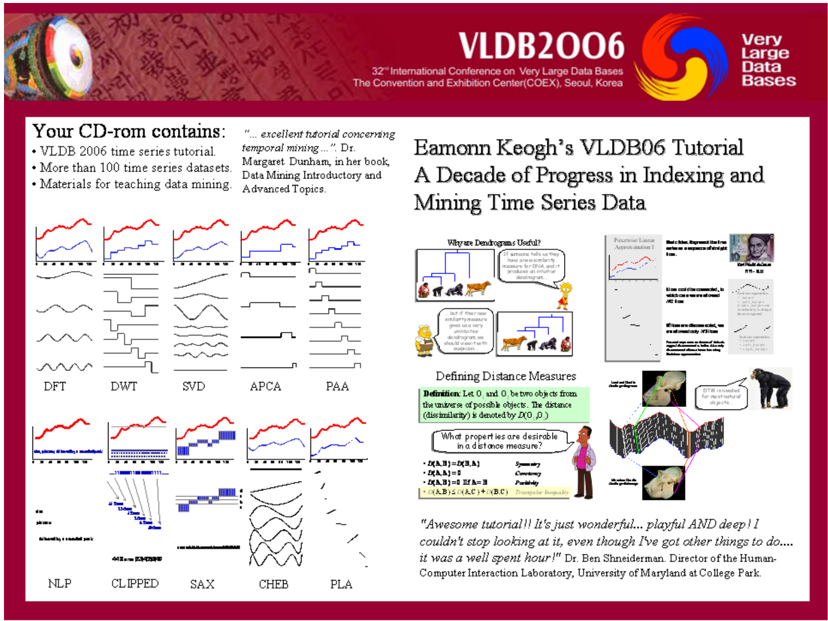
Introduction to Time Series Mining

Slides edited from Keogh Eamonn's tutorial:



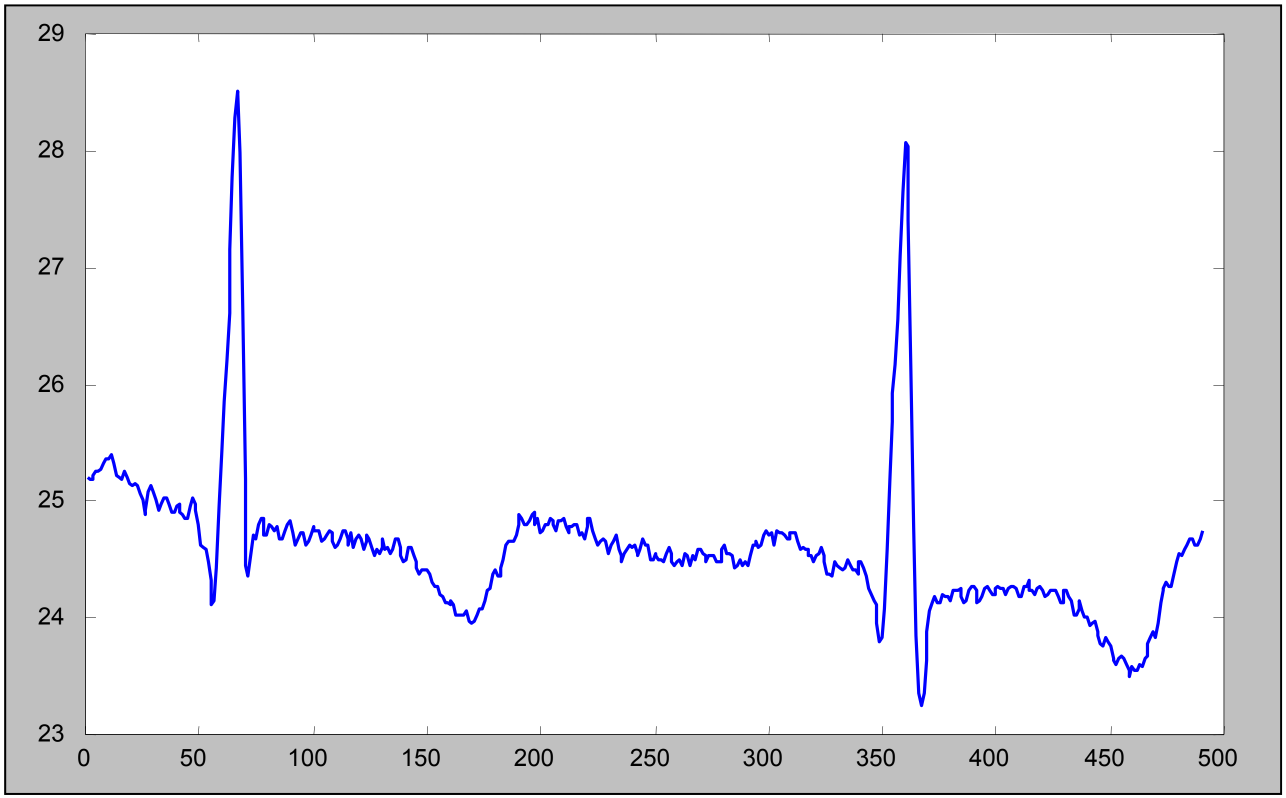
25.1750
25.2250
25.2500
25.2500
25.2750
25.3250
25.3500
25.3500
25.4000
25.4000
25.3250
25.2250
25.2000
25.1750

••

••
24.6250
24.6750
24.6750
24.6250
24.6250
24.6250
24.6750
24.7500

What are Time Series?

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

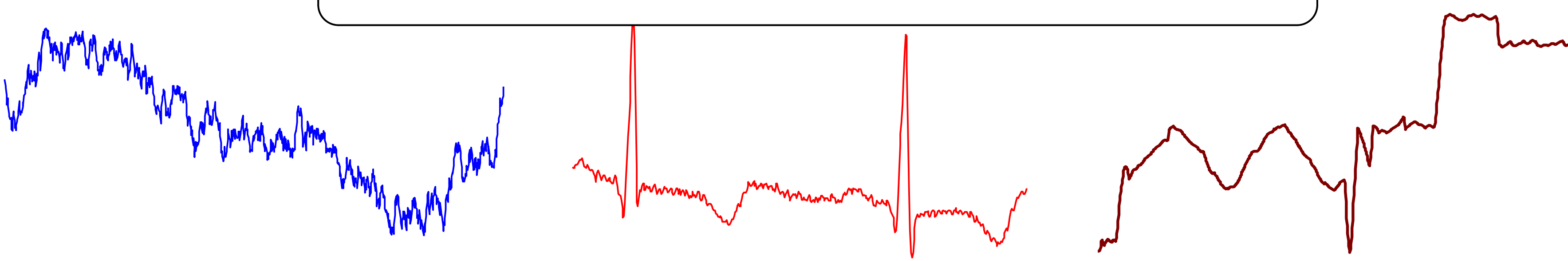


Time Series are Ubiquitous! I

People measure things...

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

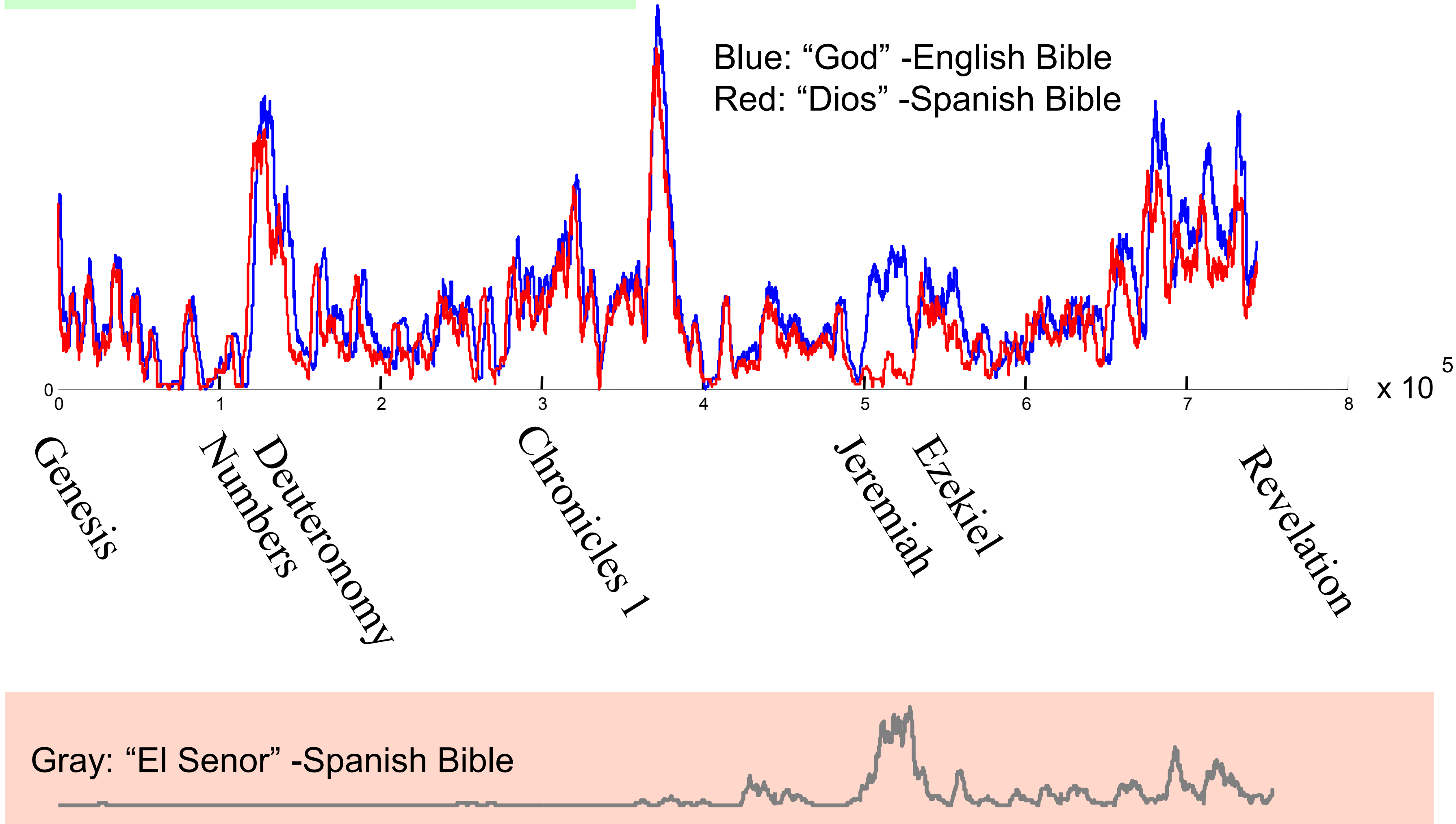
...and things change over time...



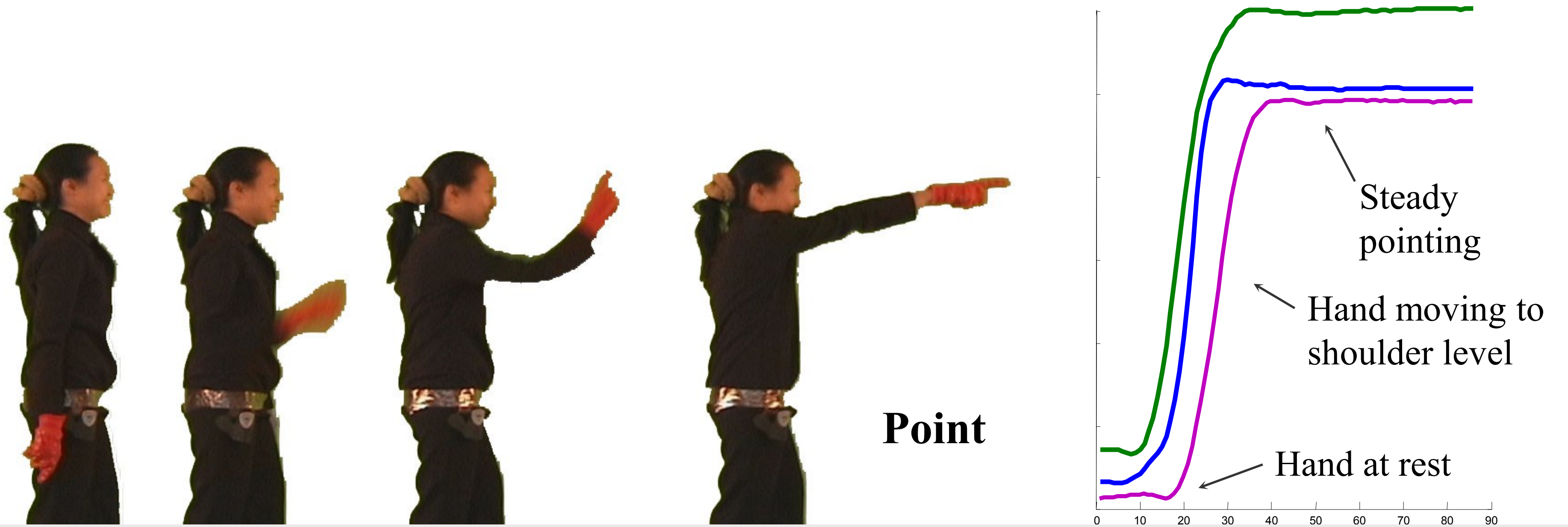
Thus time series occur in virtually every medical, scientific and businesses domain

