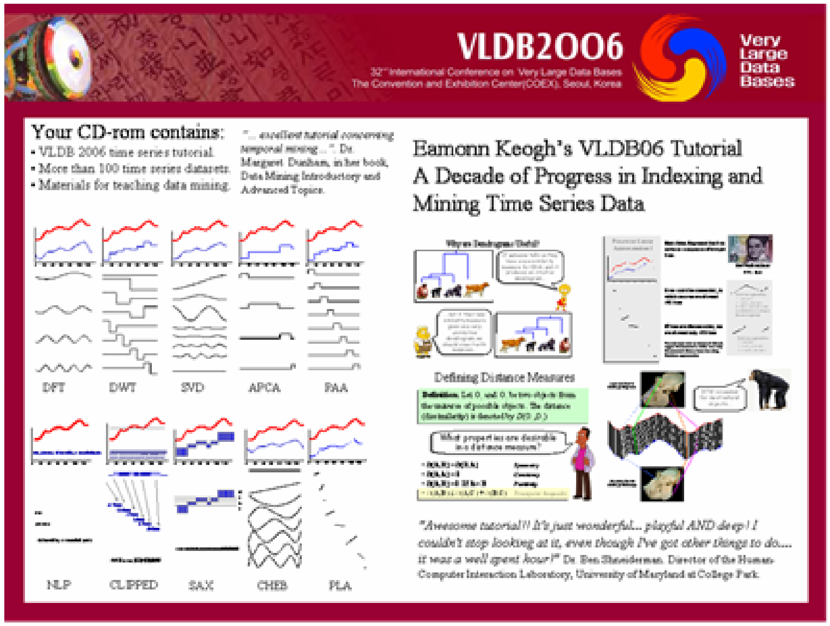
Introduction to Time Series Mining

Slides from Keogh Eamonn's tutorial:



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

What are Time Series?

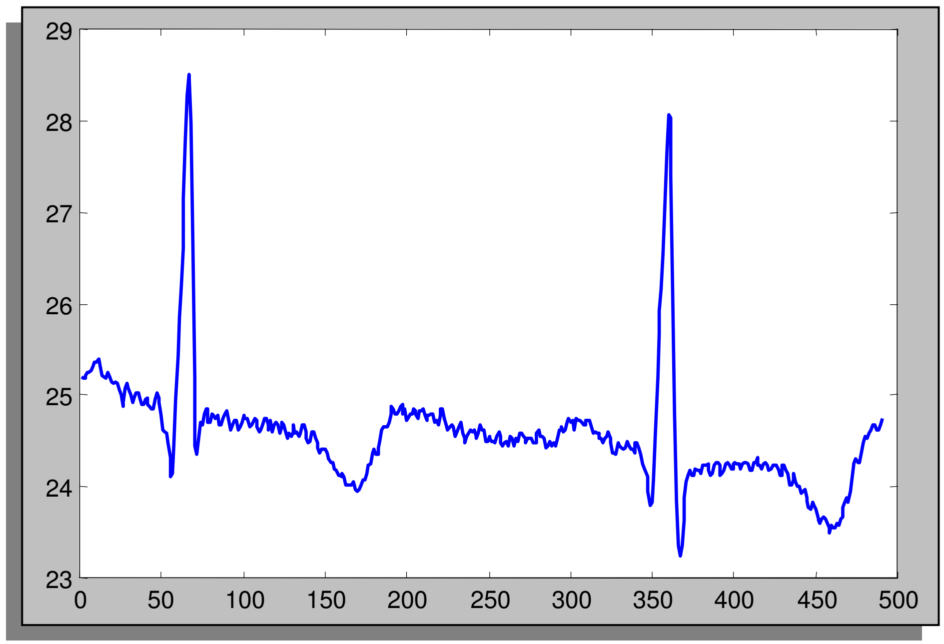
A time series is a collection of observations made sequentially in time.

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



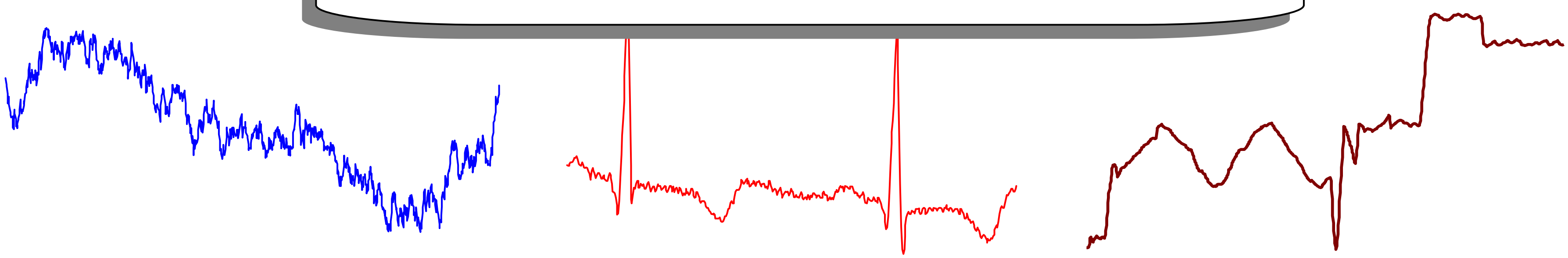
Virtually all similarity measurements, indexing and dimensionality reduction techniques discussed in this tutorial can be used with other data types

Time Series are Ubiquitous! I

People measure things...

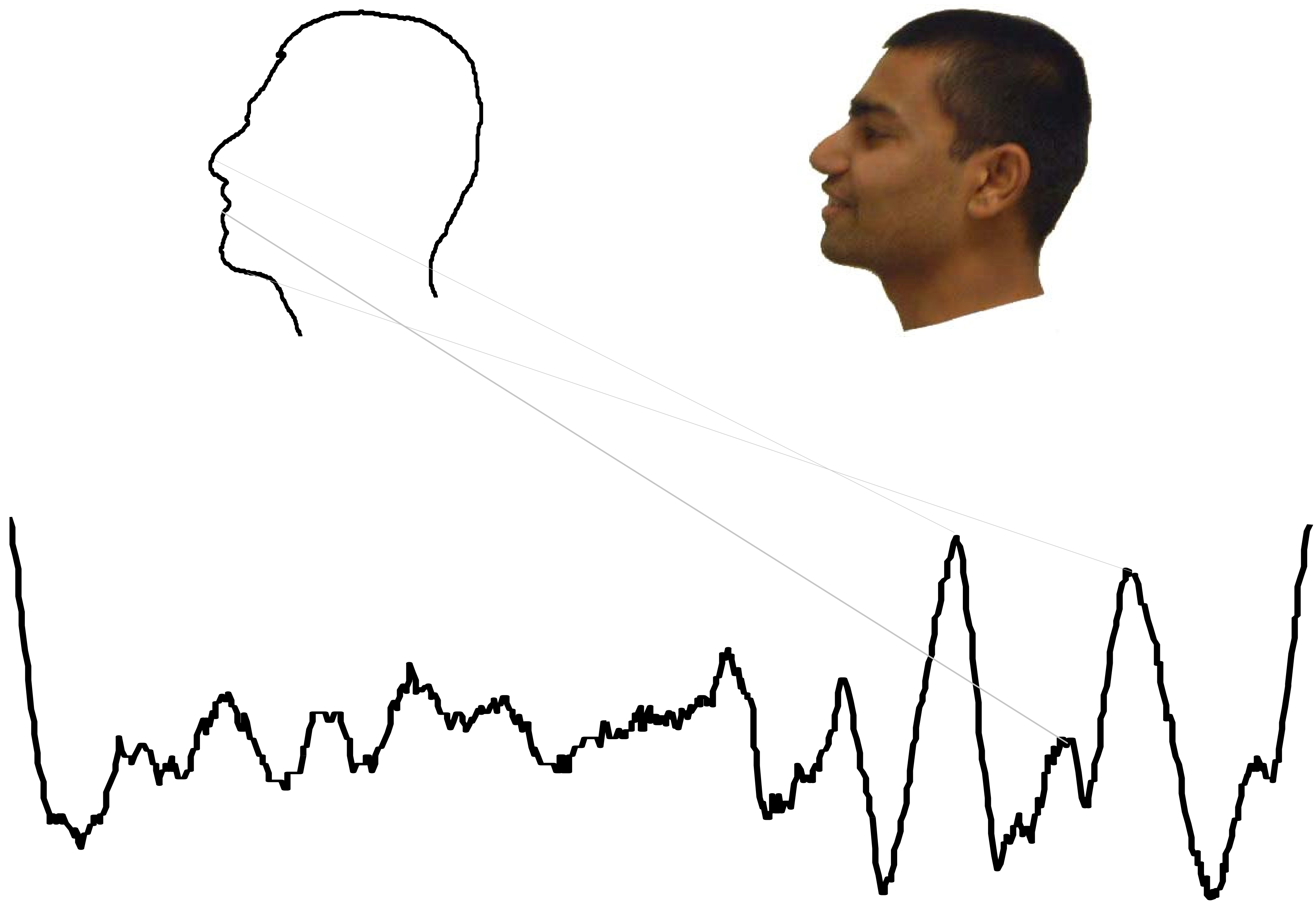
- *Their blood pressure*
- *George Bush's popularity rating*
- *The annual rainfall in Seattle*
- *The value of their Google stock*

...and things change over time...



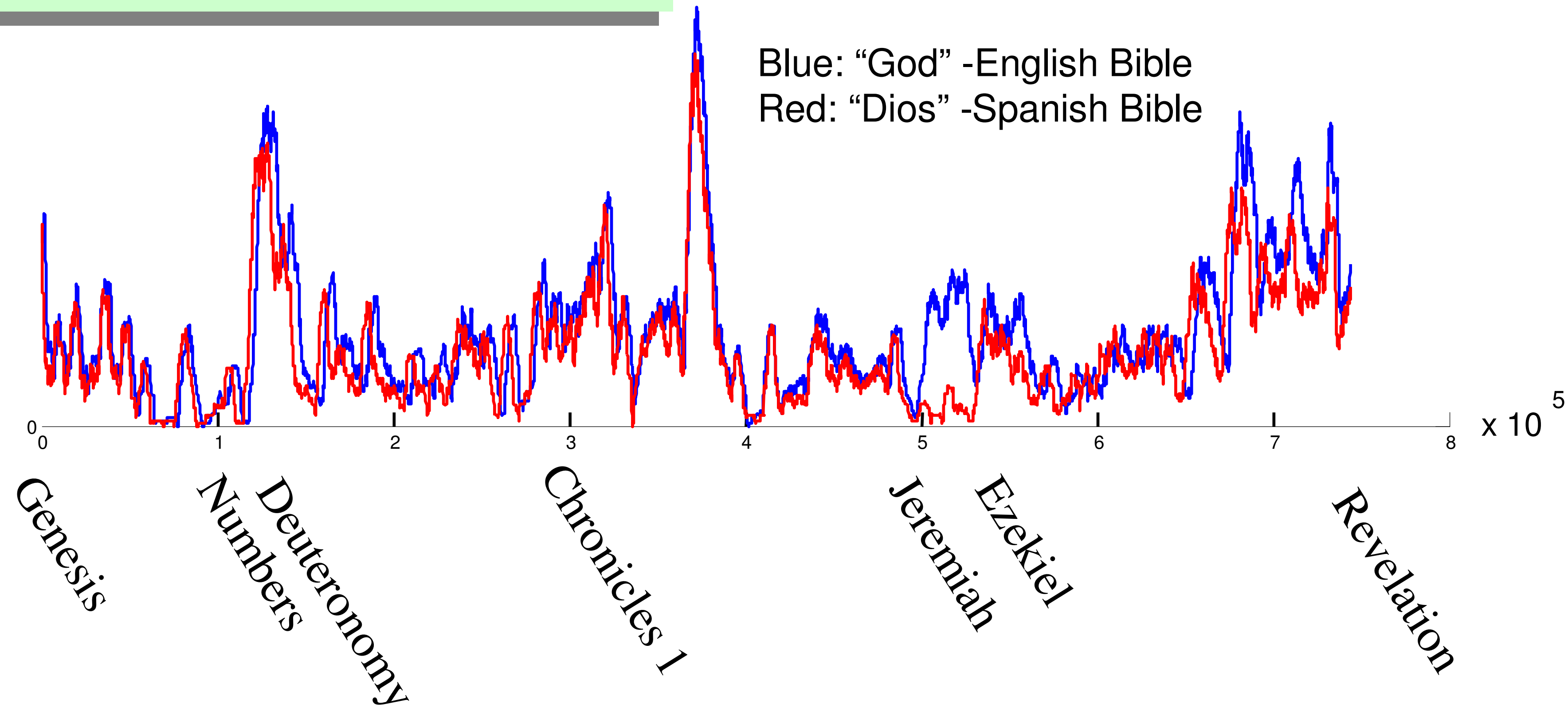
Thus time series occur in virtually every medical, scientific and businesses domain

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

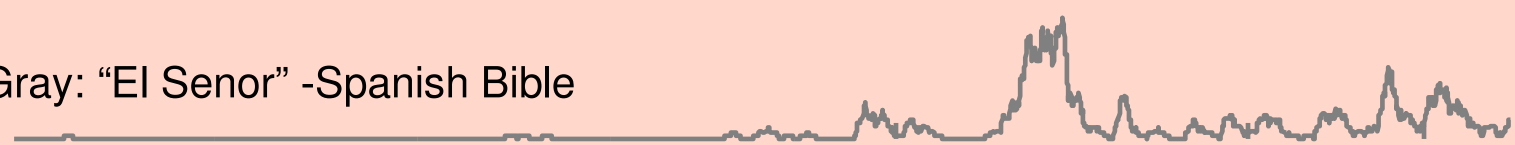


Text data, may best be thought of as time series…

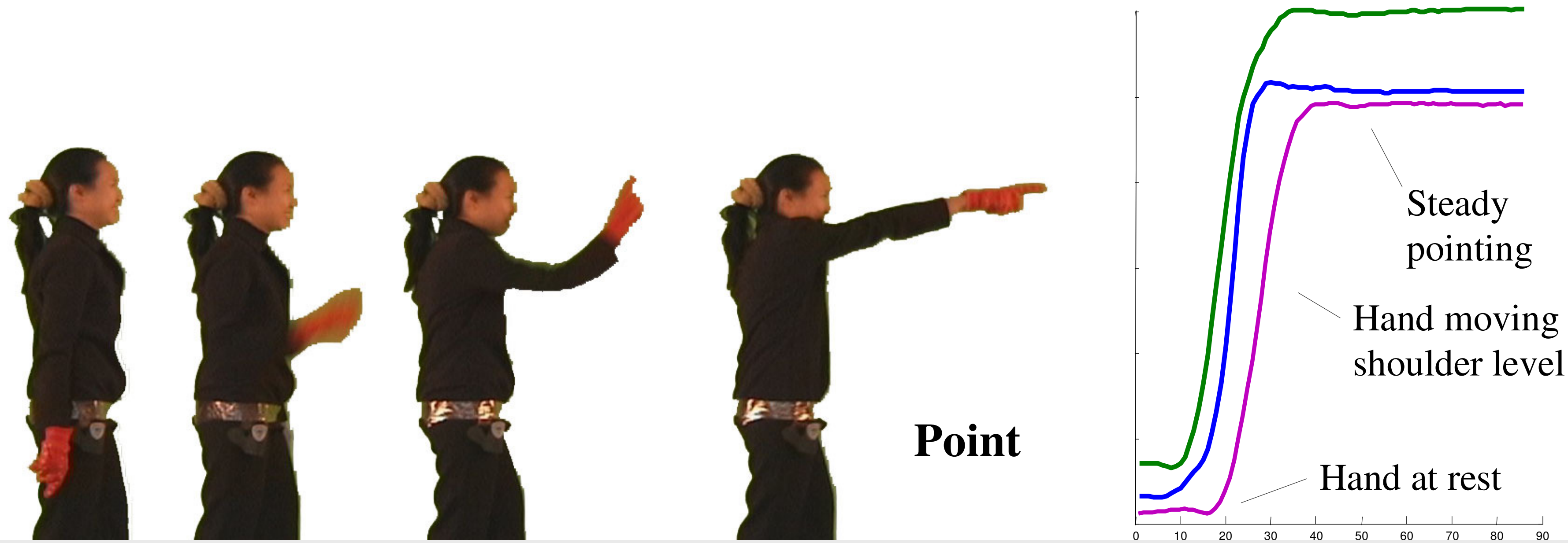
The local frequency of words in the Bible

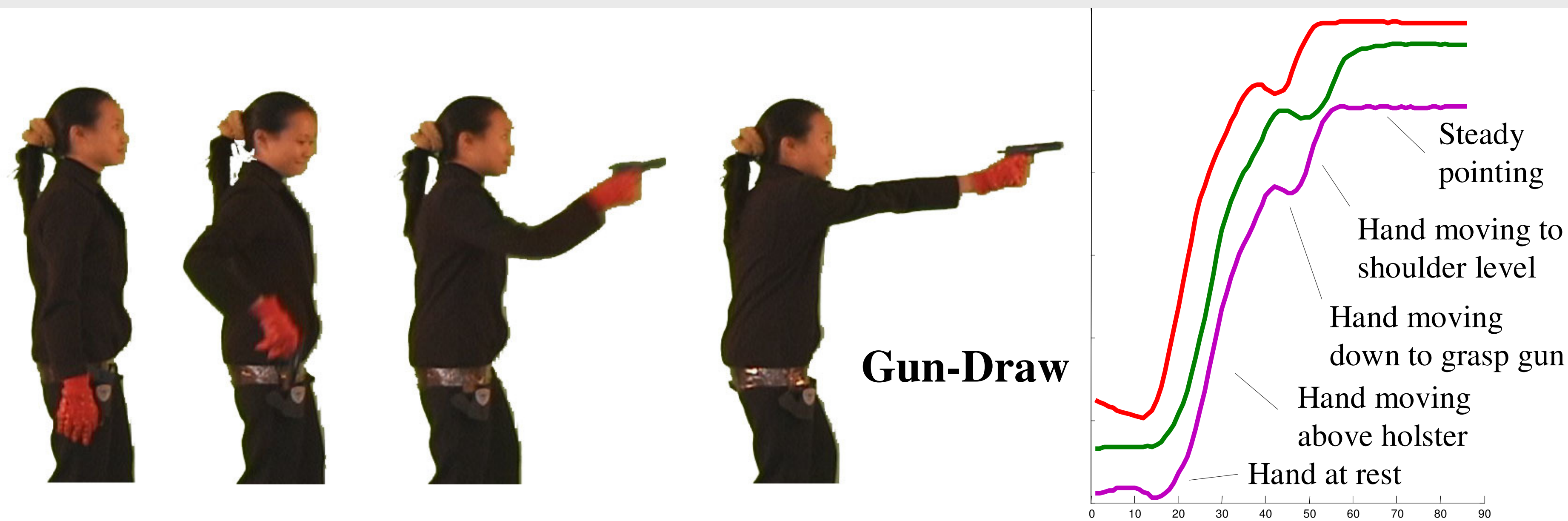


Gray: “El Senor” -Spanish Bible

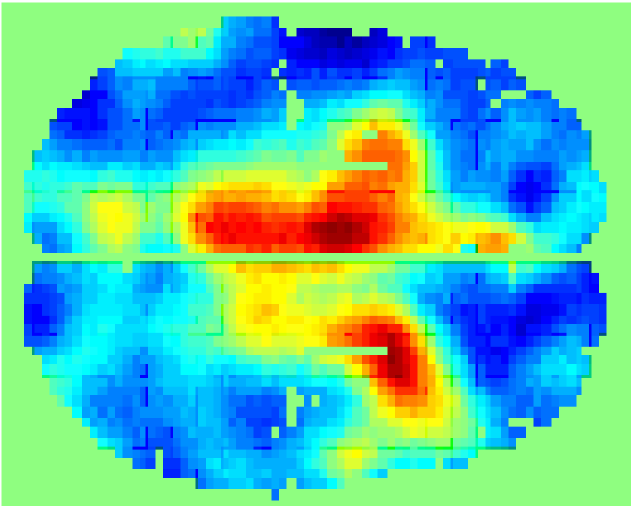


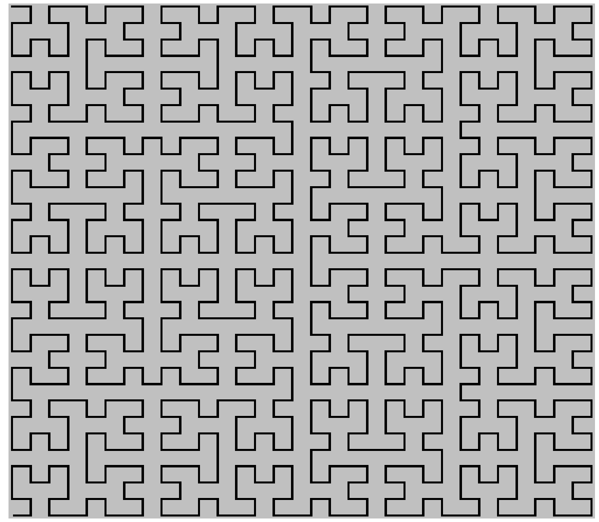
Video data, may best be thought of as time series…

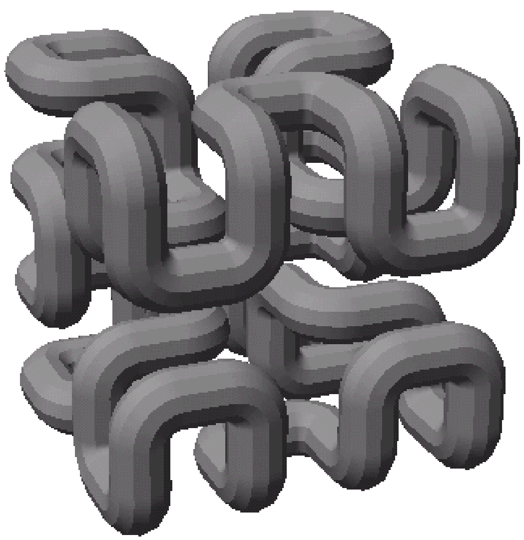


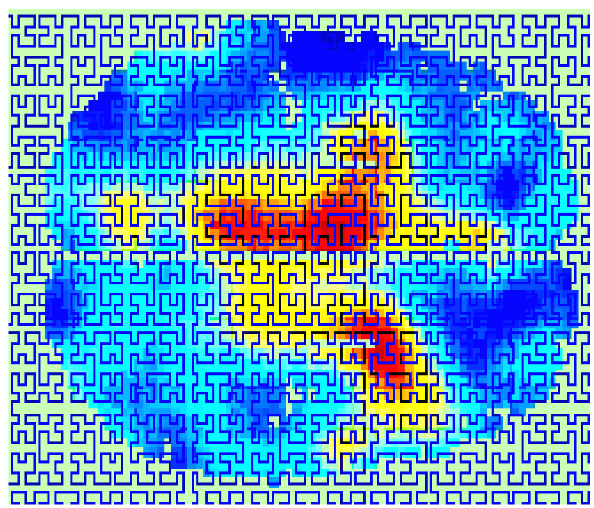


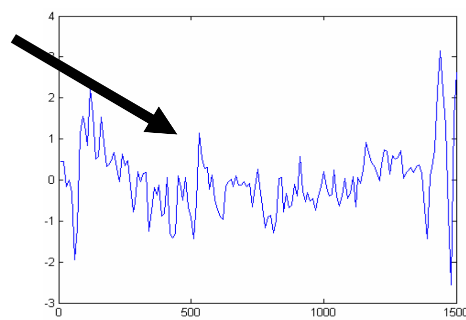
Brain scans (3D voxels), may best be thought of as time series..











Wang, Kontos, Li and Megalooikonomou ICASSP 2004

Why is Working With Time Series so Difficult? Part I

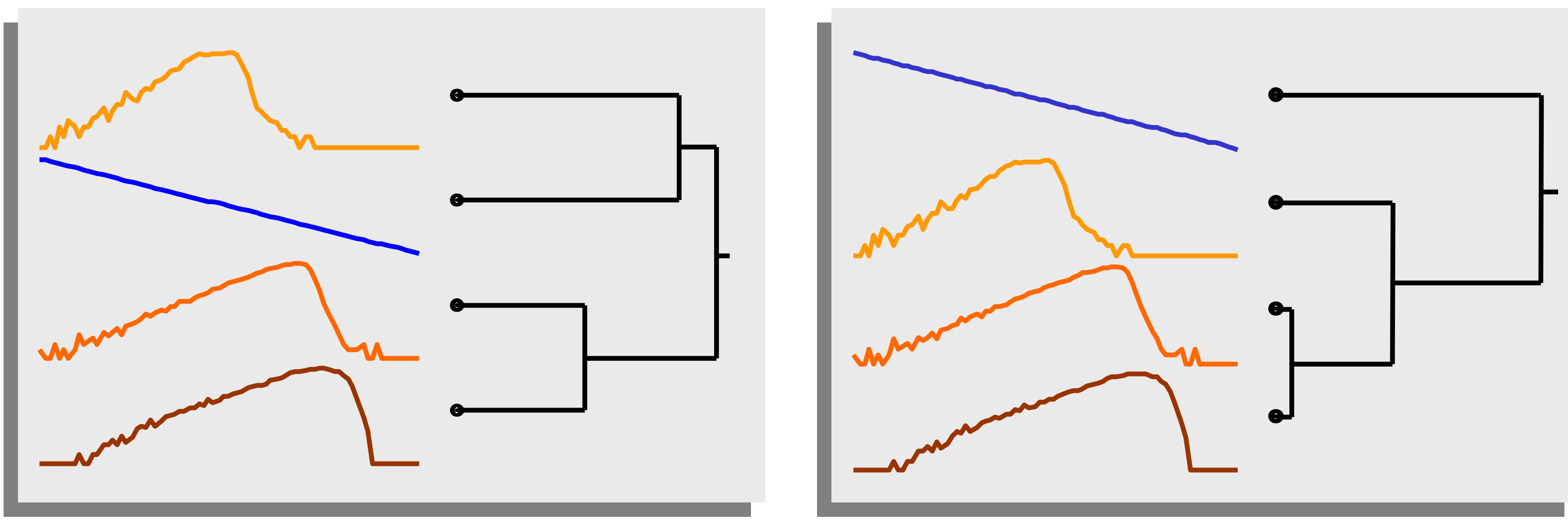
Answer: How do we work with very large databases?

- 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.

