UNIVERSITY OF CALIFORNIA, RIVERSIDE

Data Mining and Machine Learning in Time Series Databases

Dr Eamonn Keogh

Computer Science & Engineering Department
University of California - Riverside
Riverside,CA 92521
eamonn@cs.ucr.edu

Fair Use Agreement

This agreement covers the use of all slides on this CD-Rom, please read carefully.

- You may freely use these slides for teaching, if
  - You send me an email telling me the class number/ university in advance.
  - My name and email address appears on the first slide (if you are using all or most of the slides), or on each slide (if you are just taking a few slides).
- You may freely use these slides for a conference presentation, if
  - You send me an email telling me the conference name in advance.
  - My name appears on each slide you use.

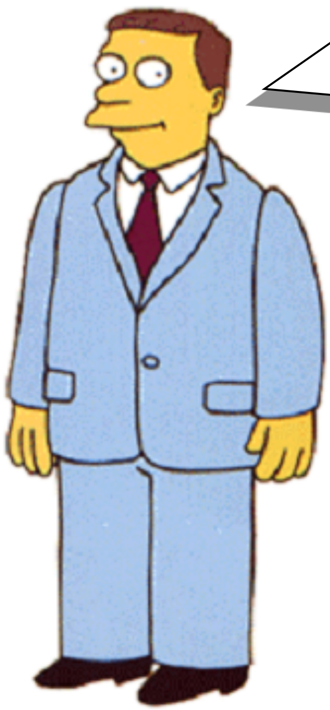
• You may not use these slides for tutorials, or in a published work (tech report/ conference paper/ thesis/ journal etc). If you wish to do this, email me first, it is highly likely I will grant you permission.

Outline of Tutorial

- Introduction, Motivation
- The Utility of Similarity Measurements
  - Properties of distance measures
  - The Euclidean distance
  - Preprocessing the data
  - Dynamic Time Warping
  - Uniform Scaling
- Indexing Time Series
  - Spatial Access Methods and the curse of dimensionality
  - The GEMINI Framework
  - Dimensionality reduction
    - Discrete Fourier Transform
    - Discrete Wavelet Transform
    - Singular Value Decomposition
    - Piecewise Linear Approximation
    - Symbolic Approximation
    - Piecewise Aggregate Approximation
    - Adaptive Piecewise Constant Approximation
  - Empirical Comparison
- Data Mining
  - Anomaly/Interestingness detection
  - Motif (repeated pattern) discovery
  - Visualization/Summarization
  - What we should be working on!

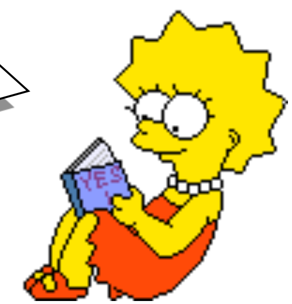
Summary, Conclusions

Disclaimers



This tutorial is presented "*math lite*". Instead we focus on communicating the *intuitions* behind the problems/ representations/algorithms!

However we have included pointers to 100's of papers and books!

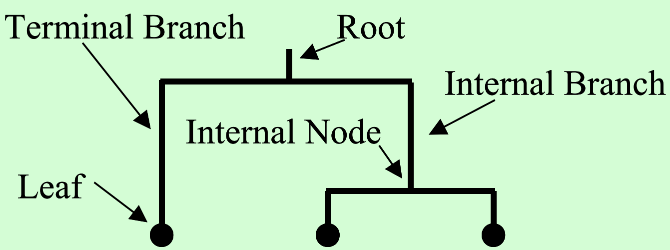


Some of the ideas presented in this tutorial are Dr. Keogh's. He will try to make his biases clear where appropriate!

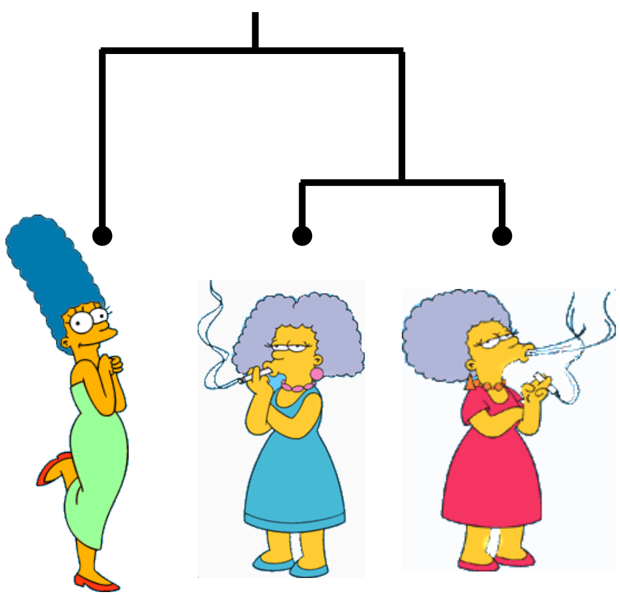
A Quick Digression…

A Useful Tool for Summarizing Similarity Measurements

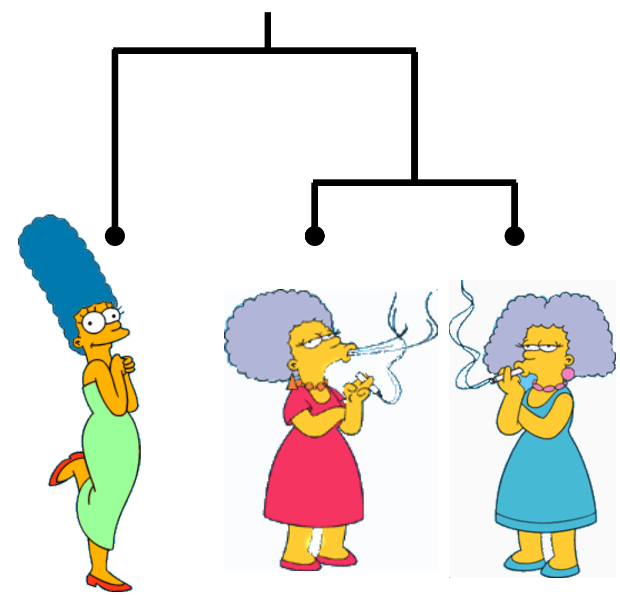
In order to better appreciate and evaluate time series similarity measures, we will quickly review the *dendrogram*.



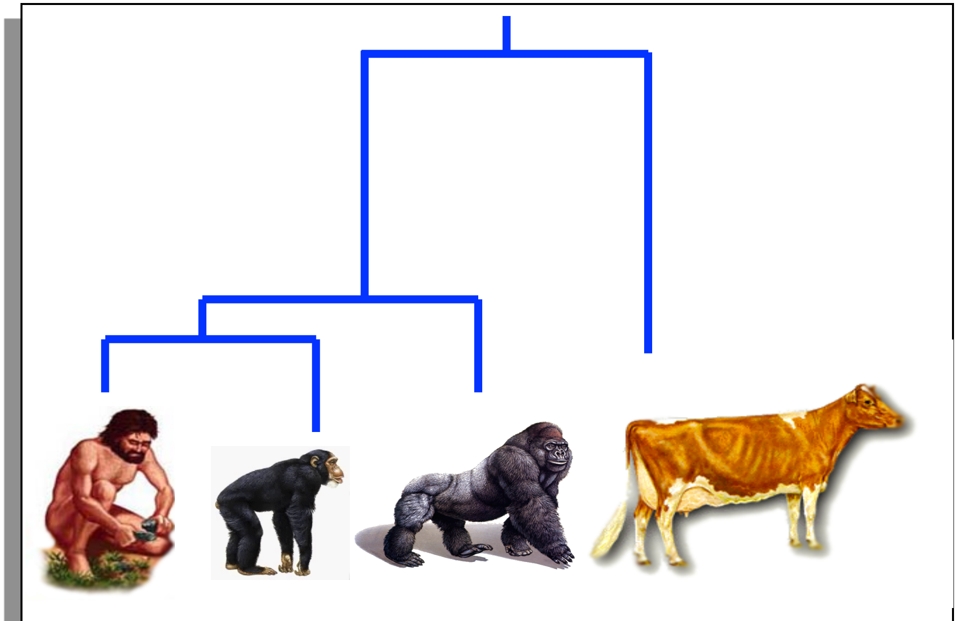
The similarity between two objects in a dendrogram is represented as the height of the lowest internal node they share



Marge Selma Patty

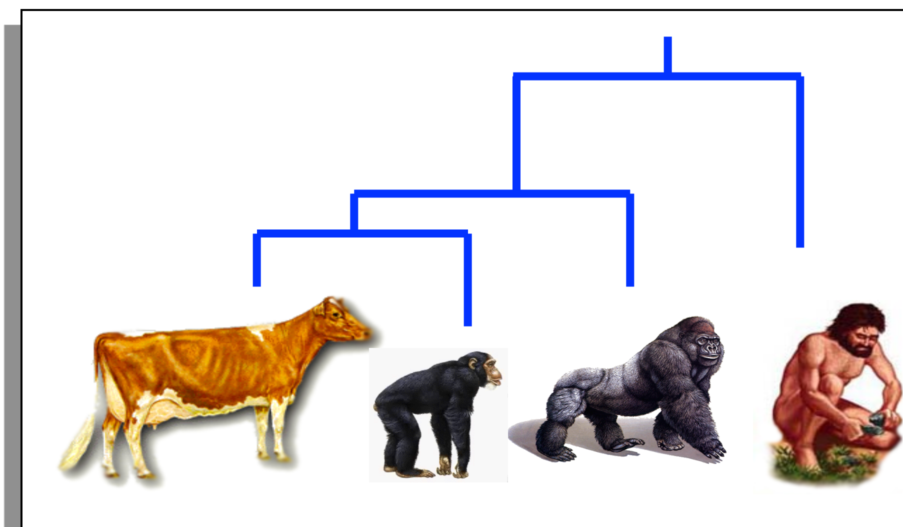


Why are Dendrograms Useful?



If someone tells us they have a new similarity measure for DNA, and it produces an *intuitive* dendrogram…

… but if their new similarity measure gives us a very *unintuitive* dendrogram, we should view it with suspicion…



What are Time Series?

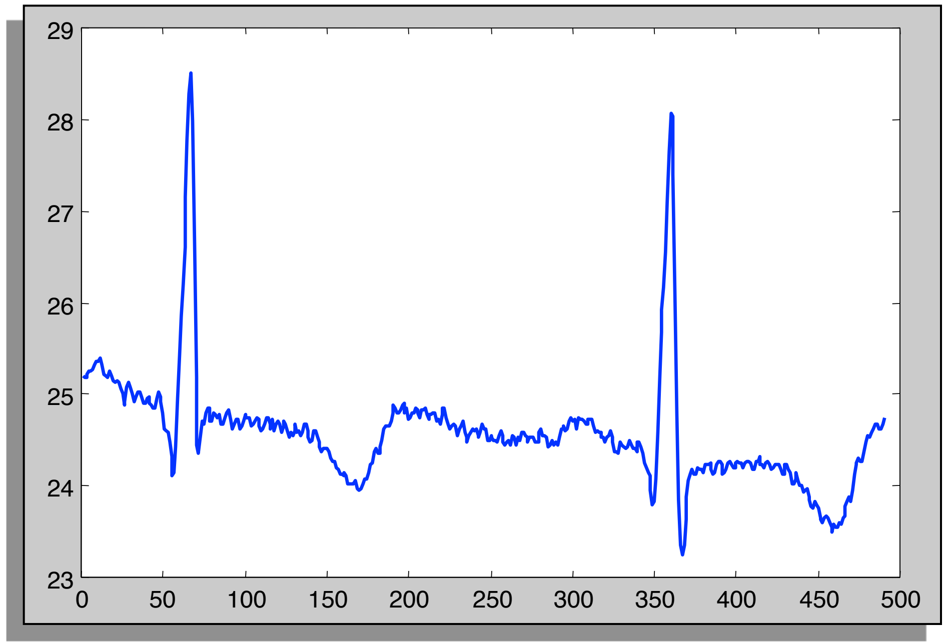
A time series is a collection of observations made sequentially in time.

25.1750
25.2250
25.2500
25.2500
25.2750
25.3250
25.3500
25.3500
25.4000
25.4000
25.3250
25.2250
25.2000
25.1750

..

..

24.6250
24.6750
24.6750
24.6250
24.6250
24.6250
24.6750
24.7500



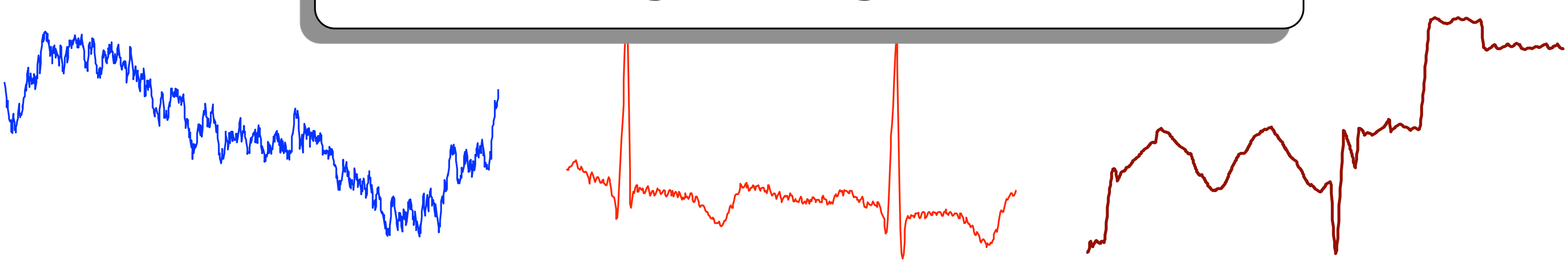
Virtually all similarity measurements, indexing and dimensionality reduction techniques discussed in this tutorial can be used with other data types

Time Series are Ubiquitous! I

People measure things…

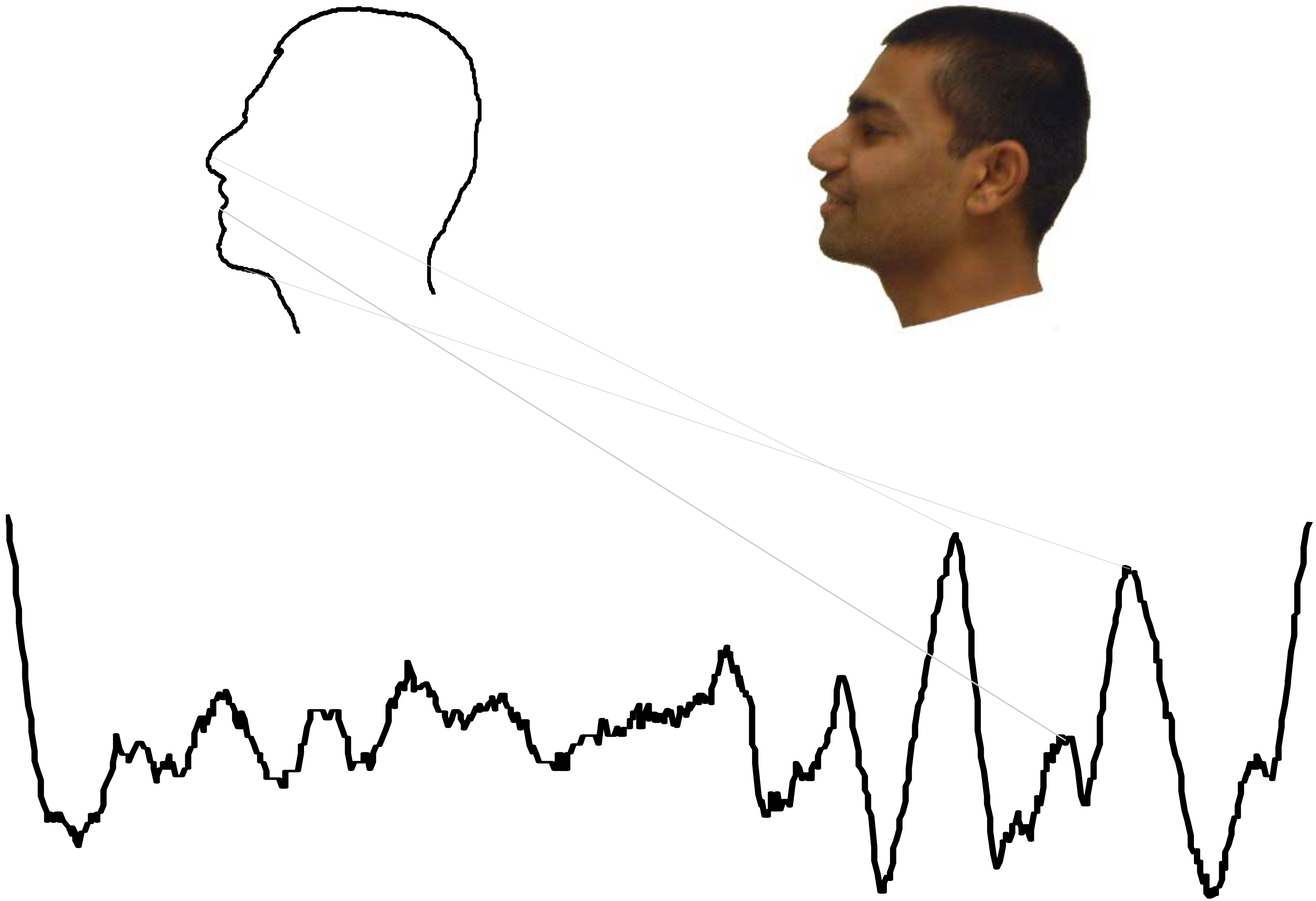
- *Their blood pressure*
- *George Bush's popularity rating*
- *The annual rainfall in Seattle*
- *The value of their Google stock*

…and things change over time…



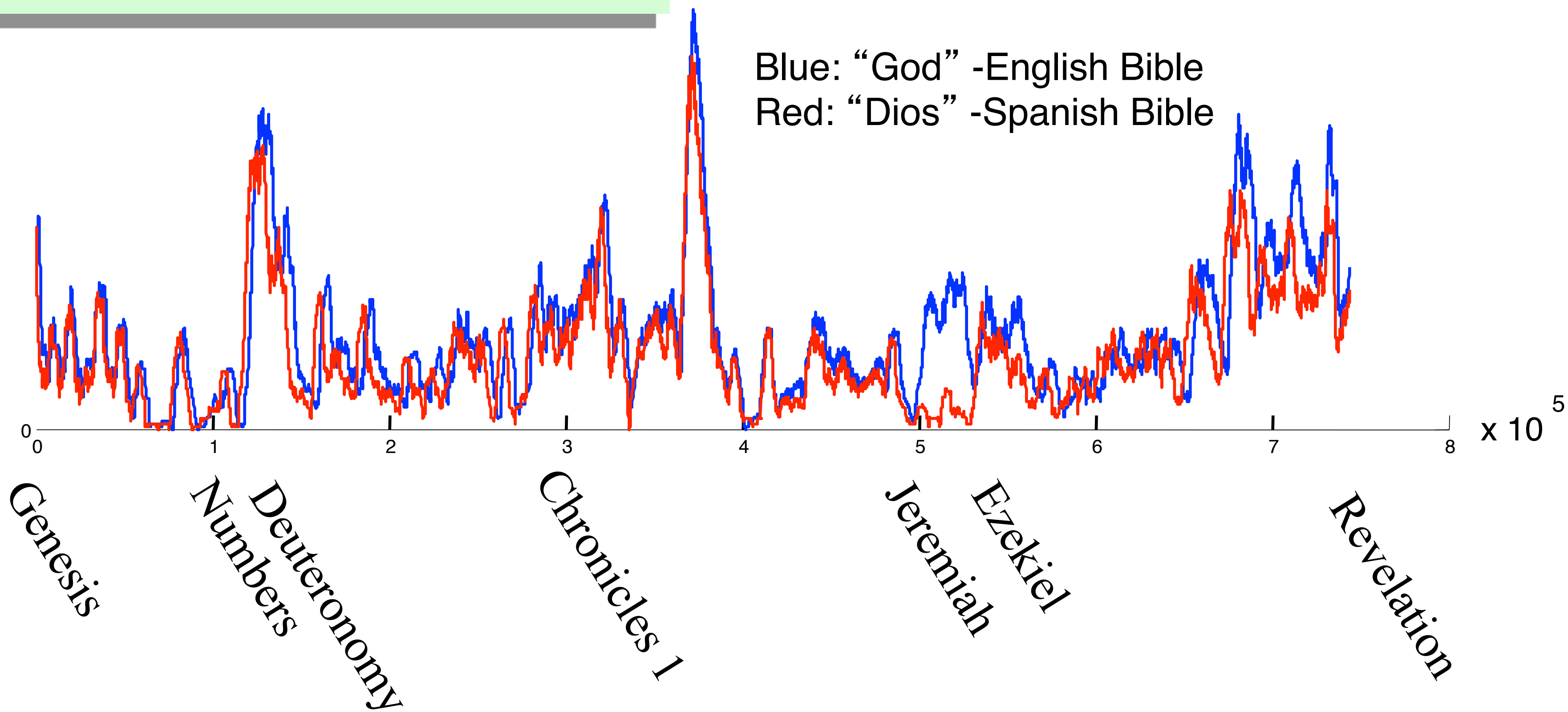
Thus time series occur in virtually every medical, scientific and businesses domain

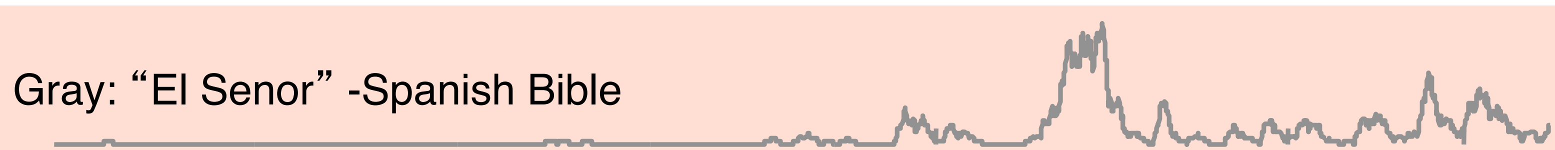
Image data, may best be thought of as time series…



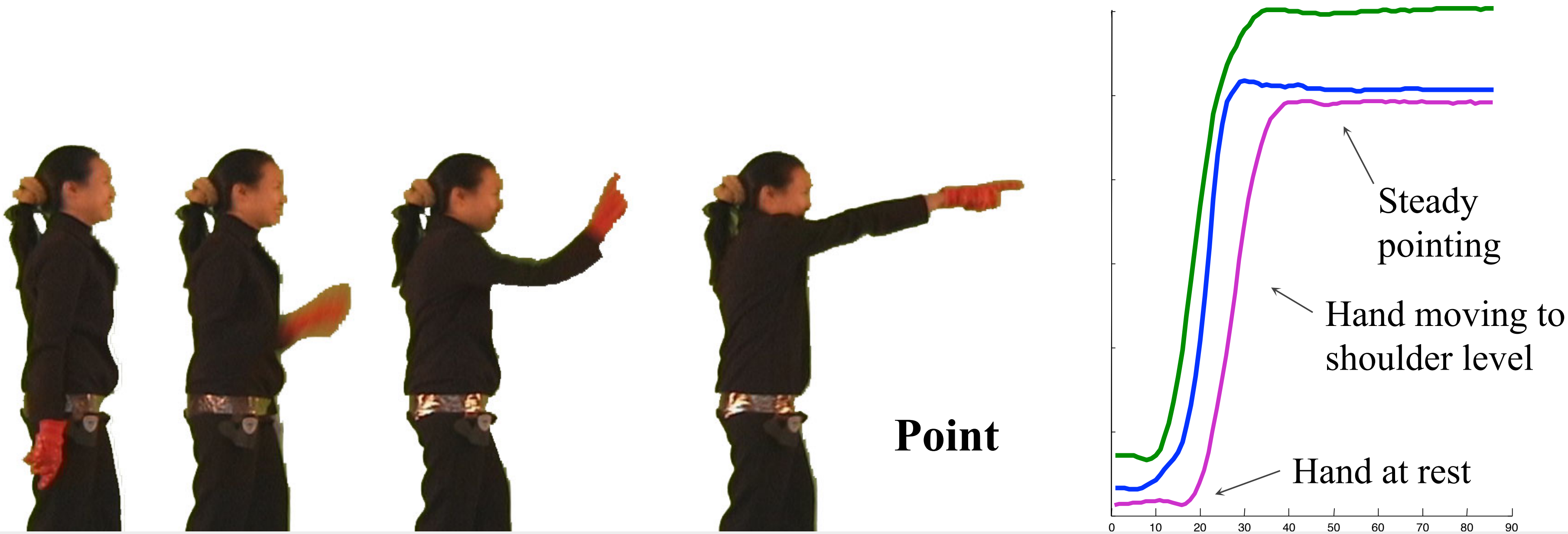
Text data, may best be thought of as time series…

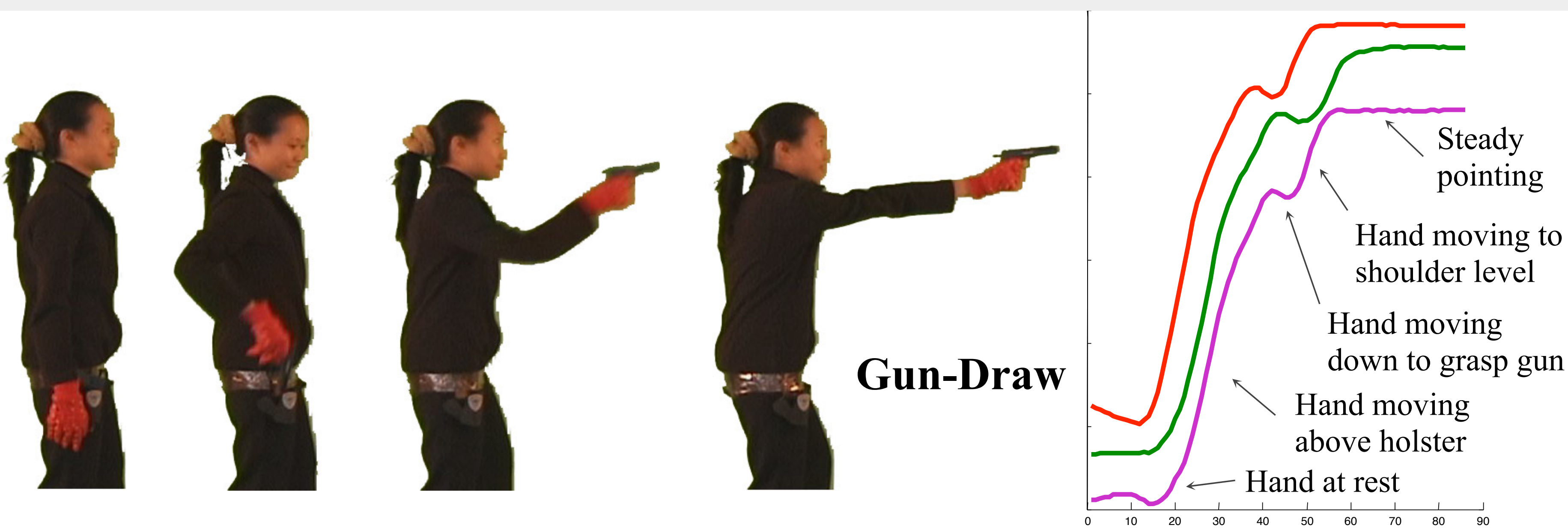
The local frequency of words in the Bible





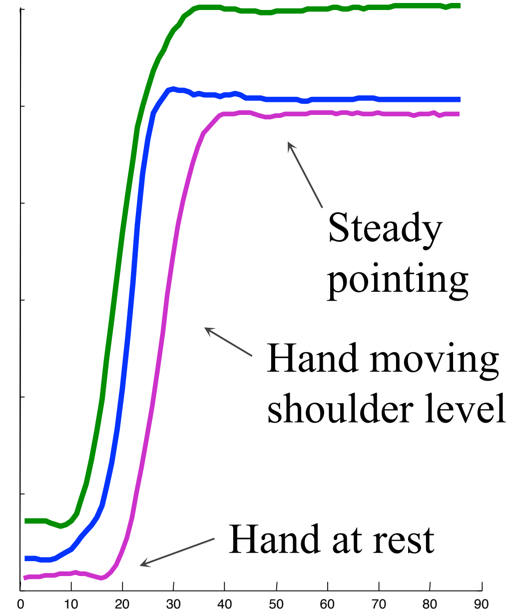
Video data, may best be thought of as time series…



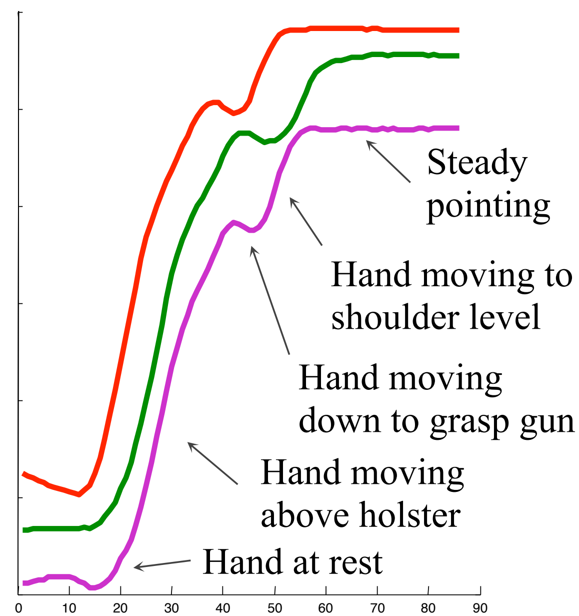


Video data, may best be thought of as time series…

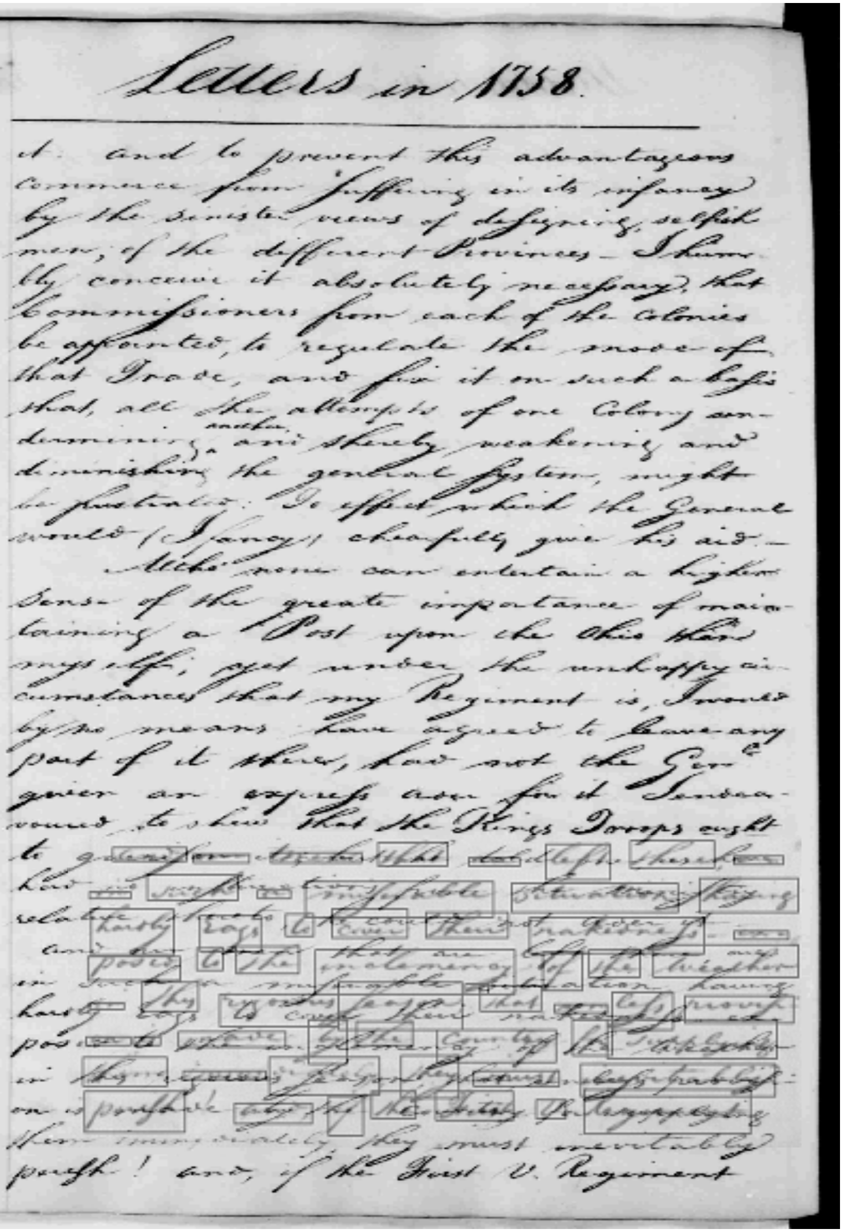
Point



Gun

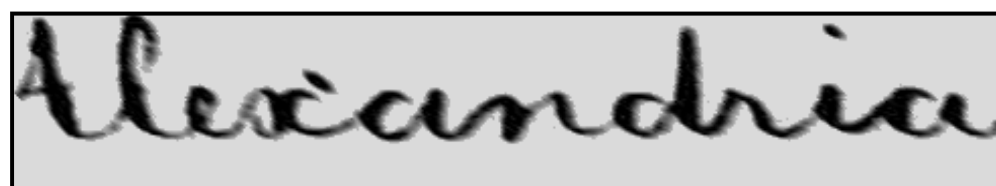


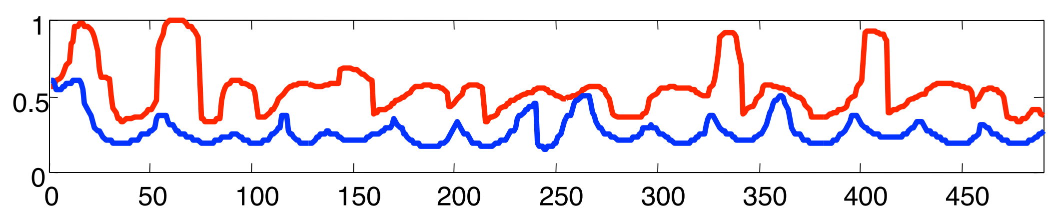
Handwriting data, may best be thought of as time series…



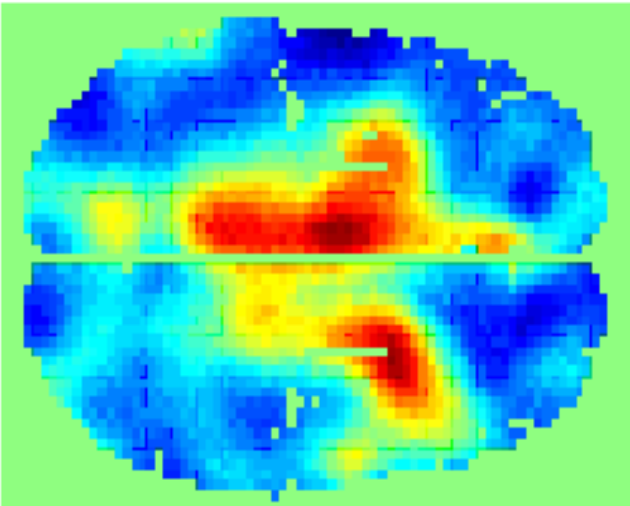
George Washington Manuscript

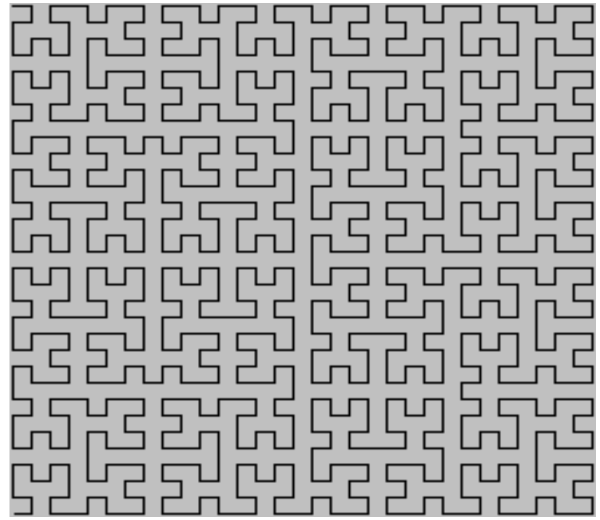
George Washington
1732-1799

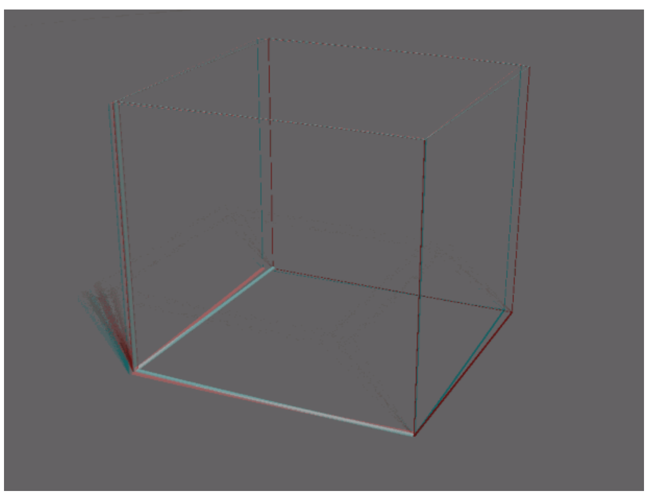


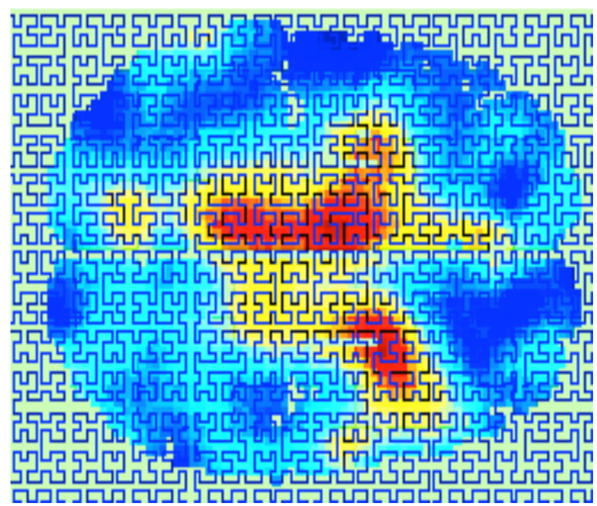


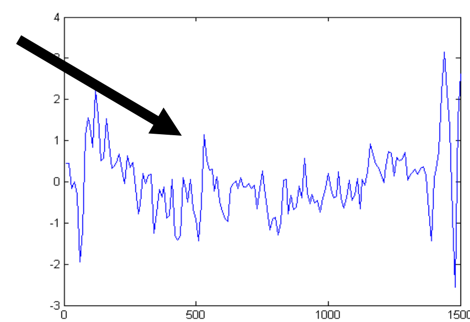
Brain scans (3D voxels), may best be thought of as time series…







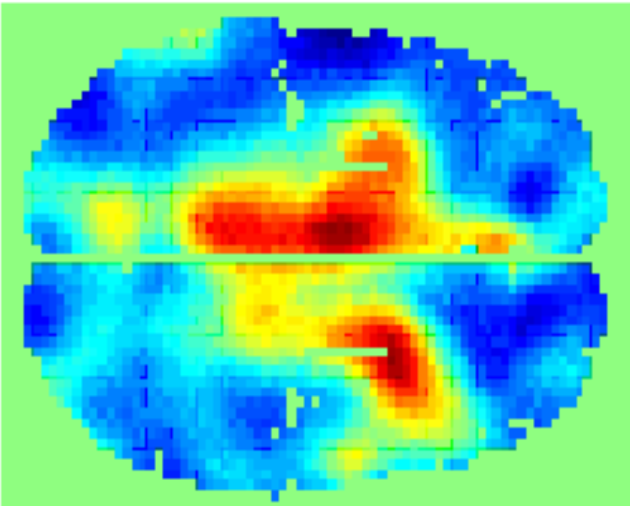


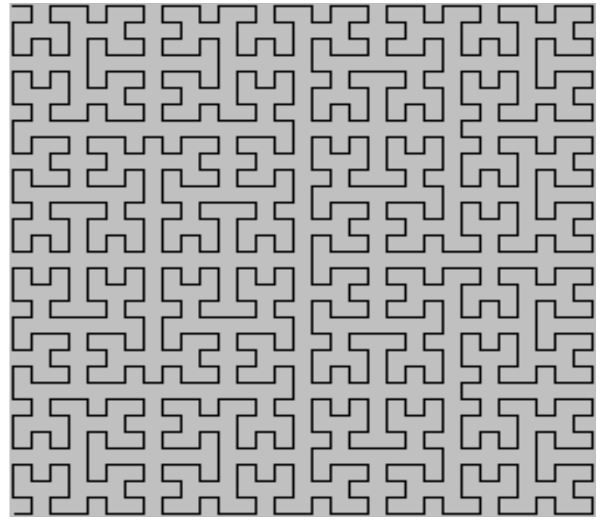


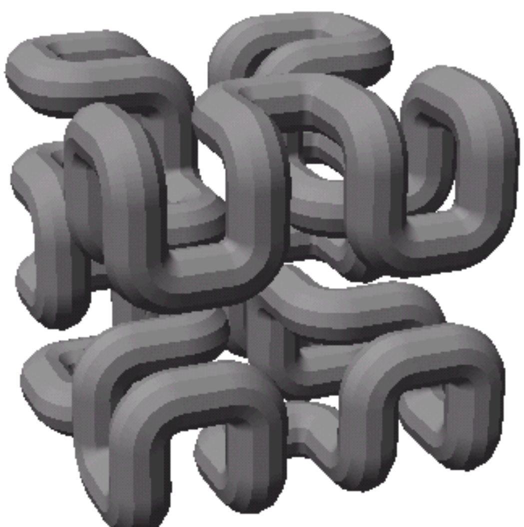
Works with
3D glasses!

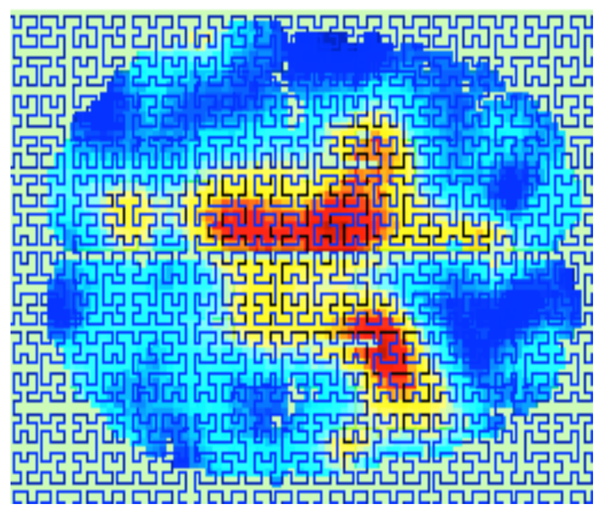
Wang, Kontos, Li and Megalooikonomou ICASSP 2004

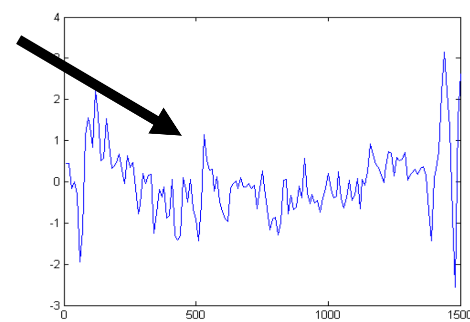
Brain scans (3D voxels), may best be thought of as time series…











Wang, Kontos, Li and Megalooikonomou ICASSP 2004

Why is Working With Time Series so Difficult? Part I

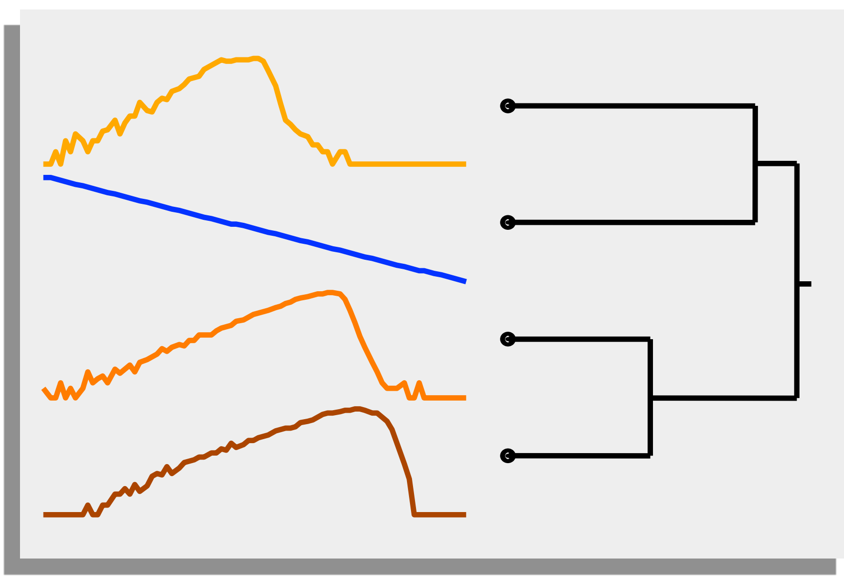
Answer: How do we work with very large databases?

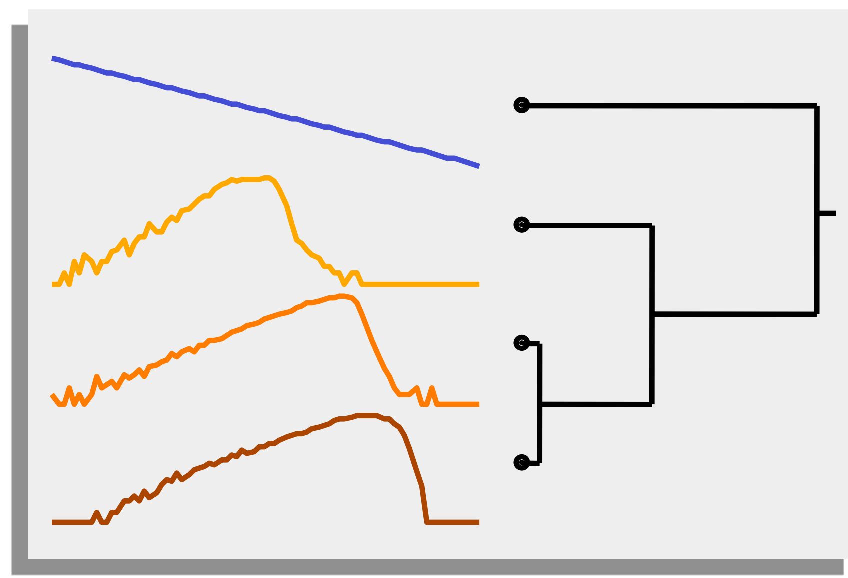
- 1 Hour of EKG data: 1 Gigabyte.
- Typical Weblog: 5 Gigabytes per week.
- Space Shuttle Database: 200 Gigabytes and growing.
- Macho Database: 3 Terabytes, updated with 3 gigabytes a day.

Since most of the data lives on disk (or tape), we need a representation of the data we can efficiently manipulate.

Why is Working With Time Series so Difficult? Part II

Answer: We are dealing with subjectivity





The definition of similarity depends on the user, the domain and the task at hand. We need to be able to handle this subjectivity.

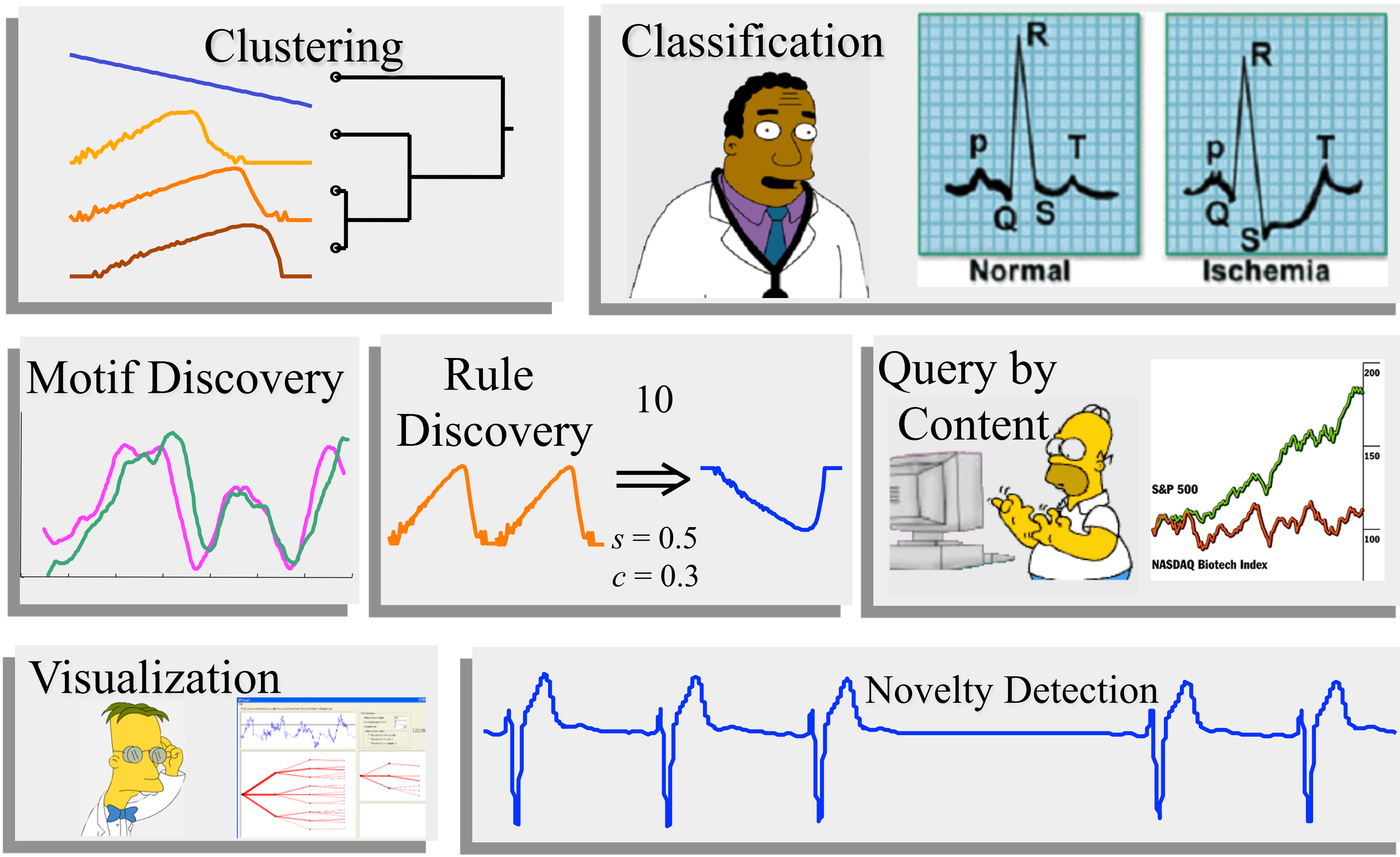
Why is working with time series so difficult? Part III

Answer: Miscellaneous data handling problems.

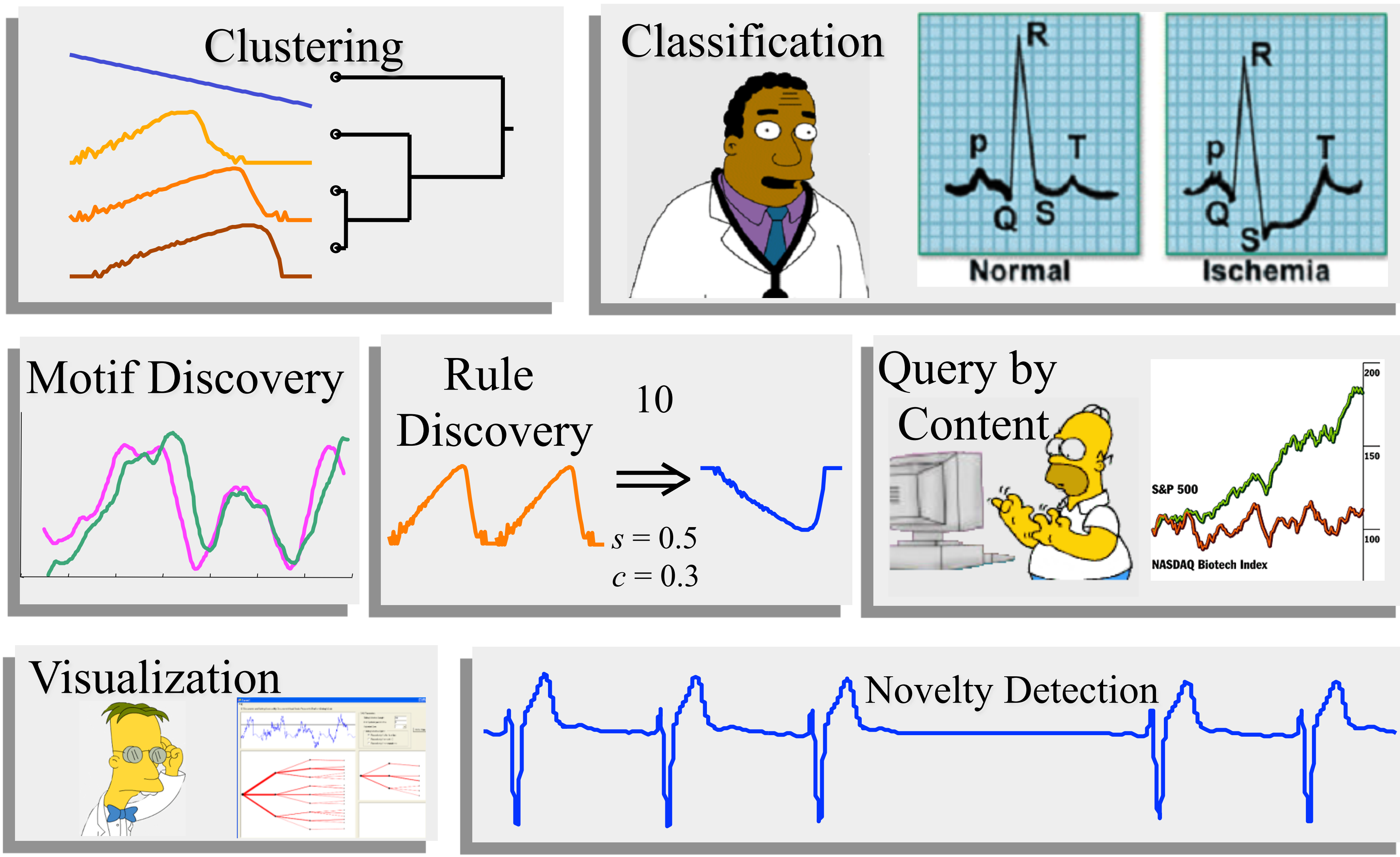
- Differing data formats.
- Differing sampling rates.
- Noise, missing values, etc.