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…



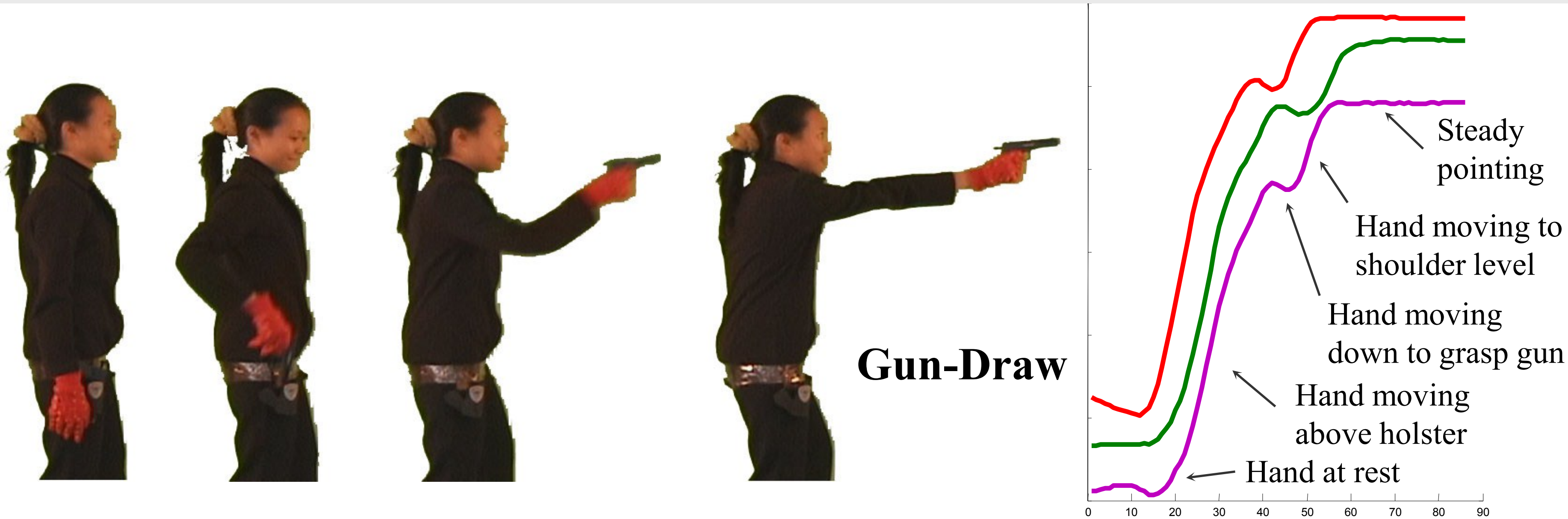
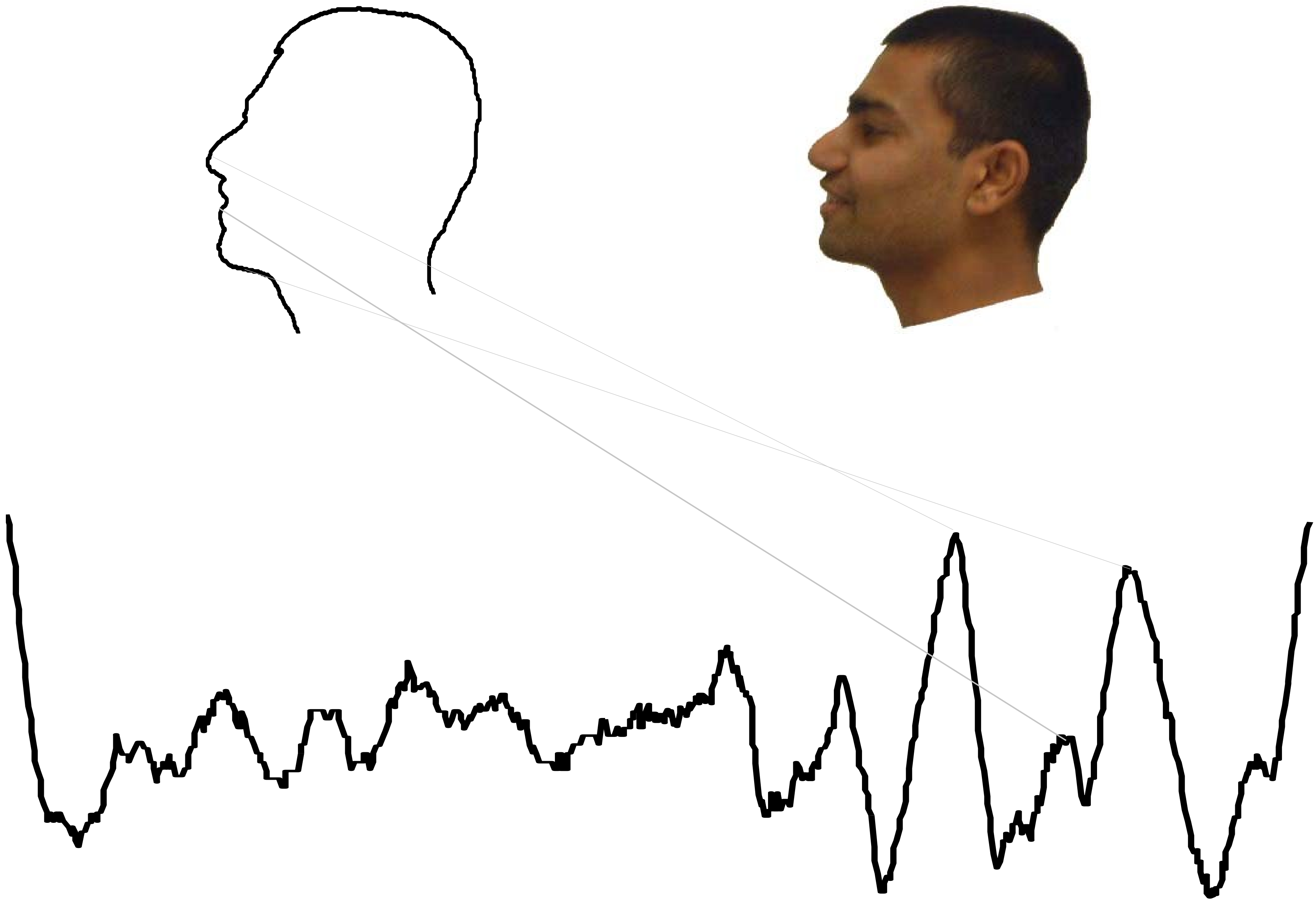
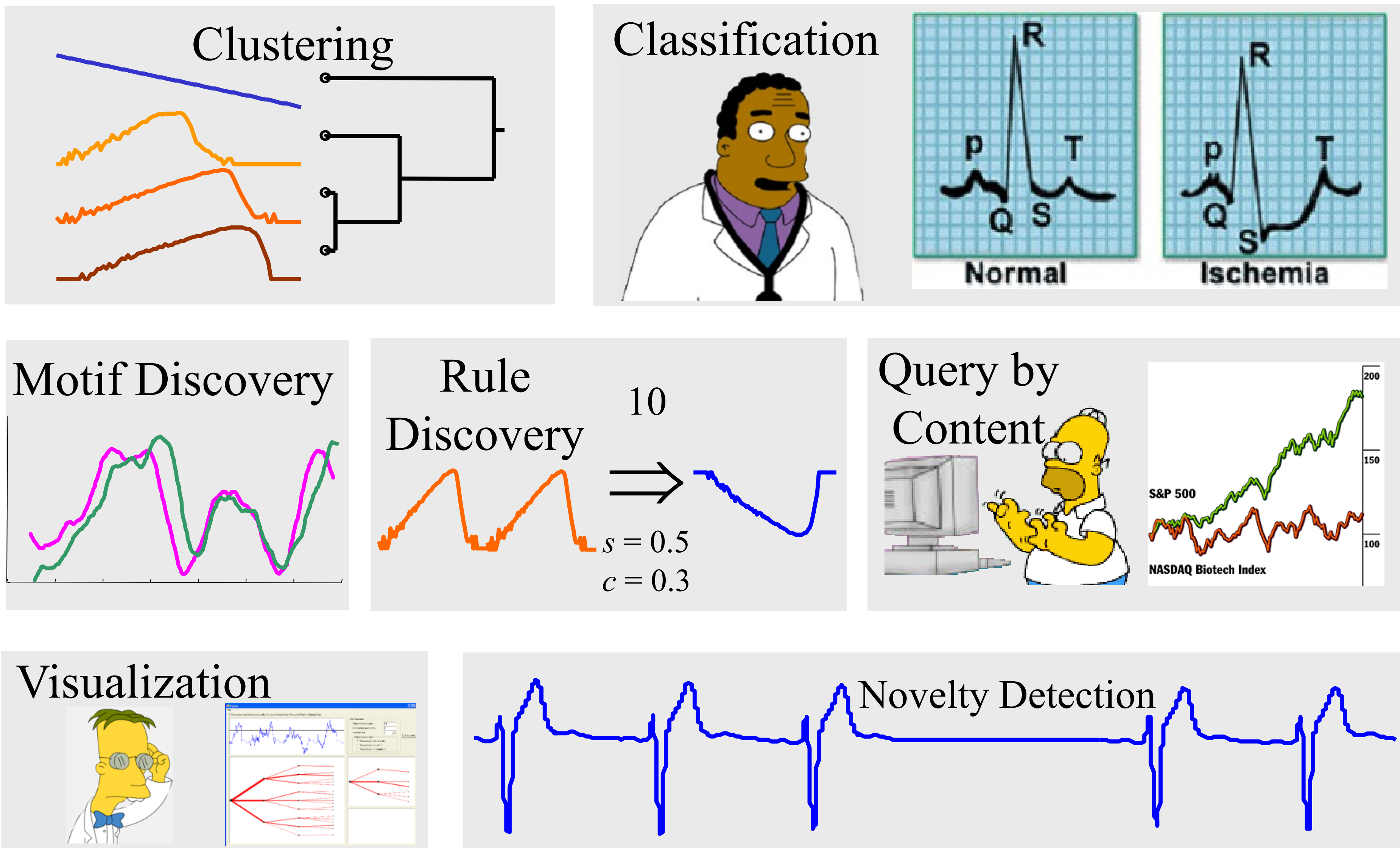


Image data, may best be thought of as time series…



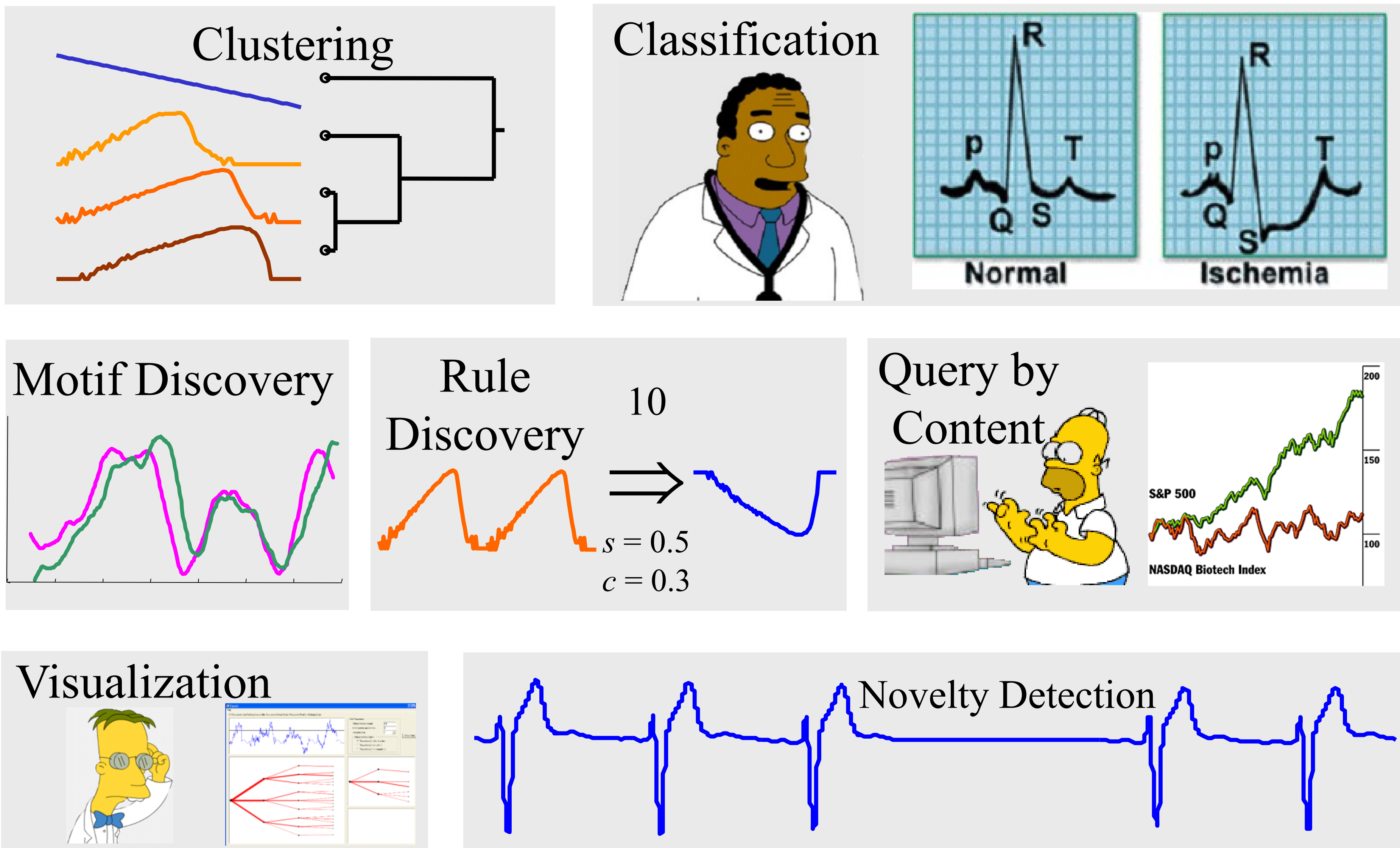
What do we want to do with the time series data?



Time series analysis tasks

- Similarity-based tasks
  - Standard clustering, classification (KNN), etc.
- Outlier detection
- Frequent patterns (Motifs)
- Prediction

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



What is Similarity?