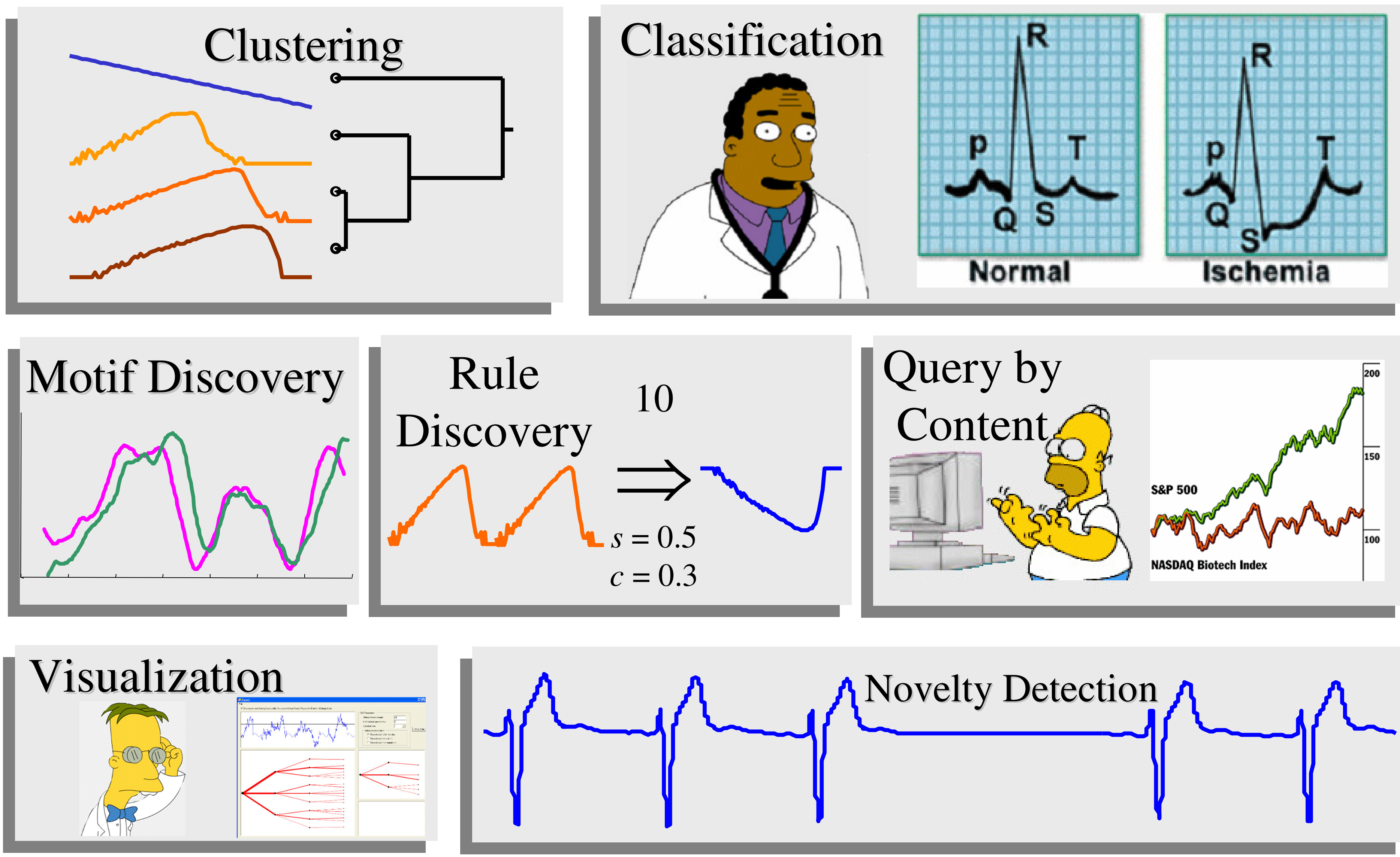
Why is working with time series so difficult? Part III

Answer: Miscellaneous data handling problems.

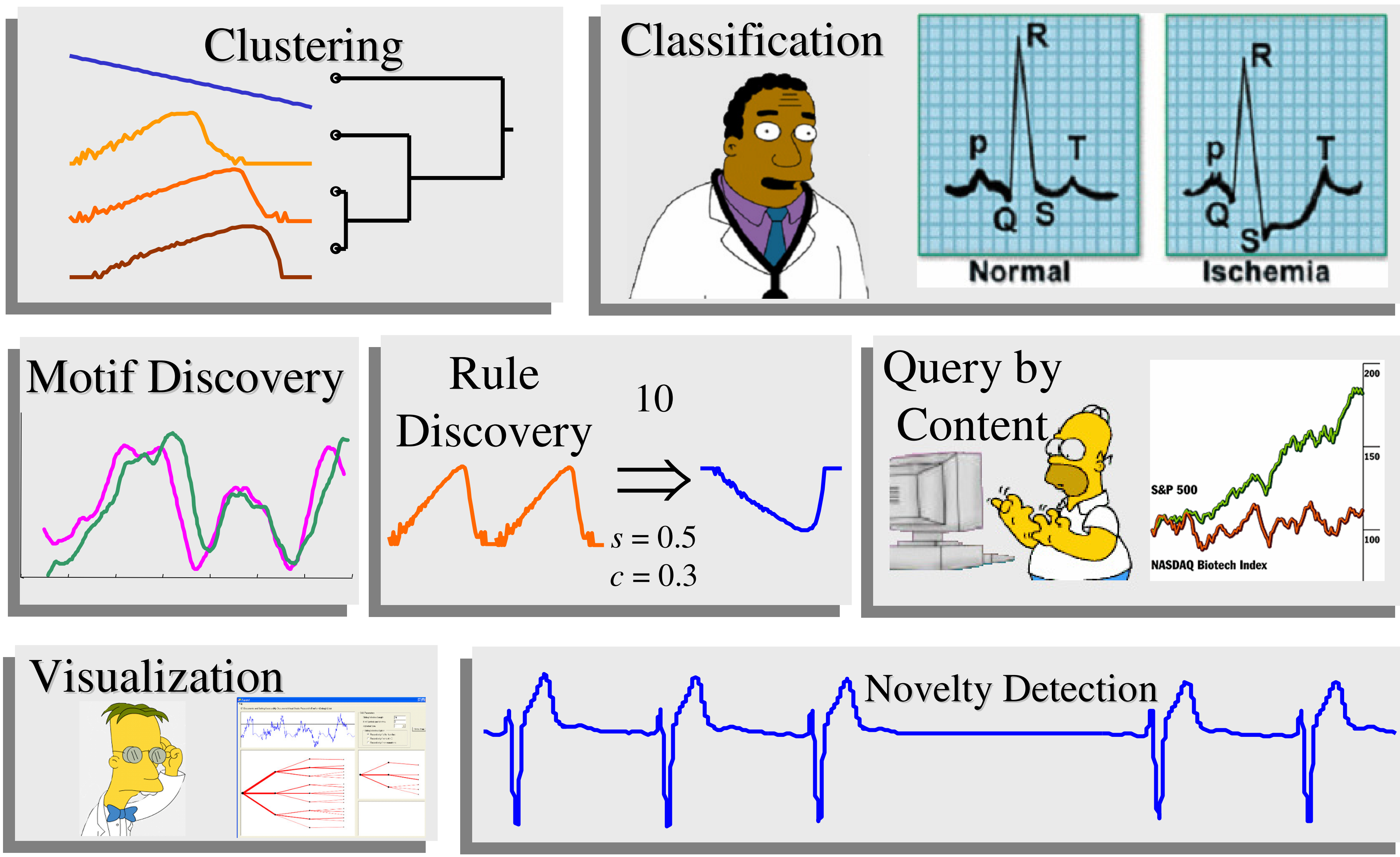
- Differing data formats.
- Differing sampling rates.
- Noise, missing values, etc.

We will not focus on these issues in this tutorial.

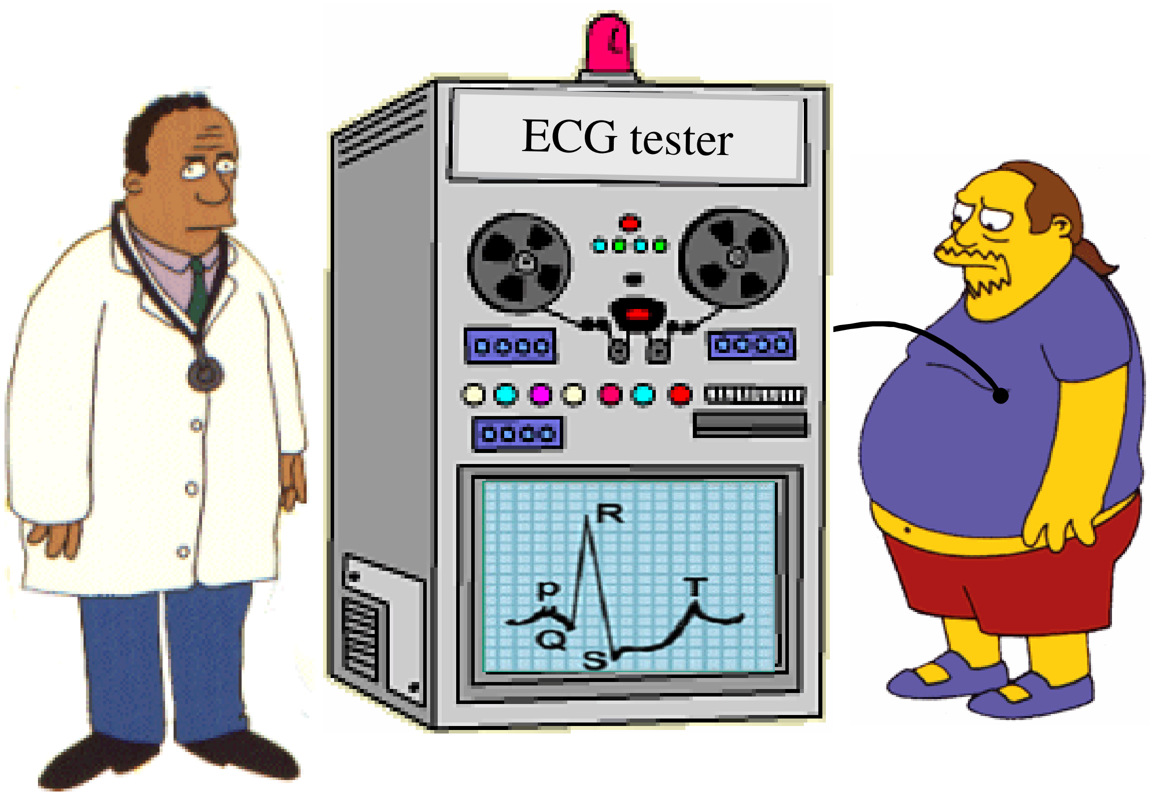
What do we want to do with the time series data?



All these problems require similarity 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*"

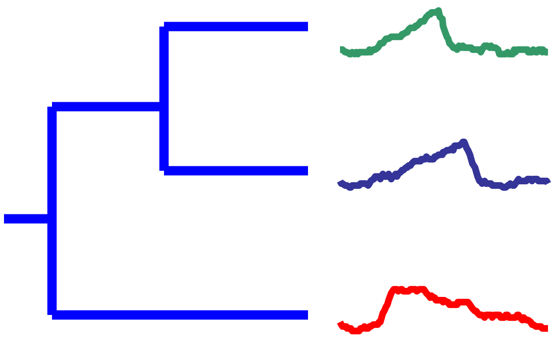
The real meaning of similarity is a philosophical question.

We will take a more pragmatic approach.

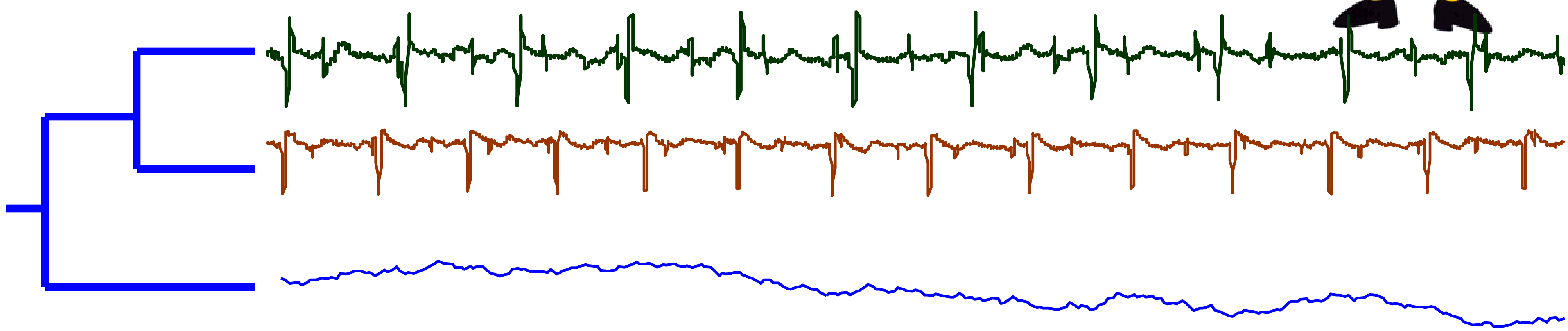
Two Kinds of Similarity

time series

Similarity at the level of *shape*
Next 40 minutes



Similarity at the *structural* level
Another 10 minutes

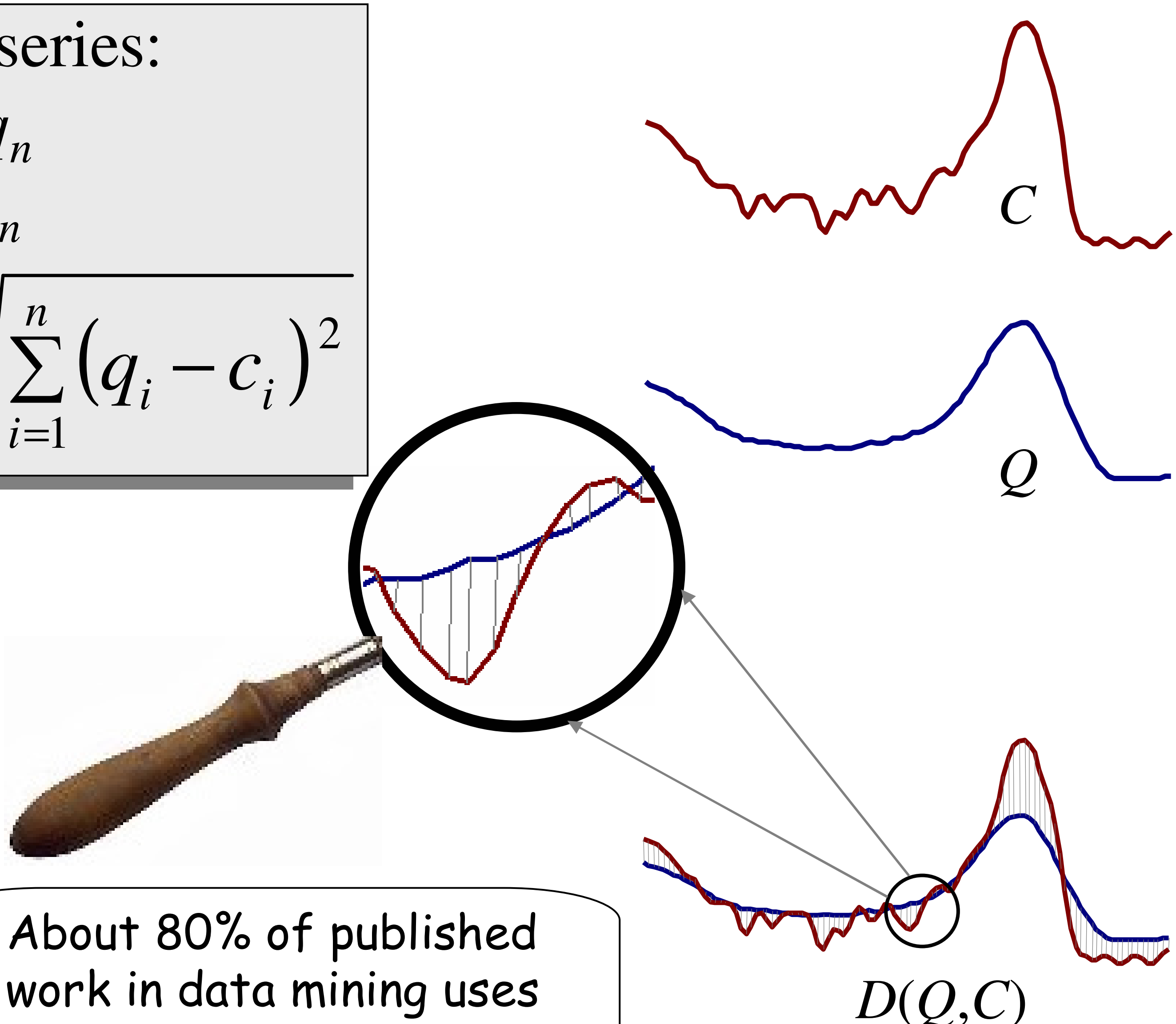


Euclidean Distance Metric

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

Preprocessing the data before distance calculations

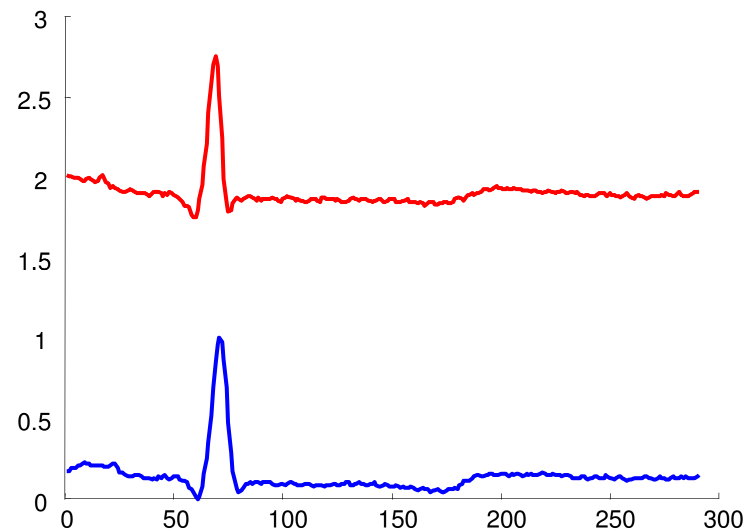
If we naively try to measure the distance between two "raw" time series, we may get very unintuitive results

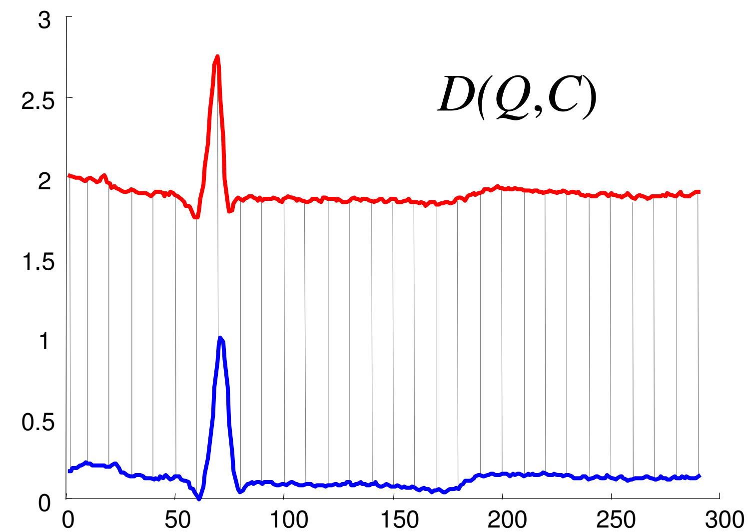
This is because Euclidean distance is very sensitive to some "distortions" in the data. For most problems these distortions are not meaningful, and thus we can and should remove them

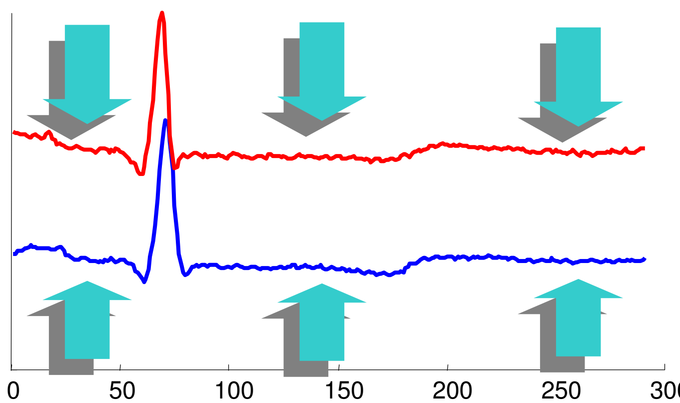
In the next few slides we will discuss the 4 most common distortions, and how to remove them

- Offset Translation
- Amplitude Scaling
- Linear Trend
- Noise

Transformation I: Offset Translation



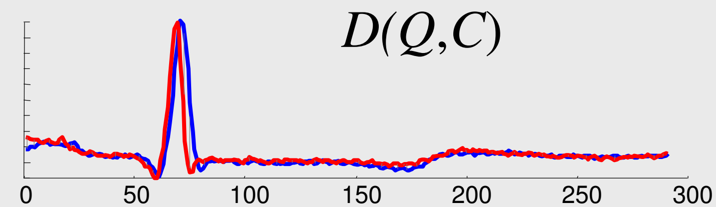




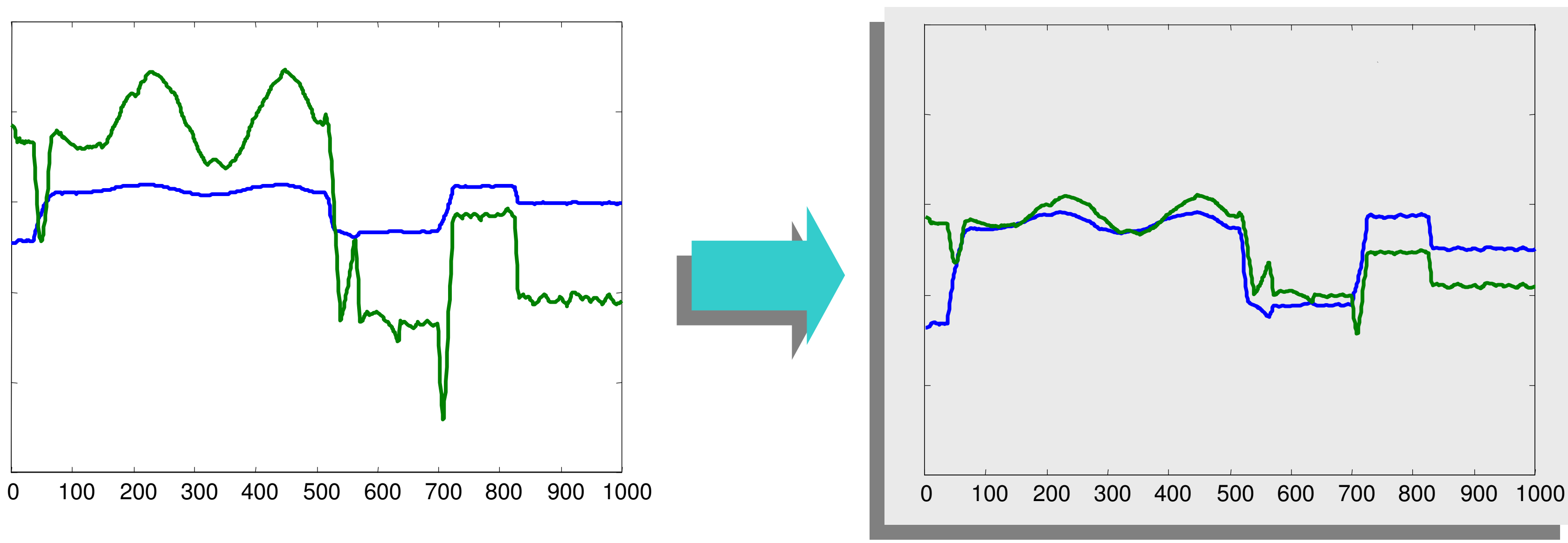
$Q = Q$ - mean(Q)

$C = C$ - mean(C)

$D(Q,C)$



Transformation II: Amplitude Scaling

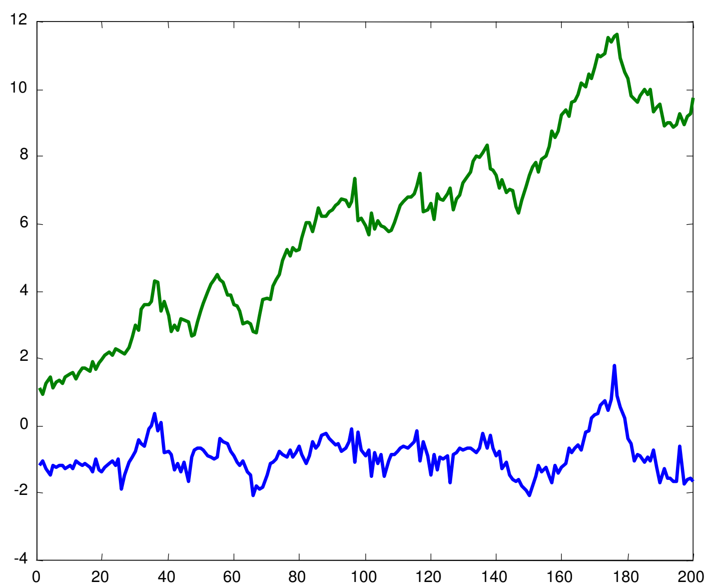


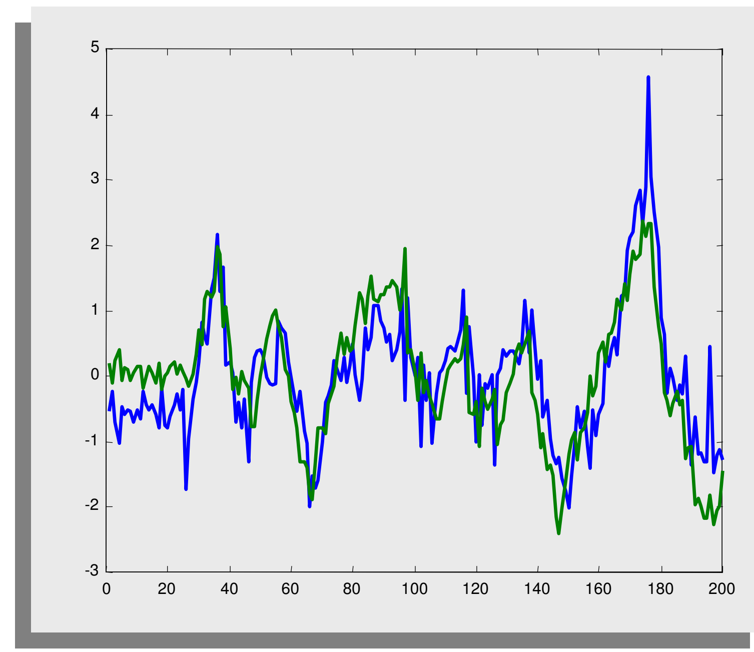
Q = (Q - mean(Q)) / std(Q)

C = (C - mean(C)) / std(C)

$D(Q,C)$

Transformation III: Linear Trend





The intuition behind removing linear trend is…

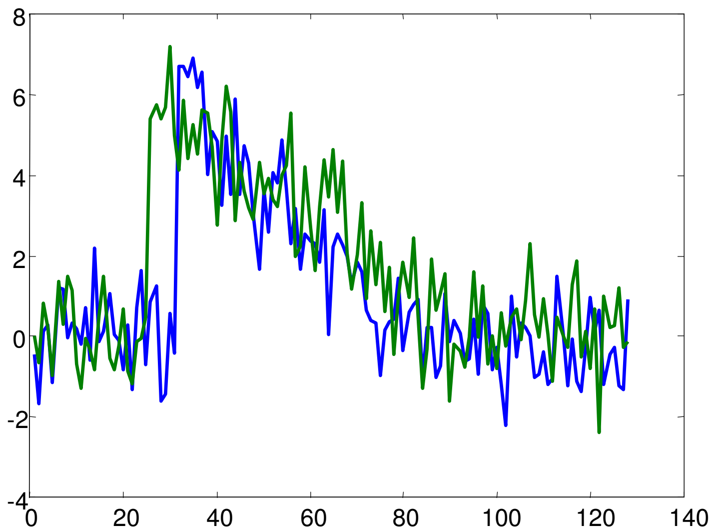
Fit the best fitting straight line to the time series, then subtract that line from the time series.