We will not focus on these issues in this tutorial.

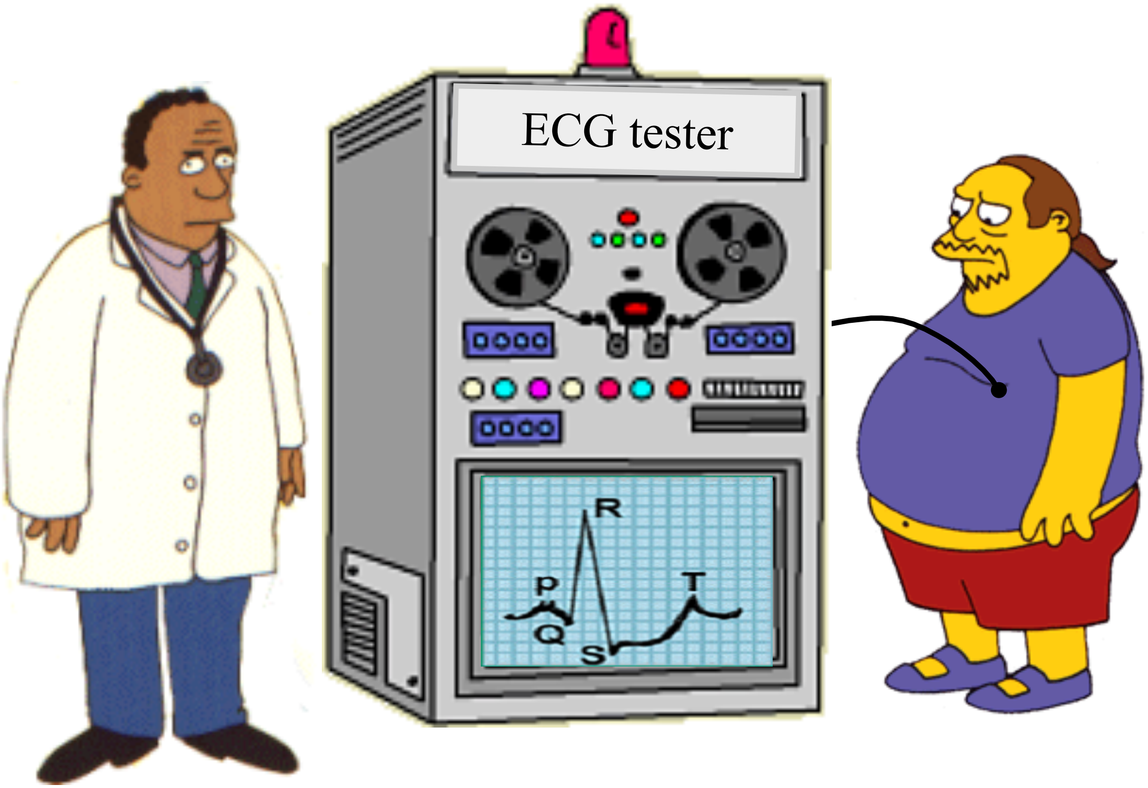
What do we want to do with the time series data?



All these problems require <u>similarity</u> matching



Here is a simple motivation for the first part of the tutorial



You go to the doctor because of chest pains. Your ECG looks strange…

You doctor wants to search a database to find **similar** ECGs, in the hope that they will offer clues about your condition...

Two questions:

- **How do we define similar?**
- How do we search quickly?

What is Similarity?

The quality or state of being similar; likeness; resemblance; as, a similarity of features. **Webster's Dictionary**

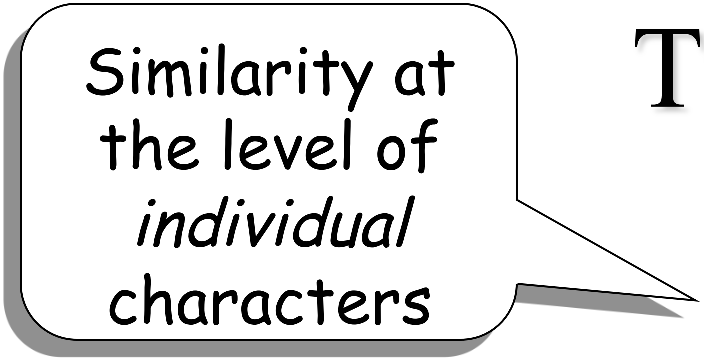
Similarity is hard to define, but…
"*We know it when we see it*"

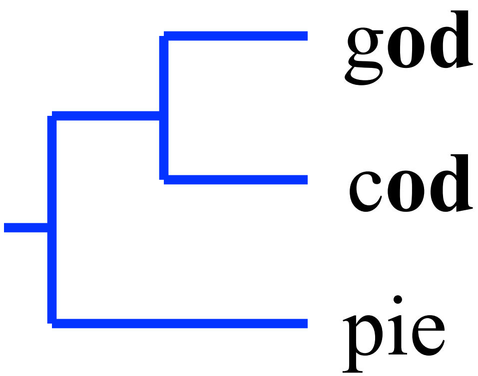
The real meaning of similarity is a philosophical question.

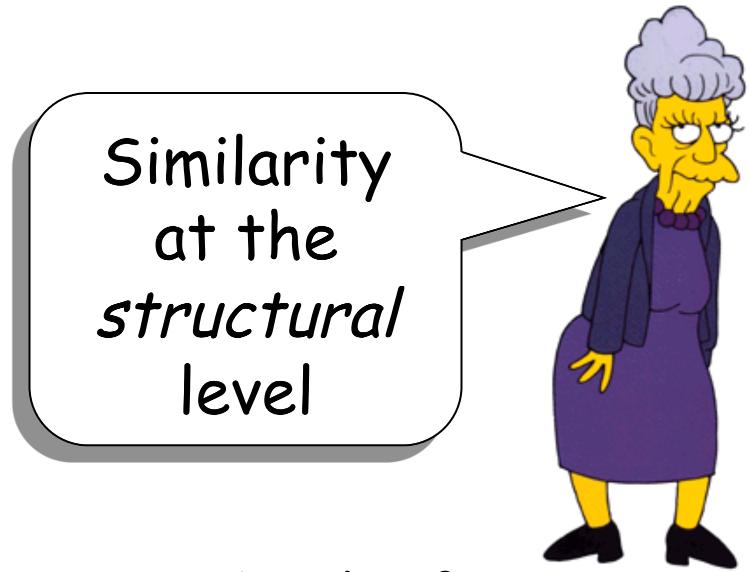
We will take a more pragmatic approach.

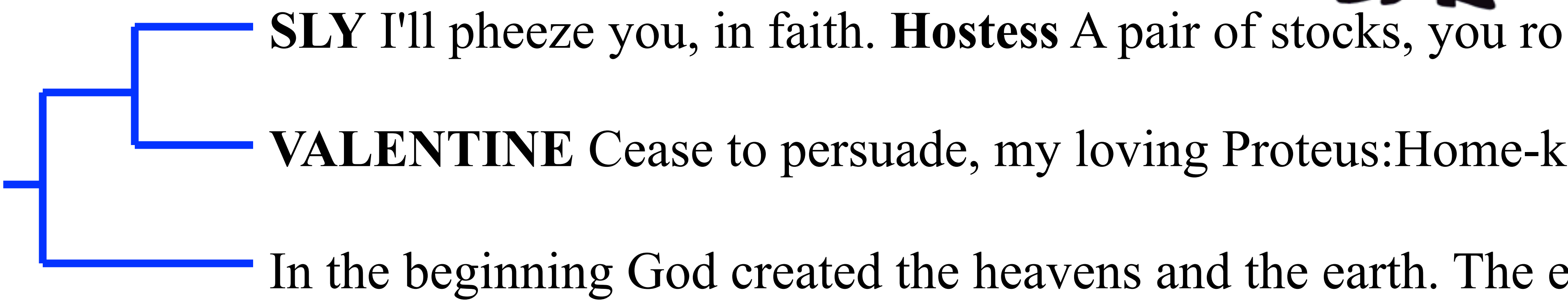
Two Kinds of Similarity

text





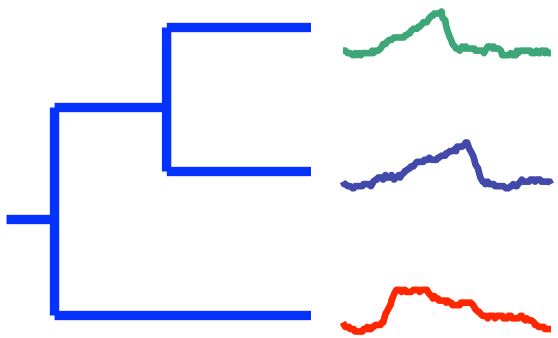




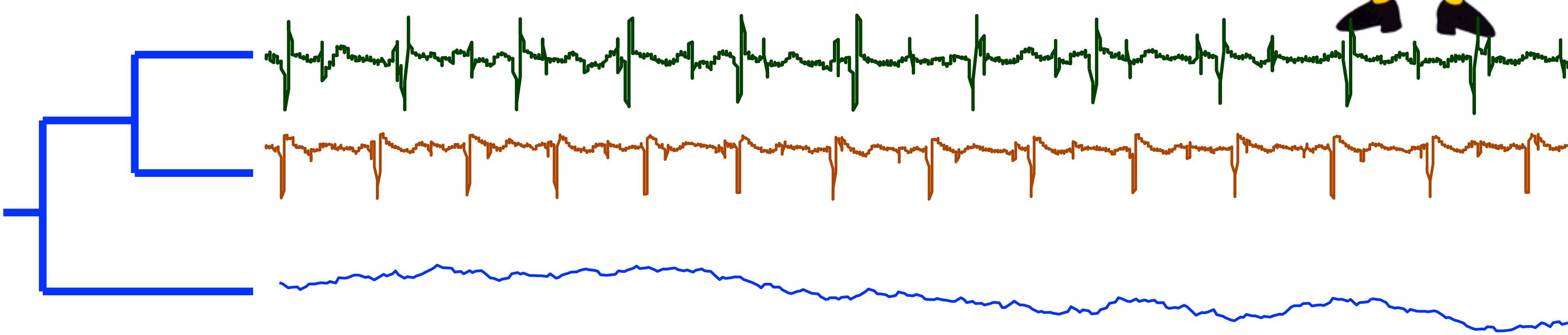
Two Kinds of Similarity

time series

Similarity at the level of *shape*
Next 40 minutes



Similarity at the *structural* level
Another 10 minutes



Defining Distance Measures

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) is denoted by $D(O_1,O_2)$

What properties are desirable in a distance measure?

- $D(A,B) = D(B,A)$ *Symmetry*
- $D(A,A) = 0$ *Constancy*
- $D(A,B) = 0$ IIf $A = B$ *Positivity*
- $D(A,B) \leq D(A,C) + D(B,C)$ *Triangular Inequality*

Intuitions behind desirable distance measure properties I

$D(\mathrm{A},\mathrm{B}) = D(\mathrm{B},\mathrm{A})$ *Symmetry*

D( , ) = D( , )

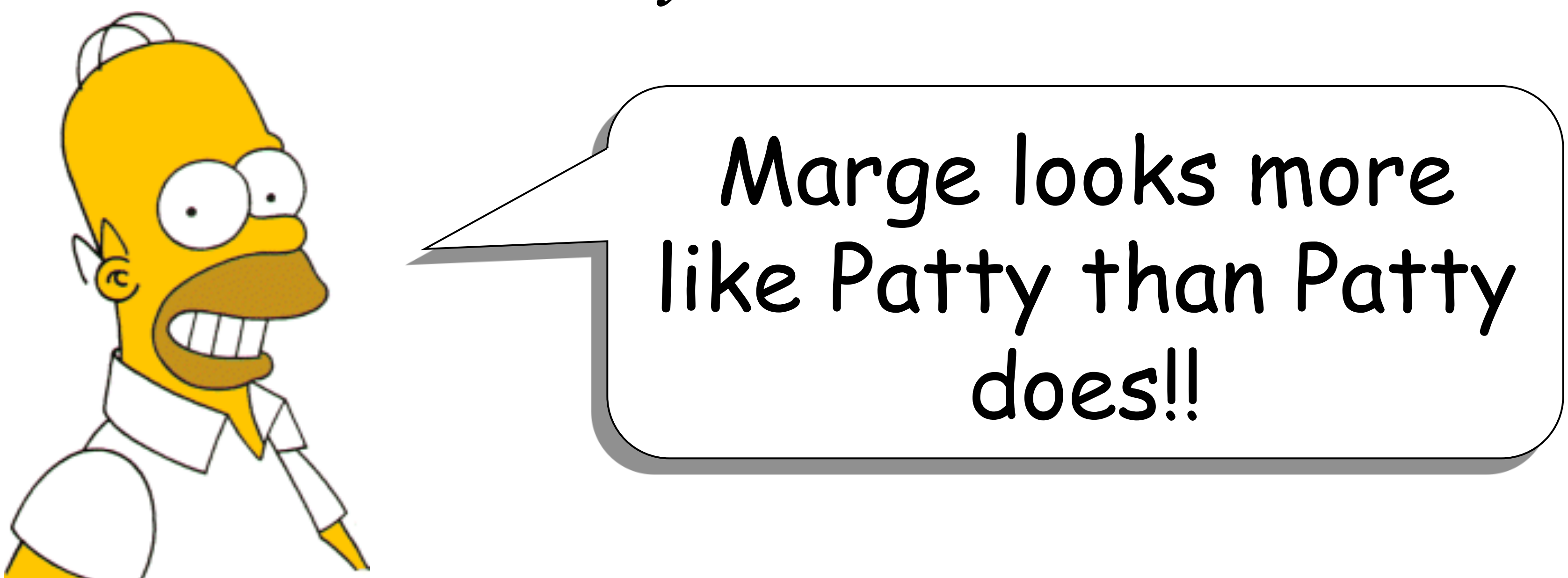
Otherwise you could claim:

Intuitions behind desirable distance measure properties II

$D(A,A) = 0$ *Constancy of Self-Similarity*

D( , ) = 0

Otherwise you could claim:

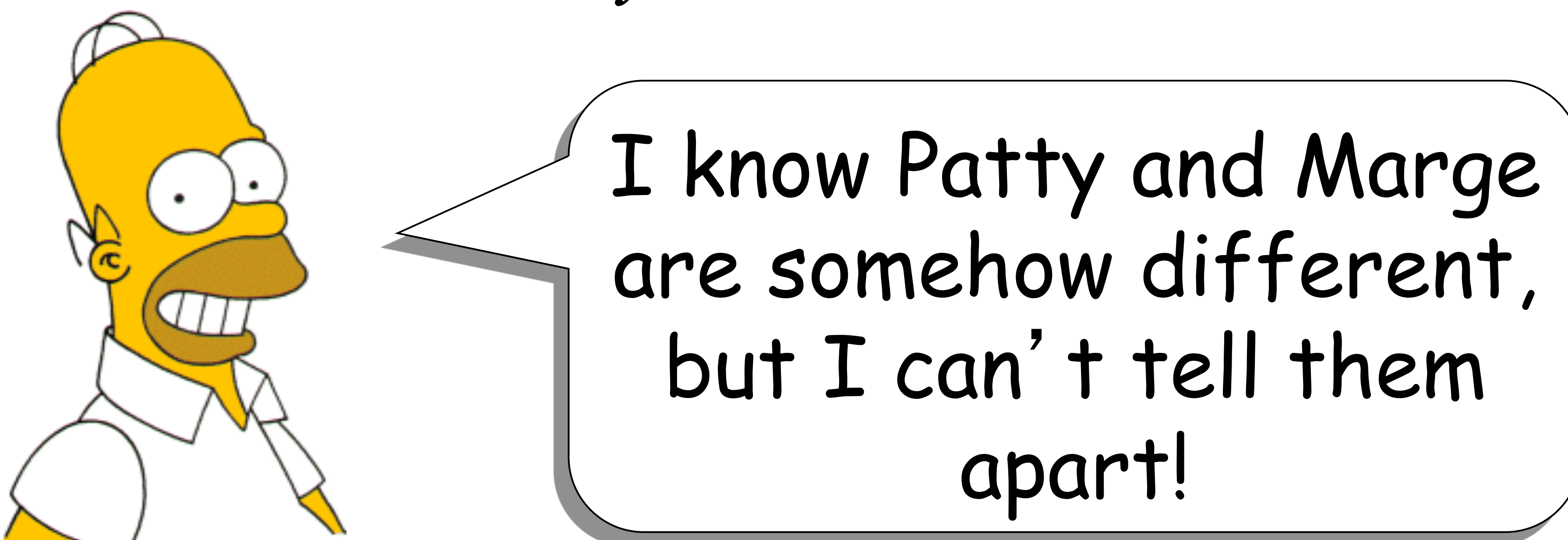


Intuitions behind desirable distance measure properties III

D(A,B) = 0, IIf A=B *Positivity*

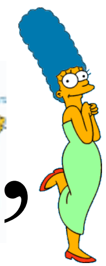
D(,) = 0, IIF  = 

Otherwise you could claim:



Intuitions behind desirable distance measure properties IIII

$D(\mathrm{A,B}) \leq D(\mathrm{A,C}) + D(\mathrm{B,C})$ *Triangular Inequality*

D( , ) ≤ D( , ) + D( , )

Otherwise you could claim:

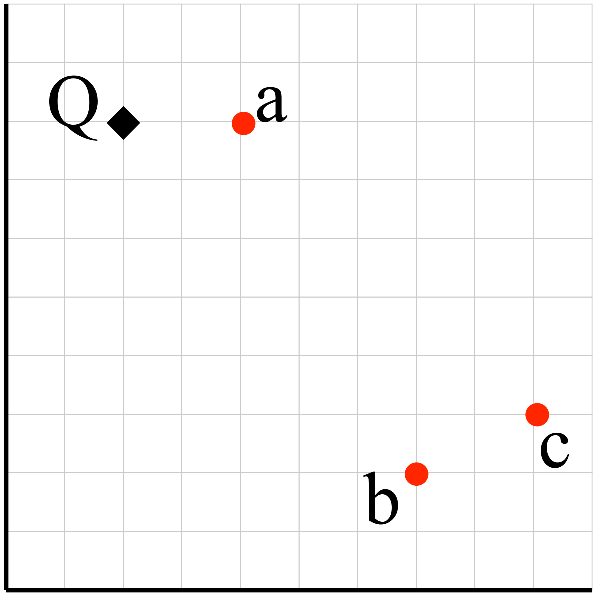
Patty looks like Marge, Selma also looks like Marge, But Patty looks nothing like Selma!

Why is the Triangular Inequality so Important?

Virtually all techniques to index data require the triangular inequality to hold.

Suppose I am looking for the closest point to Q, in a database of 3 objects.

Further suppose that the triangular inequality holds, and that we have precompiled a table of distance between all the items in the database.



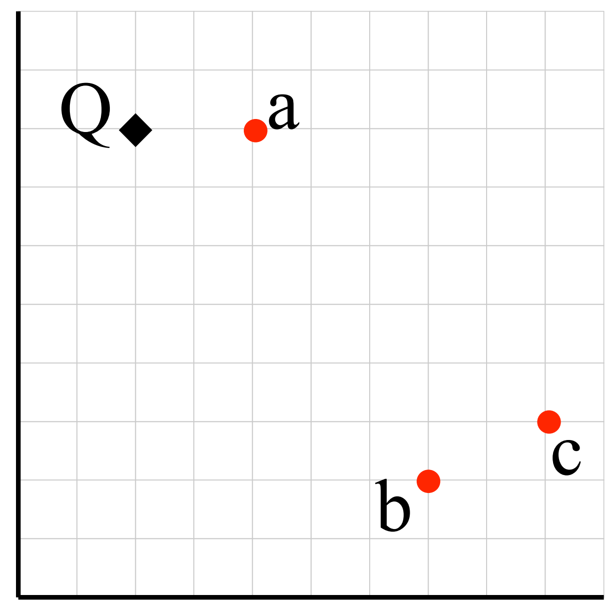
	a	b	c
a		6.70	7.07
b			2.30
c			

Why is the Triangular Inequality so Important?

Virtually all techniques to index data require the triangular inequality to hold.

I find **a** and calculate that it is 2 units from Q, it becomes my *best-so-far*. I find **b** and calculate that it is 7.81 units away from Q.

I don't have to calculate the distance from Q to **c**!



	a	b	c
a		6.70	7.07
b			**2.30**
c			

I know $D(Q,\mathbf{b}) \leq D(Q,\mathbf{c}) + D(\mathbf{b},\mathbf{c})$

$$D(Q,\mathbf{b}) - D(b,\mathbf{c}) \leq D(Q,\mathbf{c})$$

$$7.81 - 2.30 \leq D(Q,\mathbf{c})$$

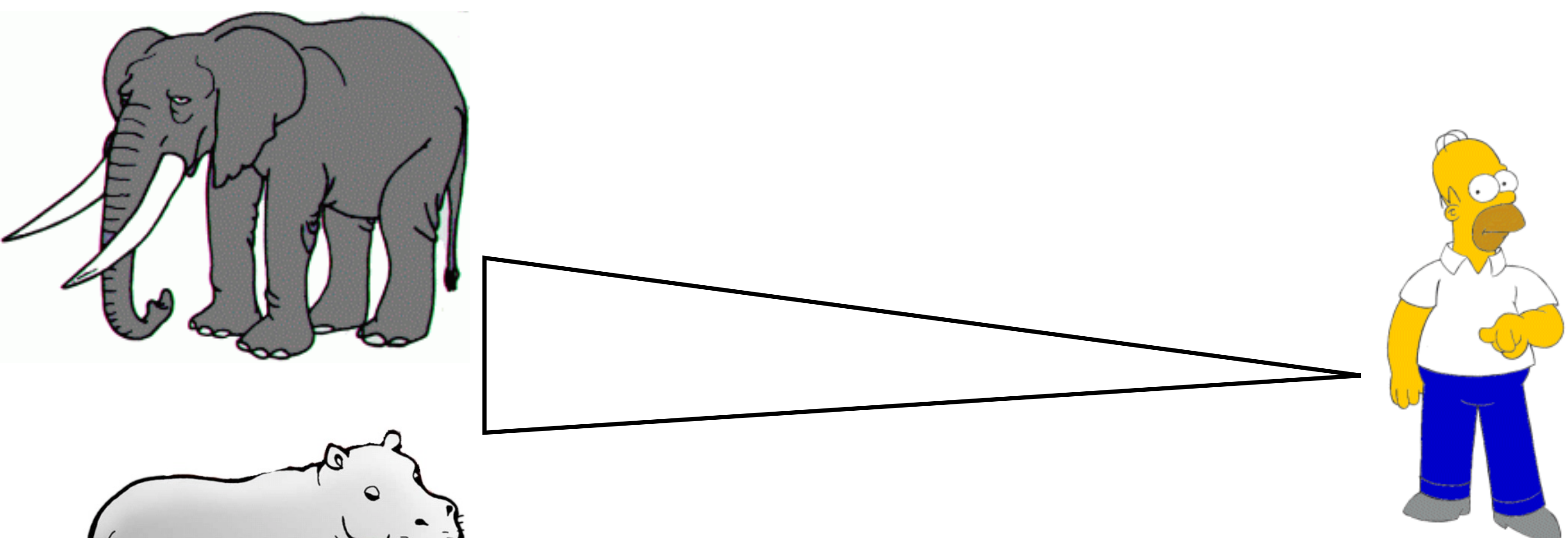
$$5.51 \leq D(Q,\mathbf{c})$$

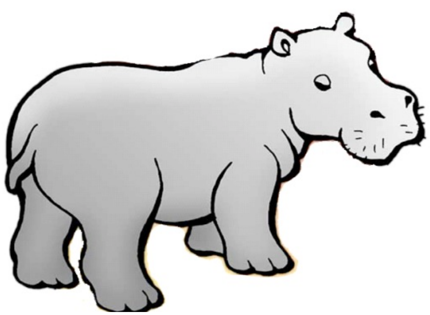
So I know that **c** is at least 5.51 units away, but my *best-so-far* is only 2 units away.

A Final Thought on the Triangular Inequality I

Sometimes the triangular inequality requirement maps nicely onto human intuitions.

Consider the similarity between a hippo, an elephant and a man.



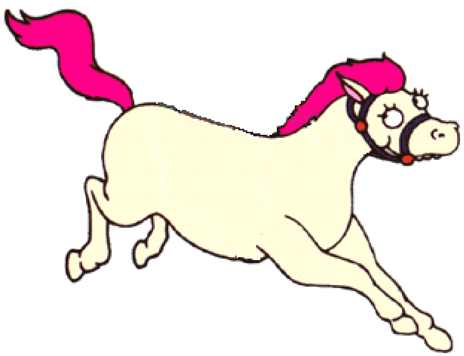


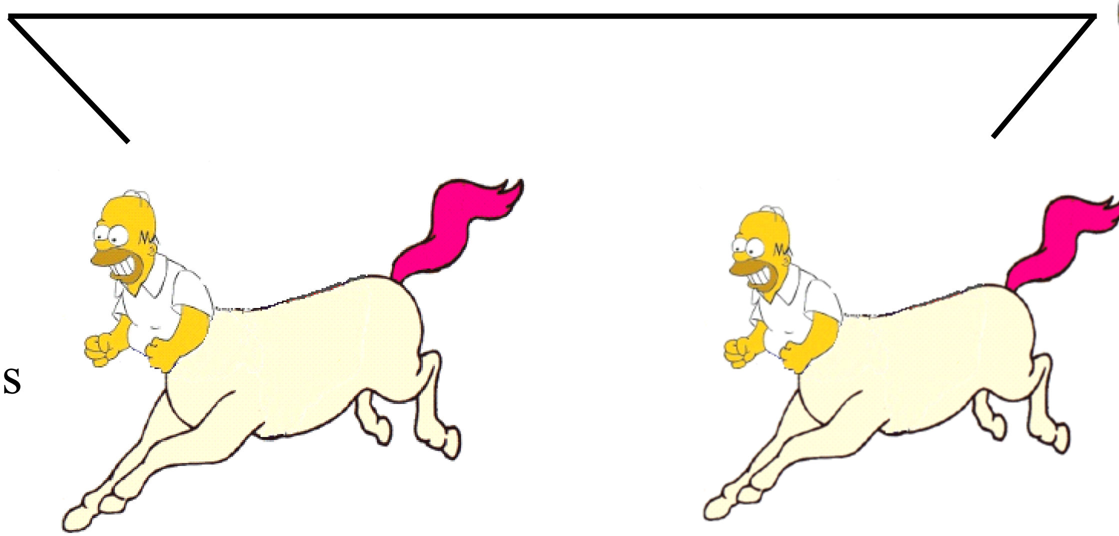
The hippo and the elephant are very similar, and both are very unlike the man.

A Final Thought on the Triangular Inequality II

Sometimes the triangular inequality requirement *fails* to map onto human intuition.

Consider the similarity between the horse, a man and the centaur…





The **horse** and the **man** are very different, but both share many features with the **centaur**. This relationship does not obey the triangular inequality.

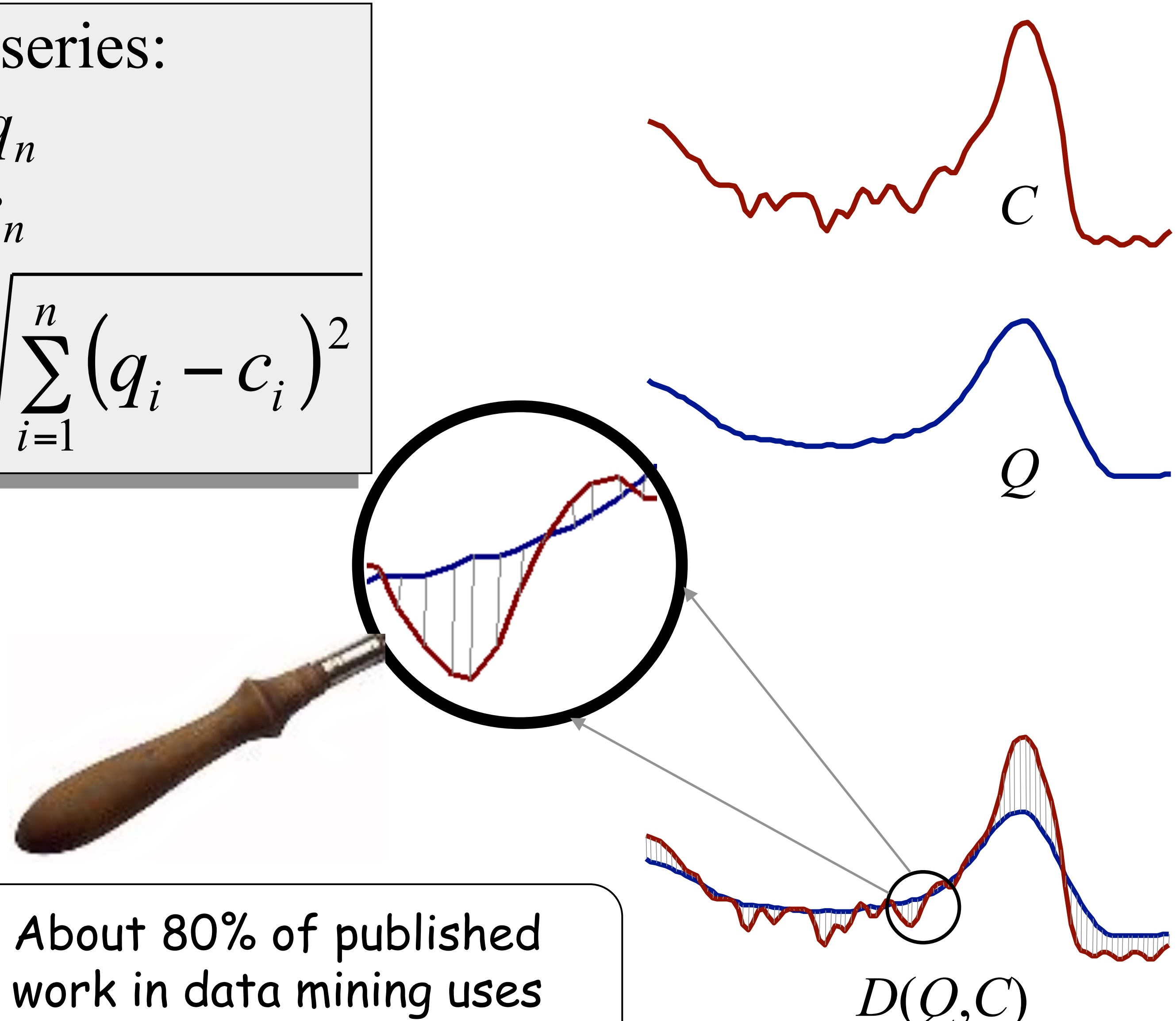
This example due to Remco C. Veltkamp

Euclidean Distance Metric

Given two time series:

$$Q = q_1 \ldots q_n$$
$$C = c_1 \ldots c_n$$

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$



About 80% of published work in data mining uses Euclidean distance

Optimizing the Euclidean Distance Calculation

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2}$$

$$D_{squared}(Q,C) \equiv \sum_{i=1}^{n}(q_i - c_i)^2$$

Euclidean distance and Squared Euclidean distance are equivalent in the sense that they return the same rankings, clusterings and classifications

Instead of using the **Euclidean distance** we can use the **Squared Euclidean distance**

This optimization helps with CPU time, but most problems are I/O bound.

Preprocessing the data before distance calculations

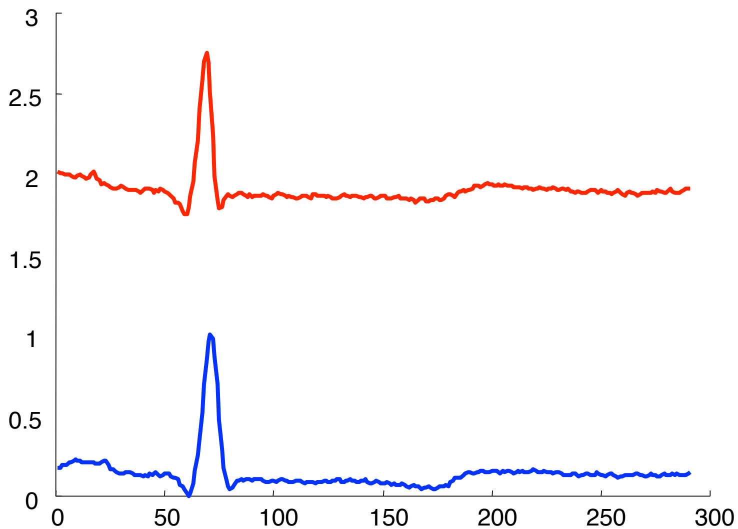
If we naively try to measure the distance between two "raw" time series, we may get very unintuitive results

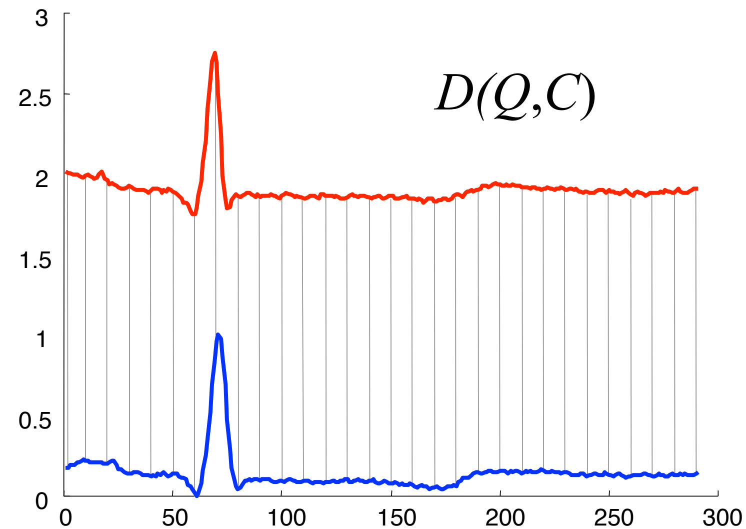
This is because Euclidean distance is very sensitive to some "distortions" in the data. For most problems these distortions are not meaningful, and thus we can and should remove them

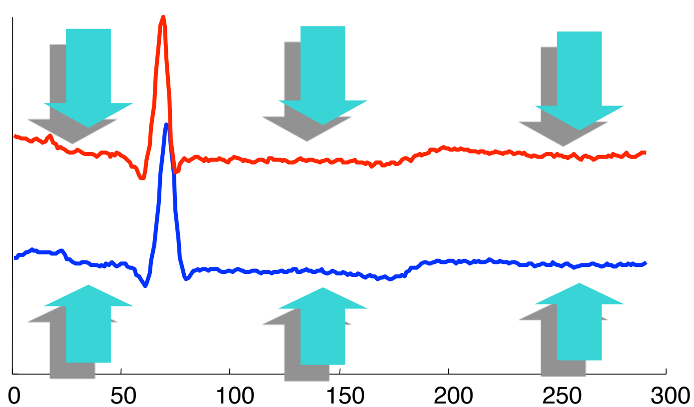
In the next few slides we will discuss the 4 most common distortions, and how to remove them

- Offset Translation
- Amplitude Scaling
- Linear Trend
- Noise

Transformation I: Offset Translation



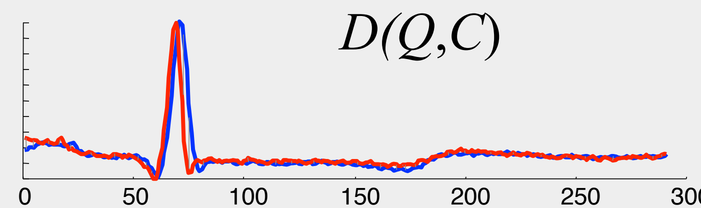




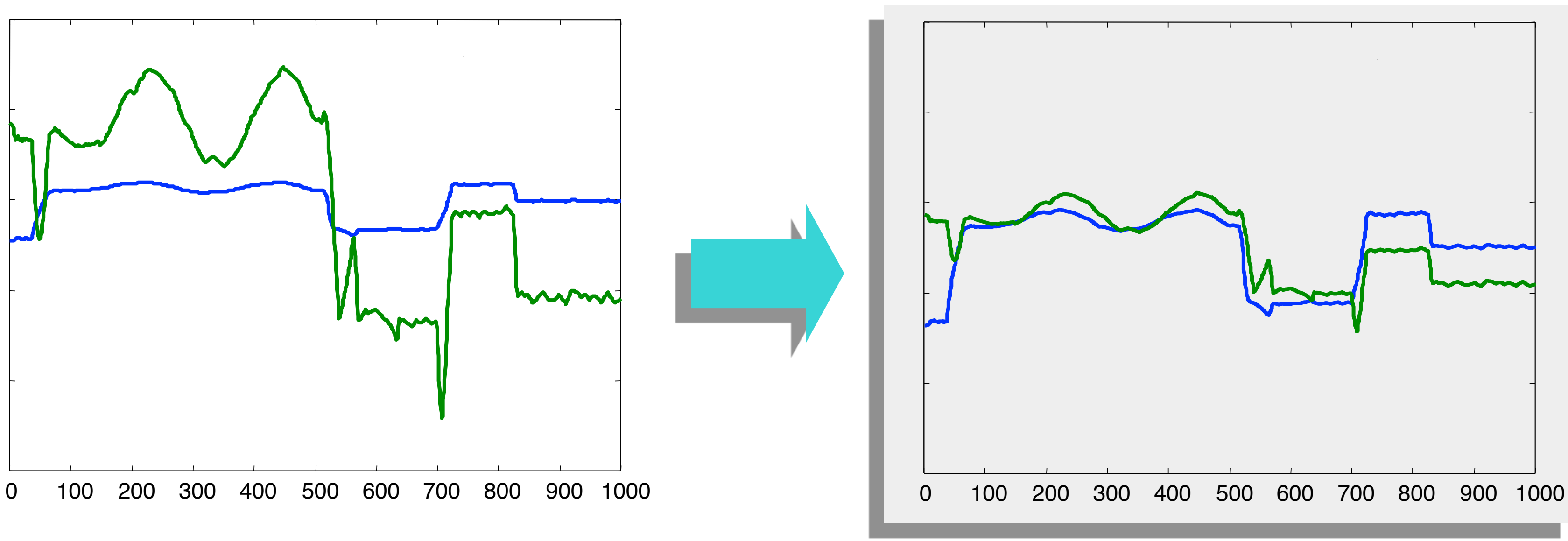
$Q = Q$ - mean(Q)

$C = C$ - mean(C)

$D(Q,C)$



Transformation II: Amplitude Scaling

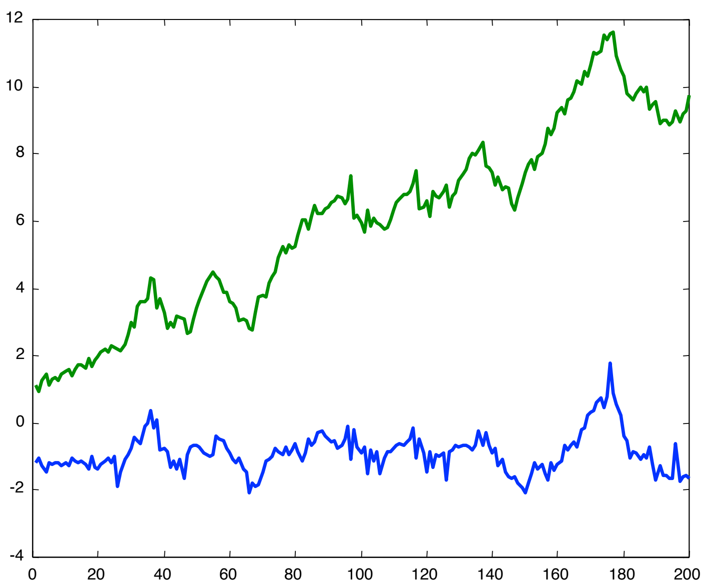


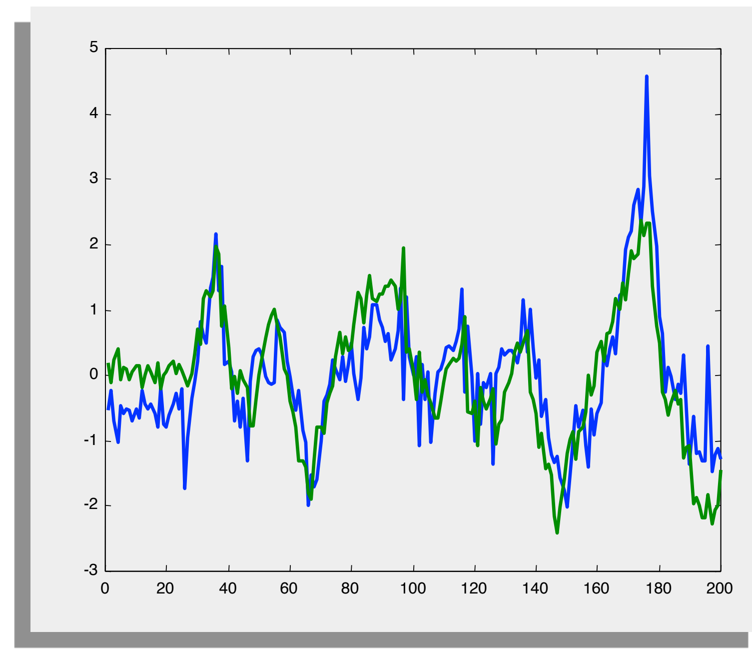
$Q = (Q - \text{mean}(Q)) \;/\; \text{std}(Q)$

$C = (C - \text{mean}(C)) \;/\; \text{std}(C)$

$D(Q,C)$

Transformation III: Linear Trend





The intuition behind removing linear trend is…

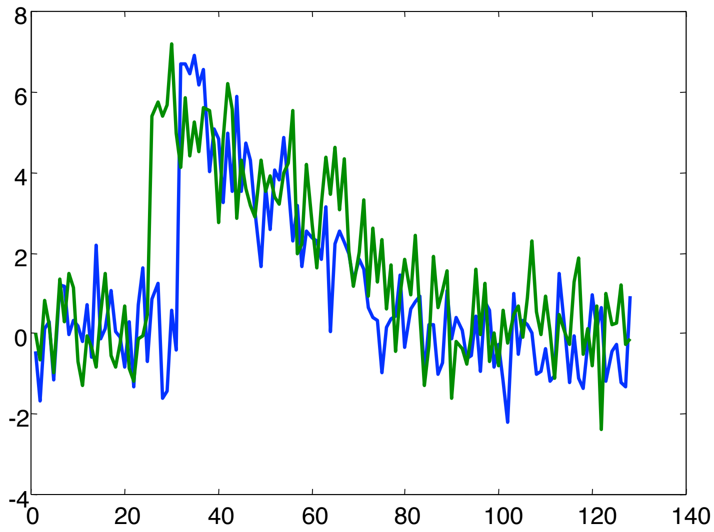
Fit the best fitting straight line to the time series, then subtract that line from the time series.

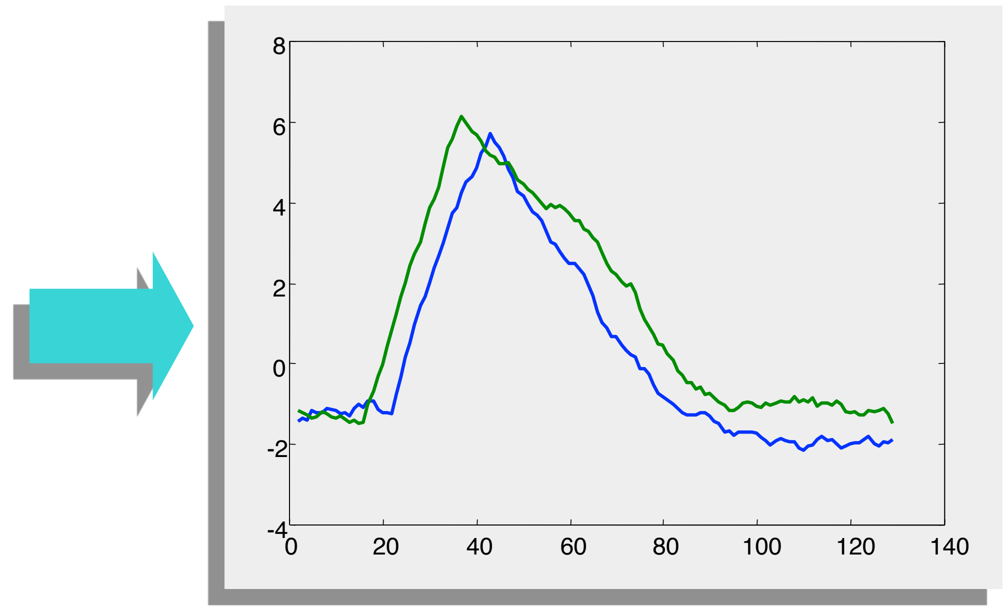
Removed **linear trend**

Removed offset translation

Removed amplitude scaling

Transformation IIII: Noise





$Q = \text{smooth}(Q)$

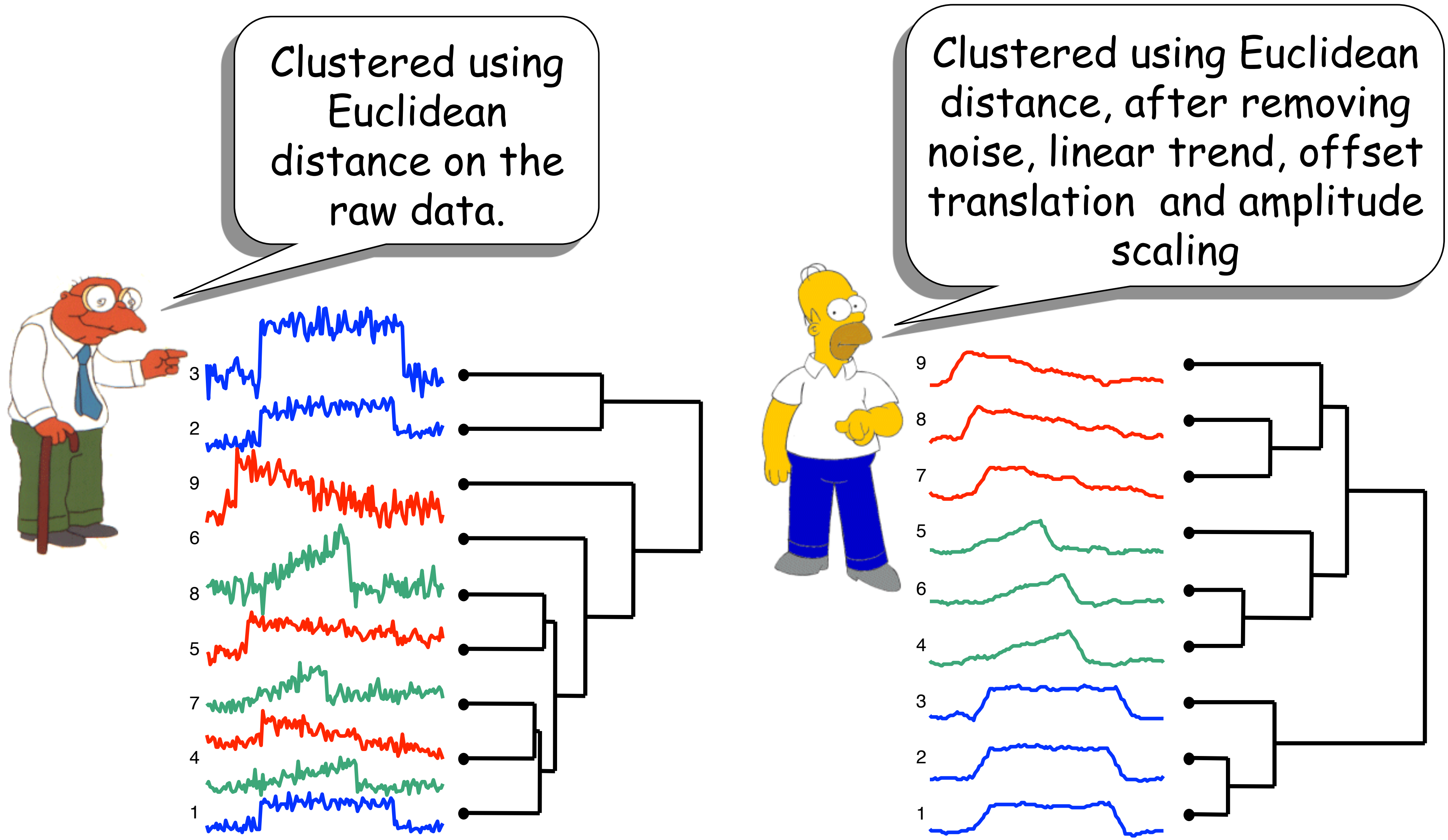
$C = \text{smooth}(C)$

$D(Q,C)$

The intuition behind removing noise is...

Average each datapoints value with its neighbors.

A Quick Experiment to Demonstrate the Utility of Preprocessing the Data



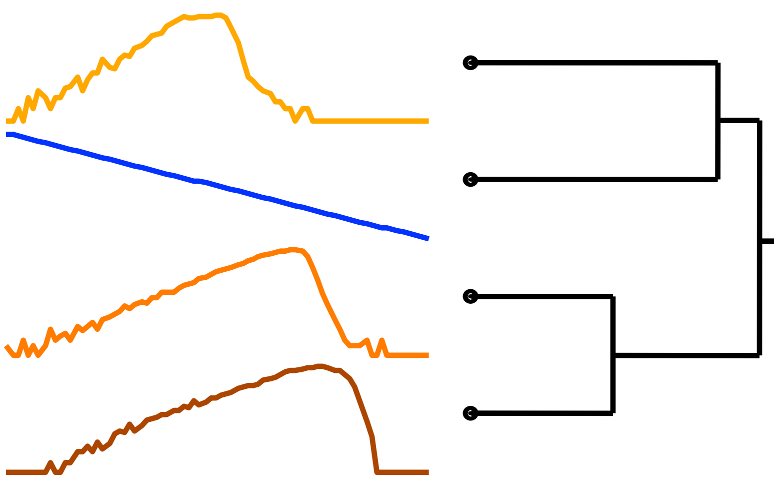
Summary of Preprocessing

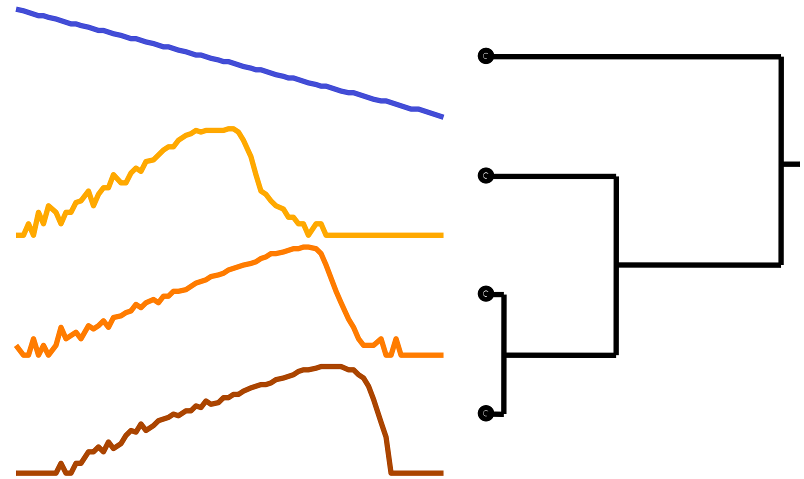
The "raw" time series may have distortions which we should remove before clustering, classification etc

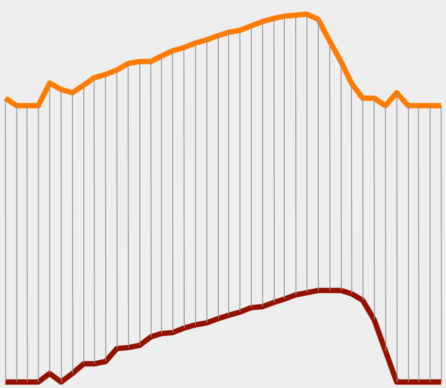
Of course, sometimes the distortions are the most interesting thing about the data, the above is only a general rule

We should keep in mind these problems as we consider the high level representations of time series which we will encounter later (DFT, Wavelets etc). Since these representations often allow us to handle distortions in elegant ways

Dynamic Time Warping

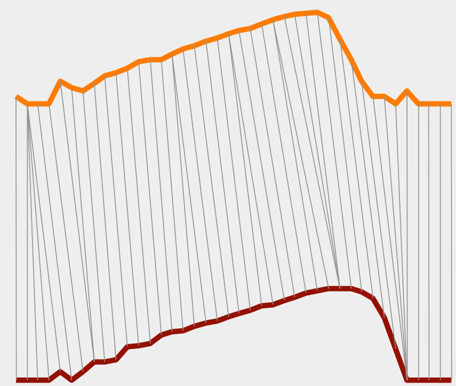






Fixed Time Axis

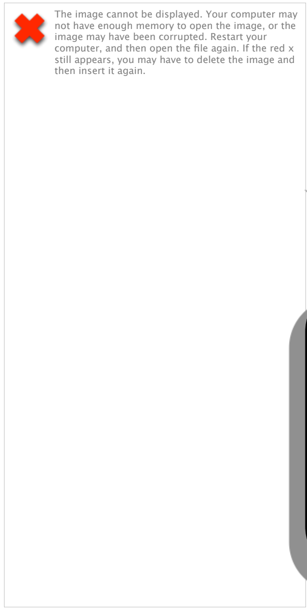
Sequences are aligned "one to one".

