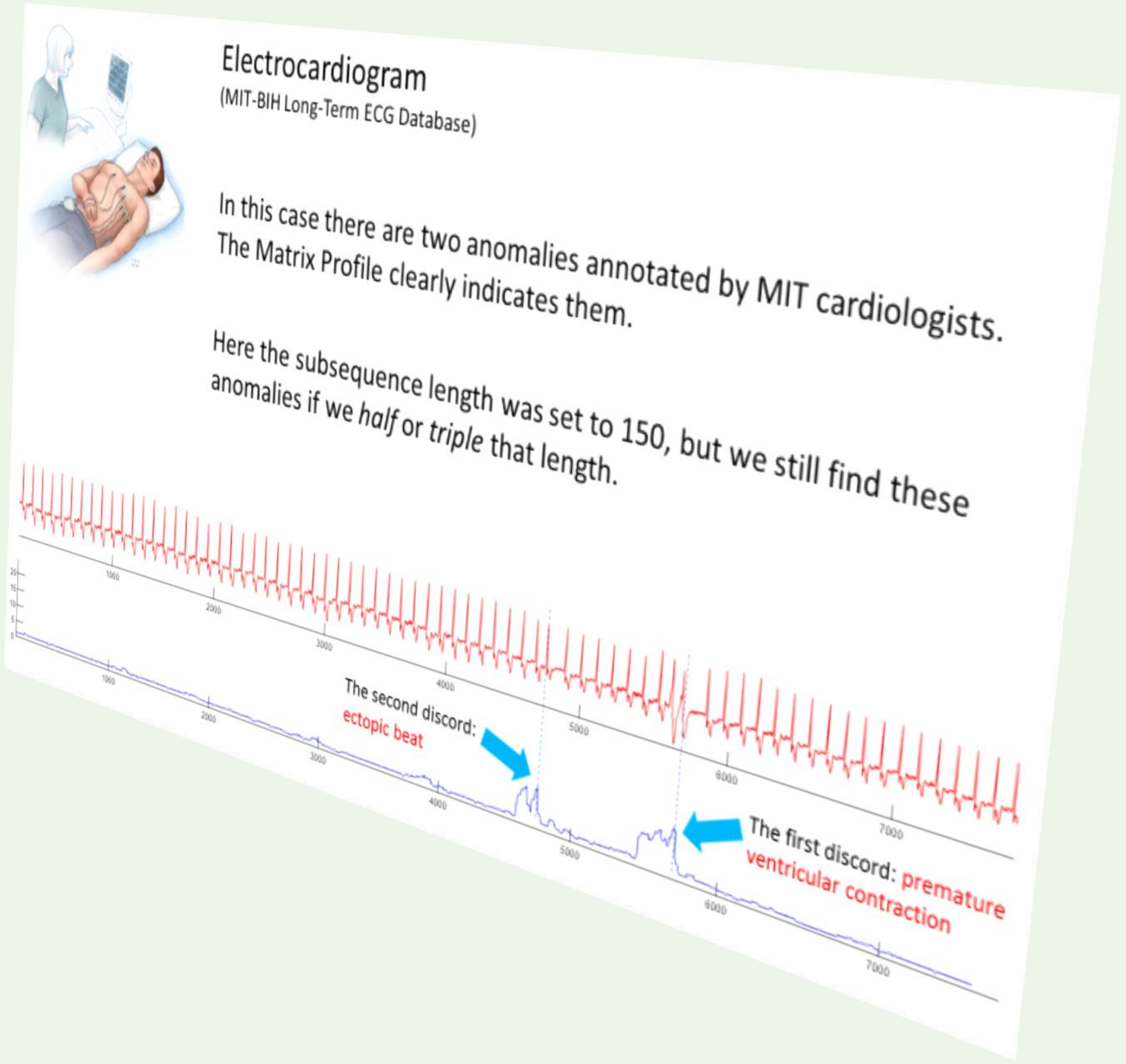
The subsequence D that has the maximum distance from its (non-trivial match) K nearest neighbor.

This is a trivial change/addition to the MP

We have already seen examples of time series discords (although we did not explicitly call them that) so we will not revisit this here.

Discords are simply high values in the Matrix Profile.

There *are* many other algorithms to find discords. But why bother with them, when the Matrix Profile gives you them for free?



Electrocardiogram

(MIT-BIH Long-Term ECG Database)

In this case there are two anomalies annotated by MIT cardiologists. The Matrix Profile clearly indicates them.

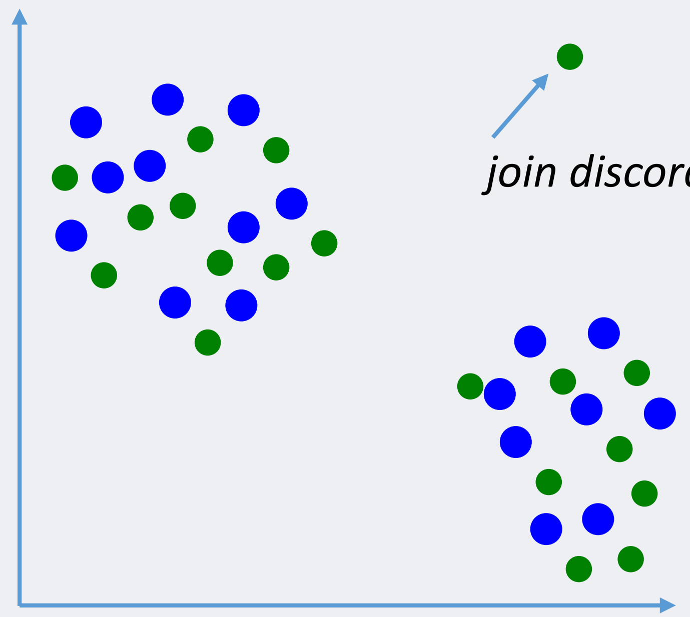
Here the subsequence length was set to 150, but we still find these anomalies if we *half* or *triple* that length.

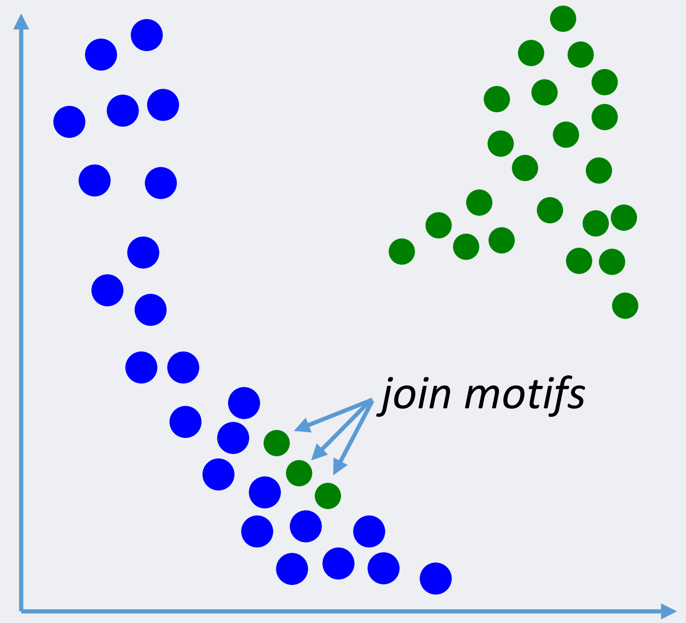
The second discord: ectopic beat
The first discord: premature ventricular contraction
1000
2000
3000
4000
5000
6000
7000

Generalizing to Joins

- We can think of the MP as a type of similarity self-join. For every subsequence in T_A, we join it with its nearest (non- trivial) neighbor in T_A, or $\mathbf{J}_{T_A T_A}$ *or* $T_A \bowtie 1\text{nn } T_A$
- This is also known as all-pairs-similarity-search (or similarity join).
- However, we can genialize to an **AB-**similarity join. For every subsequence in **A**, we join can it with its nearest neighbor in **B**, or $\mathbf{J}_{T_A T_B}$ *or* $T_A \bowtie 1\text{nn } T_B$
- Note that in general: $\mathbf{J}_{T_A T_B} \neq \mathbf{J}_{T_B T_A}$
- Note that **A** and **B** can be radically different sizes
- We may be interested in:
    - What is *conserved* between two time series (the join motifs)
    - What is *different* between two time series (the join discords)
- The tricks for understanding and reading join-based MPs are the same as before, we will see some examples to make that clear.

Generalizing to Joins





Two scenarios of interest: we do a $\mathbf{J}_{T_A T_B}$…

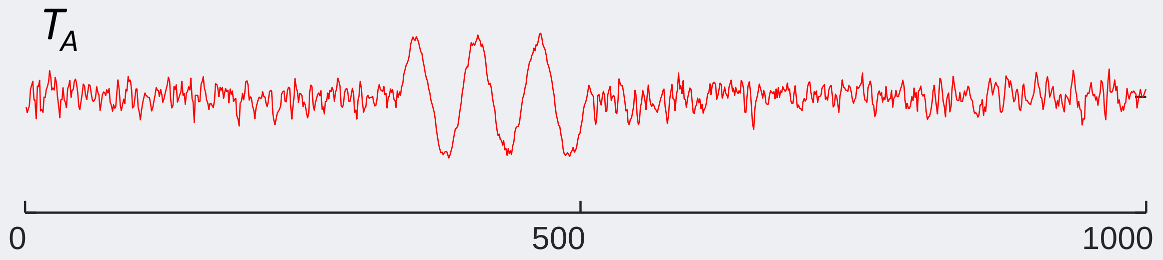
1) **The Golden Batch**: Here we have two time series that we think should be about the same. But when we join them, there is a *join discord*, a subsequence that appears only in only in **A**, but not in **B**, but why? (*spoken word* example below)

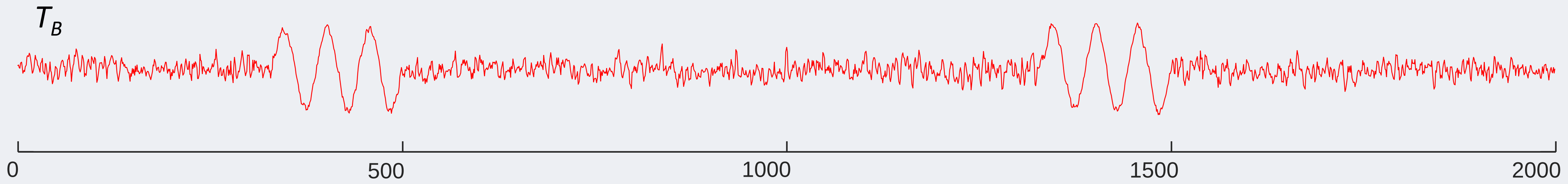
2) **The Suspicious Similarity:** Here we have two time series that we have *no* reason to think should be the same. But when we join them, there are *join motifs*, some subsequences from **A** appear in **B**, but why? (*music* example below. Another example would result from "meter swapping")

Now let us consider the join of two time series.

Assume we have two time series T_A and T_B ...

Note that they can be of different lengths

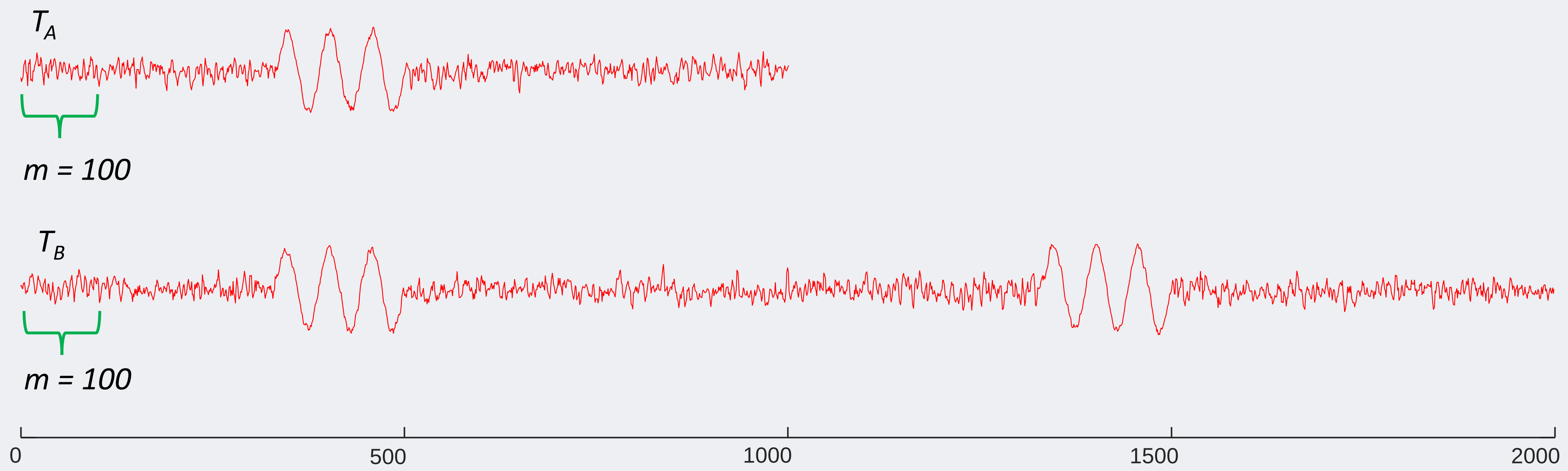




$|\ T_A\ | = 1{,}000$ $|\ T_B\ | = 2{,}000$

As before, we are not interested in any *global* properties of the time series, we are only interested in small *local* subsequences, of this length, *m* $\underbrace{\hspace{3em}}$

These subsequences might be about the length of individual heartbeats (for ECGs), individual days (for social media behavior), individual words (for speech analysis) etc.



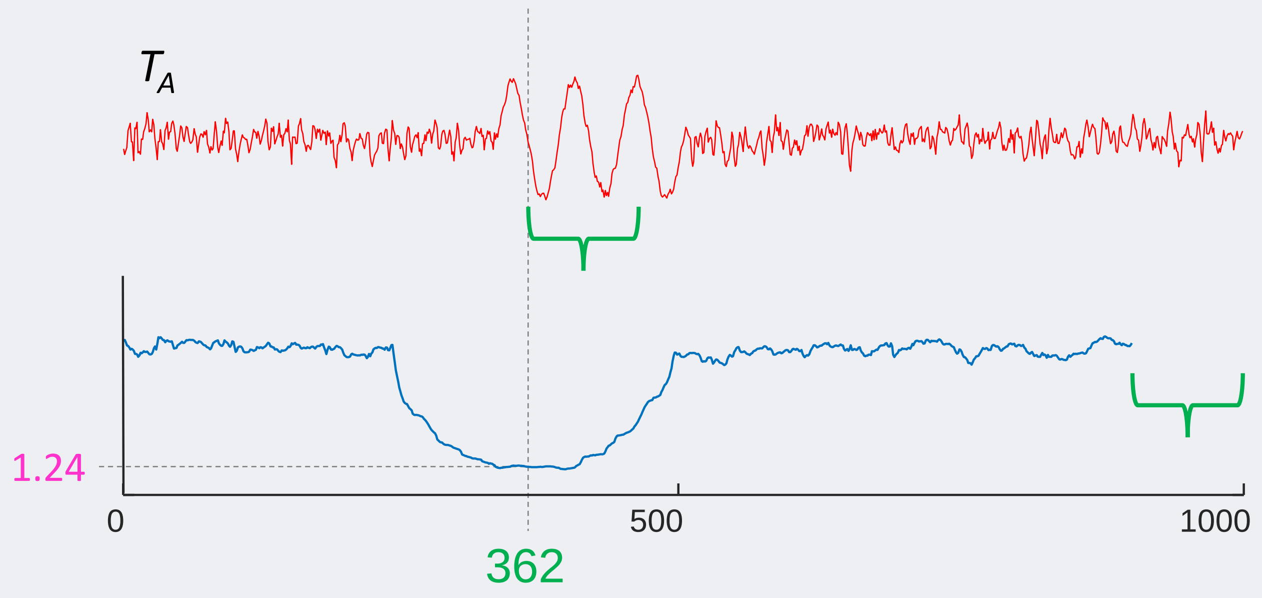
We can create a companion Matrix Profile of T_A.

For every subsequence in T_A, we look for its nearest neighbor in T_B.

The Matrix Profile at the ith location records the distance of the subsequence in T_A, at the ith location, to its nearest neighbor in T_B, under z-normalized Euclidean Distance.

The Matrix Profile is almost the same length as T_A, it is shorter by just m $\underbrace{\qquad\qquad}$

For example, in the below, the subsequence of length 100 starting at 362 happens to have a distance of 1.24 to its nearest neighbor (wherever it is) in T_B .



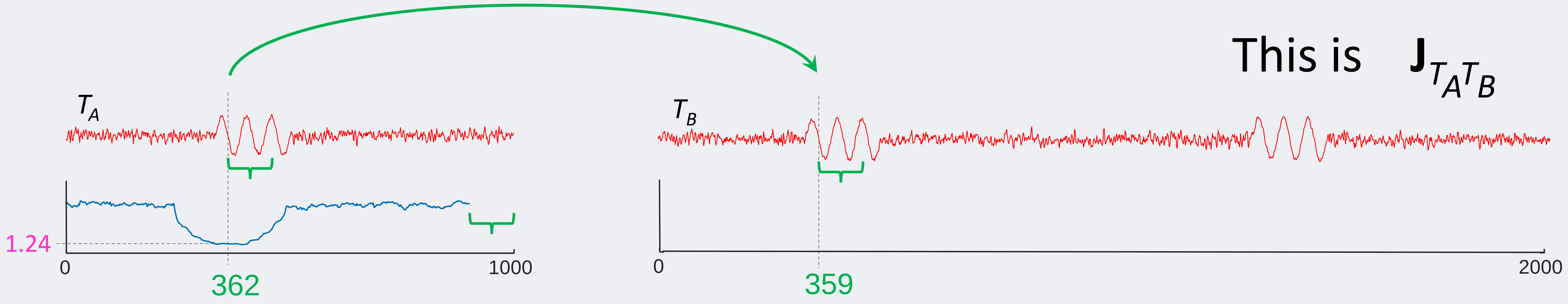
Informally: how far is each subsequence in T_A, from its nearest neighbor in T_b?

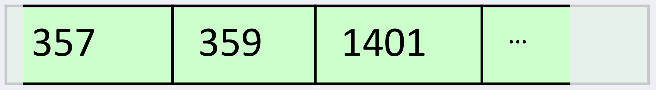
Recall that the Matrix Profile at the i^{th} location records the distance of the subsequence in T_A, at the i^{th} location, to its nearest neighbor in $T_{B,}$ under z-normalized Euclidean Distance.

However, it does not tell us where the *location* of the nearest neighbor in $T_{B.}$ To store this information, we can create another companion sequence, called a matrix profile index.