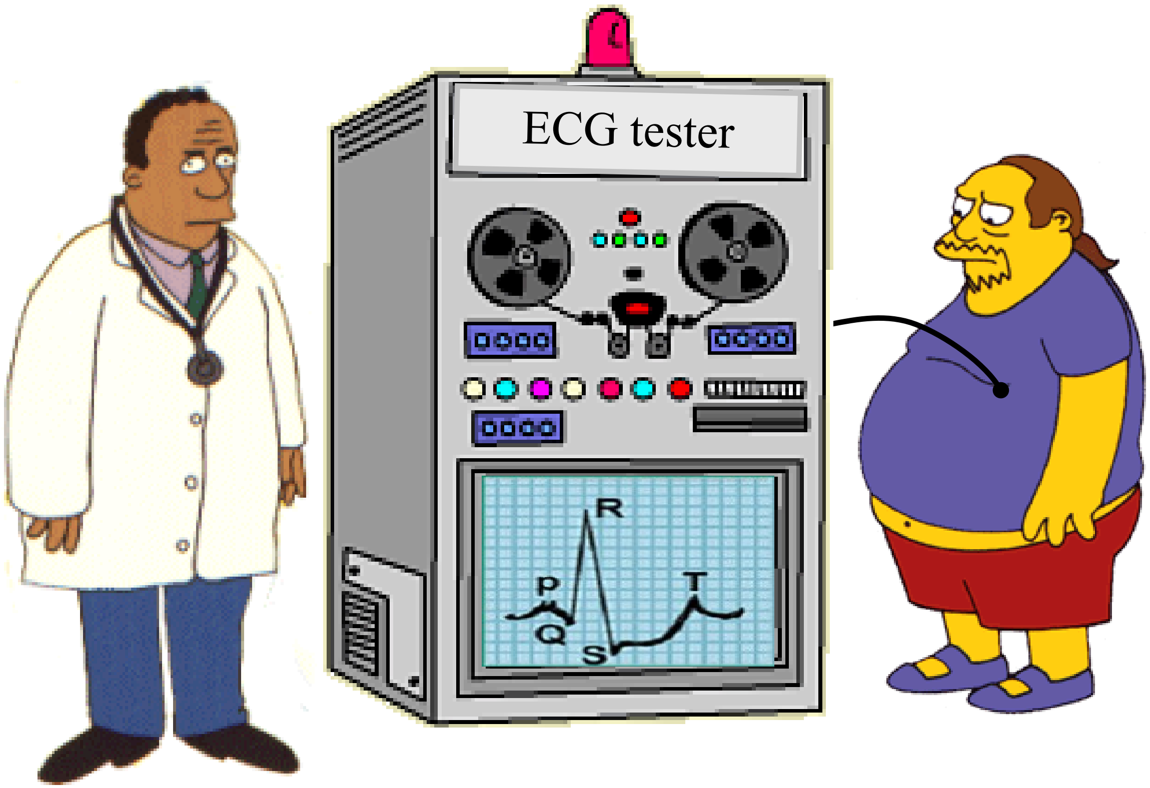
The quality or state of being similar; likeness; resemblance; as, a similarity of features. **Webster's Dictionary**

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.

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…

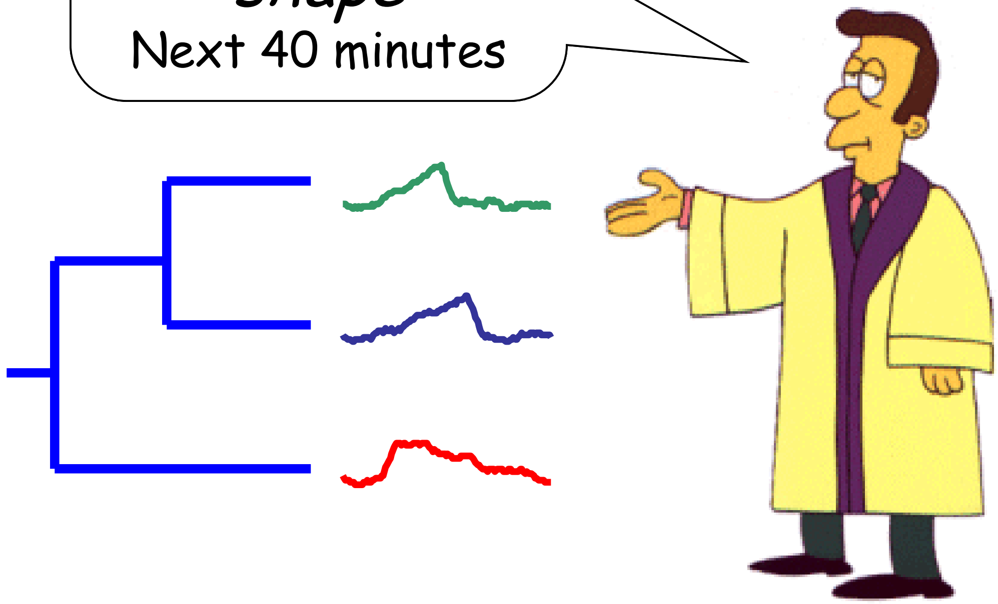
You doctor wants to search a database to find **similar** ECGs, in the hope that they will offer clues about your condition...

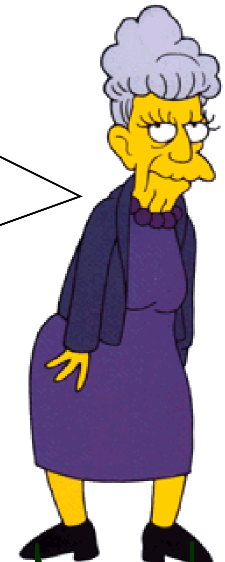
Two questions:

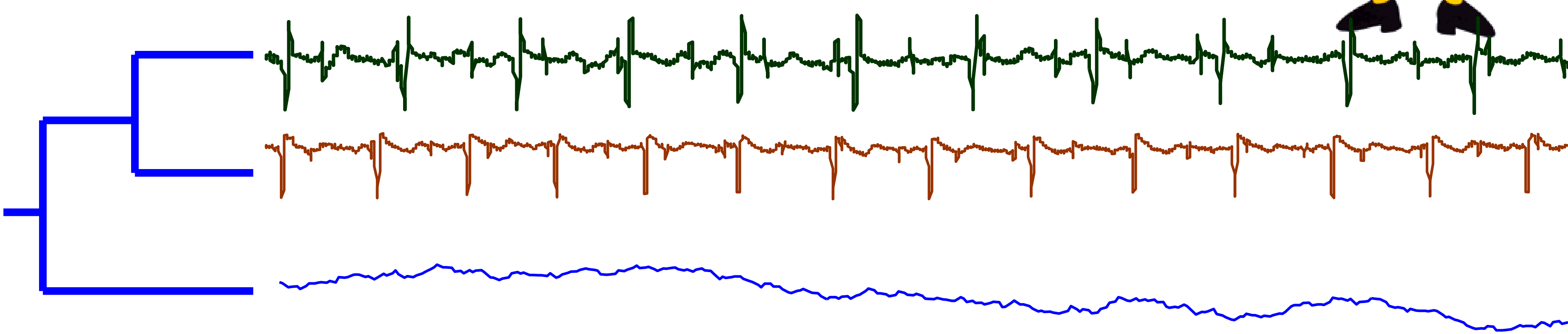
- **How do we define similar?**
- How do we search quickly?

Two Kinds of Similarity

time series





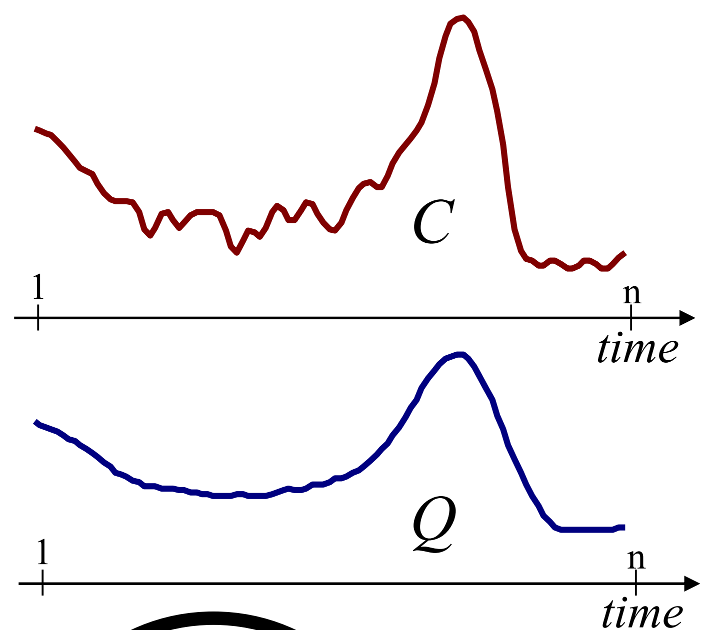


Euclidean Distance Metric

Given two time series:

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

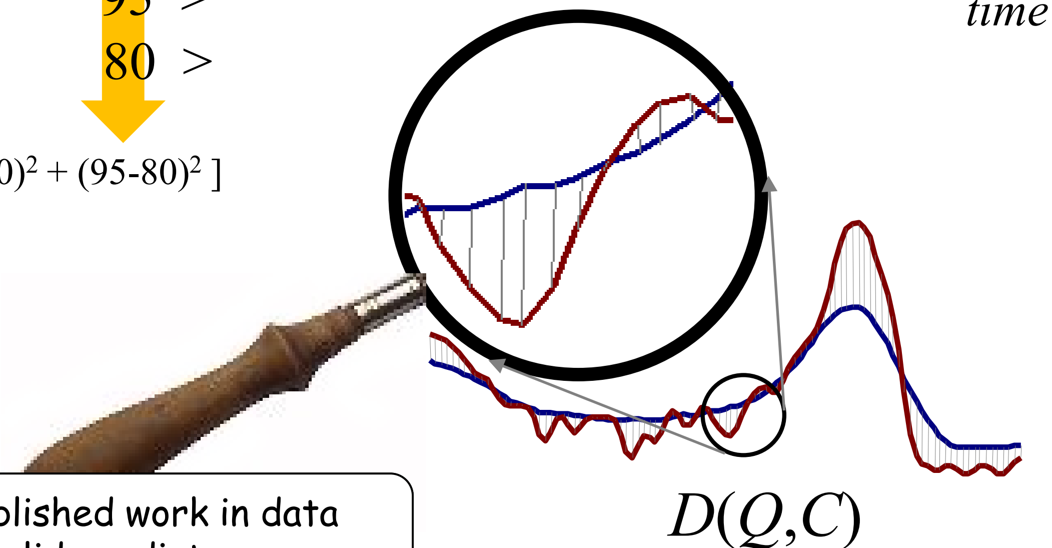
$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

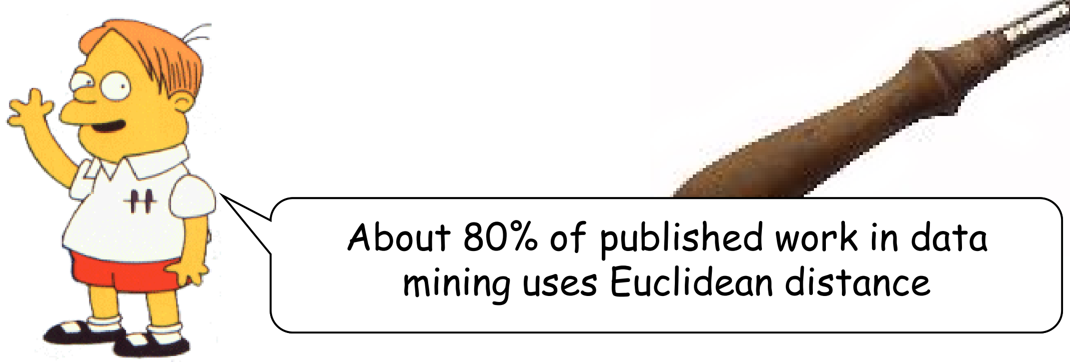


Same meaning as in transaction data:

- schema: <age, height, income, tenure>
- T1 = < 56, 176, 110, 95 >
- T2 = < 36, 126, 180, 80 >

D(T1,T2) = sqrt [$(56-36)^2 + (176-126)^2 + (110-180)^2 + (95-80)^2$]

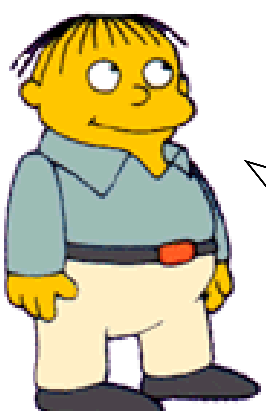




Preprocessing the data before distance calculations

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

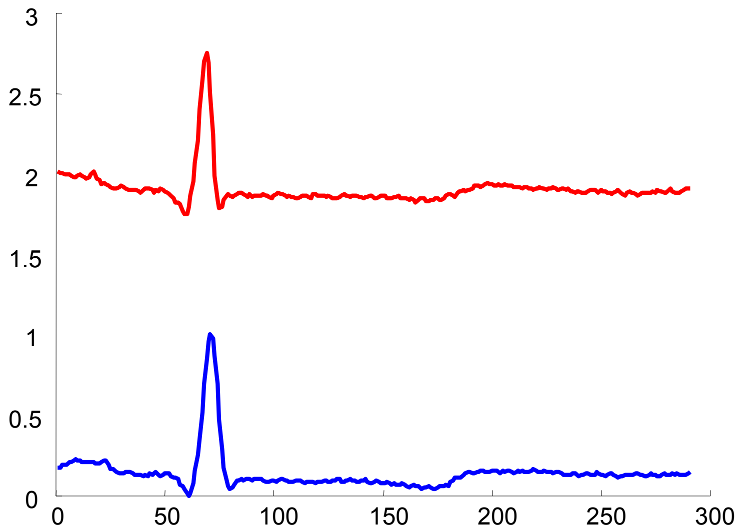
Euclidean distance is very sensitive to some "distortions" in the data.
For most problems these distortions are not meaningful => should remove them