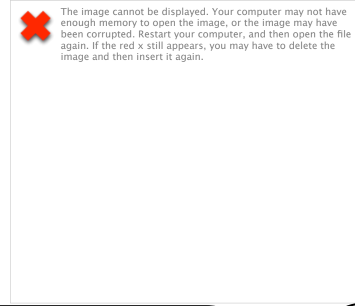


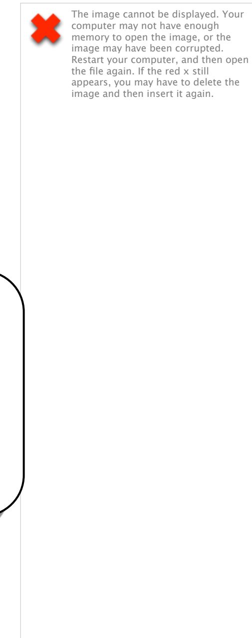
"Warped" Time Axis

Nonlinear alignments are possible.

Note: We will first see the utility of DTW, then see how it is calculated.

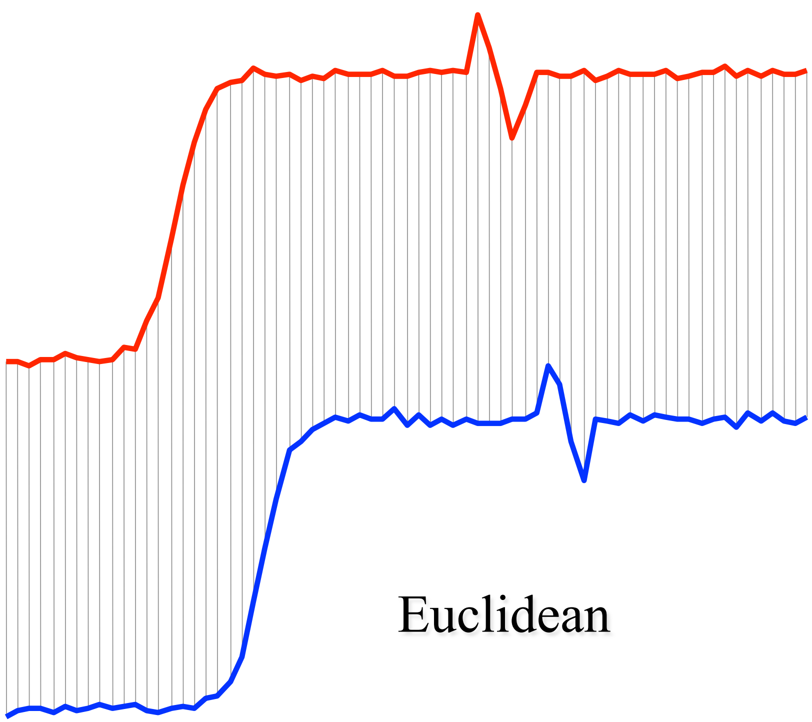


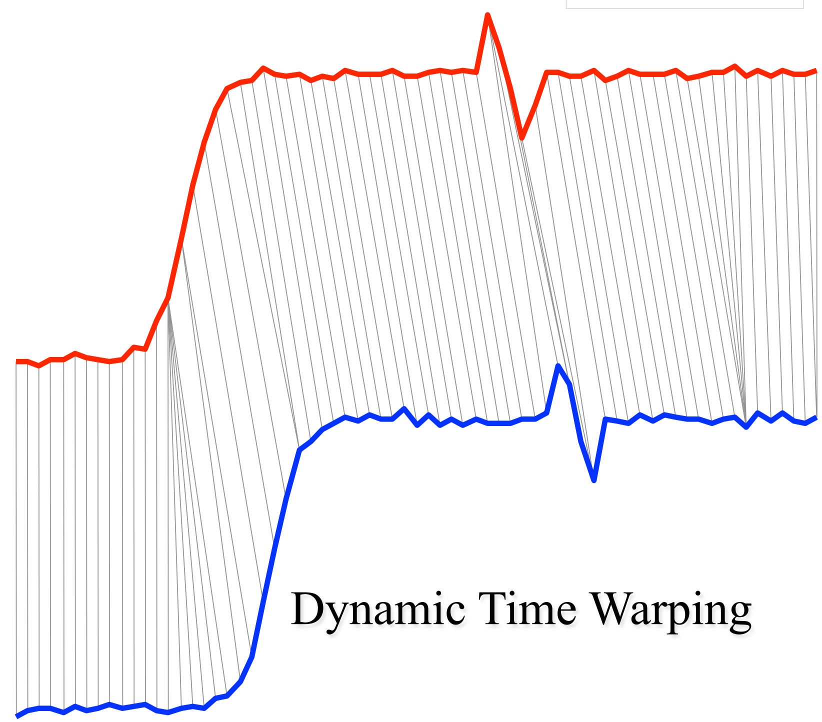




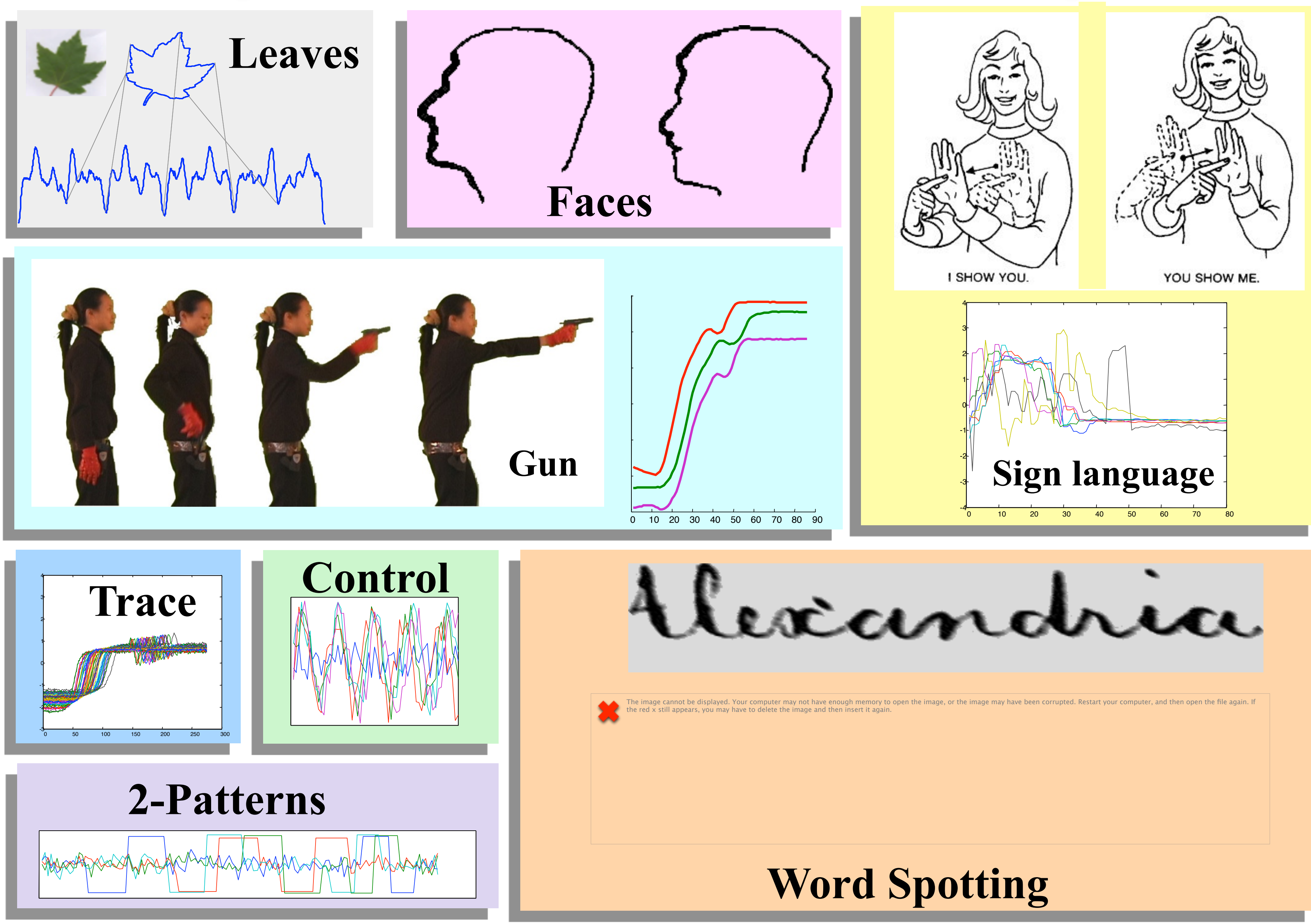
Here is another example on nuclear power plant trace data, to help you develop an intuition for DTW

Nuclear Power Excellent!



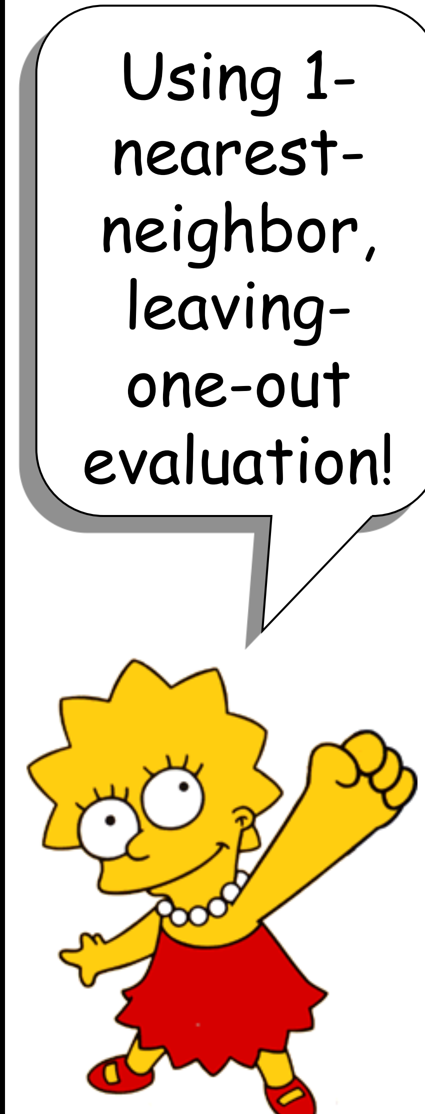


Let us compare Euclidean Distance and DTW on some problems



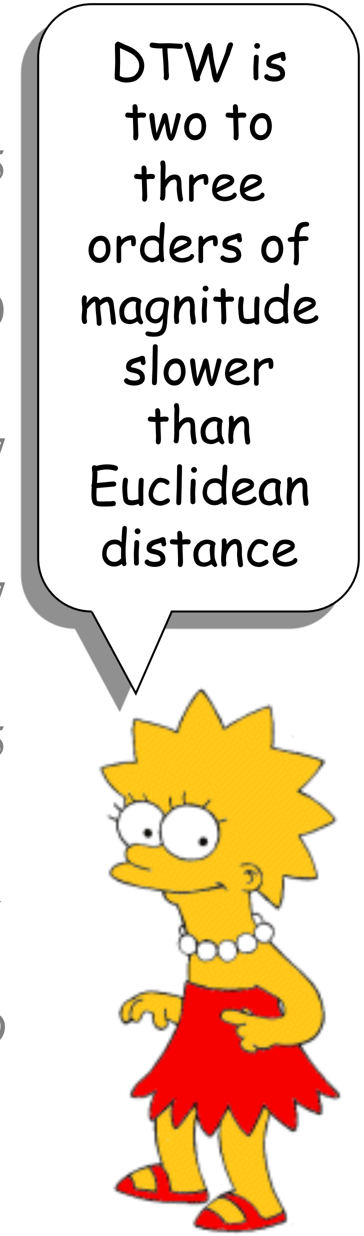
Results: Error Rate

Dataset	Euclidean	DTW
Word Spotting	4.78	1.10
Sign language	28.70	25.93
GUN	5.50	1.00
Nuclear Trace	11.00	0.00
Leaves#	33.26	4.07
(4) Faces	6.25	2.68
Control Chart*	7.5	0.33
2-Patterns	1.04	0.00



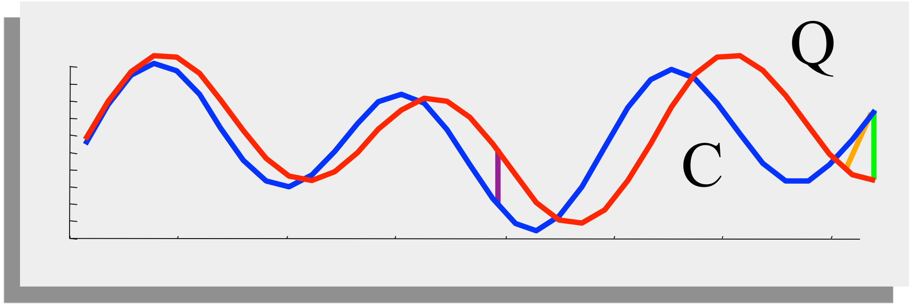
Results: Time (**msec**)

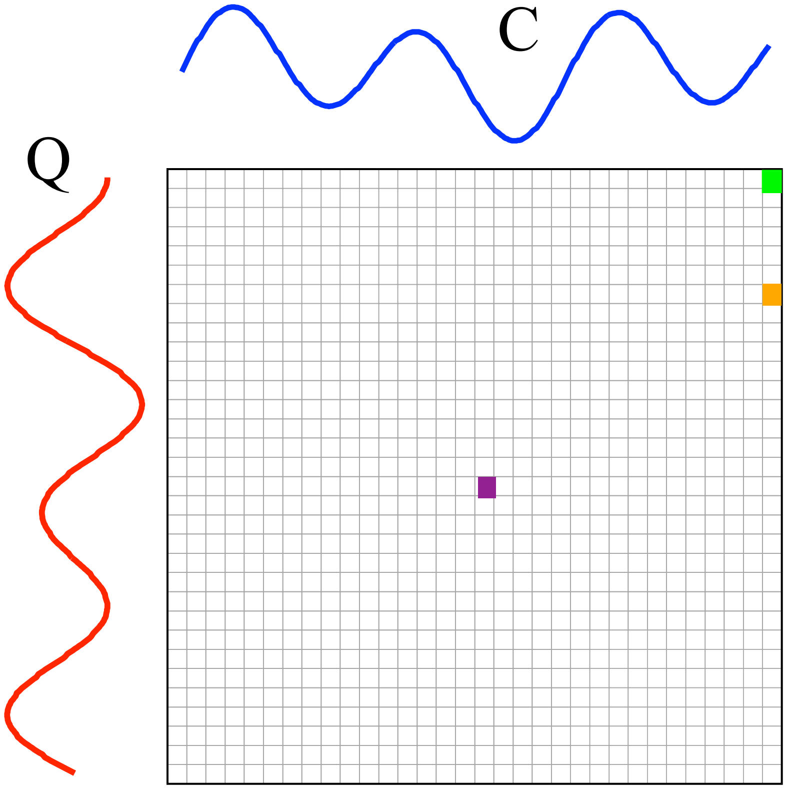
Dataset	Euclidean	DTW	
Word Spotting	40	8,600	215
Sign language	10	1,110	110
GUN	60	11,820	197
Nuclear Trace	210	144,470	687
Leaves	150	51,830	345
(4) Faces	50	45,080	901
Control Chart	110	21,900	199
2-Patterns	16,890	545,123	32



How is DTW Calculated? I

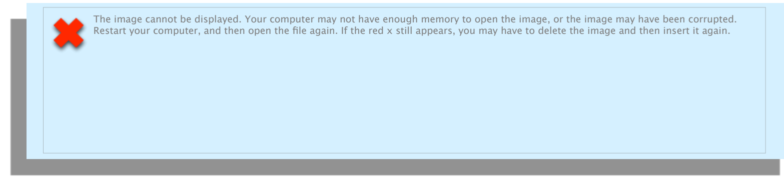
We create a matrix the size of |Q| by |C|, then fill it in with the distance between every pair of point in our two time series.

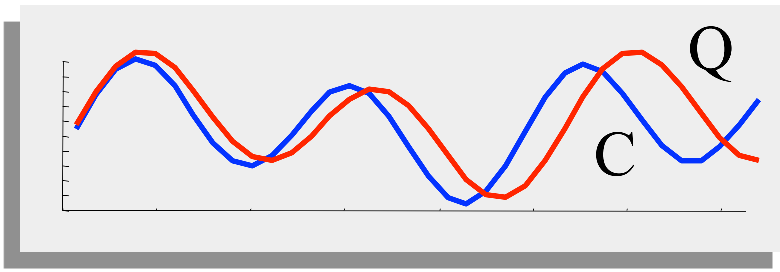


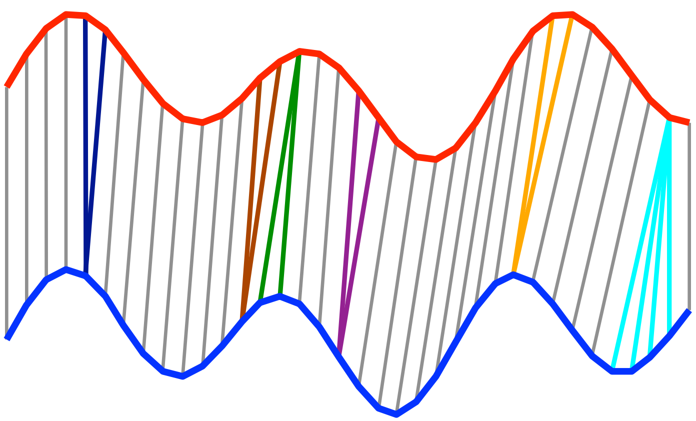


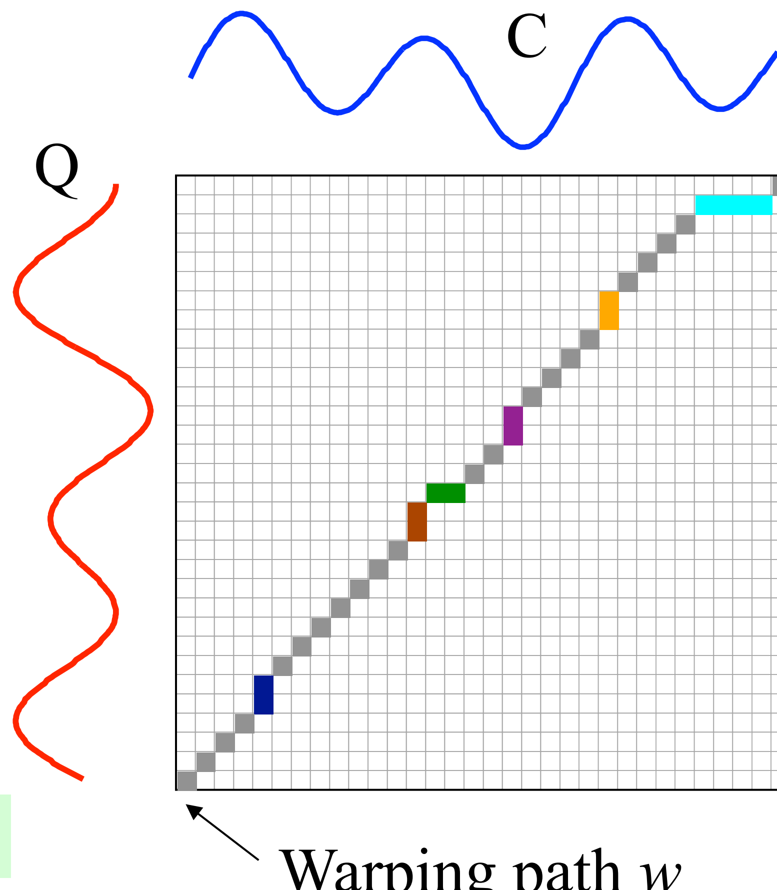
How is DTW Calculated? II

Every possible warping between two time series, is a path though the matrix. We want the best one…





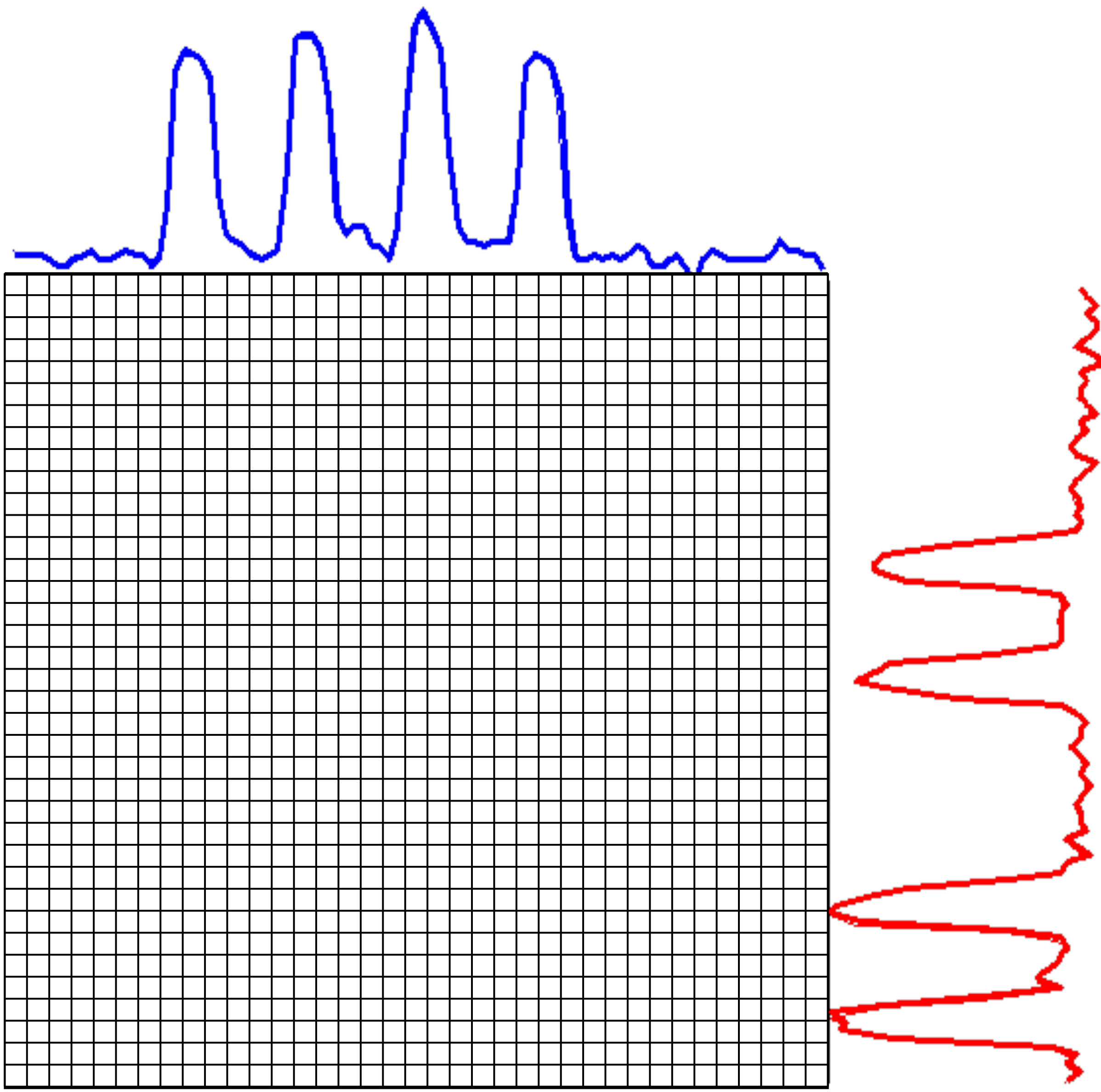


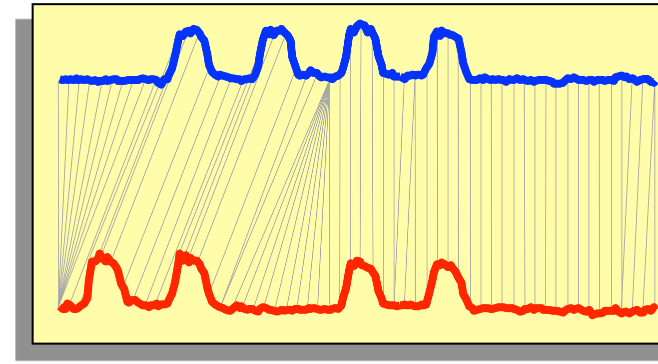


This recursive function gives us the minimum cost path

$$\gamma(i,j) = d(q_i,c_j) + \min\{ \gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1) \}$$

Let us visualize the cumulative matrix on a real world problem I

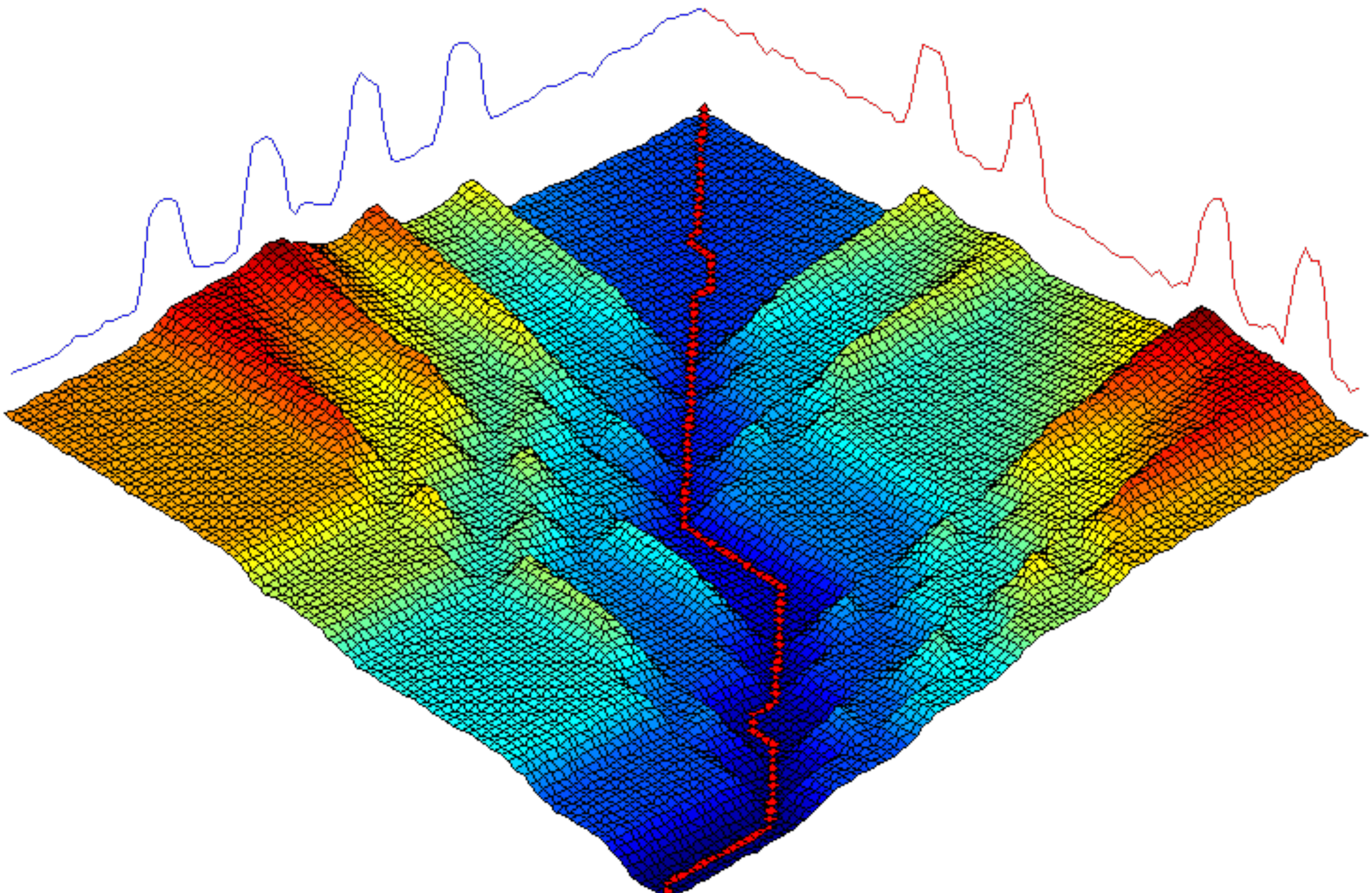




This example shows 2 one-week periods from the power demand time series.

Note that although they both describe 4-day work weeks, the blue sequence had Monday as a holiday, and the red sequence had Wednesday as a holiday.

Let us visualize the cumulative matrix on a real world problem II



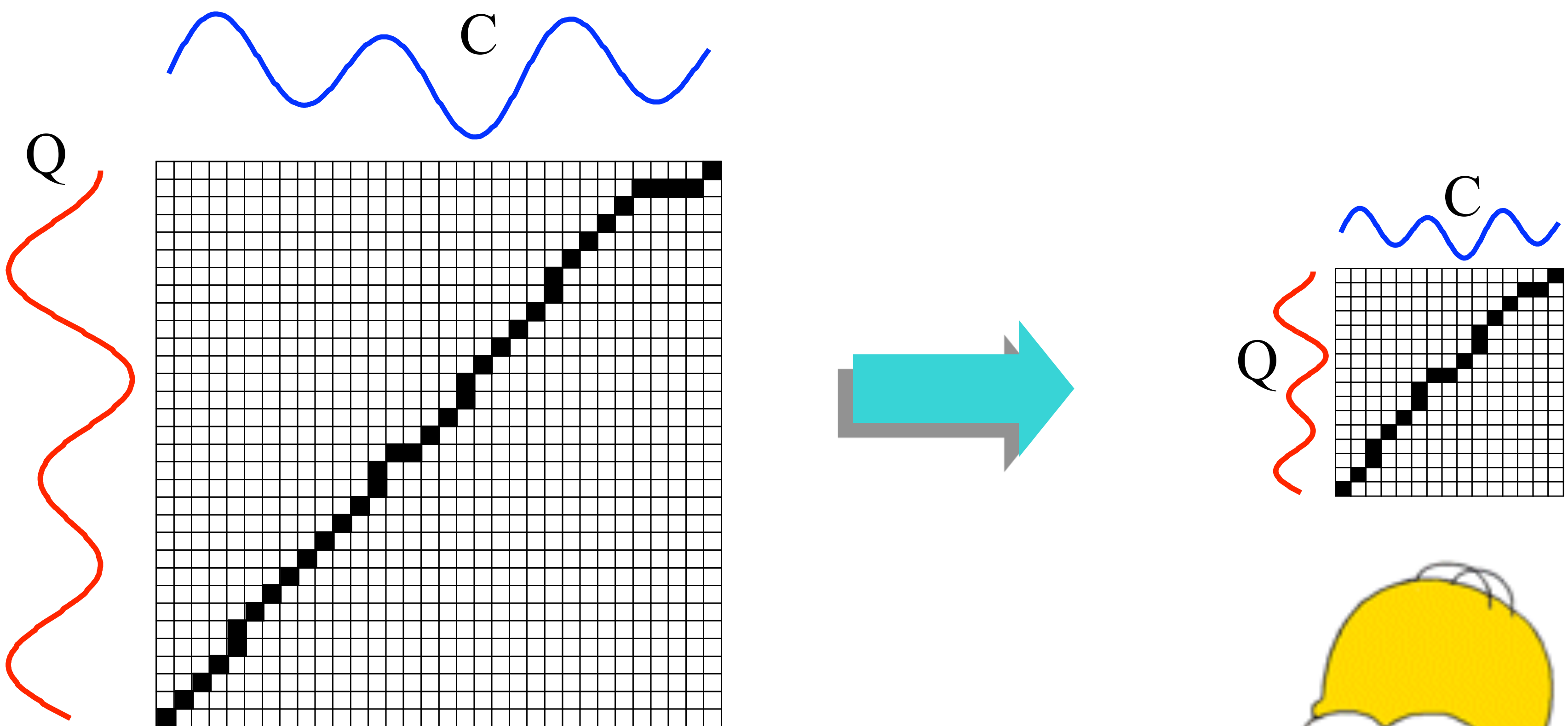
What we have seen so far…

- Dynamic Time Warping gives **much better** results than Euclidean distance on virtually all problems.

- Dynamic Time Warping is very very slow to calculate!

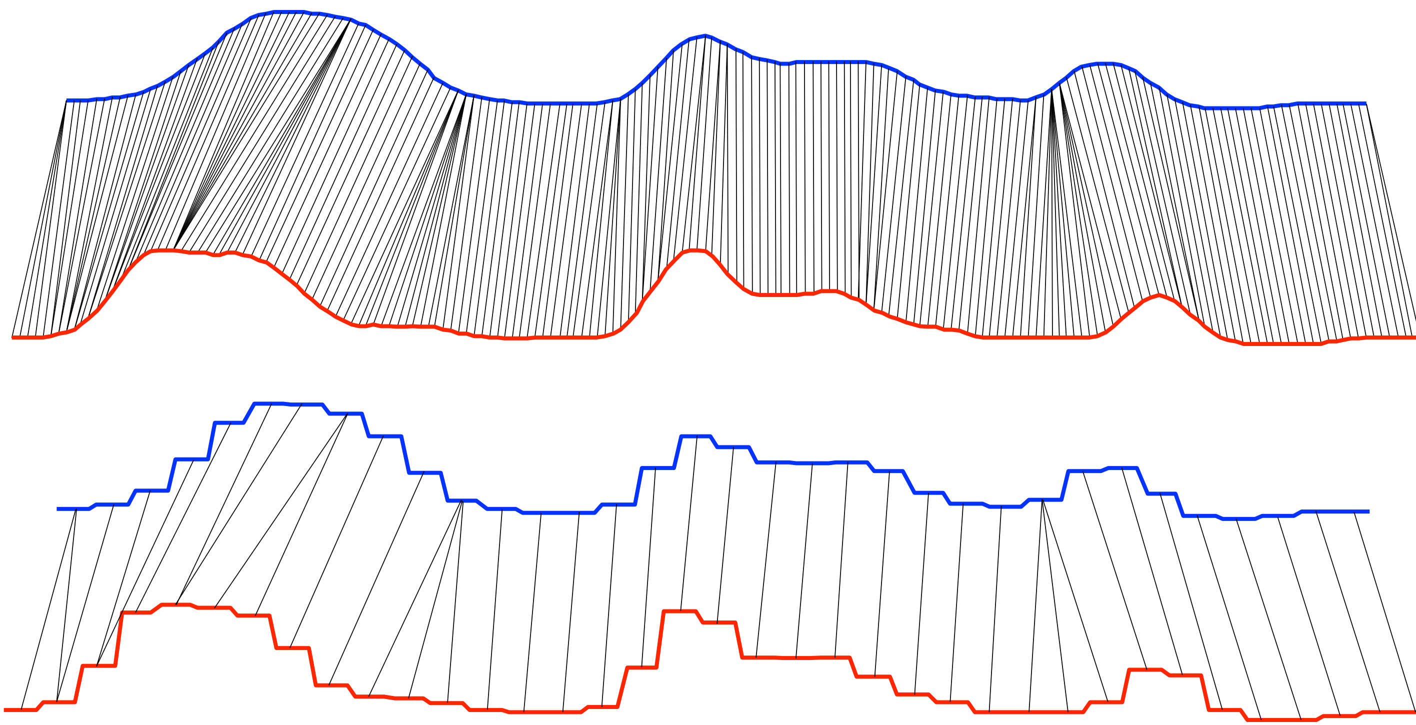
Is there anything we can do to speed up similarity search under DTW?

Fast Approximations to Dynamic Time Warp Distance I



Simple Idea: Approximate the time series with some compressed or downsampled representation, and do DTW on the new representation. How well does this work...

Fast Approximations to Dynamic Time Warp Distance II

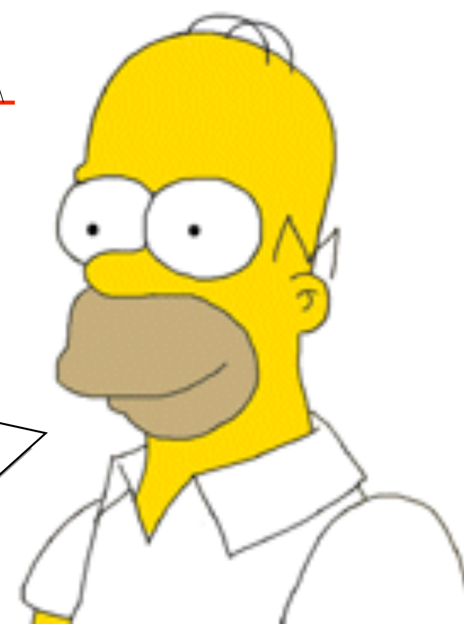


1.03 sec

0.07 sec

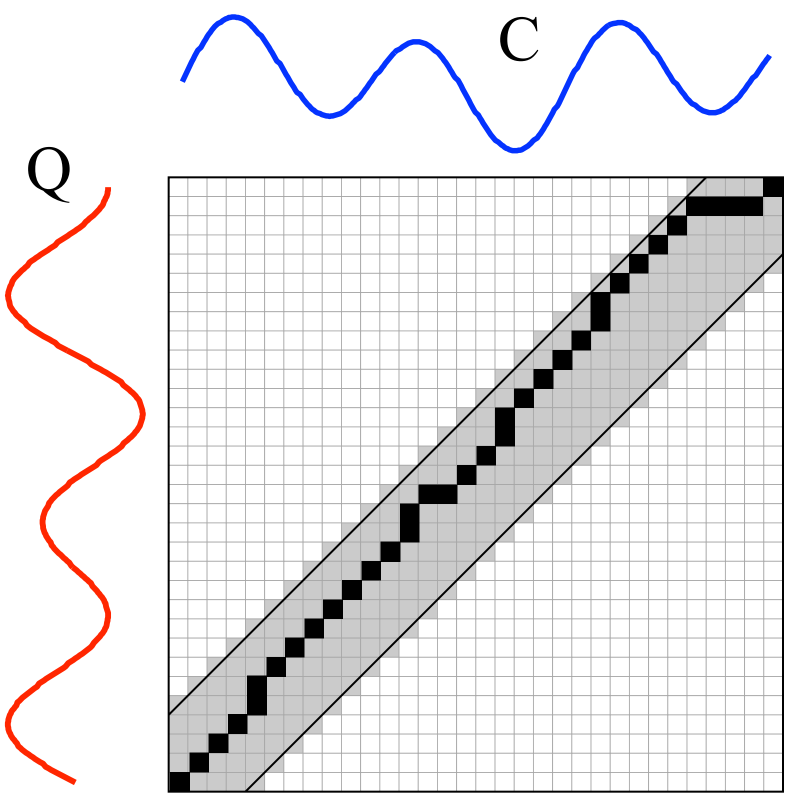
… there is strong visual evidence to suggests it works well

There is good experimental evidence for the utility of the approach on clustering, classification, etc

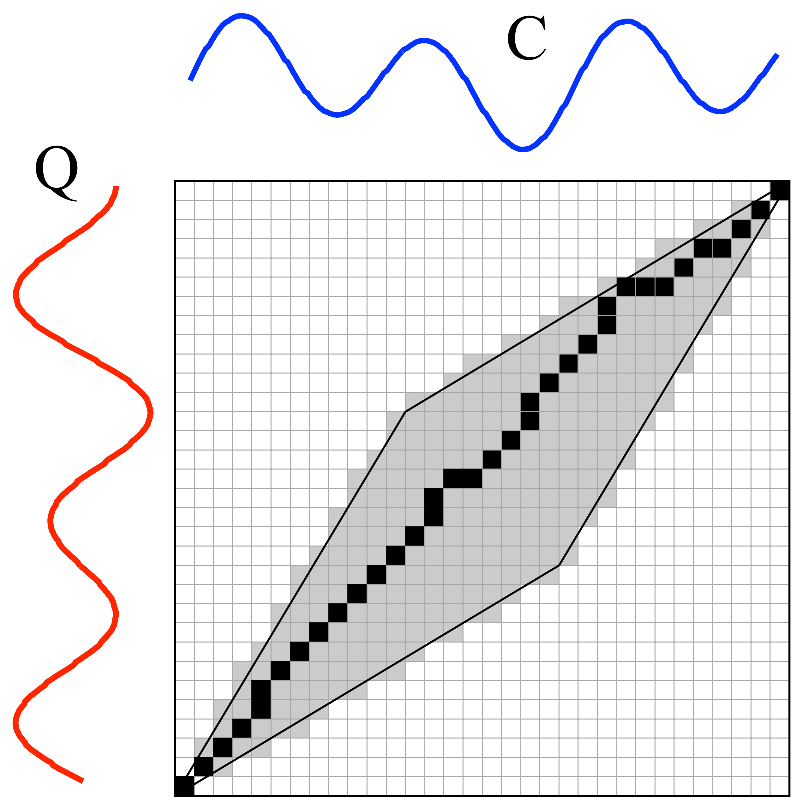


Global Constraints

- Slightly speed up the calculations
- Prevent pathological warpings

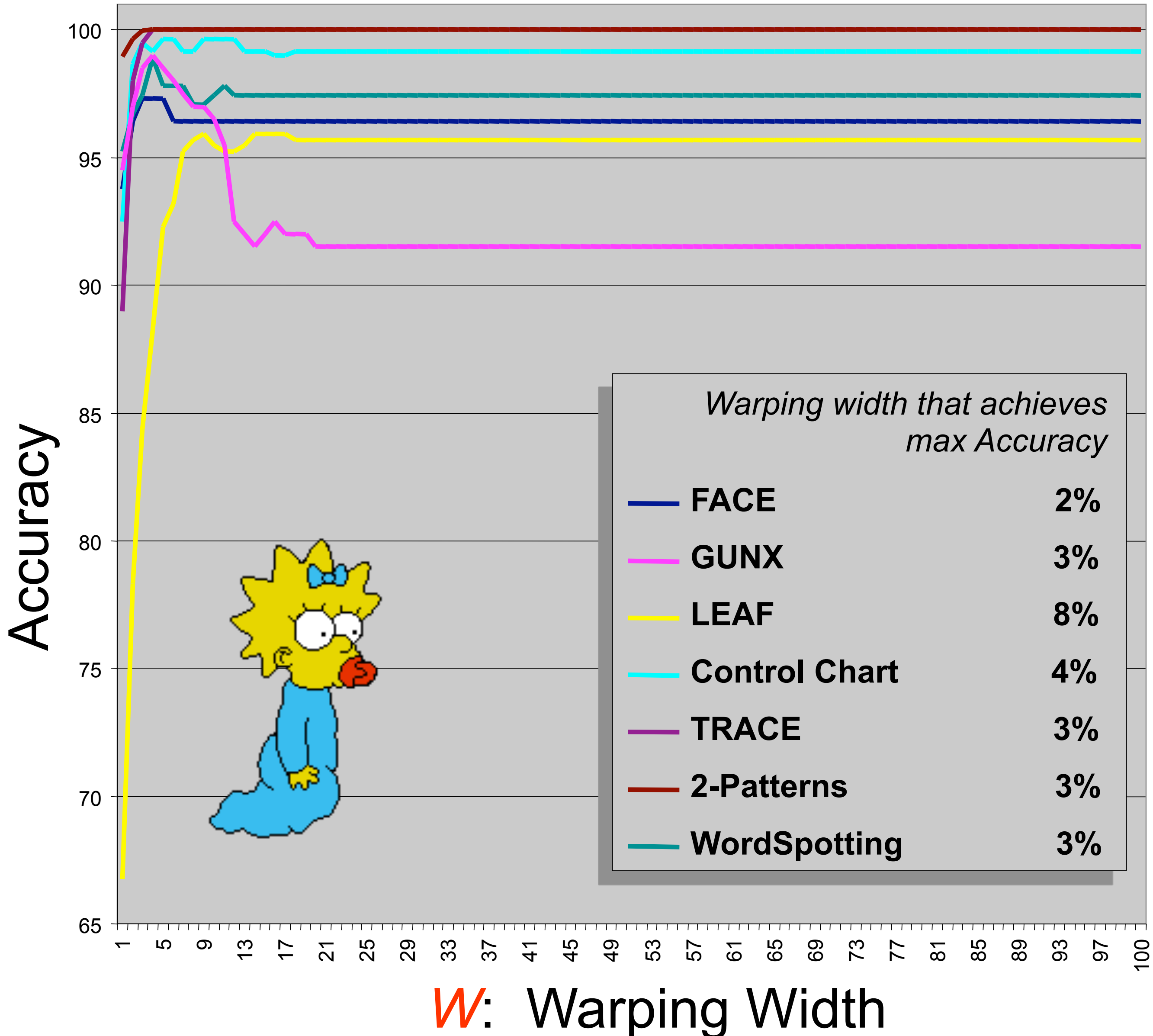


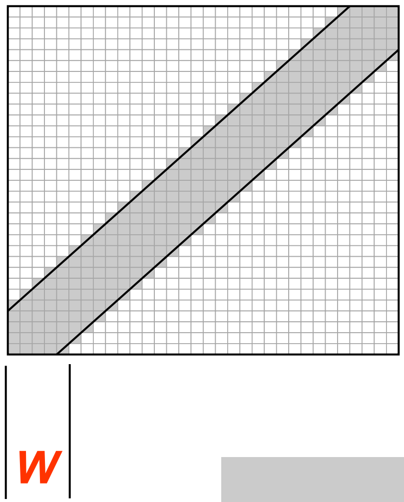
Sakoe-Chiba Band



Itakura Parallelogram

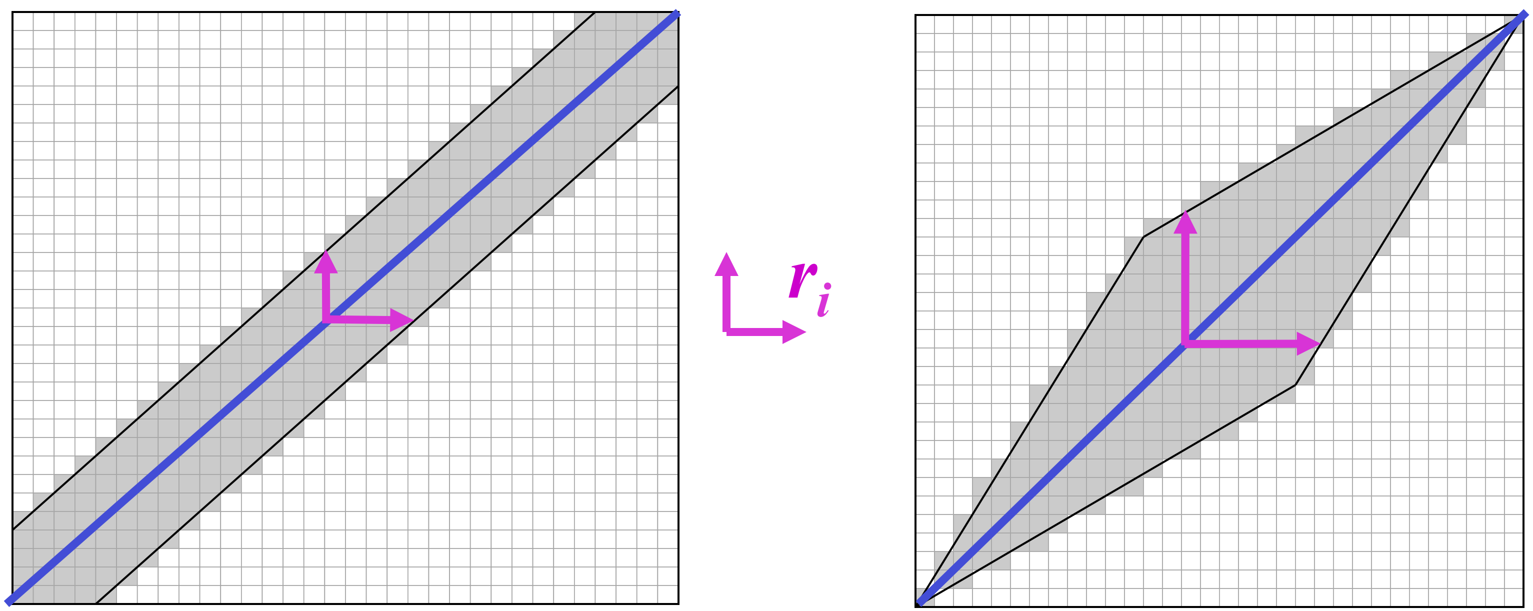
Accuracy vs. Width of Warping Window





A global constraint constrains the indices of the warping path $w_k = (i,j)_k$ such that $j-r \leq i \leq j+r$

Where r is a term defining allowed range of warping for a given point in a sequence.



Sakoe-Chiba Band

Itakura Parallelogram

In general, it's hard to speed up a single DTW calculation

However, if we have to make many DTW calculations (which is almost always the case), we can potentiality speed up the whole process by *lowerbounding*.

Keep in mind that the *lowerbounding* trick works for any situation were you have an expensive calculation that can be *lowerbounded* (string edit distance, graph edit distance etc)

I will explain how *lowerbounding* works in a generic fashion in the next two slides, then show concretely how *lowerbounding* makes dealing with massive time series under DTW possible…

Lower Bounding I

Assume that we have two functions:

- DTW(A,B)
- lower_bound_distance(A,B)

The true DTW function is very slow…

The *lower bound* function is very fast…

By definition, for all A, B, we have

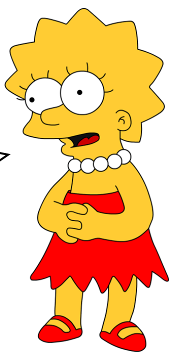
lower_bound_distance(A,B) ≤ DTW(A,B)

Lower Bounding II

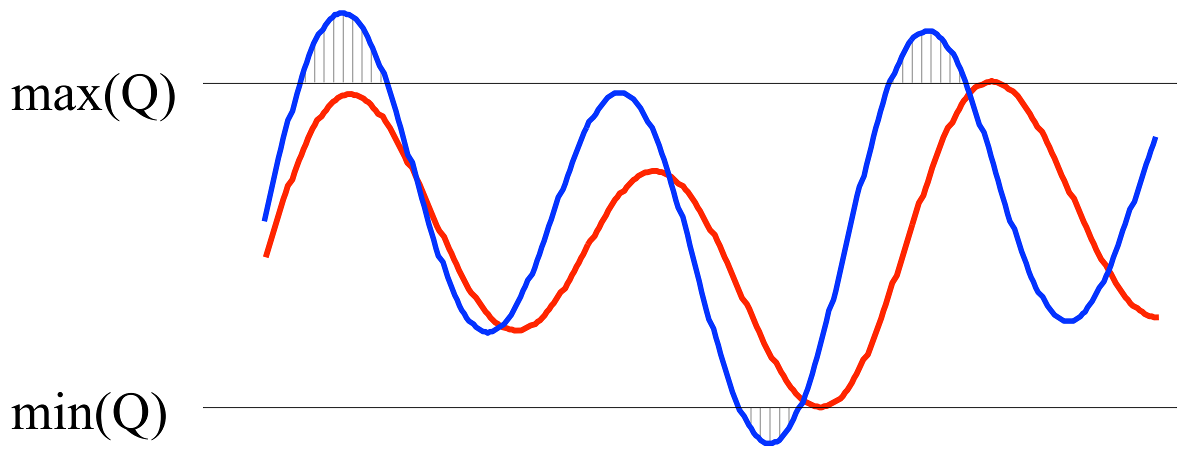
We can speed up similarity search under DTW by using a lower bounding function

Algorithm Lower_Bounding_Sequential_Scan(Q)

```
best_so_far = infinity;
for all sequences in database
    LB_dist = lower_bound_distance(C_i, Q);
    if LB_dist < best_so_far
        true_dist = DTW(C_i, Q);
        if true_dist < best_so_far
            best_so_far = true_dist;
            index_of_best_match = i;
        endif
    endif
endfor
```



Lower Bound of Yi

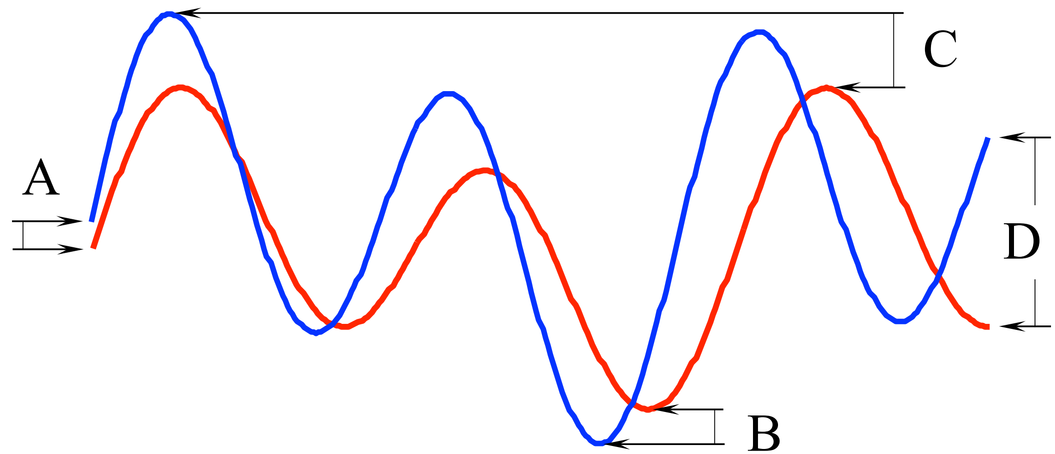


LB_Yi

Yi, B, Jagadish, H & Faloutsos, C. *Efficient retrieval of similar time sequences under time warping*. ICDE 98, pp 23-27.