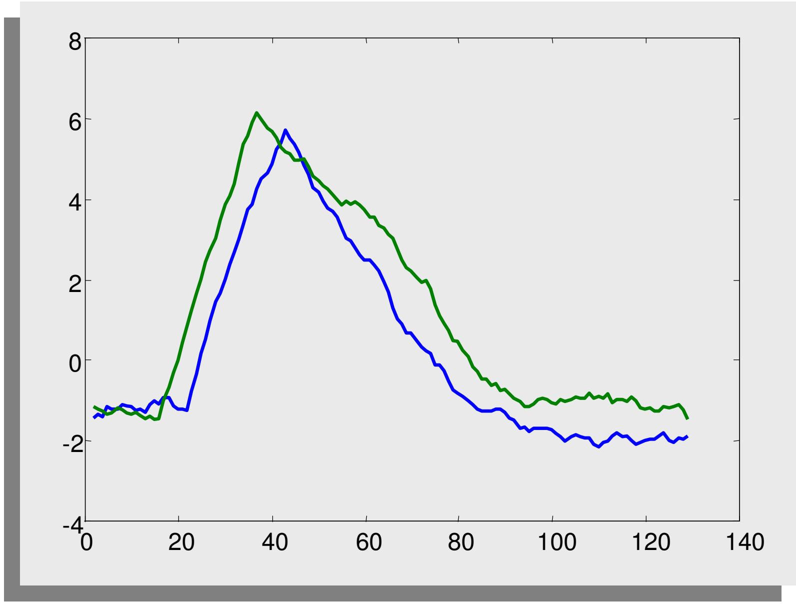
Removed **linear trend**

Removed offset translation

Removed amplitude scaling

Transformation IIII: Noise





The intuition behind removing noise is...

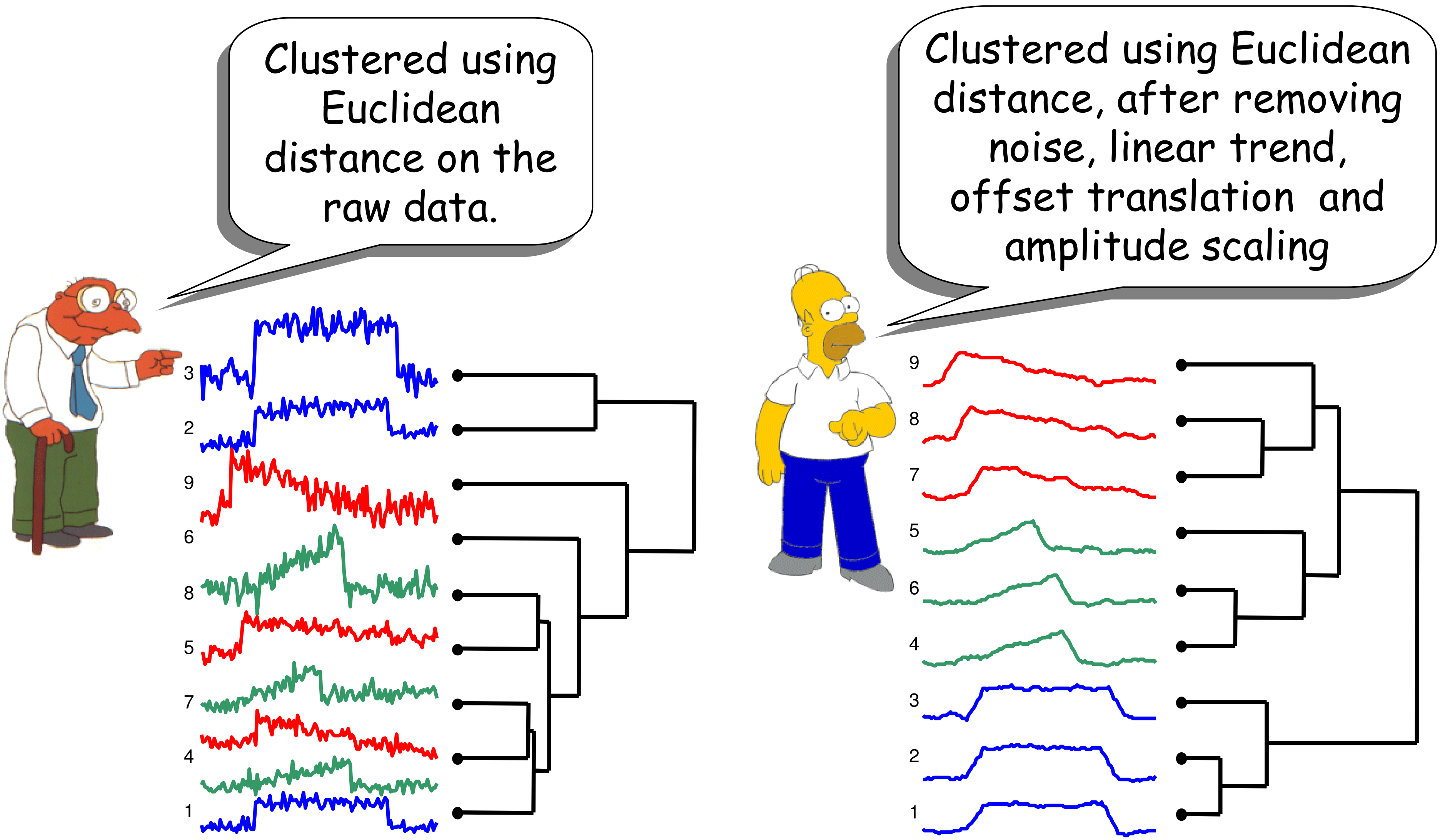
Average each datapoints value with its neighbors.

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

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

$D(Q,C)$

A Quick Experiment to Demonstrate the Utility of Preprocessing the Data



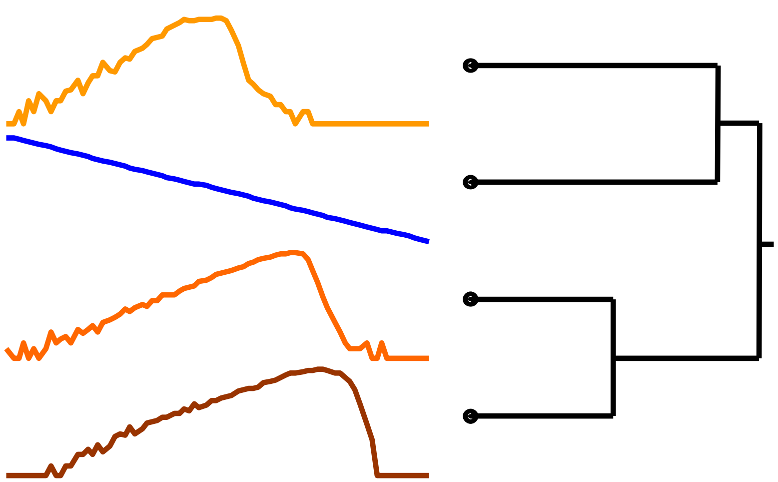
Summary of Preprocessing

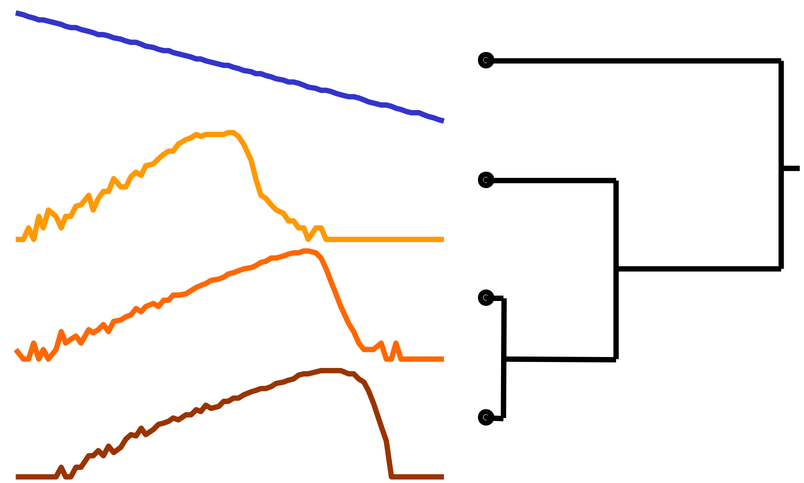
The "raw" time series may have distortions which we should remove before clustering, classification etc

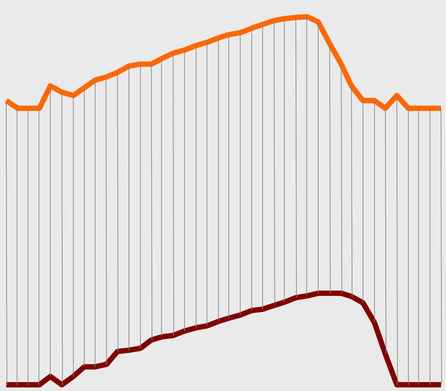
Of course, sometimes the distortions are the most interesting thing about the data, the above is only a general rule

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

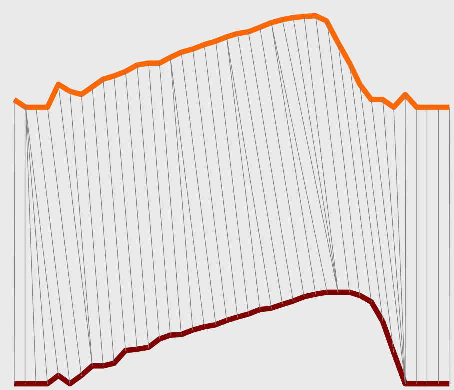






Fixed Time Axis

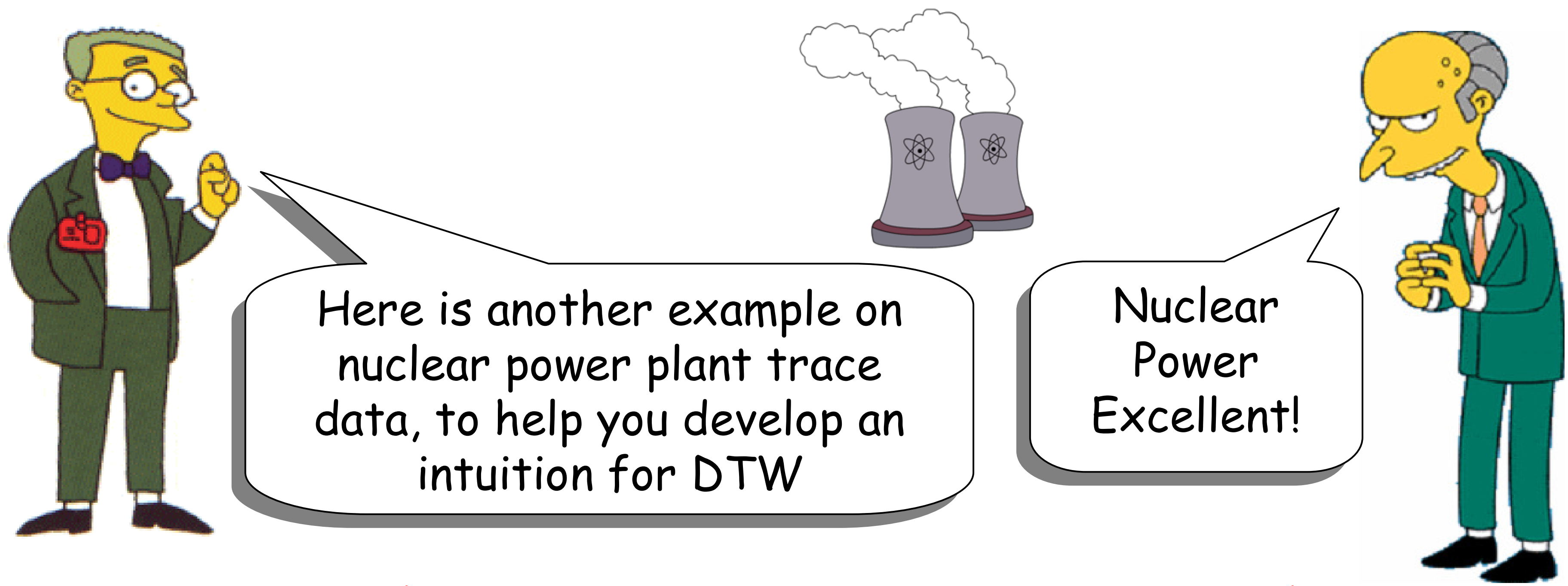
Sequences are aligned "one to one".

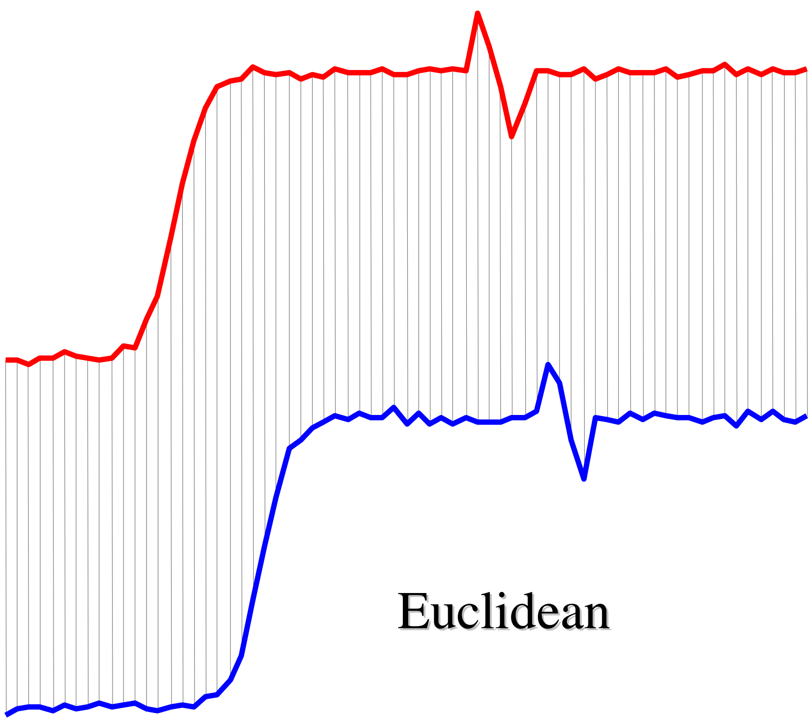


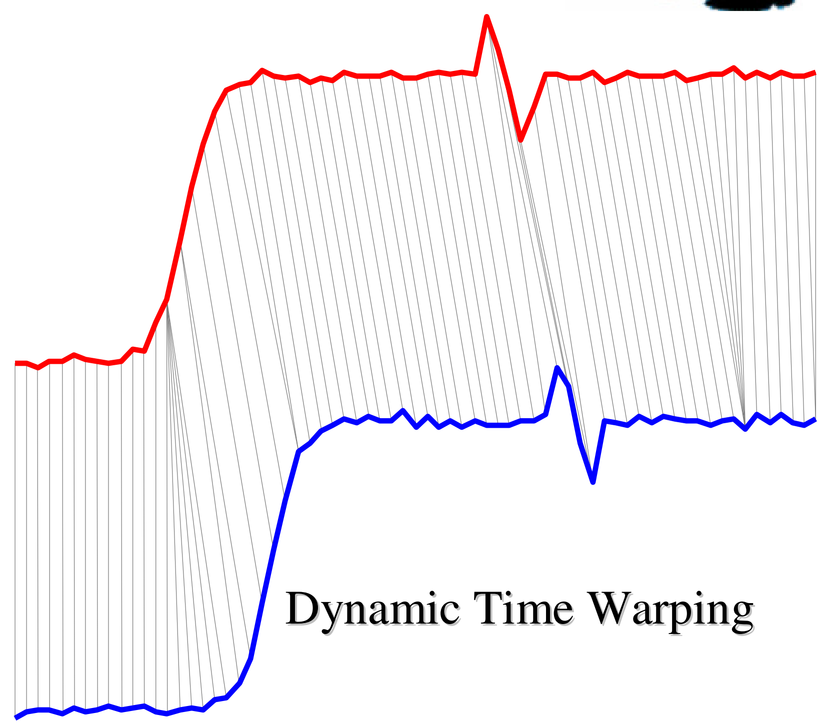
"Warped" Time Axis

Nonlinear alignments are possible.

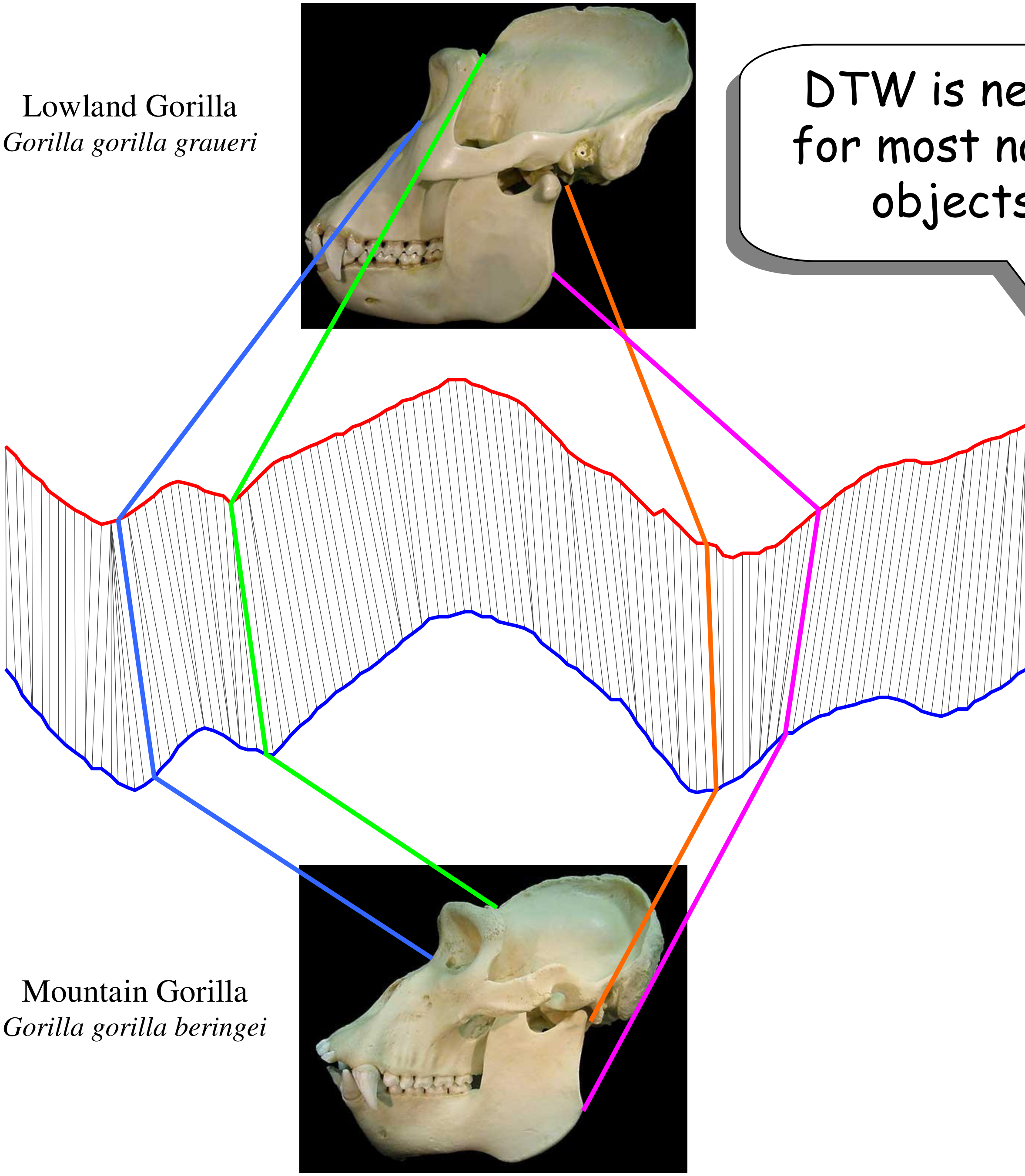
Note: We will first see the utility of DTW, then see how it is calculated.


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

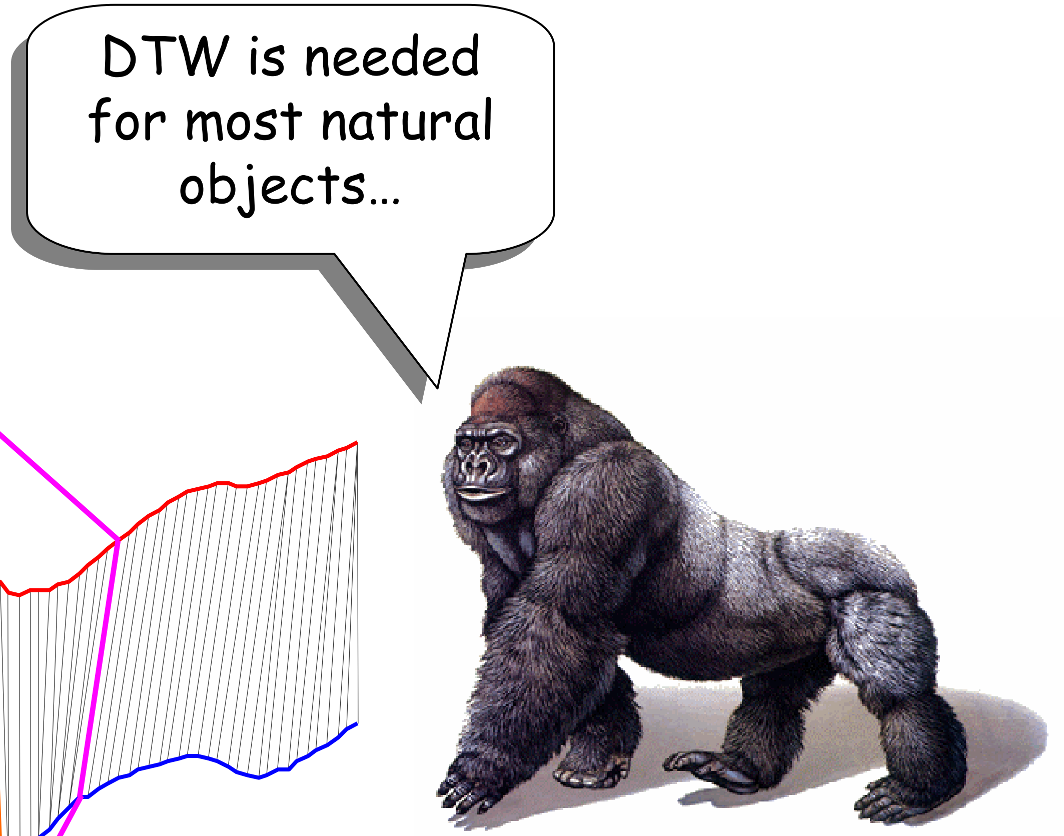

Euclidean

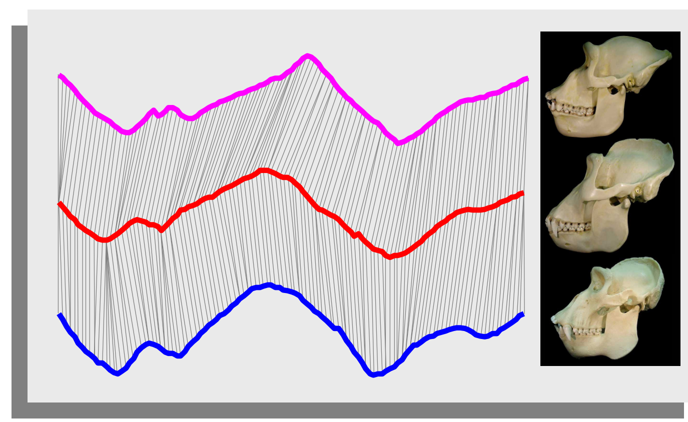

Dynamic Time Warping

Lowland Gorilla
Gorilla gorilla graueri

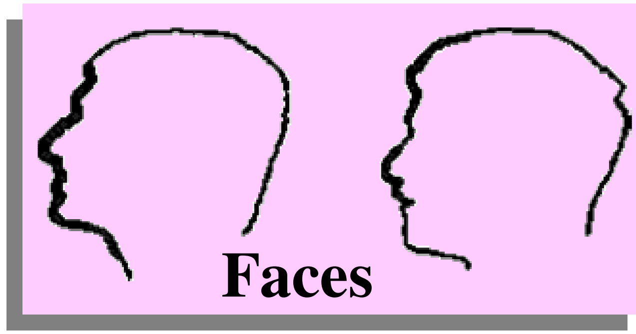


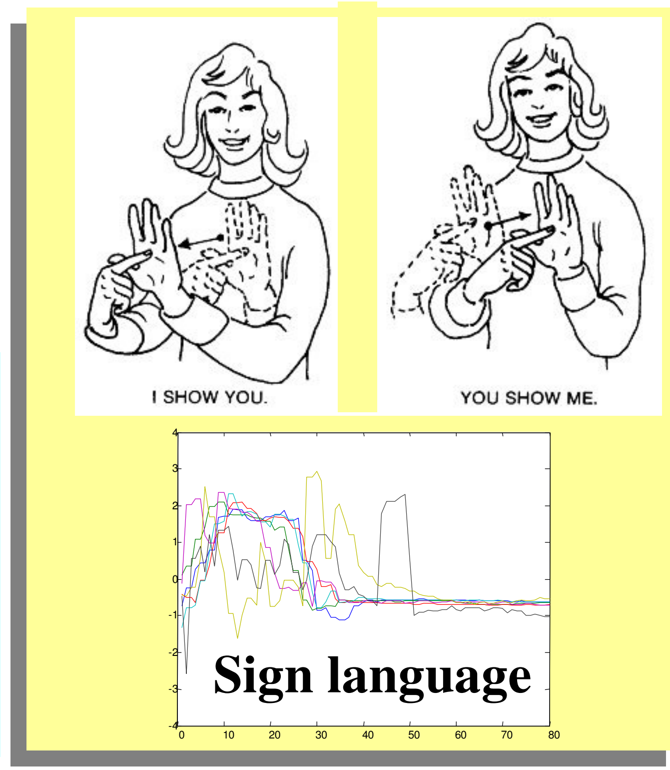
Mountain Gorilla
Gorilla gorilla beringei