The green arrow points from the subsequence of T_A starting at 362 to its nearest neighbor in $T_{B.}$ The nearest neighbor locates at 359 of $T_{B\,.}$

This is $\mathbf{J}_{T_A T_B}$





357	359	1401	···

matrix profile index
(zoom in)

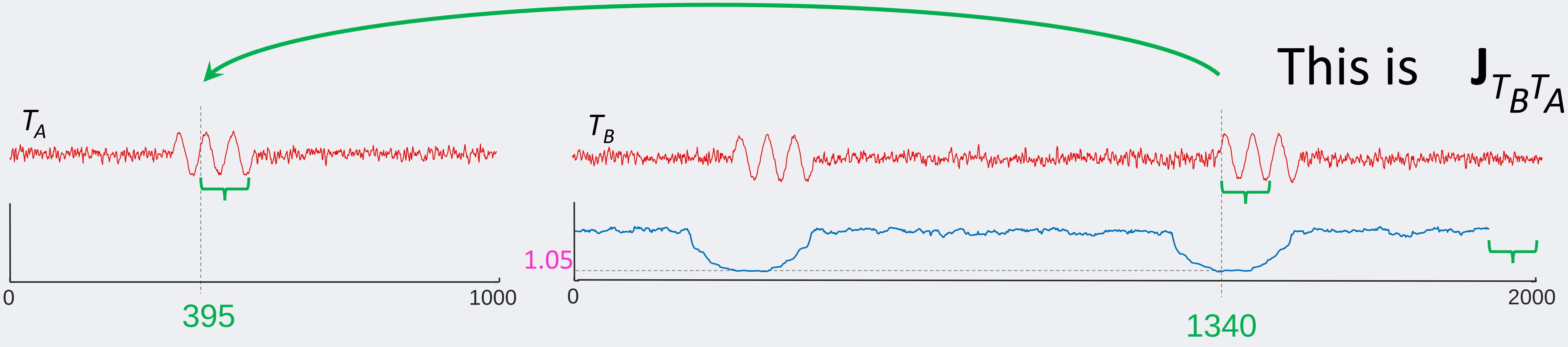
Informally: For each subsequence in T_A, point to its nearest neighbor in T_b

We can reverse the direction of the join…

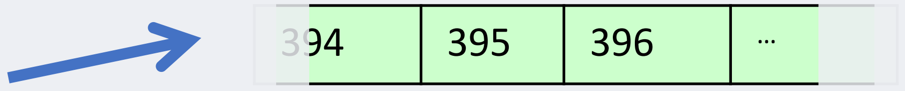
Here the matrix profile index tell us the *location* of the nearest neighbor of each subsequence in T_B.

The green arrow points from the subsequence of T_B starting at 1340 to its nearest neighbor in T_A. The nearest neighbor locates at 395 of T_A.

This is $\mathbf{J}_{T_B T_A}$



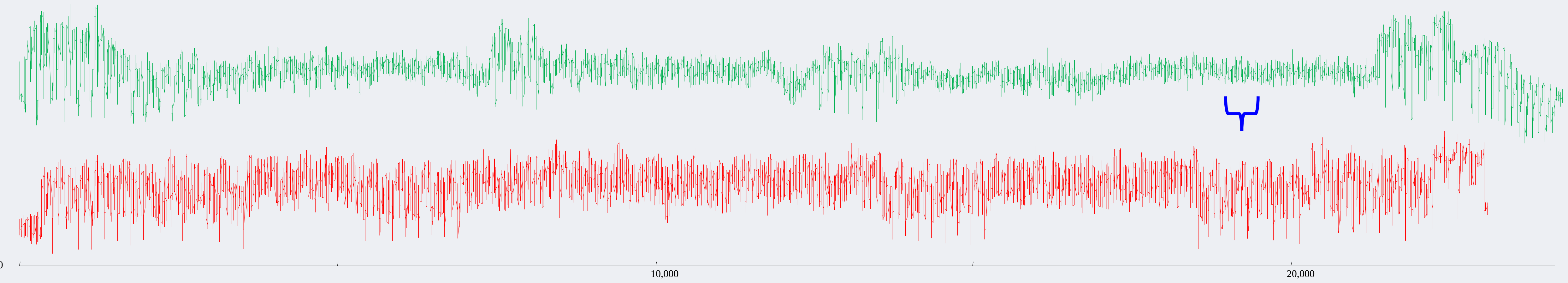
matrix profile index
(zoom in)



394	395	396	…

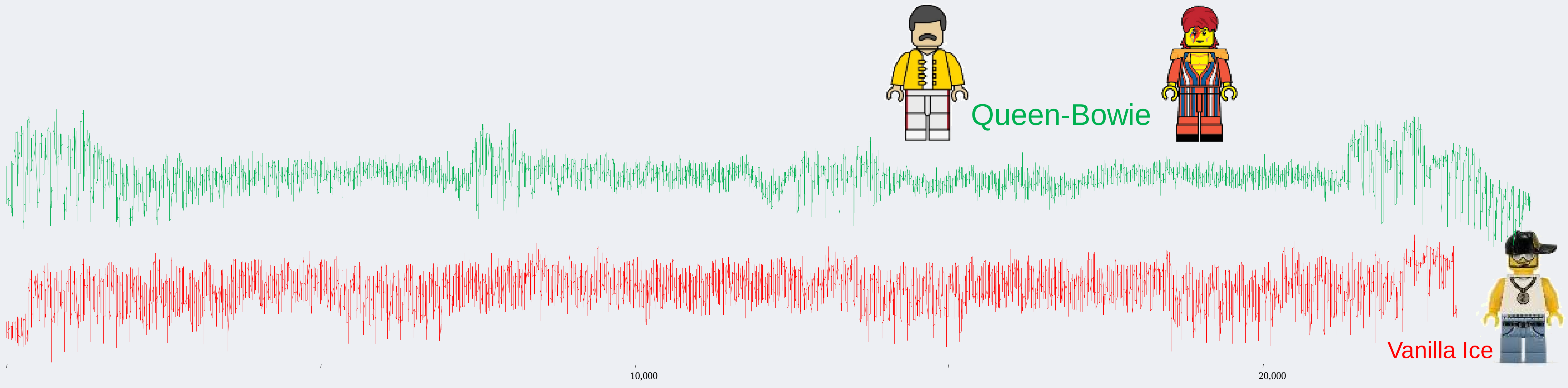
Music I (join case)

Can you see any common structure between the two time series below?
Hint, it is probably about this length $\underbrace{\qquad}$

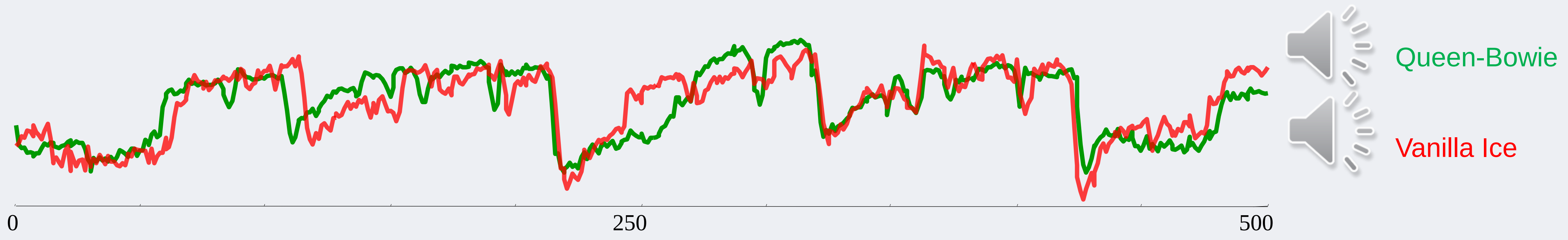


Music II (join case)

The data is the 2nd MFCC of two songs, *Under Pressure* and *Ice Ice Baby*



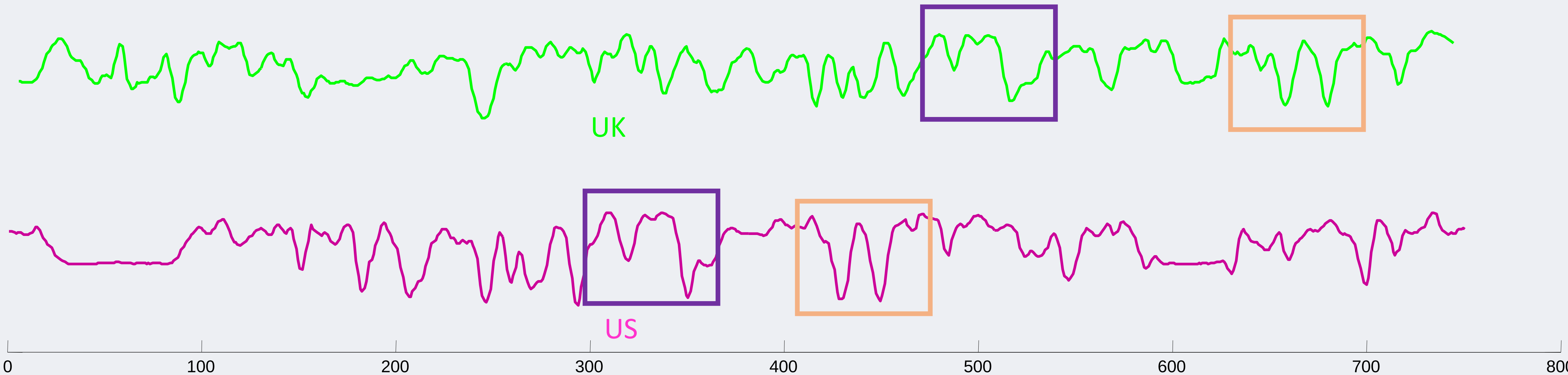
A zoom-in of the best conserved region between the two time series (the similarity join)



In the previous example we asked you to find "*common structure between the two time series*" Now I am going to ask you the opposite question.
What is *different* between the two time series?

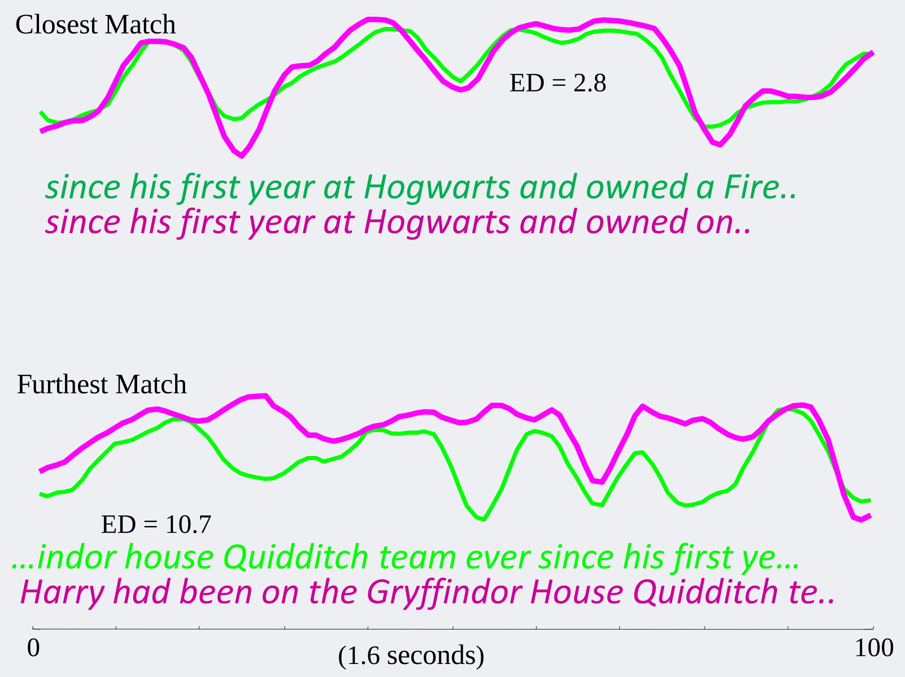
Hint, it is probably about this length $\underbrace{\qquad\qquad}$

Hint, it cant be the regions in the matching boxes, since they have matches…



SPOKEN WORD ||

Here the difference is due to a unique phrase that only appears in the USA version of the Harry Potter books.



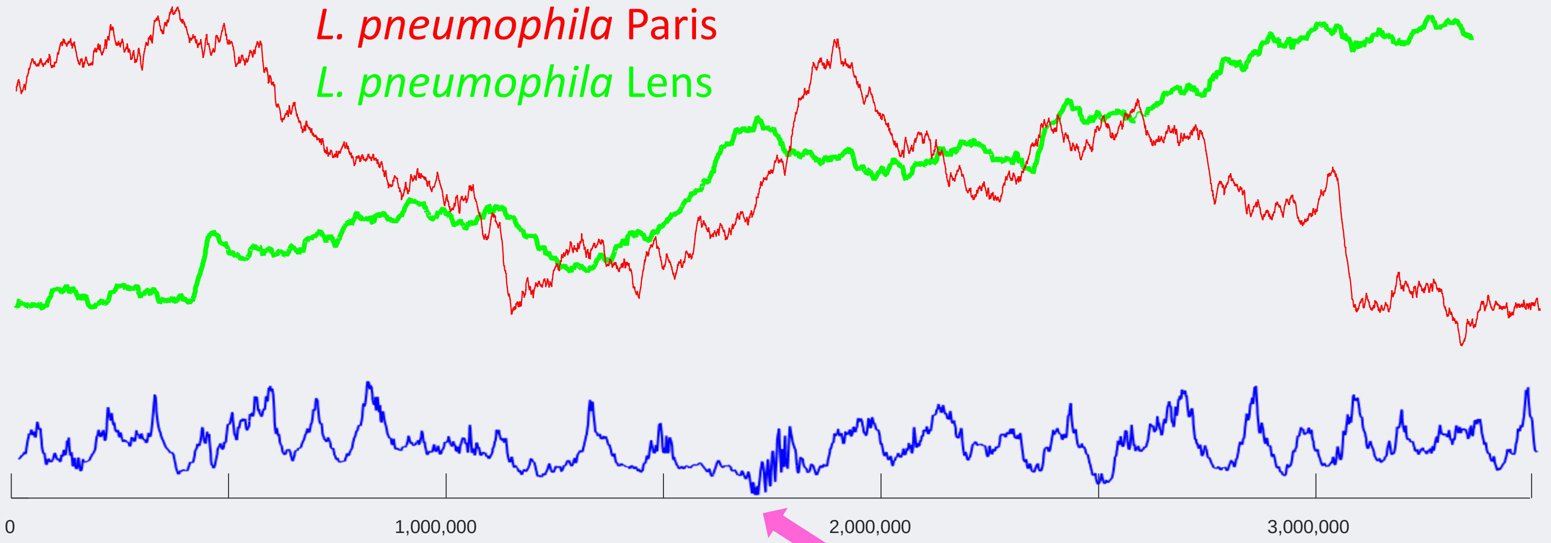
UK version : *Harry was passionate about Quidditch. He had played as Seeker on the Gryffindor house Quidditch team ever since his first year at Hogwarts and owned a Firebolt, one of the best racing brooms in the world*...

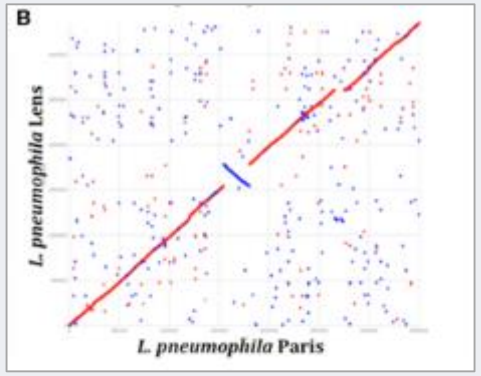
USA version : ***Harry had been on the Gryffindor House Quidditch te****am ever since his first year at Hogwarts and owned one of the best racing brooms in the world, a Firebolt*.

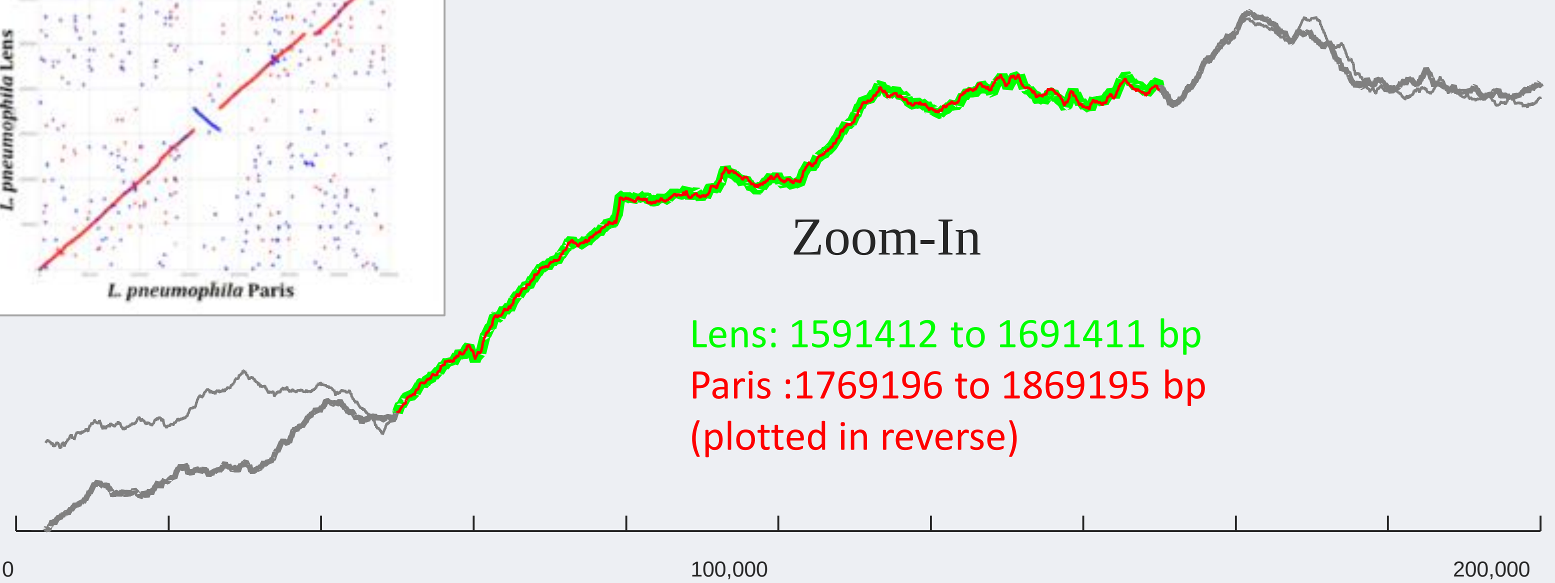
DNA (join case)

We consider two strains of Legionella bacteria, *L. pneumophila Paris* and *L. pneumophila Lens*, which consist of 3,503,504 and 3,345,567 bp respectively. We consider a subsequence length of 100,000

However, we flipped one of the time series "backwards", before computing the join.