The sum of the squared length of gray lines represent the minimum the corresponding points contribution to the overall DTW distance, and thus can be returned as the lower bounding measure

Lower Bound of Kim

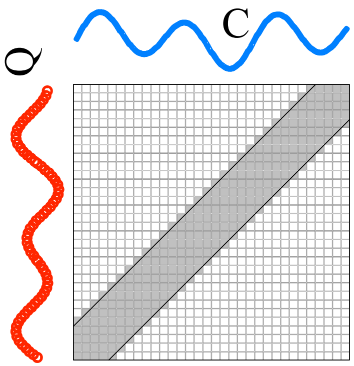


LB_Kim

Kim, S, Park, S, & Chu, W. *An index-based approach for similarity search supporting time warping in large sequence databases*. ICDE 01, pp 607-614

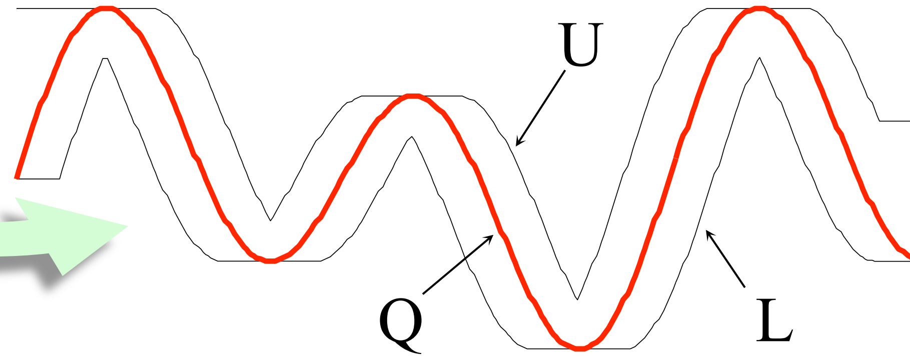
The squared difference between the two sequence's first (A), last (D), minimum (B) and maximum points (C) is returned as the lower bound

Lower Bound of Keogh



Sakoe-Chiba Band

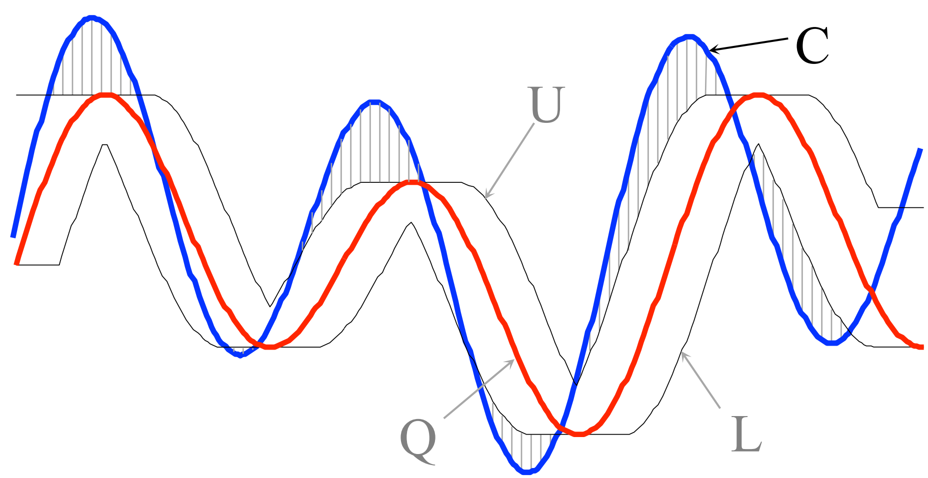
$U_i = \max(q_{i-r} : q_{i+r})$
$L_i = \min(q_{i-r} : q_{i+r})$



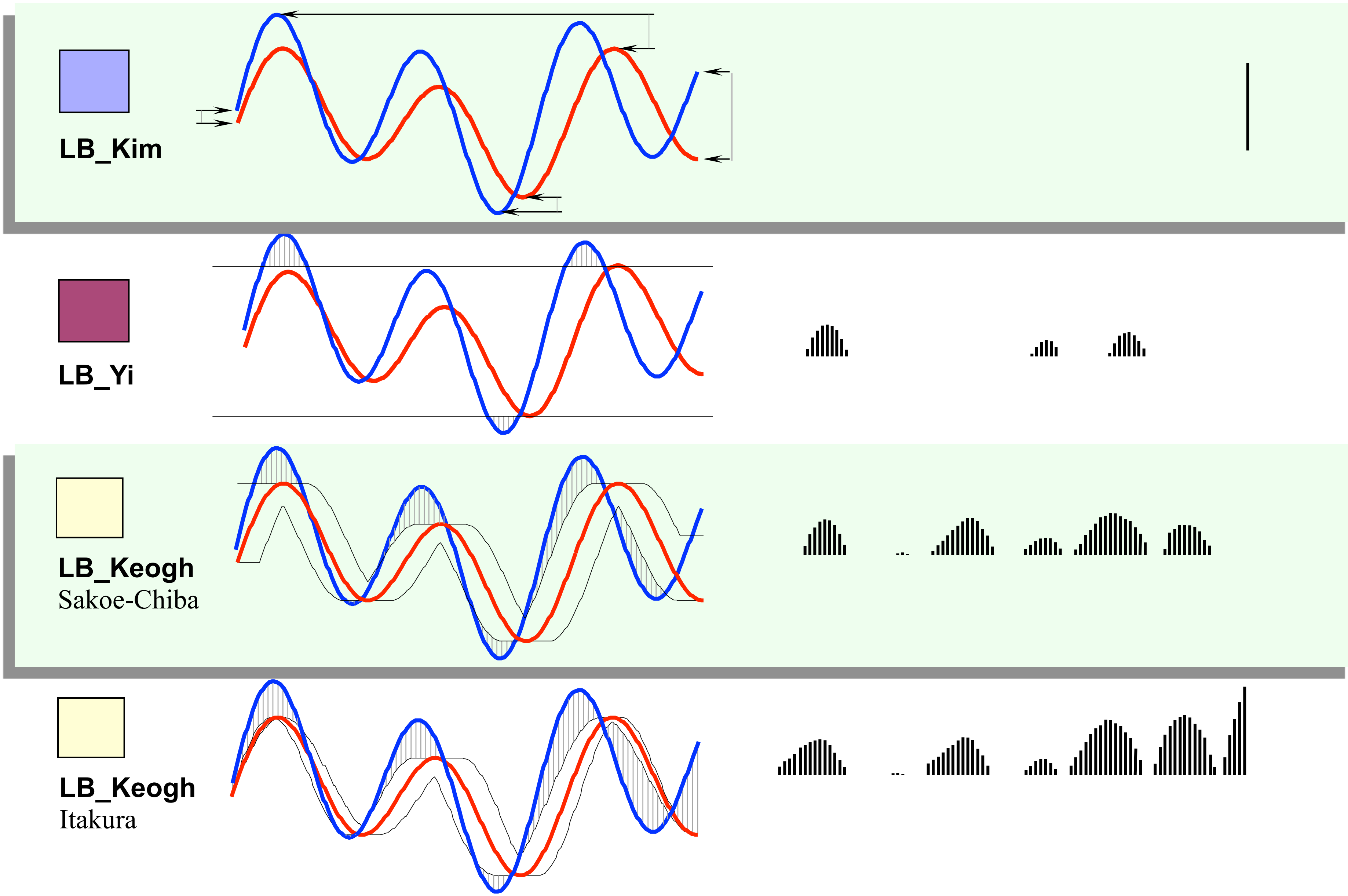
LB_Keogh

Envelope-Based
Lower Bound

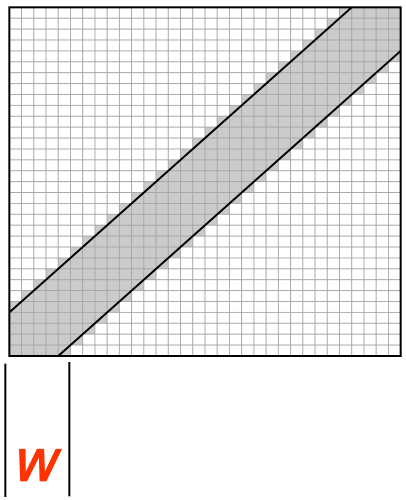
$$LB\_Keogh(Q,C) = \sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2 & if\ q_i > U_i \\ (q_i - L_i)^2 & if\ q_i < L_i \\ 0 & otherwise \end{cases}$$



The tightness of the lower bound for each technique is proportional to the length of gray lines used in the illustrations



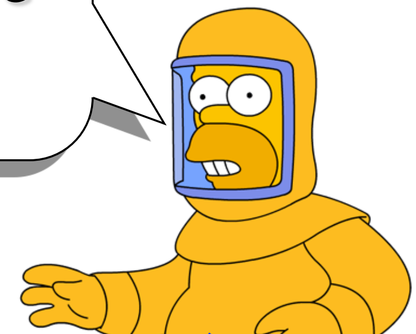
How Useful are Lower Bounds?

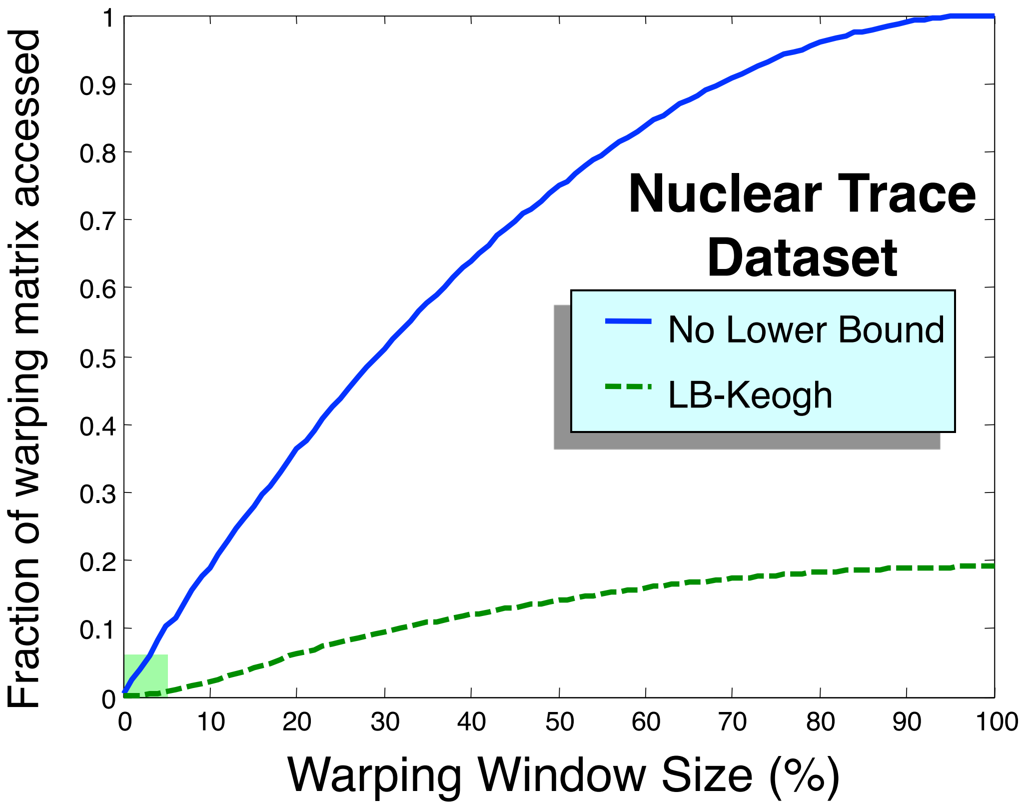


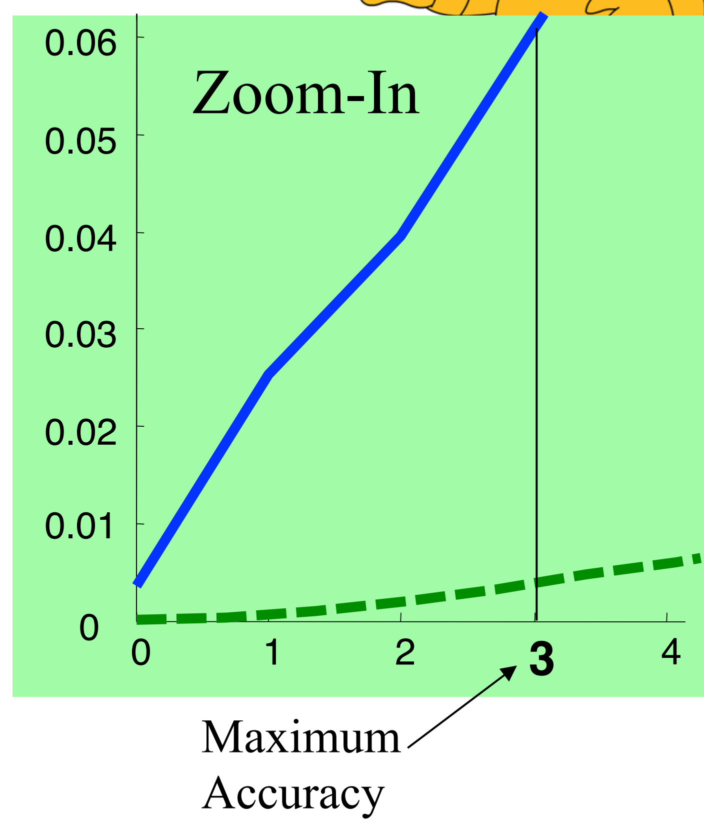
Lets do some experiments!

We will measure the average fraction of the n^2 matrix that we must calculate, for a one nearest neighbor search.
We will do this for every possible value of *w*, the warping window width.
By testing this way, we are deliberately ignoring implementation details, like index structure, buffer size etc…

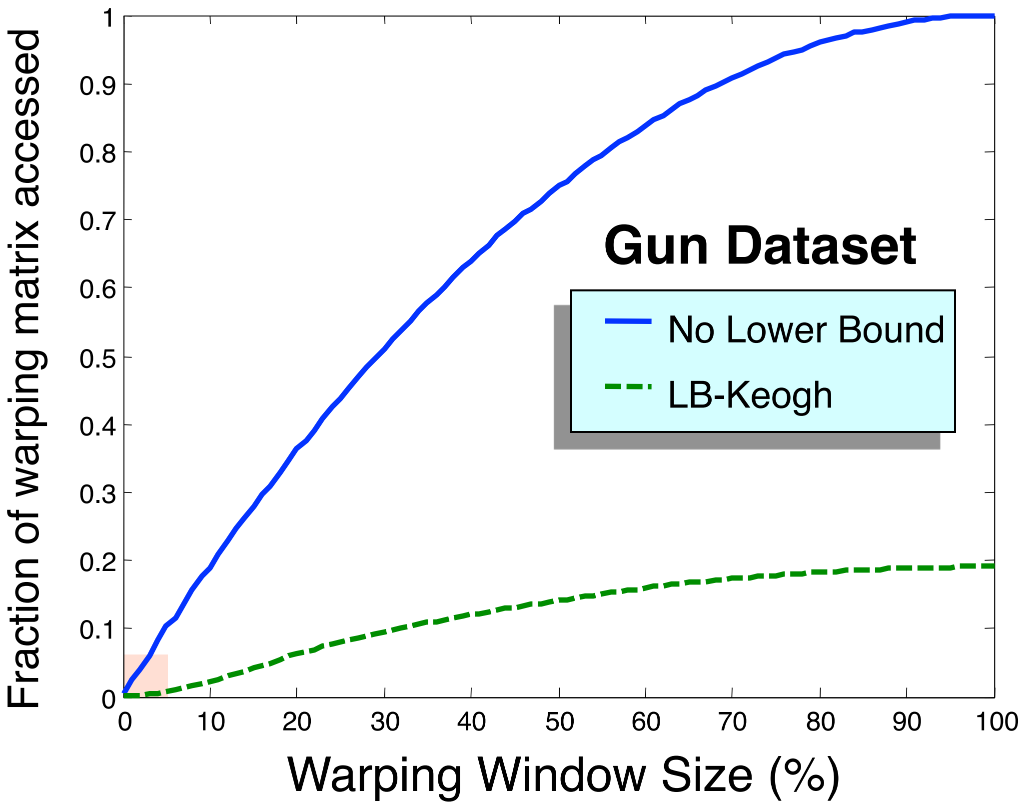
This plot tells us that although DTW is $O(n^2)$, after we set the warping window for maximum accuracy for this problem, we only have to do 6% of the work, and if we use the LB_Keogh lower bound, we only have to do **0.3%** of the work!

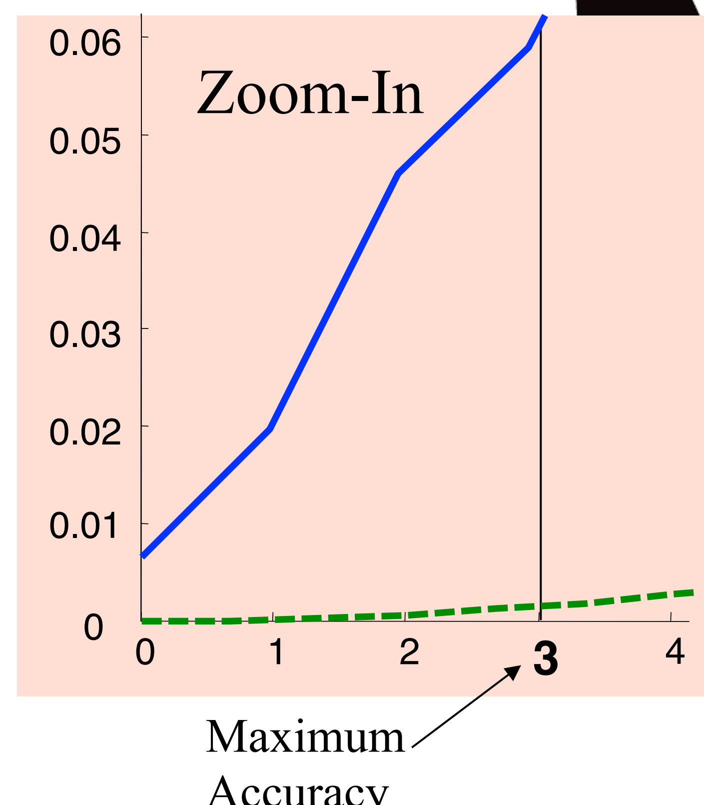


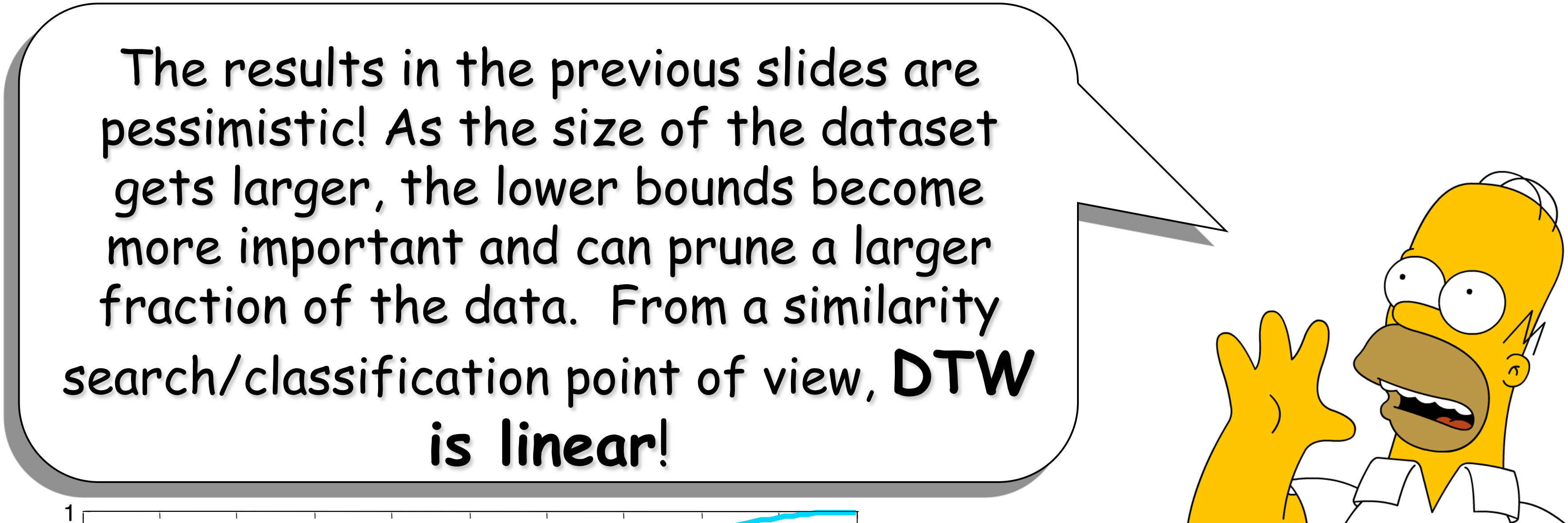


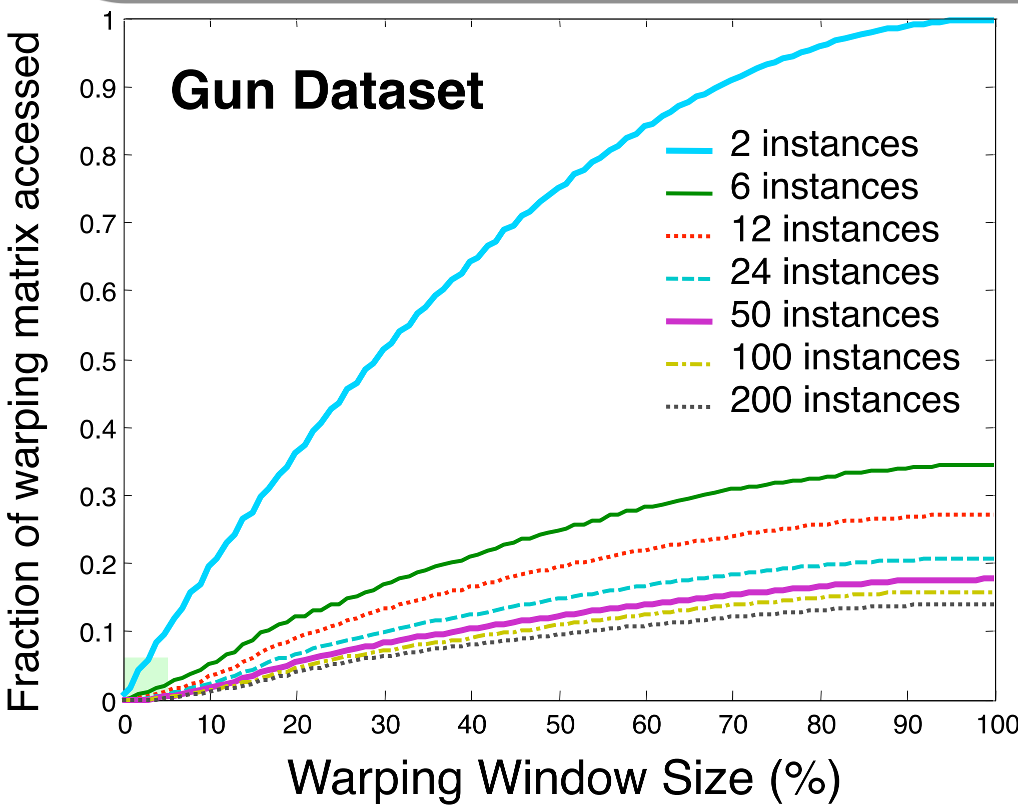


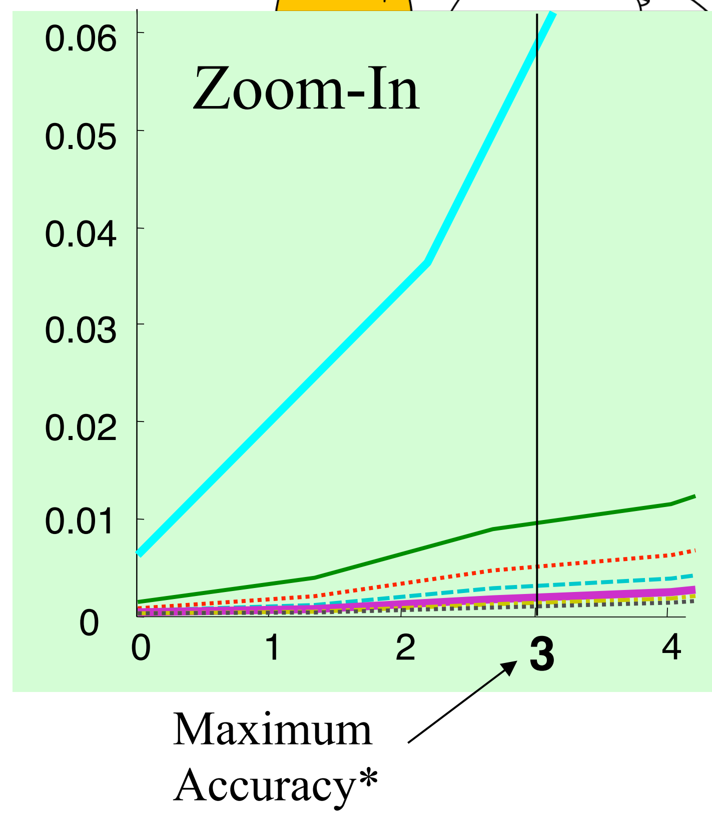
This plot tells us that although DTW is $O(n^2)$, after we set the warping window for maximum accuracy for this problem, we only have to do 6% of the work, and if we use the LB_Keogh lower bound, we only have to do **0.21%** of the work!






The results in the previous slides are pessimistic! As the size of the dataset gets larger, the lower bounds become more important and can prune a larger fraction of the data. From a similarity search/classification point of view, **DTW is linear**!


Gun Dataset
Fraction of warping matrix accessed
Warping Window Size (%)
2 instances
6 instances
12 instances
24 instances
50 instances
100 instances
200 instances
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100


Zoom-In
0
0.01
0.02
0.03
0.04
0.05
0.06
0
1
2
3
4
Maximum Accuracy*

…DTW is linear for data mining problems!

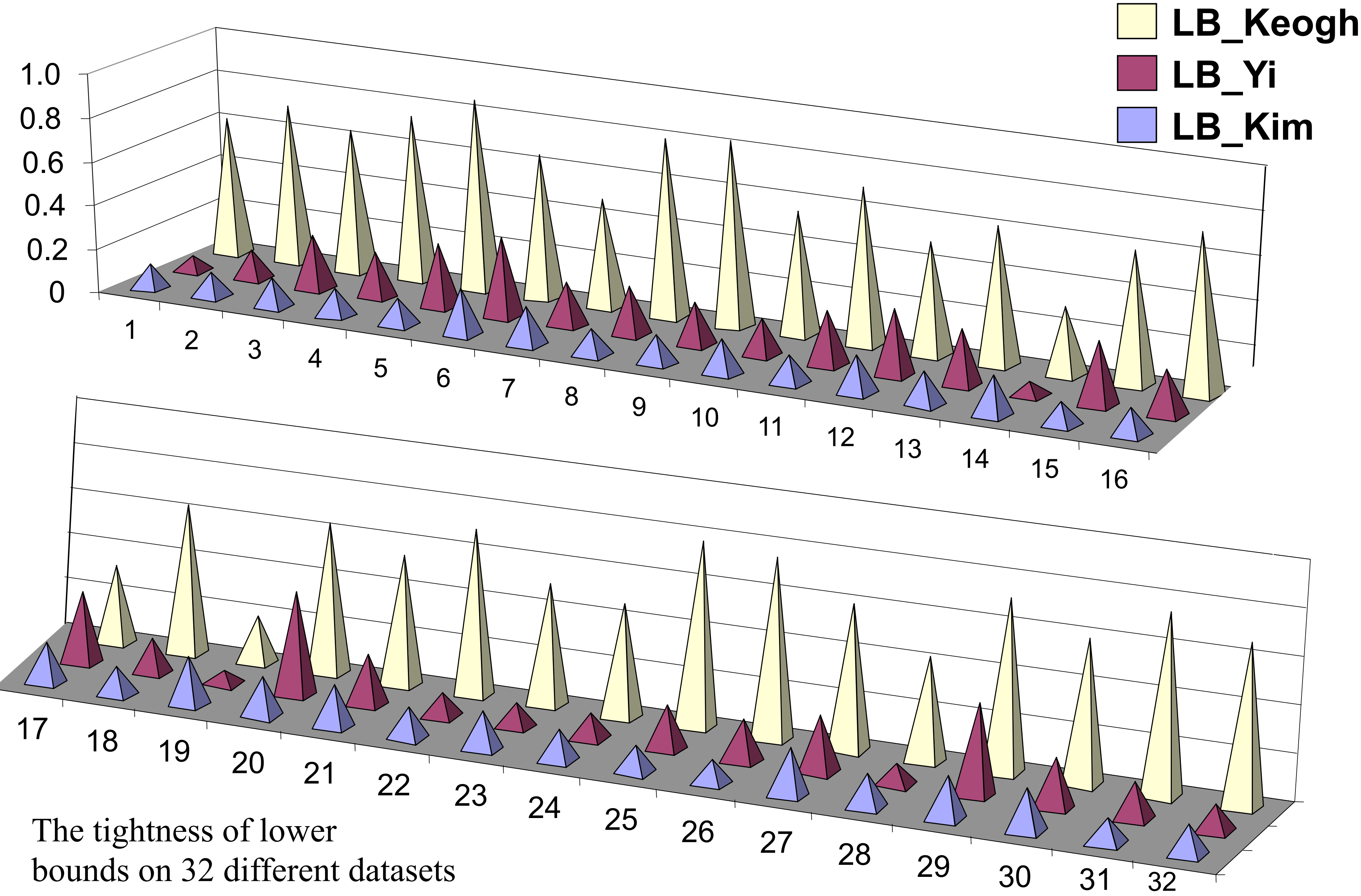
Papers published in the last year suggest…

- *"DTW incurs a heavy CPU cost"*[1]
- *"DTW is limited to only small time series datasets"*[2]
- *"(DTW) quadratic cost makes its application on databases of long time series very expensive"*[3]
- *"(DTW suffers from) serious performance degradation in large databases"*[4]

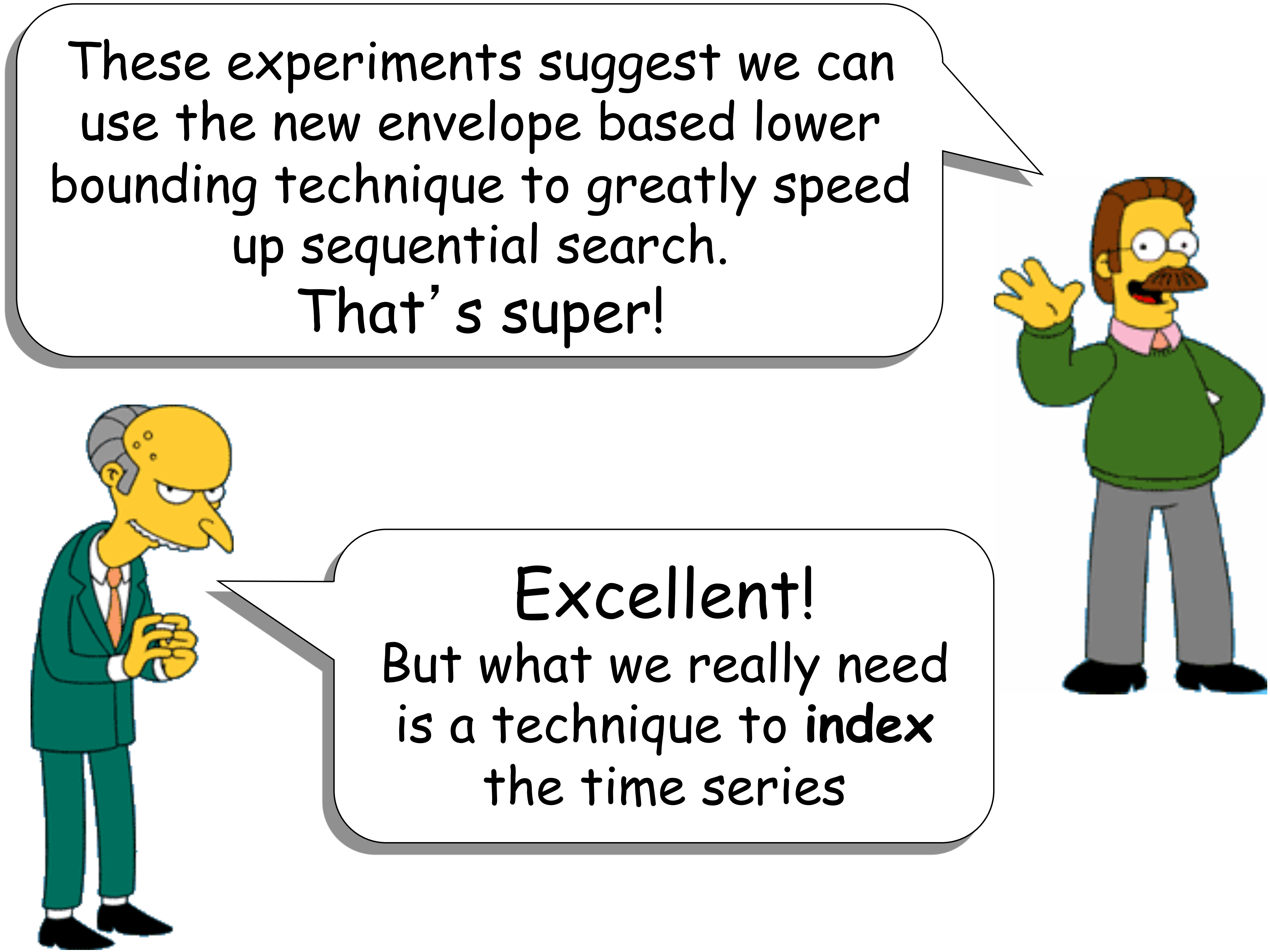
This is simply not true!

Why did the previous slides consider only one type of lower bound?

$$T = \frac{\text{Lower Bound Estimate of Dynamic Time Warp Distance}}{\text{True Dynamic Time Warp Distance}}$$



The tightness of lower bounds on 32 different datasets

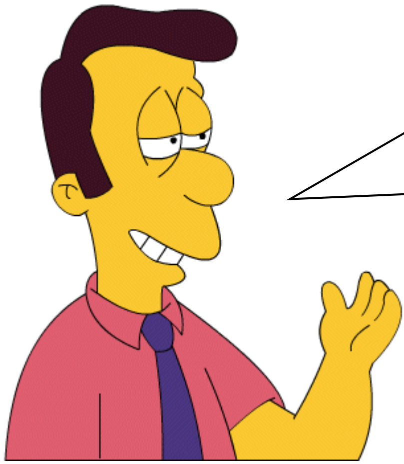

These experiments suggest we can use the new envelope based lower bounding technique to greatly speed up sequential search.
That's super!
Excellent!
But what we really need is a technique to **index** the time series

According to the most referenced paper on time series similarity searching "*dynamic time warping cannot be speeded up by indexing* *",

As we noted in an earlier slide, virtually all indexing techniques require the triangular inequality to hold.

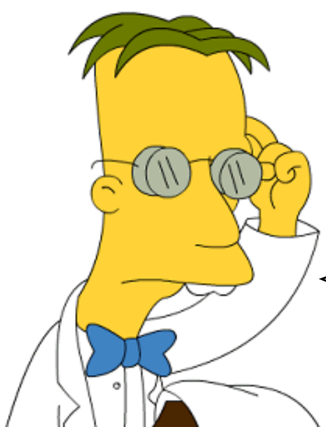
DTW does NOT obey the triangular inequality!

* Agrawal, R., Lin, K. I., Sawhney, H. S., & Shim, K. (1995). Fast similarity search in the presence of noise, scaling, and translation in times-series databases. VLDB pp. 490-501.



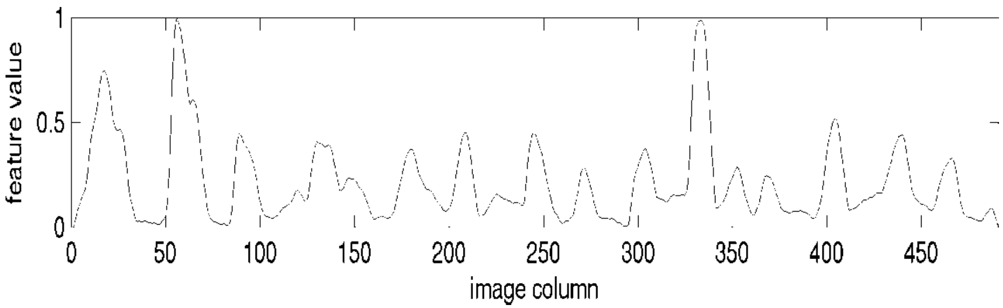
In fact, it has been was shown that DTW can be indexed! (VLDB02)

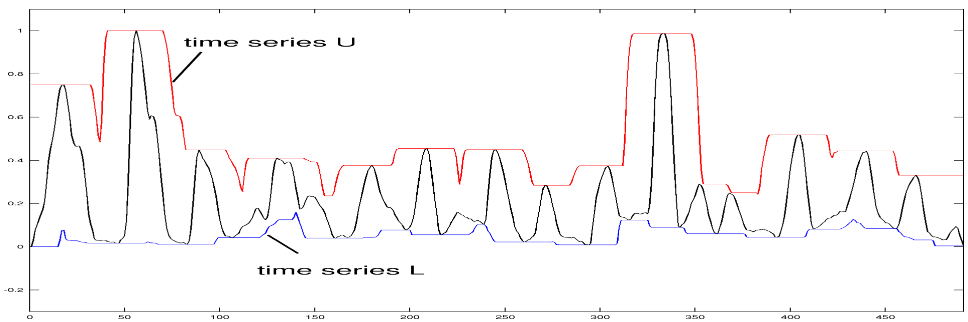
We won't give details here, other than to note that the technique is based on the envelope lower bounding technique we have just seen



Let us quickly see some success stories, problems that we now solve, given that we can *index* DTW

Success Story I





The lower bounding technique has been used to support indexing of massive archives of handwritten text.

Surprisingly, DTW works better on this problem that more sophisticated approaches like Markov models

R. Manmatha, T. M. Rath: Indexing of Handwritten Historical Documents - Recent Progress. In: Proc. of the 2003 Symposium on Document Image Understanding Technology (SDIUT), Greenbelt, MD, April 9-11, 2003, pp. 77-85.

T. M. Rath and R. Manmatha (2002): Lower-Bounding of Dynamic Time Warping Distances for Multivariate Time Series. Technical Report MM-40, Center for Intelligent Information Retrieval, University of Massachusetts Amherst.

Success Story II

The lower bounding technique has been used to support "*query by humming*", by several groups of researchers

Best 3 Matches

1) Bee Gees: Grease
2) Robbie Williams: Grease
3) Sarah Black: Heatwave

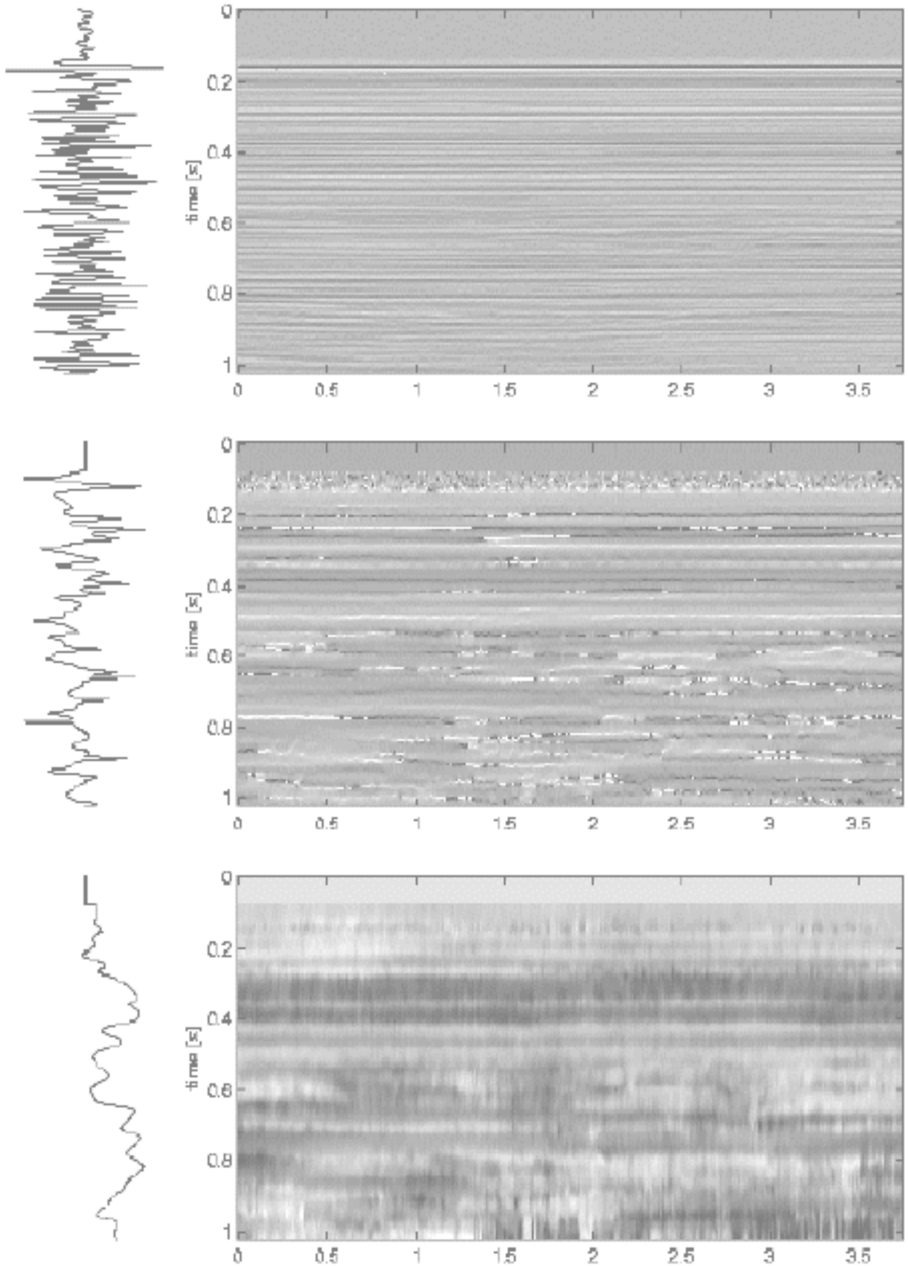
Ning Hu, Roger B. Dannenberg (2003). Polyphonic Audio Matching and Alignment for Music Retrieval

Yunyue Zhu, Dennis Shasha (2003). Query by Humming: a Time Series Database Approach, SIGMOD

Success Story III

The lower bounding technique is being used for indexing motion capture data.

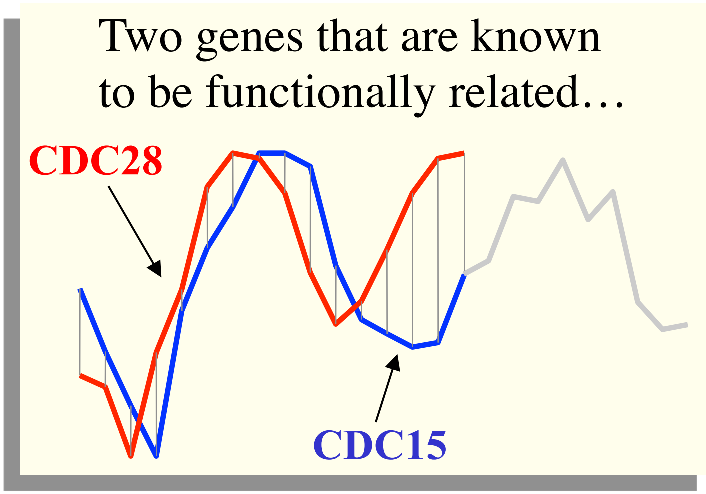
Thanks to Marc Cardle for this example



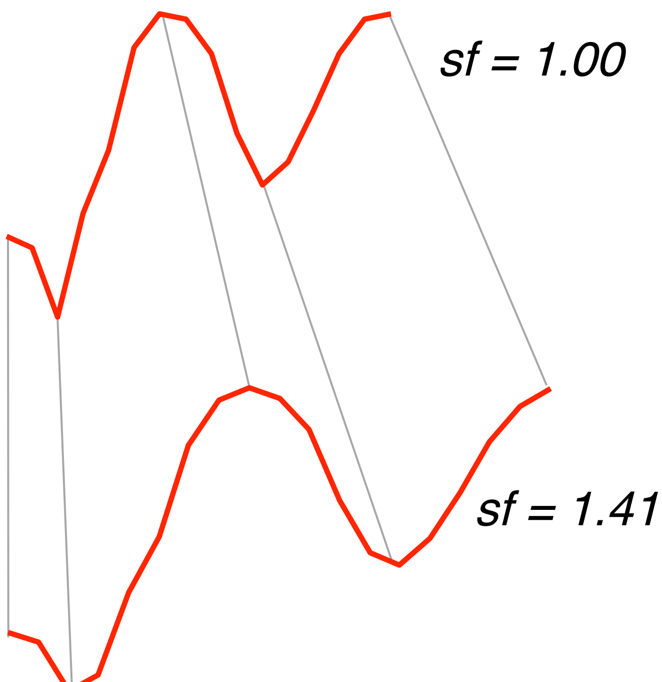
Success Story IIII

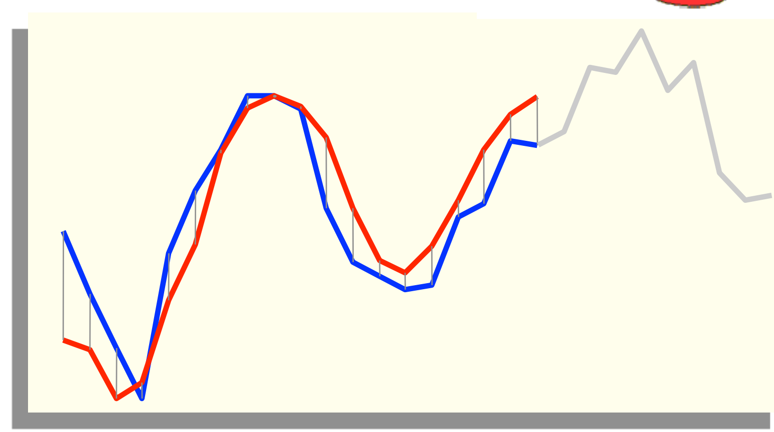
The lower bounding technique is being used by ChevronTexaco for comparing seismic data

Uniform Scaling I

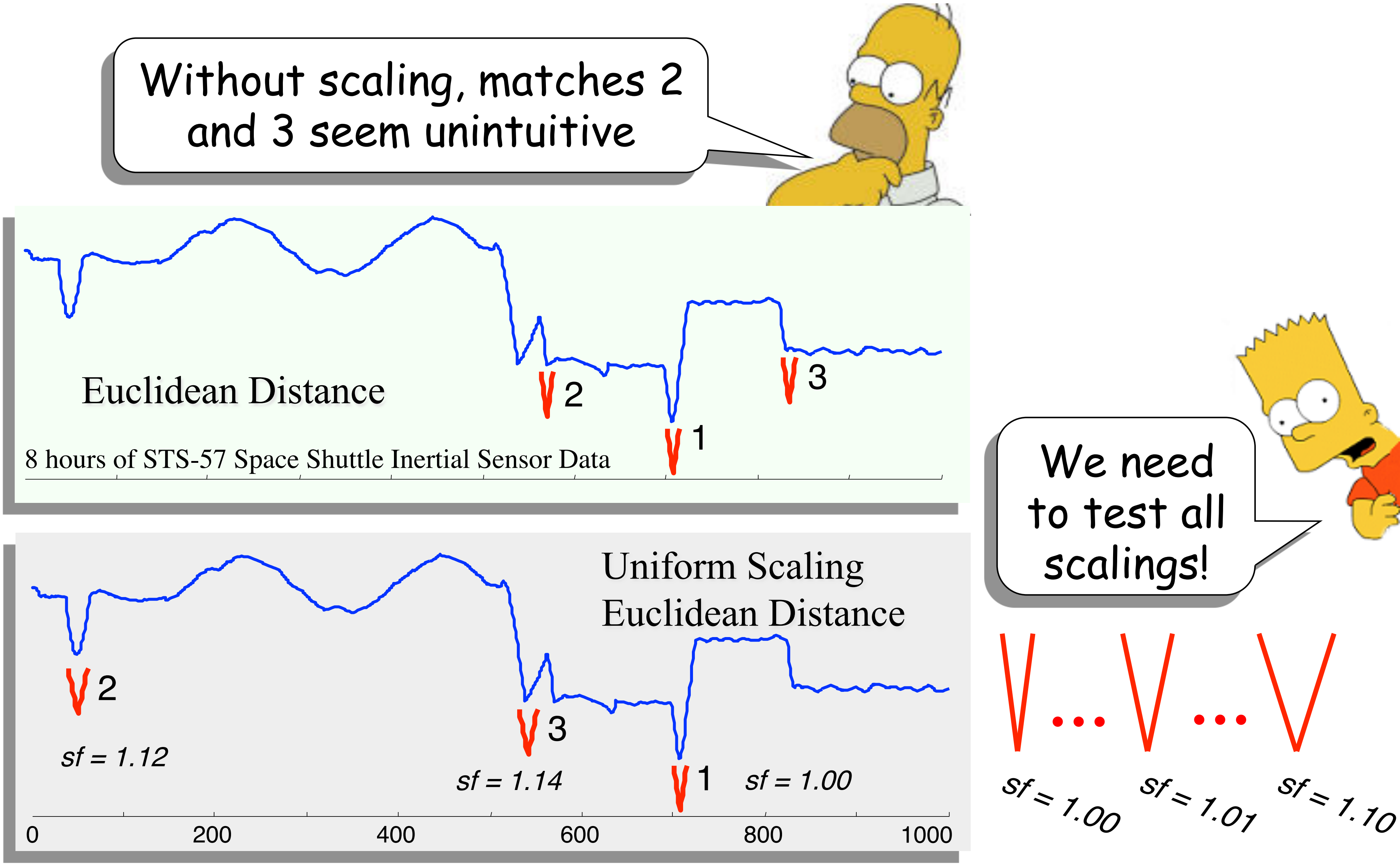


Sometimes global or *uniform scaling* is as important as DTW



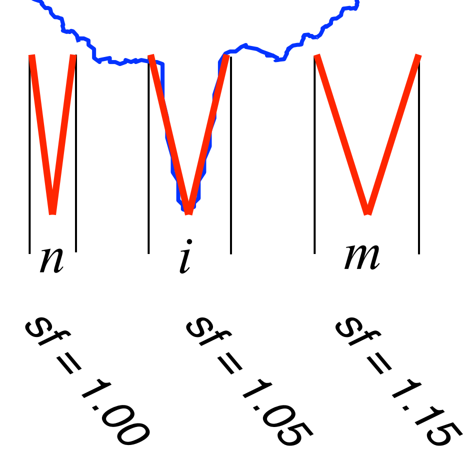


Uniform Scaling II



Here is some notation, the shortest scaling we consider is length *n*, and the largest is length *m*.

The *scaling factor (sf)* is the ratio i/n , n <= i <= m



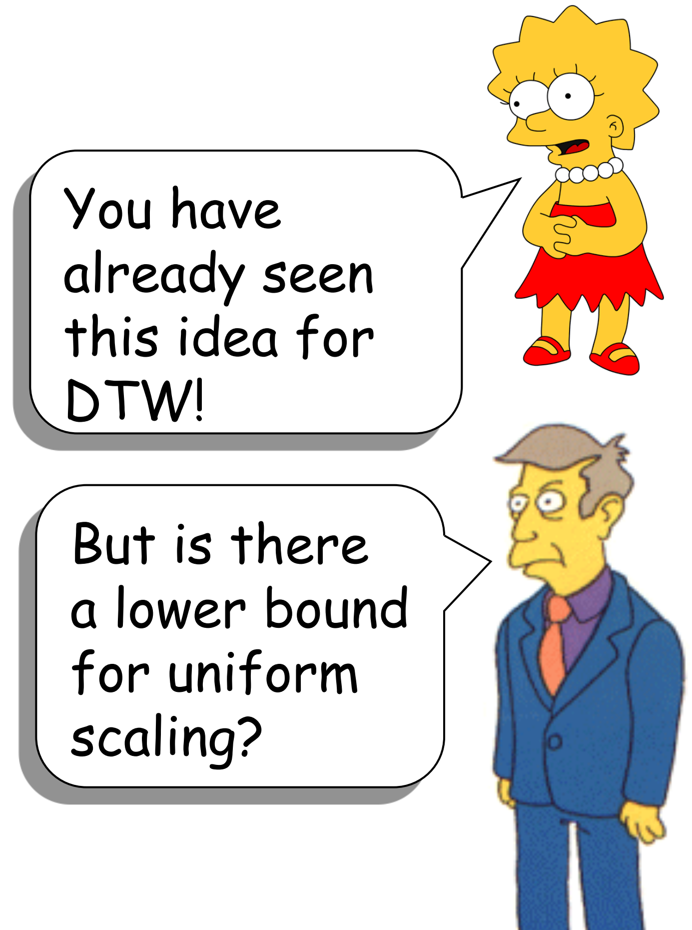

```
Algorithm: Test_All_Scalings(Q,C)
        best_match_val      = inf;
        best_scaling_factor        = null;
         for p = n to m
      QP = rescale(Q,p);
       distance = squared_Euclidean_distanc
       if distance <  best_match_val
          best_match_val = distance;
          best_scaling_factor = p/n;
      end;
         end;
     return(best_match_val, best_scaling_fa
```

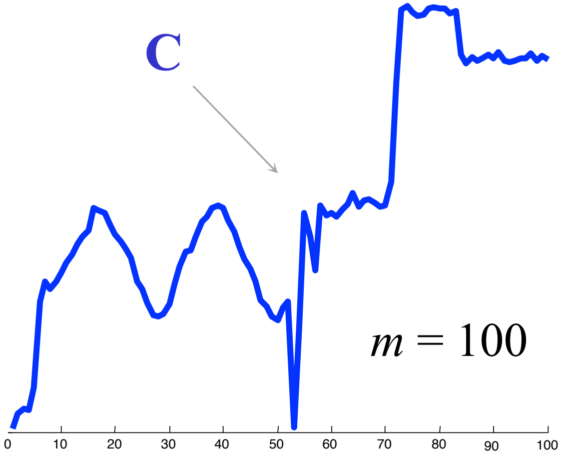
Here is the code to **Test_All_Scalings**, the time complexly is only O((*m*-*n*) * *n*), but we may have to do this many times…

Lower Bounding Revisited!

We can speed up similarity search under uniform scaling by using a lower bounding function, just like we did for DTW.

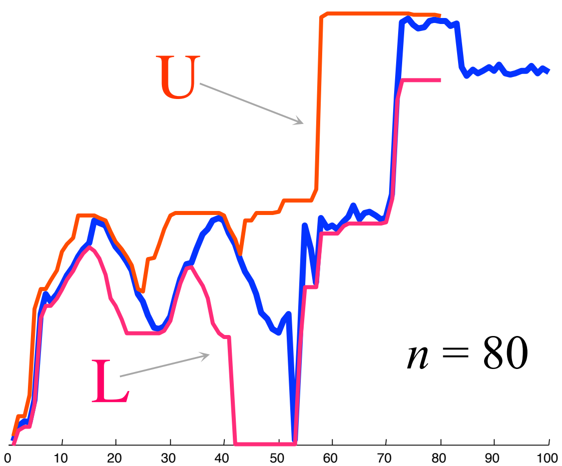
```
Algorithm: Lower_Bounding_Sequential_Scan(Q,C)
overall_best_time_series = null;
overall_best_match_val   = inf;
for i = 1 to number_of_time_series_in_(C)
  if lower_bound_distance(Q,Ci) < overall_best_match_val
    [dist, scale] = Test_All_Scalings(Q,C_i)
     if dist <  overall_best_match_val
            overall_best_time_series =  i;
            overall_best_match_val   = dist;
    end;
  end;
end;
```





Assume that you have a database of time series C_i, all of length 100.