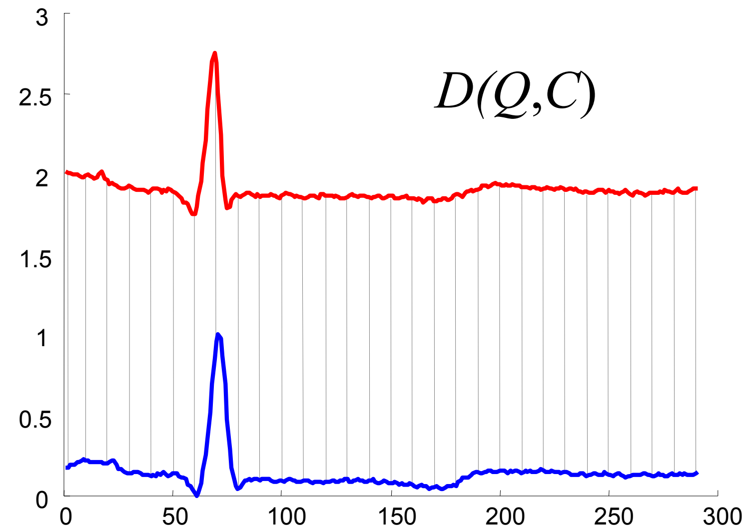


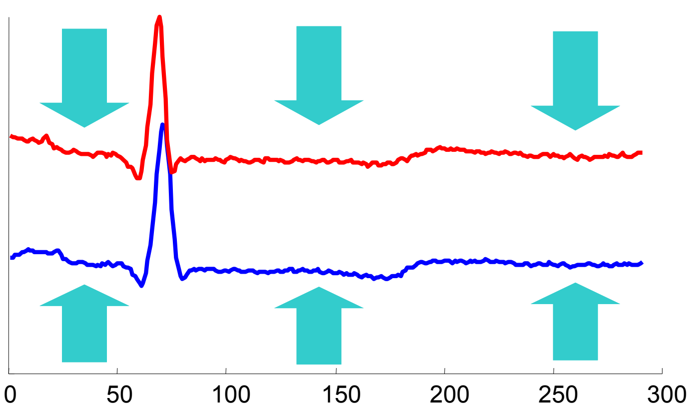
In the next few slides we will discuss the 4 most common distortions, and how to remove them

- Offset Translation
- Amplitude Scaling
- Linear Trend
- Noise

Transformation I: Offset Translation



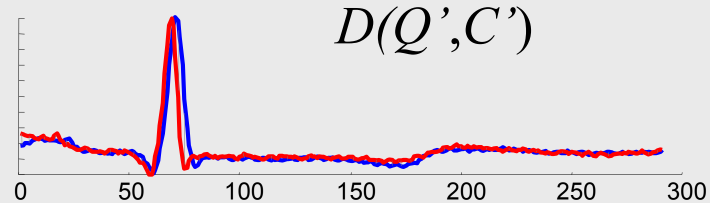




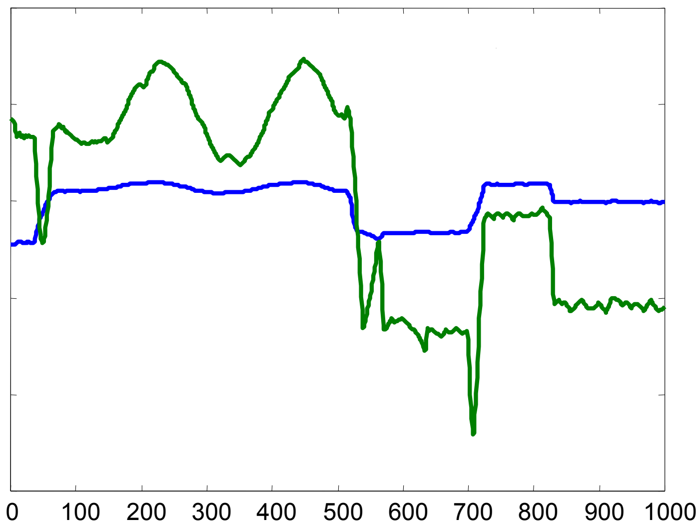
$Q' = Q$ - mean(Q)

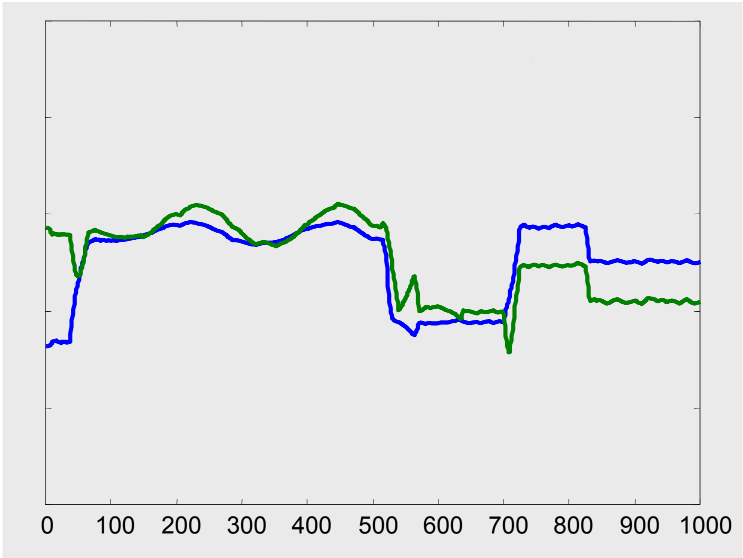
$C' = C$ - mean(C)

$D(Q',C')$



Transformation II: Amplitude Scaling





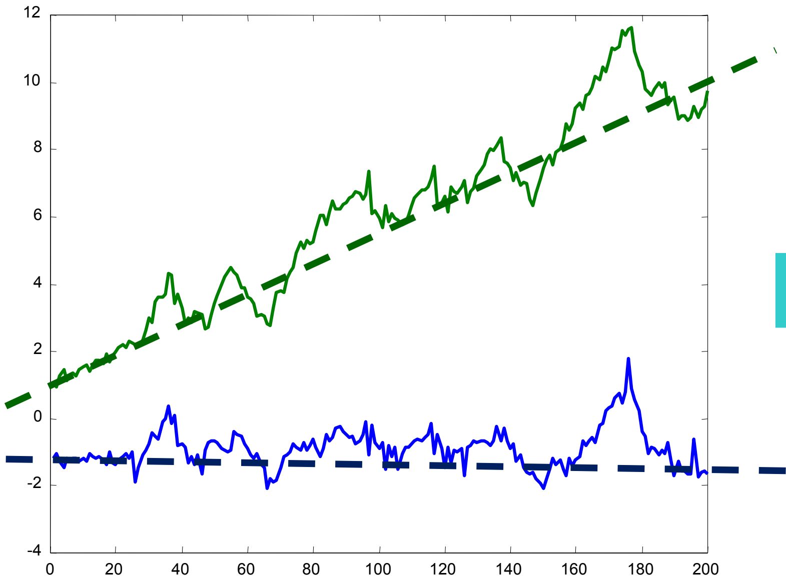
Z-score of Q

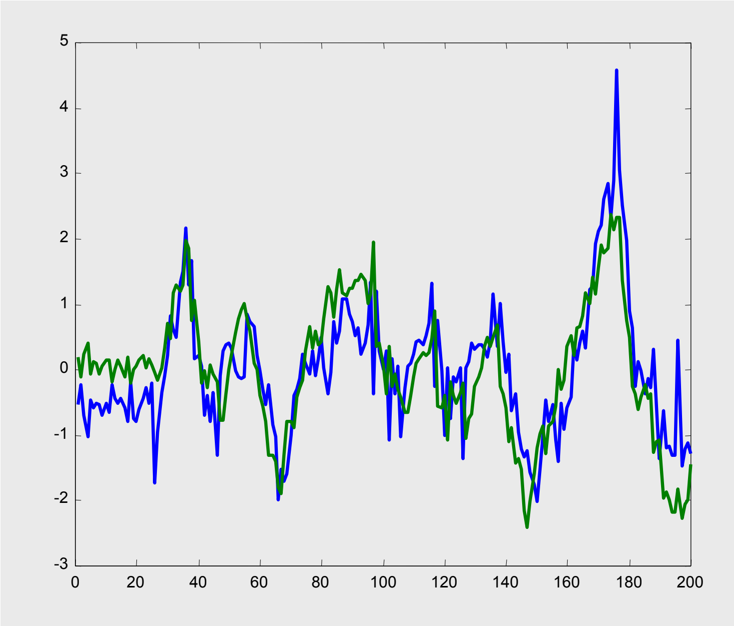
$Q'' = (Q - \mathrm{mean}(Q)) \,/\, \mathbf{std}(\boldsymbol{Q})$

$C'' = (C - \mathrm{mean}(C)) \,/\, \mathbf{std}(\boldsymbol{C})$

$D(Q'',C'')$

Transformation III: Linear Trend





Removing linear trend:

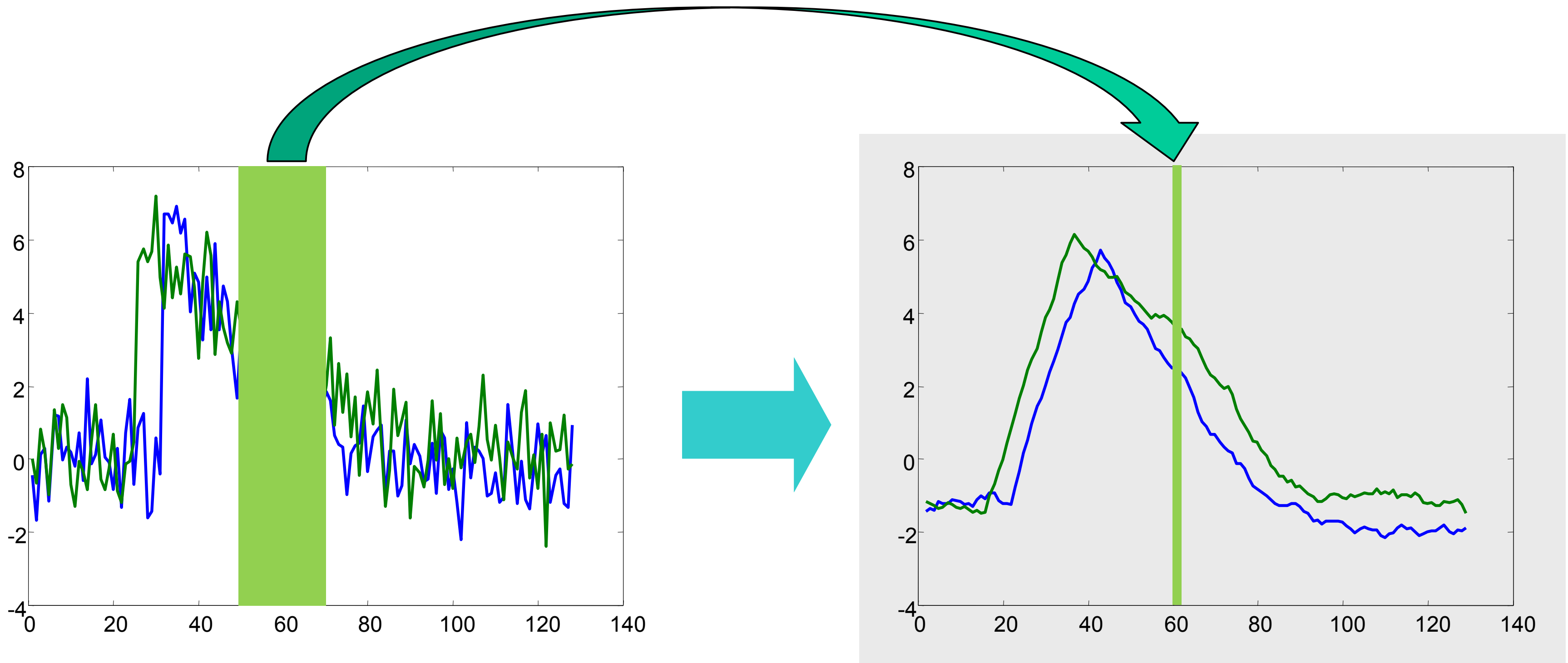
- fit the best fitting straight line to the time series, then
- subtract that line from the time series.

Removed **linear trend**

Removed offset translation

Removed amplitude scaling

Transformation IV: Noise



The intuition behind removing noise is...

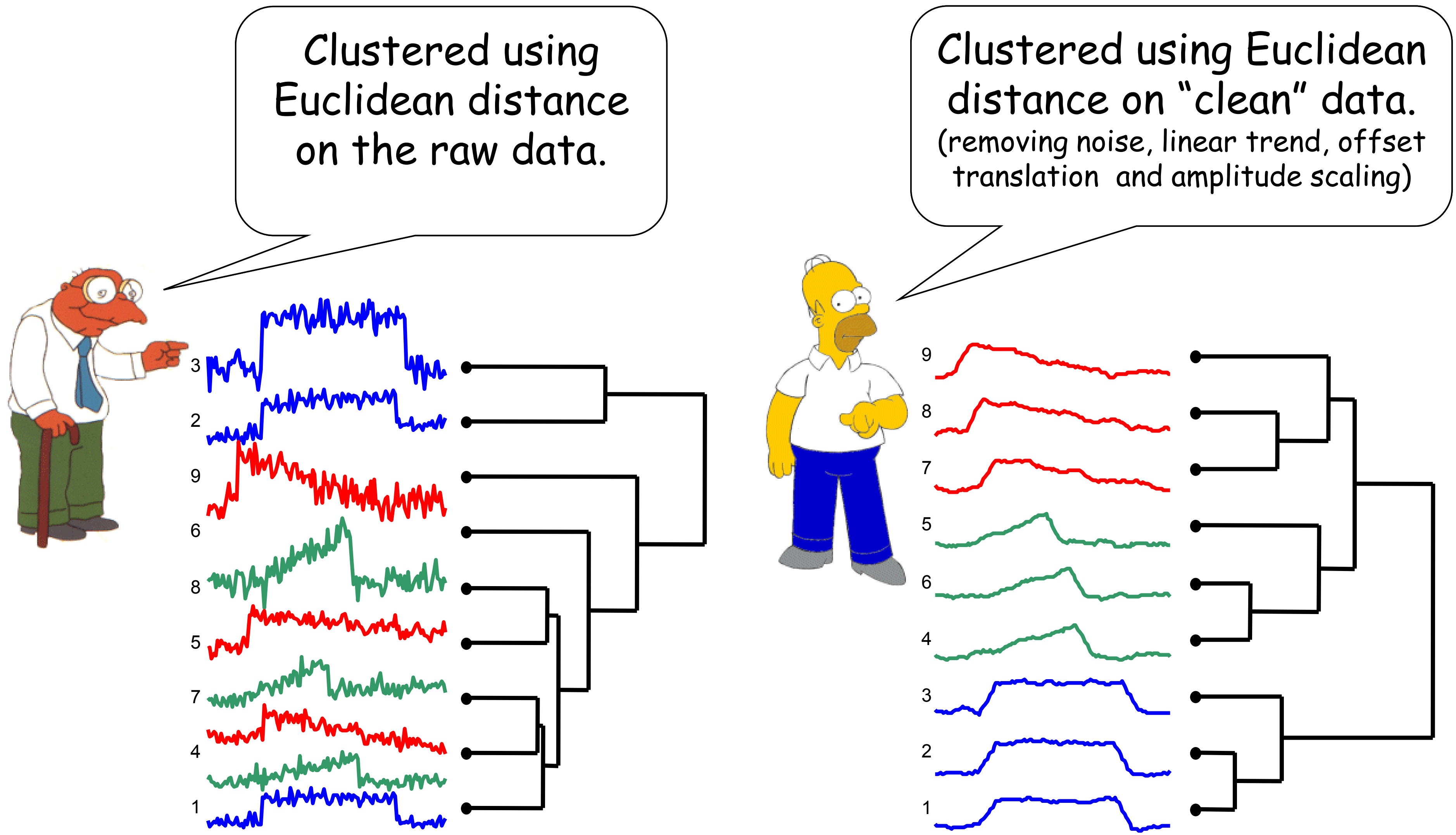
Average each datapoints value with its neighbors.