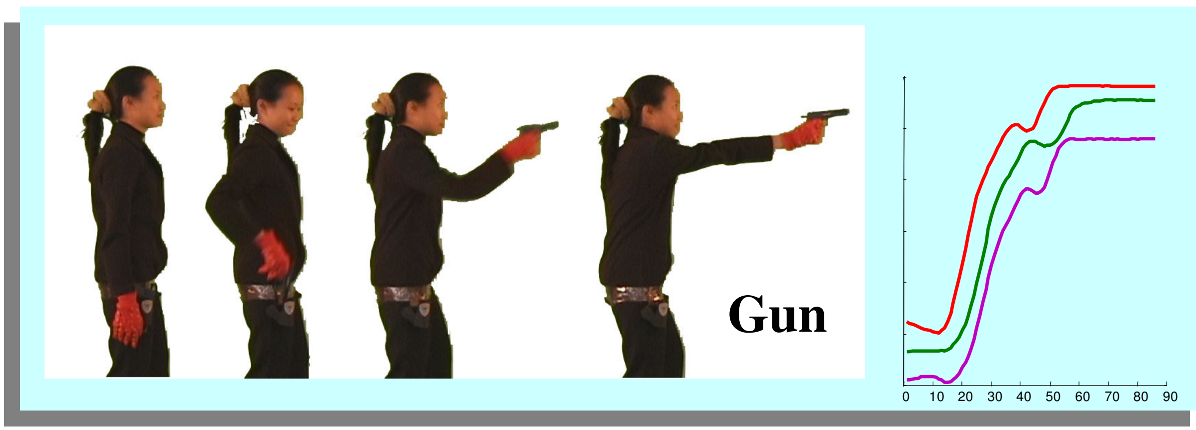


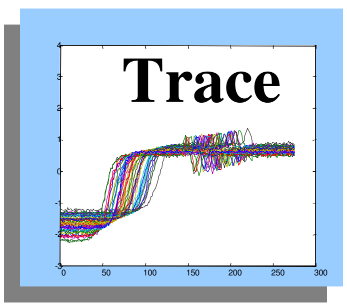


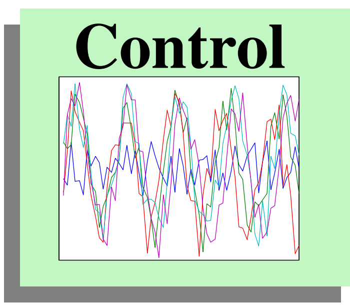
Let us compare Euclidean Distance and DTW on some problems

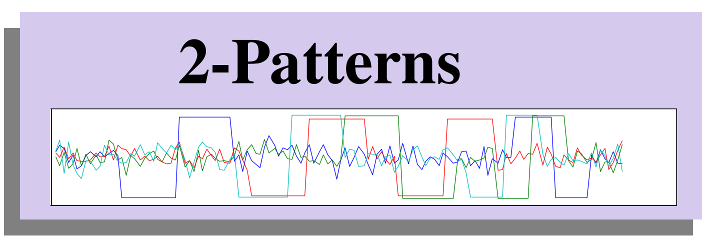


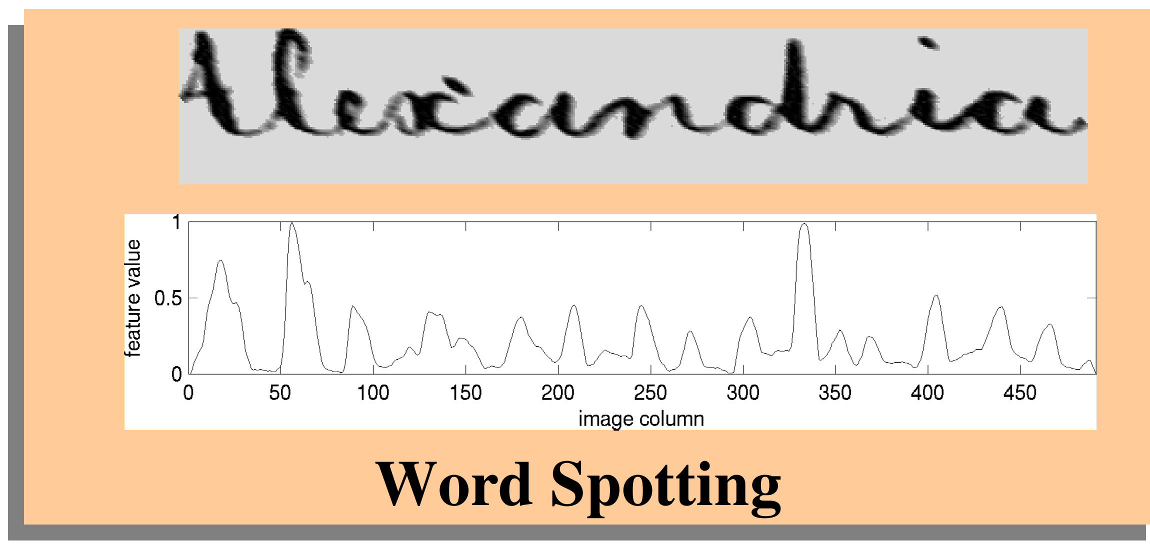






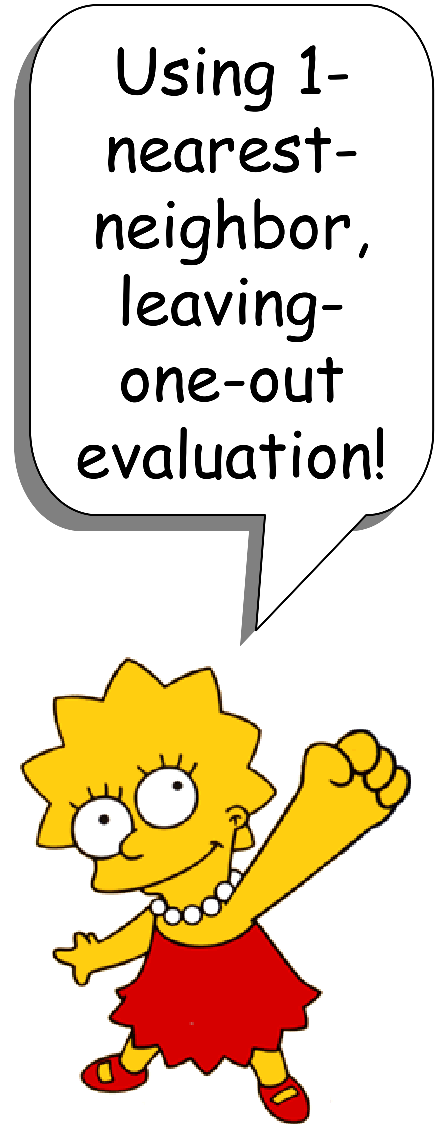






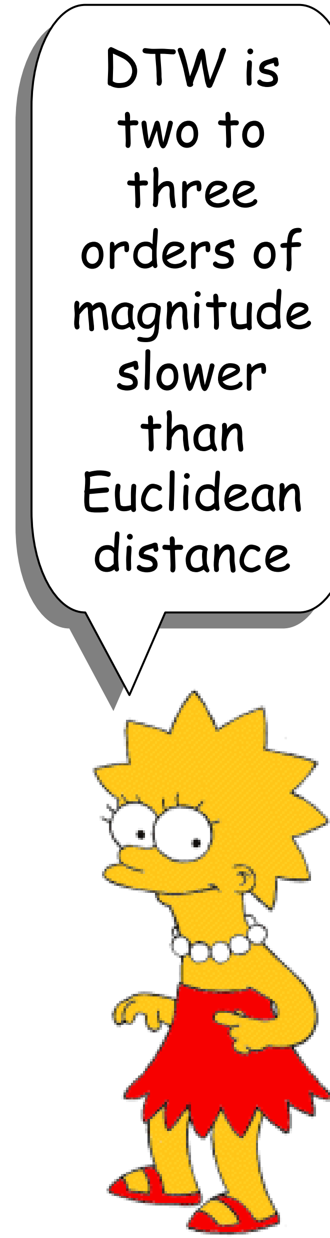
Results: Error Rate

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



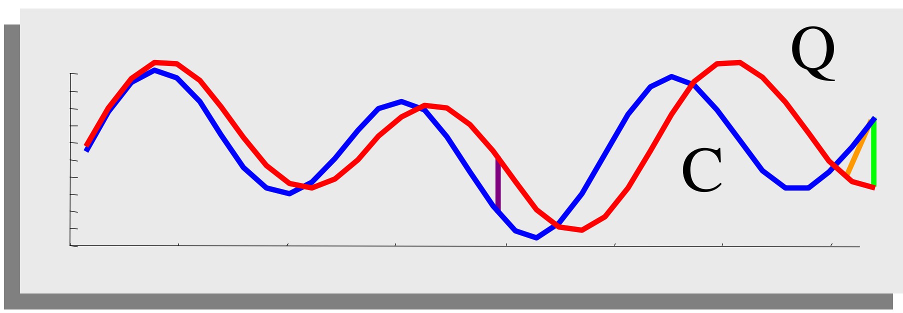
Results: Time (**msec**)

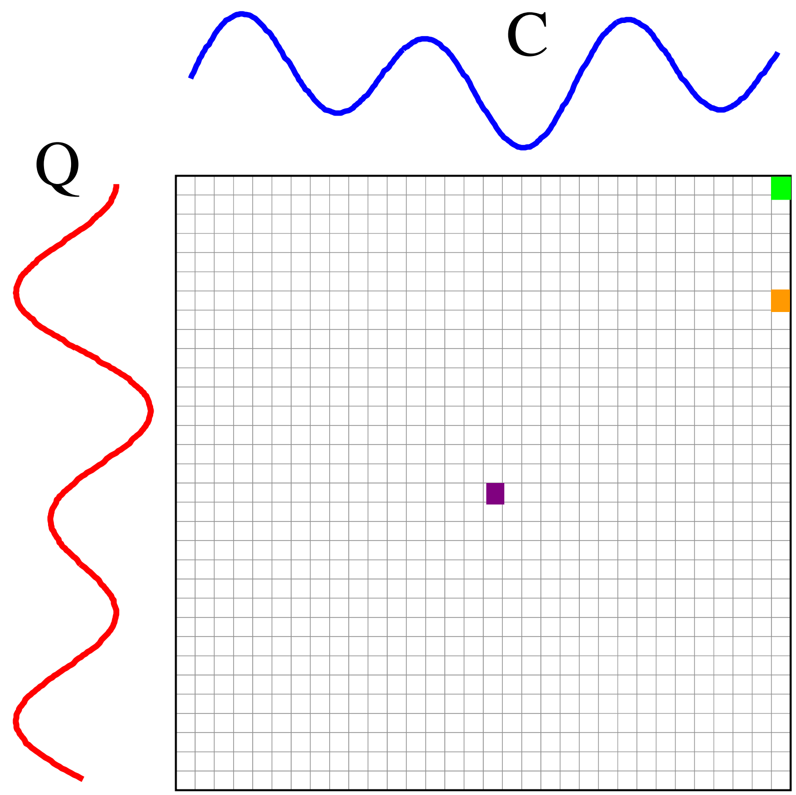
Dataset	**Euclidean**	**DTW**	
Word Spotting	40	8,600	215
Sign language	10	1,110	110
GUN	60	11,820	197
Nuclear Trace	210	144,470	687
Leaves	150	51,830	345
(4) Faces	50	45,080	901
Control Chart	110	21,900	199
2-Patterns	16,890	545,123	32



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.

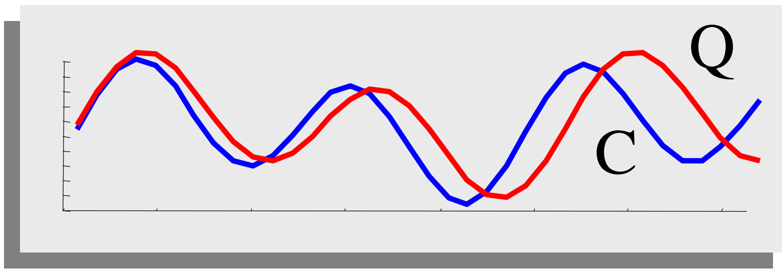


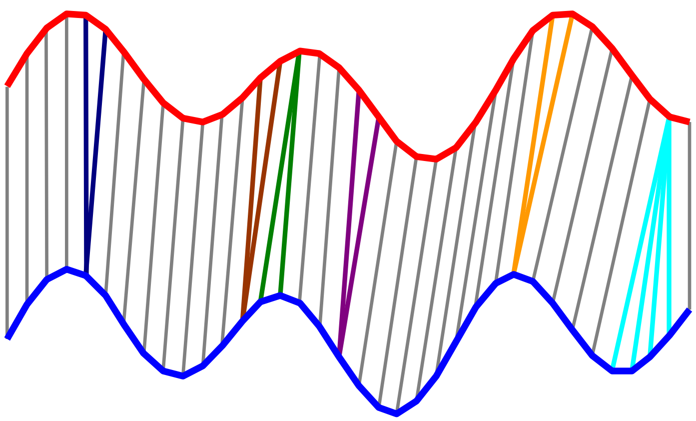


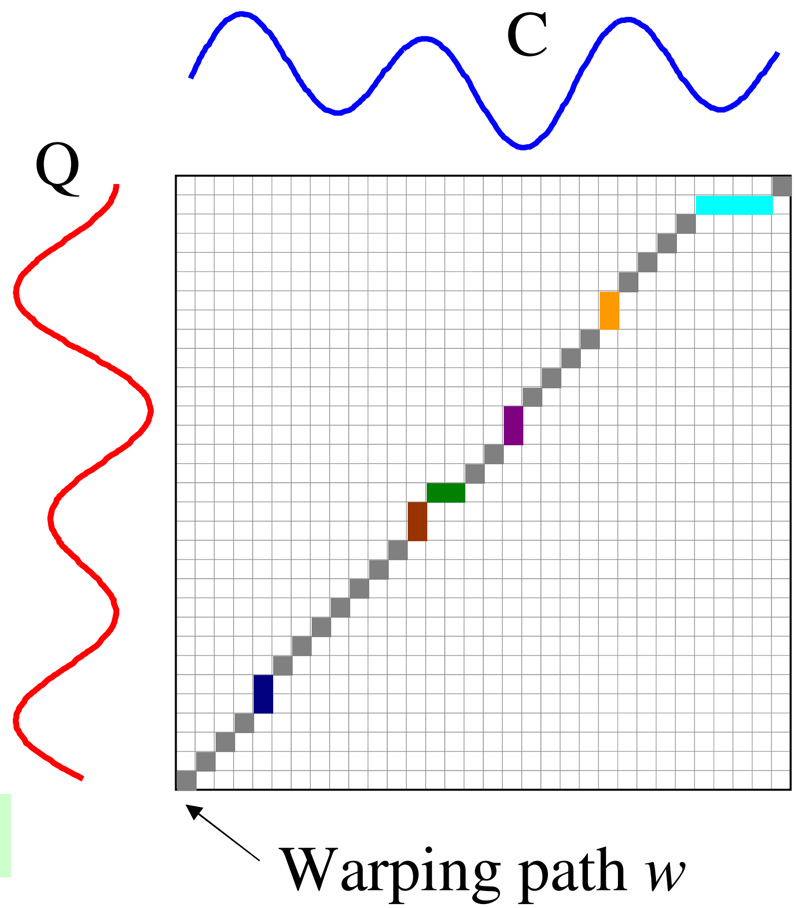
How is DTW Calculated? II

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

$$DTW(Q,C) = \min\left\{ \sqrt{\sum_{k=1}^{K} w_k} \middle/ K \right.$$



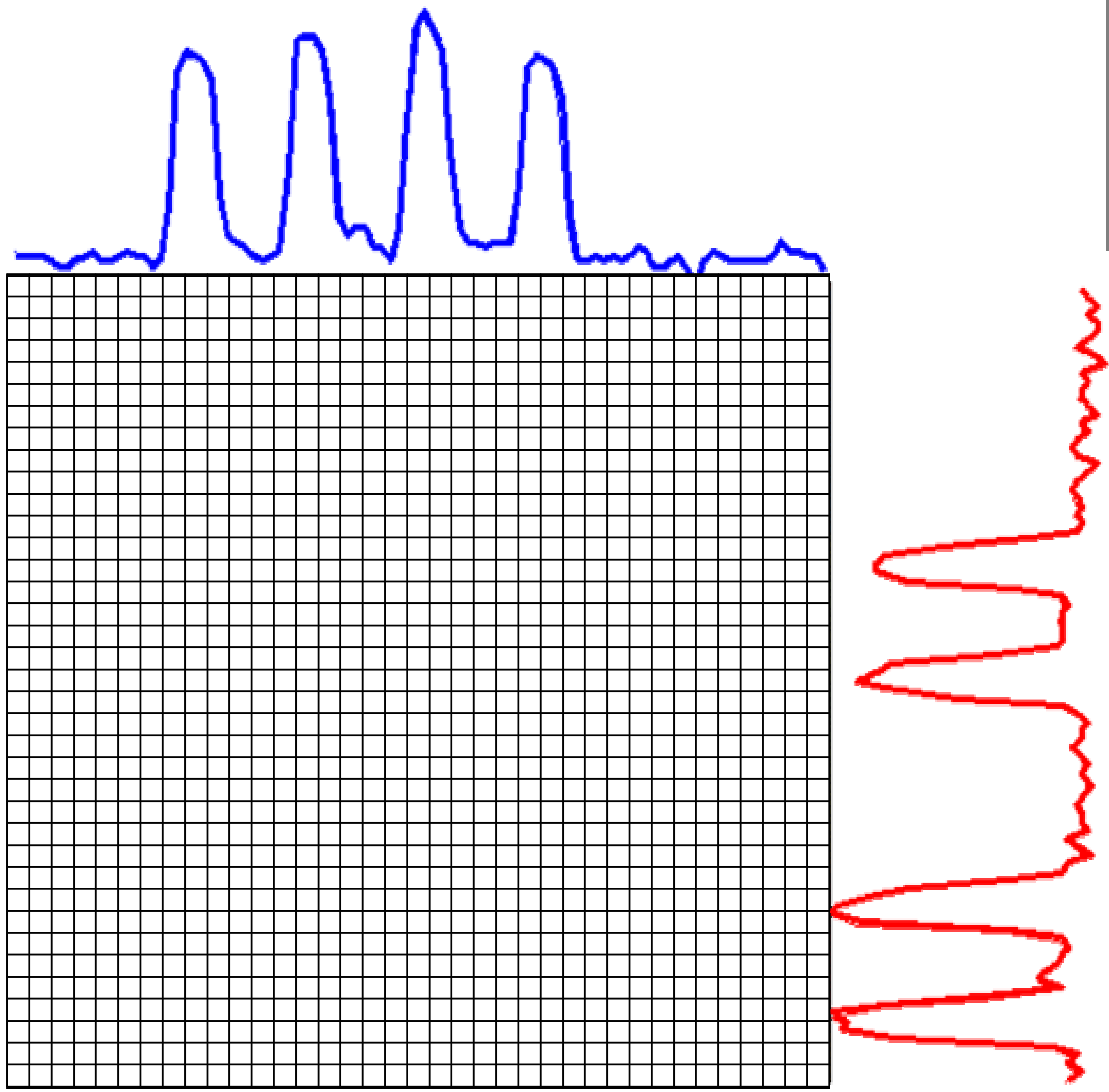


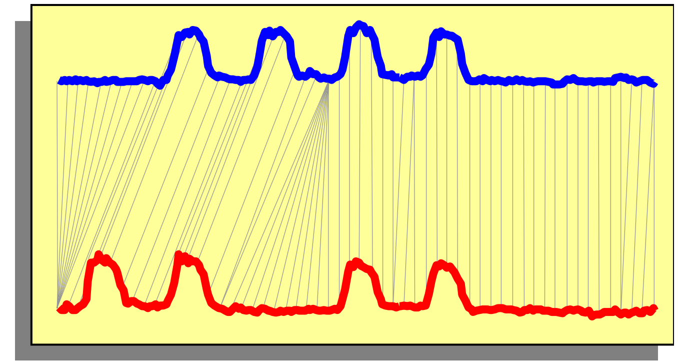


This recursive function gives us the minimum cost path

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

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

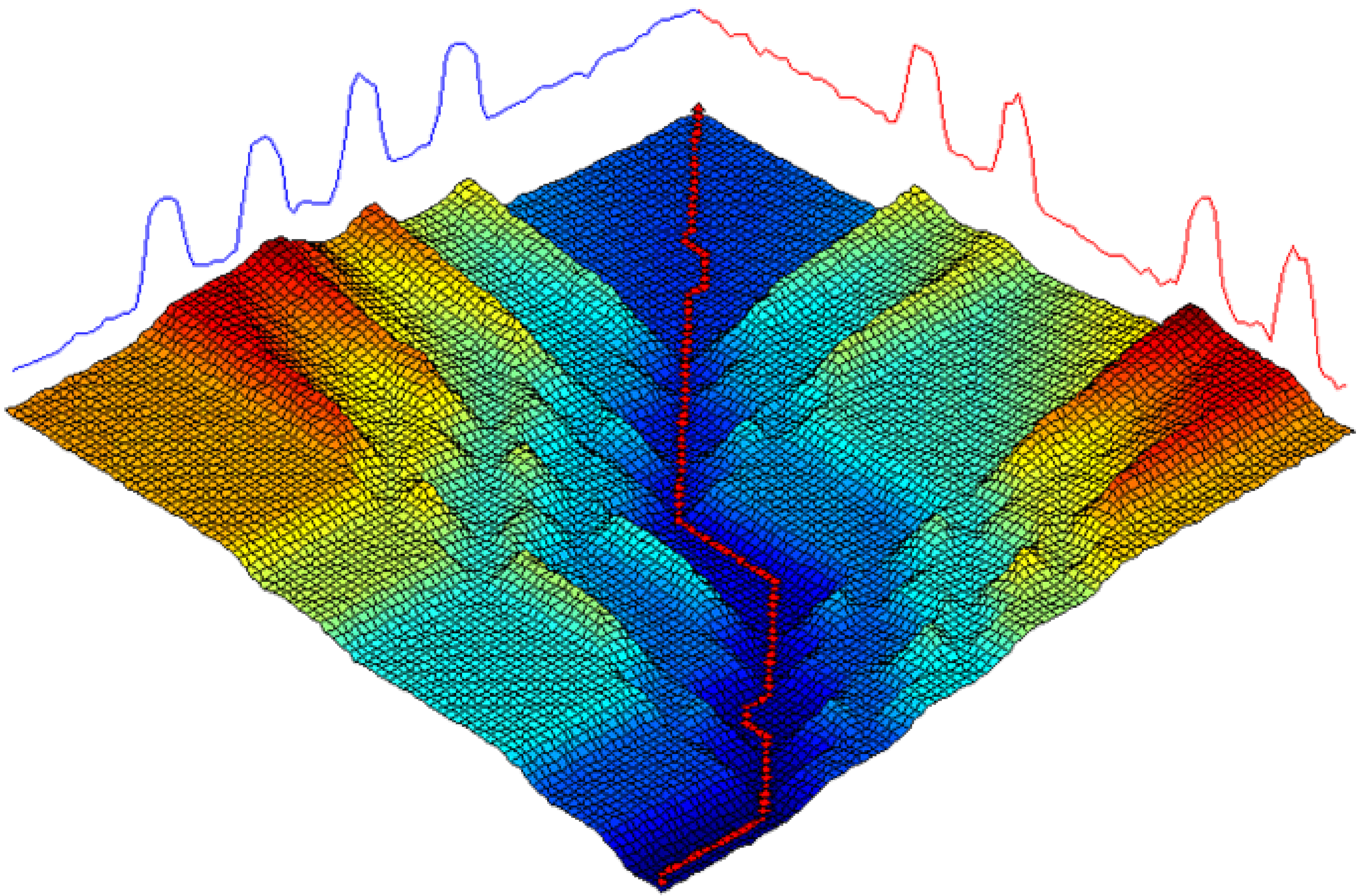




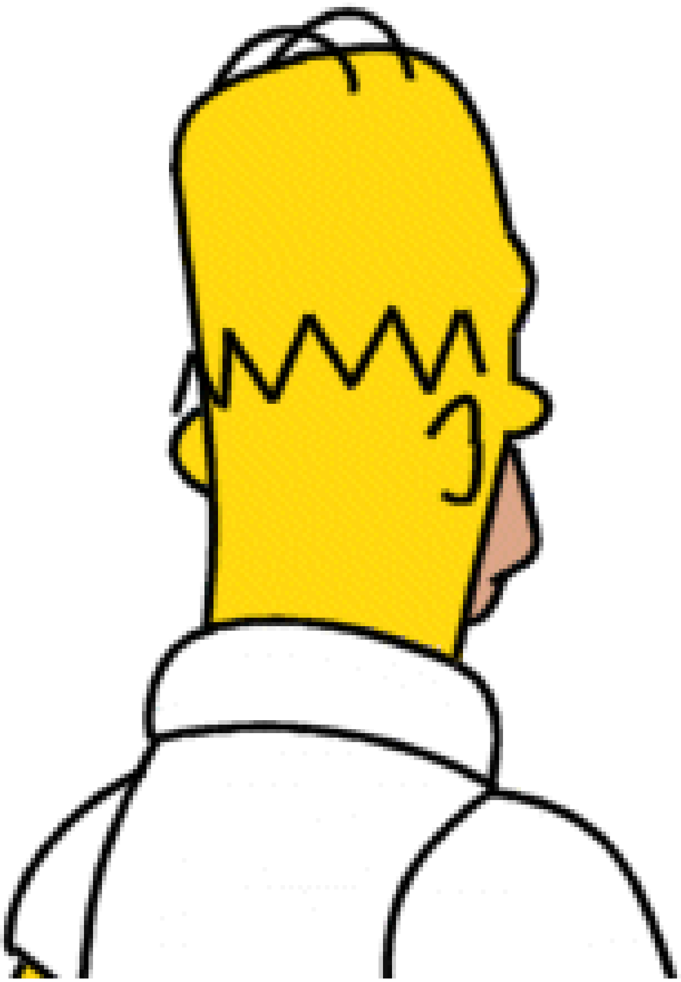
This example shows 2 one-week periods from the power demand time series.