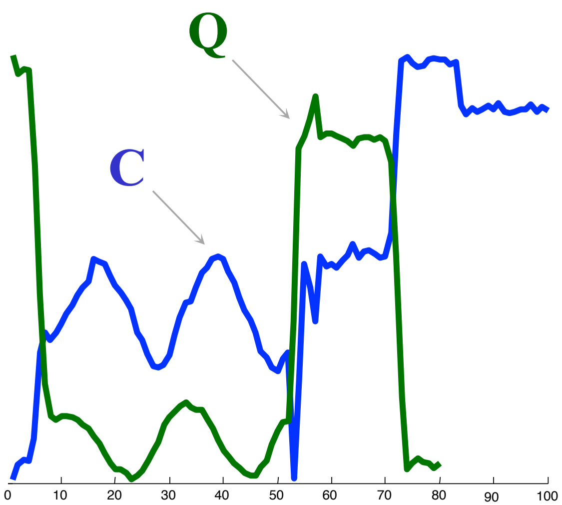
You have a query Q, of length 80, and you want to find the best match in the database under any scaling of Q, from 80 to 100.



We can build envelopes around all candidates time series C_i, in our database, just like we did for DTW, except the definition of the envelopes is different.

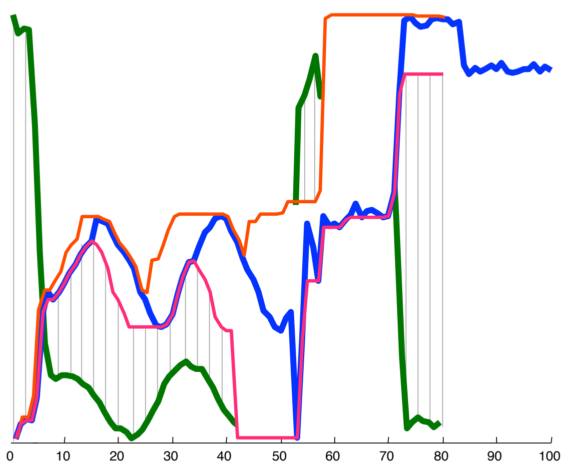
$$U_i = \max(c_{\lfloor (i-1)*m/n \rfloor +1}, \ldots, c_{\lfloor i*m/n \rfloor})$$

$$L_i = \min(c_{\lfloor (i-1)*m/n \rfloor +1}, \ldots, c_{\lfloor i*m/n \rfloor})$$



Once the envelopes have been built, we can lower bound **Test_All_Scalings**.

What's more, the lower bound is one we have already seen!



LB_Keogh

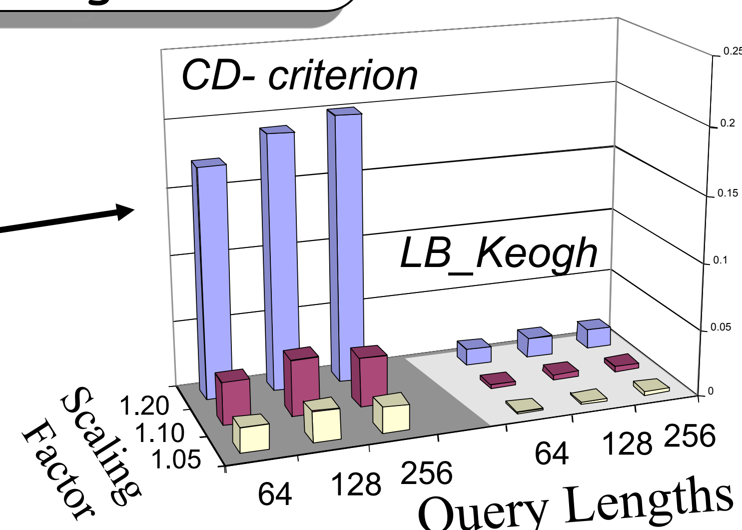
Envelope-Based
Lower Bound

$$LB\_Keogh(Q,C) = \sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2 & if\ q_i > U_i \\ (q_i - L_i)^2 & if\ q_i < L_i \\ 0 & otherwise \end{cases}$$

An experiment to test the utility of lower bounding uniform scaling, over different scaling factors (Y-axis) and scaling lengths (X-axis). The dataset was a "mixed bag" of 10,000 assorted time series.

This is the time taken by brute force search

CD-criterion is the only other lower bound for uniform scaling

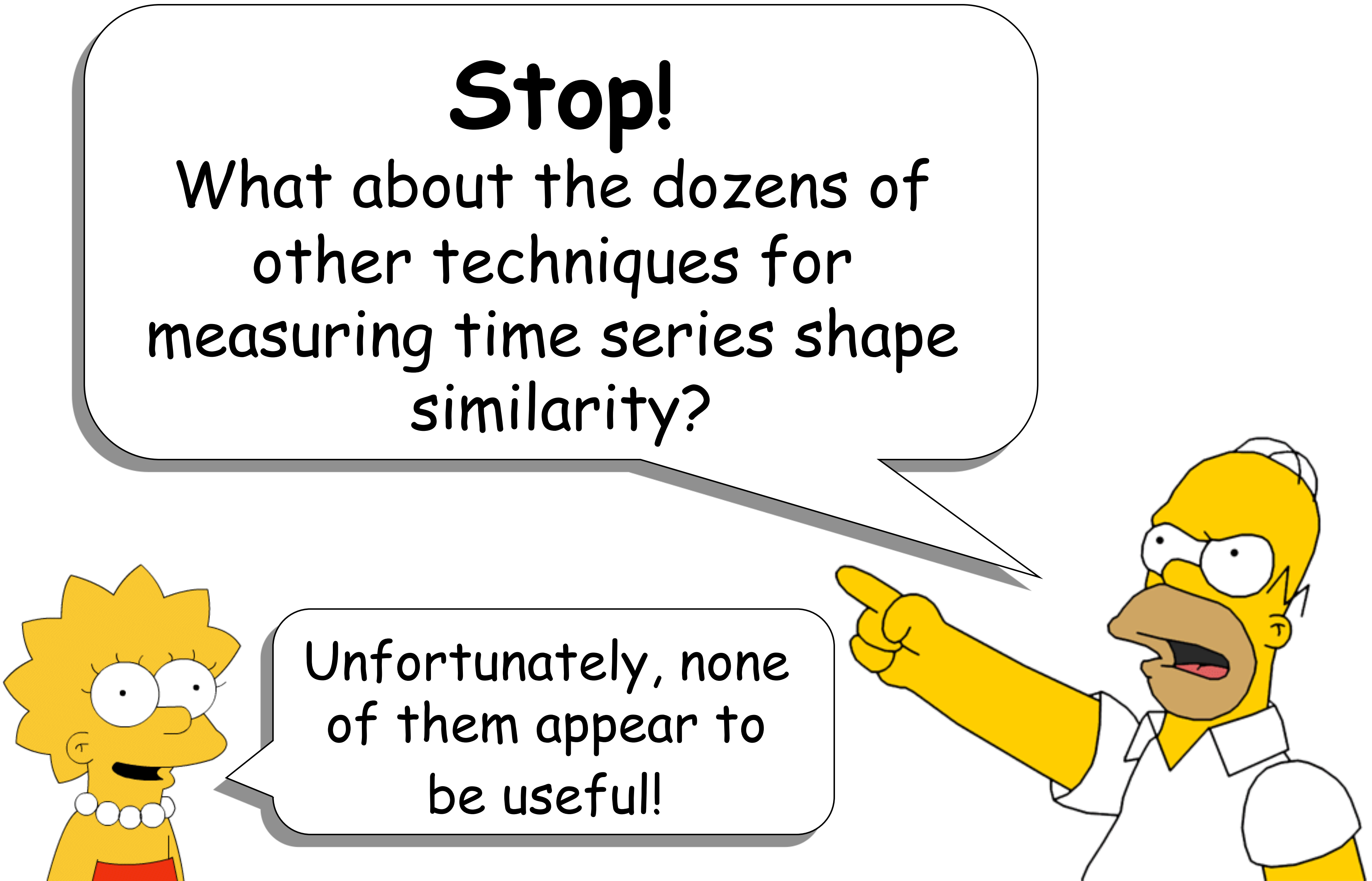


Apart from making DTW tractable for data mining for the first time, *envelope based* techniques also allow…

1. More accurate classification (SDM04)
2. Indexing with uniform scaling (VLDB04)
3. Faster Euclidean indexing (TKDE04)
4. Subsequence matching (IDEAS03)
5. Multivariate time series indexing (SIGKDD03)
6. Rotation invariant indexing (SIGKDD04)
7. DTW on Streaming time series (to appear)
8. Indexing of Images (TPAMI-04, VIS-05)

We *strongly* feel that *envelope based techniques* are the best solutions for time series similarity

Only Euclidean and DTW Distance are Useful



Classification Error Rates on
two publicly available datasets

Approach	Cylinder-Bell-F'	Control-Chart
Euclidean Distance	*0.003*	*0.013*
Aligned Subsequence	**0.451**	**0.623**
Piecewise Normalization	**0.130**	**0.321**
Autocorrelation Functions	**0.380**	**0.116**
Cepstrum	**0.570**	**0.458**
String (Suffix Tree)	**0.206**	**0.578**
Important Points	**0.387**	**0.478**
Edit Distance	**0.603**	**0.622**
String Signature	**0.444**	**0.695**
Cosine Wavelets	**0.130**	**0.371**
Hölder	**0.331**	**0.593**
Piecewise Probabilistic	**0.202**	**0.321**

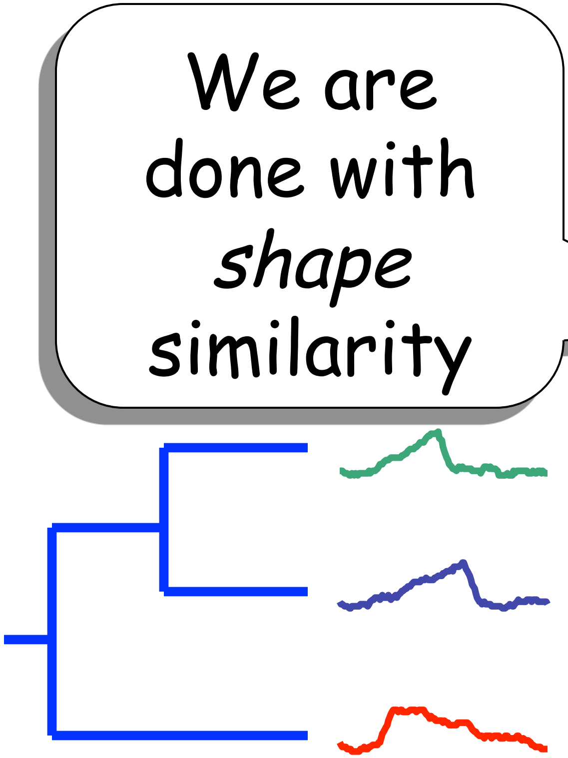
We stand by our claim. At this point there is no evidence that there any *shape* based distance measures better than DTW[1]

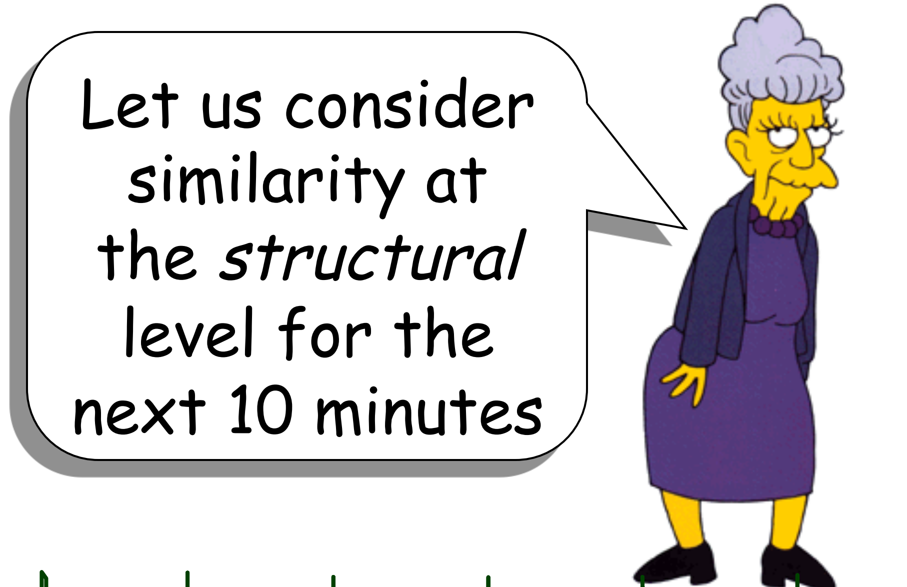
Accuracy

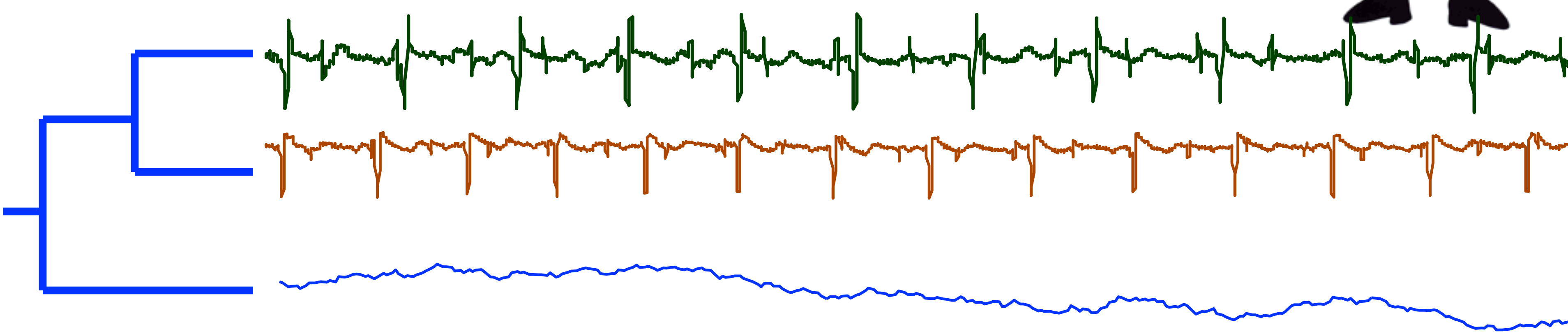
Euclidean Distance ≤ Dynamic Time Warping ≤ RK-Band Dynamic Time Warping

Dr. Keogh is offering a prize of $300 for the first similarity measure that can beat DTW on any 2 real *shape* based datasets[2]

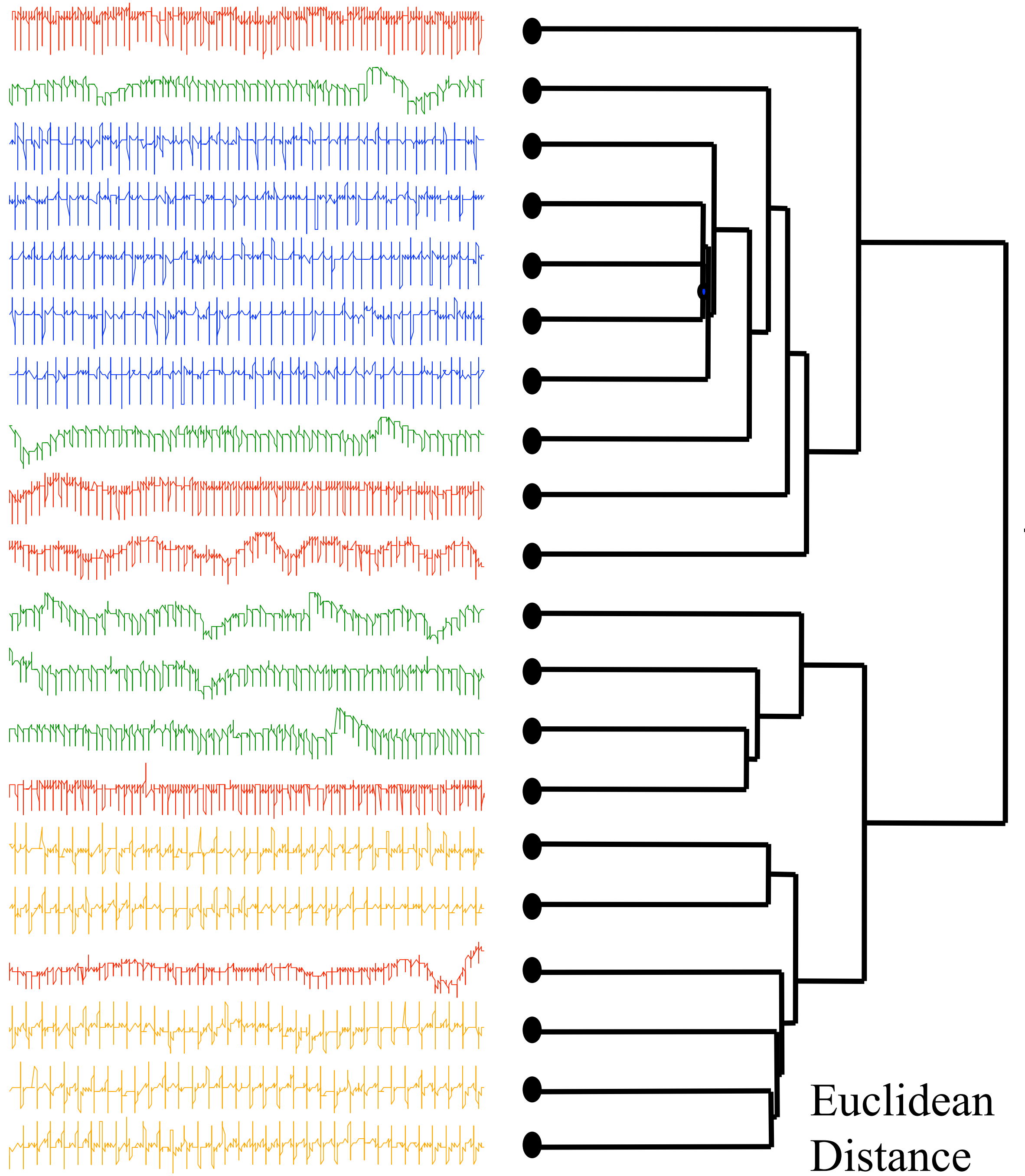
Two Kinds of Similarity





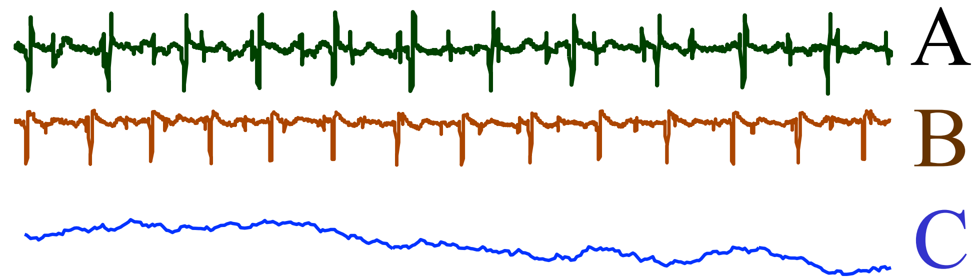


For long time series, *shape* based similarity will give very poor results. We need to measure similarly based on high level *structure*



Structure or Model Based Similarity

The basic idea is to extract *global* features from the time series, create a feature vector, and use these feature vectors to measure similarity and/ or classify



Feature \ Time Series	A	B	C
Max Value	11	12	19
Autocorrelation	0.2	0.3	0.5
Zero Crossings	98	82	13
ARIMA	0.3	0.4	0.1
…	…	…	…

But which
- **features**?
- **distance measure/ learning algorithm**?

Feature-based Classification of Time-series Data

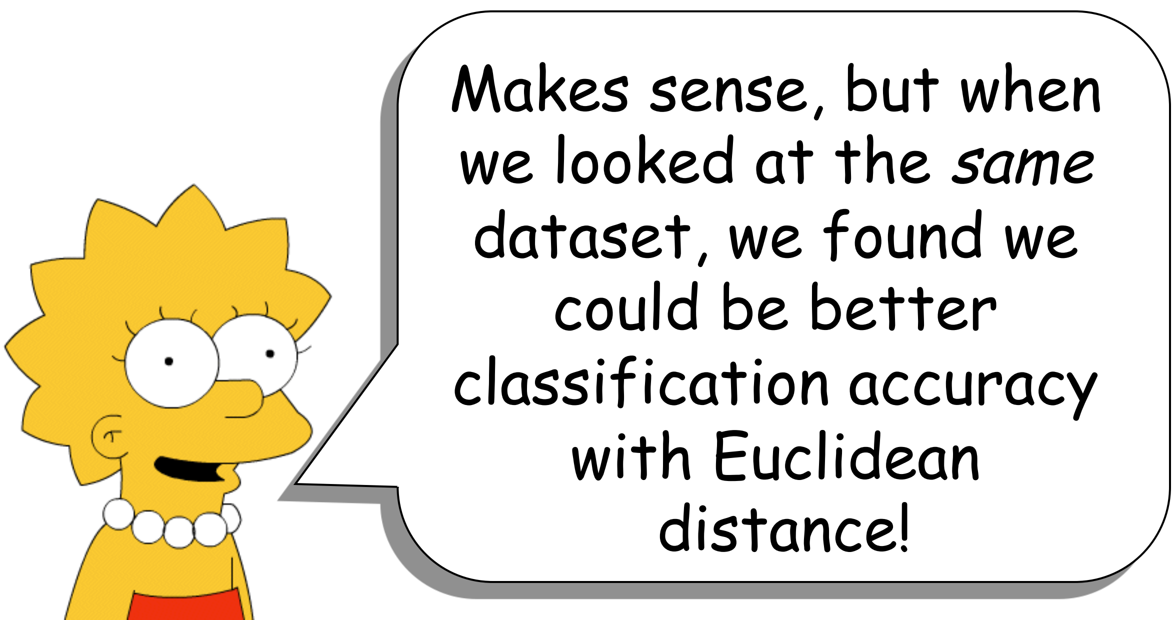
Nanopoulos, Alcock, and Manolopoulos

- features?
- distance measure/ learning algorithm?

Learning Algorithm

multi-layer perceptron neural network

Features
mean
variance
skewness
kurtosis
mean (1st derivative)
variance (1st derivative)
skewness (1st derivative)
kurtosis (1st derivative)



Learning to Recognize Time Series: Combining ARMA Models with Memory-Based Learning

Deng, Moore and Nechyba

- features?
- distance measure/ learning algorithm?

Features

The parameters of the Box Jenkins model.

More concretely, the coefficients of the ARMA model.

"*Time series must be invertible and stationary*"

Distance Measure

Euclidean distance (between coefficients)

- Use to detect drunk drivers!
- Independently rediscovered and generalized by Kalpakis et. al. and expanded by Xiong and Yeung

Deformable Markov Model Templates for Time Series Pattern Matching

Ge and Smyth

Part 1

- features?
- distance measure/ learning algorithm?

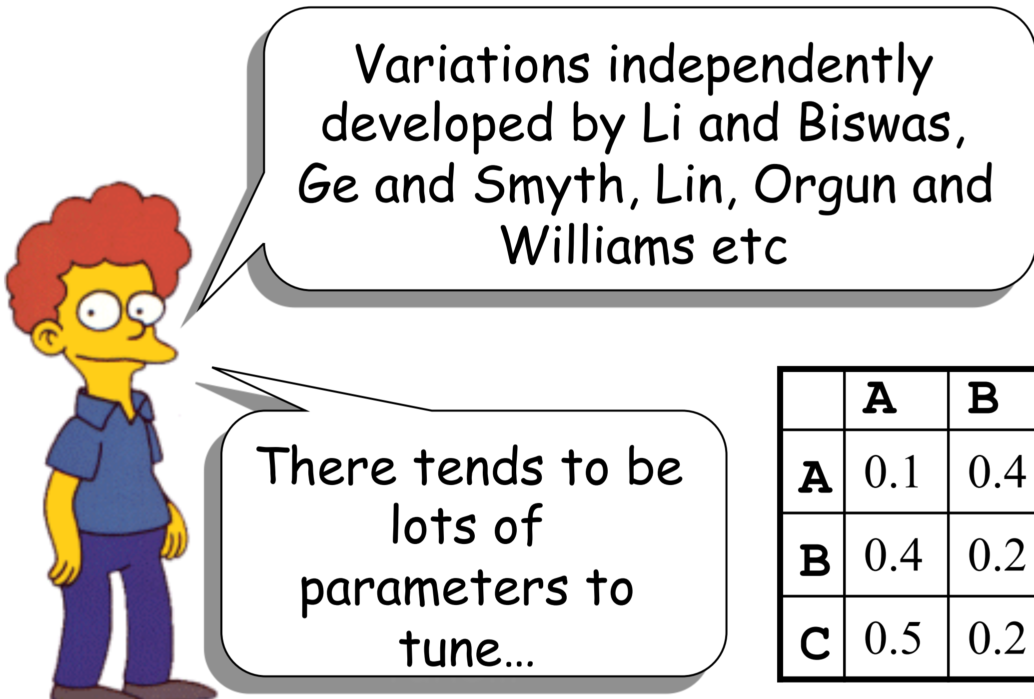
Features

The parameters of a Markov Model

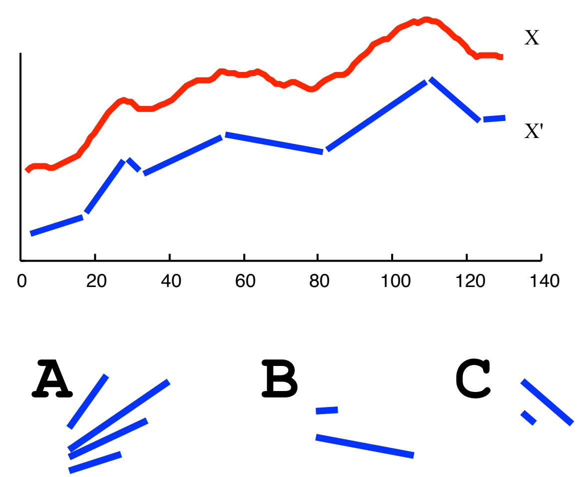
The time series is first converted to a piecewise linear model

Distance Measure

"Viterbi-Like" Algorithm



	A	B	C
A	0.1	0.4	0.5
B	0.4	0.2	0.2
C	0.5	0.2	0.3

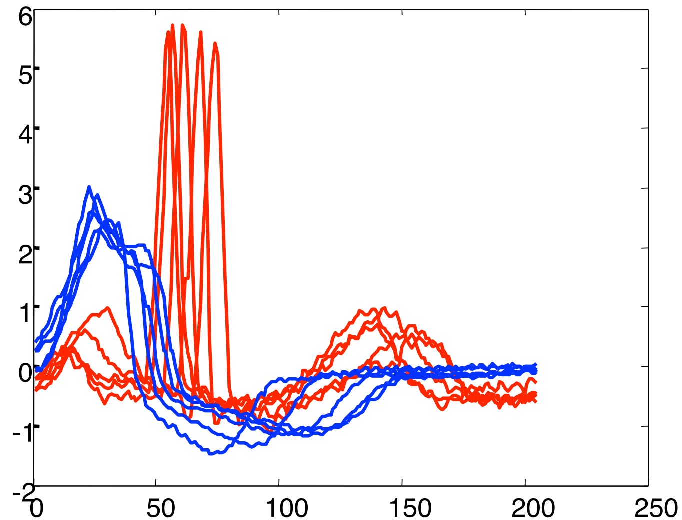


Deformable Markov Model Templates for Time Series Pattern Matching

Ge and Smyth

Part 2

On this problem the approach gets 98% classification accuracy*...



Features
The parameters of a Markov Model The time series is first converted to a piecewise linear model

But Euclidean distance gets 100%! And has no parameters to tune, and is tens of thousands times faster...

Compression Based Dissimilarity

(**In general**) Li, Chen, Li, Ma, and Vitányi: (**For time series**) Keogh, Lonardi and Ratanamahatana

- features?
- distance measure/ learning algorithm?

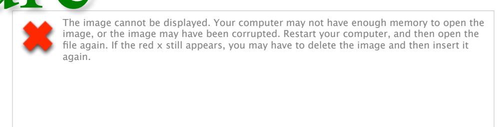
Features

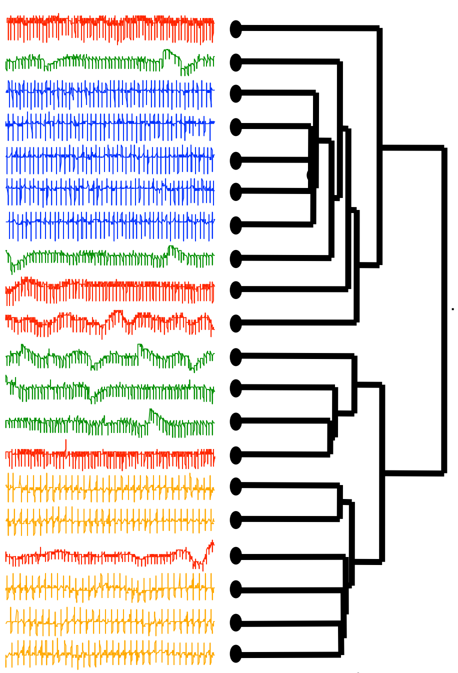
Whatever structure the compression algorithm finds...

The time series is first converted to the SAX symbolic representation*

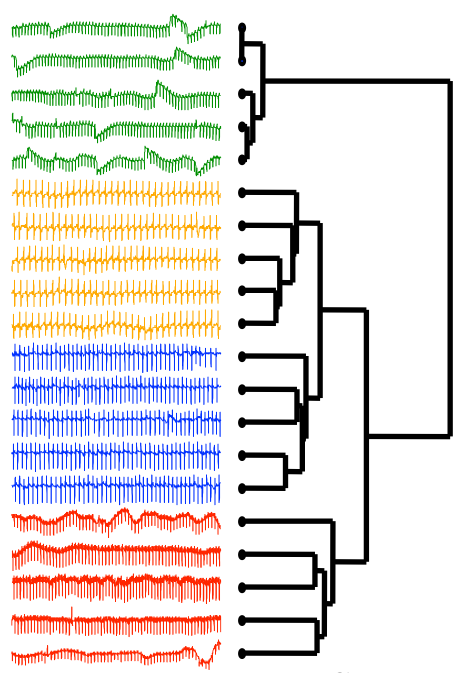
Distance Measure

Co-Compressibility



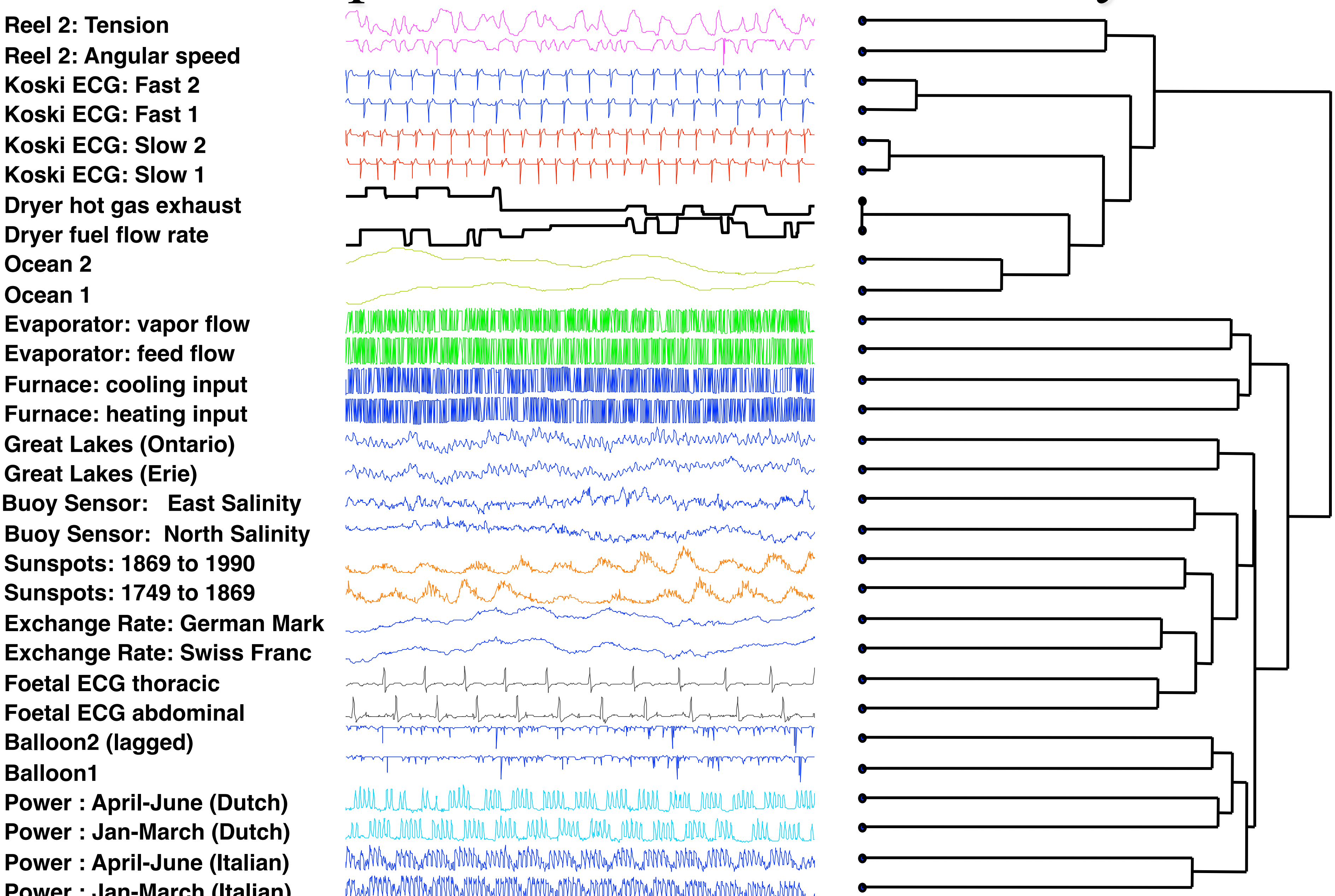


Euclidean



CDM

Compression Based Dissimilarity

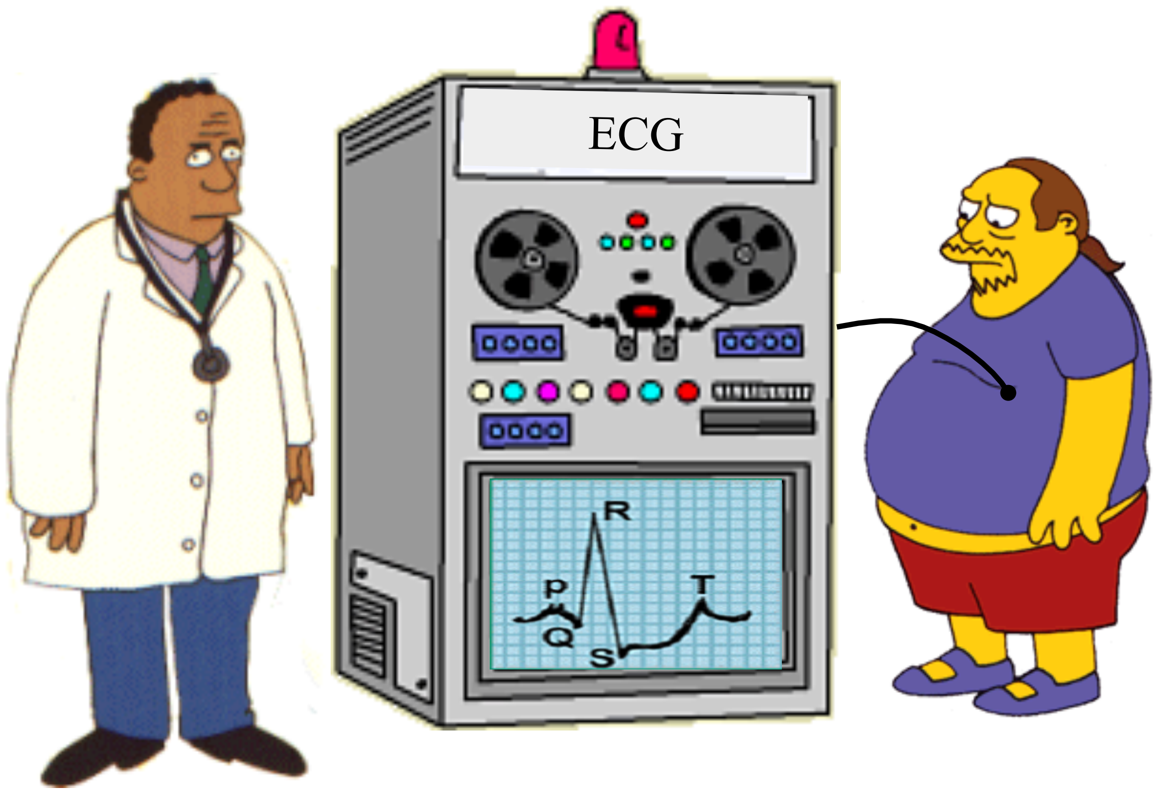


Summary of Time Series Similarity

- If you have *short* time series, use DTW after searching over the warping window size[1] (and shape[2])
- Then use envelope based lower bounds to speed things up[3].

- If you have *long* time series, and you know nothing about your data, try compression based dissimilarity.
- If you do know something about your data, try to leverage of this knowledge to extract features.

Motivating example revisited…



You go to the doctor because of chest pains. Your ECG looks strange…

Your doctor wants to search a database to find **similar** ECGs, in the hope that they will offer clues about your condition...

Two questions:

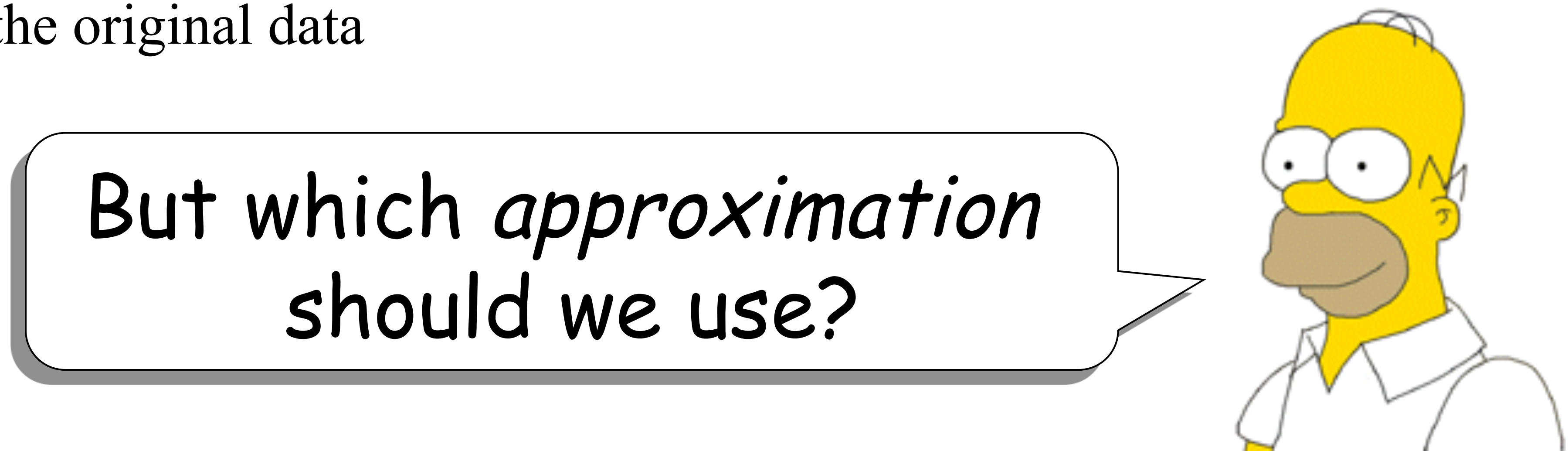
- How do we define similar?
- **How do we search quickly?**

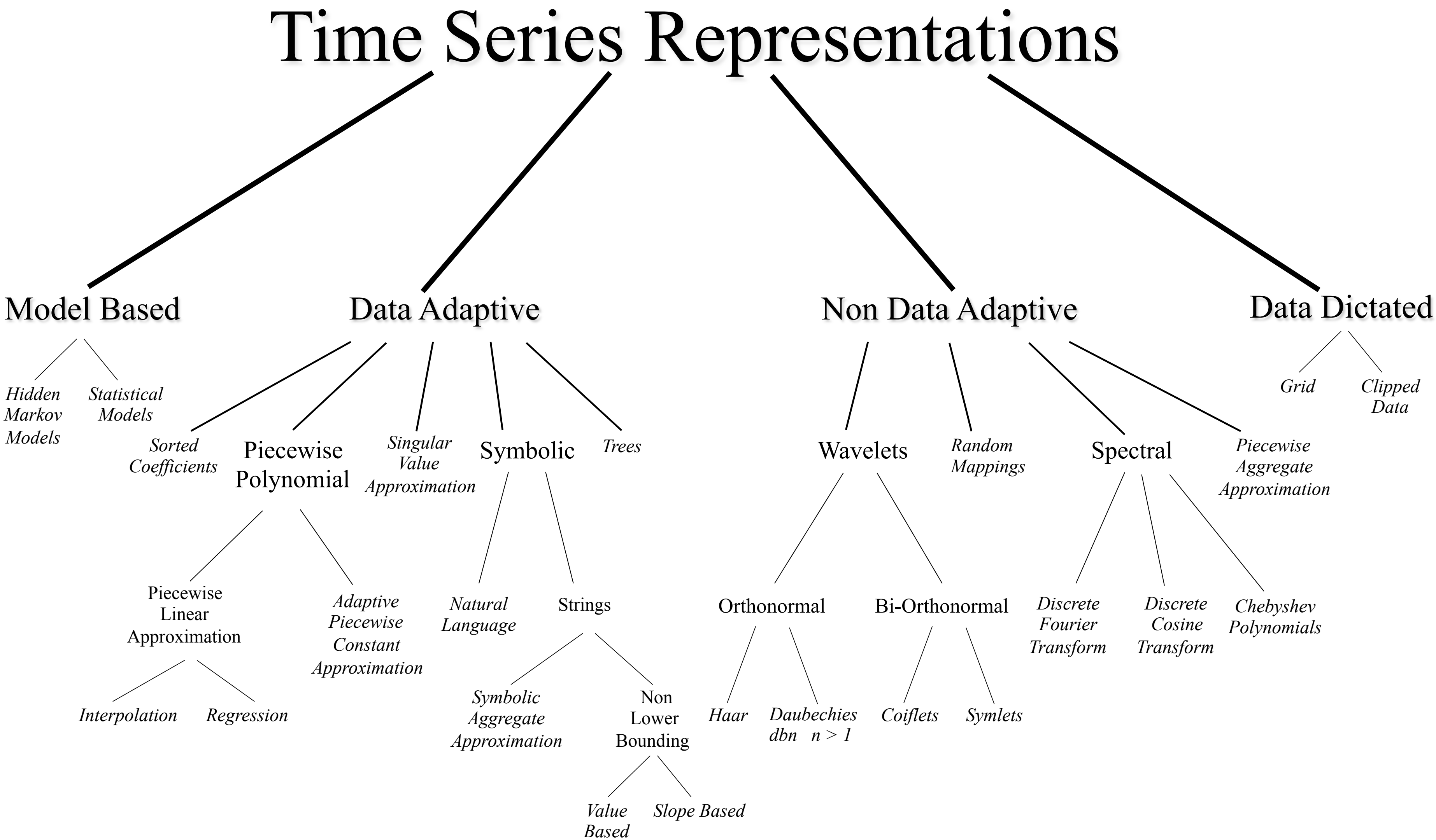
The Generic Data Mining Algorithm

- Create an *approximation* of the data, which will fit in main memory, yet retains the essential features of interest

- Approximately solve the problem at hand in main memory

- Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data





Time Series Representations
Model Based
Hidden Markov Models
Statistical Models
Data Adaptive
Sorted Coefficients
Piecewise Polynomial
Piecewise Linear Approximation
Interpolation
Regression
Adaptive Piecewise Constant Approximation
Singular Value Approximation
Symbolic
Natural Language
Strings
Symbolic Aggregate Approximation
Non Lower Bounding
Value Based
Slope Based
Trees
Non Data Adaptive
Wavelets
Orthonormal
Haar
Daubechies dbn n > 1
Bi-Orthonormal
Coiflets
Symlets
Random Mappings
Spectral
Discrete Fourier Transform
Discrete Cosine Transform
Chebyshev Polynomials
Piecewise Aggregate Approximation
Data Dictated
Grid
Clipped Data

The Generic Data Mining Algorithm (revisited)

• Create an *approximation* of the data, which will fit in main memory, yet retains the essential features of interest

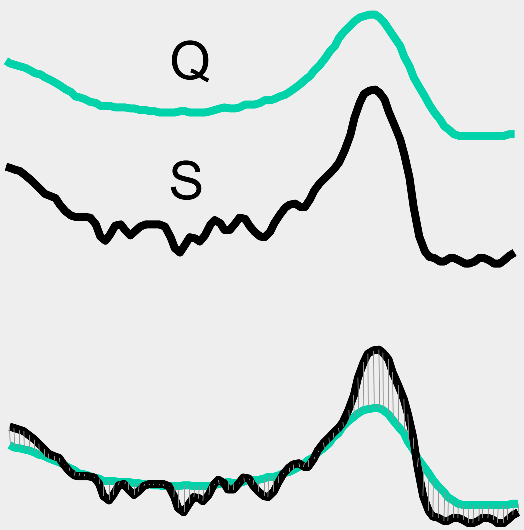
• Approximately solve the problem at hand in main memory

• Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data

This *only* works if the approximation allows **lower bounding**

What is Lower Bounding?

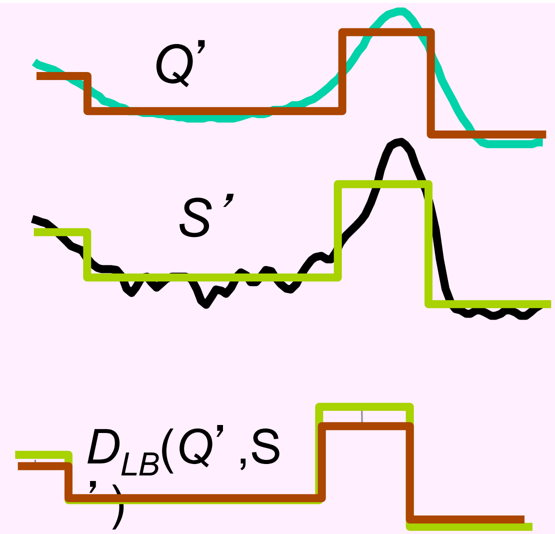
• Recall that we have seen lower bounding for **distance measures** (DTW and uniform scaling) Lower bounding for **representations** is a similar idea…



$$D(Q,S) \equiv \sqrt{\sum_{i=1}^{n} (q_i - s_i)^2}$$

Raw Data

Approximation or "Representation"



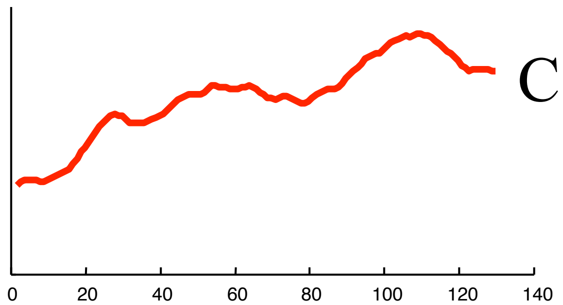
$$D_{LB}(Q',S') \equiv \sqrt{\sum_{i=1}^{M} (sr_i - sr_{i-1})(qv_i - sv_i)^2}$$

Lower bounding means that for all Q and S, we have: $D_{LB}(Q',S') \leq D(Q,S)$

In a seminal* paper in SIGMOD 93, I showed that lower bounding of a representation is a necessary and sufficient condition to allow time series indexing, with the guarantee of no false dismissals

Christos work was originally with indexing time series with the Fourier representation. Since then, there have been hundreds of follow up papers on other data types, tasks and representations

An Example of a Dimensionality Reduction Technique I



$n = 128$

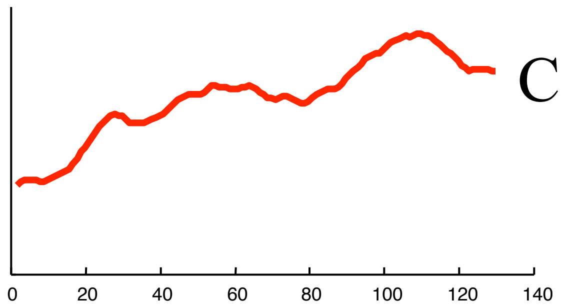
Raw Data

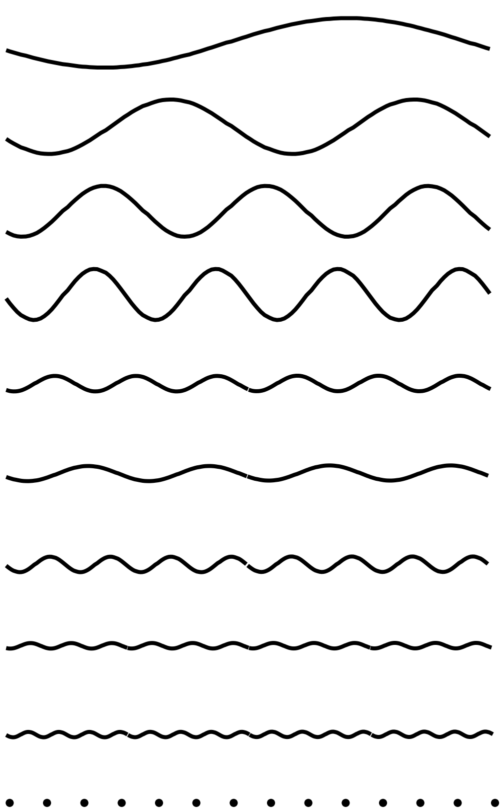
0.4995
0.5264
0.5523
0.5761
0.5973
0.6153
0.6301
0.6420
0.6515
0.6596
0.6672
0.6751
0.6843
0.6954
0.7086
0.7240
0.7412
0.7595
0.7780
0.7956
0.8115
0.8247
0.8345
0.8407
0.8431
0.8423
0.8387
…
…

The graphic shows a time series with 128 points.

The raw data used to produce the graphic is also reproduced as a column of numbers (just the first 30 or so points are shown).

An Example of a Dimensionality Reduction Technique II





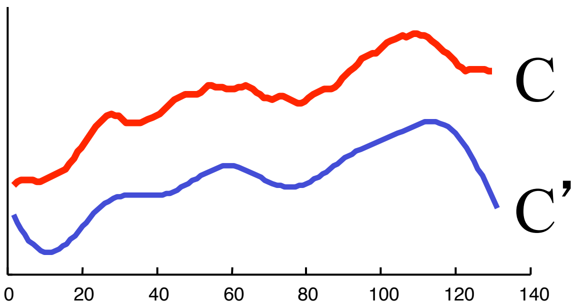
Raw Data	Fourier Coefficients
0.4995	1.5698
0.5264	1.0485
0.5523	0.7160
0.5761	0.8406
0.5973	0.3709
0.6153	0.4670
0.6301	0.2667
0.6420	0.1928
0.6515	0.1635
0.6596	0.1602
0.6672	0.0992
0.6751	0.1282
0.6843	0.1438
0.6954	0.1416
0.7086	0.1400
0.7240	0.1412
0.7412	0.1530
0.7595	0.0795
0.7780	0.1013
0.7956	0.1150
0.8115	0.1801
0.8247	0.1082
0.8345	0.0812
0.8407	0.0347
0.8431	0.0052
0.8423	0.0017
0.8387	0.0002
…	…
…	…

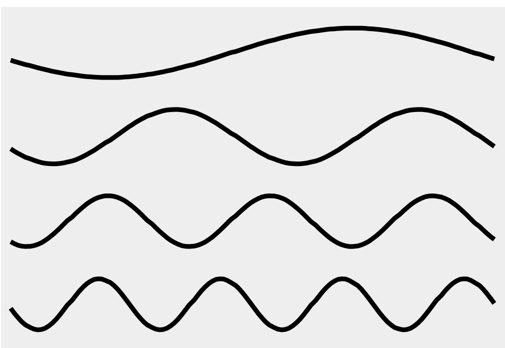
We can decompose the data into 64 pure sine waves using the Discrete Fourier Transform (just the first few sine waves are shown).

The Fourier Coefficients are reproduced as a column of numbers (just the first 30 or so coefficients are shown).

Note that at this stage we have **not** done dimensionality reduction, we have merely changed the representation...

An Example of a Dimensionality Reduction Technique III





We have discarded $\frac{15}{16}$ of the data.

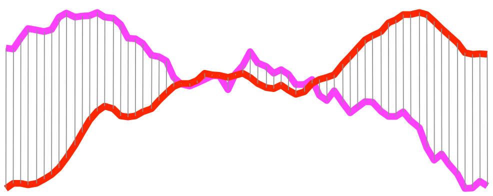
Raw Data	Fourier Coefficients	Truncated Fourier Coefficients
0.4995	1.5698	1.5698
0.5264	1.0485	1.0485
0.5523	0.7160	0.7160
0.5761	0.8406	0.8406
0.5973	0.3709	0.3709
0.6153	0.4670	0.4670
0.6301	0.2667	0.2667
0.6420	0.1928	0.1928
0.6515	0.1635	
0.6596	0.1602	
0.6672	0.0992	
0.6751	0.1282	
0.6843	0.1438	
0.6954	0.1416	
0.7086	0.1400	
0.7240	0.1412	
0.7412	0.1530	
0.7595	0.0795	
0.7780	0.1013	
0.7956	0.1150	
0.8115	0.1801	
0.8247	0.1082	
0.8345	0.0812	
0.8407	0.0347	
0.8431	0.0052	
0.8423	0.0017	
0.8387	0.0002	
…	…	
…	…	

$n = 128$
$N = 8$
$C_{ratio} = 1/16$

… however, note that the first few sine waves tend to be the largest (equivalently, the magnitude of the Fourier coefficients tend to decrease as you move down the column).

We can therefore truncate most of the small coefficients with little effect.