Q' = smooth(*Q*)

C' = smooth(*C*)

D(*Q'*,*C'*)

A Quick Experiment to Demonstrate the Utility of Preprocessing the Data



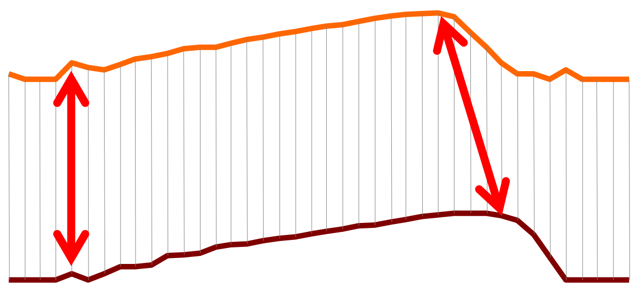
Summary of Preprocessing

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

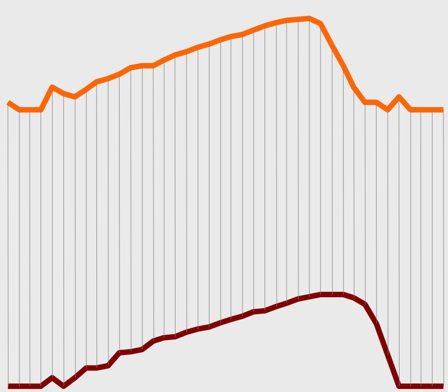
Of course, sometimes the distortions are the most interesting thing about the data, the above is only a general rule

Beyond Euclidean: Dynamic Time Warping

- Sometimes two time series that are conceptually equivalent evolve at different speeds, at least in some moments

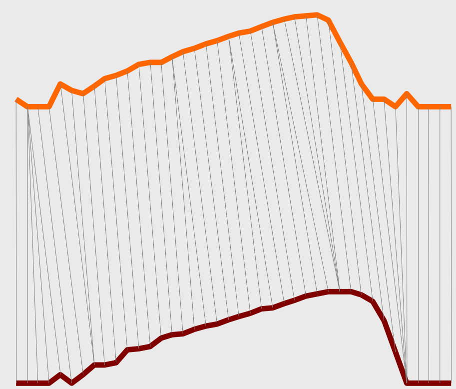


E.g. correspondence of peaks in two similar time series



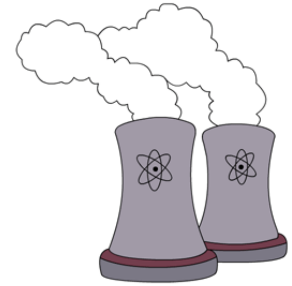
Fixed Time Axis

Sequences are aligned "one to one".
Greatly suffers from the misalignment in data.



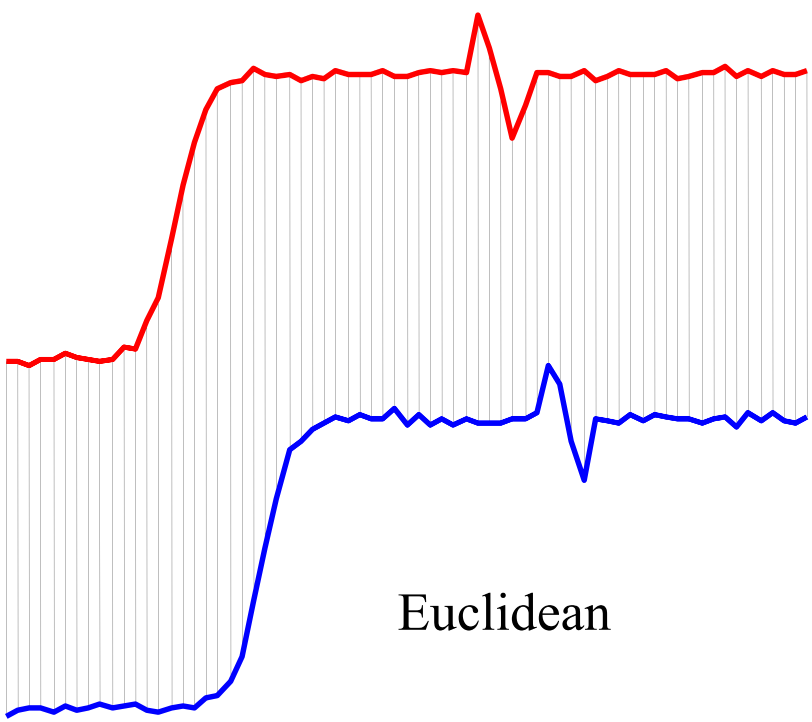
"Warped" Time Axis

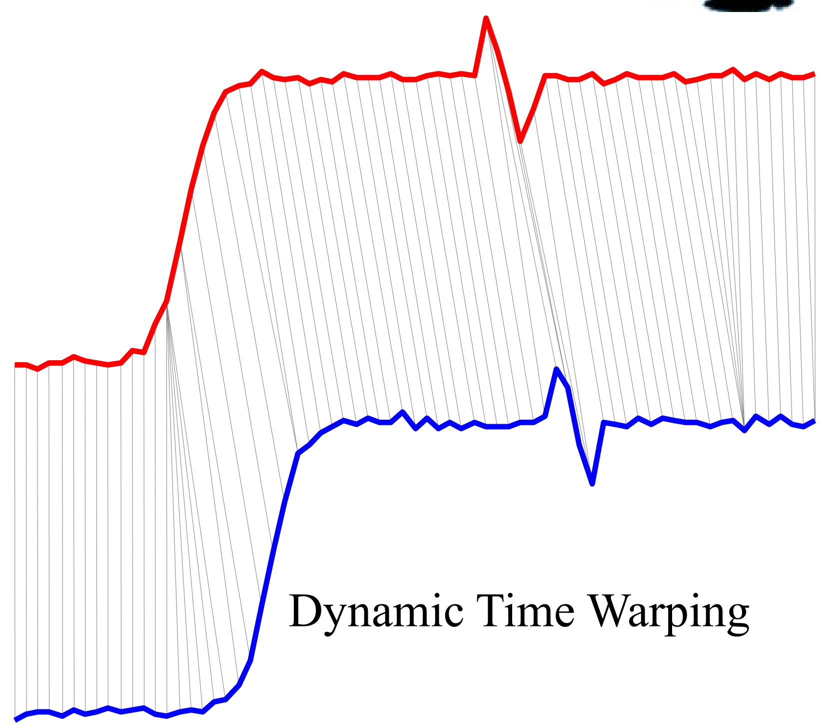
Nonlinear alignments are possible.
Can correct misalignments in data.

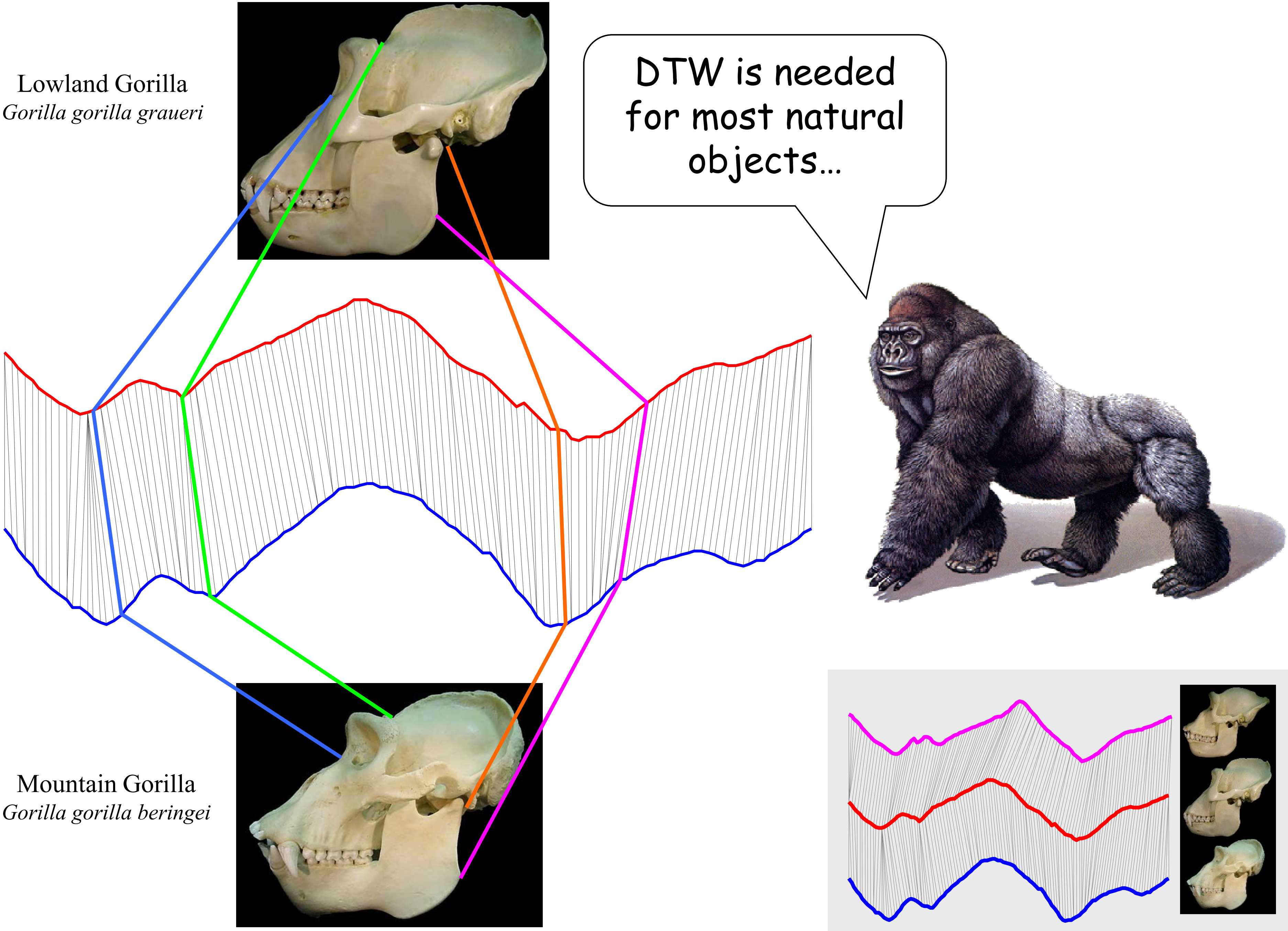


Here is another example on nuclear power plant trace data, to help you develop an intuition for DTW

Nuclear Power Excellent!