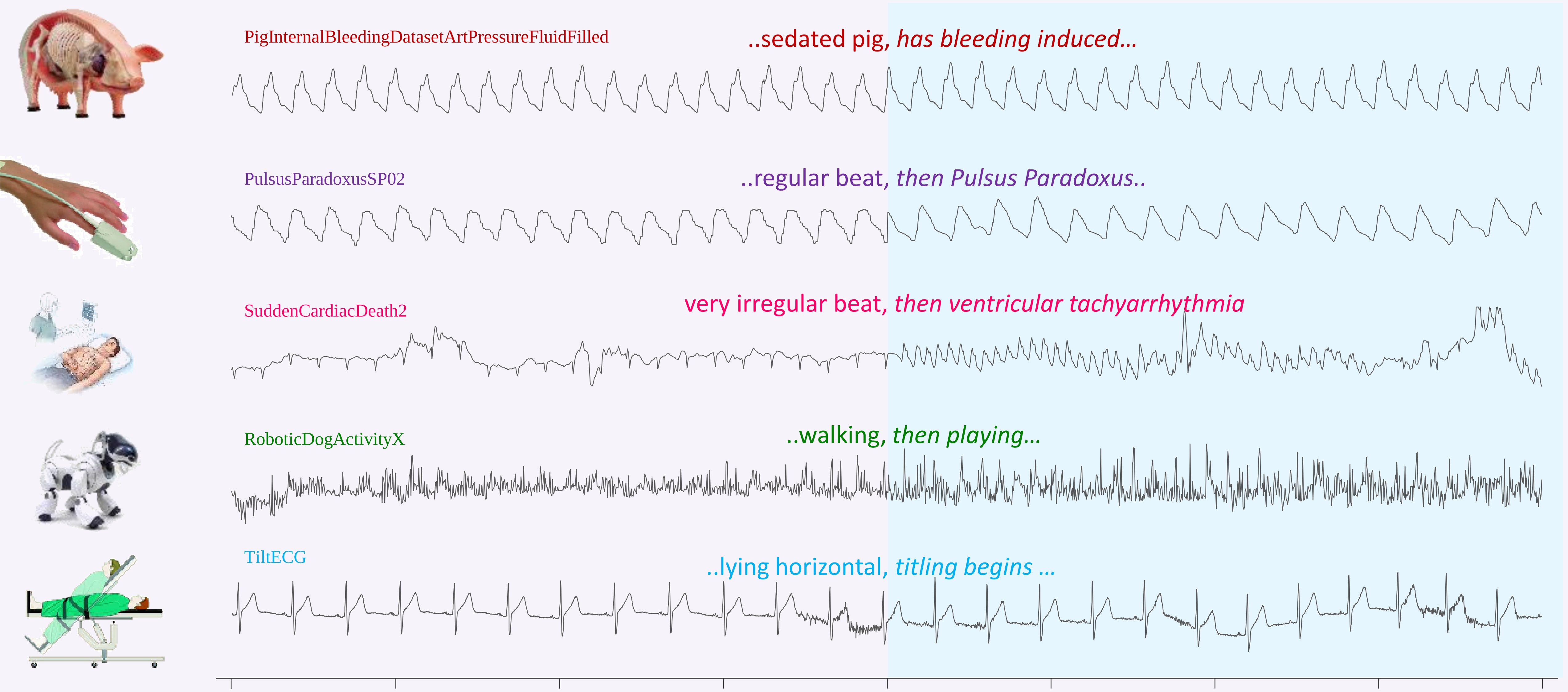




Laura Gomez-Valero et al. Comparative and functional genomics of *Legionella* identified eukaryotic like proteins as key players in host–pathogen interactions

Time Series Semantic Segmentation

Sometime the system we are monitoring changes regimes, can we detect such changes?

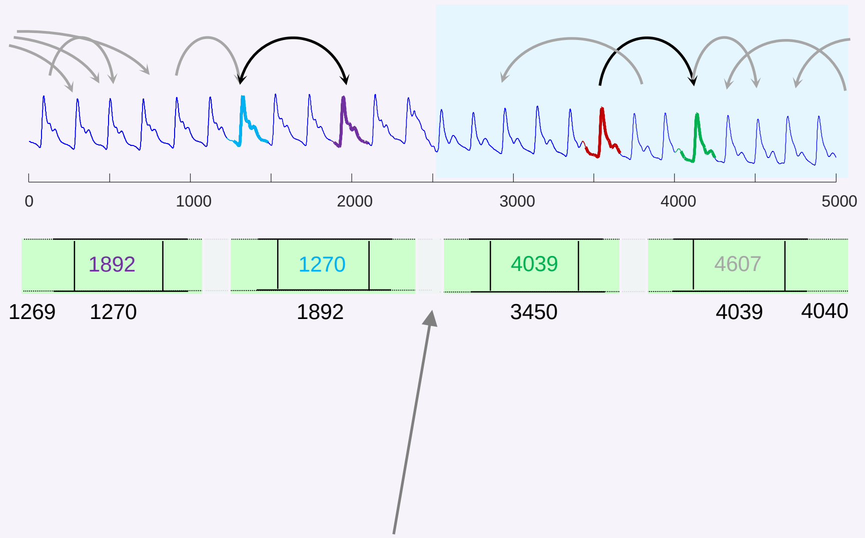


FLOSS: Matrix Profile Segmentation

What do we want in a Semantic Segmentation Algorithm?

- **Handle fast online, or huge batch data**
- **Domain agnosticism**. It would be nice to have a single algorithm the works on all kinds of data
- **Parameter-Lite**. (Tuning parameters almost guarantees overfitting).
- **High accuracy**.
- **Able to report *degree* of segmentation or confidence** (A binary decision is too brittle in most cases).

- **Claim:** We can do all this by looking at the Matrix Profile Index…



No arcs seem to cross here

Key Observation

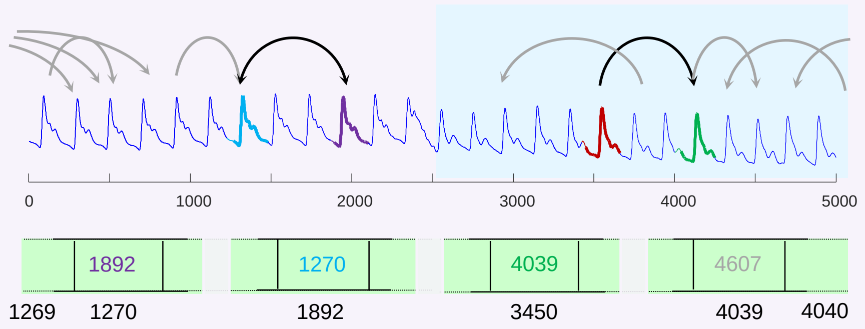
Recall that the Matrix Profile Index has pointers (*arrows*, *arcs*) that point to the nearest neighbor of each subsequence.

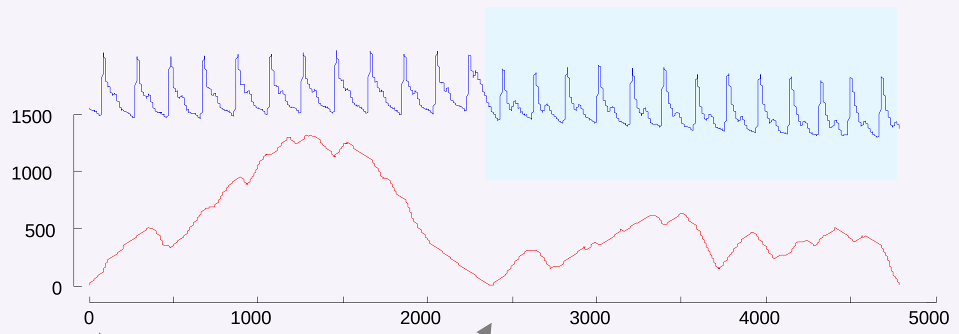
If we have a change of behavior, say from *walk* to *run*, we should expect very few arrows to span that change of behavior.

- Most *walk* subsequences will point to another *walk* subsequence
- Most *run* subsequences will point to another *run* subsequence
- Rarely, if ever, will a *run* point to a *walk*.

So, if we slide across the Matrix Profile Index, and count how many arrows cross each particular point, we expect to find few that span the change of behavior.

Lets try this, next slide…





The arc count here is almost zero!

But the beginning (and end) are also near zero.

This works!

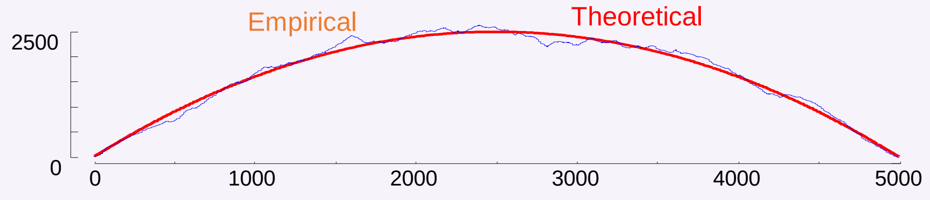
If we use the sliding arc count to produce an arc-curve, we find it is near zero at the point of system transition. This low value signals the location of the system change.

There is one flaw. The arc-curve, tends to be low near the beginning and end of the time series, just because there are fewer arcs that *could* cross at those locations.

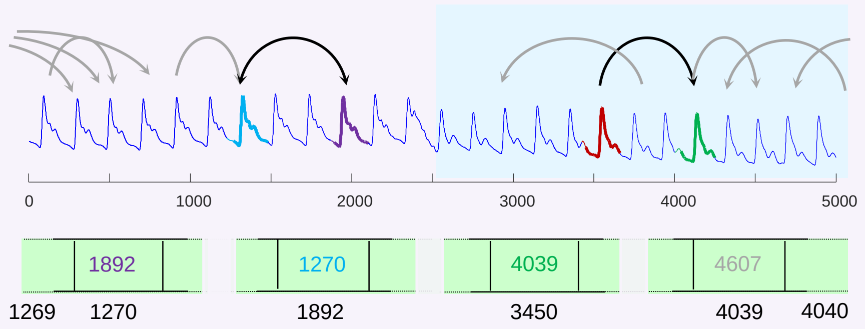
What we can do is calculate what the arc-curve would look like if there was *no* system transition, and use that to *correct* the arc-curve.

If there was no system transition structure, the arc-curve would be a inverted parabola, with a height ½ the time series length.

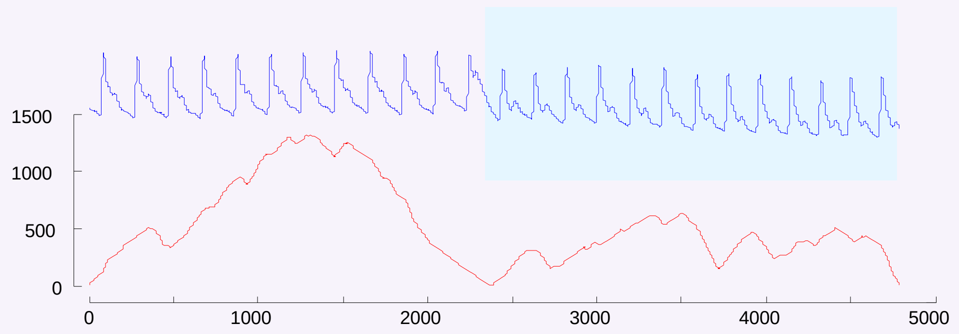
Lets try this, next slide…



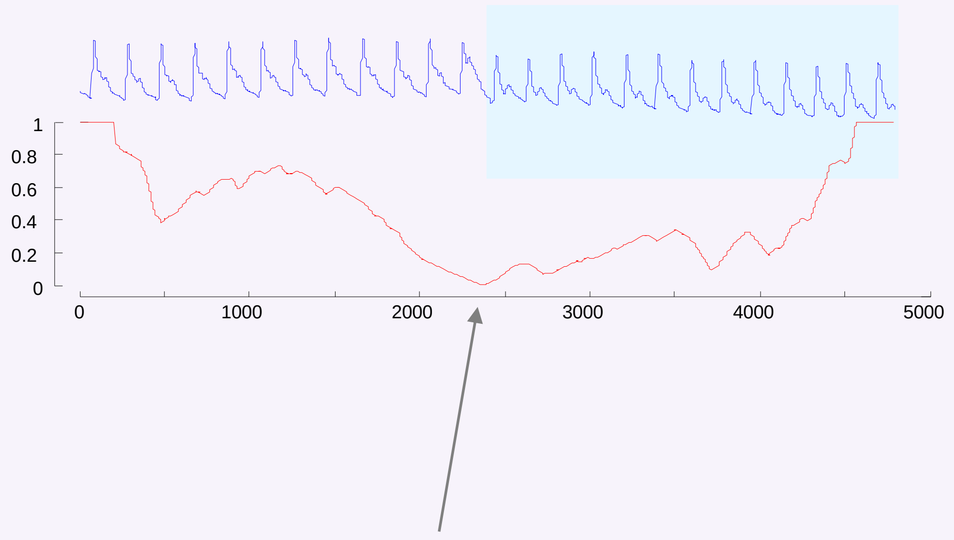
The number of arcs that cross a given index, if the links are assigned randomly



This works even better!



The corrected arc-curve minimizes in the right place, and does not have spurious minimizations at the beginning and end.



How robust is the corrected arc-curve? Lets add some distortions to the data, and see what it does. Lets try this, next slide…

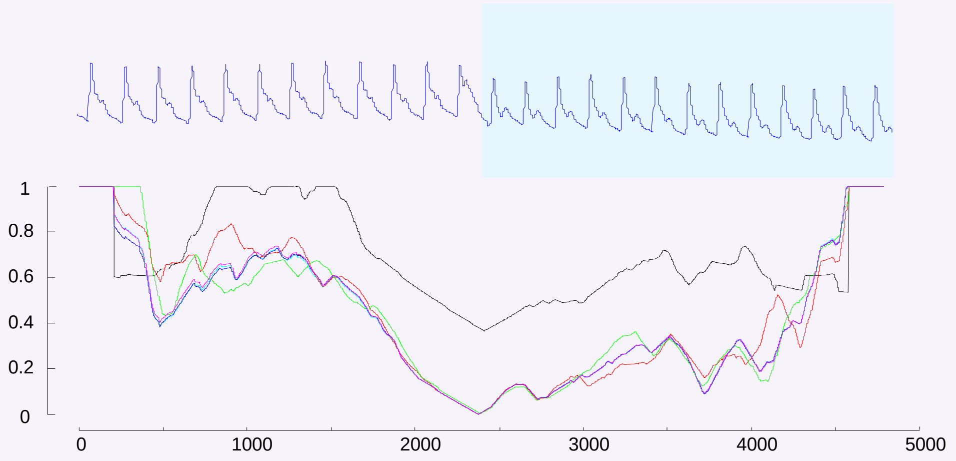
The corrected arc-curve here is almost zero

FLOSS is very robust to the data's properties

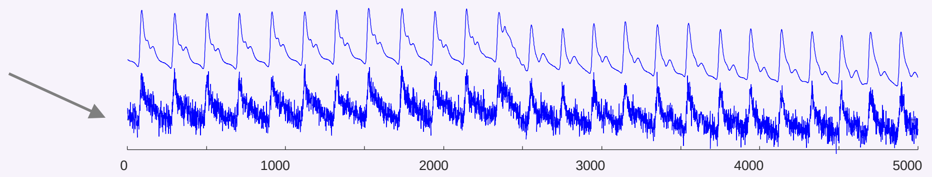
Consider the following distortions

- Downsampling from the original 250 Hz to 125 Hz (red).
- Reducing the bit depth from 64-bit to 8-bit (blue).
- Adding a linear trend of ten degrees (cyan).
- Adding 20dB of white noise (black).
- Smoothing, with MATLAB's default settings (pink).
- Randomly deleting 3% of the data, and filling it back in with simple linear interpolation (green).

Most distortions make almost no difference. The only one that does move the CAC significantly was adding noise. But even then we **still** find the correct segmentation, and we added a *lot* of noise



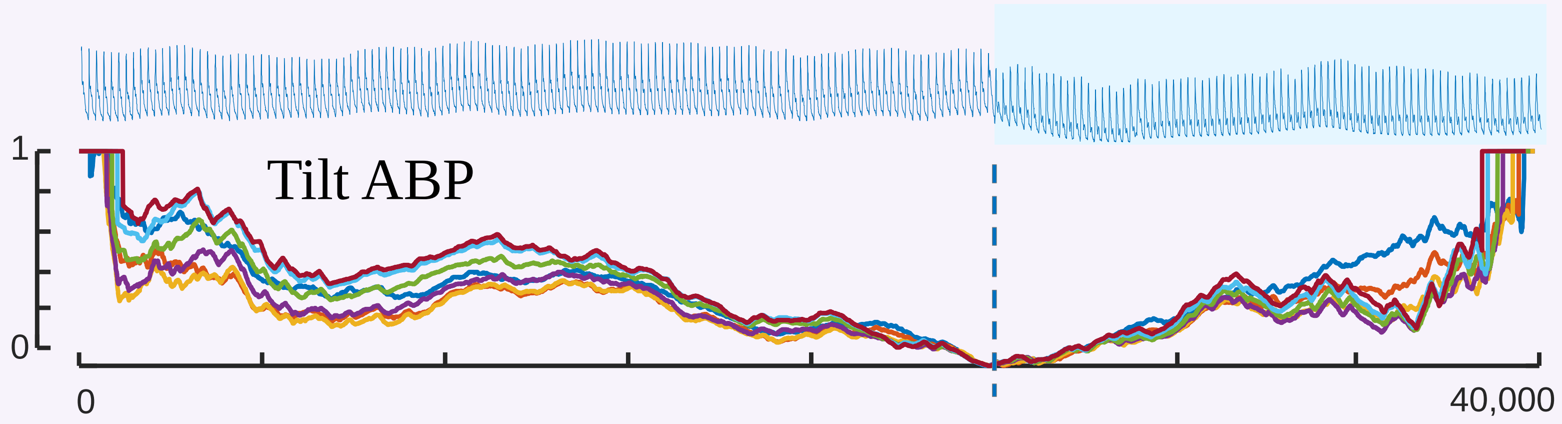
We added a *lot* of noise

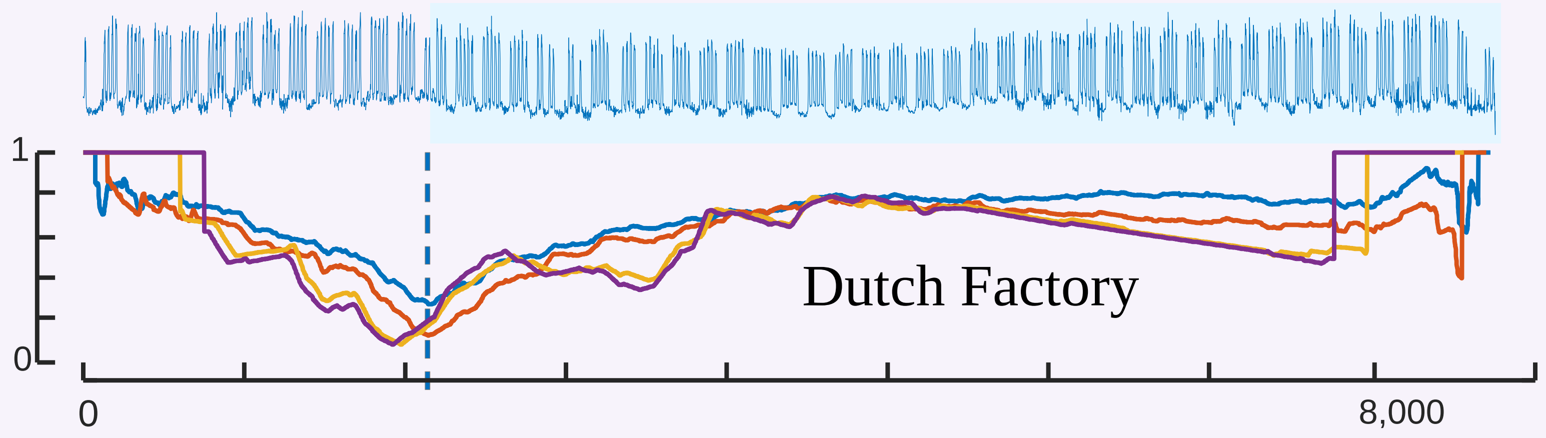


FLOSS is very robust to its only parameter

The CAC has a single parameter, the subsequence length *m*.

But we can typically change it by an order of magnitude, and get very good results.