An Example of a Dimensionality Reduction Technique IIII



$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}\left(q_i - c_i\right)^2}$$

Raw Data 1		Raw Data 2
0.4995	-	0.7412
0.5264	-	0.7595
0.5523	-	0.7780
0.5761	-	0.7956
0.5973	-	0.8115
0.6153	-	0.8247
0.6301	-	0.8345
0.6420	-	0.8407
0.6515	-	0.8431
0.6596	-	0.8423
0.6672	-	0.8387
0.6751	-	0.4995
0.6843	-	0.5264
0.6954	-	0.5523
0.7086	-	0.5761
0.7240	-	0.5973
0.7412	-	0.6153
0.7595	-	0.6301
0.7780	-	0.6420
0.7956	-	0.6515
0.8115	-	0.6596
0.8247	-	0.6672
0.8345	-	0.6751
0.8407	-	0.6843
0.8431	-	0.6954
0.8423	-	0.7086
0.8387	-	0.7240
…		…
…		…

$\geq$

Truncated Fourier Coefficients 1		Truncated Fourier Coefficients 2
1.5698	-	1.1198
1.0485	-	1.4322
0.7160	-	1.0100
0.8406	-	0.4326
0.3709	-	0.5609
0.4670	-	0.8770
0.2667	-	0.1557
0.1928	-	0.4528

The Euclidean distance between the two truncated Fourier coefficient vectors is always less than or equal to the Euclidean distance between the two raw data vectors*.

So DFT allows lower bounding!

*Parseval's Theorem

Mini Review for the Generic Data Mining Algorithm

We *cannot* fit all that raw data in main memory.
We *can* fit the dimensionally reduced data in main memory.

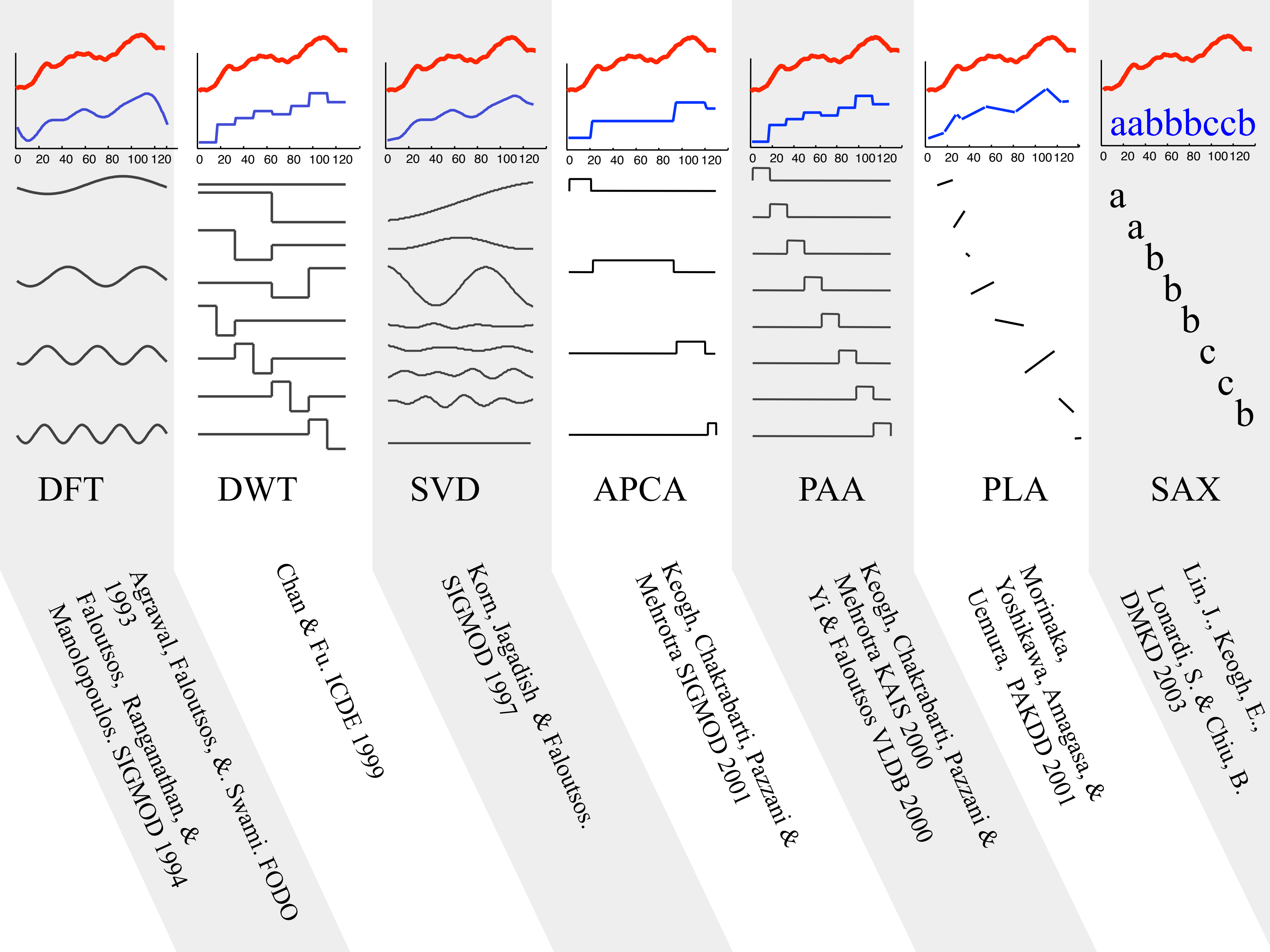
So we will solve the problem at hand on the dimensionally reduced data, making a few accesses to the raw data were necessary, and, if we are careful, the lower bounding property will insure that we get the right answer!

Raw ata 1	Raw Data 2	Raw Data n
.4995	0.7412	0.8115
.5264	0.7595	0.8247
.5523	0.7780	0.8345
.5761	0.7956	0.8407
.5973	0.8115	0.8431
.6153	0.8247	0.8423
.6301	0.8345	0.8387
.6420	0.8407	0.4995
.6515	0.8431	0.7412
.6596	0.8423	0.7595
.6672	0.8387	0.7780
.6751	0.4995	0.7956
.6843	0.5264	0.5264
.6954	0.5523	0.5523
.7086	0.5761	0.5761
.7240	0.5973	0.5973
.7412	0.6153	0.6153

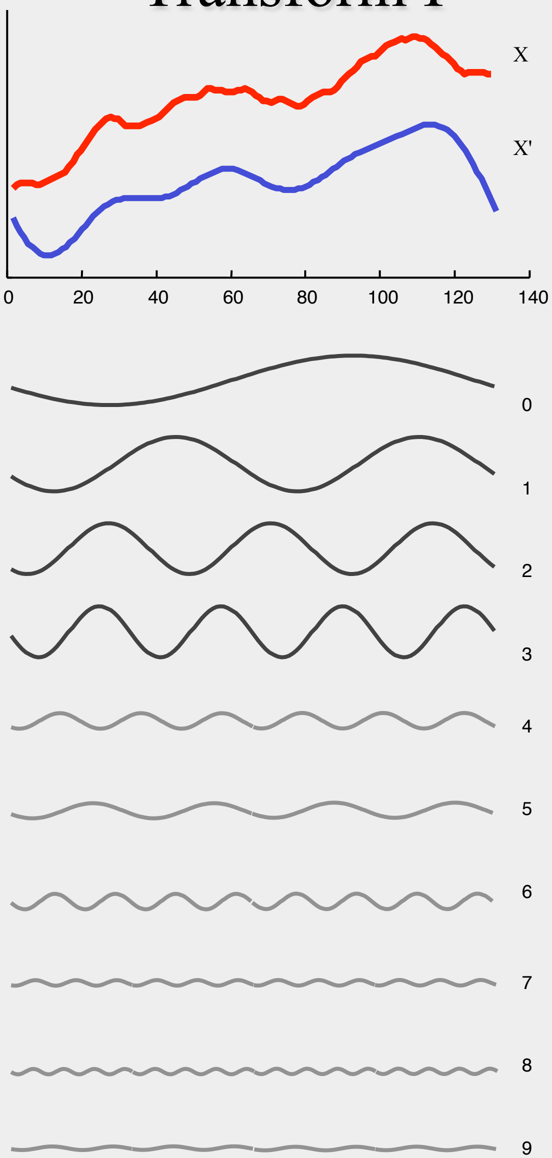
Disk

Main Memory

Truncated Fourier Coefficients 1	Truncated Fourier Coefficients 2	Truncated Fourier Coefficients n
1.5698	1.1198	1.3434
1.0485	1.4322	1.4343
0.7160	1.0100	1.4643
0.8406	0.4326	0.7635
0.3709	0.5609	0.5448
0.4670	0.8770	0.4464
0.2667	0.1557	0.7932
0.1928	0.4528	0.2126


0 20 40 60 80 100 120
aabbbccb
a a b b b c c b
DFT
DWT
SVD
APCA
PAA
PLA
SAX
Agrawal, Faloutsos, &. Swami. FODO 1993
Faloutsos, Ranganathan, & Manolopoulos. SIGMOD 1994
Chan & Fu. ICDE 1999
Korn, Jagadish & Faloutsos. SIGMOD 1997
Keogh, Chakrabarti, Pazzani & Mehrotra SIGMOD 2001
Keogh, Chakrabarti, Pazzani & Mehrotra KAIS 2000
Yi & Faloutsos VLDB 2000
Morinaka, Yoshikawa, Amagasa, & Uemura, PAKDD 2001
Lin, J., Keogh, E., Lonardi, S. & Chiu, B. DMKD 2003

Discrete Fourier Transform I



Basic Idea: Represent the time series as a linear combination of sines and cosines, but keep only the first *n/2* coefficients.

Why *n*/2 coefficients? Because each sine wave requires 2 numbers, for the phase (*w*) and amplitude (*A*,*B*).

Jean Fourier

1768-1830

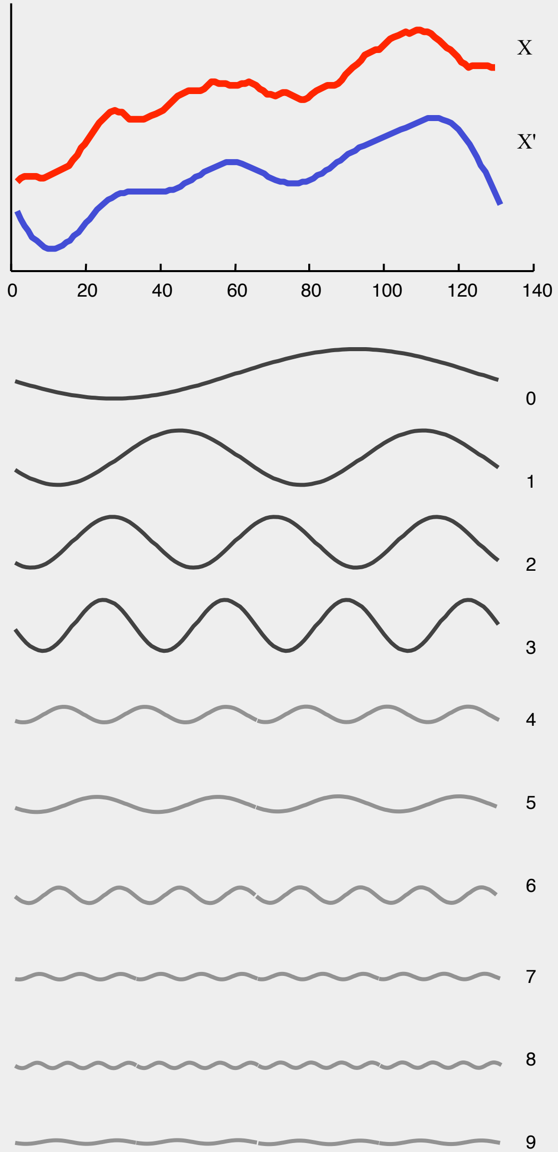
$$C(t) = \sum_{k=1}^{n} \left(A_k \cos(2\pi w_k t) + B_k \sin(2\pi w_k t) \right)$$

<u>Excellent free Fourier Primer</u>

Hagit Shatkay, The Fourier Transform - a Primer", Technical Report CS-95-37, Department of Computer Science, Brown University, 1995.

http://www.ncbi.nlm.nih.gov/CBBresearch/Postdocs/Shatkay/

Discrete Fourier Transform II



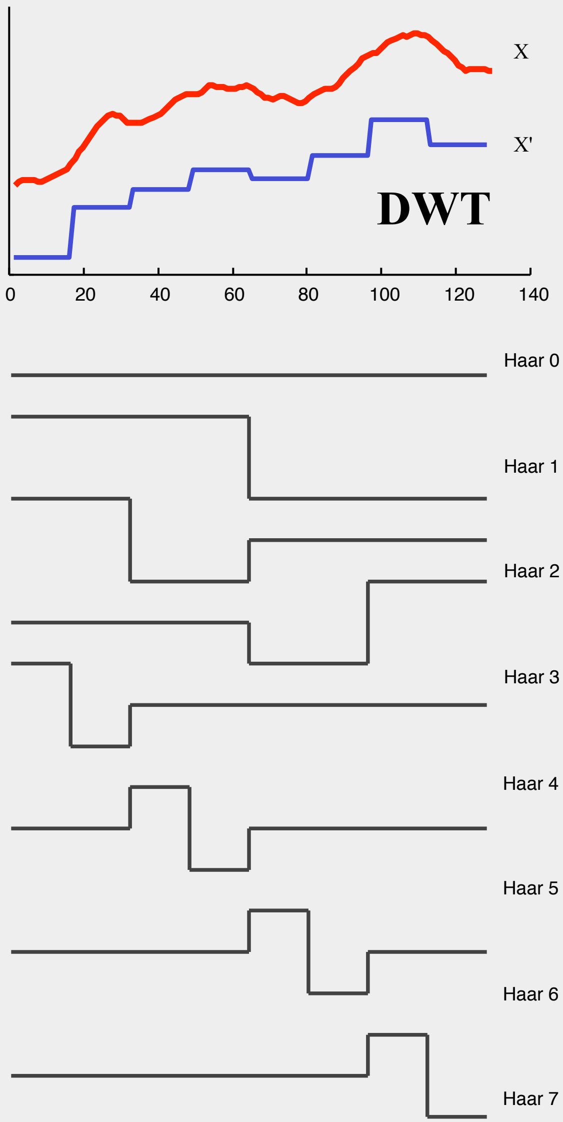
Pros and Cons of DFT as a time series representation.

- Good ability to compress most natural signals.
- Fast, off the shelf DFT algorithms exist. O($n$log(n)).
- (Weakly) able to support time warped queries.

- Difficult to deal with sequences of different lengths.
- Cannot support weighted distance measures.

Note: The related transform DCT, uses only cosine basis functions. It does not seem to offer any particular advantages over DFT.

Discrete Wavelet Transform I



Basic Idea: Represent the time series as a linear combination of Wavelet basis functions, but keep only the first *N* coefficients.

Although there are many different types of wavelets, researchers in time series mining/indexing generally use Haar wavelets.

Haar wavelets seem to be as powerful as the other wavelets for most problems and are very easy to code.

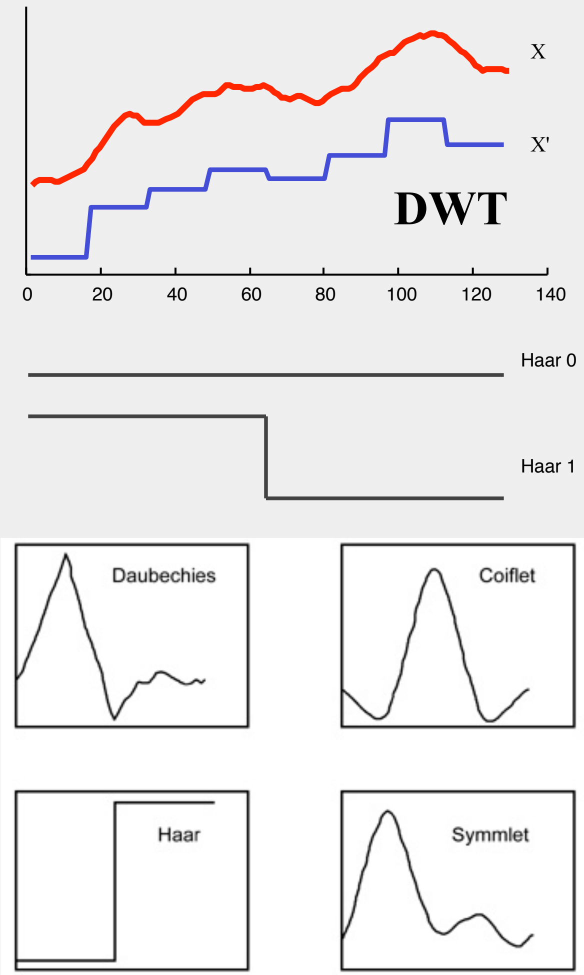
Alfred Haar

1885-1933

<u>Excellent free Wavelets Primer</u>

Stollnitz, E., DeRose, T., & Salesin, D. (1995). *Wavelets for computer graphics A primer: IEEE Computer Graphics and Applications.*

Discrete Wavelet Transform II



Ingrid Daubechies

1954 -

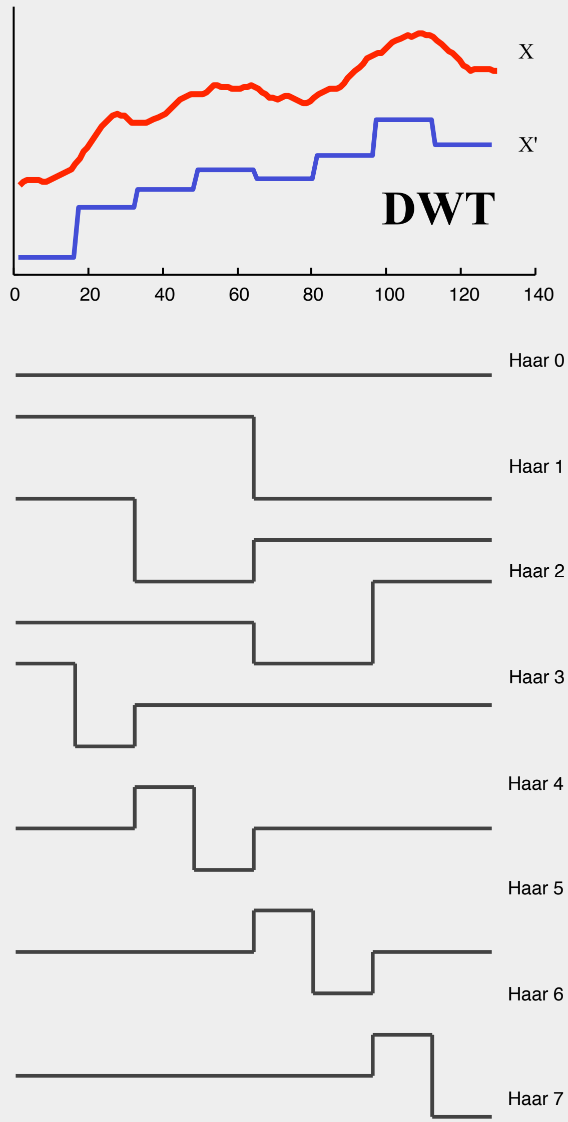
We have only considered one type of wavelet, there are many others.
Are the other wavelets better for indexing?

YES: I. Popivanov, R. Miller. *Similarity Search Over Time Series Data Using Wavelets*. ICDE 2002.

NO: K. Chan and A. Fu. *Efficient Time Series Matching by Wavelets*. ICDE 1999

Later in this tutorial I will answer this question.

Discrete Wavelet Transform III

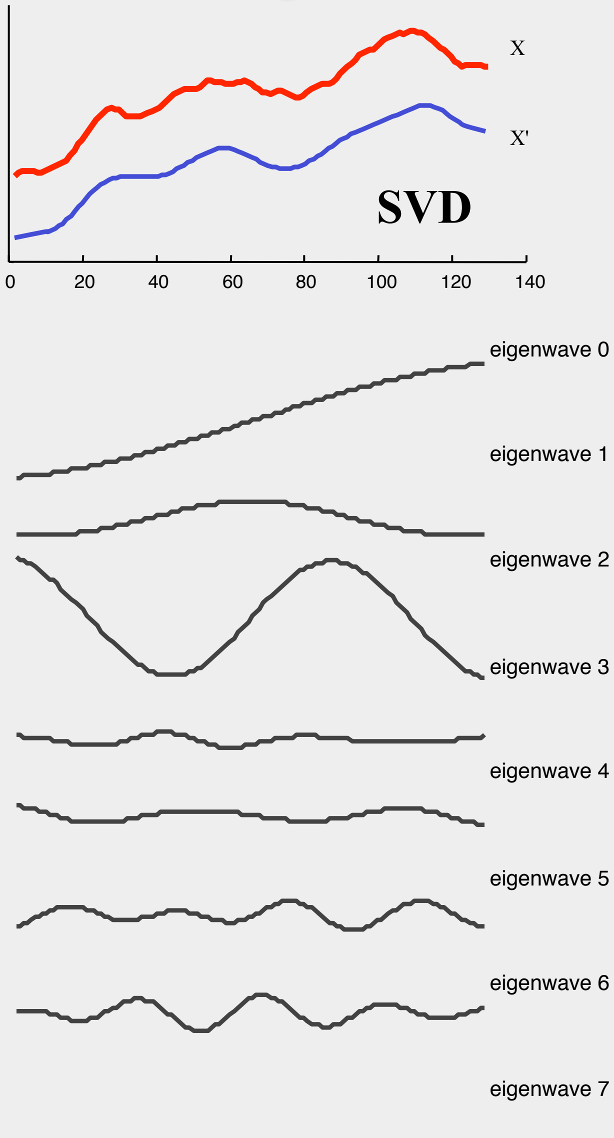


Pros and Cons of Wavelets as a time series representation.

- Good ability to compress stationary signals.
- Fast linear time algorithms for DWT exist.
- Able to support some interesting non-Euclidean similarity measures.

- Signals must have a length $n = 2^{\text{some\_integer}}$
- Works best if N is $= 2^{\text{some\_integer}}$. Otherwise wavelets approximate the left side of signal at the expense of the right side.
- Cannot support weighted distance measures.

Singular Value Decomposition I



Basic Idea: Represent the time series as a linear combination of *eigenwaves* but keep only the first *N* coefficients.

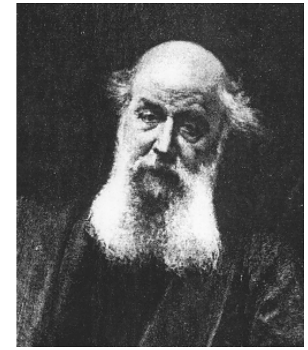
SVD is similar to Fourier and Wavelet approaches is that we represent the data in terms of a linear combination of shapes (in this case *eigenwaves*).

SVD differs in that the *eigenwaves* are data dependent.

SVD has been successfully used in the text processing community (where it is known as *Latent Symantec Indexing*) for many years.

<u>Good free SVD Primer</u>

Singular Value Decomposition - A Primer.
Sonia Leach



James Joseph Sylvester

1814-1897

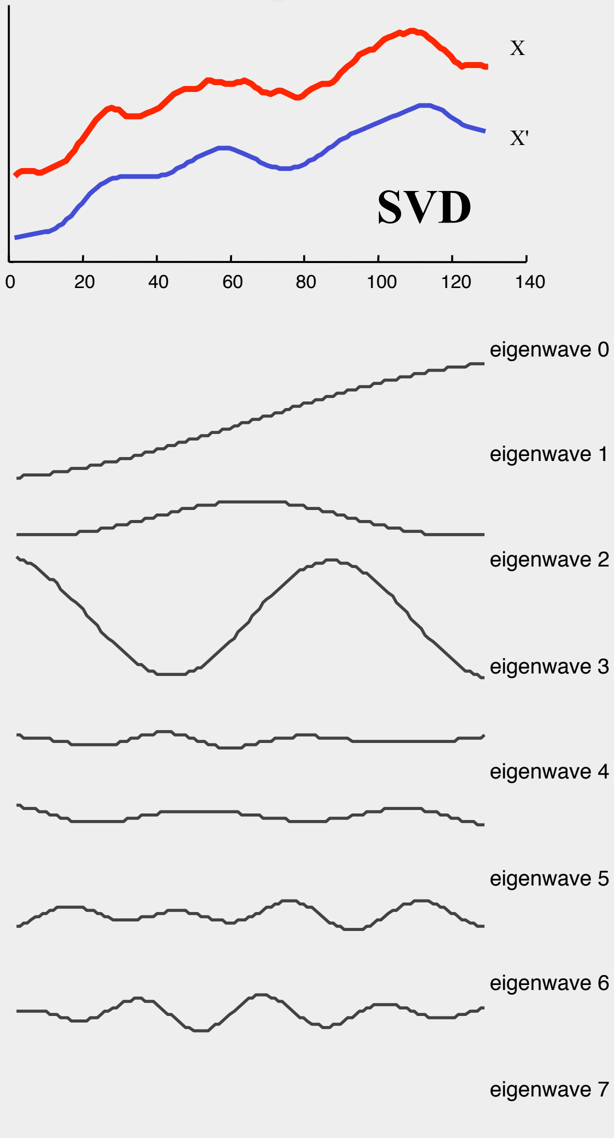
Camille Jordan

(1838--1921)

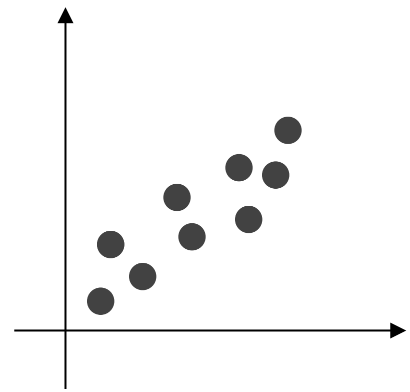
Eugenio Beltrami

1835-1899

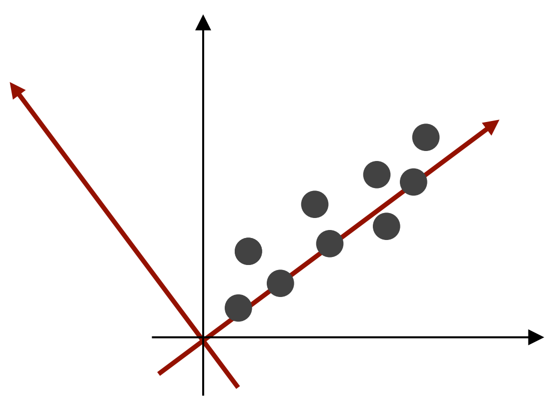
Singular Value Decomposition II



How do we create the eigenwaves?



We have previously seen that we can regard time series as points in high dimensional space.



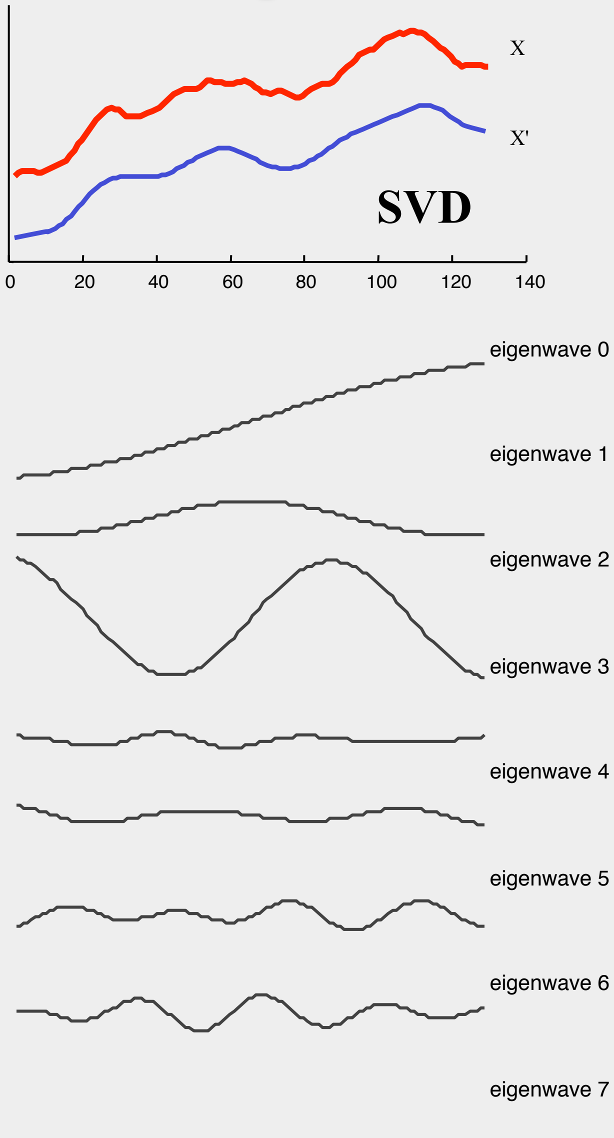
We can rotate the axes such that axis 1 is aligned with the direction of maximum variance, axis 2 is aligned with the direction of maximum variance orthogonal to axis 1 etc.

Since the first few eigenwaves contain most of the variance of the signal, the rest can be truncated with little loss.

$$A = U\Sigma V^T$$

This process can be achieved by factoring a M by n matrix of time series into 3 other matrices, and truncating the new matrices at size N.

Singular Value Decomposition III



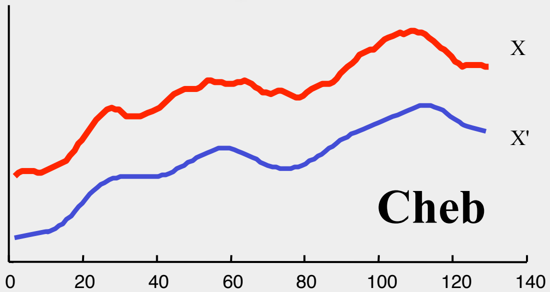
Pros and Cons of SVD as a time series representation.

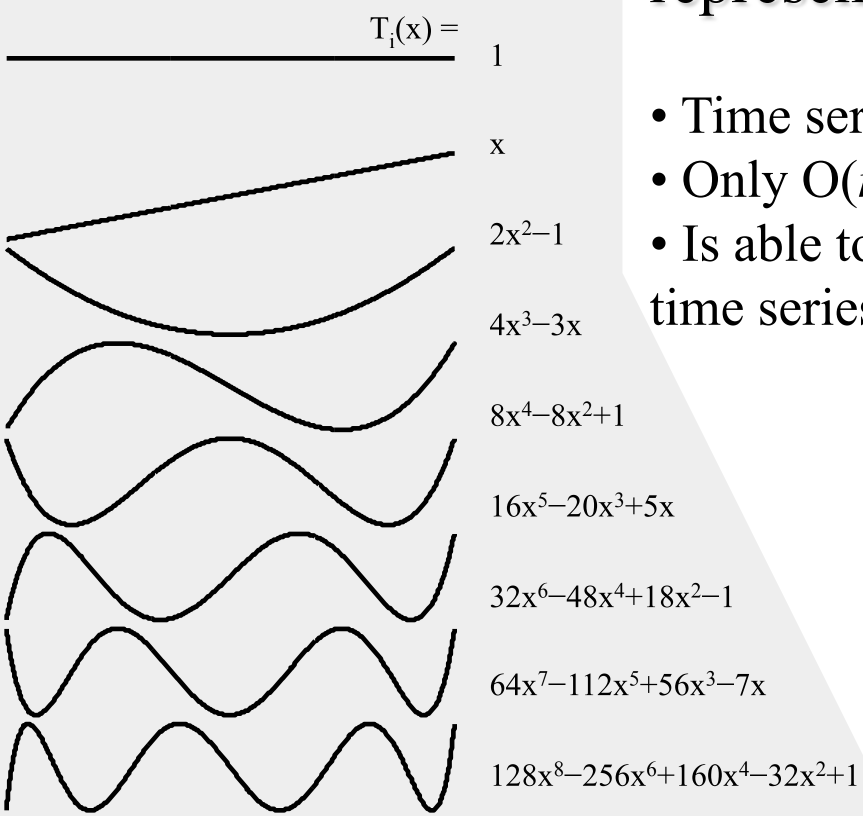
- Optimal linear dimensionality reduction technique .
- The eigenvalues tell us something about the underlying structure of the data.

- Computationally very expensive.
  - Time: $O(Mn^2)$
  - Space: $O(Mn)$
- An insertion into the database requires recomputing the SVD.
- Cannot support weighted distance measures or non Euclidean measures.

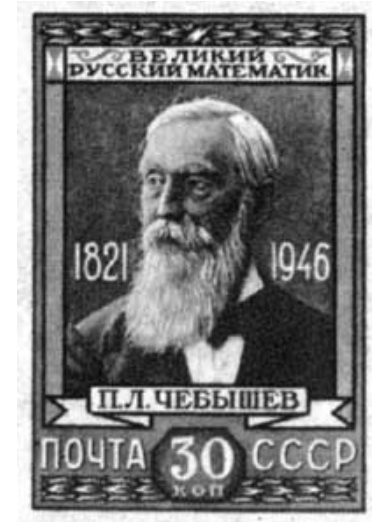
Note: There has been some promising research into mitigating SVDs time and space complexity.

Chebyshev Polynomials





Basic Idea: Represent the time series as a linear combination of Chebyshev Polynomials



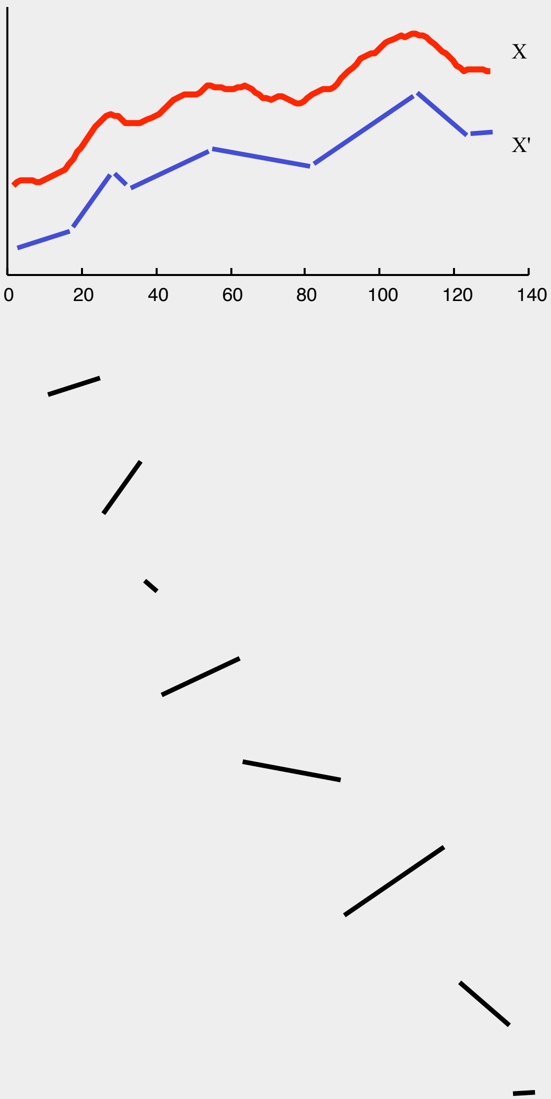
Pafnuty Chebyshev
1821-1946

Pros and Cons of Chebyshev Polynomials as a time series representation.

- Time series can be of arbitrary length
- Only O(*n*) time complexity
- Is able to support multi-dimensional time series*.

- Time series must be renormalized to have length between –1 and 1

Piecewise Linear Approximation I

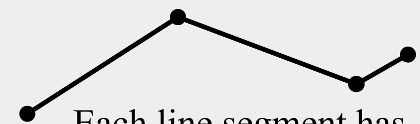


Basic Idea: Represent the time series as a sequence of straight lines.

Karl Friedrich Gauss

1777 - 1855

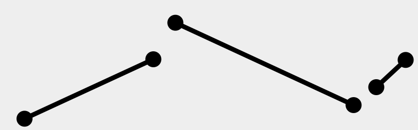
Lines could be **connected**, in which case we are allowed *N*/2 lines



Each line segment has
- `length`
- `left_height`

(`right_height` can be inferred by looking at the next segment)

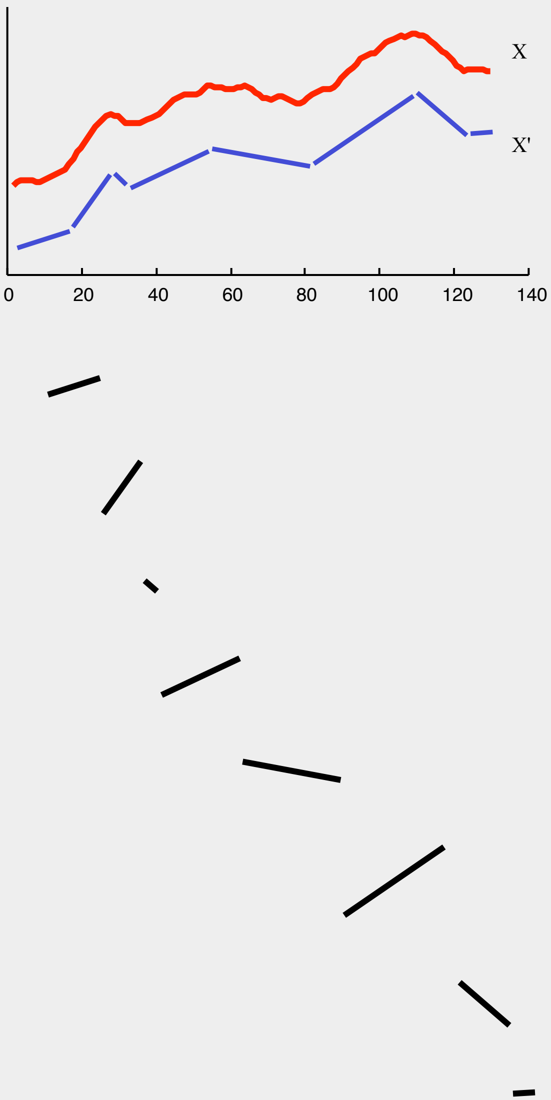
If lines are **disconnected**, we are allowed only *N*/3 lines



Each line segment has
- `length`
- `left_height`
- `right_height`

Personal experience on dozens of datasets suggest **disconnected** is better. Also only **disconnected** allows a lower bounding Euclidean approximation

Piecewise Linear Approximation II



How do we obtain the Piecewise Linear Approximation?

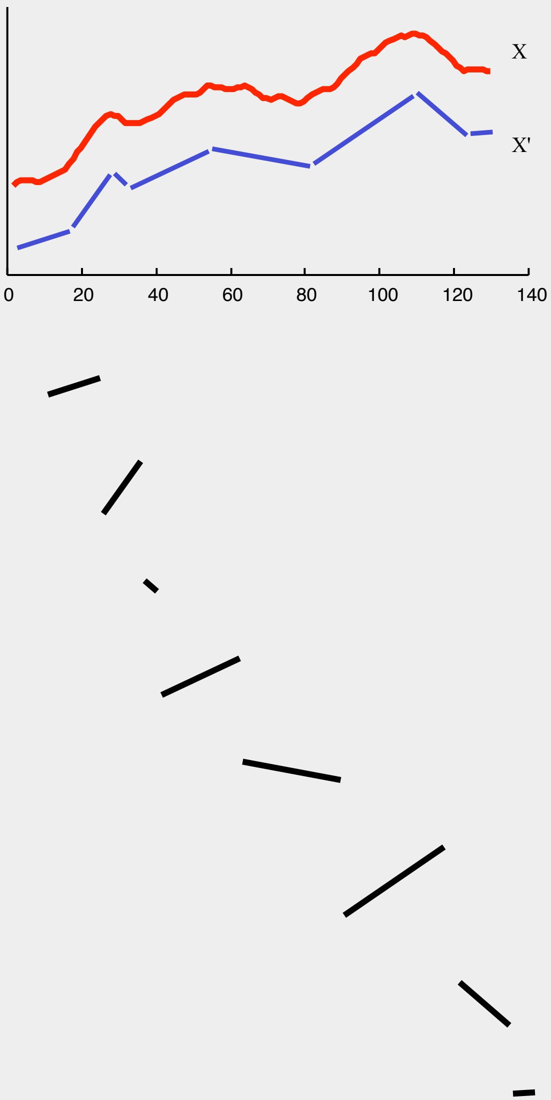
Optimal Solution is $O(n^2N)$, which is too slow for data mining.

A vast body on work on faster heuristic solutions to the problem can be classified into the following classes:

- **Top-Down**
- **Bottom-Up**
- **Sliding Window**
- **Other** (genetic algorithms, randomized algorithms, Bspline wavelets, MDL etc)

Extensive empirical evaluation* of all approaches suggest that Bottom-Up is the best approach overall.

Piecewise Linear Approximation III

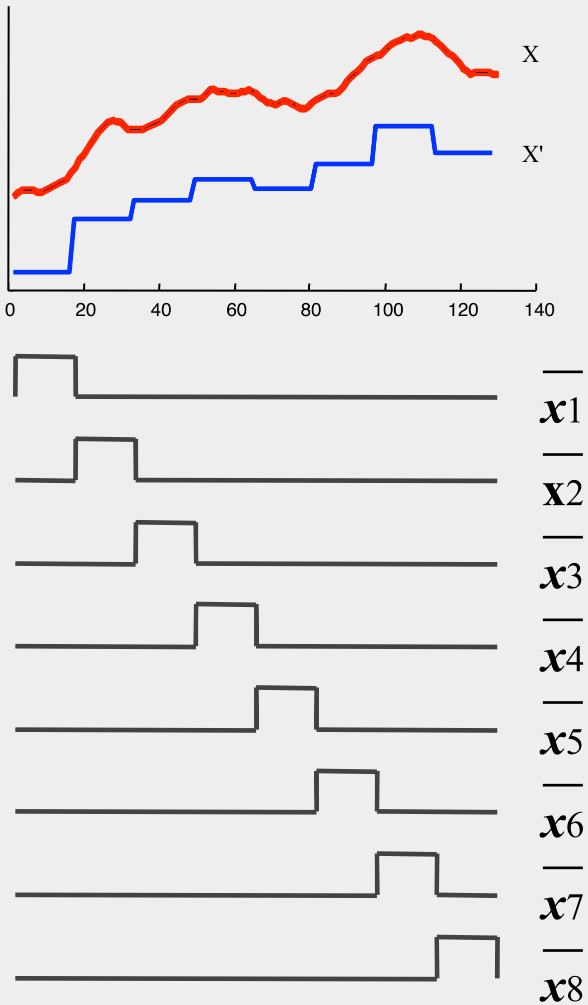


Pros and Cons of PLA as a time series representation.

• Good ability to compress natural signals.
• Fast linear time algorithms for PLA exist.
• Able to support some interesting non-Euclidean similarity measures. Including weighted measures, relevance feedback, fuzzy queries…
•Already widely accepted in some communities (ie, biomedical)

• Not (currently) indexable by any data structure (but does allows fast sequential scanning).

Piecewise Aggregate Approximation I



Basic Idea: Represent the time series as a sequence of box basis functions.

Note that each box is the same length.

$$\bar{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j$$

Given the reduced dimensionality representation we can calculate the approximate Euclidean distance as...

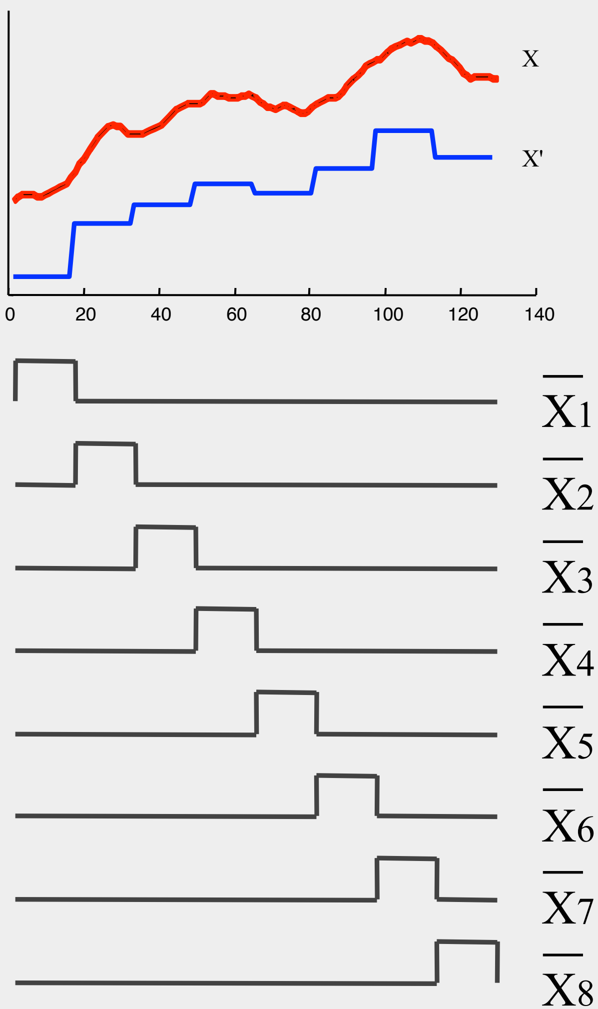
$$DR(\bar{X},\bar{Y}) \equiv \sqrt{\tfrac{n}{N}} \sqrt{\sum_{i=1}^{N} \left(\bar{x}_i - \bar{y}_i\right)^2}$$

This measure is provably lower bounding.

Independently introduced by two authors

- Keogh, Chakrabarti, Pazzani & Mehrotra, KAIS (2000) / Keogh & Pazzani PAKDD *April* 2000
- Byoung-Kee Yi, Christos Faloutsos, VLDB *September* 2000

Piecewise Aggregate Approximation II



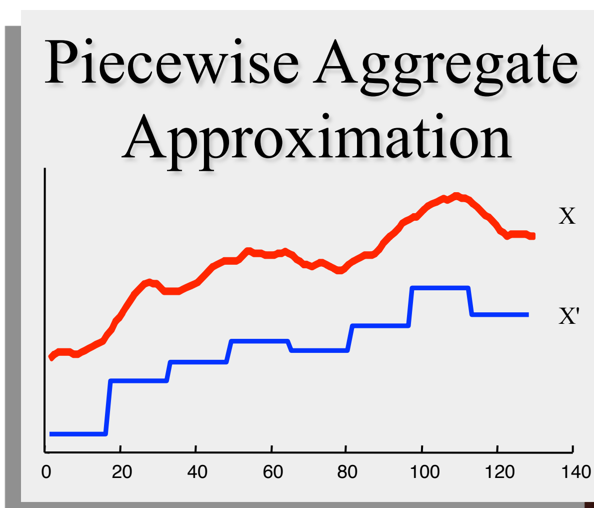
Pros and Cons of PAA as a time series representation.

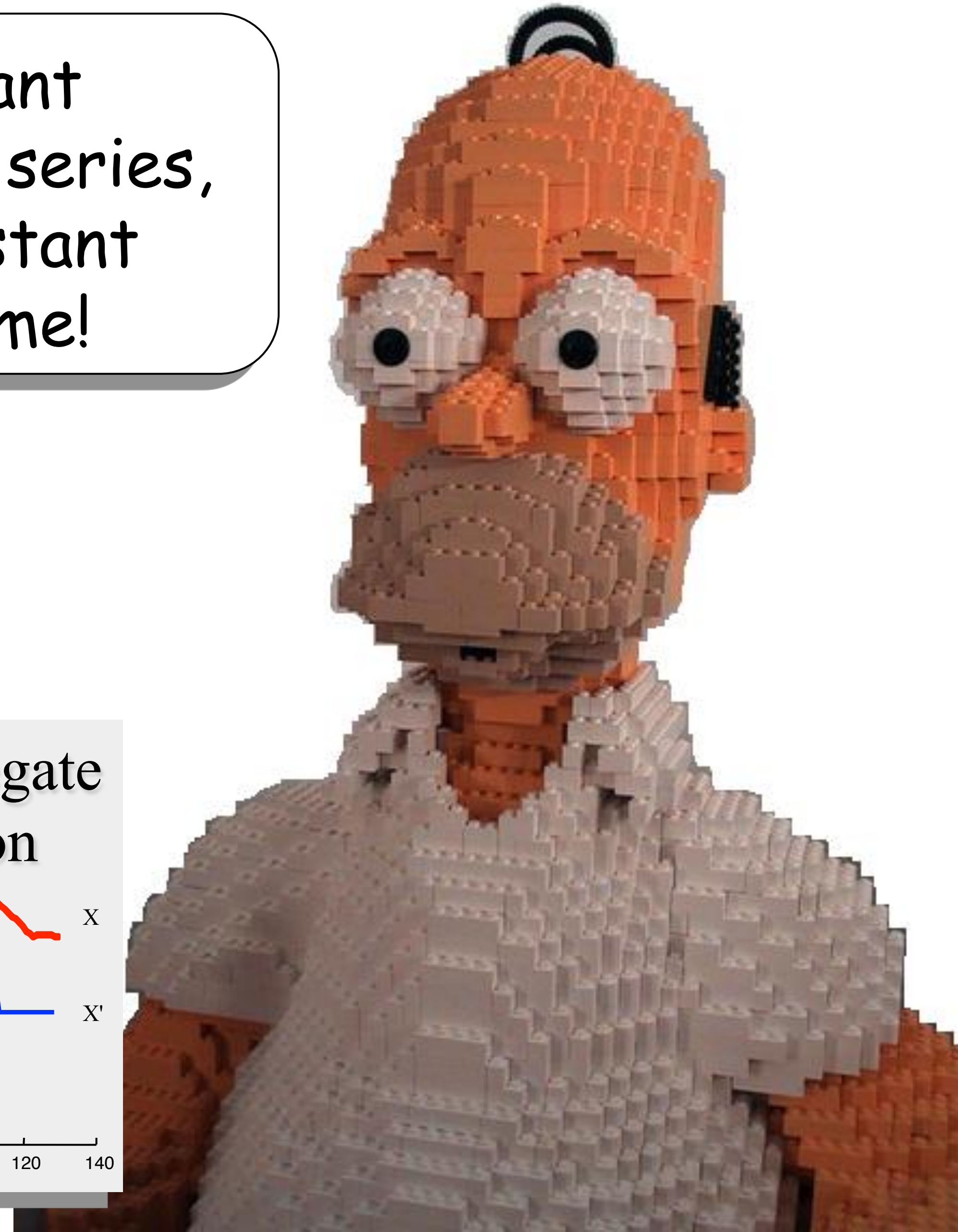
- *Extremely* fast to calculate
- As efficient as other approaches (empirically)
- Support queries of arbitrary lengths
- Can support any Minkowski metric@
- Supports non Euclidean measures
- Supports weighted Euclidean distance
- Can be used to allow indexing of DTW and uniform scaling*
- Simple! Intuitive!

- If visualized directly, looks ascetically unpleasing.

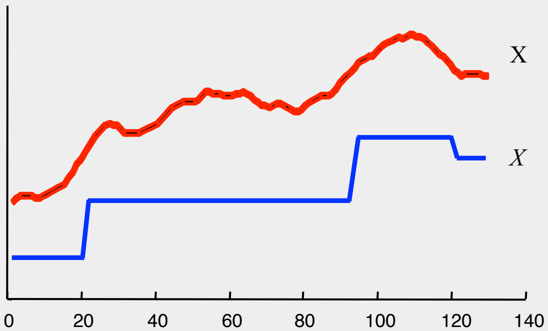
A Completely Pointless Slide

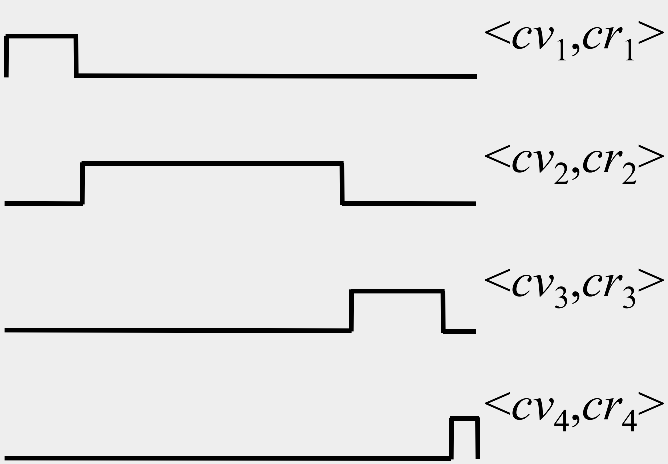
A piecewise constant approximate of a time series, and a piecewise constant approximation of me!





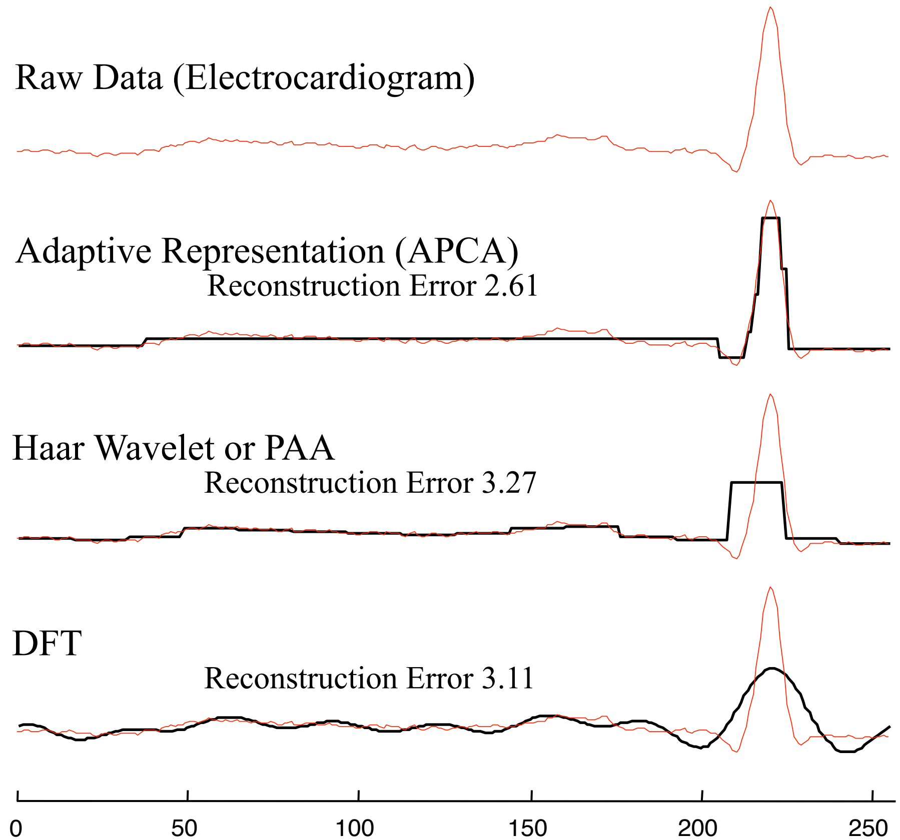
Adaptive Piecewise Constant Approximation I





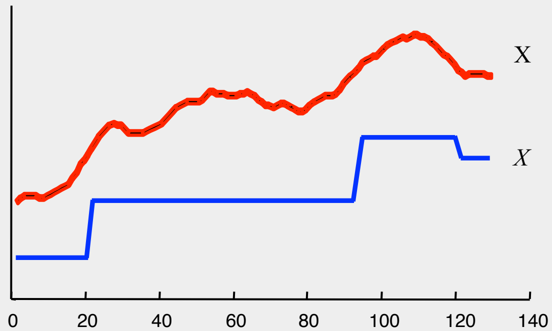
Basic Idea: Generalize PAA to allow the piecewise constant segments to have arbitrary lengths.