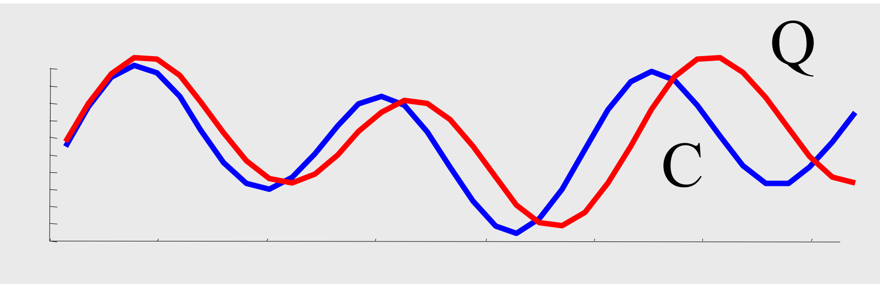


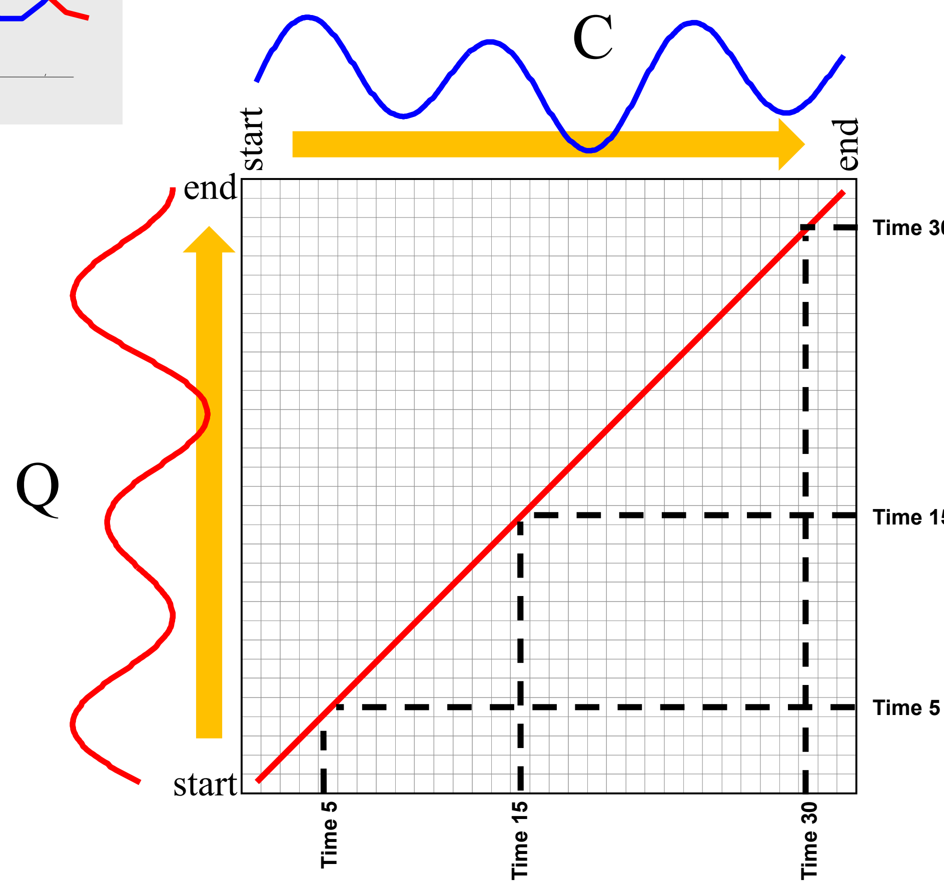



Lowland Gorilla
Gorilla gorilla graueri
DTW is needed for most natural objects…
Mountain Gorilla
Gorilla gorilla beringei

How is DTW Calculated? I

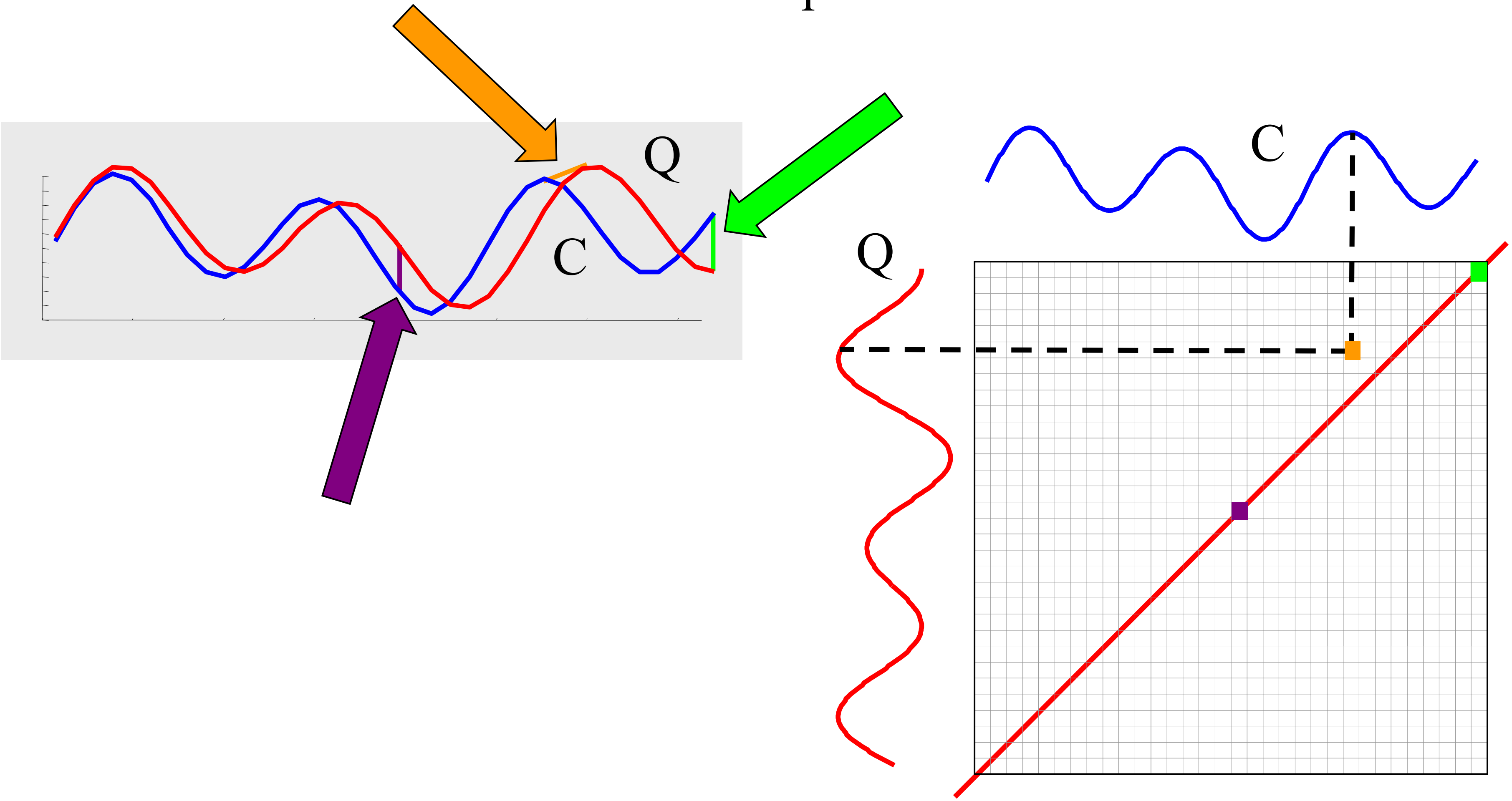
We create a matrix the size of |Q| by |C|, then fill it in with the distance between every pair of point in our two time series.





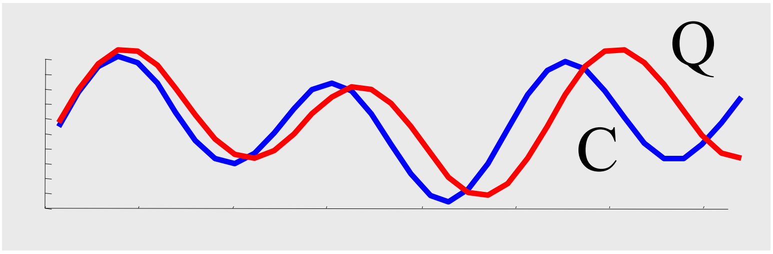
How is DTW Calculated? I

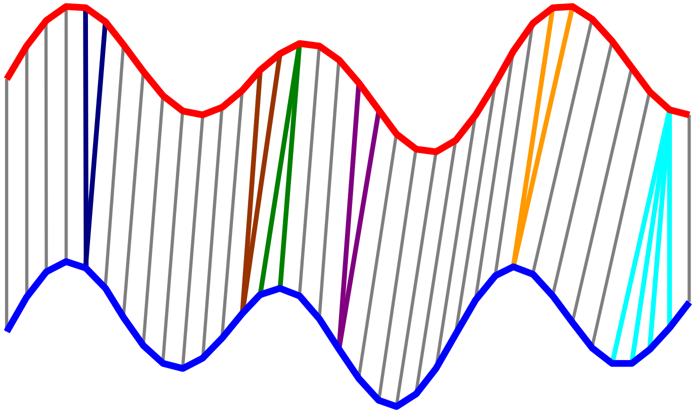
We create a matrix the size of |Q| by |C|, then fill it in with the distance between every pair of point in our two time series.

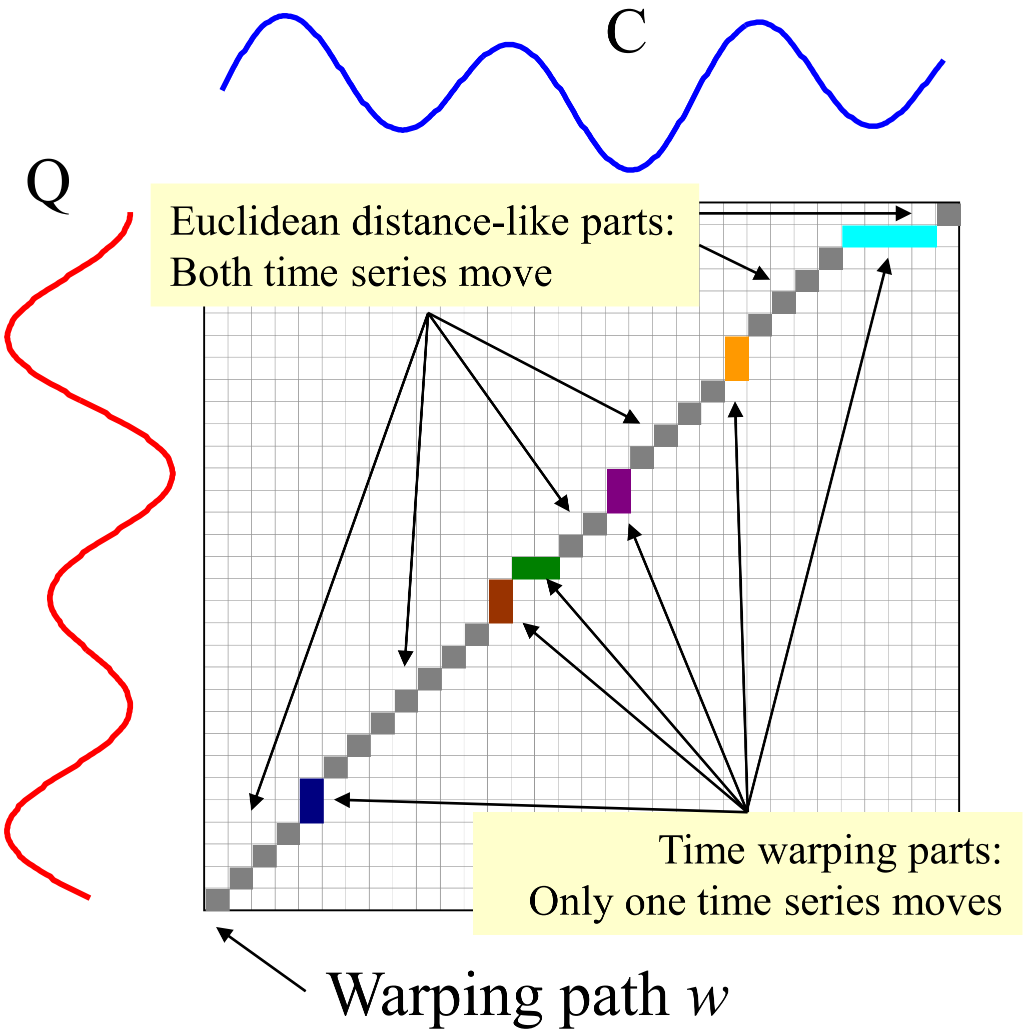


How is DTW Calculated? II

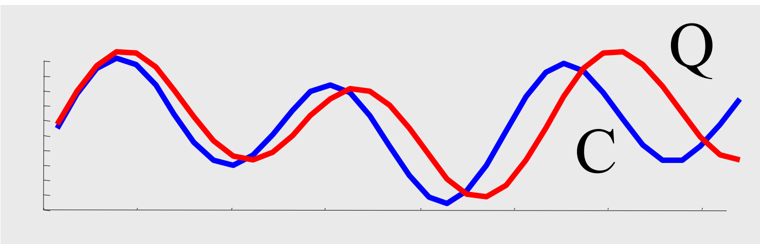
Every possible warping between two time series, is a path though the matrix.







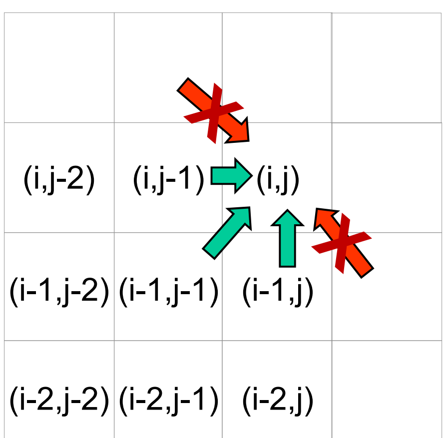
How is DTW Calculated? II



Definition of DTW as recursive function:

$$\gamma(i,j) = \text{cost of best path reaching cell } (i,j)$$
$$= d(q_i,c_j) + \min\{\ \gamma(i-1,j-1),\ \gamma(i-1,j),\ \gamma(i,j-1)\ \}$$

Idea: best path must pass through (i-1,j), (i-1,j-1) or (i,j-1)



We want the best warping path:

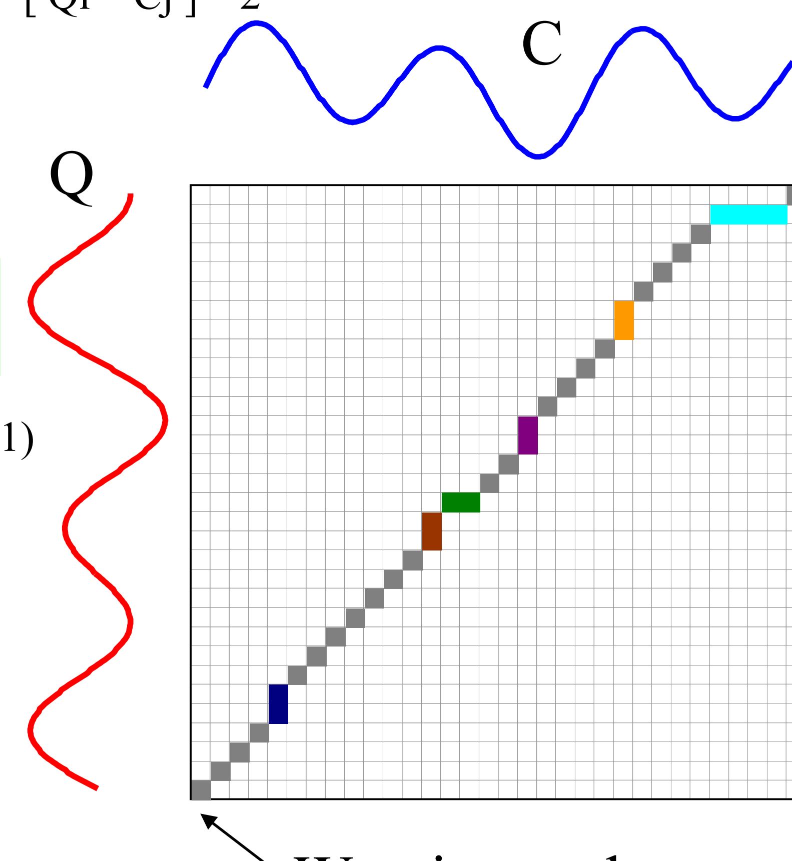
$$DTW(Q,C) = \min_{w \in PATHS} \sum_{k=1}^{|w|} w_k$$

w_k = cost of the k-th points comparison

Alternatives:

- $w_k = Qi - Cj$
- $w_k = [\ Qi - Cj\ ]\ \hat{}\ 2$

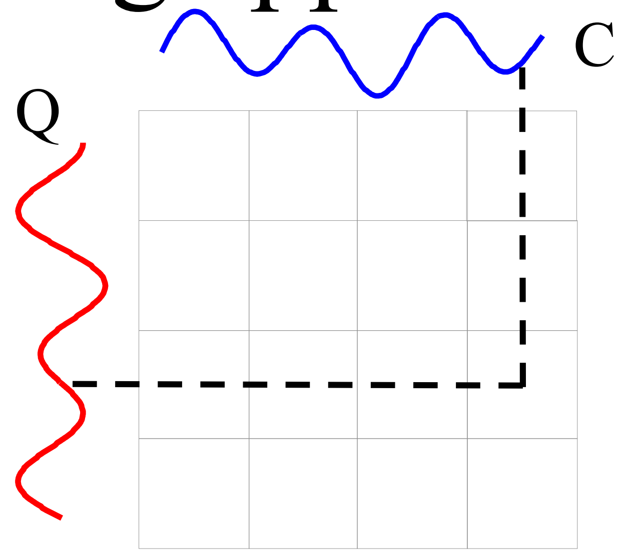
(i,j) = position of w_k



Warping path *w*

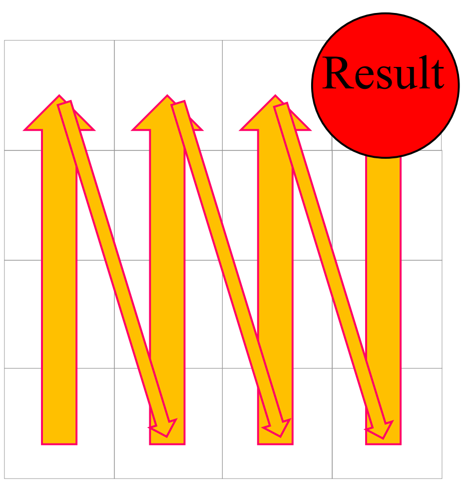
Dynamic programming approach

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



Step 1: compute the matrix of all $d(q_i,c_j)$

- Point-to-point distances
- $D(i,j) = |\, Qi - Cj \,|$



Step 2: compute the matrix of all path costs $\gamma(i,j)$

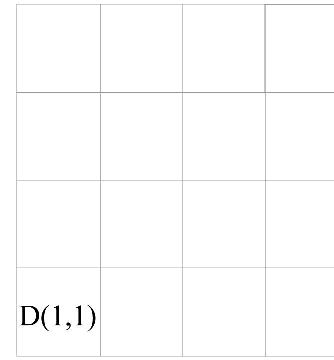
- Start from cell (1,1)
- Compute (2,1), (3,1), …, (n,1)
- Repeat for columns 2, 3, …, n
- Final result in last cell computed

Dynamic programming approach

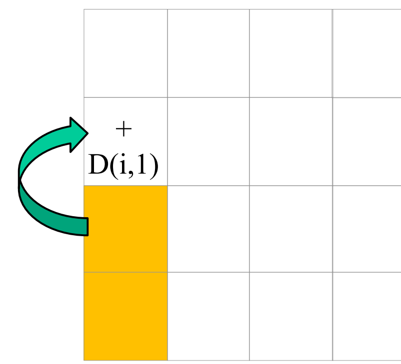
$\gamma(i,j) \;=\; d(q_i,c_j) + \min\{\, \gamma(i\text{-}1,j\text{-}1),\, \gamma(i\text{-}1,j\,),\, \gamma(i,j\text{-}1)\,\}$

Step 2: compute the matrix of all path costs $\gamma(i,j)$

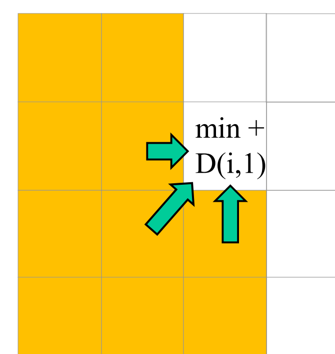
- Start from cell (1,1)
  - $\gamma(1,1) \;=\; d(q_1,c_1) + \min\{\, \gamma(\cancel{0},0),\, \gamma(\cancel{0},1),\, \gamma(\cancel{1},0)\}$
    $= d(q_1,c_1)$
    $= D(1,1)$



- Compute (2,1), (3,1), …, (n,1)
  - $\gamma(i,1) \;=\; d(q_i,c_1) + \min\{\, \gamma(i\text{-}\cancel{1},0),\, \gamma(i\text{-}1,1),\, \gamma(\cancel{i},0)\,\}$
    $= d(q_i,c_1) + \gamma(i\text{-}1,1)$
    $= D(i,1) + \gamma(i\text{-}1,1)$



- Repeat for columns 2, 3, …, n
  - The general formula applies



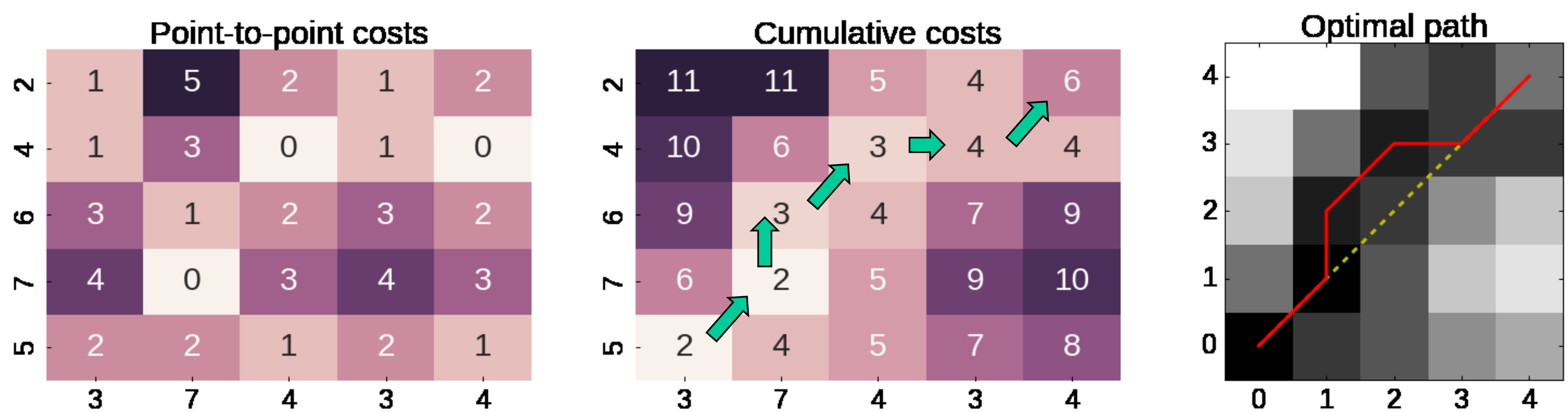
Dynamic programming approach

Example and How to infer the optimal path

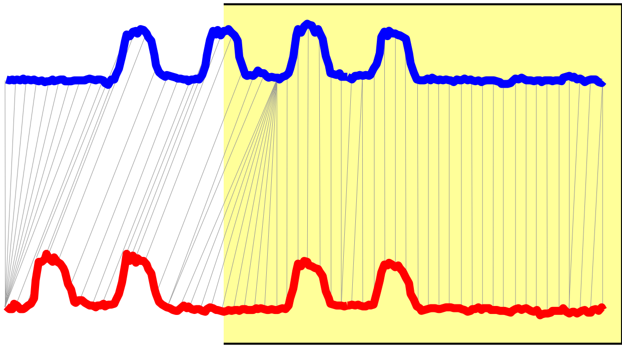
t = < 2, 3, 2, 7, 4 >

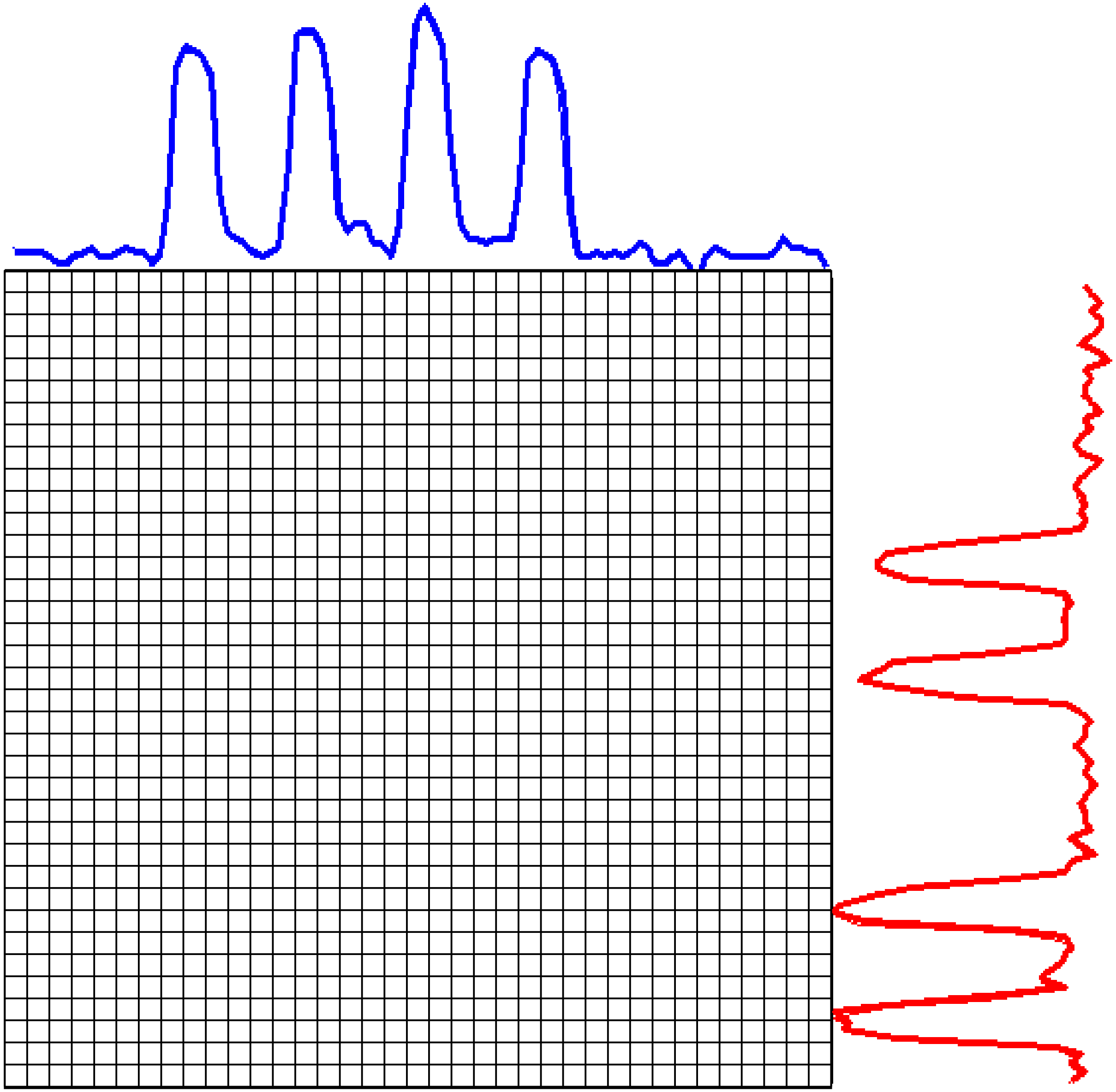
q = < 5, 7, 6, 4, 2 >

$$\gamma(\mathrm{i},\mathrm{j}) \quad = \quad d(q_{\mathrm{i}},c_{\mathrm{j}}) + \min\{\ \gamma(i\text{-}1,j\text{-}1),\ \gamma(i\text{-}1,j\ ),\ \gamma(i,j\text{-}1)\ \}$$