The CAC computed for:

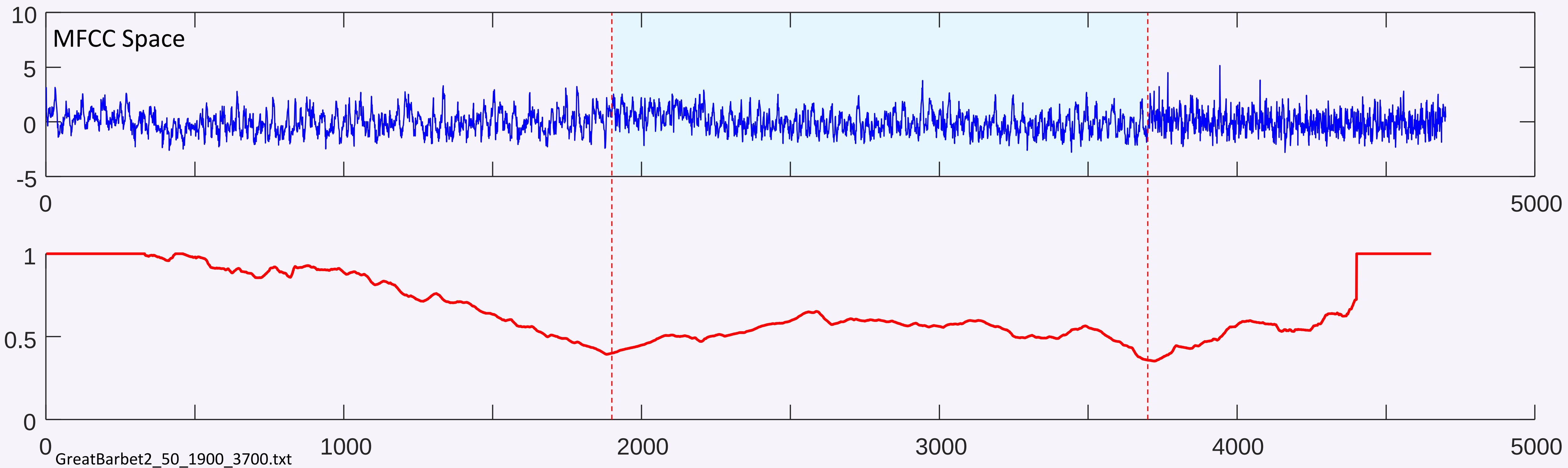
(*top*) TiltABP with *m* = {100, 150, 200, 250, 300, 350, 400}

(*bottom*) DutchFactory for *m* = {25, 50, 200, 250}. Even for this huge range of values for *L*, the output of FLOSS is essentially unchanged.

Great Barbet

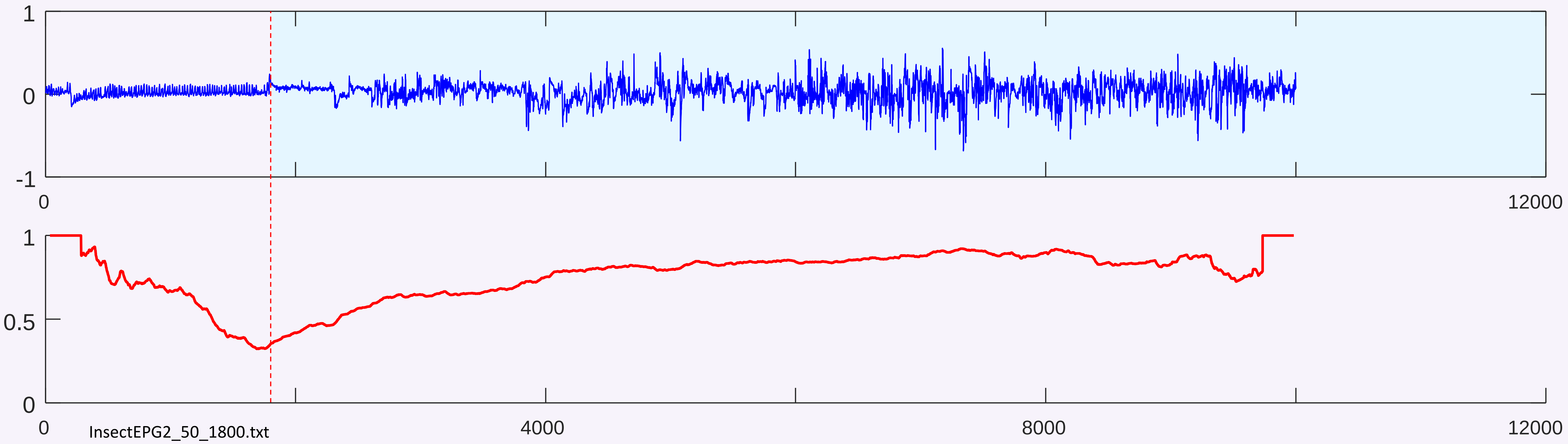
(Psilopogon virens)

One individual Great Barbet sings…, ….another takes over…, …yet another takes over

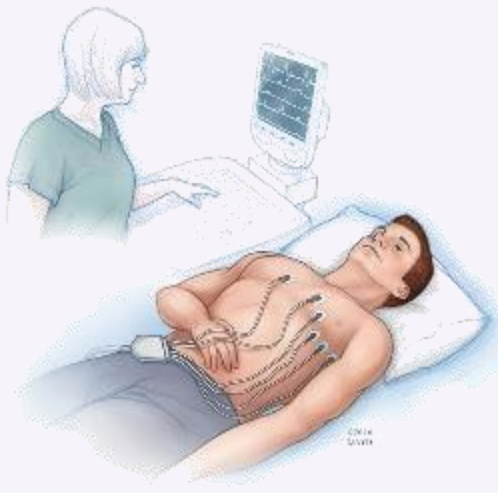


Asian citrus psyllid
(Diaphorina citri)

This dataset was hand annotated by an entomologist. The insect changes its feeding behavior at about time 1,800.

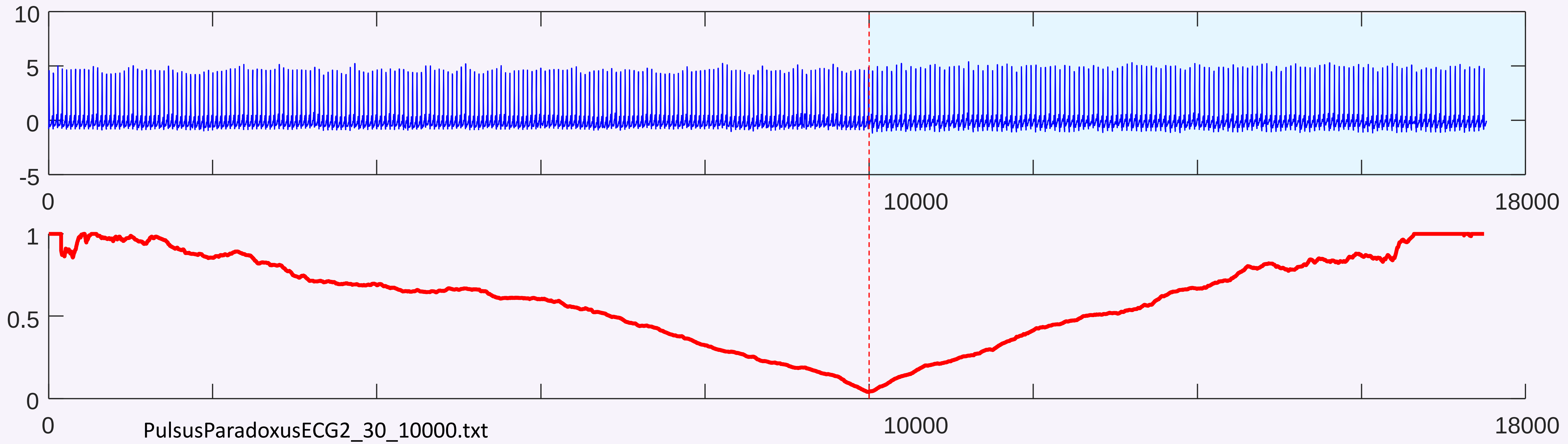


Pulsus Paradoxus is often visually apparent in the $SP0_2$ trace. Here we deliberately ignore this fact, and look *only* in the ECG trace, which is normally considered as *not* predictive of Pulsus Paradoxus.



Note that the clinician that annotated this data was *in the room* at the time and may have had access to information that is simply not available in this signal.

Pulsus paradoxus (PP), also paradoxic pulse or paradoxical pulse, is an abnormally large decrease in systolic blood pressure and pulse wave amplitude during inspiration.
See also https://www.youtube.com/watch?v=7AXIYQK5BBM



PulsusParadoxusECG2_30_10000.txt

Summary for Time Series Segmentation

The Matrix Profile allows a simple algorithm, FLUSS/FLOSS, for time series segmentation.

Because it is built on the MP, it inherits all the MPs properties

- It is incrementally computable, i.e. *online* (This variant is called FLOSS)
- It is fast (at least 40 times faster than real-time for typical accelerometer data)
- It is domain agnostic (But you can use the AV to add domain knowledge, see below)
- It is parameter-lite (only one parameter, and it is not sensitive to its value)

- It has been tested on the largest and most diverse collection of time series ever considered for this problem, and in spite *of* (or perhaps, *because of*) its simplicity, it is state-of-the-art. Better than rival methods, and better than *humans* (details offline).

From Domain Agnostic to Domain Aware*

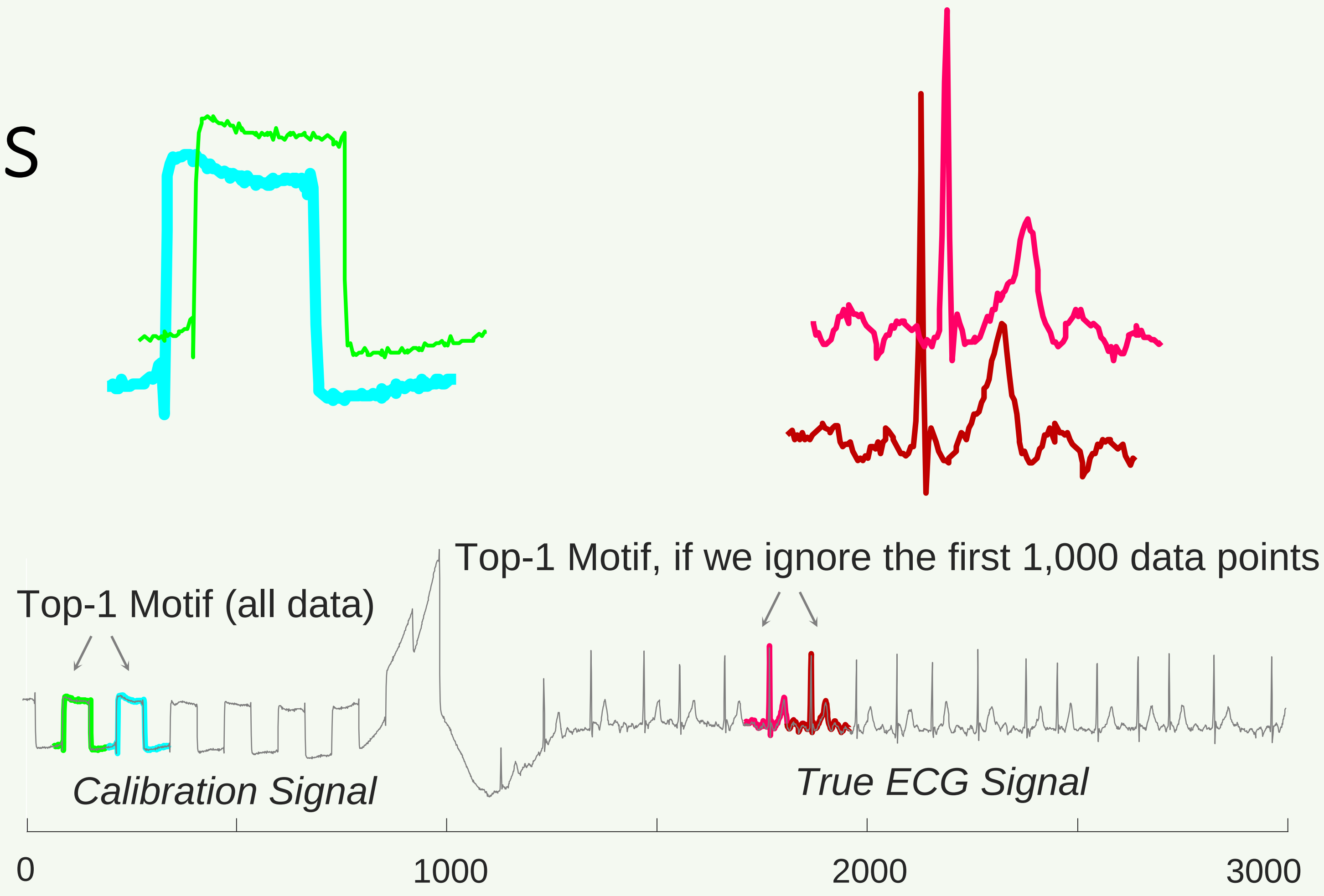
- The great strength of the MP is that is *domain agnostic*. A single black box algorithm works for taxi demand, seismology, DNA, power demand, heartbeats, music, bird vocalizations....
- However, in a handful of cases, there is a need to, or some utility in, incorporating some domain knowledge/constraints.
- There is a simple, generic and elegant way to do this, using the *Annotation Vector* (AV).
- In the following slides we will show you the *annotation vector* in the context of motif discovery, but you can use it with any MP algorithm.
- We will begin by showing you some examples of spurious motifs that can be discovered in particular domains, then we will show you how the AV mitigates them.

*Hoang Anh Dau and Eamonn Keogh. Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. KDD'17, Halifax, Canada.

Motivating Example 1: Stop-word Motif Bias

In some medical datasets, the true motifs may be "swamped" by more frequent, but biologically meaningless patterns. Much like the stop words "*and*" and "*the*" in text mining.

Here the approximate square wave is just a calibration signal, sent when the sensor has weak contact with the skin. It is a frequent, but spurious motif.



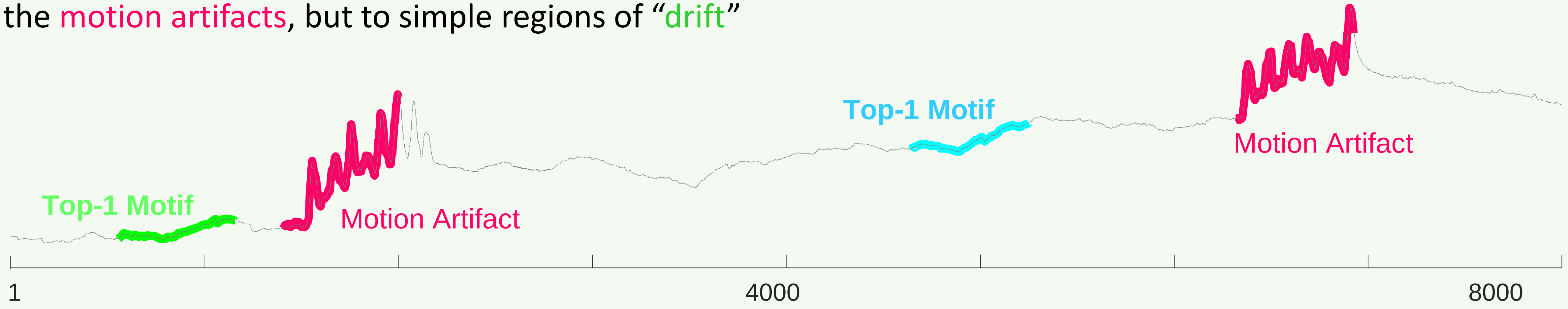
A snippet of ECG data from the LTAF-71 Database. The top motifs come from regions of the calibration signal because they are much more similar than the motifs discovered if we search only data that contains true ECGs.

Motivating Example 2: Simplicity Bias

Euclidean Distance has a bias toward simple shapes.

“*Pairs of complex objects, even those which subjectively may seem very similar to the human eye, tend to be further apart under* (Euclidean) *distance than pairs of simple objects.*” [1]

Surprisingly, the top-motif does not correspond to the motion artifacts, but to simple regions of “drift”



A snippet of ECG time series in which two motion artifacts were deliberately introduced by the attending physician.

[1] Batista et al. “*CID: an efficient complexity-invariant distance for time series*.” Data Mining and Knowledge Discovery, 2014

Motivating Example 3:
Actionability Bias

In many cases a domain expert wants to find not simply the best motif, but regularities in the data which are *exploitable or actionable* in some domain specific ways.

"I want to find motifs in this web-click data, preferably occurring on or close to the weekend."

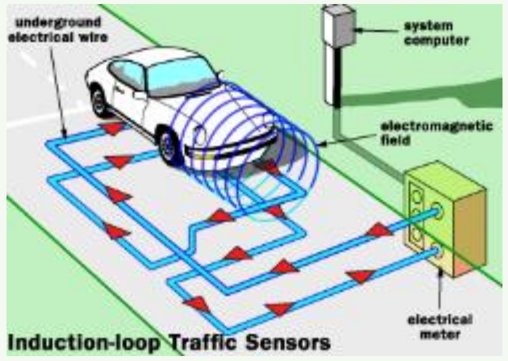
"I want to find motifs in this oil pressure data, but they would be more useful if they end with a rising trend."

Such queries can be almost seen as a hybrid between motif search and similarity search.

Key Insight/Claim

- We could try to have different definitions of motifs, for different domains/people/preferences/situations.
- However, we might have to devise a new search algorithm for each one, and maybe some such algorithms could be hard to speed up.
- That would mean having to abandon our nice, fast, one-size-fits-all matrix profile.
- Instead, we can do the following:
    - Use our one-size-fits-all matrix profile algorithm to find the basic matrix profile
    - Then use a domain dependent function, the Annotation Vector, to "nudge" the matrix profile to better suit the individual desired domains/people/preferences/situations

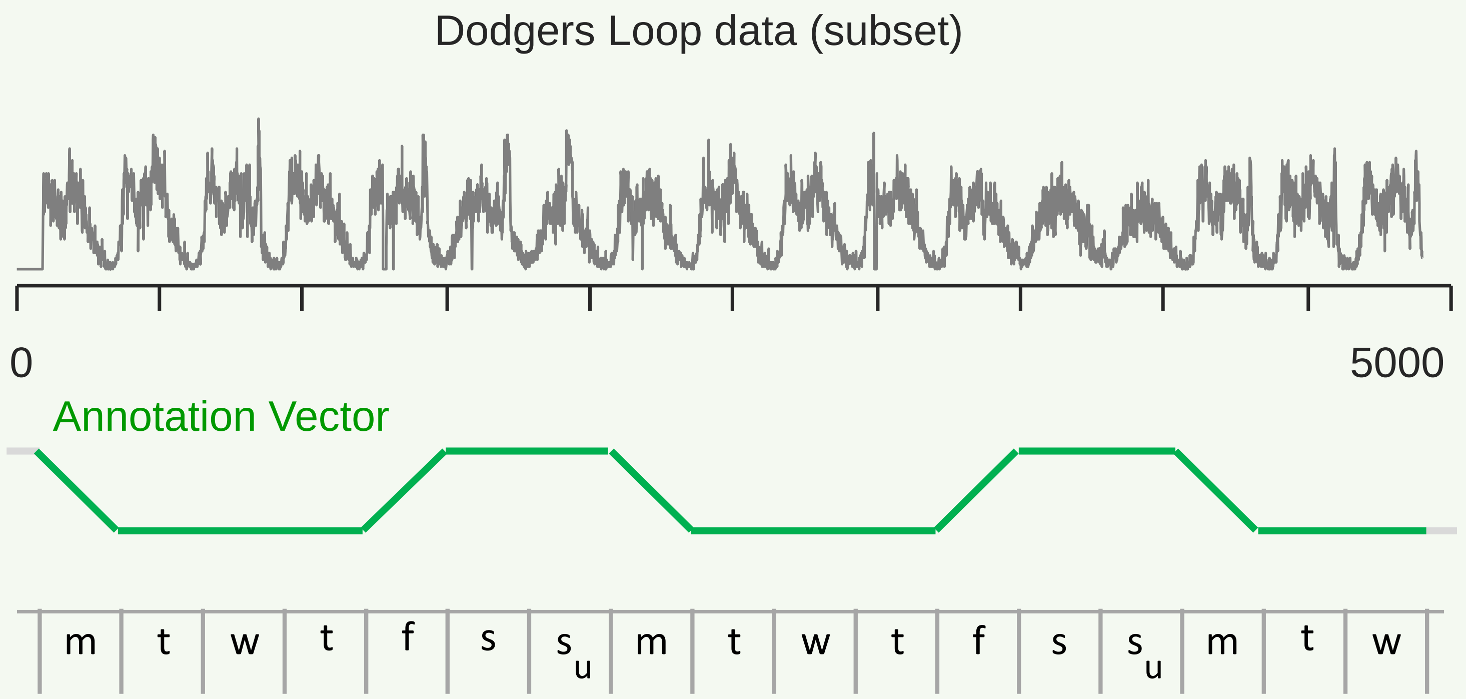
The annotation vector framework



The annotation vector (AV) is a time series consisting of real-valued numbers between [0 - 1].