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…

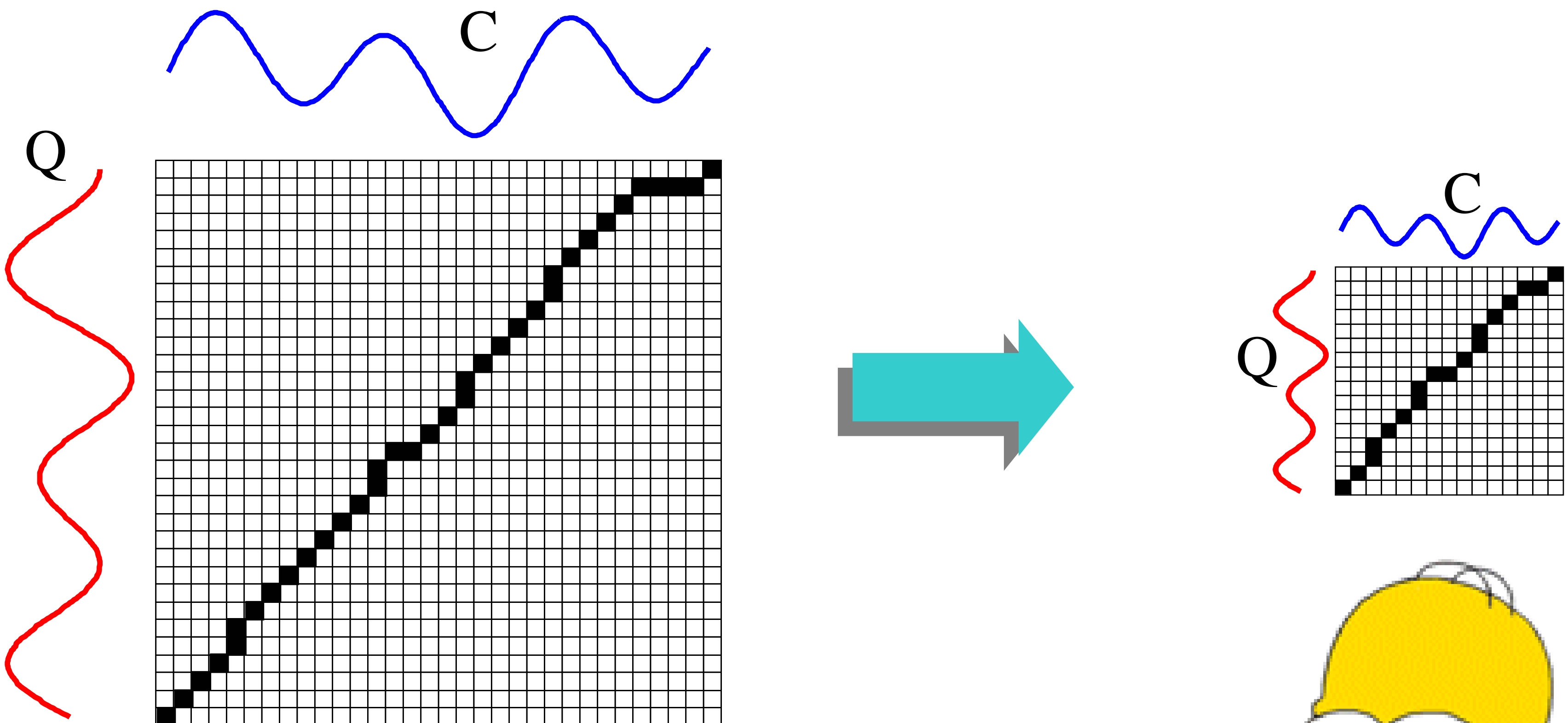


• Dynamic Time Warping gives **much better** results than Euclidean distance on virtually all problems.

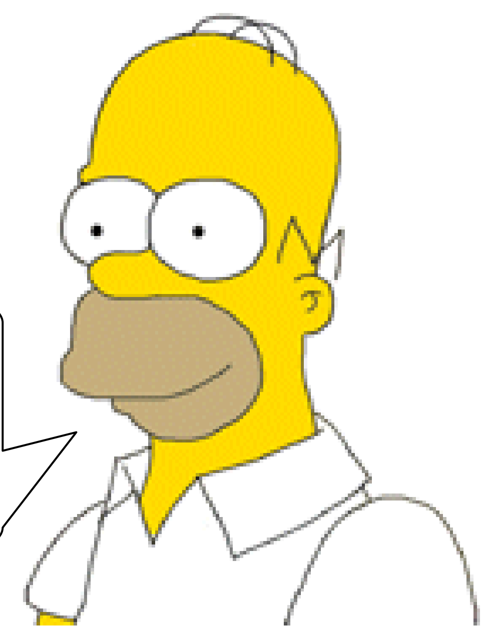
• Dynamic Time Warping is very very slow to calculate!

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

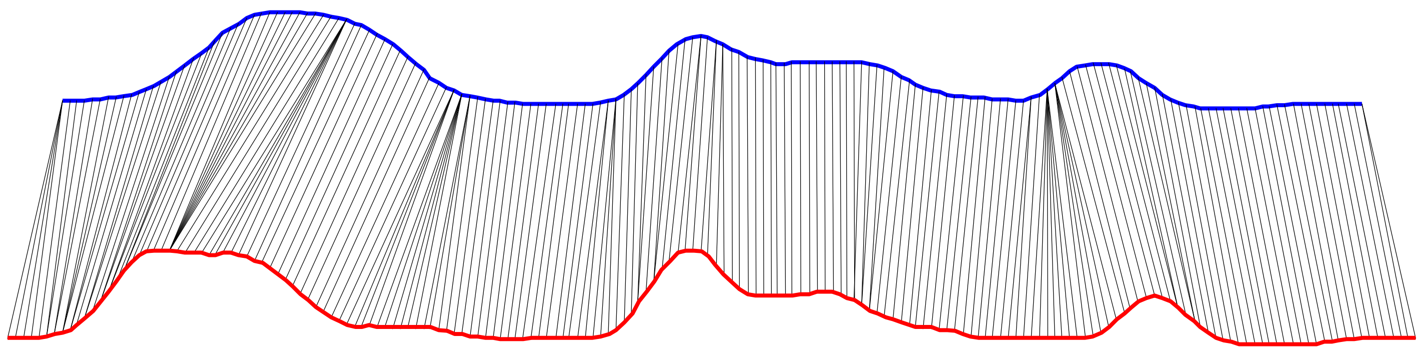
Fast Approximations to Dynamic Time Warp Distance I

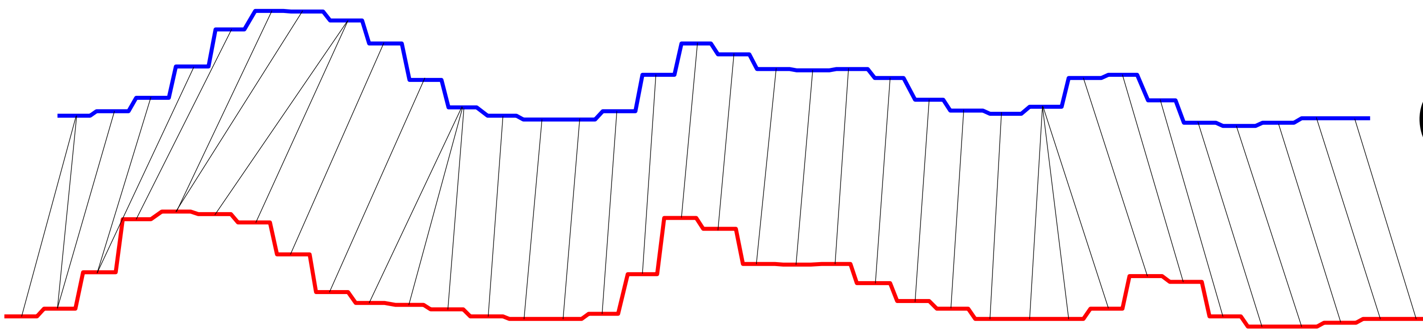


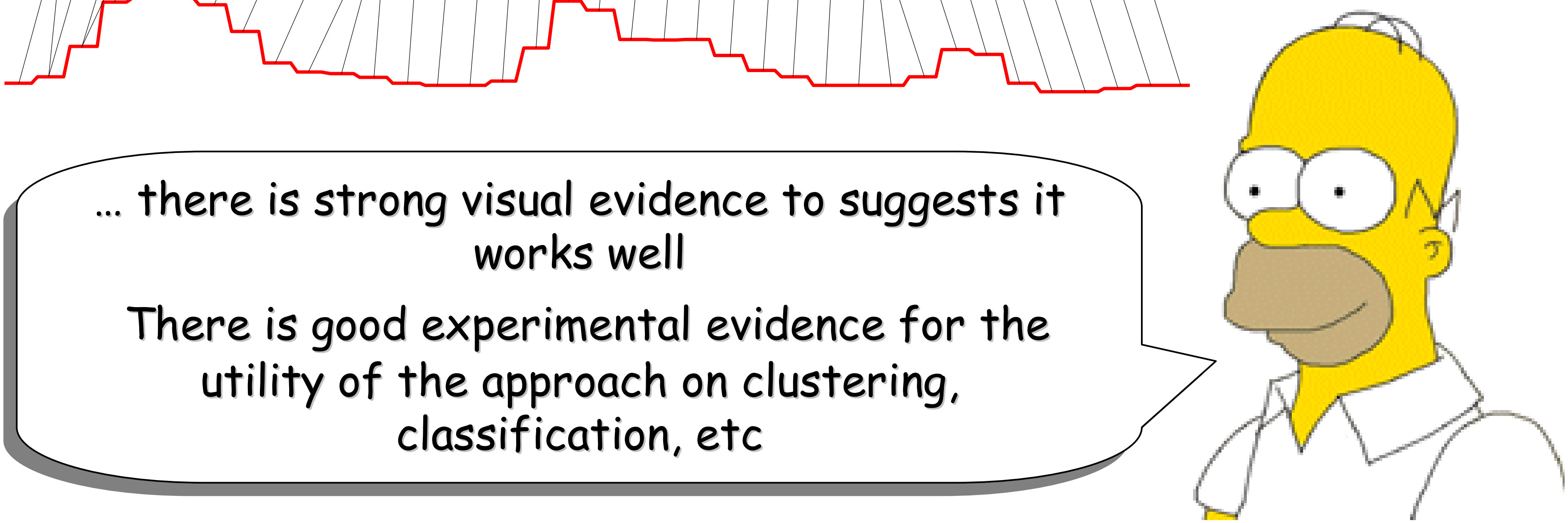
Simple Idea: Approximate the time series with some compressed or downsampled representation, and do DTW on the new representation. How well does this work...



Fast Approximations to Dynamic Time Warp Distance II

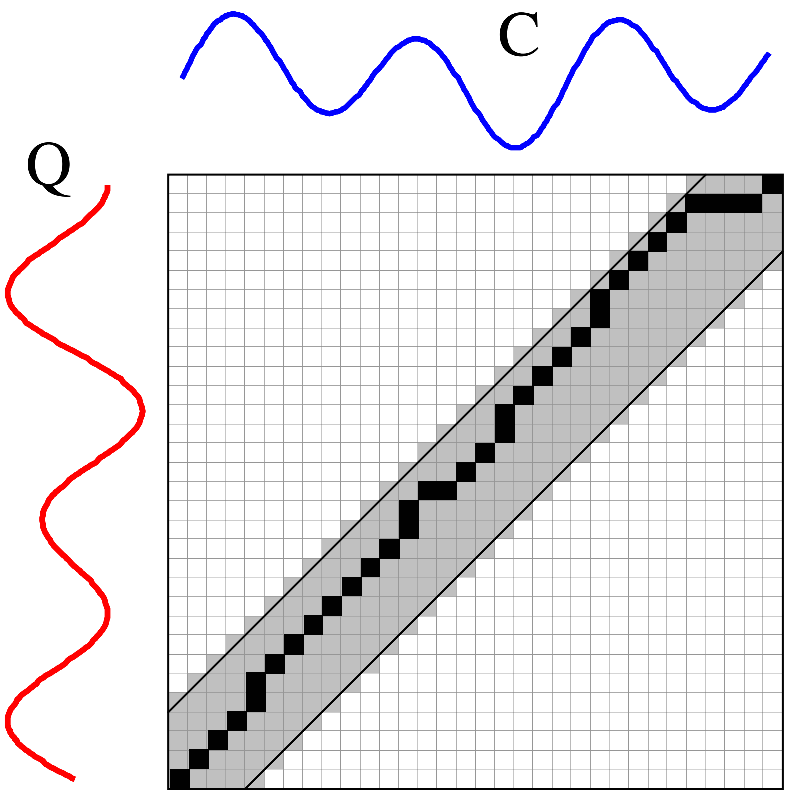




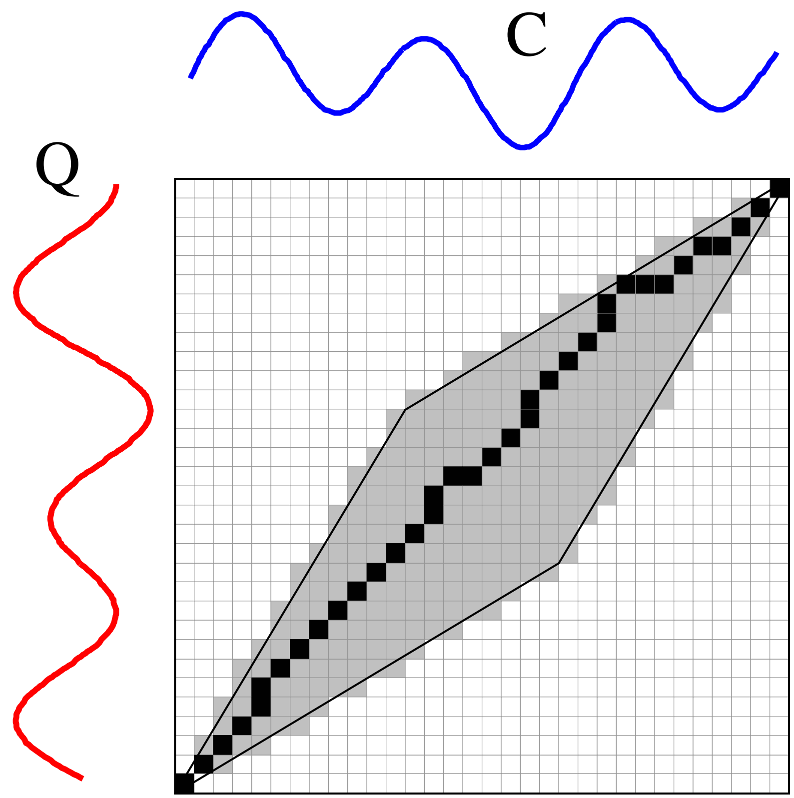


Global Constraints

- Slightly speed up the calculations
- Prevent pathological warpings

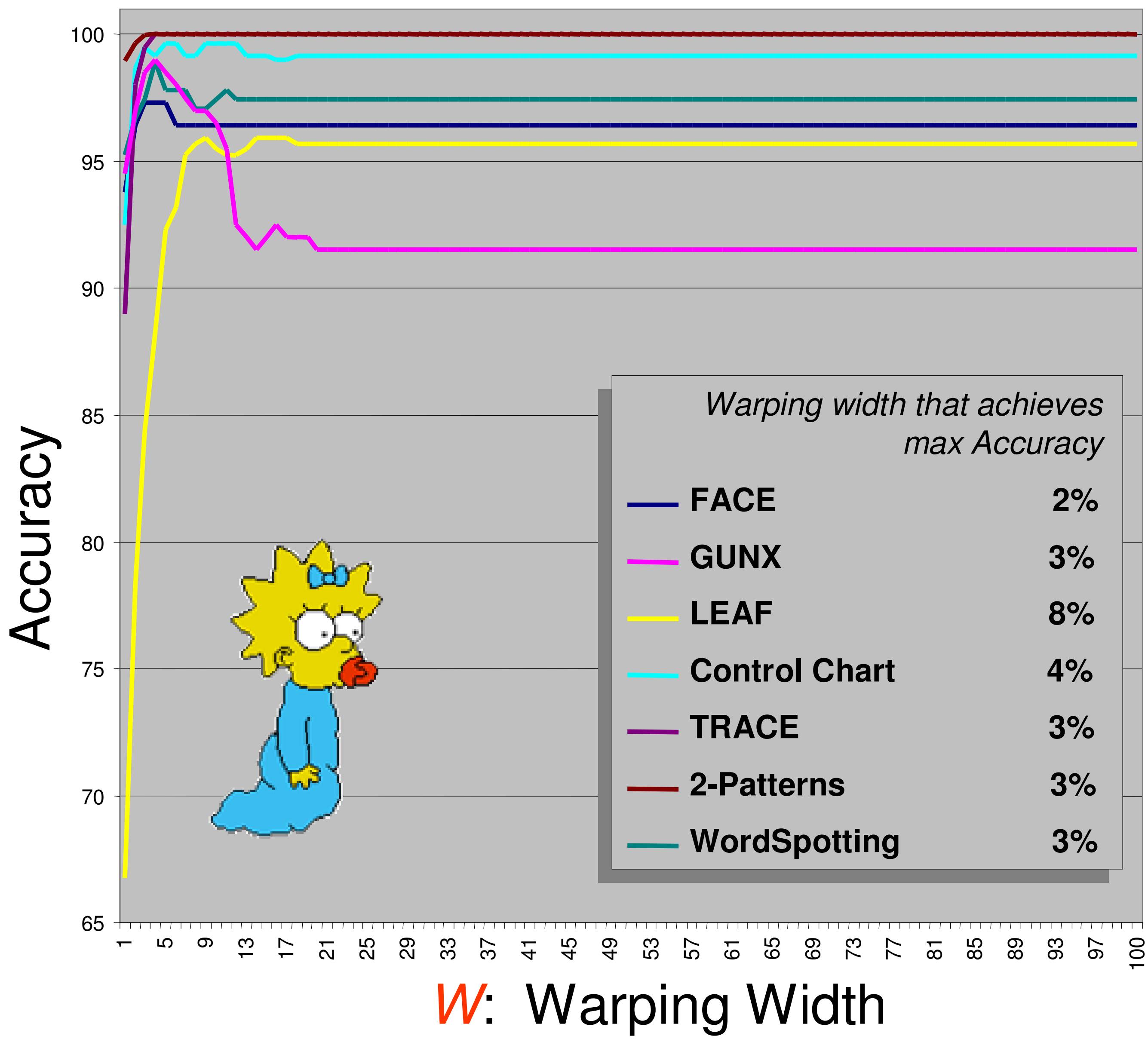


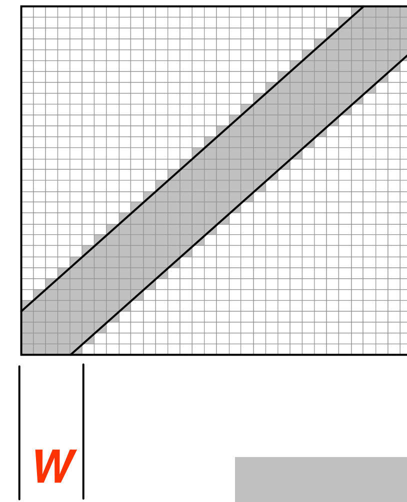
Sakoe-Chiba Band



Itakura Parallelogram

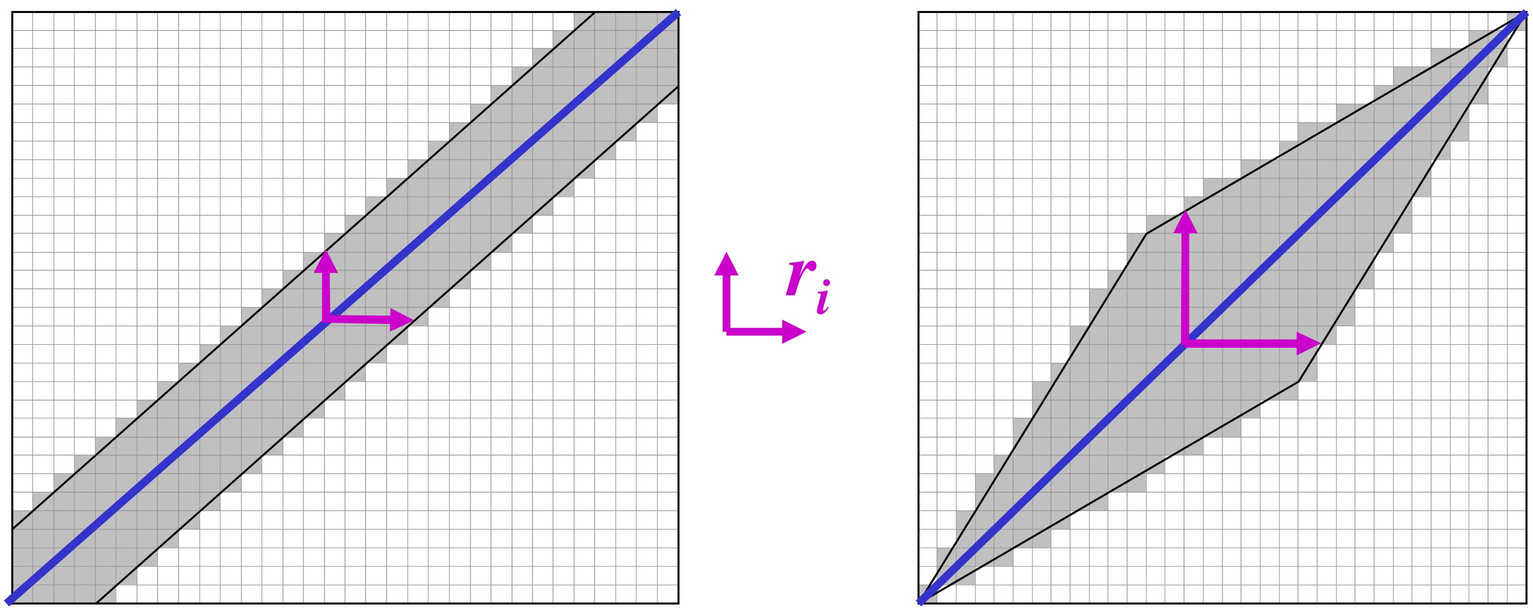
Accuracy vs. Width of Warping Window





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

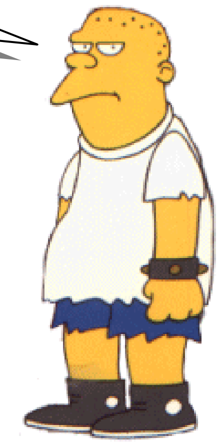
Where r is a term defining allowed range of warping for a given point in a sequence.



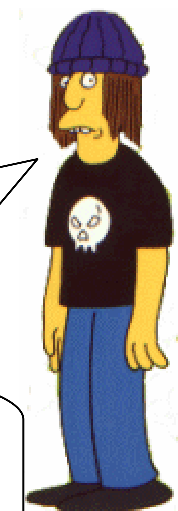
Sakoe-Chiba Band

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

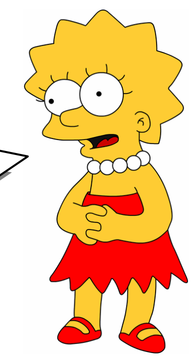
$$\text{lower\_bound\_distance}(A,B) \leq \text{DTW}(A,B)$$