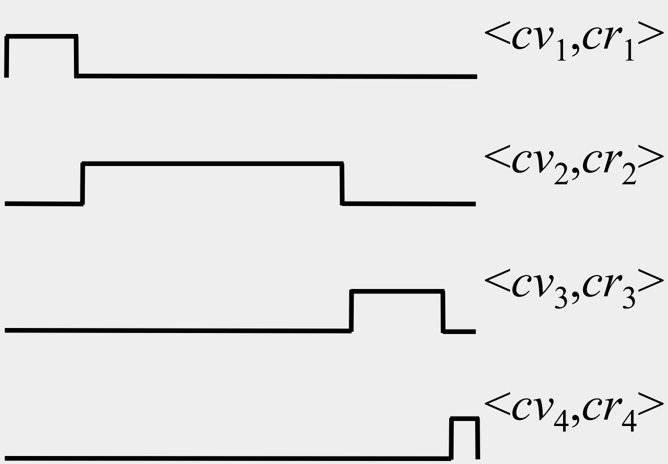
Note that we now need 2 coefficients to represent each segment, its value and its length.



The intuition is this, many signals have little detail in some places, and high detail in other places. APCA can *adaptively* fit itself to the data achieving better approximation.

Adaptive Piecewise Constant Approximation II





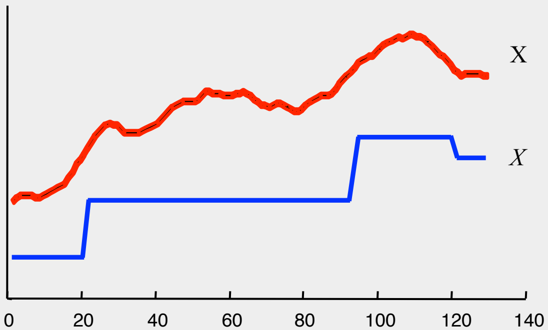
The high quality of the APCA had been noted by many researchers.
However it was believed that the representation could not be indexed because some coefficients represent values, and some represent lengths.

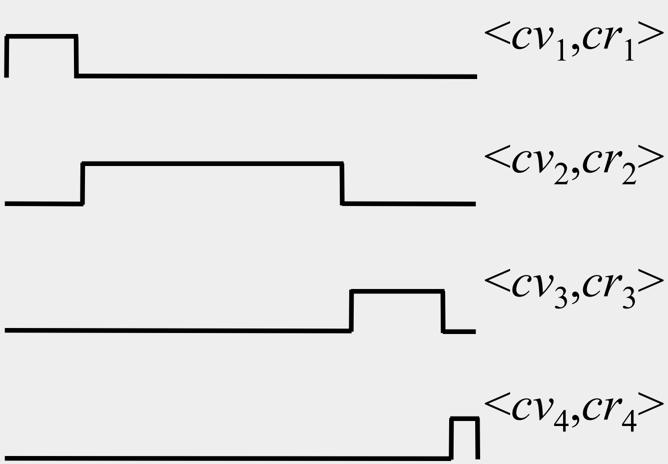
However an indexing method was discovered!

(SIGMOD 2001 best paper award)

Unfortunately, it is non-trivial to understand and implement and thus has only been reimplemented once or twice (In contrast, more than 50 people have reimplemented PAA).

Adaptive Piecewise Constant Approximation III



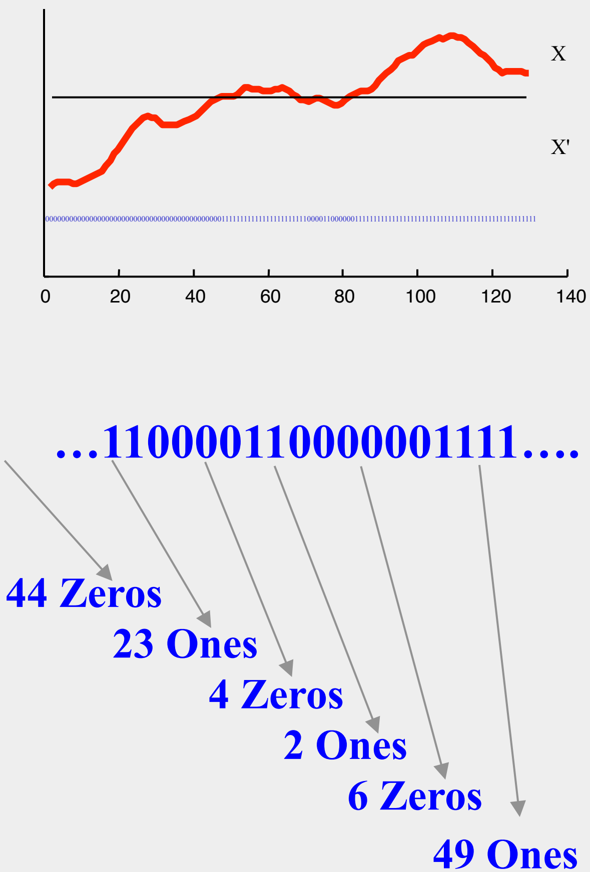


- Pros and Cons of APCA as a time series representation.

- Fast to calculate O(*n*).
- *More* efficient as other approaches (on some datasets).
- Support queries of arbitrary lengths.
- Supports non Euclidean measures.
- Supports weighted Euclidean distance.
- Support fast exact queries , and even faster approximate queries on the same data structure.

- Somewhat complex implementation.
- If visualized directly, looks ascetically unpleasing.

Clipped Data

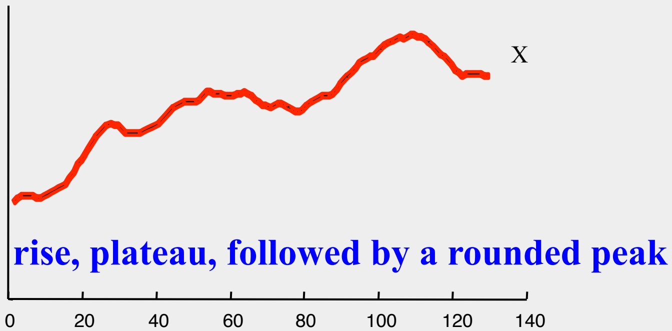


44 Zeros|23|4|2|6|49

No details available, this paper is in this conference

Bagnall, A.J. and Janacek, G.A., "Clustering time series from ARMA models with clipped data", *In International Conference on Knowledge Discovery in Data and Data Mining (ACM SIGKDD 2004) Accepted*, Seattle, USA, 2004

Natural Language



rise,

plateau,

followed by a rounded peak

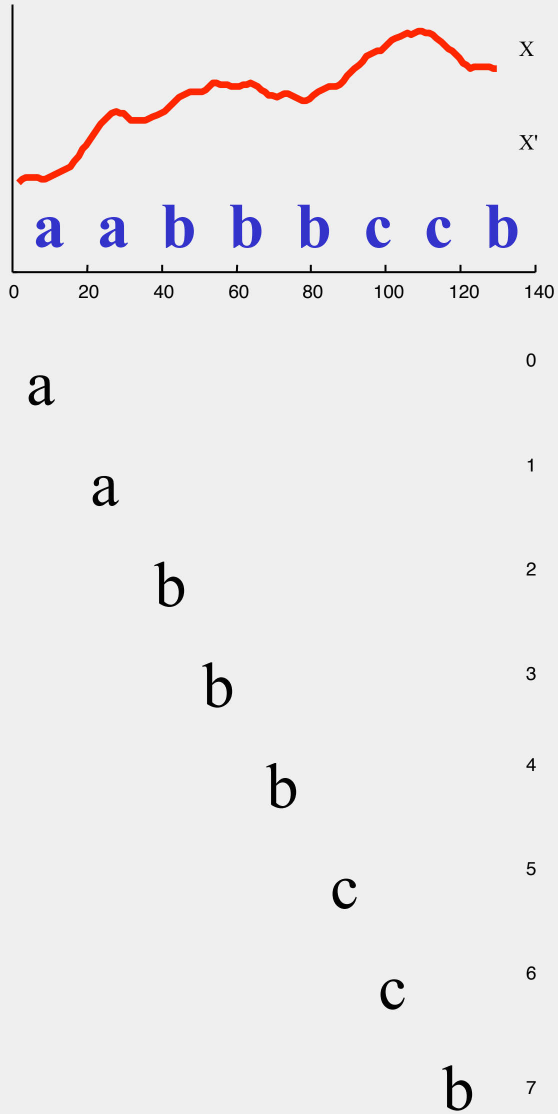
• Pros and Cons of natural language as a time series representation.

• The most intuitive representation!
• Potentially a good representation for low bandwidth devices like text-messengers

• Difficult to evaluate.

To the best of my knowledge only one group is working seriously on this representation. They are the University of Aberdeen SUMTIME group, headed by Prof. Jim Hunter.

Symbolic Approximation I



Basic Idea: Convert the time series into an alphabet of discrete symbols. Use string indexing techniques to manage the data.

Potentially an interesting idea, but all work thus far are very ad hoc.

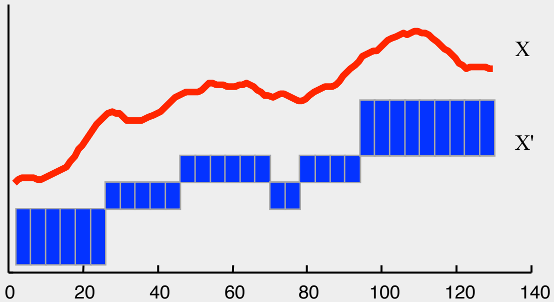
Pros and Cons of Symbolic Approximation as a time series representation.

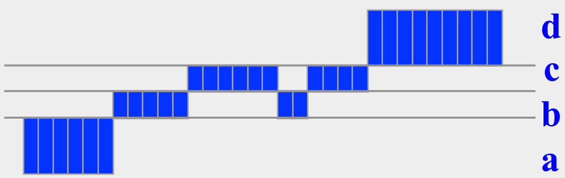
• Potentially, we could take advantage of a wealth of techniques from the very mature field of string processing and bioinformatics.

• It is not clear how we should discretize the times series (discretize the values, the slope, shapes? How big of an alphabet? etc).

• There are more than 210 different variants of this, at least 35 in data mining conferences.

SAX: **S**ymbolic **A**ggregate appro**X**imation

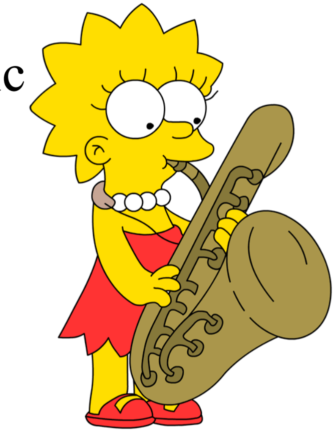




aaaaaabbbbbccccccbbccccdddddddd

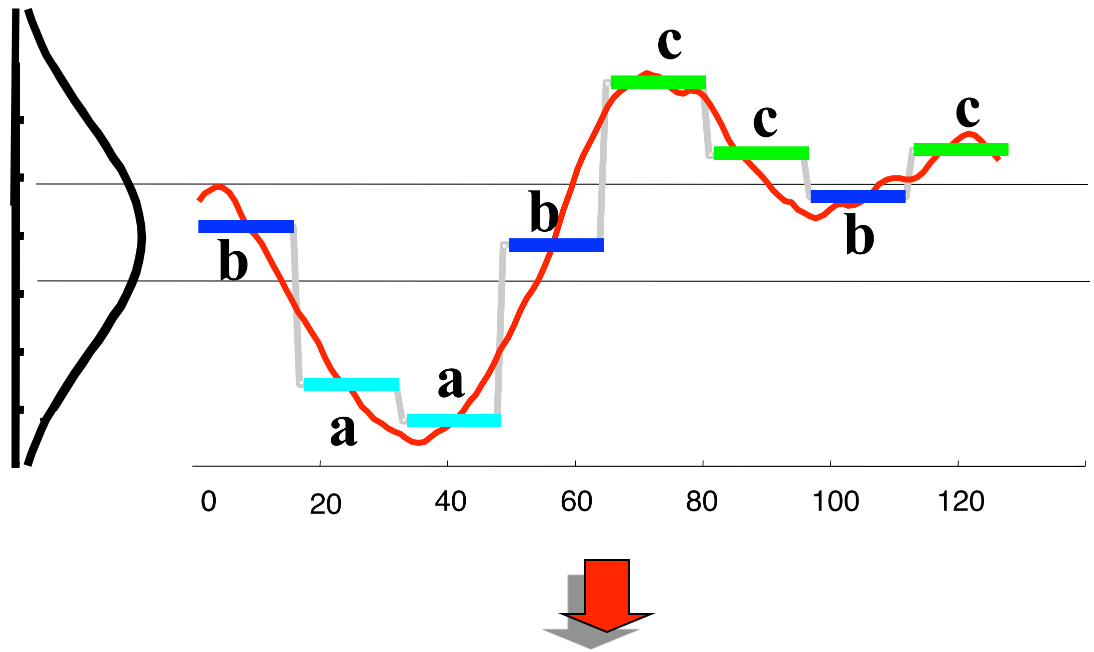
SAX allows (for the first time) a symbolic representation that allows

- Lower bounding of Euclidean distance
- Dimensionality Reduction
- Numerosity Reduction



Jessica Lin

1976-



baabccbc

Comparison of all Dimensionality Reduction Techniques

- We have already compared features (Does representation X allow weighted queries, queries of arbitrary lengths, is it simple to implement…

- We can compare the indexing efficiency. How long does it take to find the best answer to out query.

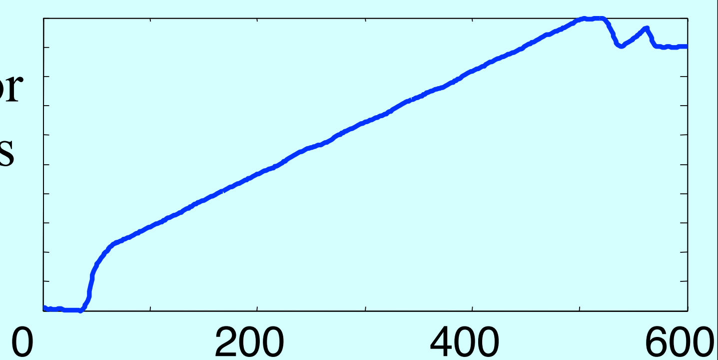
- It turns out that the fairest way to measure this is to measure the number of times we have to retrieve an item from disk.

Data Bias

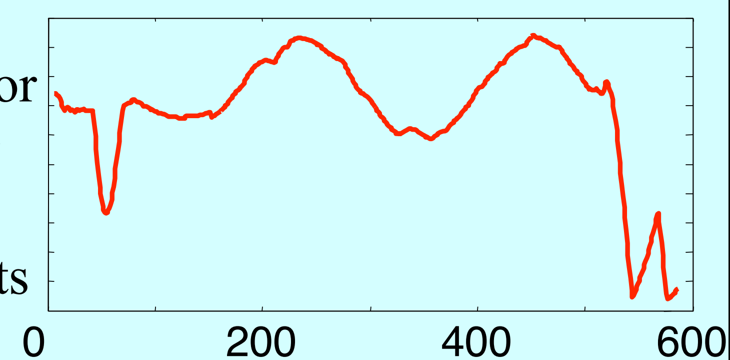
Definition: *Data bias* is the conscious or unconscious use of a particular set of testing data to confirm a desired finding.

Example: Suppose you are comparing Wavelets to Fourier methods, the following datasets will produce drastically different results…

Good for wavelets bad for Fourier



Good for Fourier bad for wavelets



Example of Data Bias: Whom to Believe?

For the task of indexing time series for similarity search, which representation is best, the Discrete Fourier Transform (DFT), or the Discrete Wavelet Transform (Haar)?

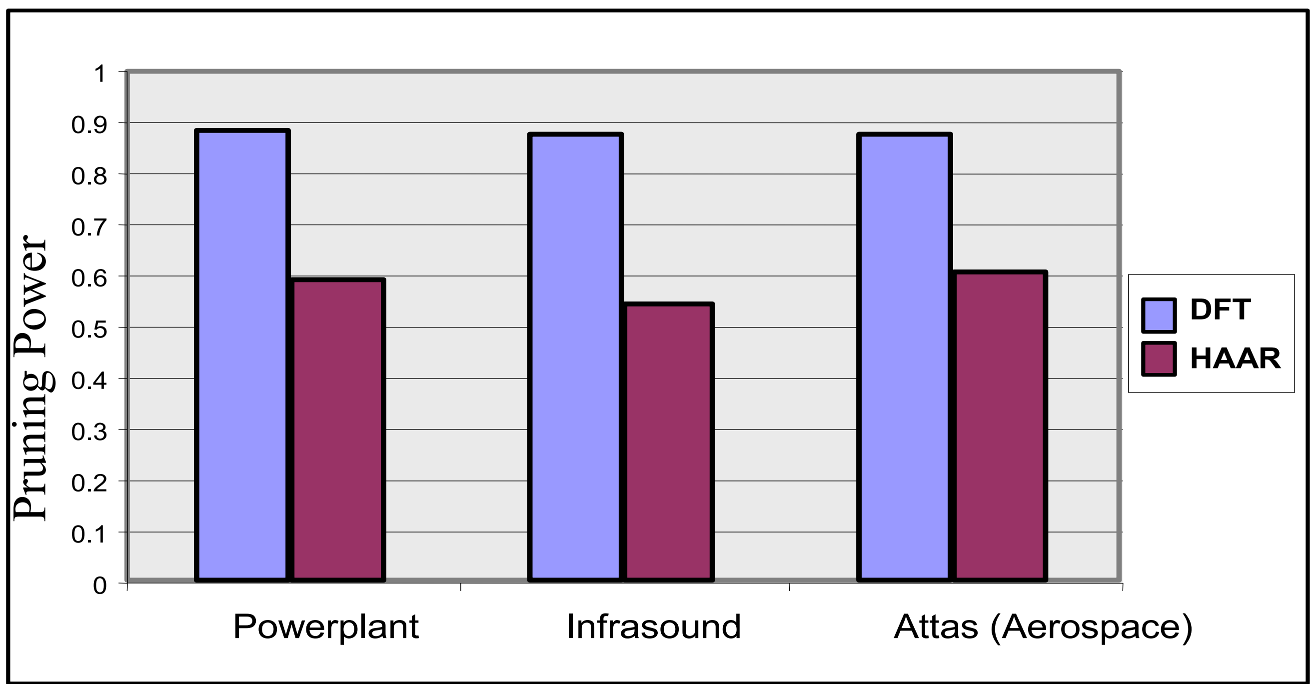
- "*Several wavelets outperform the DFT*".
- "*DFT-based and DWT-based techniques yield comparable results*".
- "*Haar wavelets perform slightly better that DFT*"
- "*DFT filtering performance is superior to DWT*"

Example of Data Bias: Whom to Believe II?

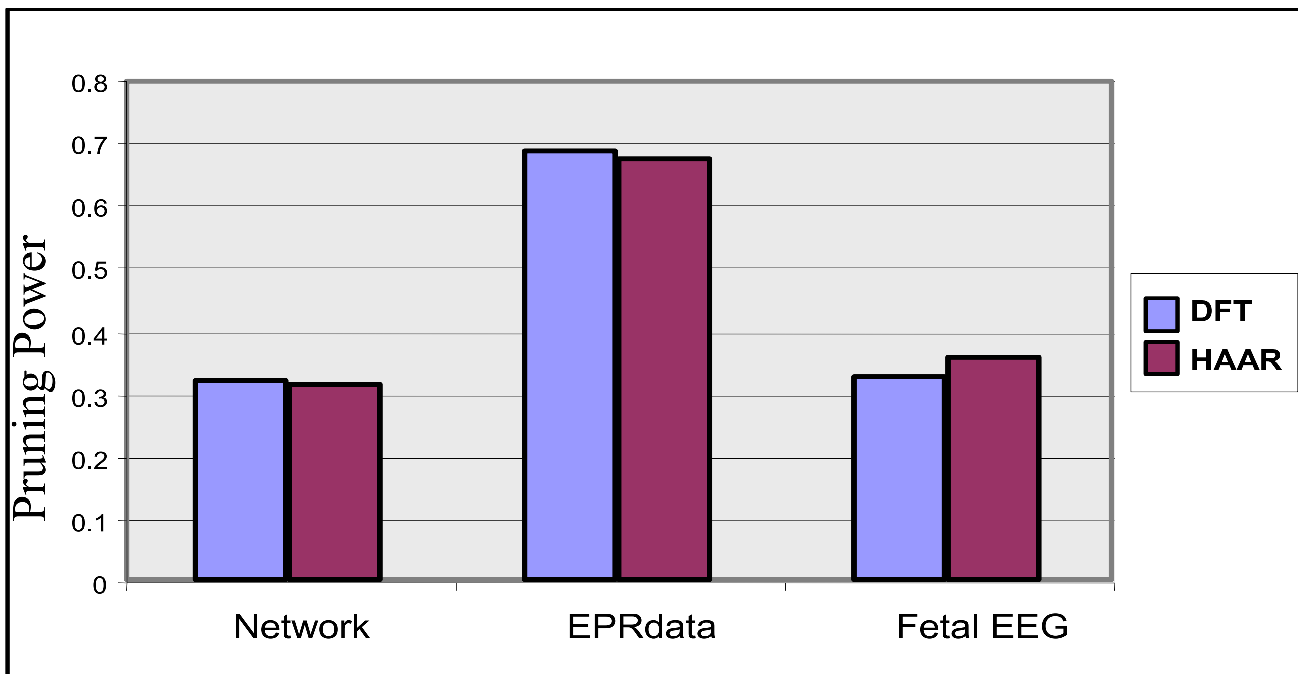
To find out who to believe (if anyone) we performed an extraordinarily careful and comprehensive set of experiments. For example…

- We used a quantum mechanical device generate random numbers.
- We averaged results over 100,000 experiments!
- For fairness, we use the same (randomly chosen) subsequences for both approaches.

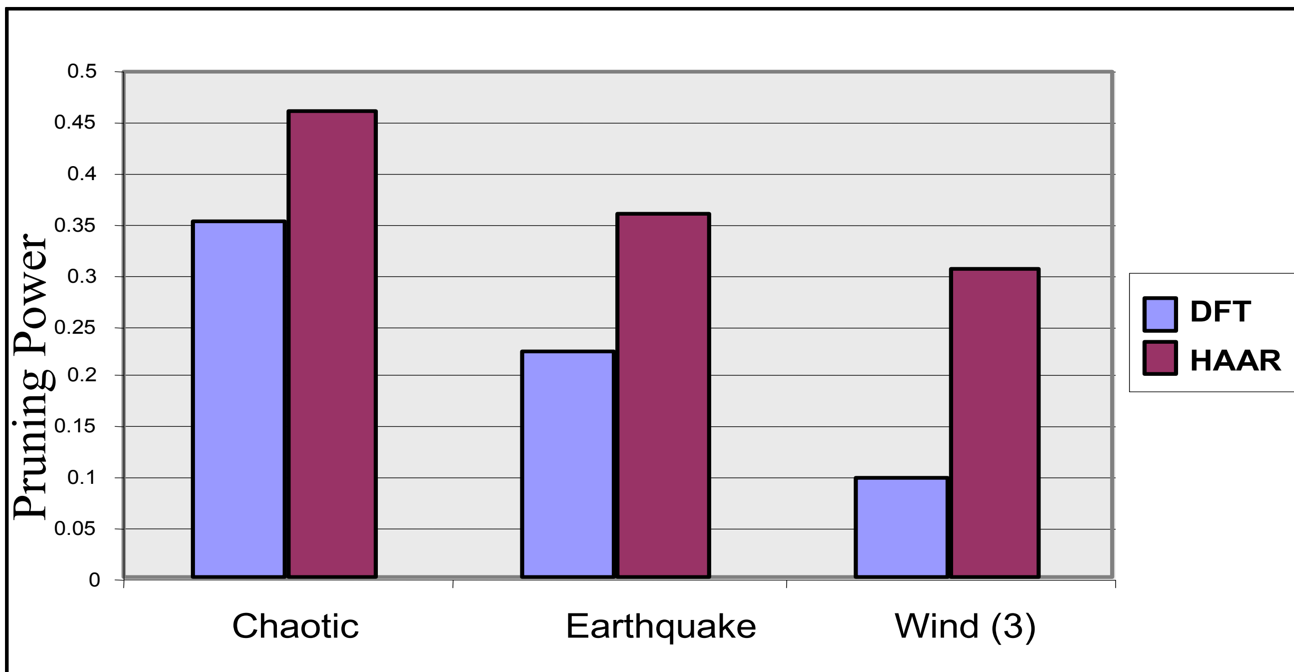
I tested on the Powerplant, Infrasound and Attas datasets, and I know DFT outperforms the Haar wavelet



Stupid Flanders! I tested on the Network, ERPdata and Fetal EEG datasets and I know that there is no real difference between DFT and Haar

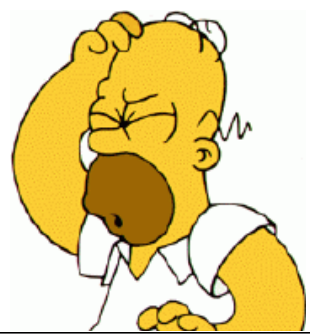


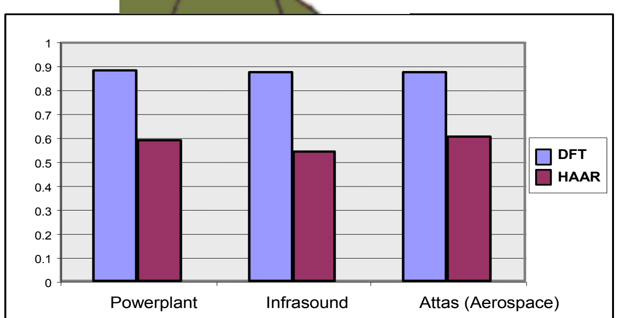
Those two clowns are both wrong!
I tested on the Chaotic, Earthquake and Wind datasets, and I am sure that the Haar wavelet outperforms the DFT

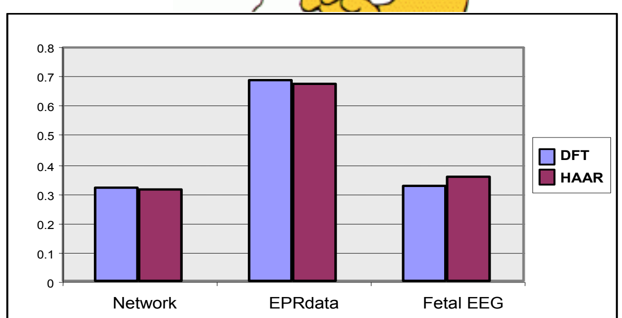


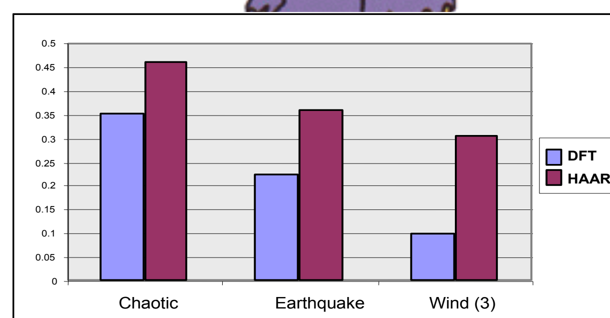
The Bottom Line

Any claims about the relative performance of a time series indexing scheme that is empirically demonstrated on only 2 or 3 datasets are worthless.









So which is really the best technique?

I experimented with all the techniques (DFT, DCT, Chebyshev, PAA, PLA, PQA, APCA, DWT (most wavelet types), SVD) on **65** datasets, and as a sanity check, Michail Vlachos independently implemented and tested on the same **65** datasets.

On average, they are all about the same. In particular, on 80% of the datasets they are all within 10% of each other.

If you want to pick a representation, don’t do so based on the reconstruction error, do so based on the features the representation has.

On bursty datasets* APCA can be significantly better

Lets take a tour of other time series problems

- Before we do, let us briefly revisit SAX, since it has some implications for the other problems…

Exploiting Symbolic Representations of Time Series

- One central theme of this tutorial is that *lowerbounding* is a very useful property. (recall the *lower bounds* of DTW /uniform scaling, also recall the importance of lower bounding dimensionality reduction techniques).
- Another central theme is that dimensionality reduction is very important. That's why we spend so long discussing DFT, DWT, SVD, PAA etc.

- Until last year there was no lowerbounding, dimensionality reducing representation of time series. In the next slide, let us think about what it means to have such a representation…

Exploiting Symbolic Representations of Time Series

• If we had a lowerbounding, dimensionality reducing representation of time series, we could…

• Use data structures that are only defined for discrete data, such as suffix trees.

• Use algorithms that are only defined for discrete data, such as hashing, association rules etc

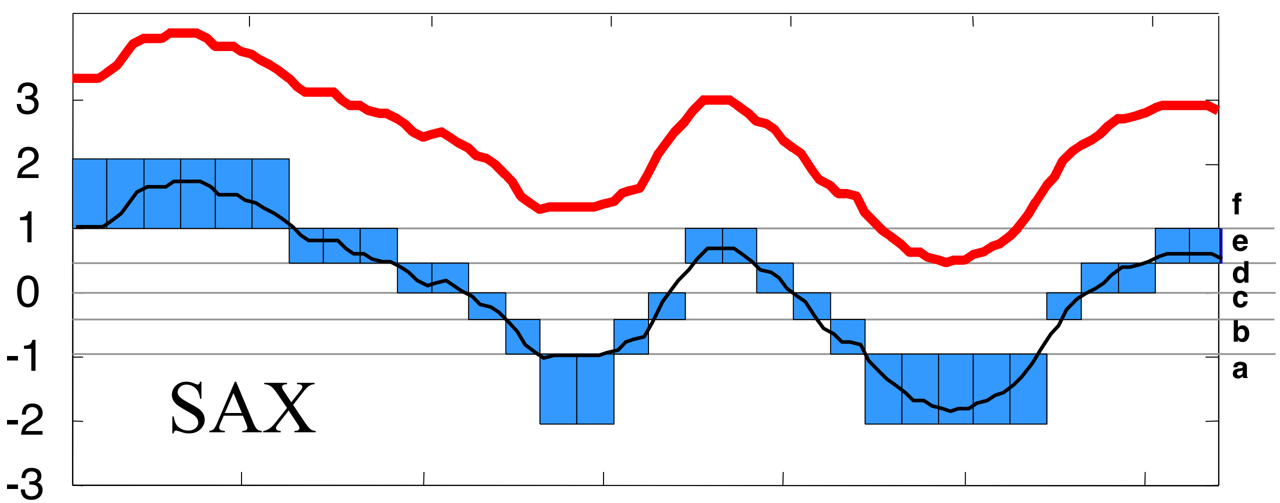
• Use definitions that are only defined for discrete data, such as Markov models, probability theory

• More generally, we could utilize the vast body of research in text processing and bioinformatics

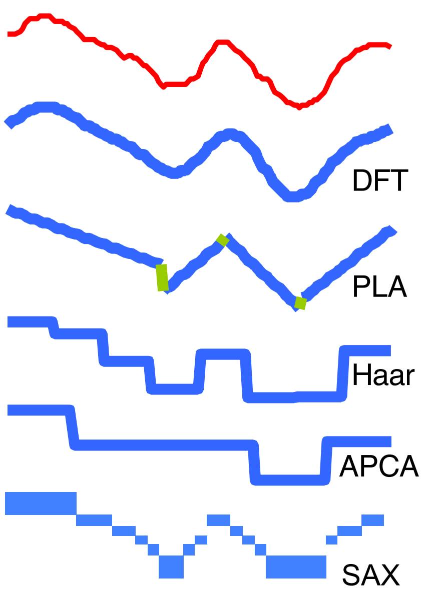
Exploiting Symbolic Representations of Time Series

There is now a lower bounding dimensionality reducing time series representation! It is called SAX (**S**ymbolic **A**ggregate Appro**X**imation)

I expect SAX to have a major impact on time series data mining in the coming years…



ffffffeeeddcbaabceedcbaaaaacddee



Anomaly (interestingness) detection

We would like to be able to discover surprising (unusual, interesting, anomalous) patterns in time series.

Note that we don't know in advance in what way the time series might be surprising

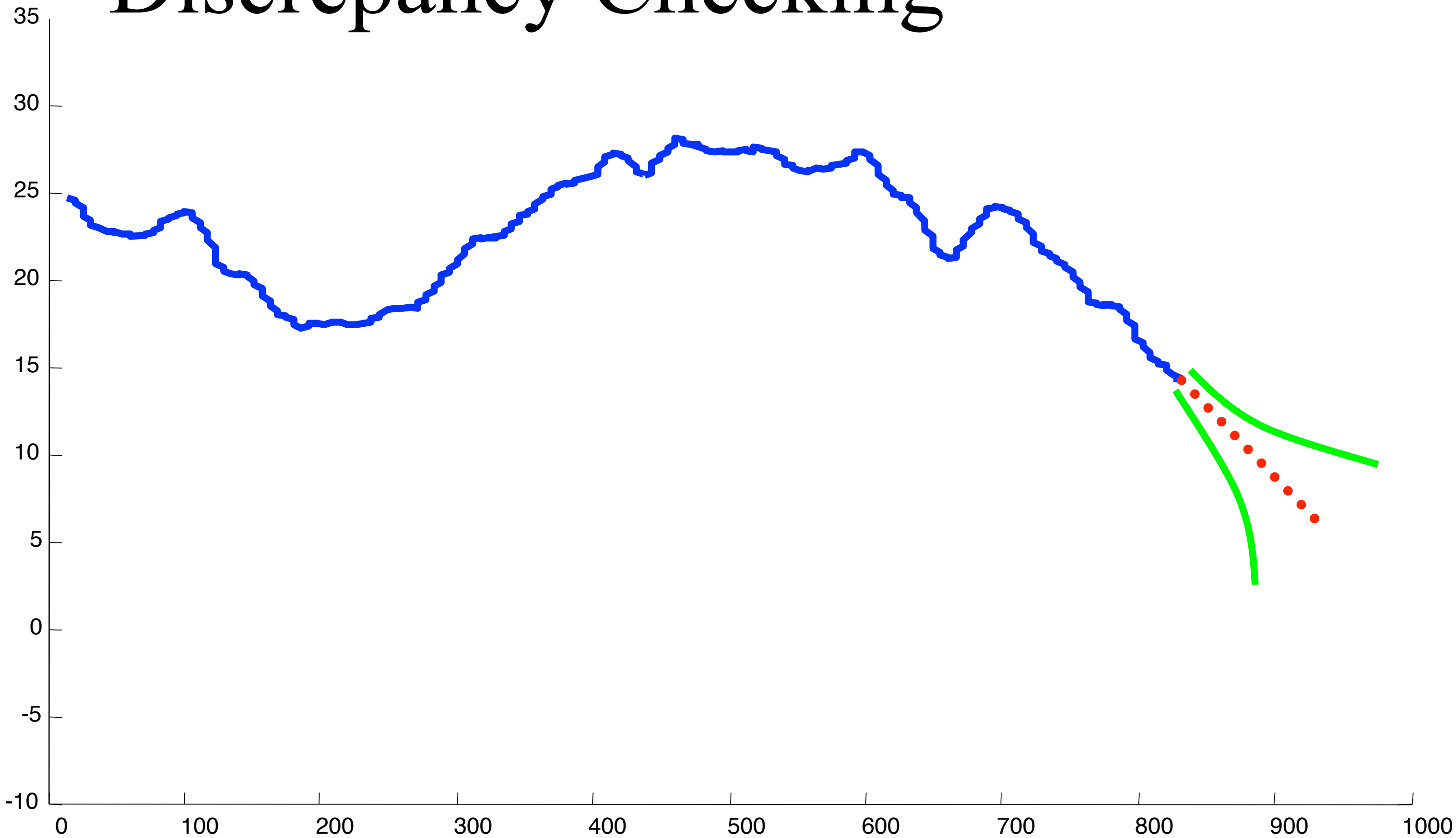
Also note that "surprising" is very context dependent, application dependent, subjective etc.

Simple Approaches I

Limit Checking

Simple Approaches II

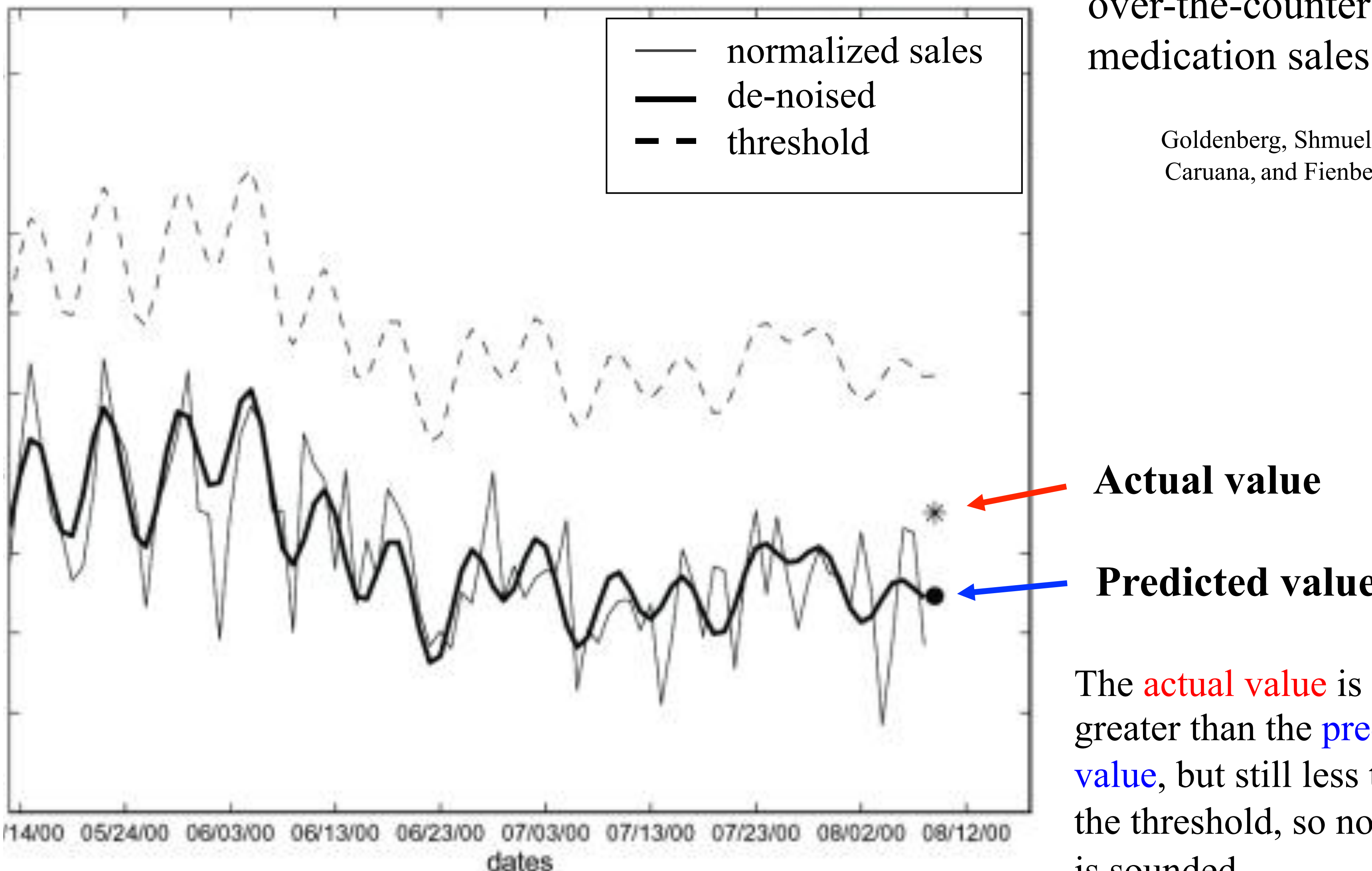
Discrepancy Checking



Discrepancy Checking: Example

Early statistical detection of anthrax outbreaks by tracking over-the-counter medication sales

Goldenberg, Shmueli, Caruana, and Fienberg



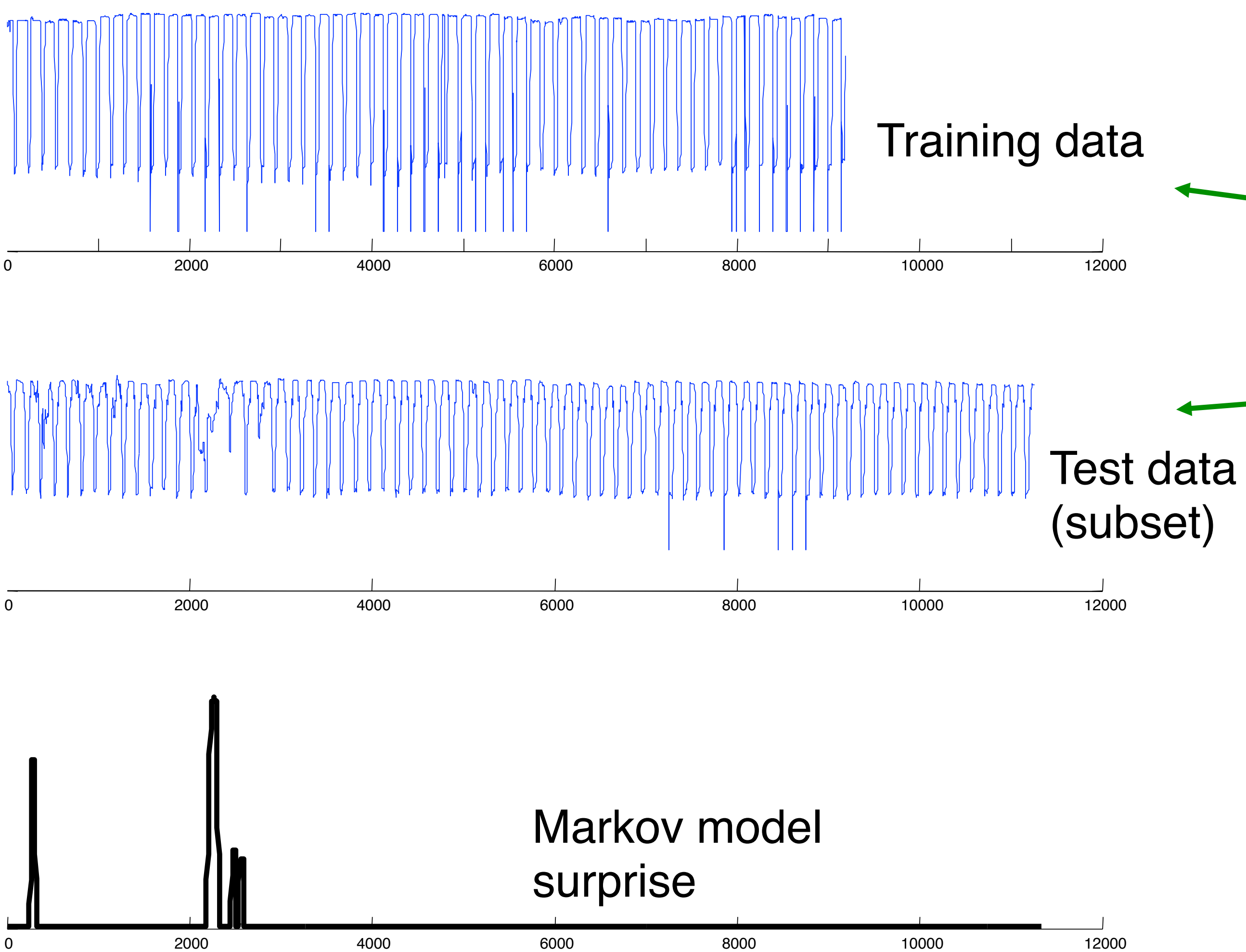
The actual value is greater than the predicted value, but still less than the threshold, so no alarm is sounded.

- Note that this problem has been solved for text strings

- You take a set of text which has been labeled "normal", you learn a Markov model for it.

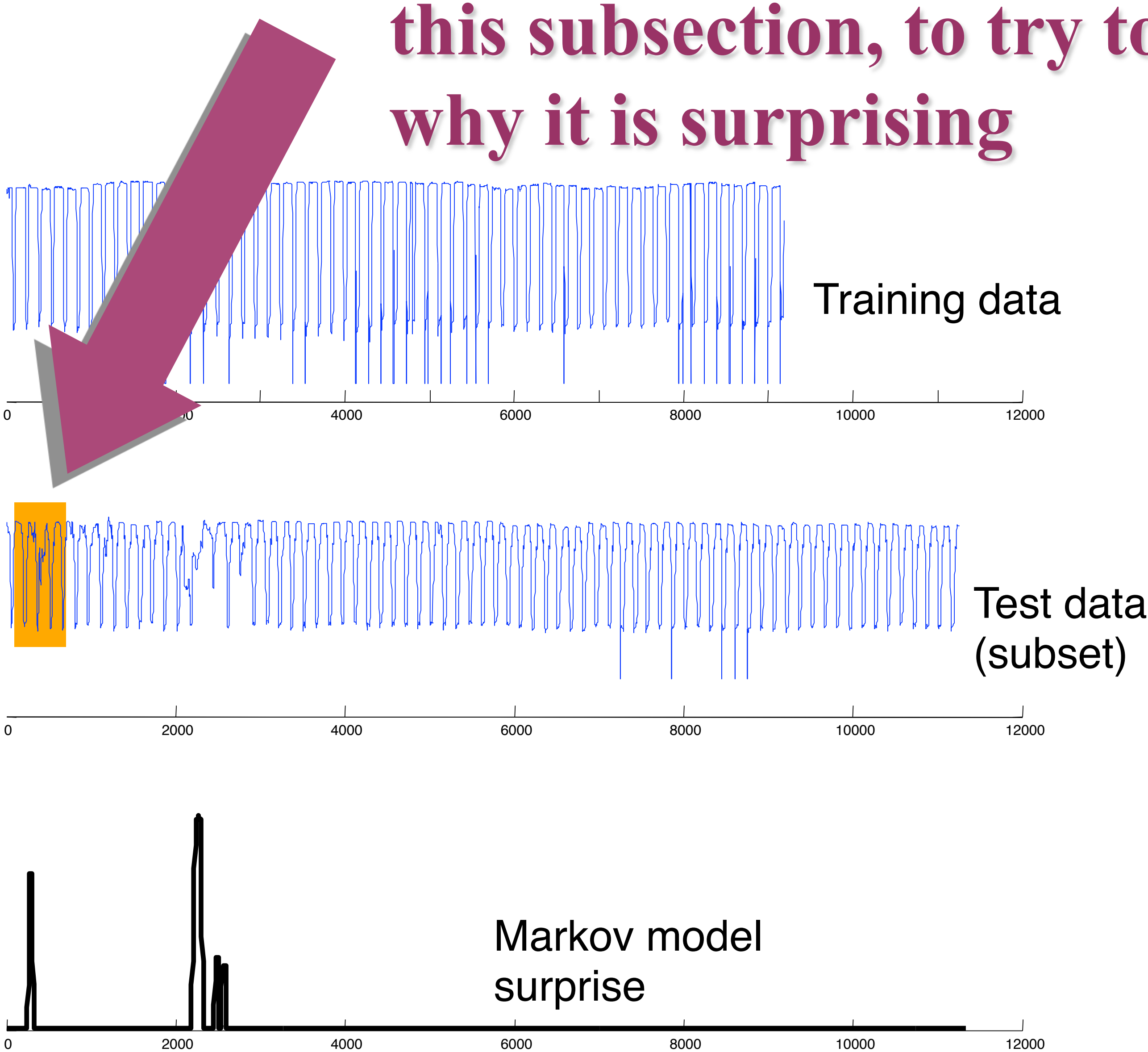
- Then, any future data that is not modeled well by the Markov model you annotate as *surprising*.

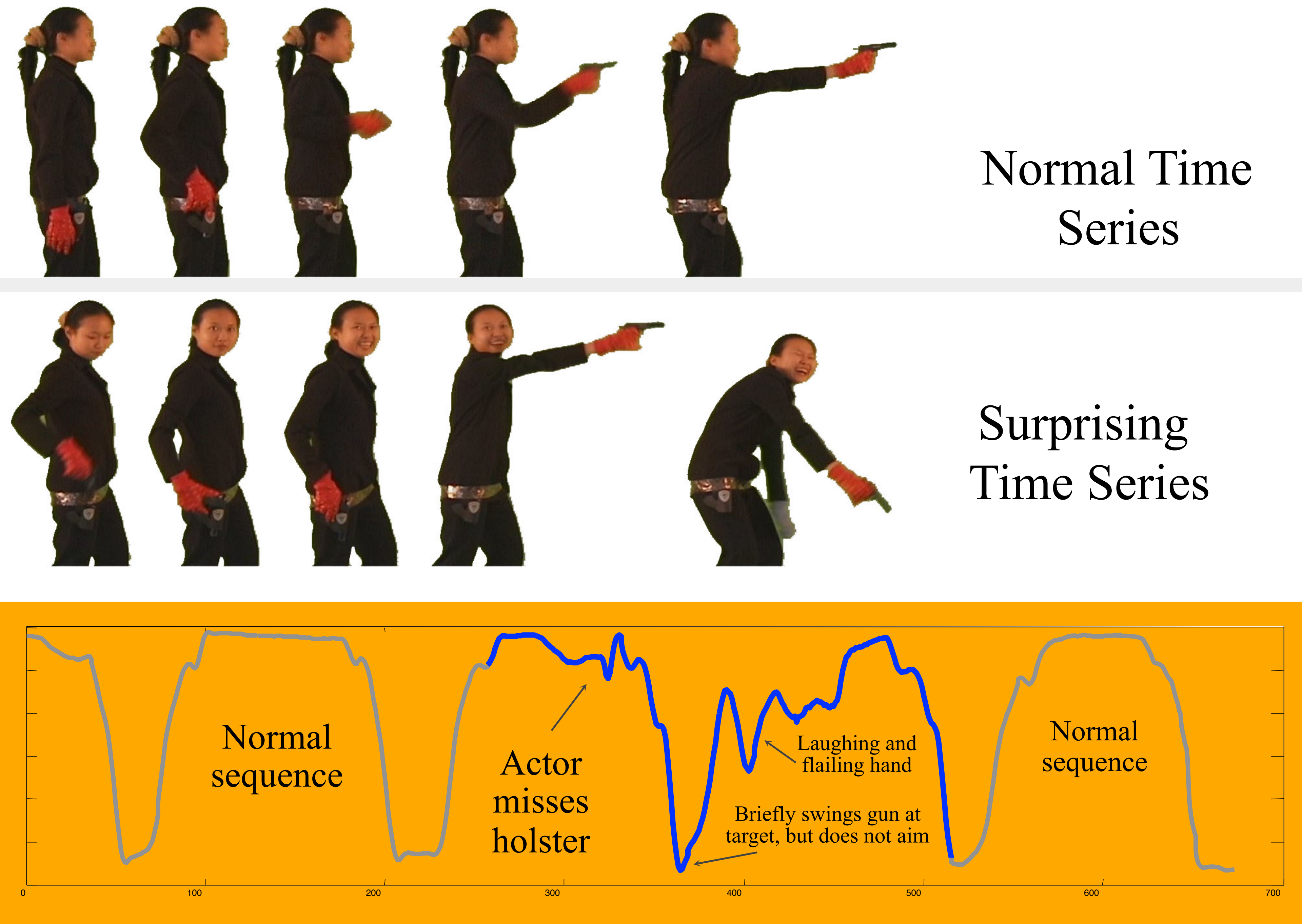
- Since we have just seen that we can convert time series to text (i.e SAX). Lets us quickly see if we can use Markov models to find surprises in time series…



These were converted to the symbolic representation. I am showing the original data for simplicity

In the next slide we will zoom in on this subsection, to try to understand why it is surprising




Normal Time Series
Surprising Time Series
Normal sequence
Actor misses holster
Laughing and flailing hand
Briefly swings gun at target, but does not aim
Normal sequence
0
100
200
300
400
500
600
700

Anomaly (interestingness) detection

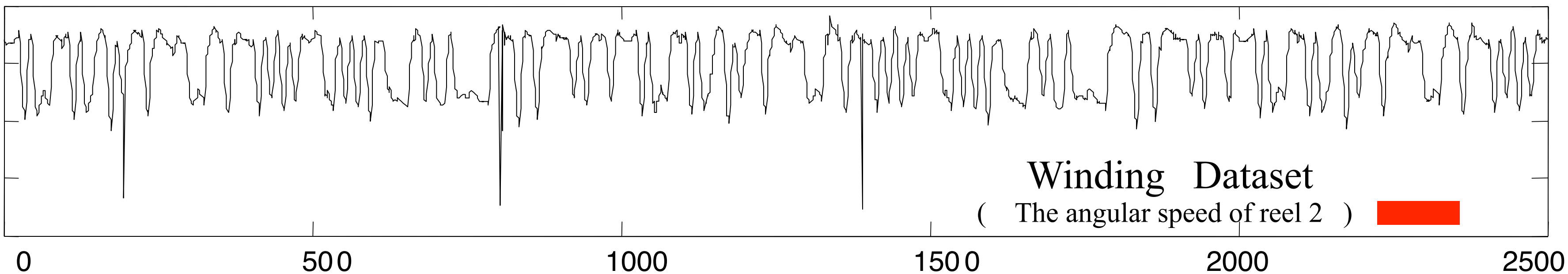
In spite of the nice example in the previous slide, the anomaly detection problem is wide open.