A lower value indicates the subsequence starting at that index is less desirable, and therefore should be biased *against*.

Conversely, higher values mean the corresponding subsequences should be favored *for* the potential motif pool.



top) Seventeen days from the Dodger Loop dataset.

bottom) The AV that encodes a preference for motifs occurring on or near the weekend.

The annotation vector framework

Combines the matrix profile (MP) with the annotation vector (AV) to produce a new “adjusted” matrix profile.

We refer to this as the “Corrected” MP (CMP), as it correctly incorporates the contextual bias for the problem.

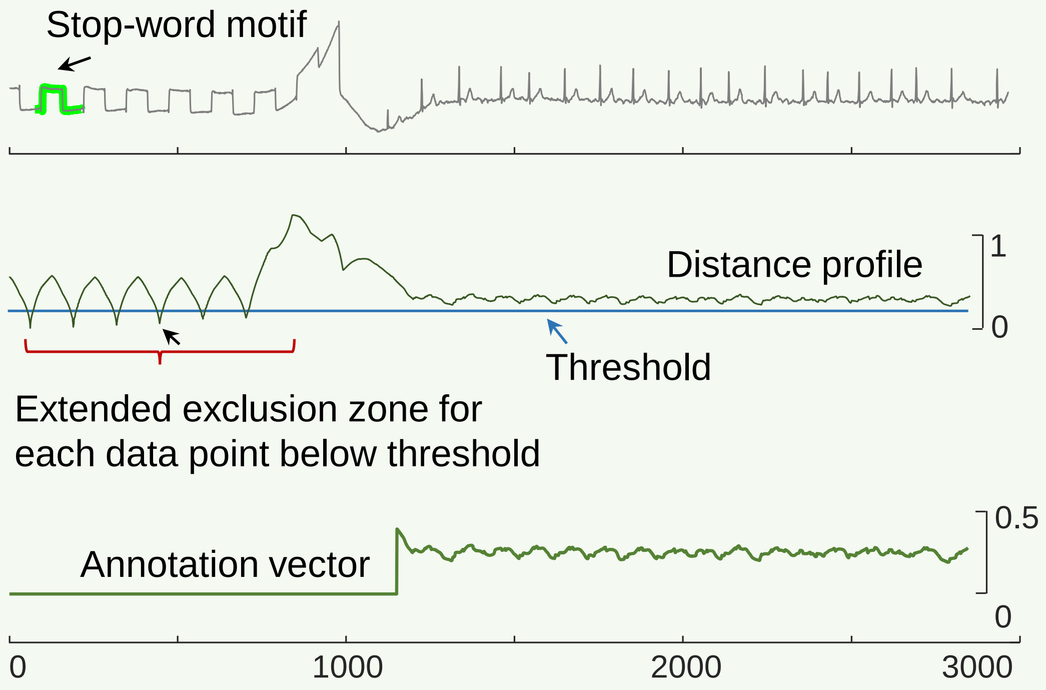
$$CMP_i = MP_i + (1 - AV_i) * \max(MP)$$

If $AV_i = 0 \rightarrow CMP_i = MP_i + \max(MP)$: raises MP value in order to remove the subsequence from potential motif pool
If $AV_i = 1 \rightarrow CMP_i = MP_i$: retains original MP value to allow the motifs that best balance the fidelity of conservation with the user’s constraints to rise to the top

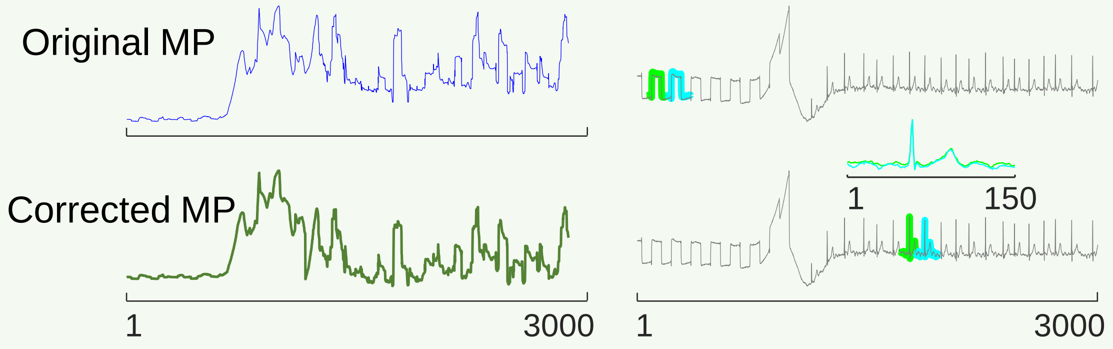
This only leaves the question of how do we create such an AV for our domain of interest?

Key Claim: For most problems, a domain expert can design an appropriate AV with 5 minutes of introspection, and implement it in 2 or 3 lines of code or an excel script.

Case study: Stop-word motif bias



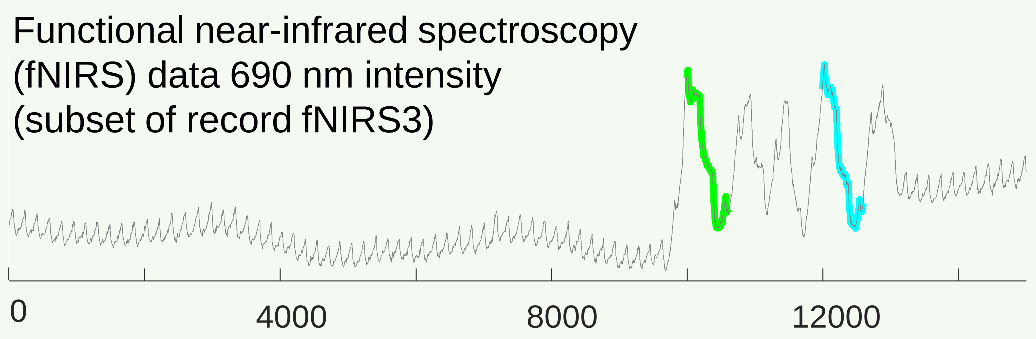
top) We annotated a single stop-word from the LTAF-71 dataset. *middle*) The stop-word distance profile to the entire dataset was thresholded to create an exclusion zone, which was used to create an AV (*bottom*).



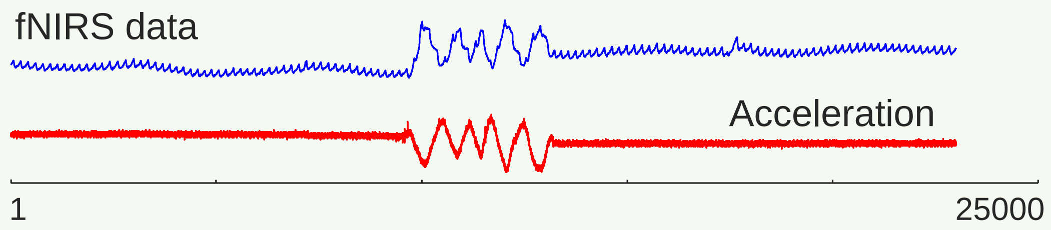
By correcting the MP to bias away from stop-word motifs, we can discover medically meaningful motifs.

Case study: Actionability bias (i)

Suppressing motion artifacts



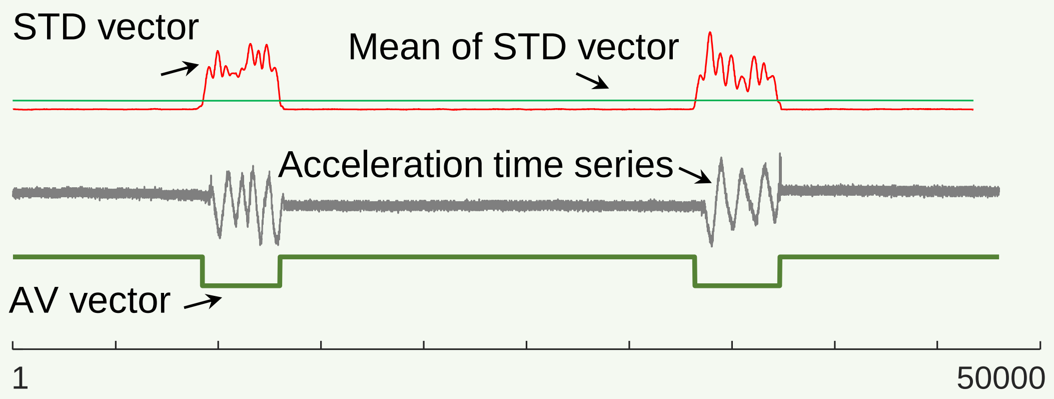
A snippet of fNIRS searched for motifs of length 600. The motifs correspond to an atypical region, which (using external data, see Fig. 7 below) we know is due to a sensor artifact.



The synchronization between the fNIRS data and accompanying accelerator data.

How to make the AV

- Slides a window of length m across the acceleration time series.
- Compares the STD of each subsequence with the mean of all the subsequences' STDs, and assign the corresponding AV value to be either 0 or 1



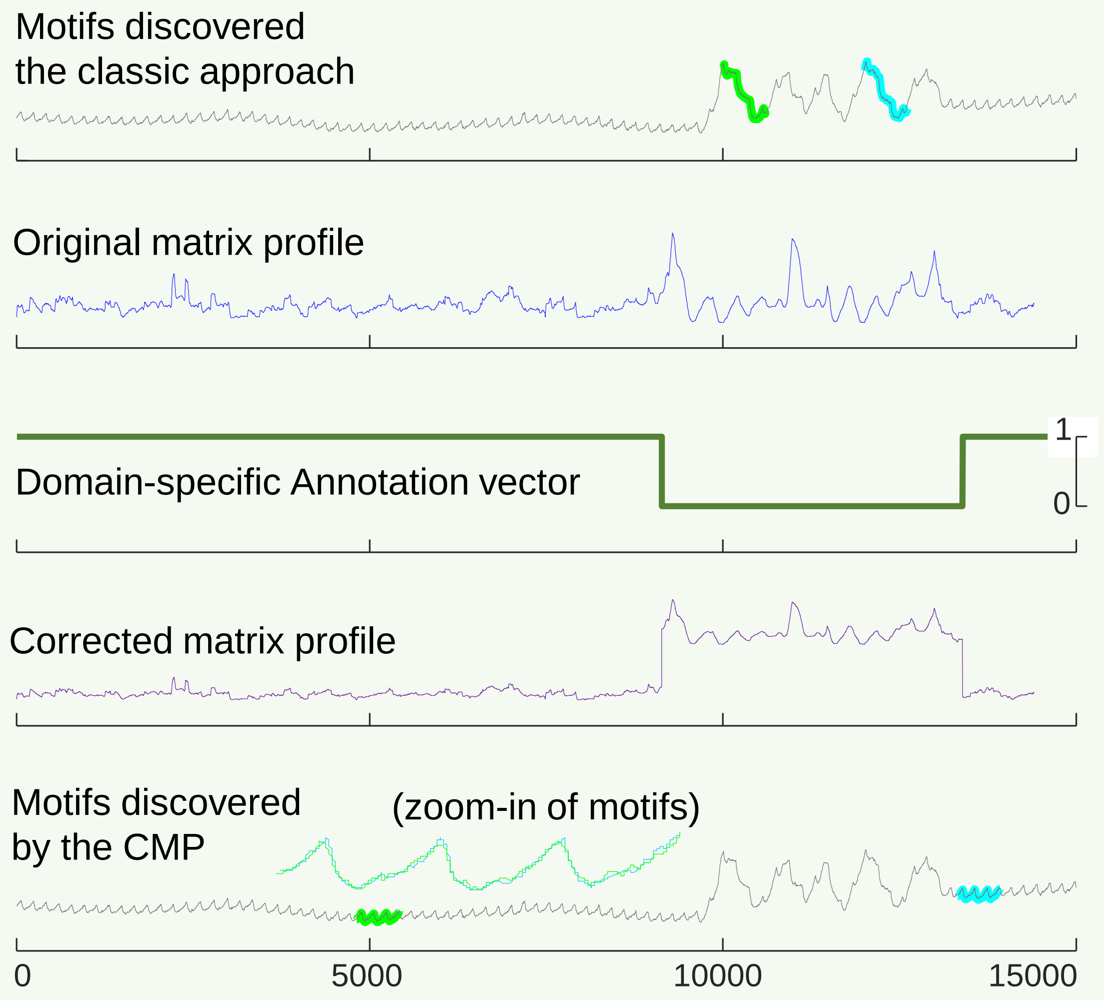
Points above the mean of all subsequences' standard deviation are well aligned with regions of motion artifacts. The corresponding AV values for these points are 0 and 1 for the rest.

Case study: Actionability bias (ii)

Suppressing motion artifacts

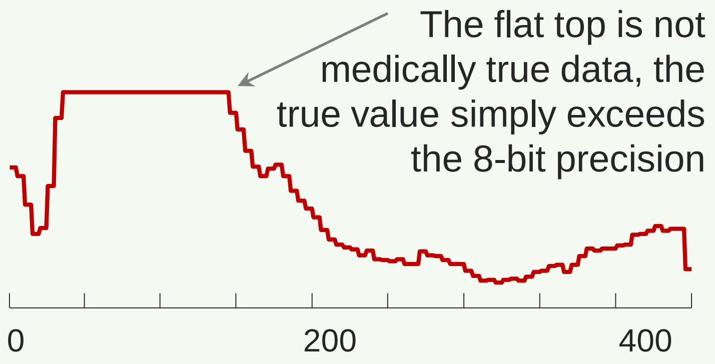
(*top to bottom*) Motifs in fNIRS data discovered using classic motif search tend to be spurious motion artifacts, because the matrix profile is minimized by the highly conserved but specious patterns.

If we use an AV to correct the MP, then that CMP allows us to find medically significant motifs.

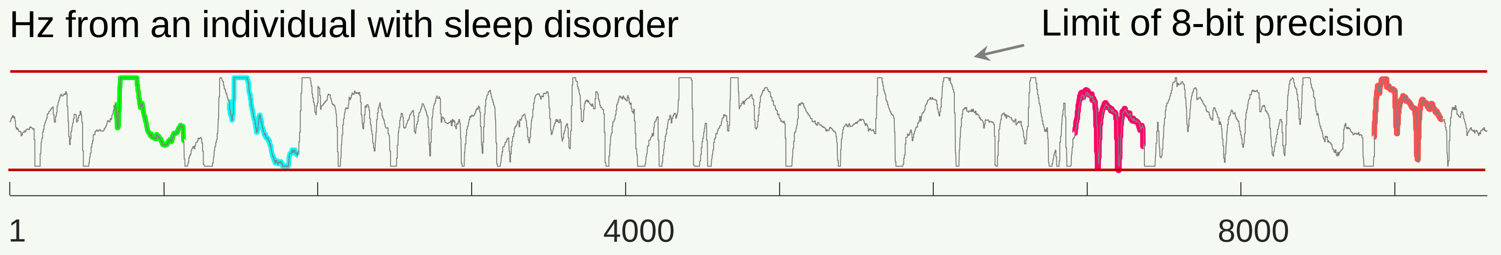


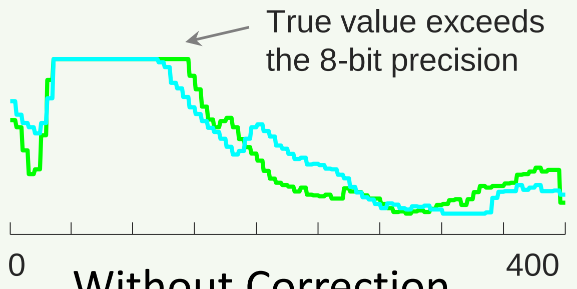
Case Study: Actionability Bias

Suppressing hard-limited artifacts

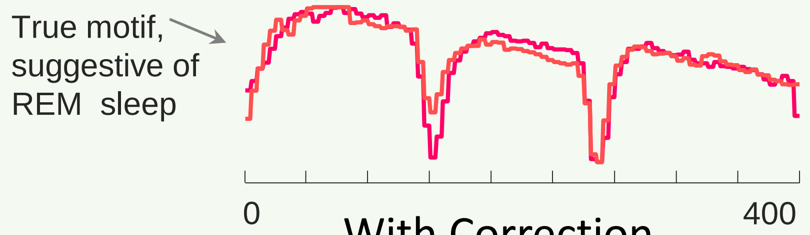


A snippet of a left-eye EOG sampled at 50 Hz from an individual with sleep disorder





Without Correction



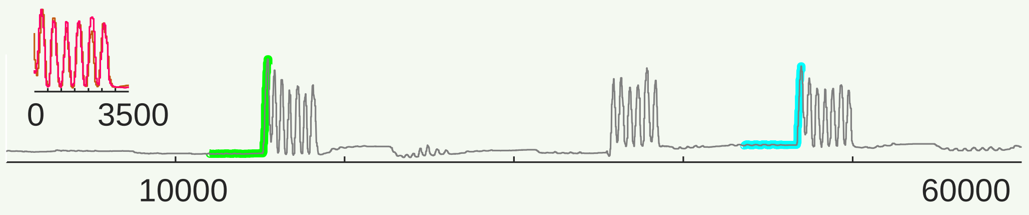
With Correction

How to make the AV

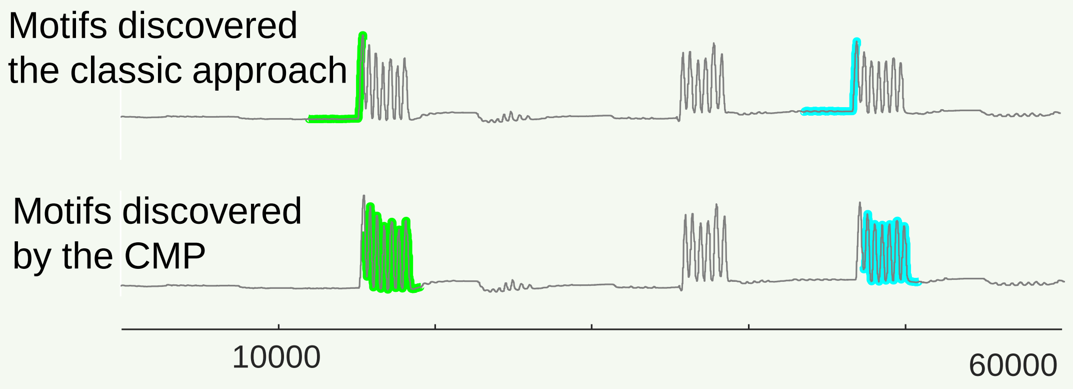
- record the maximum and minimum values of the time series (the constant values touching the red bars)
- slide a window across the time series to extract subsequences
- count the number of constant values (from being hard-limited above or below) in each subsequence
- This number over the subsequence length is used as the bias function.
- This take 3 minutes, and five lines of MATLAB.