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
```

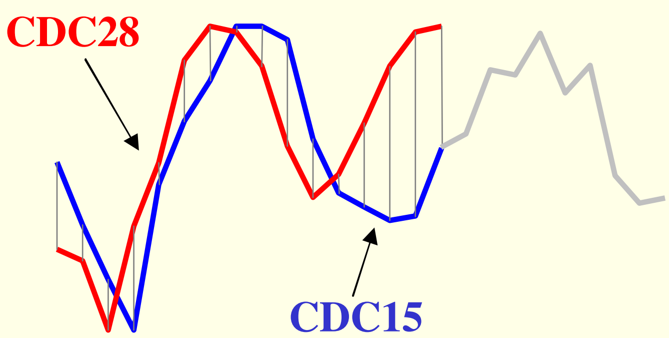
Try to use a cheap lower bounding calculation as often as possible.



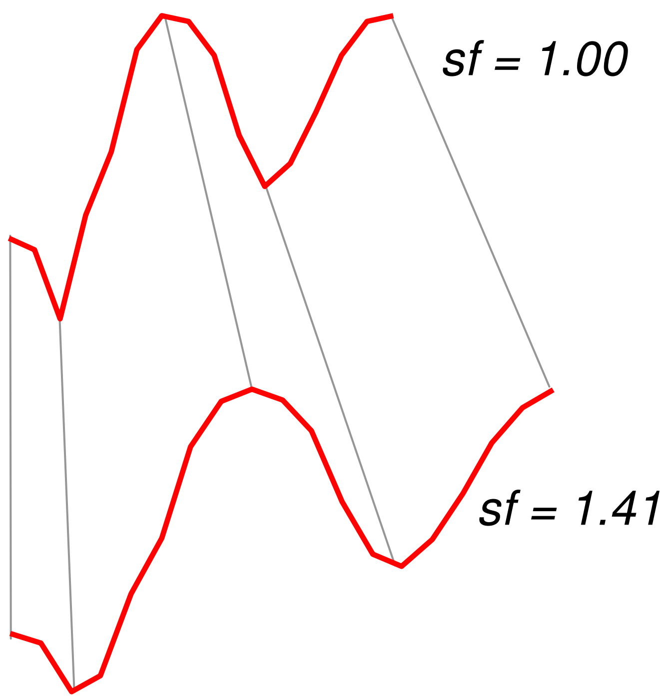
Only do the expensive, full calculations when it is absolutely necessary

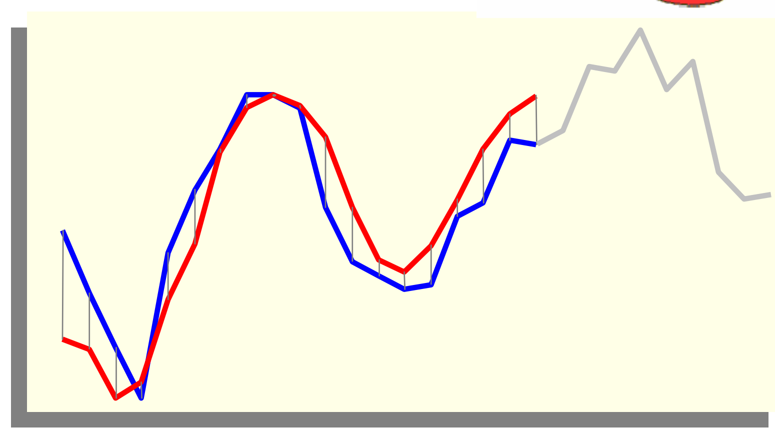
Uniform Scaling I

Two genes that are known to be functionally related…



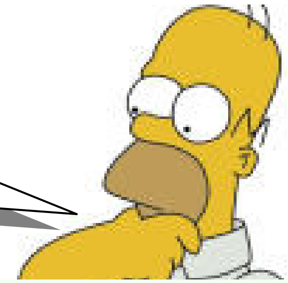
Sometimes global or *uniform scaling* is as important as DTW

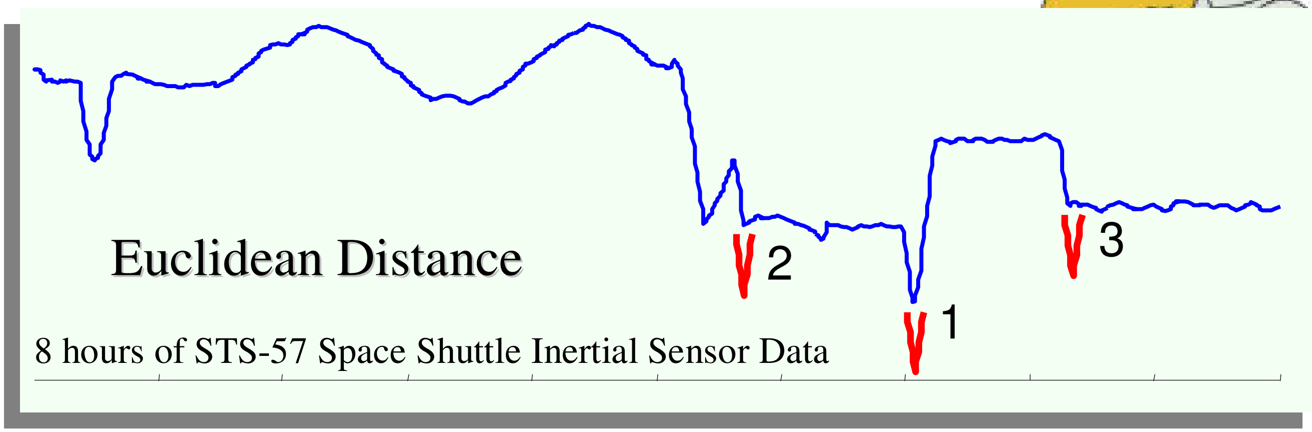


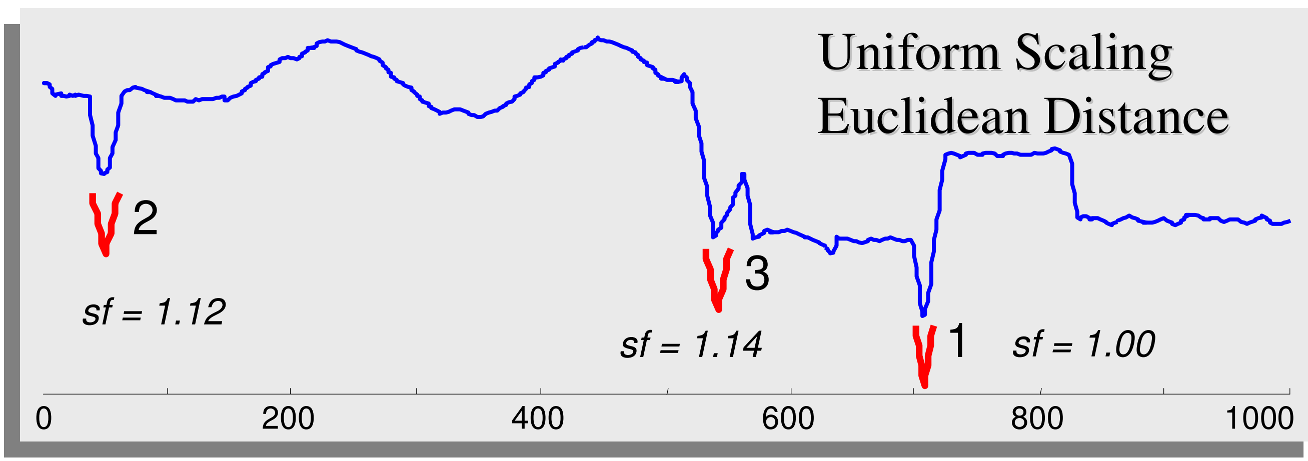


Uniform Scaling II

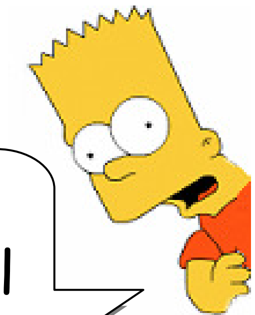
Without scaling, matches 2 and 3 seem unintuitive

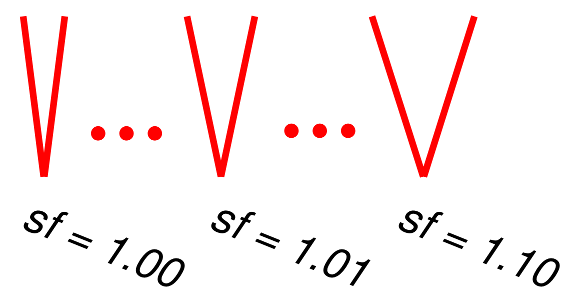






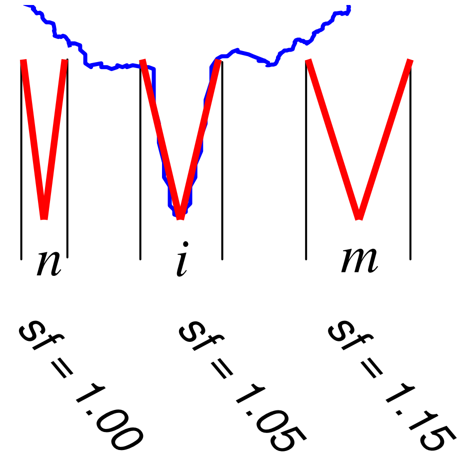
We need to test all scalings!



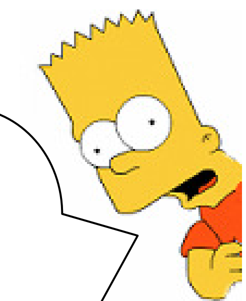


Here is some notation, the shortest scaling we consider is length *n*, and the largest is length *m*.

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

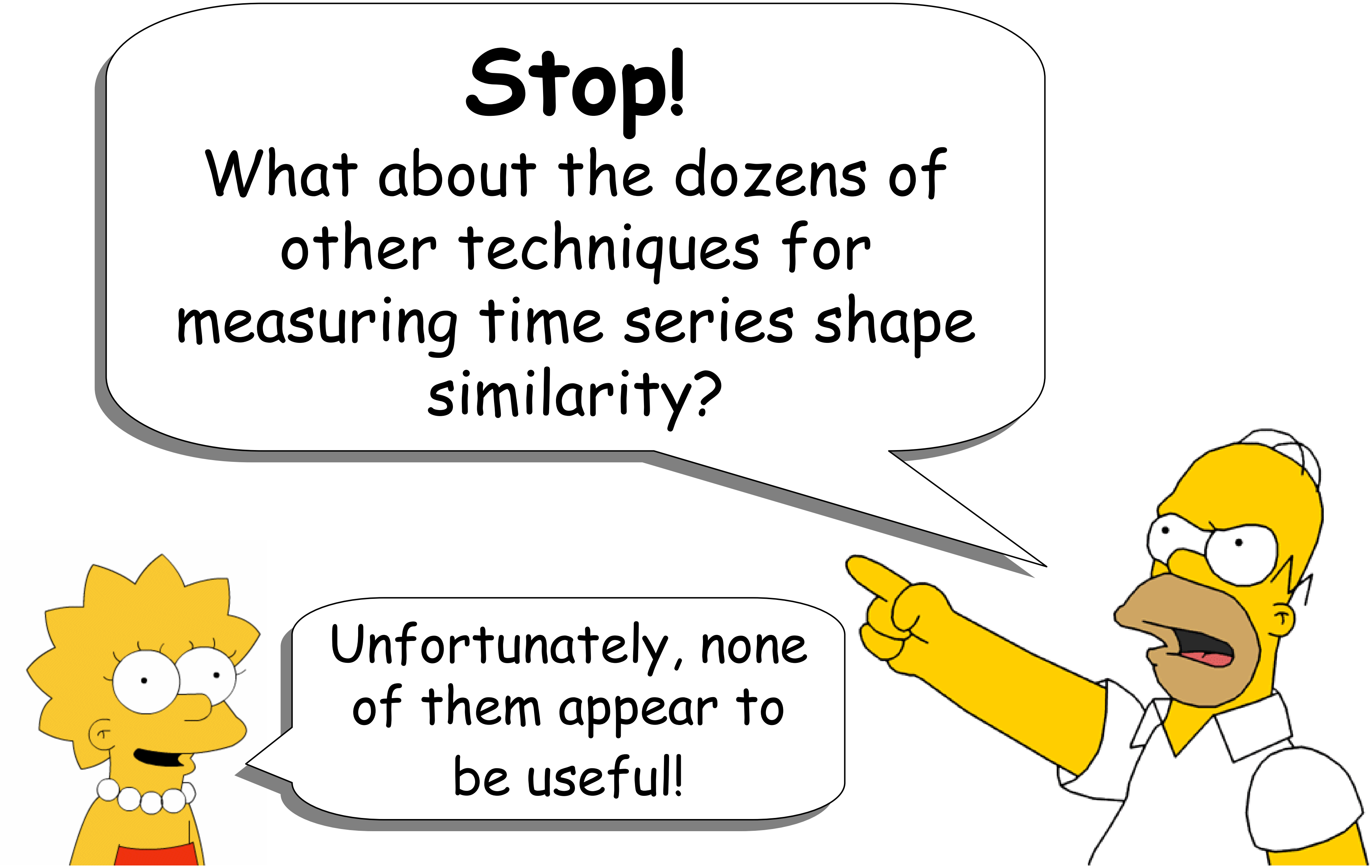