How can we find interesting patterns…

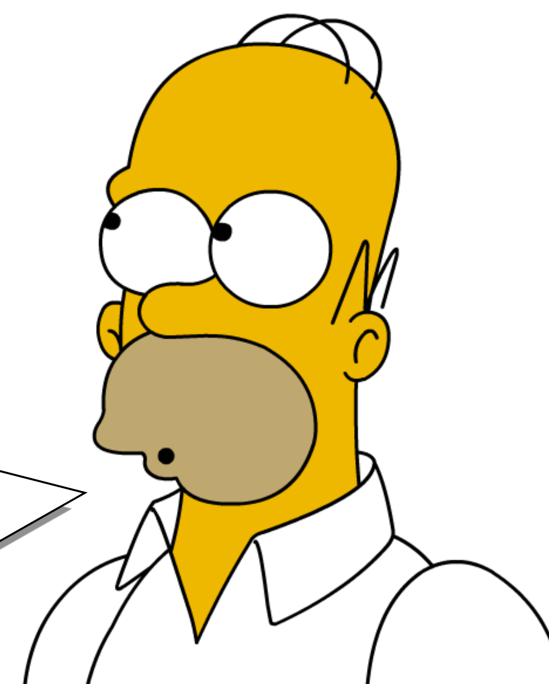
- Without (or with very few) false positives…
- In truly massive datasets...
- In the face of concept drift…
- With human input/feedback…
- With annotated data…

Time Series Motif Discovery

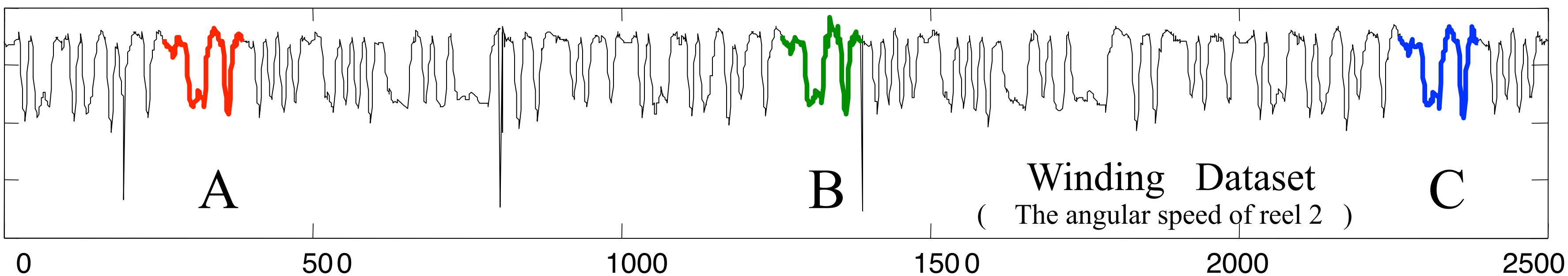
(finding repeated patterns)

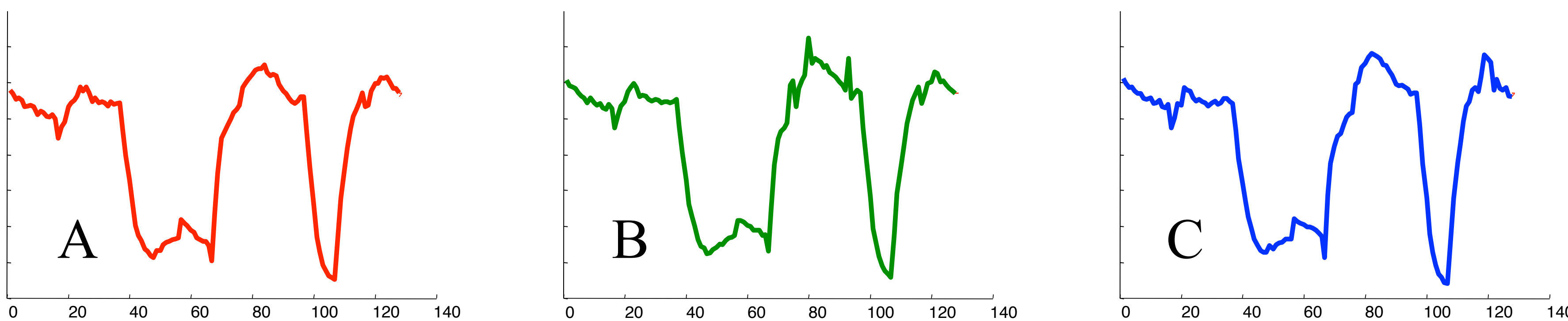


Are there any repeated patterns, of about this length ▬ in the above time series?



Time Series Motif Discovery
(finding repeated patterns)





Why Find Motifs?

· Mining **association rules** in time series requires the discovery of motifs. These are referred to as *primitive shapes* and *frequent patterns*.

· Several time series **classification algorithms** work by constructing typical prototypes of each class. These prototypes may be considered motifs.

· Many time series **anomaly/interestingness detection** algorithms essentially consist of modeling normal behavior with a set of typical shapes (which we see as motifs), and detecting future patterns that are dissimilar to all typical shapes.

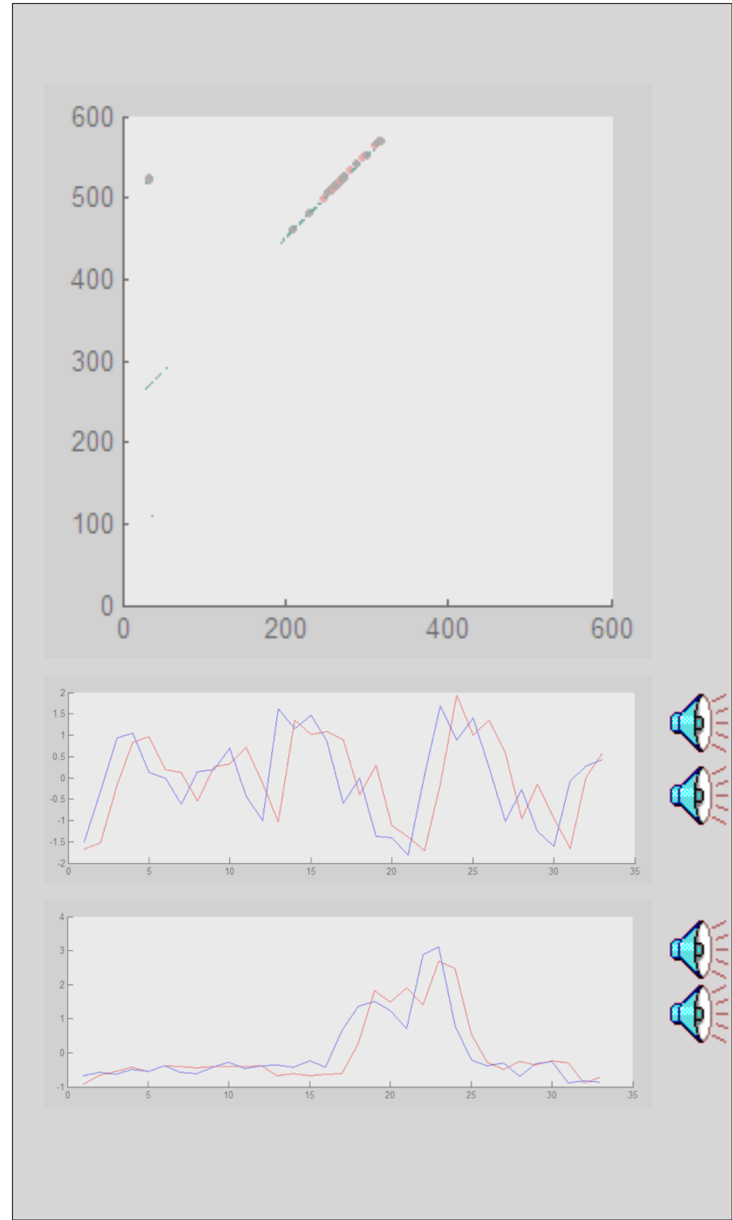
· In **robotics**, Oates et al., have introduced a method to allow an autonomous agent to generalize from a set of qualitatively different *experiences* gleaned from sensors. We see these "*experiences*" as motifs.

· In **medical data mining**, Caraca-Valente and Lopez-Chavarrias have introduced a method for characterizing a physiotherapy patient's recovery based of the discovery of *similar patterns*. Once again, we see these "*similar patterns*" as motifs.

• **Animation and video capture…** (Tanaka and Uehara, Zordan and Celly)

Motifs in Music

Radio Jingle

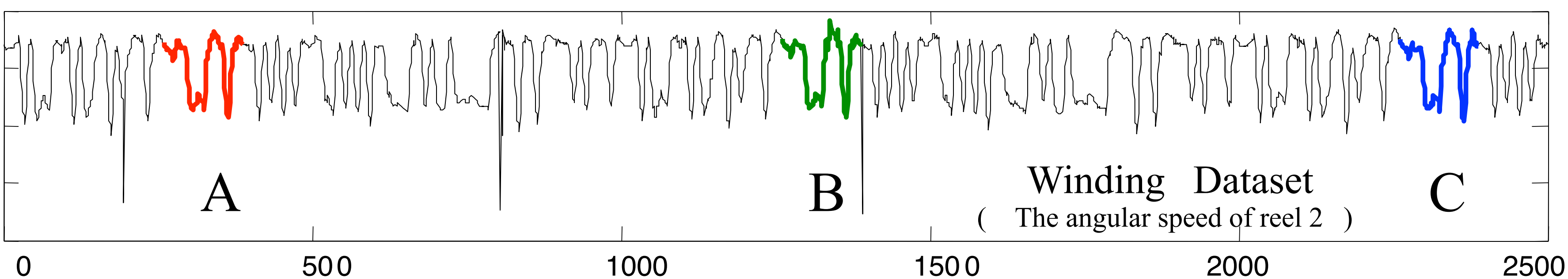


- Single channel (mono) 225000 samples at sample rate of 6000 samples/sec, 32bits per sample.
- Pre-processing: Absolute-valued and down-sampled to total of 600 samples and new sample rate of 16 samples/sec.
- 400 projections with instance length equal to 2 seconds of sample. w=16, a=8.

Motifs Discovery Challenges

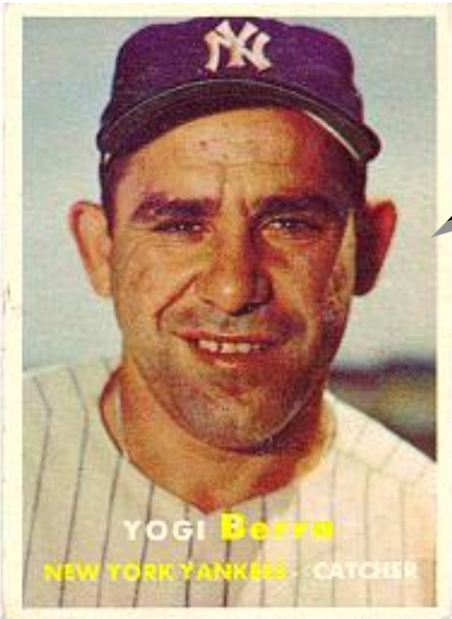
How can we find motifs…

- Without having to specify the length/other parameters
- In massive datasets
- While ignoring “background” motifs (ECG example)
- Under time warping, or uniform scaling
- While assessing their significance



Finding these 3 motifs requires about 6,250,000 calls to the Euclidean distance function

Time Series Prediction



Yogi Berra
1925 -

Prediction is hard, especially about the future

There are two kinds of time series prediction

- **Black Box**: Predict tomorrows electricity demand, given *only* the last ten years electricity demand.
- **White Box (side information)**: Predict tomorrows electricity demand, given the last ten years electricity demand *and* the weather report, *and* the fact that fact that the world cup final is on and…

Black Box Time Series Prediction

- A paper in SIGMOD 04 claims to be able to get better than 60% accuracy on black box prediction of financial data (random guessing should give about 50%). The authors agreed to test blind on a dataset which I gave them, they again got more than 60%. But I gave them quantum-mechanical random walk data!
- A paper in SIGKDD in 1998 did black box prediction using association rules, more than twelve papers extended the work… but then it was proved that the approach *could* not work*!

Nothing I have seen suggests to me that any non-trivial contributions have been made to this problem. (To be fair, it is a *very* hard problem)

White Box Time Series Prediction

Time Series Visualization

Warning! I am not an expert of visualization

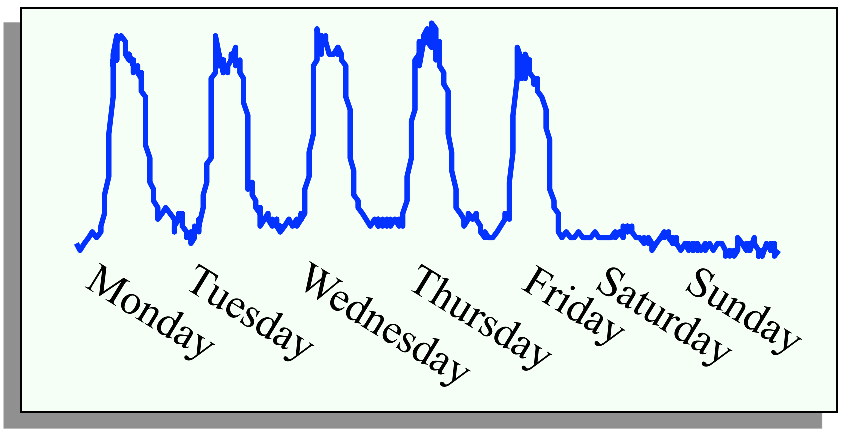
See tutorials by Ben Shneiderman, Daniel A. Keim, Marti Hearst etc

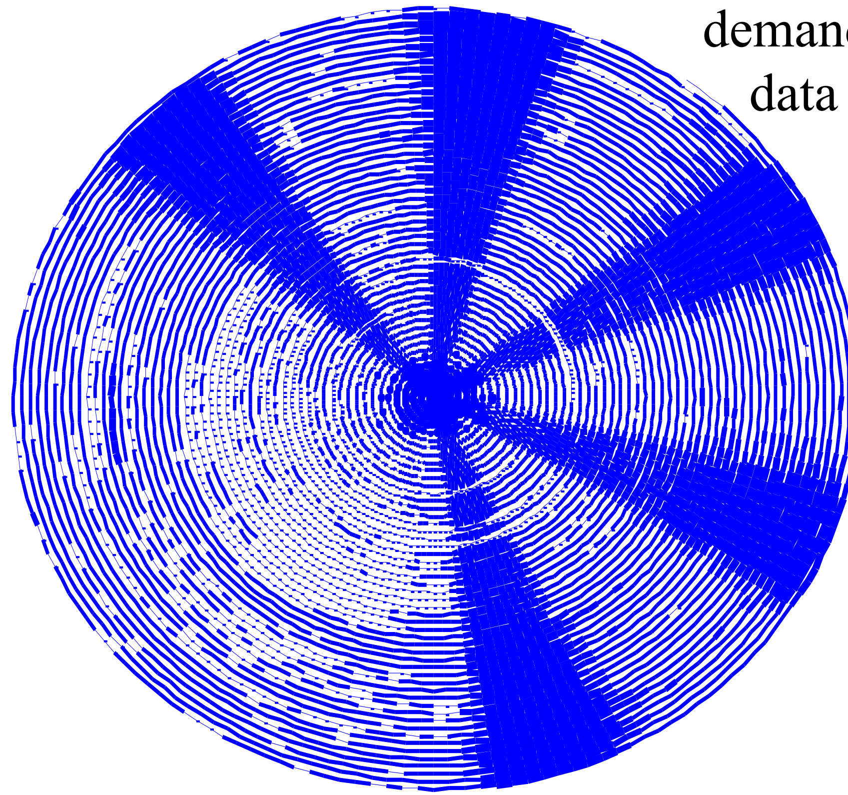
However, we will spend 10 minutes looking at some of the major time series visualization tools

Time Series Spirals

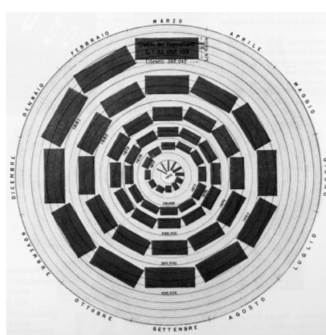
One year of power demand data

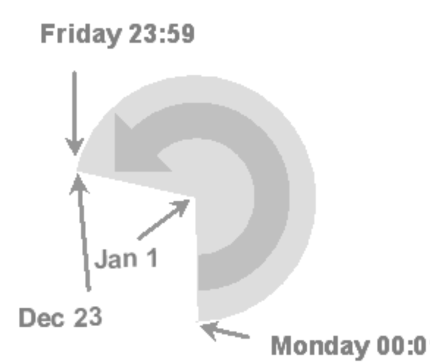
- **Spiral Axis** = serial attributes are encoded as line thickness
- **Radii** = periodic attributes





Carlis & Konstan. UIST-98
Independently rediscovered by
Weber, Alexa & Müller InfoVis-01
But dates back to 1888!



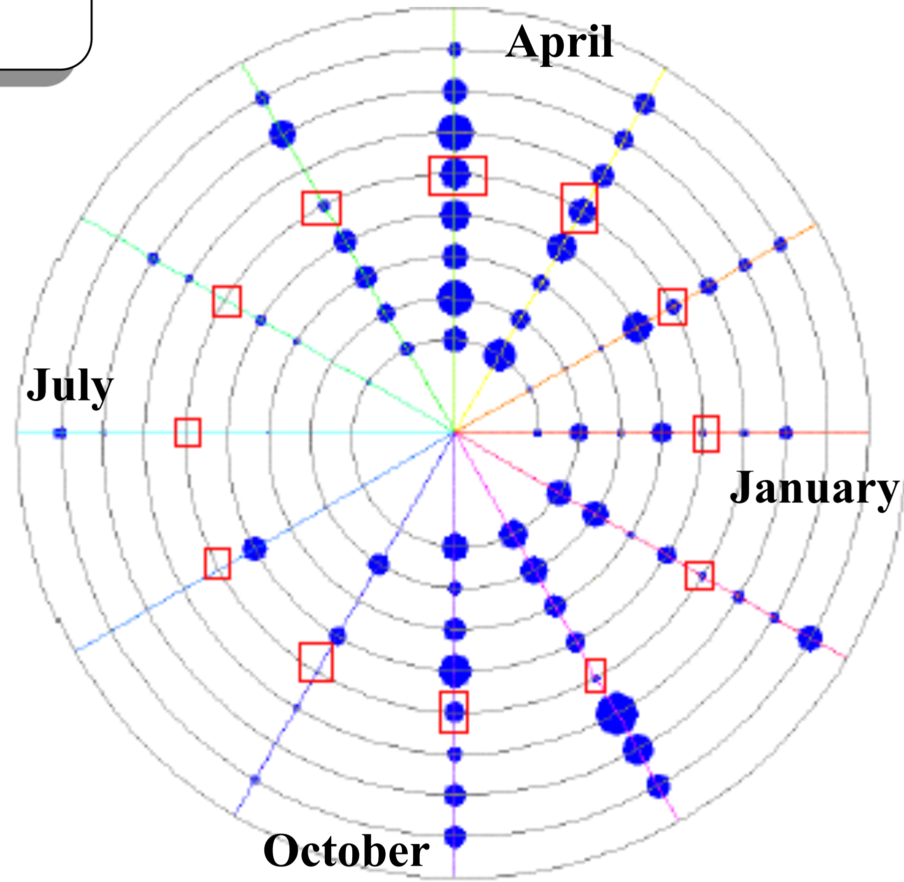


Time Series Spirals

The spokes are months, and spiral guide lines are years

• "chimpanzees eat new leaves of this plant, which are produced at the beginning and the end of the rainy season which is approximately October – April, and, more particularly, late rainy season consumption was steadier than that in early season"

• "in 1984 (red boxes), which was a drought year, consumption was considerably lower in the early rainy season, and high consumption in August 1983 occurred when the rainy season came early"



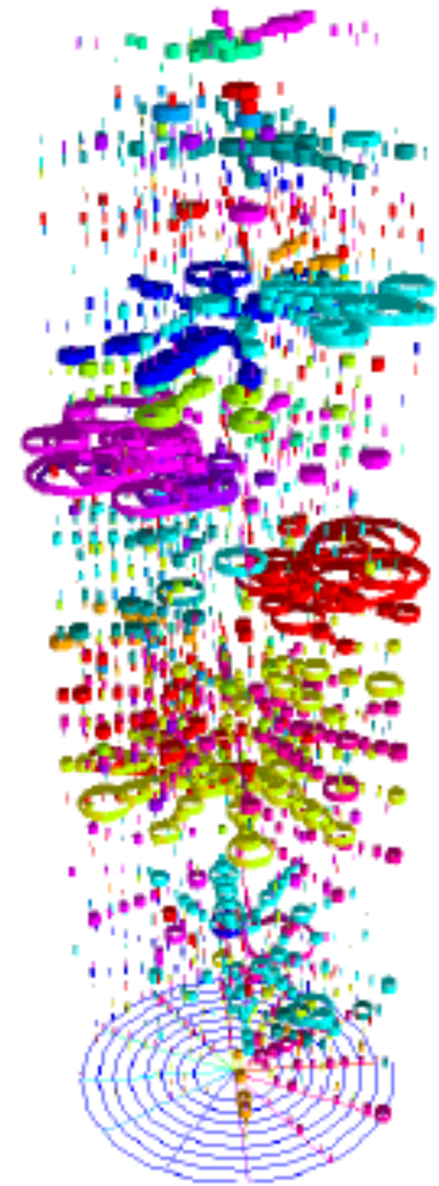
Chimpanzees Monthly Food Intake
1980-1988

Time Series Spirals

Comments

- Simple and intuitive
- Many extensions possible
- Scalability is still an issue
- Only useful on periodic data, and only then if you know the period

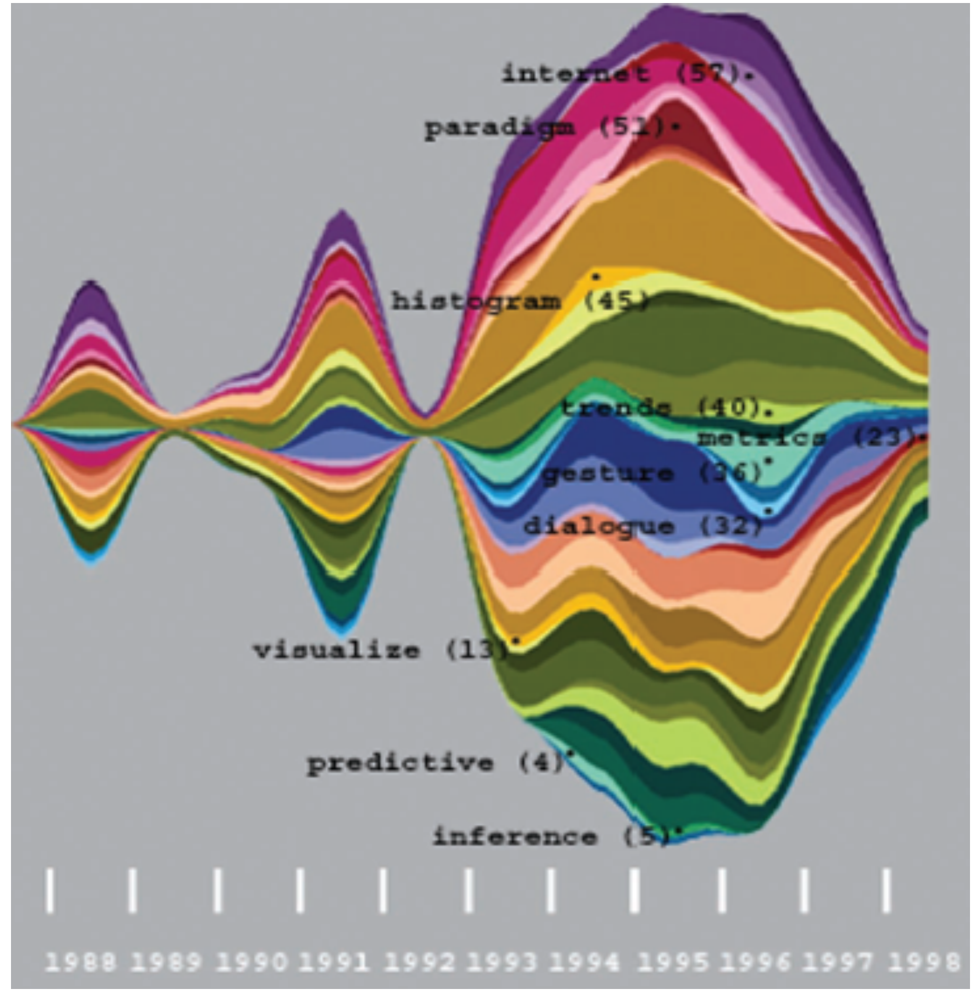
Effect of changing the period



112 types of food

ThemeRiver

- Current width = strength of theme
- River width = global strength
- Color mapping (similar themes/same color family)
- Time axis
- External events can be linked

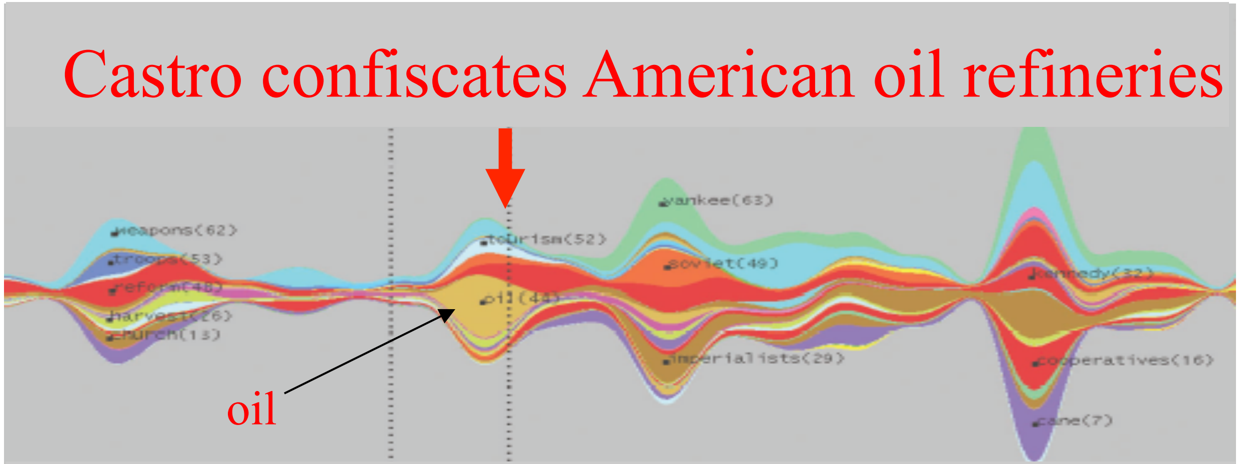


A company's patent activity

1988 to 1998

Havre, Hetzler, Whitney & Nowell
InfoVis 2000

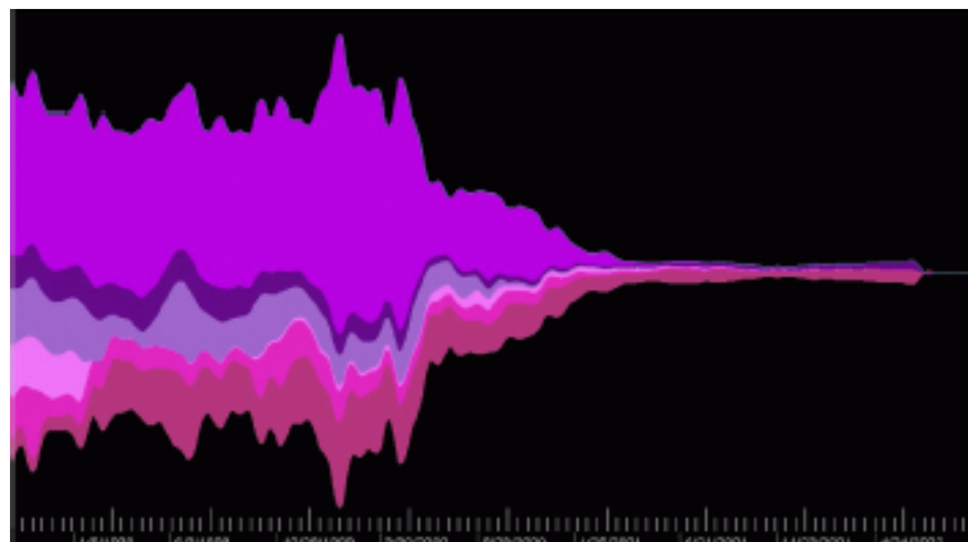
ThemeRiver



Fidel Castro's speeches 1960-1961

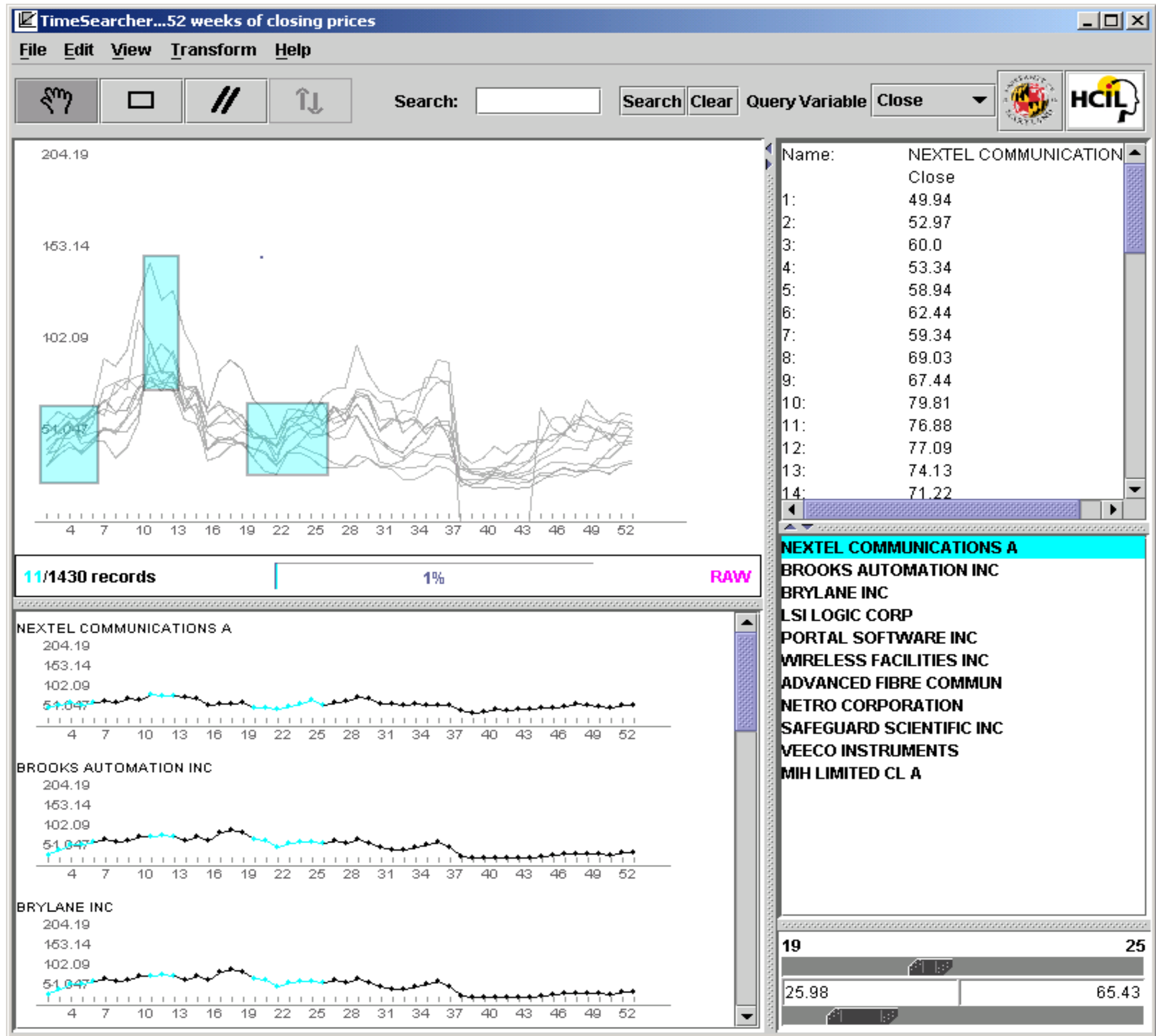
Comments

- Simple and intuitive
- Many extensions possible
- Scalability is still an issue



dot.com stocks 1999-2002

TimeSearcher



Comments

- Simple and intuitive
- Highly dynamic exploration

- Query power may be limited and simplistic
- Limited scalability

Hochheiser, and Shneiderman

VizTree

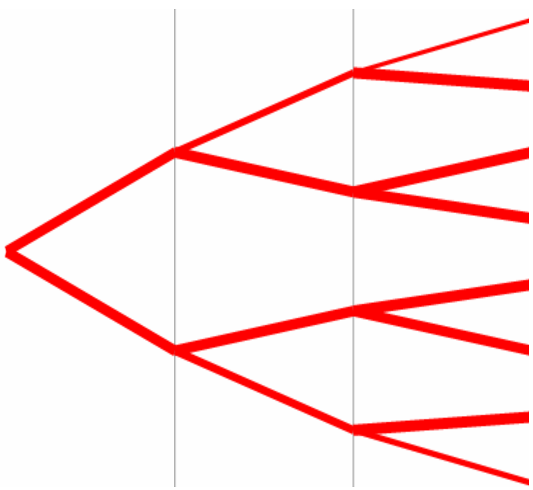
0101100101111001101001000001000
101001101101011100001010101110
1111100011011011011111101001100
100100011010001111001101101000
101111000101101001101100110100
0000100110001001110000001110100
1100101100001010010

10001000101001000101010100001
010100010101110111101011010010
111010010101001110101010100101
00101010111010101001010101011
010101001011001011101111010001
110000101000010011101010001110
0001010101100101110101

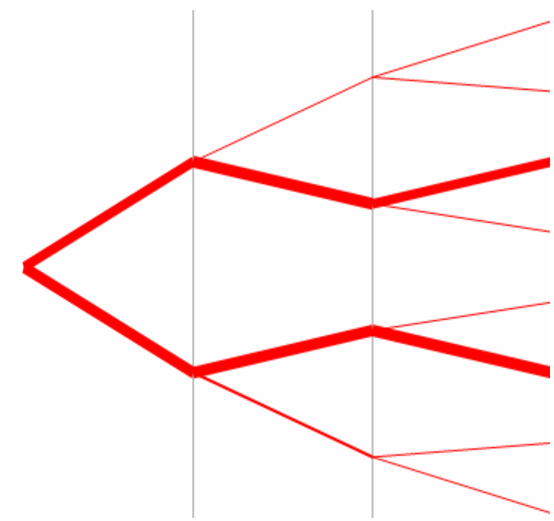
Here are two sets of bit strings. Which set is generated by a human and which one is generated by a computer?

VizTree

0101100101111001101001000001000101
0011011010111000010101011101111110
0011011011011111101001100100010001
1010001111001101101000101111100010
1101001101100110100000001001100010
0111000000111010011001011000010100
10



1000100010100100010101010000101 0
1000101011101111010110100101110 10
0101010011101010101001010010101 01
1101010100101010101101010100101 10
0101110111101000111000010100001 00
1110101000111000010101011001011 10
101

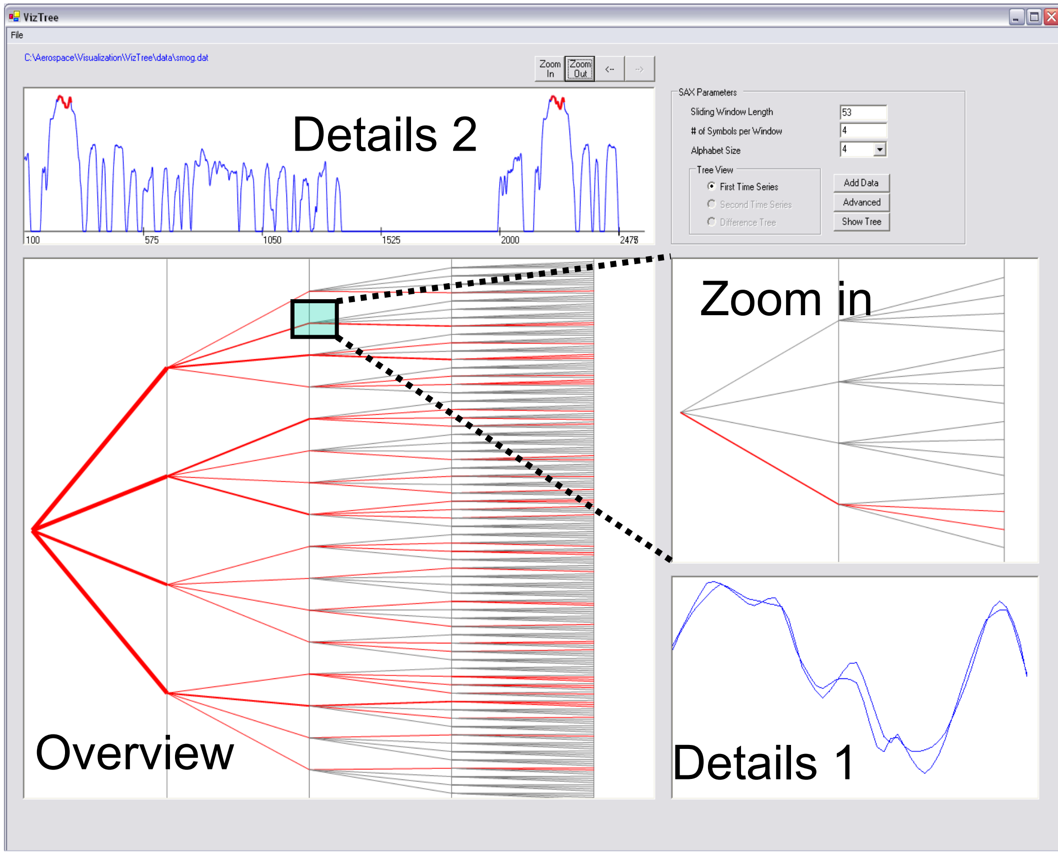


Lets put the sequences into a depth limited suffix tree, such that the frequencies of all triplets are encoded in the thickness of branches…

"humans usually try to fake randomness by alternating patterns"

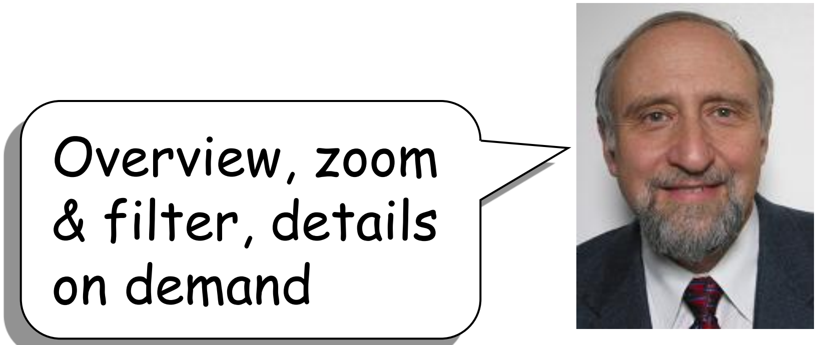
VizTree

The "trick" on the previous slide only works for discrete data, but time series are *real* valued.



VisTree

- Convert the time series to SAX
- Push the data in a depth-limited suffix tree
- Encode the frequencies as the line thickness

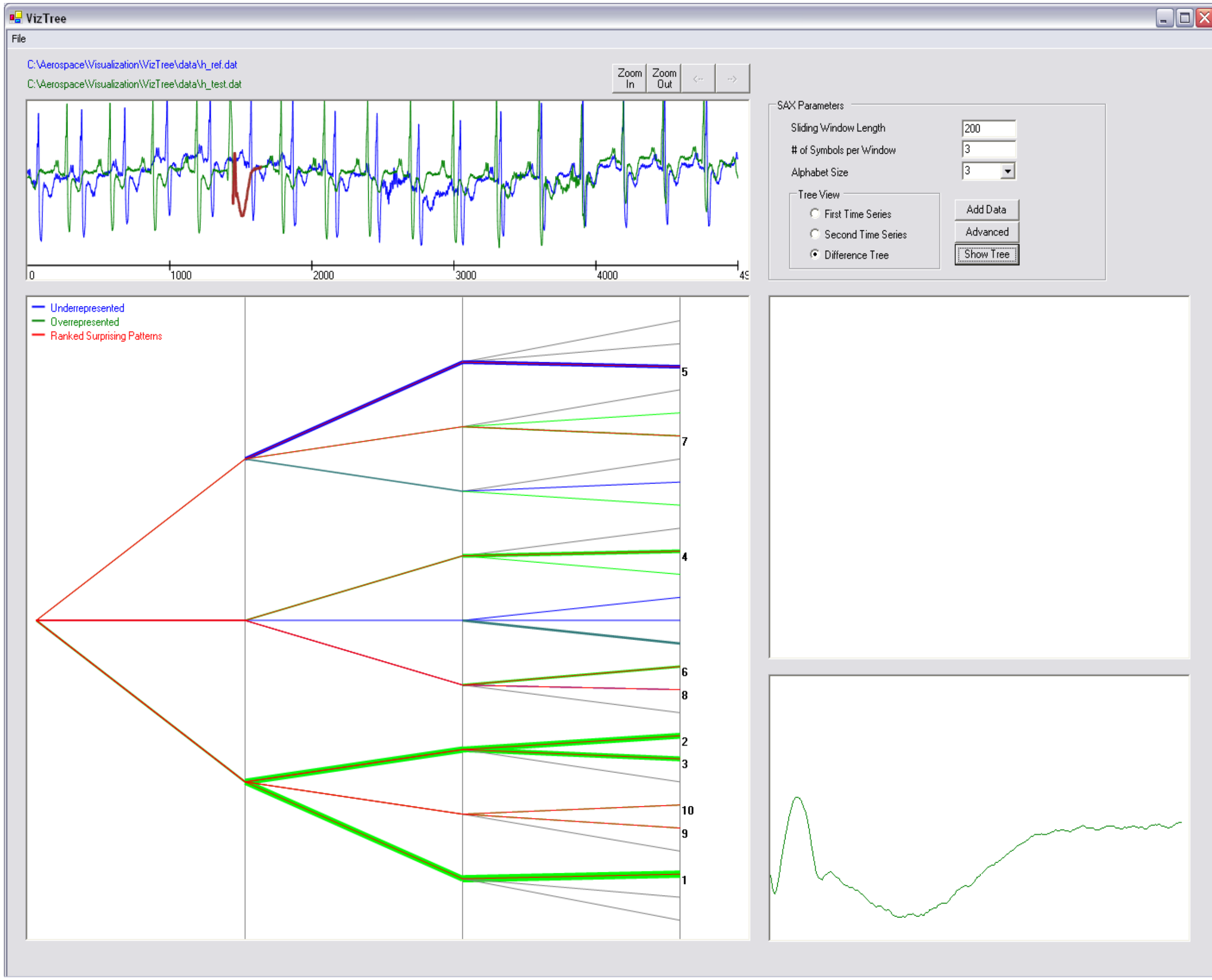


Ben Shneiderman

VizTree/ DiffTree

DiffTree

- Convert the two time series to SAX
- Push the data in a depth-limited suffix tree
- Encode the frequencies as the line thickness
- Encode the *difference* of frequencies as the line *color*



Blue lines - pattern is more common in A
Green lines - pattern is more common in B
Red lines - pattern is equi-frequent in A and B


The Last Word
The sun is setting on all other symbolic representations of time series, SAX is the *only* way to go

What should we be working on?
The Top Ten Time Series Problems

- I strongly believe that time series similarly search is dead (or at least dying)

- The good news is that there is a lot interesting unsolved problems out there

- What follows is my subjective list of the most interesting problems in time series data mining (In random order)

Problem 1

Discovering Time Series Motifs without all those hard-to-set parameters

Unlike similarity search, motif discovery really appears to have lots of applications!

However, we currently have to set 3 to 5 critical parameters. Can we find the naturally repeated patterns without specifying all these parameters?

Problem 2

Clustering streaming time series

Given an single infinite stream, can you find, then incrementally maintain, K clusters of subsequences, under Euclidean distance or DTW? (perhaps with a forgetting factor)

Note that was *apparently* solved before*!

The problem is NOT to do this fast, the problem is to do this in a *meaningful* way.

Problem 3

Time Series Joins

Given two time series, find all the subsections where they are similar.

Without normalizing the subsections, this is easy but meaningless.

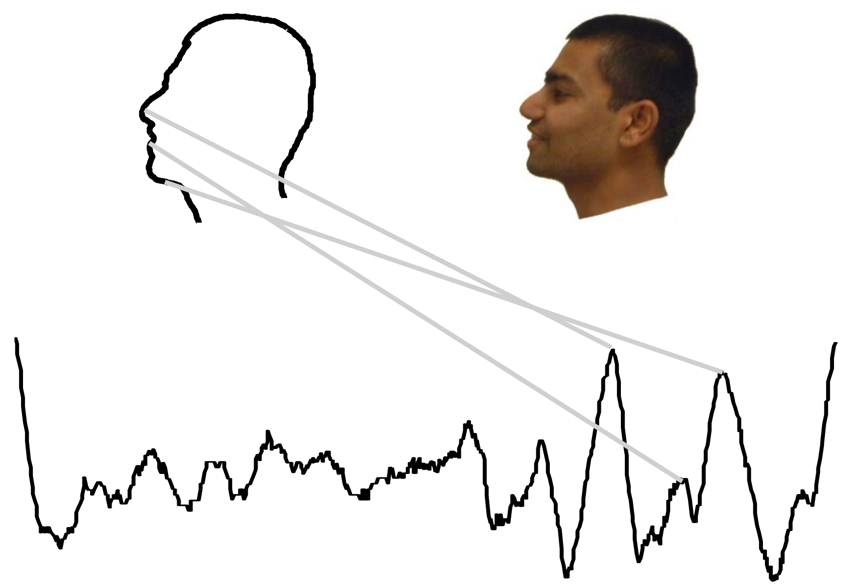
The problem is NOT to do this fast, the problem is to do this in a *meaningful* way.

Problem 4

Understanding the "why" in time series classification and clustering

Given that two time series are clustered/classified together, automatically construct an explanation of why.

Image data, may best be thought of as time series…



Problem 5

Building tools to visualize massive time series

The best data mining/pattern recognition tool is the human eye, can we exploit this fact?

How can we visually summarize massive time series, such that regularities, outliers, anomalies etc, become visible?

Image by Martin Wattenberg*

Problem 6

Classifying time series with a eager learner

While there has been work on classifying (shape-bases) time series with decision trees, neural networks, bayesian classifiers etc. None of these approaches is competitive with 1-nearest neighbor with DTW.

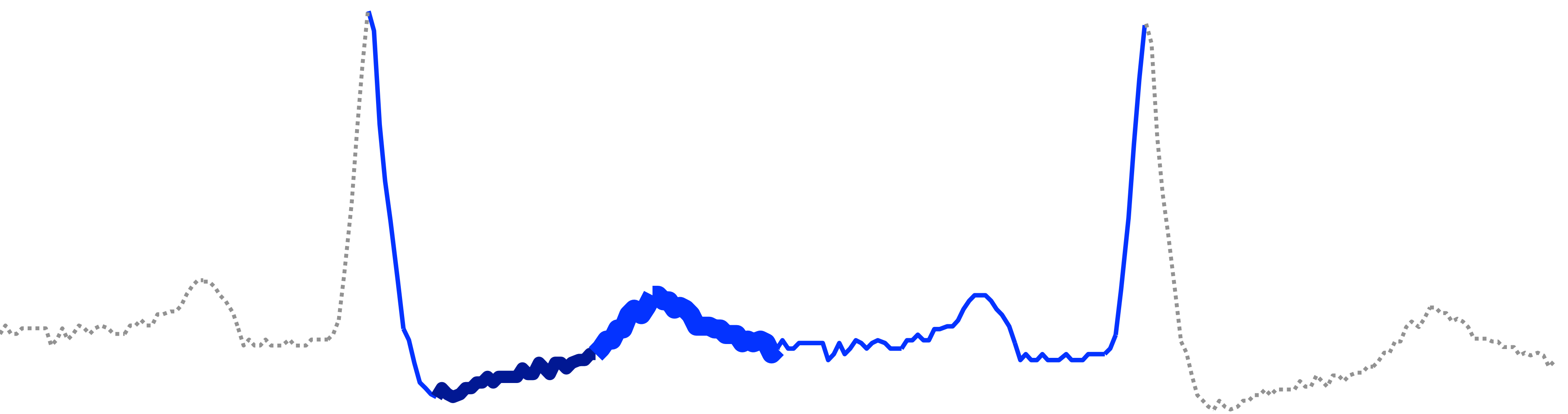
As we have seen, DTW is essentially linear, nevertheless, 1-nearest neighbor needs to visit every instance, can we do better?

Problem 7

Weighted time series representations

It is well known in the machine learning community that weighting features can greatly improve accuracy in classification and clustering tasks.

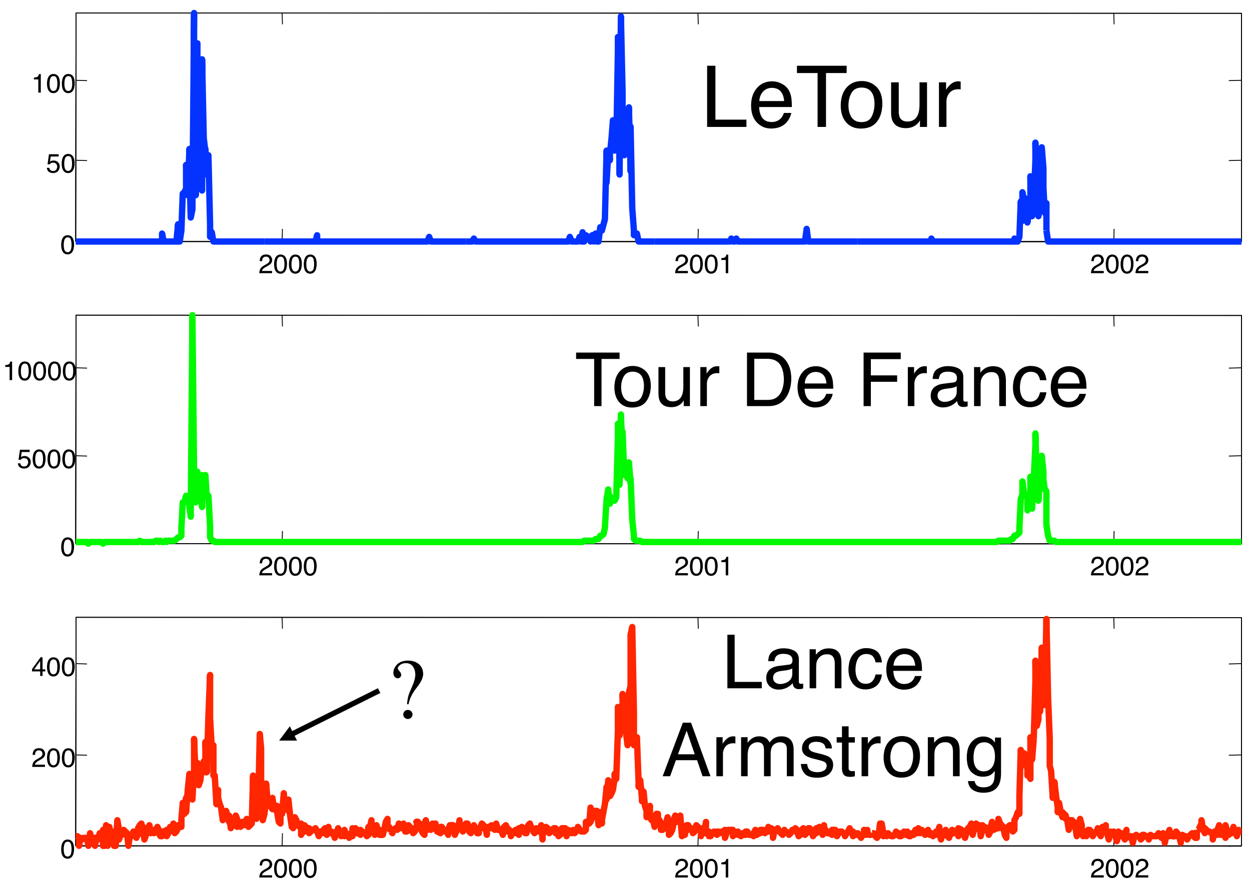
Are weighted time series representations useful?



Problem 8

Query by Burst

Search Engine Query Log



It makes sense that the bursts for "LeTour", "Tour de France" and "Lance Armstrong" are all related.

But what caused the extra interest in Lance Armstrong in August/ September 2000?

Example by M. Vlachos

Problem 9

Applications, Applications, Applications

For every one paper that shows a real application of time series data mining, there are dozens that introduce an idea of dubious real world utility.

We need to give more attention to problems with real, demonstrated applications (and give them weight when reviewing?).

Best Bets: Music, Motion Capture, Video, Web Logs…

Problem 10

You Tell Me!

Any ideas?

We can discuss them in 3 minutes.

Conclusions

- Time series are everywhere!
- While (I believe) similarly search in time series is dead or dying, there are lots of great problems to be solved.
- The right representation for the problem at hand is the key to an efficient and effective solution.
- For some reason, time series research seems vulnerable to sloppy evaluation. If we all shared our data, this would be a huge step in the right direction…

Thanks!

Thanks to the people with whom I have co-authored Time Series Papers

- Michael Pazzani
- Sharad Mehrotra
- Kaushik Chakrabarti
- Selina Chu
- David Hart
- Padhraic Smyth
- Jessica Lin
- Bill 'Yuan-chi' Chiu
- Stefano Lonardi
- Shruti Kasetty
- Chotirat (Ann) Ratanamahatana
- Pranav Patel
- Harry Hochheiser
- Ben Shneiderman
- Marios Hadjieleftheriou
- Victor Zordan
- Your name here! (I welcome collaborators)

- Dimitrios Gunopulos
- Michail Vlachos
- Marc Cardle
- Stephen Brooks
- Bhrigu Celly
- Themis Palpanas
- Jiyuan An, H. Chen, K. Furuse and N. Ohbo

Questions?

All datasets and code used in this tutorial can be found at

www.cs.ucr.edu/~eamonn/TSDMA/index.html