Motifs that are discovered using classic motif search tend to include hard-limited data (*left*), because the matrix profile is minimized by having long constant regions. By creating an AV to correct the MP, we can find true motifs corresponding to ponto-geniculo-occipital waves (*right*)

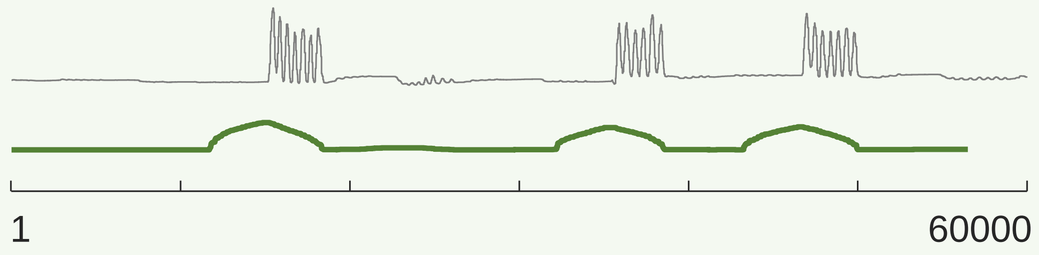
Case study: Simplicity bias



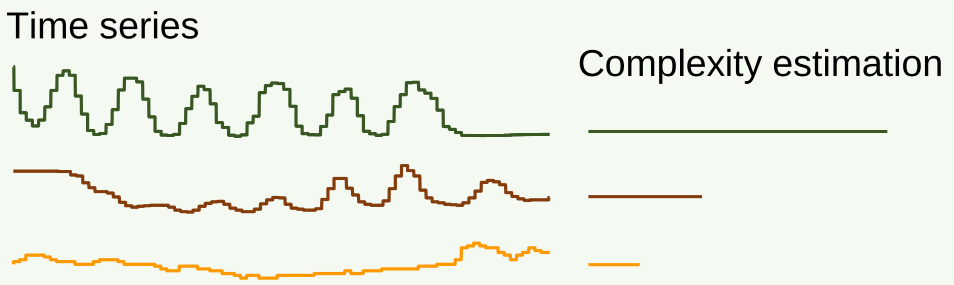
A short snippet of a time series of the flexion of a subject's little finger. Subjectively, most people would expect that two occurrences of consecutive multiple flexions to be the top motif (inset). Instead we find the simple "ramp-up" pattern.



By correcting the matrix profile with an AV based on complexity measure, we discover the true motifs of finger flexion pattern.



The complexity measure shown in parallel to the raw data. We simply normalize this complexity vector to be in range [0 - 1] to obtain the final AV.



A visual intuition of the complexity estimation of three time series subsequences of different complexity levels.

Summary of the last ten minutes: annotation vector

Most of the time, the plain vanilla MP is going to be all you need to find motifs/discords/chains etc. for you data.

In some cases, you may get spurious results. That is to say, mathematically correct results, but not what you want/need/expect for your domain.

In those cases, you can just invent a simple function to suppress the spurious motifs, code it up as an annotation vector in a handful of lines of code, use it to "correct" the MP, and then run the motifs/discords/chains algorithms as before.

Once you invent an AV, say AV_{diesel_engine} or $AV_{Turkish\_folk\_music}$, you can reuse it on similar datasets, share it with a friend, publish it etc.

The "*Matrix Profile and ten lines of code is all you need*" philosophy

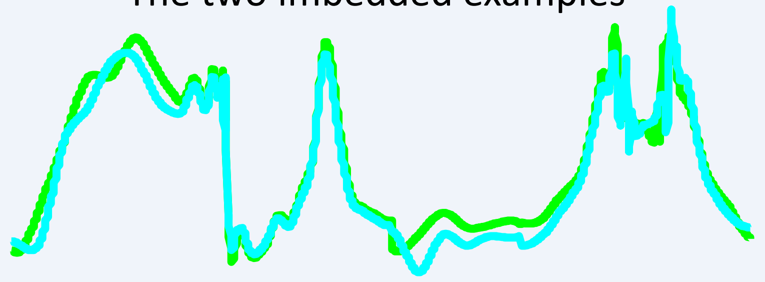
Key Idea:

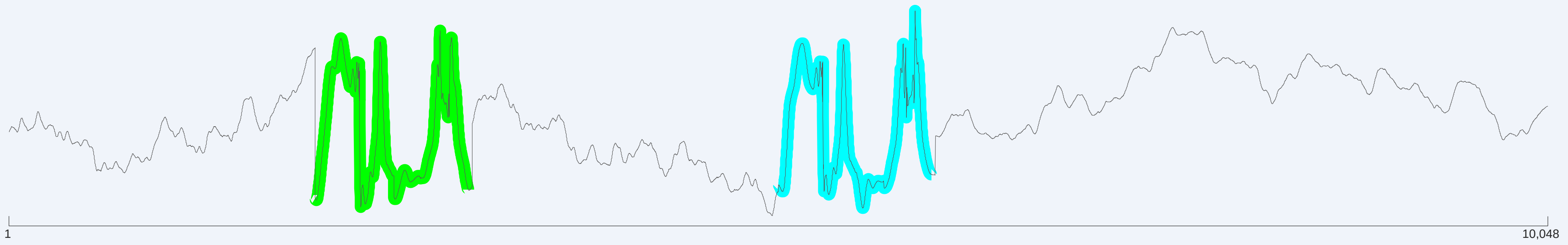
- We should think of the Matrix Profile as a black box, a *primitive*.
- As we will later see, in most cases we can think of it as being obtained essentially *for free*.
- We claim that given this primitive, and *at most* ten lines of additional code, we can reproduce the results of hundred of papers.
- This suggests that other people, may be able to take this primitive, add ten lines of code, and do amazing things that have not occurred to us. We look forward to seeing what you come up with!
- In the meantime, lets see an example of: *with ten lines of additional code, we can reproduce the results of a published paper*….

Motifs under Uniform Scaling

We took two exemplars from the same class from the MALLET dataset, and imbedded them into a random walk dataset. Even without the color-coded clue brushed onto the data by the Matrix Profile discovery tool, the repeated pattern is visually obvious.

The two imbedded examples



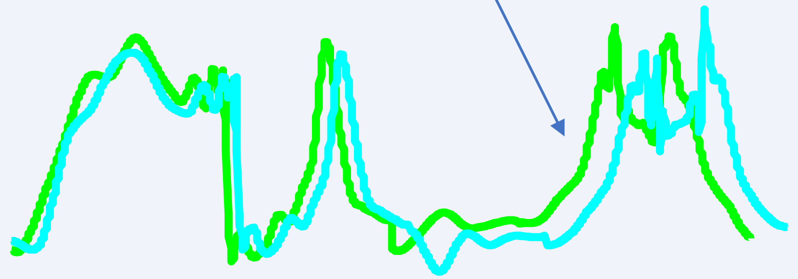


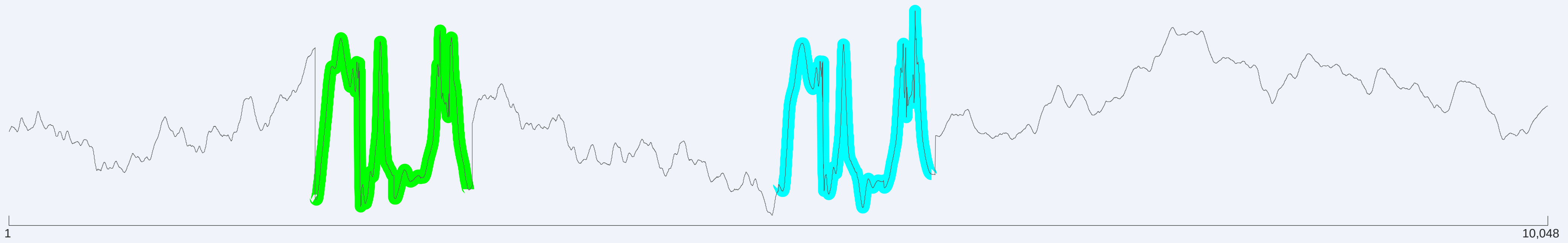
Suggestion: Toggle back and forth with last slide

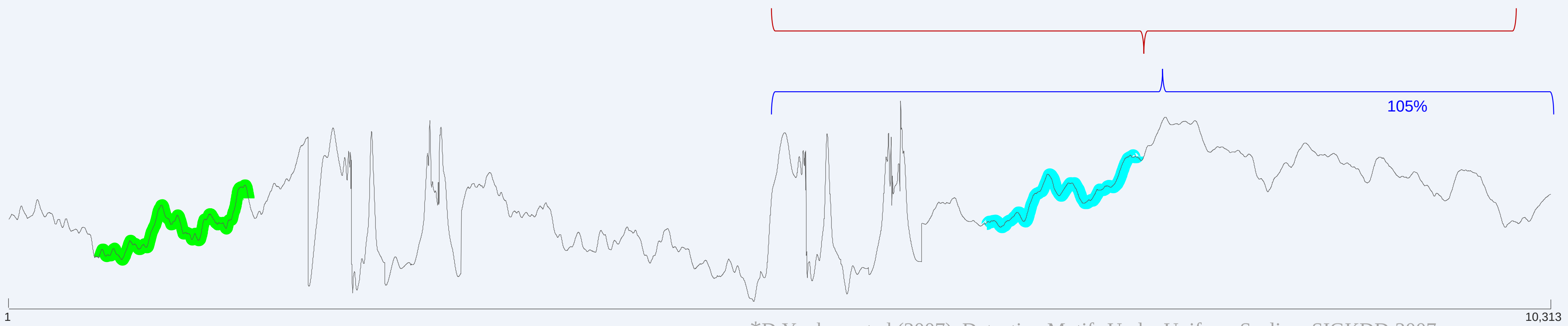
We stretched the left half of the time series by just 5%, and now the pair of imbedded patterns are no longer the top-1 motif, an unexpected and disquieting result.

There is *one* paper* that offers a solution, but it is approximate, complicated, has lots of parameters, is slow..

5% stretching means the shapes begin to go out of phase, accumulating more and more error…







*D.Yankov, et al (2007). Detecting Motifs Under Uniform Scaling. SIGKDD 2007.

This issue is easy to fix with our "*Matrix Profile and ten lines of code is all you need*" philosophy.

For example. Suppose you suspect that there are motifs in your dataset, that differ in length by 164%

Take the original dataset T, and copy a *stretched version* of it into T2, simply by using:

```
    T2 = T(1: 100/164: end); % Unofficial matlab way to resample
```

Now call:

```
    [JMP, JMPindex] = computeMatrixProfileJoin(T,T2,500);
```

The resulting Matrix Profile will discover the motifs with the appropriate uniform scaling invariance.

This issue is easy to fix with our "*Matrix Profile and ten lines of code is all you need*" philosophy.

For example. Suppose you suspect that there are motifs in your dataset, that differ in length by 164%

Take the original dataset T, and copy a *stretched version* of it into T2, simply by using:

```
    T2 = T(1: 100/164: end); % Unofficial matlab way to resample
```

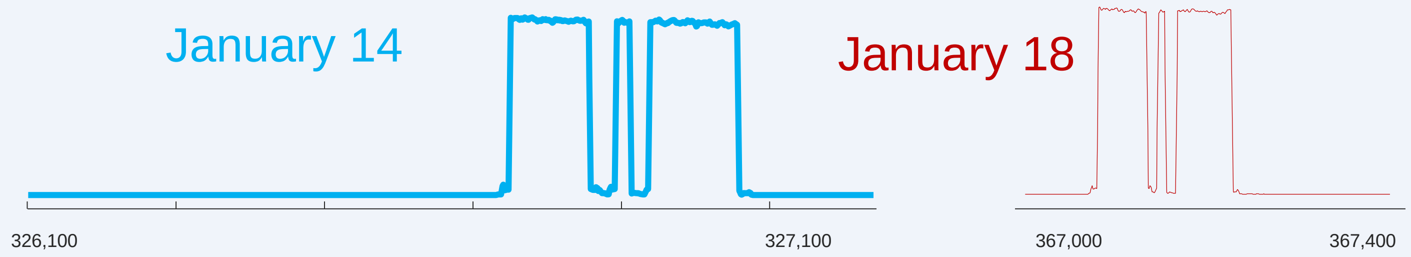
Now call:

```
    [JMP, JMPindex] = computeMatrixProfileJoin(T,T2,500);
```

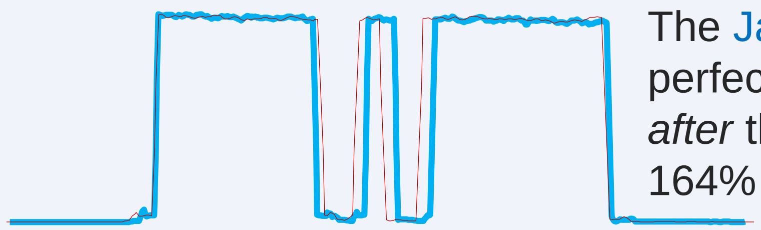
The resulting Matrix Profile will discover the motifs with the appropriate uniform scaling invariance.
We did this for the electric power demand example below...

What if you don't know the scaling factor?
It is trivial to search over all possibilities.

```
for scale_factor = 101 : 170
  T2 = T(1: 100/scale_factor: end);
  [JMP, JMPindex] = computeMatrixProfileJoin(T,T2,500);
  <trivial code to record best motifs omitted>
end
```



So, with the *Matrix Profile and ten lines of code,* we can reproduce the contribution of a SIGKDD published research effort.



The January 14th pattern is a near perfect match the January 18th pattern, *after* the latter is uniformly stretched to 164% of its original length.

The Great Divorce

- In the last hour or so we have discussed the many wonderful things you can do with the Matrix Profile, *without* explaining how to compute it!
- This divorce is *deliberate*, and very useful.
- We can completely separate the fun, interesting part of time series data mining, from the more challenging backend part.
- The divorce lets us use appropriate computational resources. For example, exploring a million datapoints in real-time with approximate *anytime* matrix profiles on a desktop, but then outsourcing to a GPU or the cloud, when we need exact answers, or we need to explore a billion datapoints.
- All will be revealed, after the coffee break…

KDD2017

Coffee Break

Dear Conference Attendee

- At this point, please switch to Mueen’s Slides
- There are some slides below, they are mostly back-up and bonus slides
- See you soon!

The End!

KDD2017

Questions?

Visit the Matrix Profile Page

www.cs.ucr.edu/~eamonn/MatrixProfile.html

Visit the MASS Page

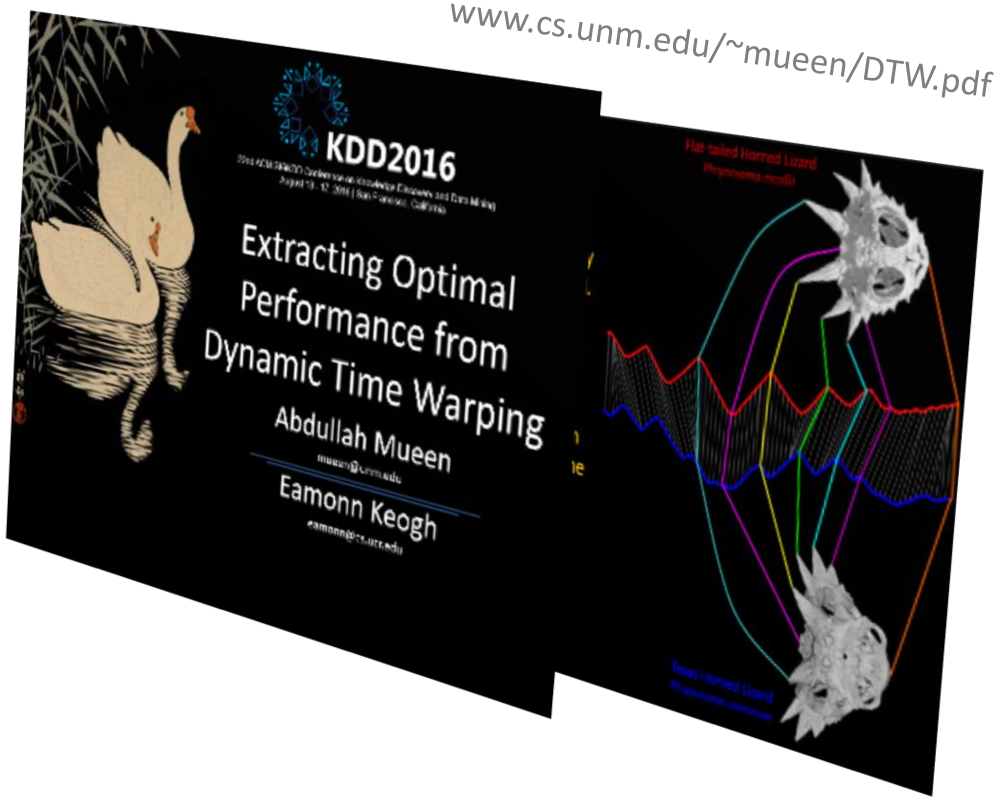
www.cs.unm.edu/~mueen/FastestSimilaritySearch.html

Please fill out an evaluation form, available in the back of the room.

Below are Bonus Slides/Back up Slides

Why does the MP use Euclidean Distance instead of DTW?

www.cs.unm.edu/~mueen/DTW.pdf


KDD2016

Extracting Optimal Performance from Dynamic Time Warping

Abdullah Mueen

Eamonn Keogh

1. Pragmatically, We have not yet figured out how to do DTW with the MP efficiently.

2. Having said that, it may not be that useful. DTW *is* very useful for..

   A. …*One-to-All* matching (i.e. similarity search). But the MP is *All-to-All* matching*.

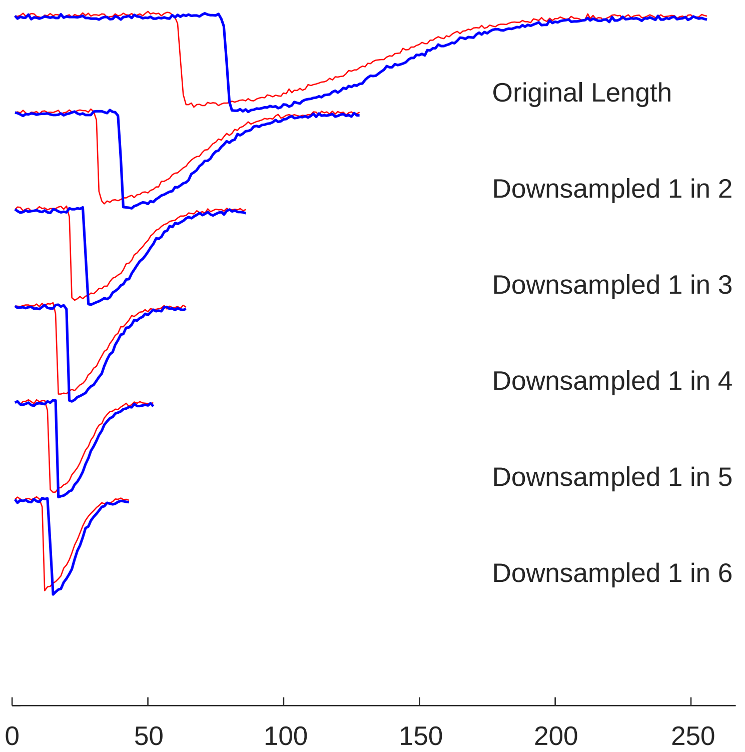
   B. …*small* datasets: For example building a ECG classifier with just five representative heartbeats. But here we are interested in *large* datasets.