```
Algorithm: Test_All_Scalings(Q,C)
      best_match_val       = inf;
      best_scaling_factor  = null;
       for p = n to m
       | QP = rescale(Q,p);
       |  distance = squared_Euclidean_distance(QP, C[1..p
       |  if distance <  best_match_val
       |  |  best_match_val = distance;
       |  |  best_scaling_factor = p/n;
       | end;
       end;
   return(best_match_val, best_scaling_factor)
```



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…

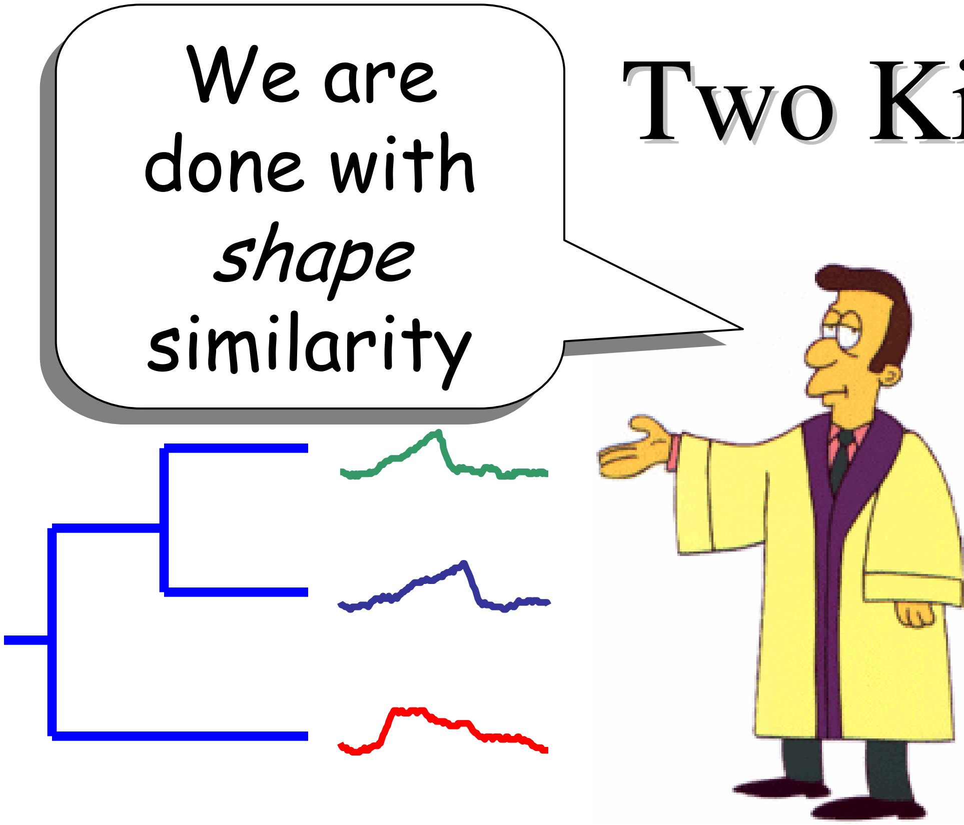
Only Euclidean and DTW Distance are Useful

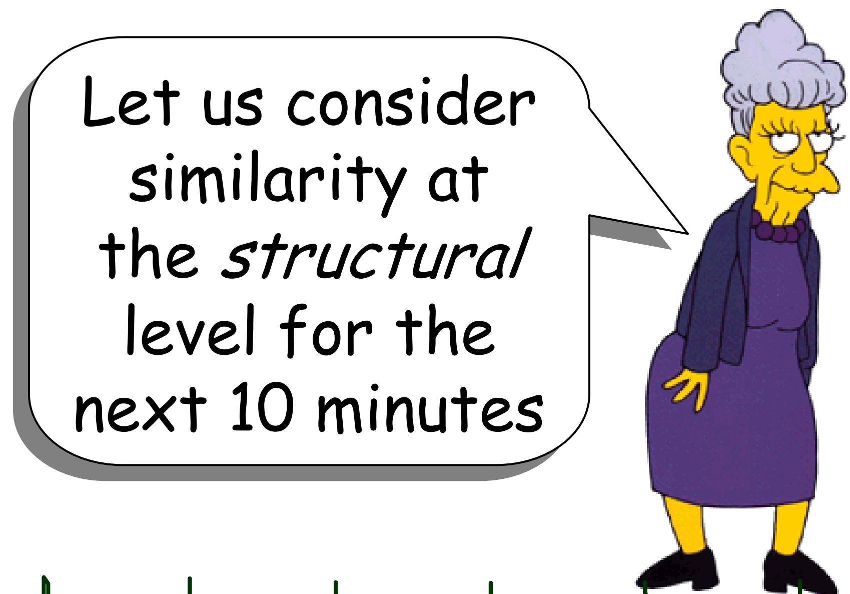


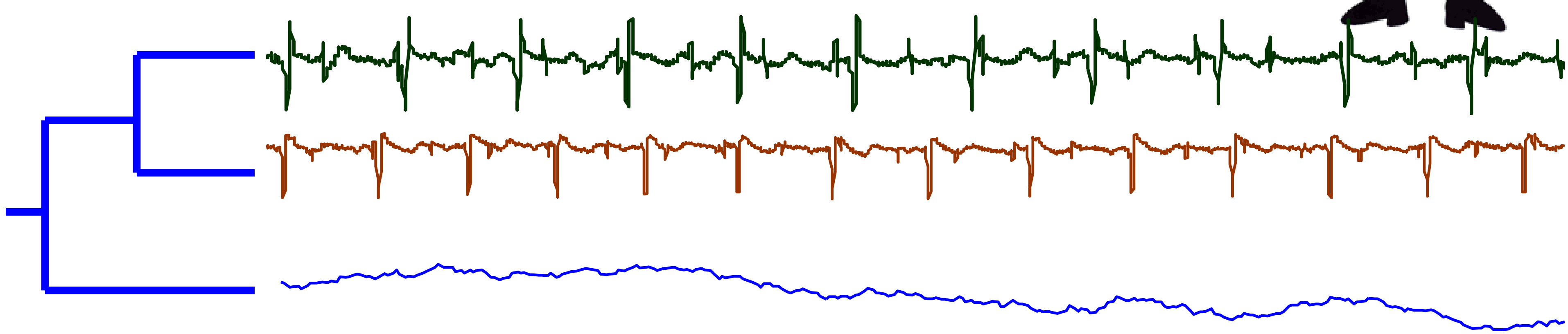
Classification Error Rates on
two publicly available datasets

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**

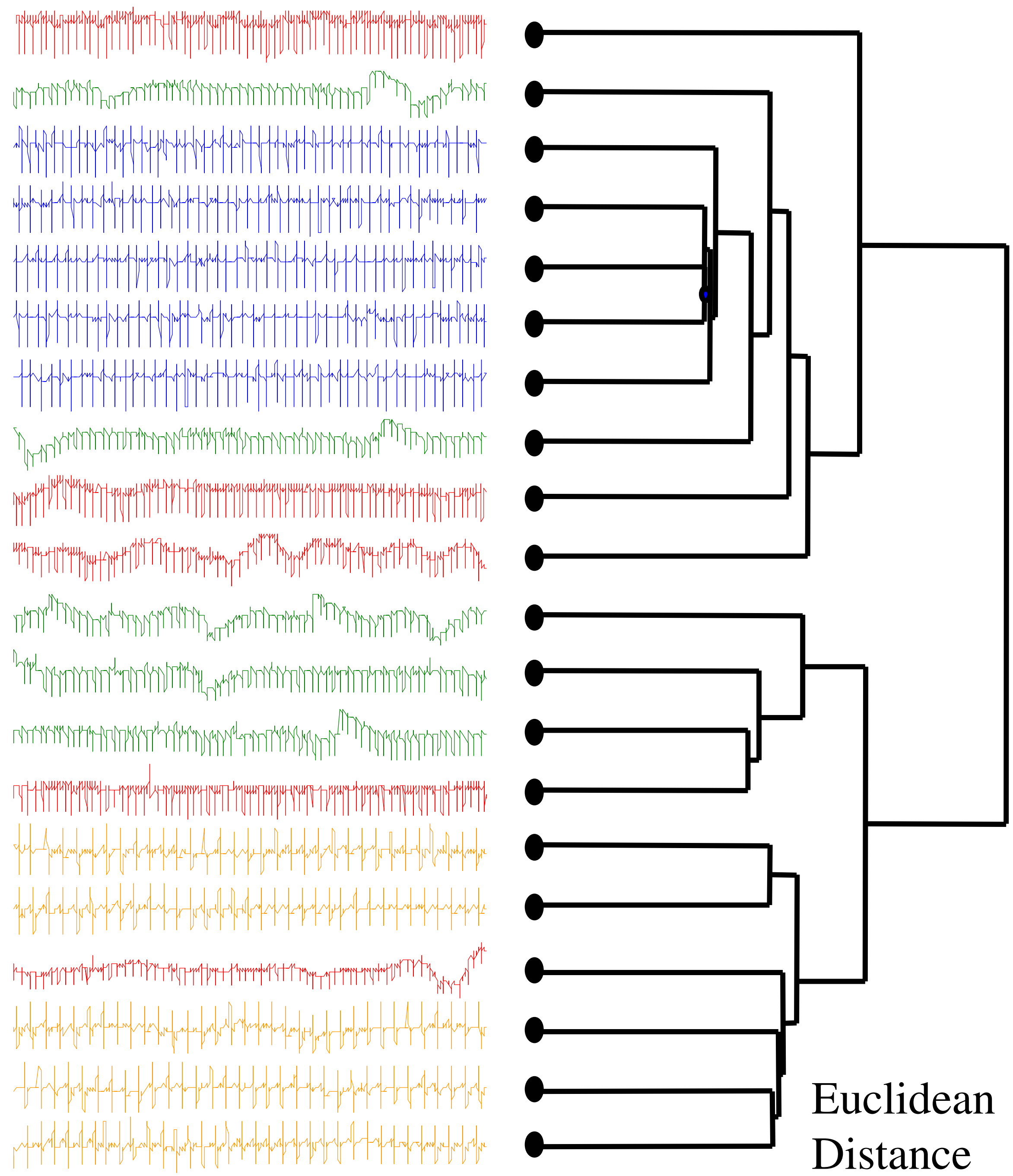
Two Kinds of Similarity





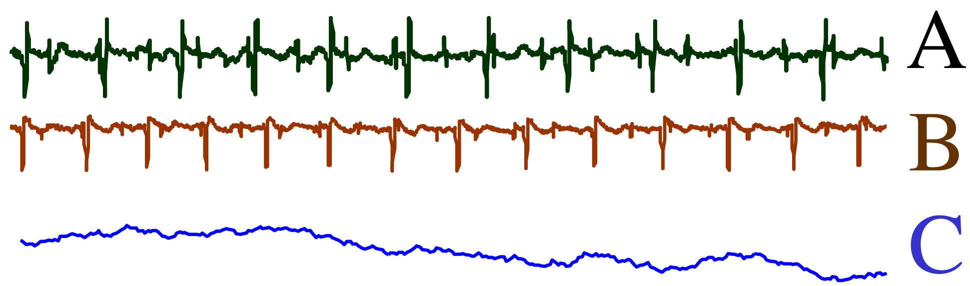


For long time series, *shape* based similarity will give very poor results. We need to measure similarly based on high level *structure*



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**?

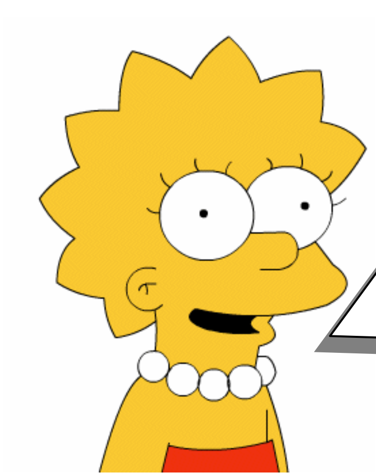
Feature-based Classification of Time-series Data

Nanopoulos, Alcock, and Manolopoulos

- features?
- distance measure/ learning algorithm?

Learning Algorithm

multi-layer perceptron neural network



Makes sense, but when we looked at the *same* dataset, we found we could be better classification accuracy with Euclidean distance!

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

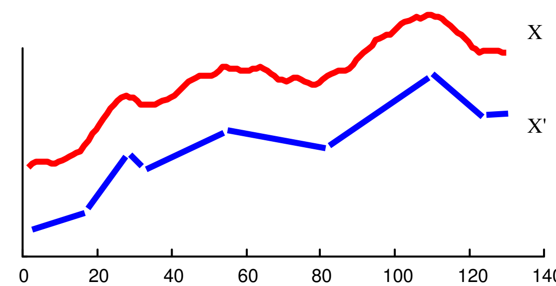
Part 1

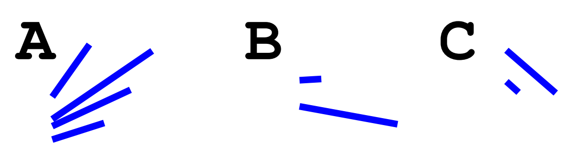
- features?
- distance measure/ learning algorithm?

Features

The parameters of a Markov Model

The time series is first converted to a piecewise linear model





Distance Measure

"Viterbi-Like" Algorithm

Variations independently developed by Li and Biswas, Ge and Smyth, Lin, Orgun and Williams etc

There tends to be lots of parameters to tune…

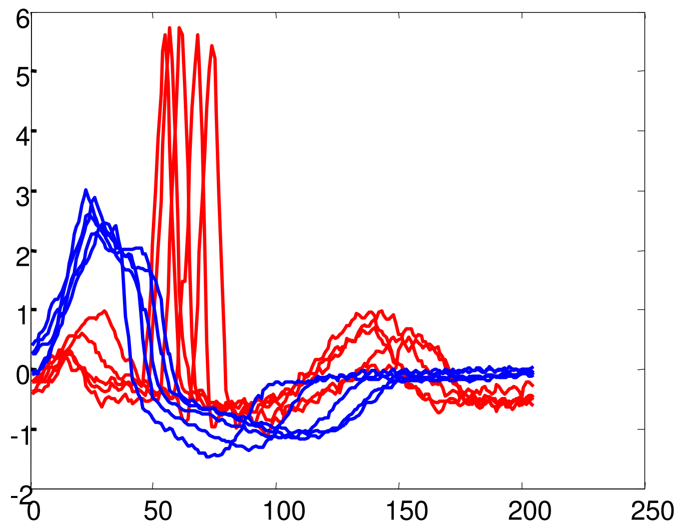
	A	B	C
A	0.1	0.4	0.5
B	0.4	0.2	0.2
C	0.5	0.2	0.3

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

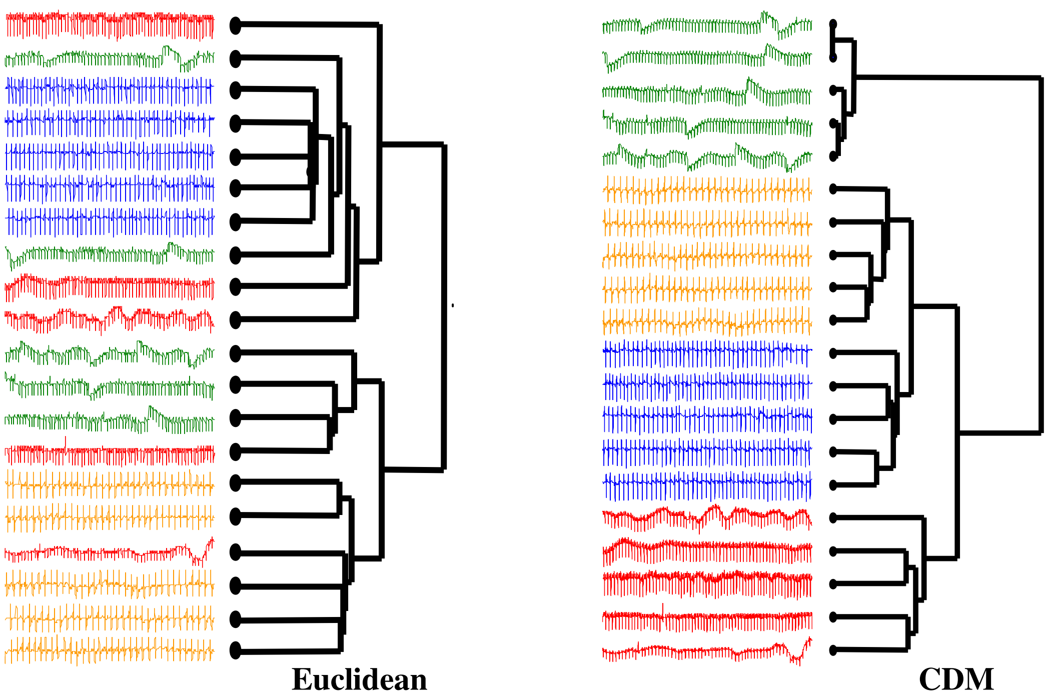
(**In general**) Li, Chen, Li, Ma, and Vitányi: (**For time series**) Keogh, Lonardi and Ratanamahatana

- features?
- distance measure/ learning algorithm?

Distance Measure

Co-Compressibility

$$CDM(x, y) = \frac{C(xy)}{C(x) + C(y)}$$

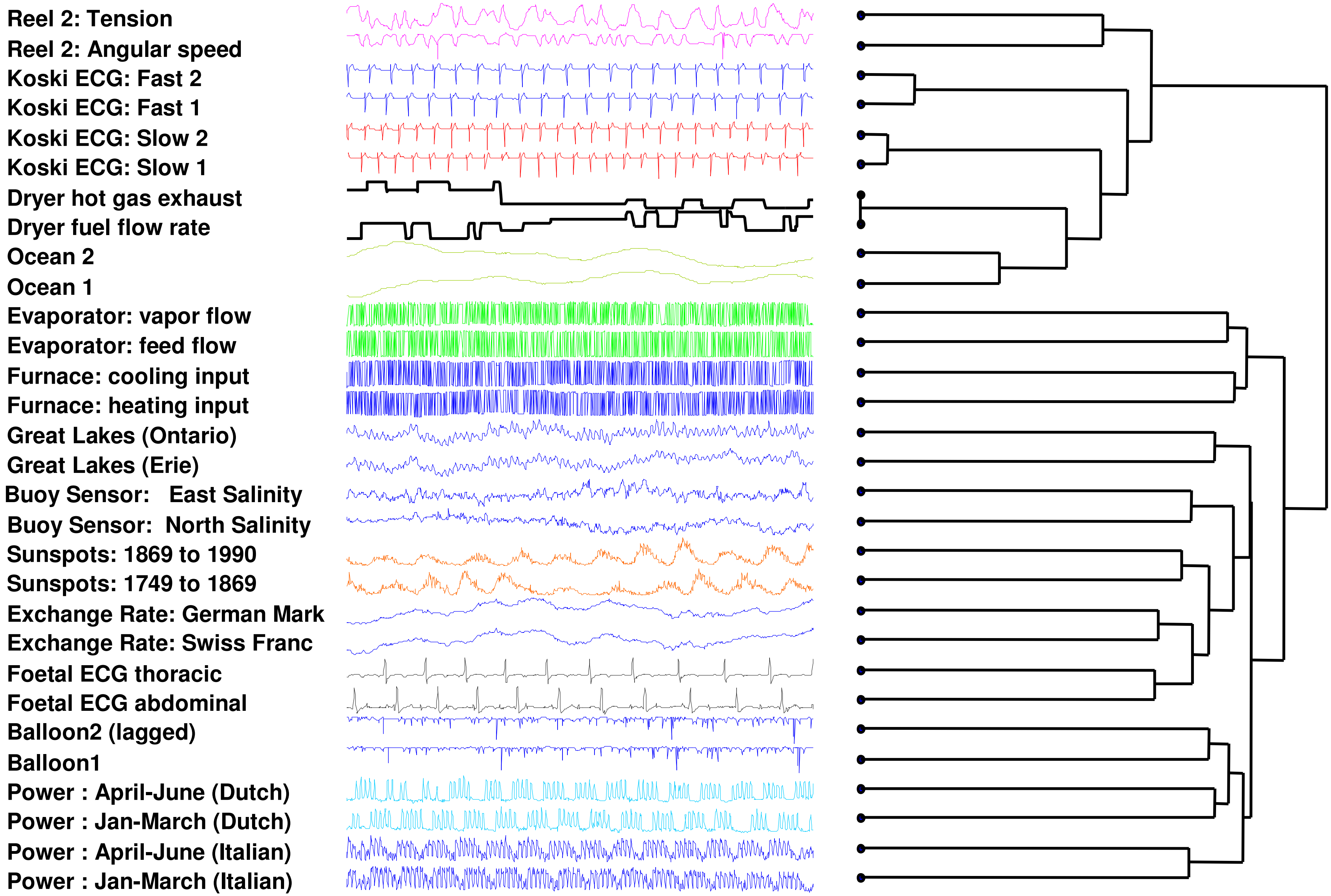


Features

Whatever structure the compression algorithm finds...

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

Compression Based Dissimilarity

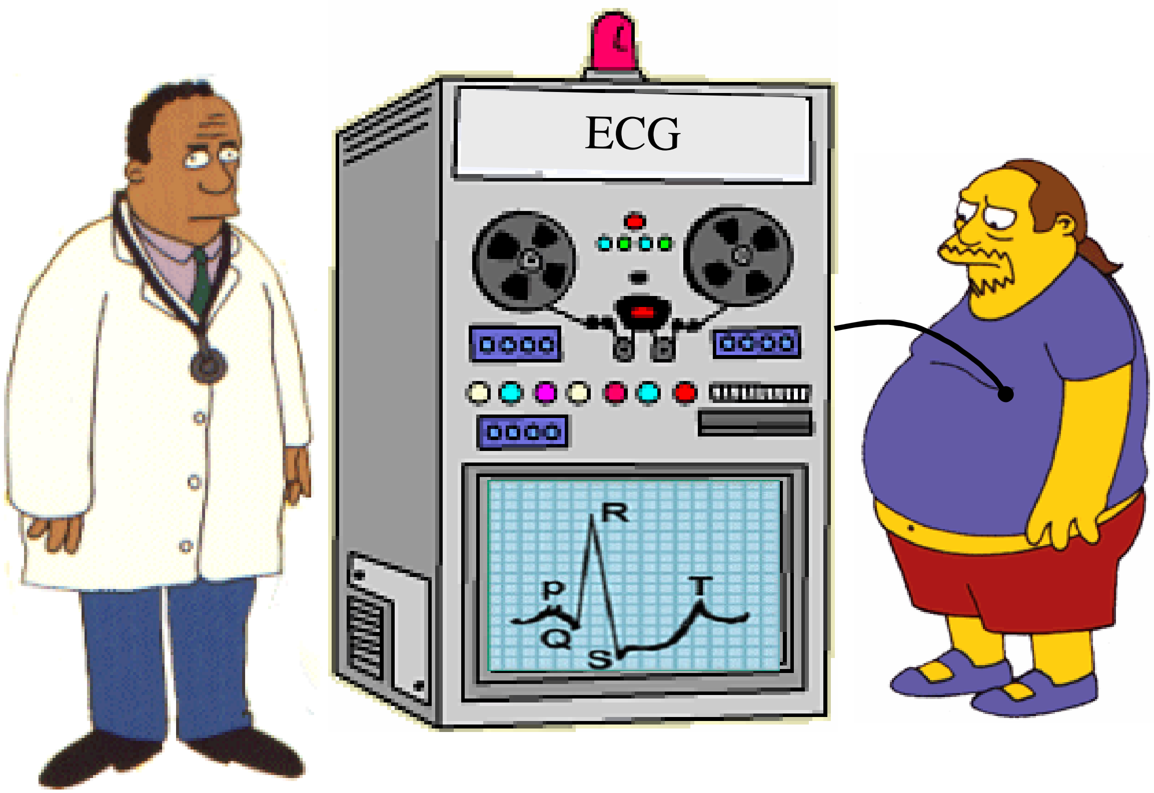


Summary of Time Series Similarity

- 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?**

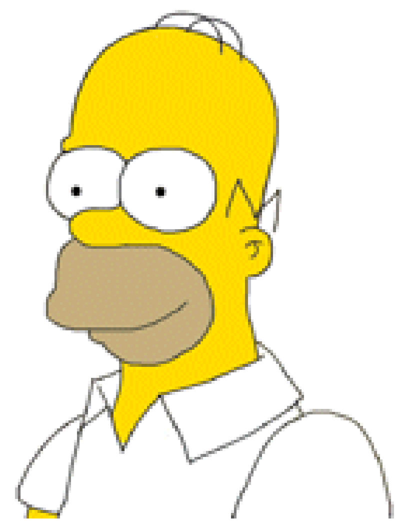
The Generic Data Mining Algorithm

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

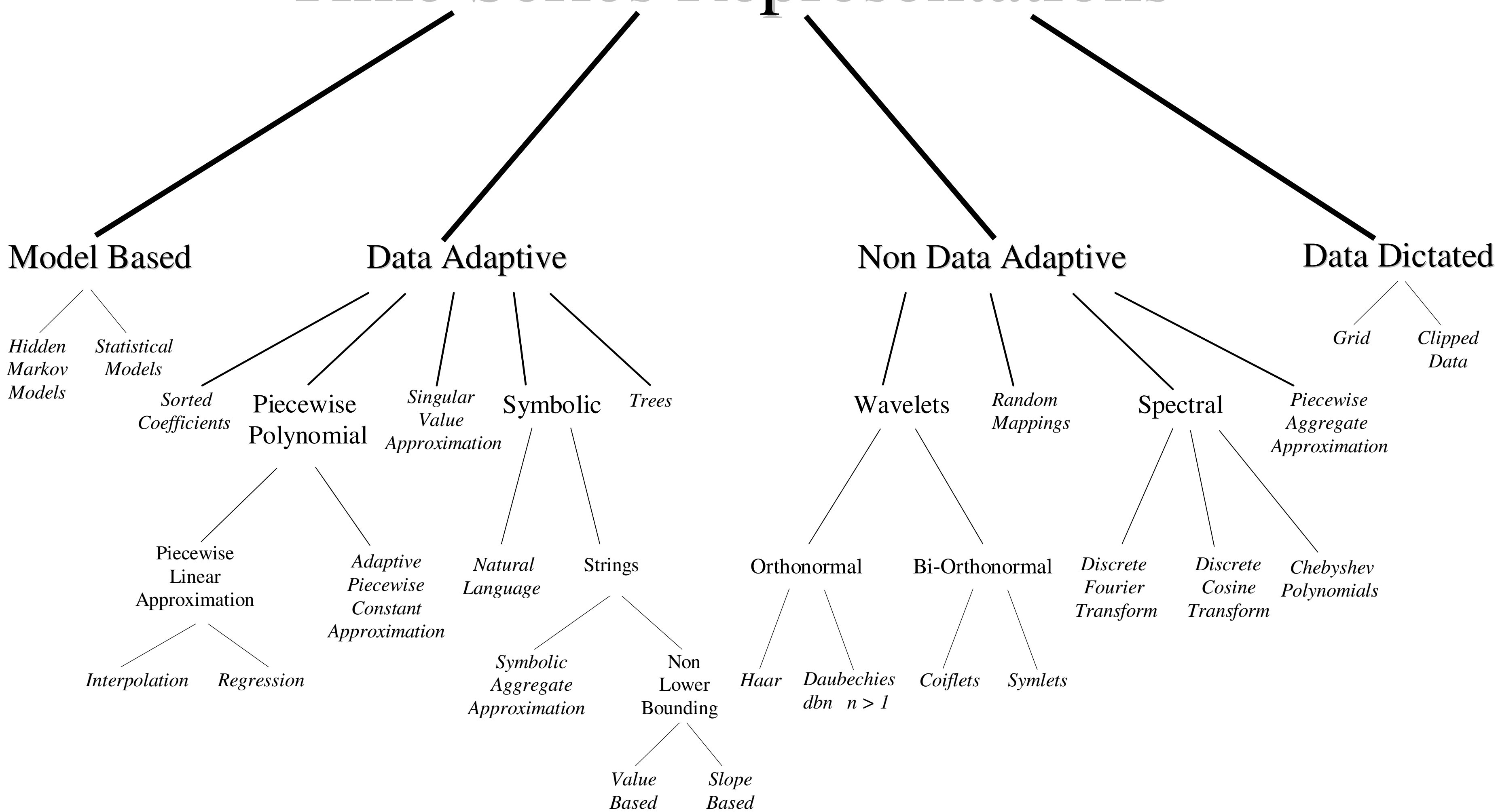
• Approximately solve the problem at hand in main memory

• 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

But which *approximation* should we use?



Time Series Representations



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**

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

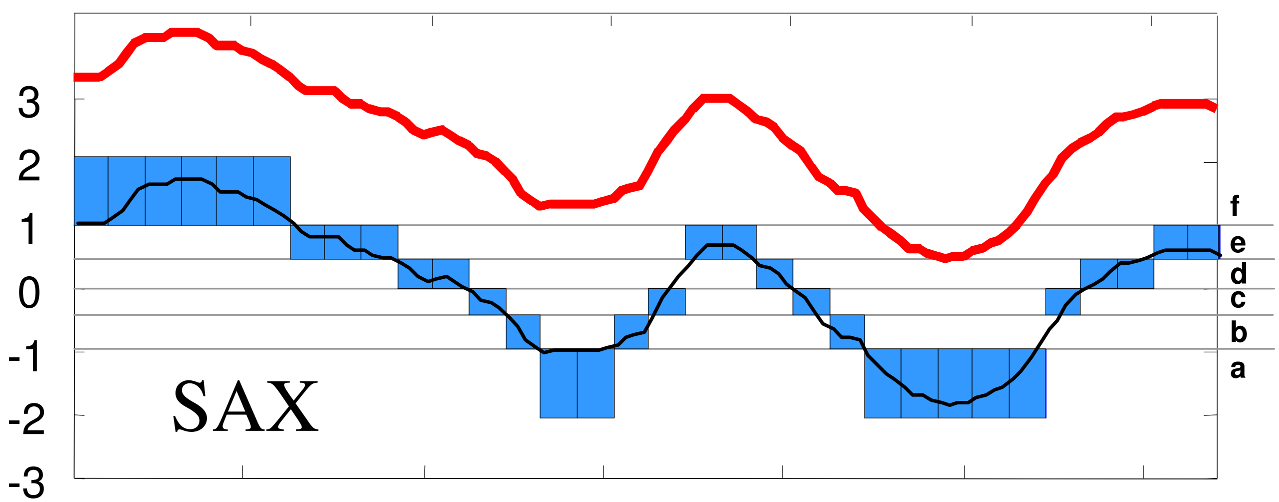
- If we had a lowerbounding, dimensionality reducing representation of time series, we could…

- 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

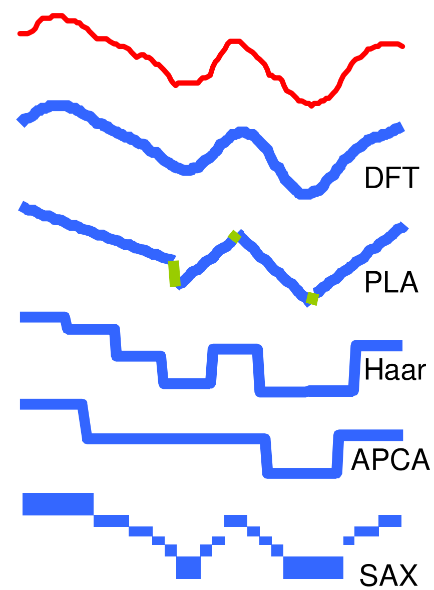
Exploiting Symbolic Representations of Time Series

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

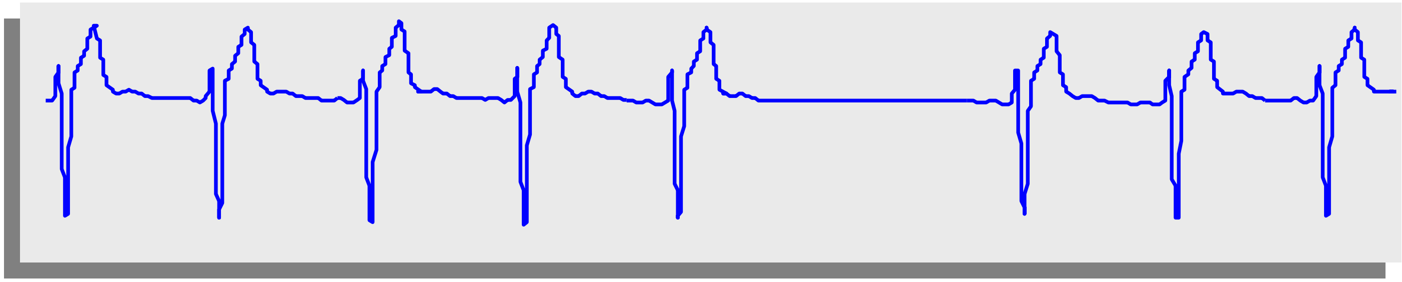


Anomaly (interestingness) detection

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

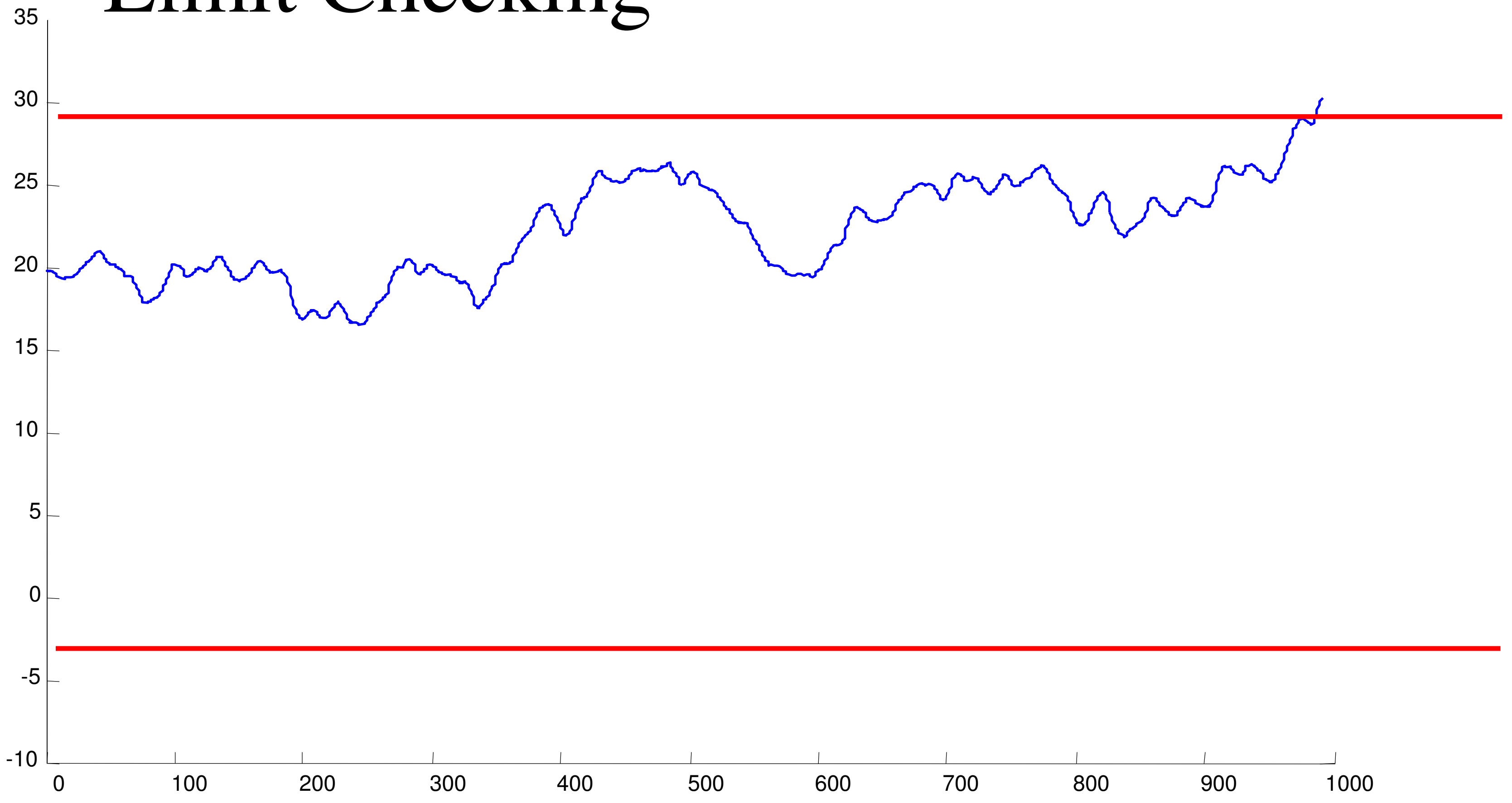