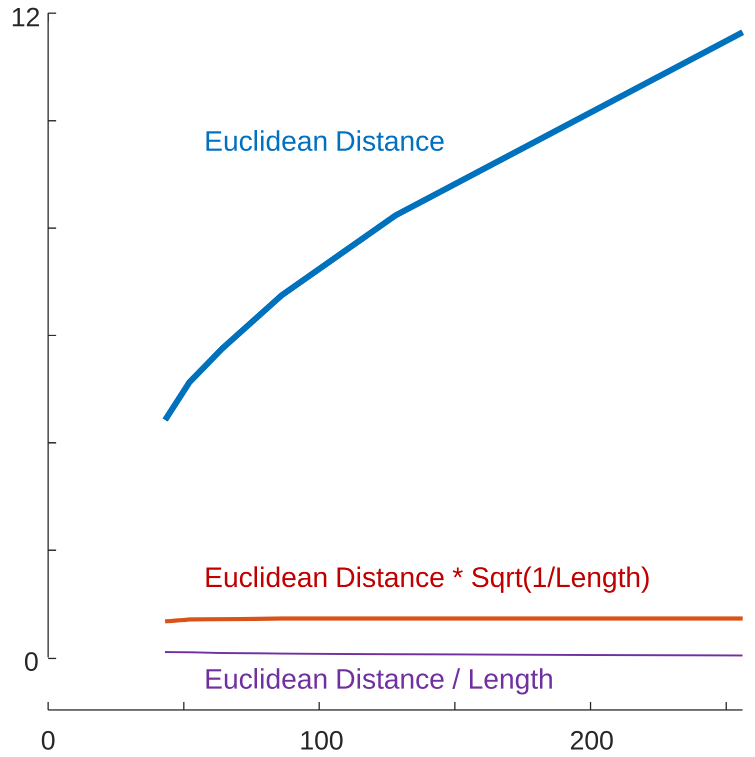
* The difference this makes is a real-valued version of the birthday paradox. See Mueen’s thesis

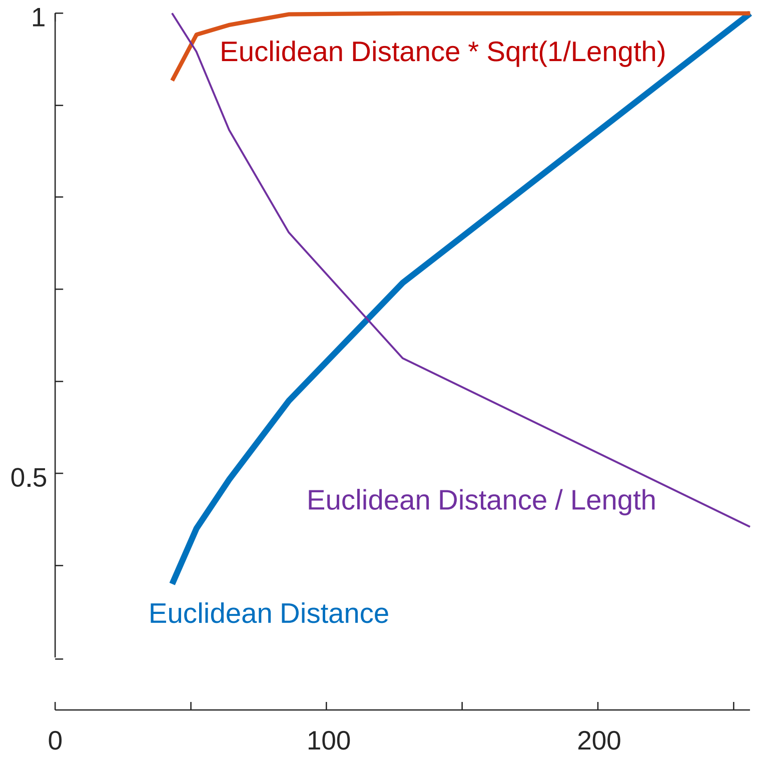
Comparing Motifs of Different Lengths

If we *find* motifs of different lengths, we need to be able to *rank* motifs of different lengths. A similar problem occurs in string processing, and the common solution is to replace the edit-distance by the length-normalized edit-distance, which is simply the classic distance measure divided by the length of the strings in question [a]. This correction would find the pair {concatenation, concameration} *more* similar than {cat, cot}, matching our intuitions. Researchers have suggested this length-normalized correction for time series, but as we will show, is the wrong correction factor.

To see this, consider the following thought experiment. Imagine some process in the system we are monitoring occasionally "injects" a pattern into the time series. As a concrete example, washing machines typically have a prototypic signature, but the signatures express themselves more slowly on a cold day, when it takes longer to heat the cooler water supplied from the city [b]. We would like all equal length instances of the signature to have approximately the same distance. In Figure 1.*left* we show two examples from the TRACE dataset which will act as proxies for a variable length signature. We produced the variable lengths by down sampling. In Figure 1.*center* we show the distances between the patterns as their length changes. With no correction, the Euclidean distance is obviously biased to the shortest length. The length-normalized Euclidean distance looks "flatter" and suggests itself as the proper correction. However, this is something of an optical illusion due to its smaller scale. In Figure 1.*right* we show all measures after dividing them by their largest value. We can now see that the length-normalized Euclidean distance has a strong bias to the shortest pattern. In contrast to the other two approaches, the "sqrt(1/length)" correction factor provides a near perfectly invariant distance over a huge range of values.





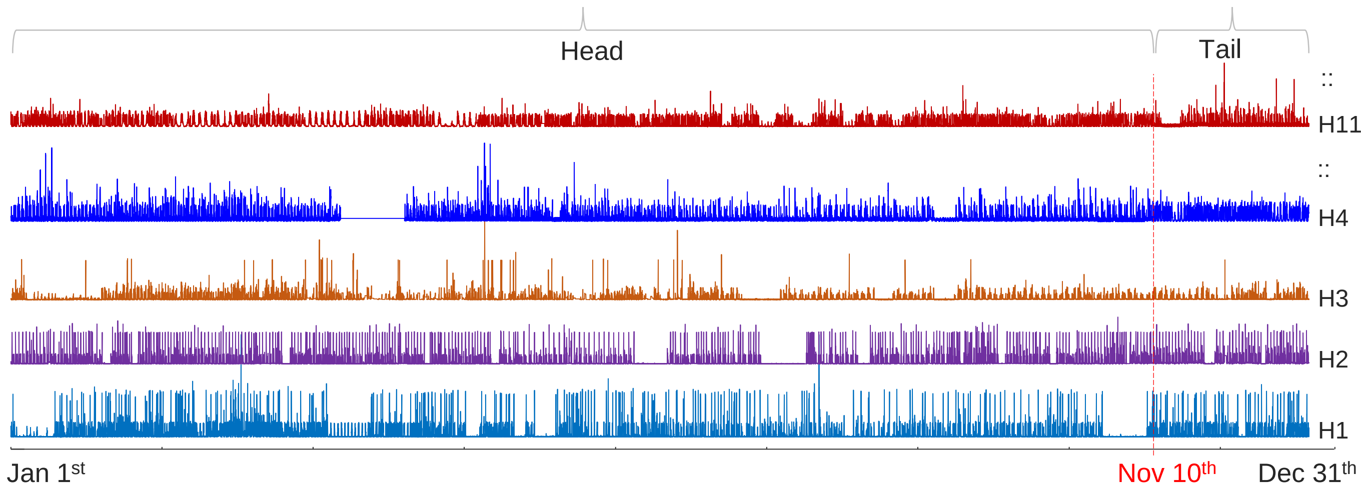


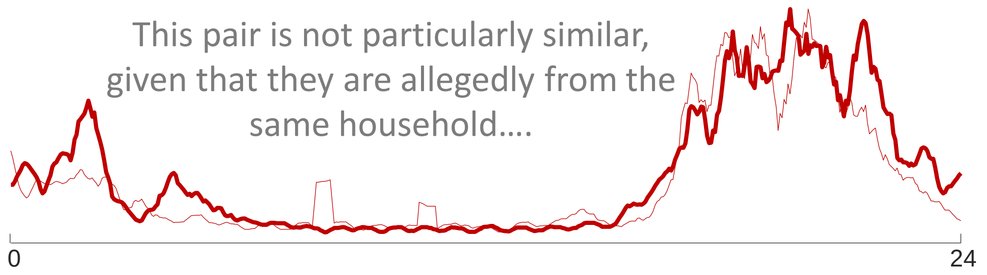
Applications of the MP: Meter-Swapping

Electricity theft is multi-billion-dollar problem worldwide. There are dozens of ways to steal power, but some modern wireless meters offer a surprisingly easy method with little chance of detection. Suppose customer **A** is a heavy consumer of electricity; perhaps he has several electric cars, or a machine shop (or marijuana nursery) in his garage. Further suppose that he notes that one of his neighbors, customer **B**, an elderly widow living alone, consumes very little power. It is possible for **A** to surreptitiously switch his meter with **B**, and thus only have to pay for her meager consumption, while she unwittingly gets lumbered with paying for his extravagant consumption. This crime is called *meter-swapping*, and has become increasing prevalent as power companies have reduced meter reading staff in favor of wireless meter reading.
It might be imagined that this would be easy for the power company to detect, as there would be a significant change in the average power consumed by two houses. However, the Fig.*top* hints at, power consumption is often bursty anyway. For example, as families take vacations, welcome a new baby, or have children return from college for a few weeks.
Our intuition to solve this problem is to note that while *volume* of consumption is not a good feature, some households may have a unique "shape" of the consumption over a day. Note that we do not expect *all* days to be conserved and unique, it is sufficient for our purposes that the household *occasionally* produces a well-conserved pattern, perhaps correspond to a low-power use on the Sabbath for an orthodox family, or a once every seven-week all-day obligation to wash and dry the soccer kits for the entire team.
We consider a dataset of household electrical power demand collected from twenty houses in the UK in 2013. To simulate a meter-swapping event, we randomly choose two of these time series, and swapped their traces starting at November 10th. As we can see the Fig.*top* this change is not readily visually obvious.
To find the swapped time series pair, we propose the following simple algorithm. We divide all the time series into two sections, the "Head", prior to November 10th, and the "Tail", subsequent to November 10th. We join all possible combinations of Heads and Tails, and record the pair Hi, Hj that minimizes the following score:

Swap-Score(i,j) = min(Head$_{Hi}$ ⋈1nn Tail$_{Hj}$) / (min(Head$_{Hi}$ ⋈1nn Tail$_{Hi}$) + eps)

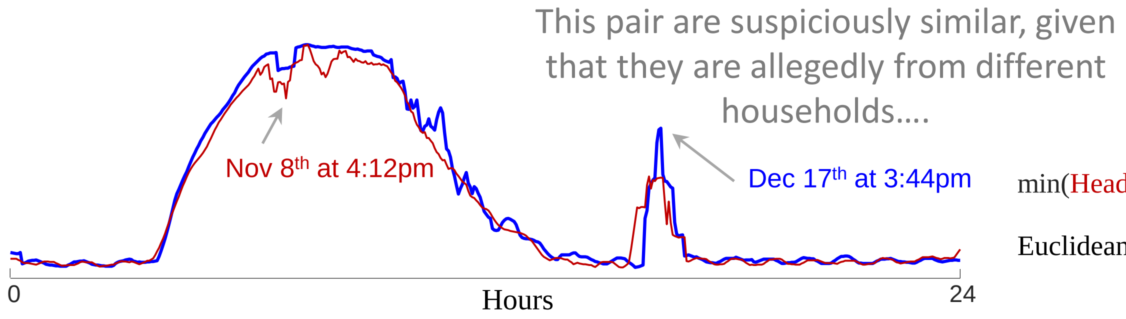
In our simple experiment, this score was minimized by i = H11 and j = H4, which, as it happens, *are* our swapped pair. As in the Fig.*bottom* shows, the motif spanning these two apparently distinct traces time series is suspiciously similar, perhaps similar enough to warrant a visit by a meter reader/fraud prevention officer.





min(Head$_{H11}$ ⋈θ1nn Tail$_{H11}$)

Euclidean Distance = 9.56



min(Head$_{H11}$ ⋈θ1nn Tail$_{H4}$)

Euclidean Distance = 2.85

Appendix A: A worked example of an AV (next 6 slides)

- Perhaps the most common problem people face with motif discovery, is finding "too-simple" motifs.
- This is not a "bug", just a property of Euclidean Distance.
- In the next six slides we will show you a concrete example of our fix.

See also: Hoang Anh Dau and Eamonn Keogh. Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. KDD'17, Halifax, Canada

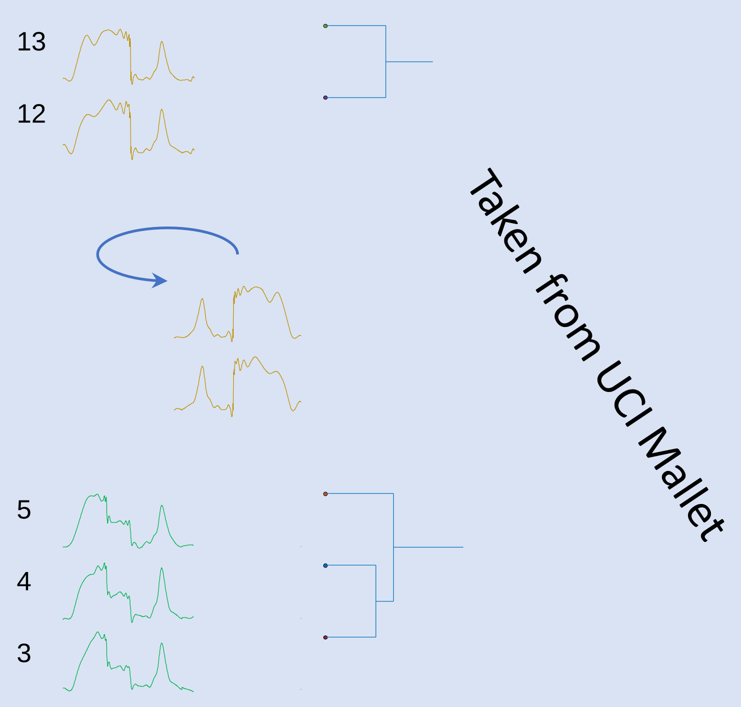
Lets start by making a test dataset

In a smoothed random walk of length 50,000, we imbedded **the reverse of** one Mallet-6 at location 10,000, and **the reverse of pattern** a different Mallet-6 at location 40,000.
We imbedded one Mallet-2 at location 15,000, and a different Mallet-2 at location 25,000, and yet another Mallet-6 at location 35,000.
Then we added noise to the entire thing:

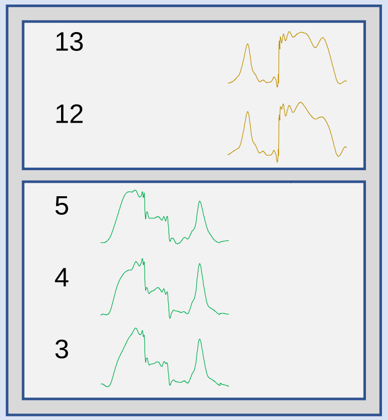
```
TAG = (TAG + randn(size(TAG))/4)
```

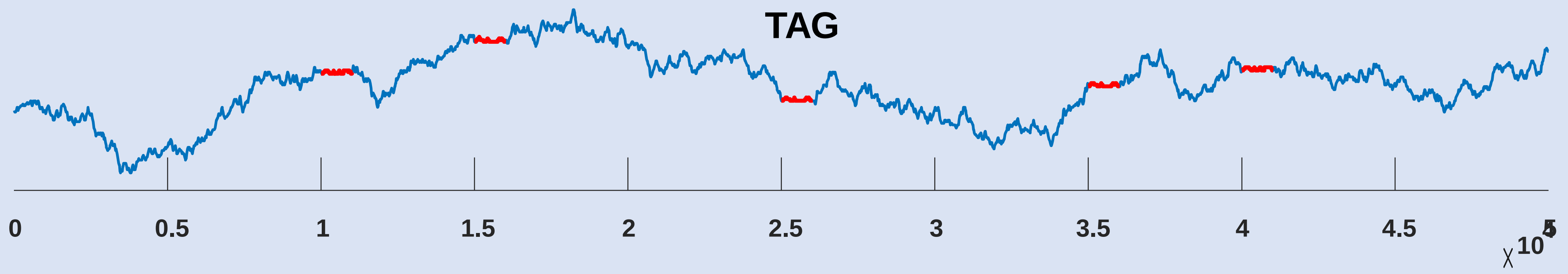
Here we would definitely expect to find the following …

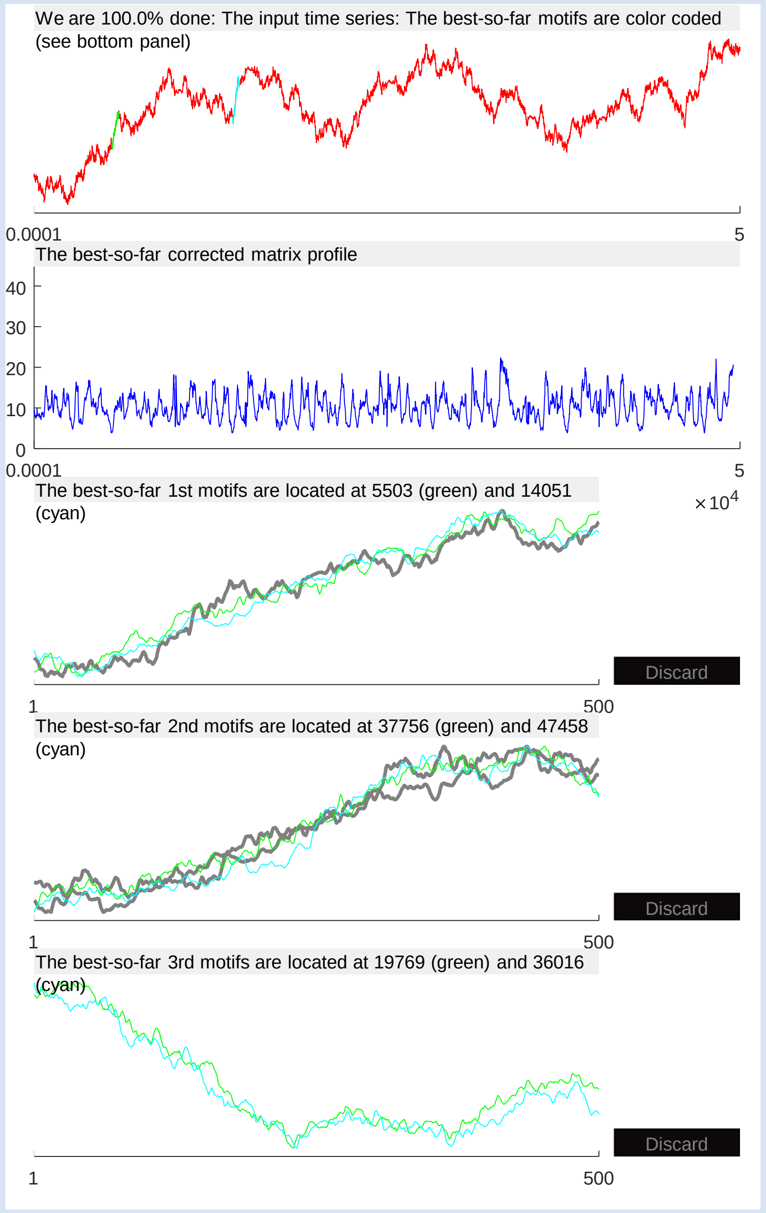
A) Two motifs, one of size 2, one of size 3.