Note that we don't know in advance in what way the time series might be surprising

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



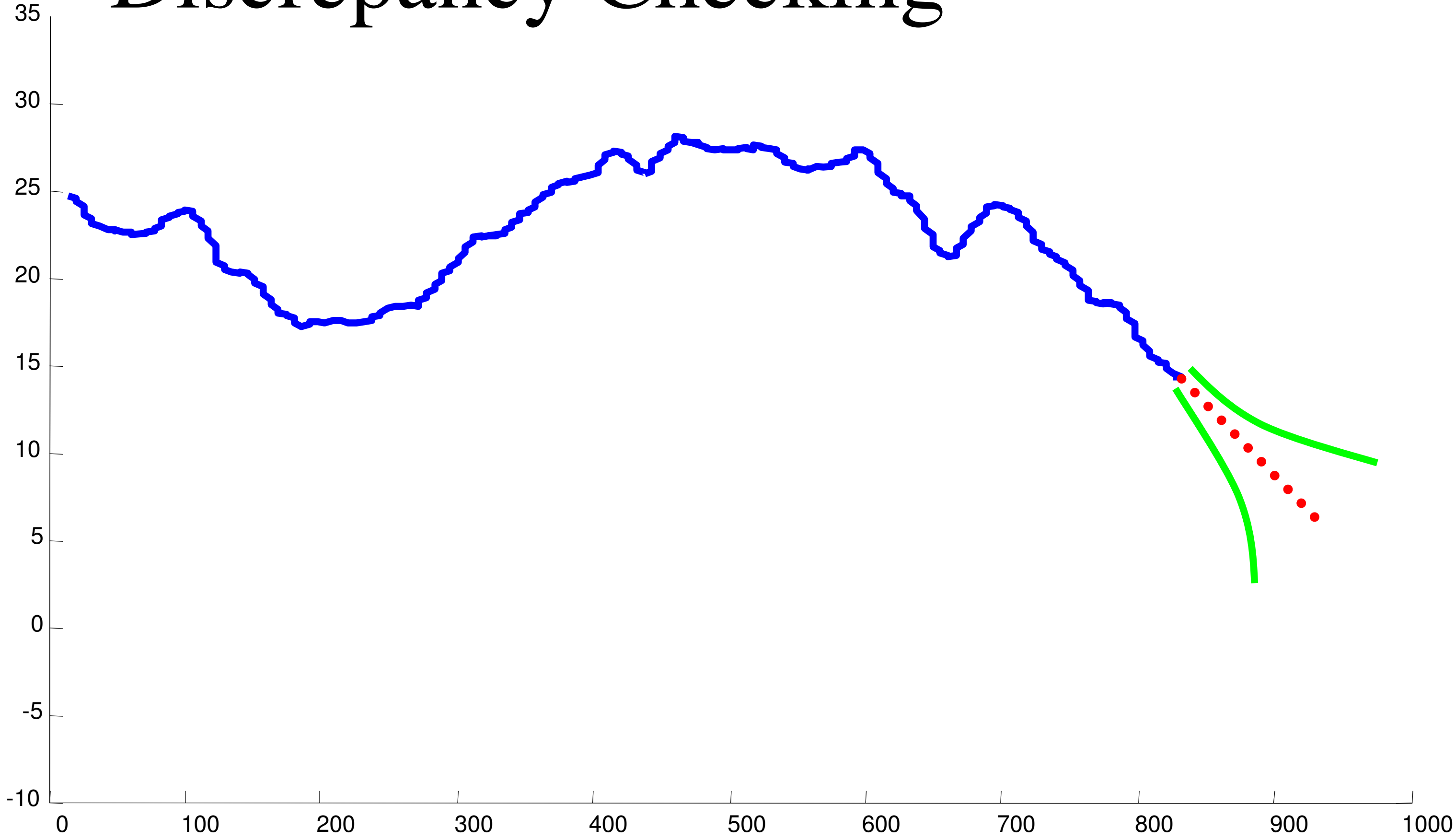
Simple Approaches I

Limit Checking

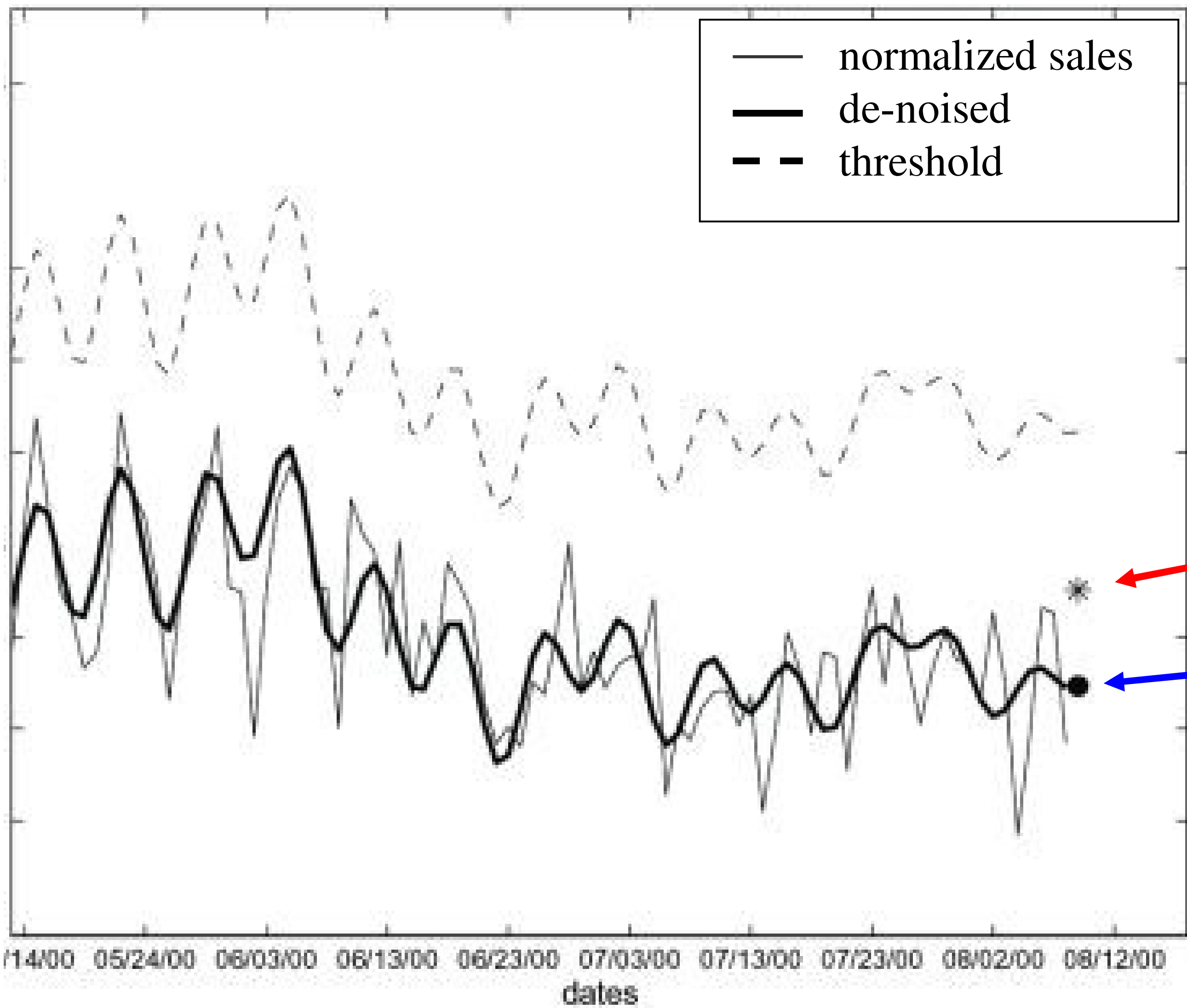


Simple Approaches II

Discrepancy Checking



Discrepancy Checking: Example



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

Goldenberg, Shmueli, Caruana, and Fienberg

Actual value

Predicted value

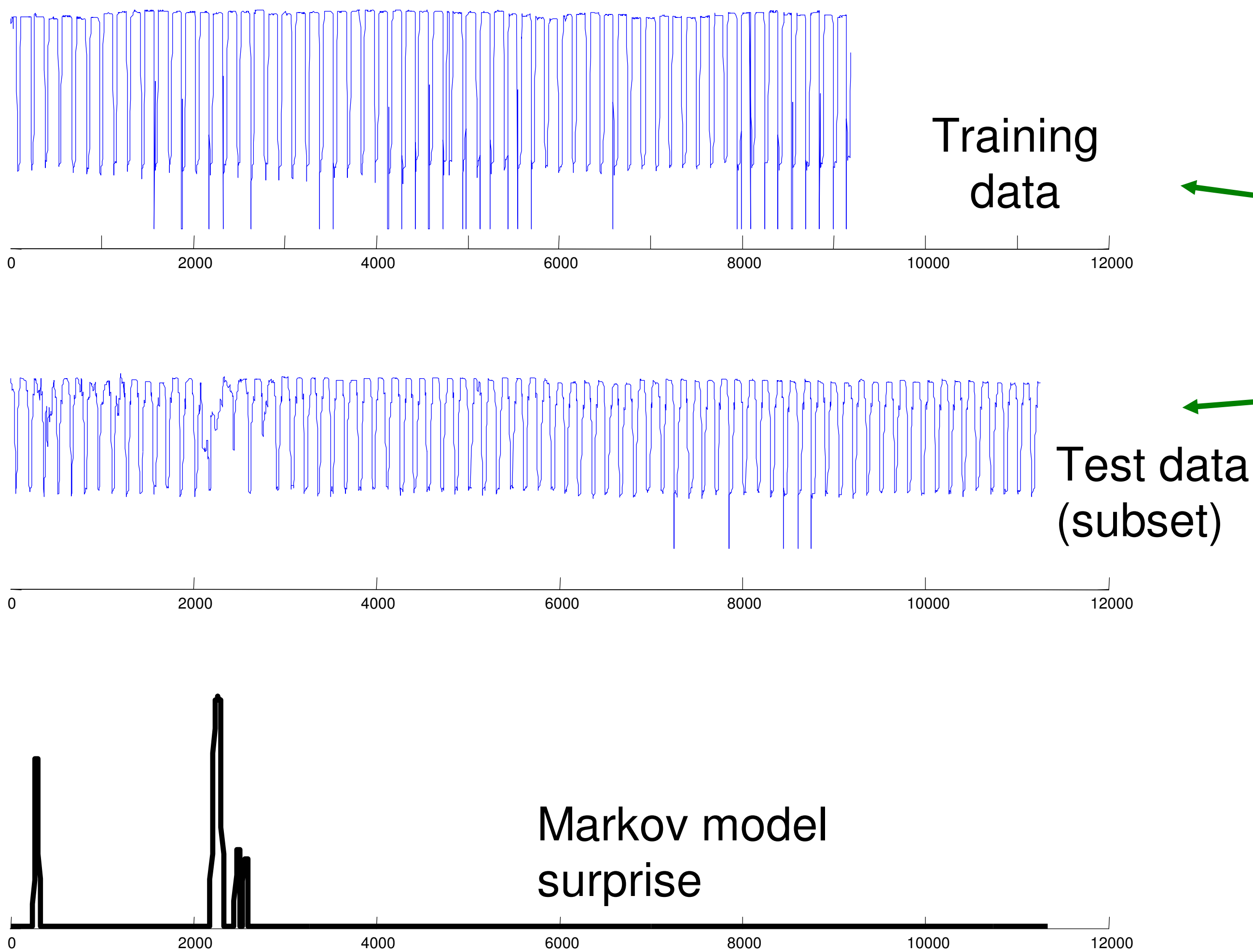
The actual value is greater than the predicted value, but still less than the threshold, so no alarm is sounded.

- 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.

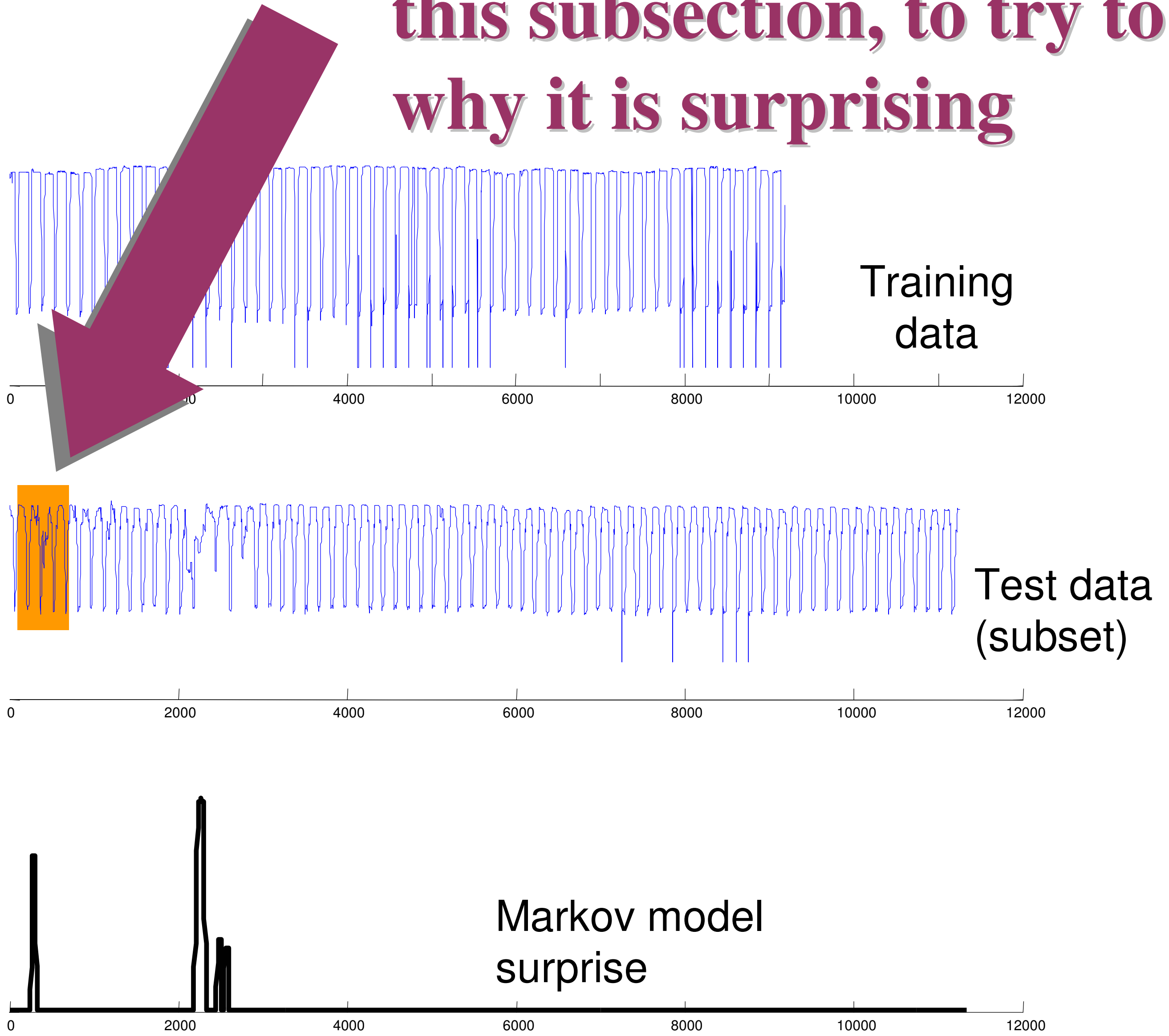
- Then, any future data that is not modeled well by the Markov model you annotate as *surprising*.

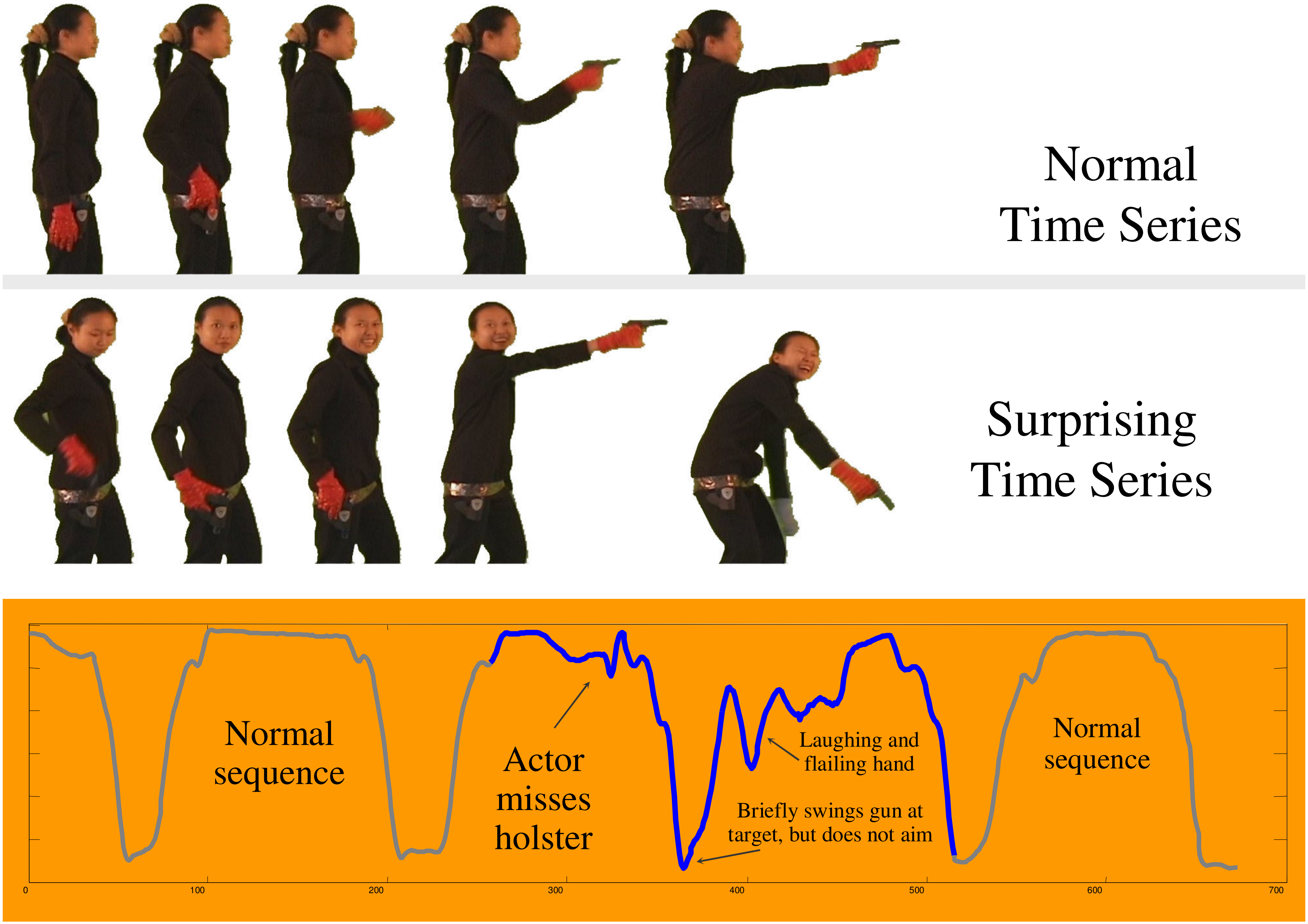
- Since we have just seen that we can convert time series to text (i.e SAX). Lets us quickly see if we can use Markov models to find surprises in time series…



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

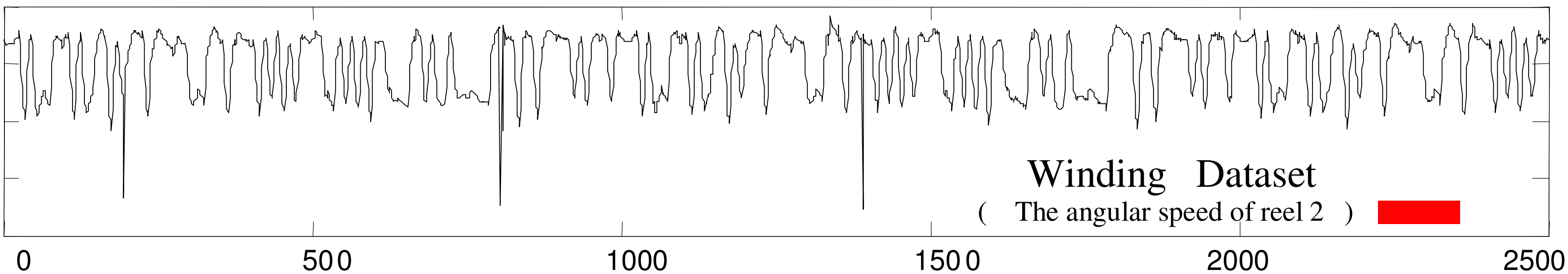
Anomaly (interestingness) detection

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

How can we find interesting patterns…

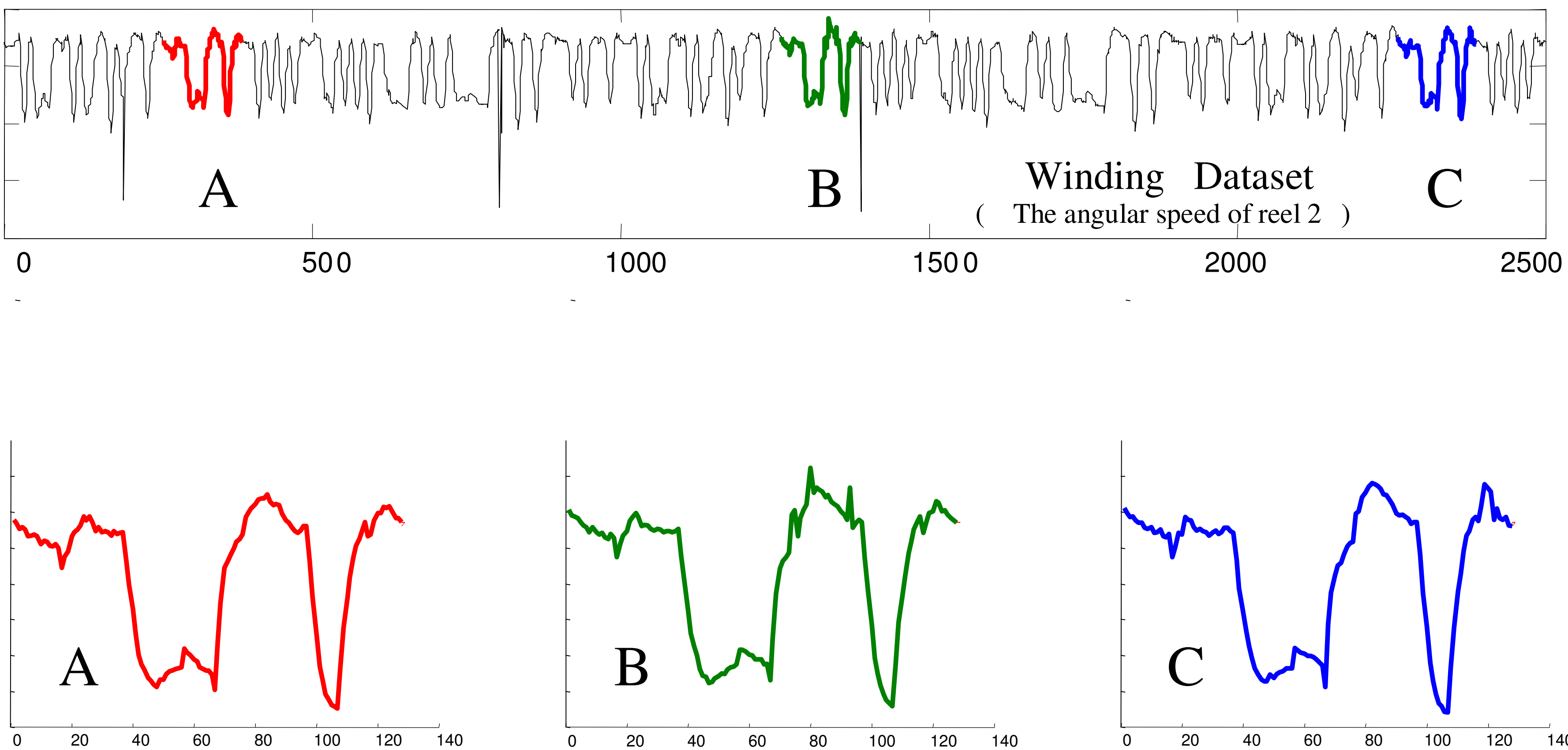
- Without (or with very few) false positives…
- In truly massive datasets...
- In the face of concept drift…
- With human input/feedback…
- With annotated data…

Time Series Motif Discovery
(finding repeated patterns)



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*.

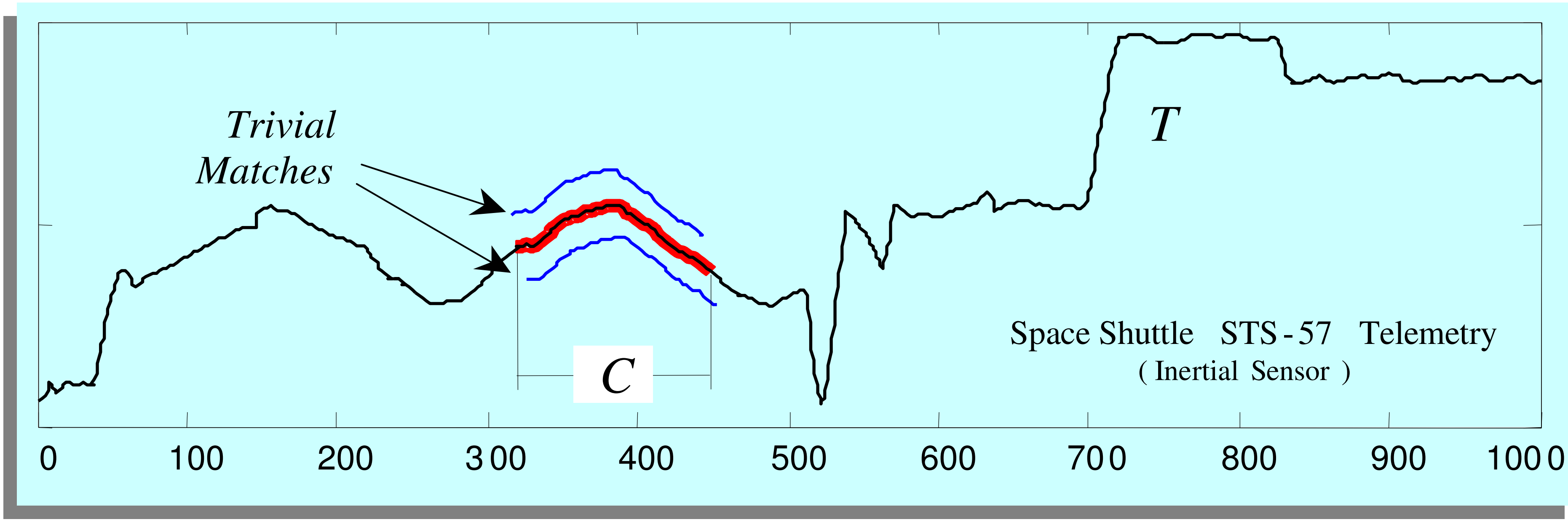
· Several time series **classification algorithms** work by constructing typical prototypes of each class. These prototypes may be considered motifs.

· 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)



Definition 1. *Match*: Given a positive real number R (called *range*) and a time series T containing a subsequence C beginning at position p and a subsequence M beginning at q, if $D(C, M) \leq R$, then M is called a *matching* subsequence of C.

Definition 2. *Trivial Match*: Given a time series T, containing a subsequence C beginning at position p and a matching subsequence M beginning at q, we say that M is a *trivial match* to C if either $p = q$ or there does not exist a subsequence M' beginning at q' such that $D(C, M') > R$, and either $q < q' < p$ or $p < q' < q$.

Definition 3. *K-Motif*(n,R): Given a time series T, a subsequence length n and a range R, the most significant motif in T (hereafter called the *1-Motif*(n,R)) is the subsequence C_1 that has highest count of non-trivial matches (ties are broken by choosing the motif whose matches have the lower variance). The K^{th} most significant motif in T (hereafter called the *K-Motif*(n,R)) is the subsequence C_K that has the highest count of non-trivial matches, and satisfies $D(C_K, C_i) > 2R$, for all $1 \leq i < K$.

OK, we can define motifs, but how do we find them?

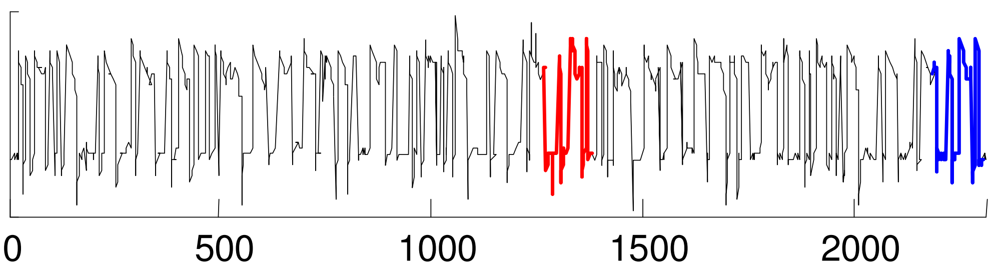
The obvious brute force search algorithm is just too slow…

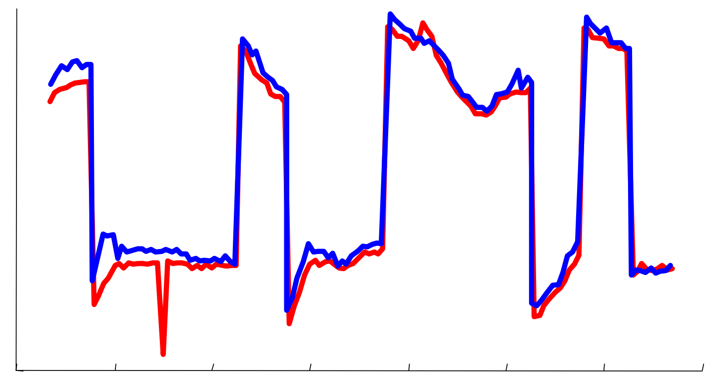
The most reference algorithm is based on a *hot* idea from bioinformatics, *random projection** and the fact that SAX allows use to lower bound discrete representations of time series.

*** J Buhler and M Tompa. *Finding motifs using random projections*. In RECOMB'01. 2001.**

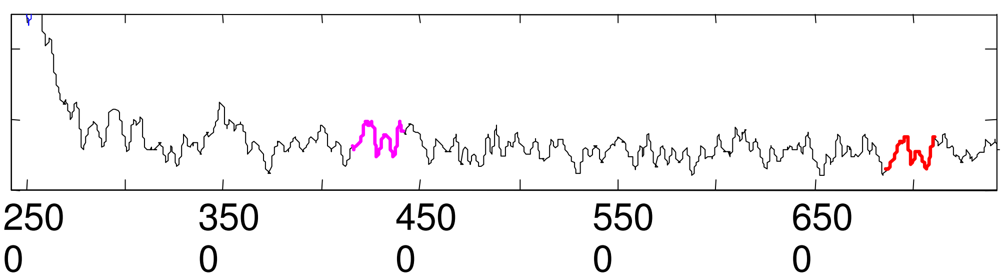
Some Examples of Real Motifs

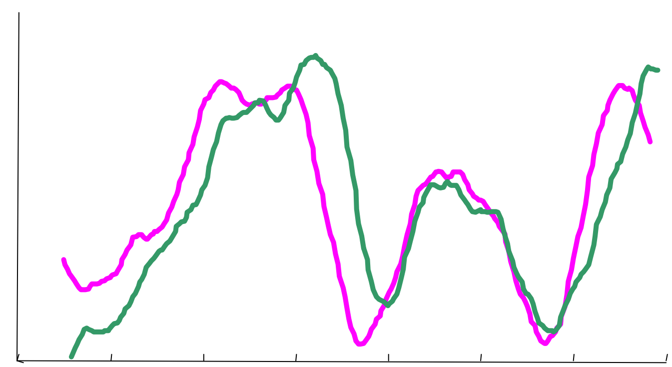
Motor 1 (DC Current)





Astrophysics (Photon Count)

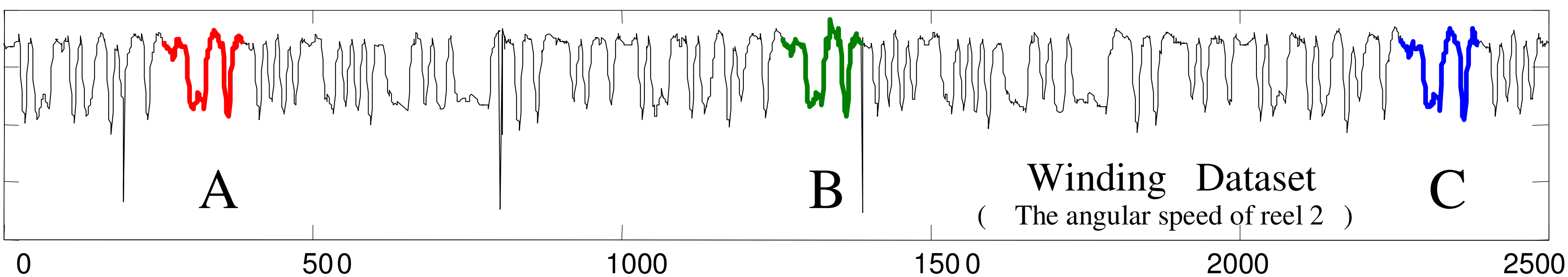




Motifs Discovery Challenges

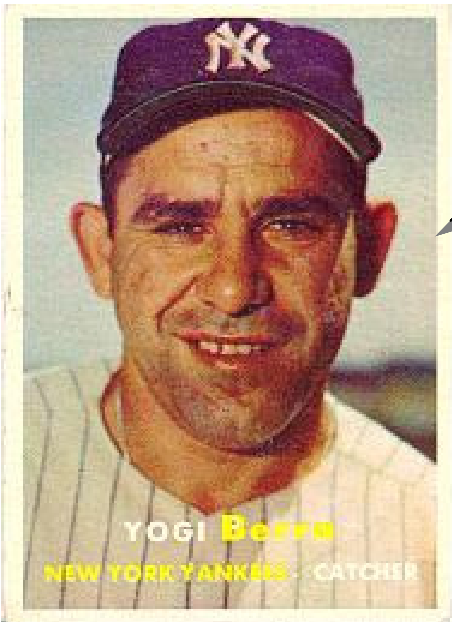
How can we find motifs…

- 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)