A



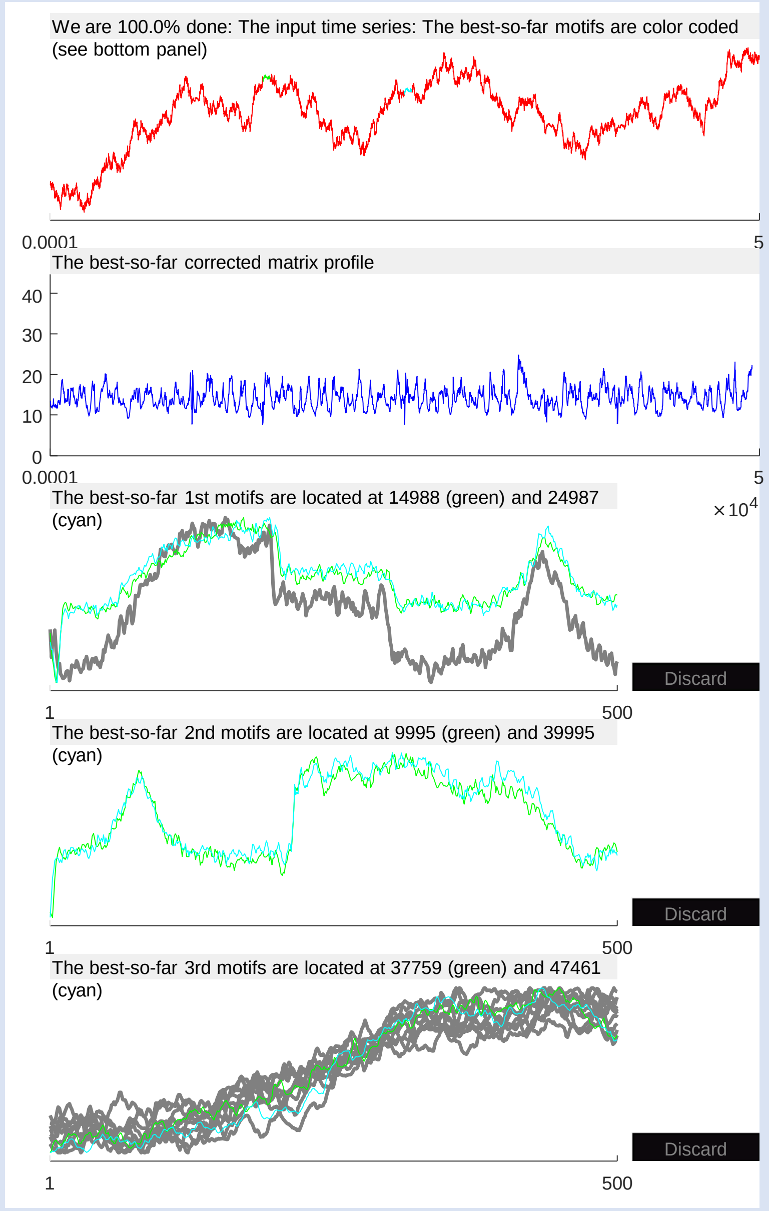




TAG: **Pure** motif search

We did *not* find the imbedded motifs ;-(

Instead, we just found these simple patterns



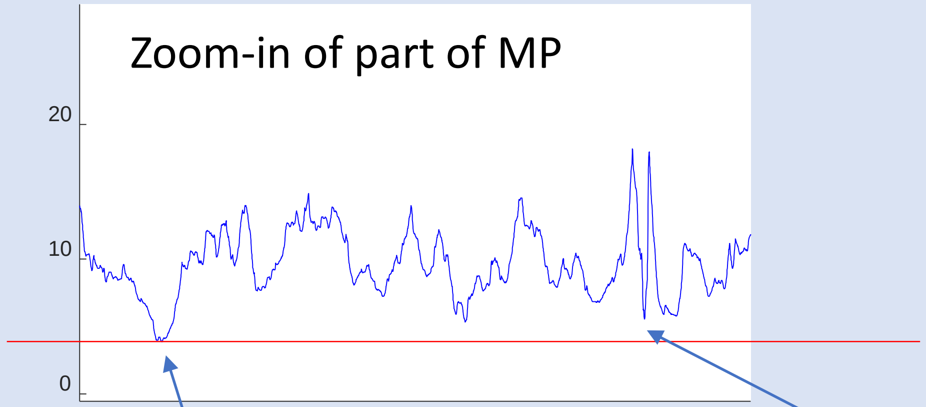
TAG: Motif search, **corrected for simplicity**

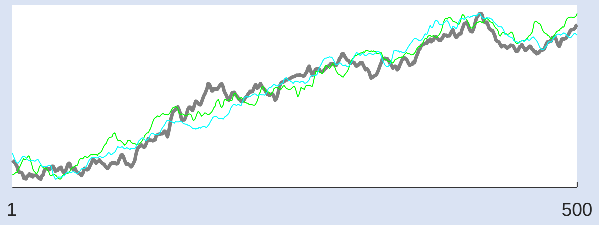
We correctly found the two imbedded motifs, one of size 3, one of size 2.

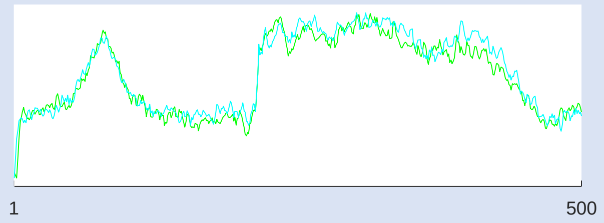
It is important to note that pure motif search "almost" works.

As you can see below, the true motif *is* low in the MP, just not *quite* low enough.

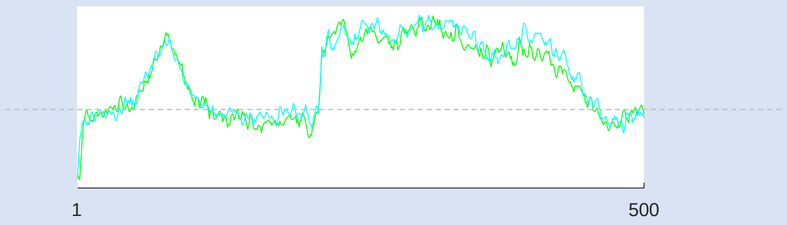
That means if we can just nudge the relevant section down a little (or equivalently, nudge everything else up), we would find the right motifs.



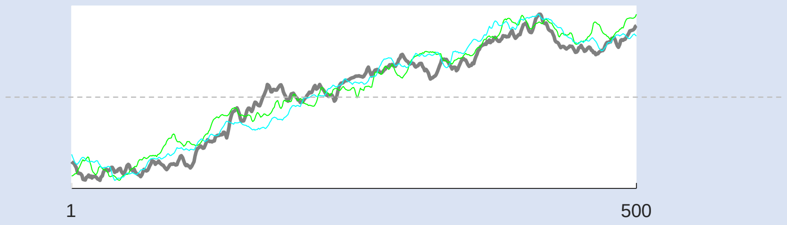




Complexity = 3.2



Complexity = 1.4



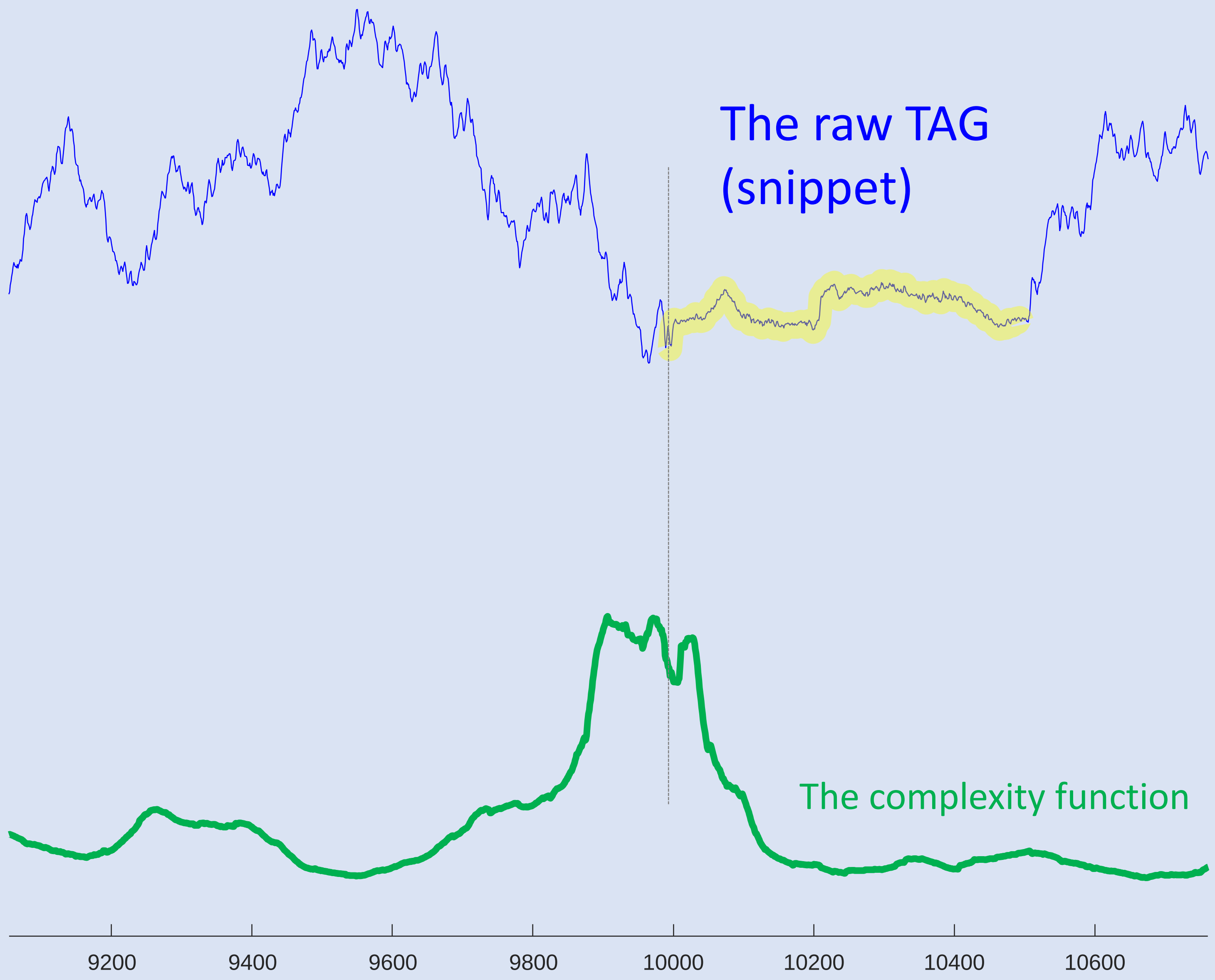
What we need is a function that recognizes that one of these patterns is too simple to be of interest.

The number of zero-crossings would probably work, but that can fail in pathological circumstances.

One such function is complexity, here it is in all its glory, for a time series subsequence x

```
function [complexity] = check_complexity(x)
    x=zscore(x);
    complexity = sqrt(sum(diff(x).^2));
end
```

The exact values of complexity (shown to the left) are not important, only the relative values will matter.



Note that the complexity function is high where the potential motif is, but low elsewhere. *Just* what we want.

We are almost done. However, there is one caveat. We have to have a parameter.

The complexity function might be too *strong*, and might push too hard for more complex motifs, even if they are not really similar.

However, we can control its strength.

The `dilution_factor` is a number greater than or equal to zero. If it is zero, there is no dilution. If it large enough, say over 40, we begin to degenerate to classic motif search.

At least for this problem, values in the range 2 to 16 work great.

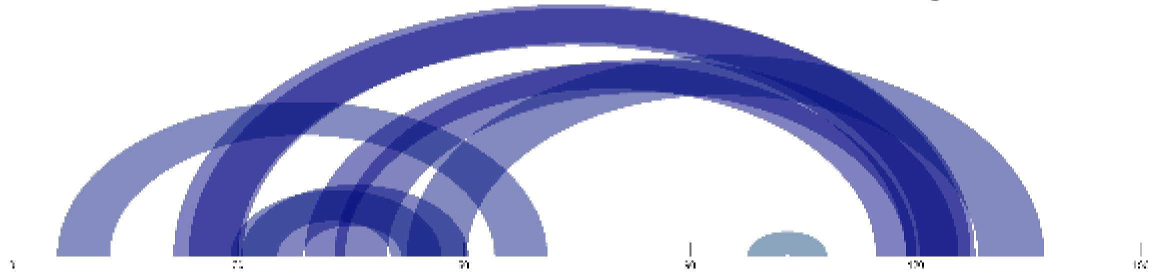
```
% Makes annotation vector that favors complexity
% Dau Hoang Anh and Eamonn Keogh
% [annotationVector] = make_AV_complexity(data, subsequenceLength);
% Output:      annotationVector: annotation vector (vector)
% Input:       data: input time series (vector)
%              subsequenceLength: motif length (scalar)
%
function [AV] = make_AV_complexity(data, subsequenceLength)
    data = zscore(data); % data is a row vector
    profile_length = length(data) - subsequenceLength + 1;
    AV = zeros(profile_length,1);
    for j = 1: profile_length
        AV(j) = check_complexity(data(j:j+subsequenceLength-1));
    end

    AV = zeroOneNorm(AV);       % zero-one normalize the AV
                                % Select dilution factor, 0 is no dilution,
    dilution_factor = 5;        % ..larger numbers are more dilution
    AV=AV+dilution_factor;
    AV=AV/(dilution_factor+1);

end
```

Music Revisited 1

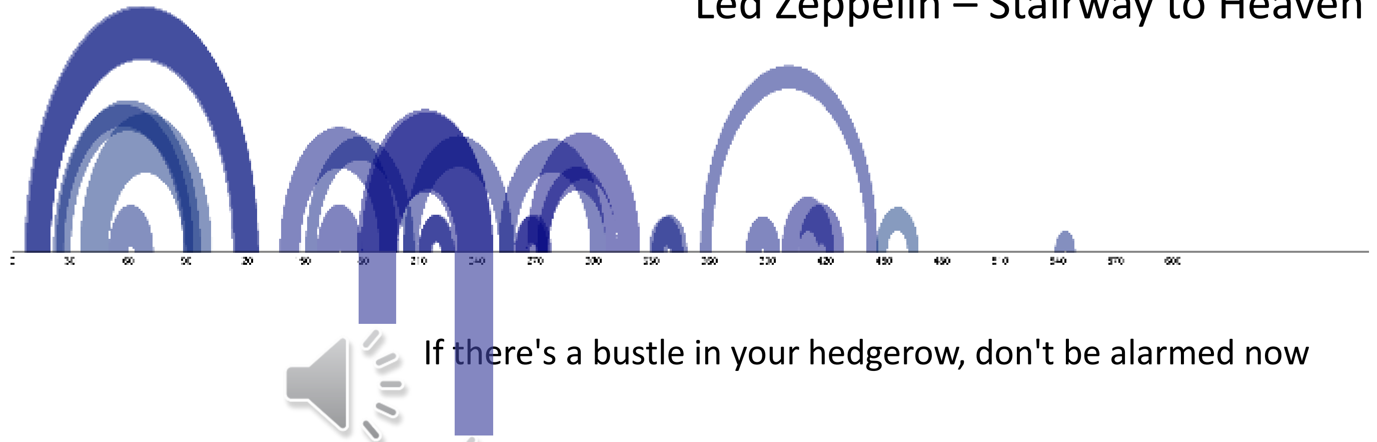
The MP is an useful tool for various music analysis tasks

The MP can be used to create arc plots, giving a good visualization of the music structure

Eagles – Hotel California



Led Zeppelin – Stairway to Heaven



If there's a bustle in your hedgerow, don't be alarmed now

Yes, there are two paths you can go by, but in the long run

These links can also be used to create an “infinite song”

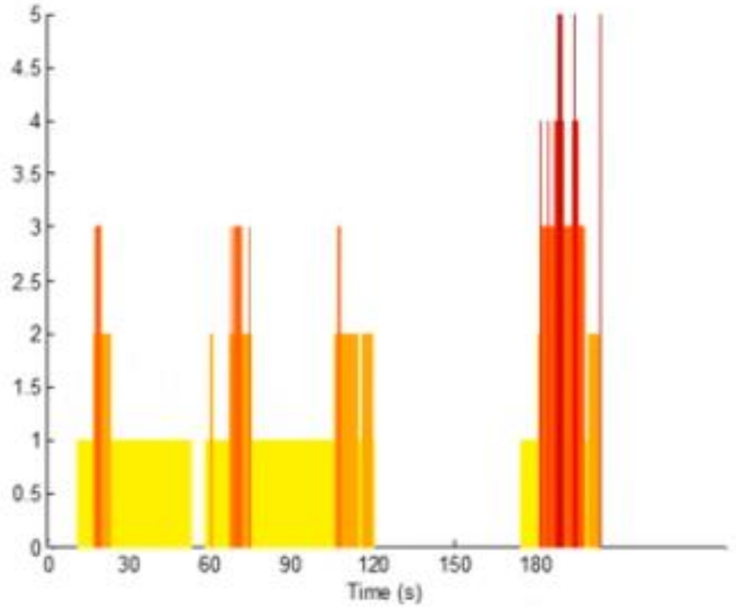
Music Revisited 2

Repeated patterns can be applied in different scenarios

The “most repeated” subsequence can be used as thumbnail

It is given by the mode of the MP-index

The plot is a histogram of the MPindex. The values record how many times a subsequence was considered NN of some other subsequence. The subsequence that maximize this plot was used as the audio thumbnail



We could have had it all
Rolling in the deep
You had my heart inside of your hand
And you played it to the beat
could have had it all

Notes on Artwork

Most are Images by Dürer (but colored by other)and other artists from

Triumph of Emperor Maximilian I, King of Hungary, Dalmatia and Croatia, Archduke of Austria

- Provenance: National Library of Spain Biblioteca Nacional de España
- Identifier: 108150 Institution: National Library of Spain   Provider: The European Library
- http://bdh-rd.bne.es/viewer.vm?id=0000012553&page=1

- The elephant 1: Elephants via some Paintings of the Mughal Era, http://ranasafvi.com/mughal-elephants/

- Elephant number 2: http://collections.vam.ac.uk/item/O15706/one-of-six-figures-from-gouache-mazhar-ali-khan/