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

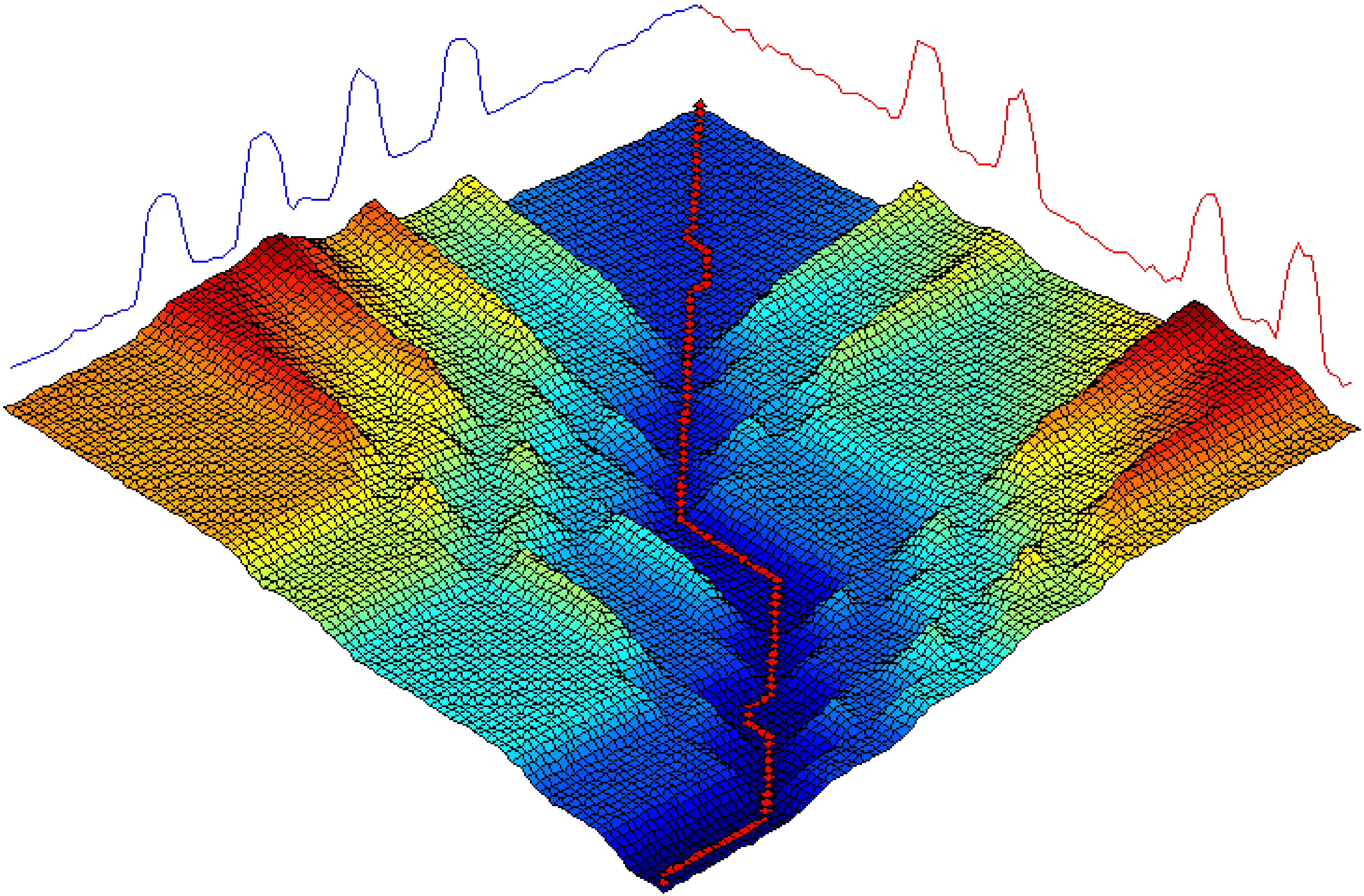




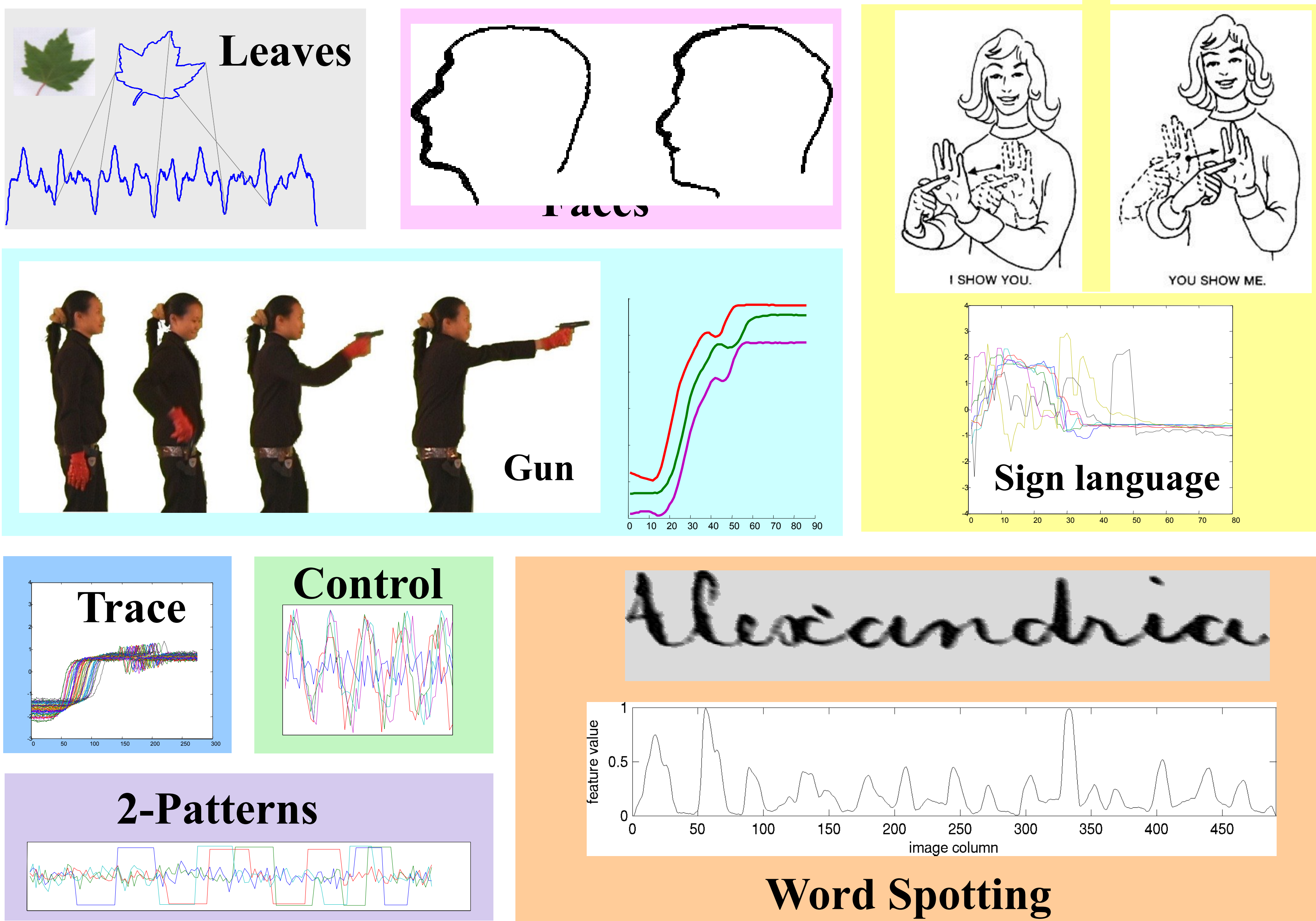
This example shows 2 one-week periods from the power demand time series.

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



Let us compare Euclidean Distance and DTW on some problems



Results: Error Rate

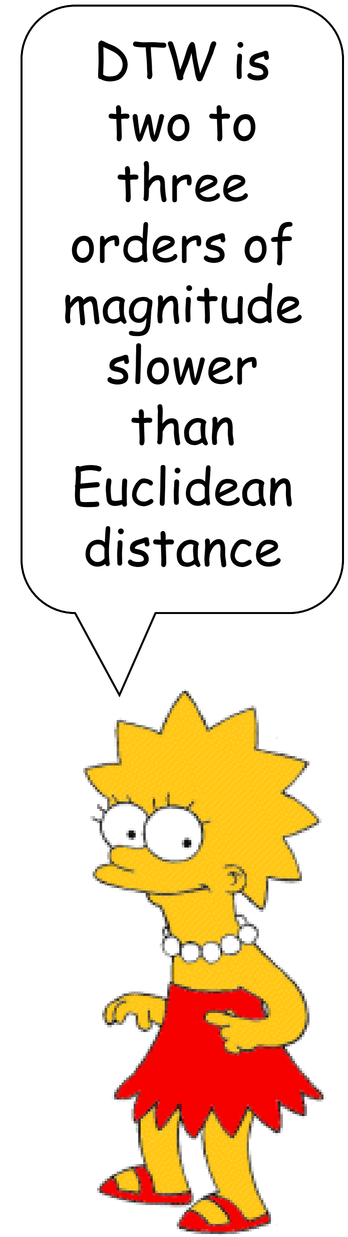
Classification using **1-nearest-neighbor**
- Class(x) = class of most similar training object

Leaving-one-out evaluation
- For each object: use it as test set, return overall average

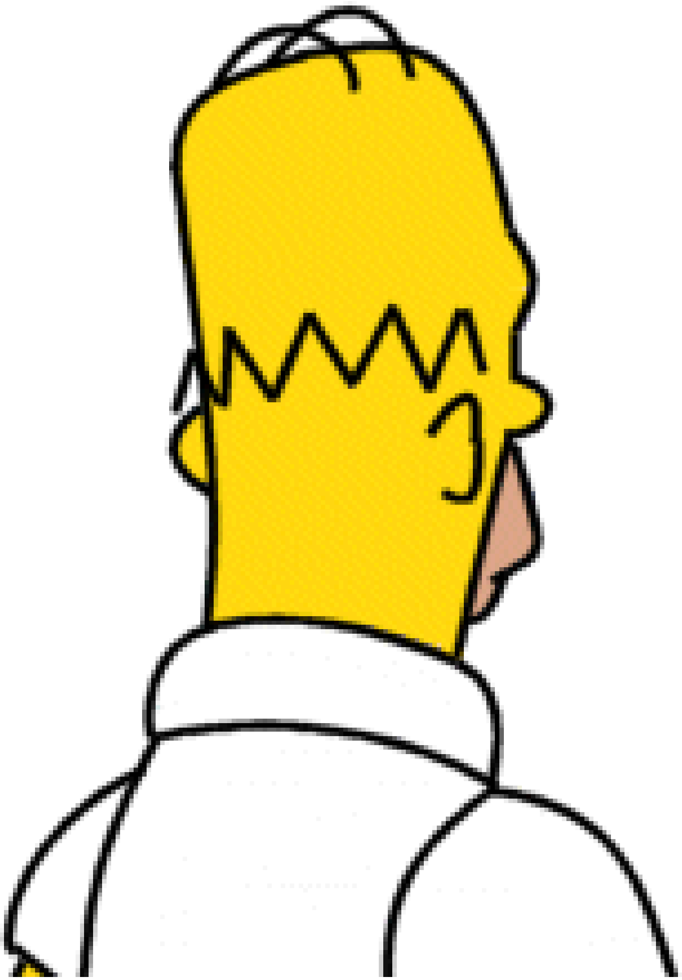
Dataset	Euclidean	DTW
Word Spotting	4.78	1.10
Sign language	28.70	25.93
GUN	5.50	1.00
Nuclear Trace	11.00	0.00
Leaves#	33.26	4.07
(4) Faces	6.25	2.68
Control Chart*	7.5	0.33
2-Patterns	1.04	0.00

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



What we have seen so far…

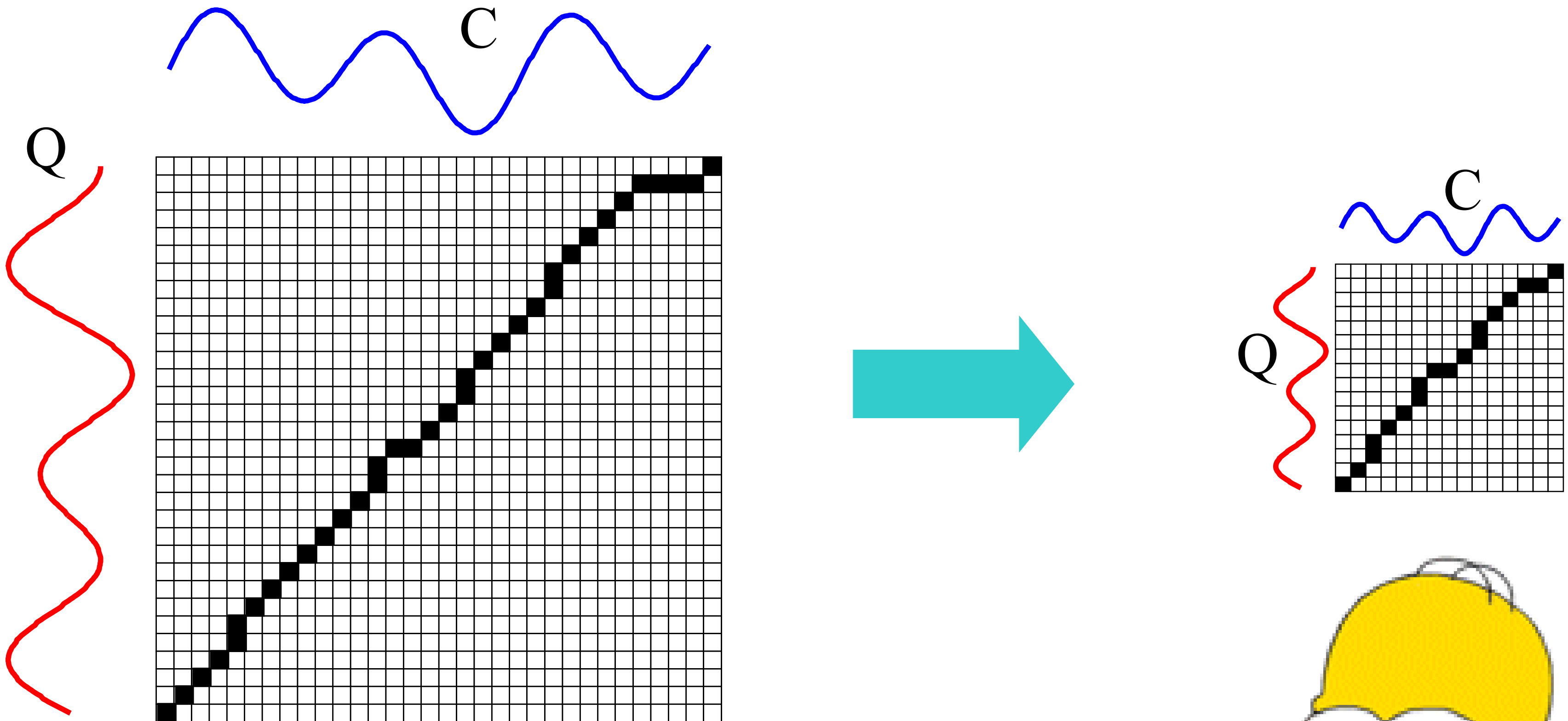


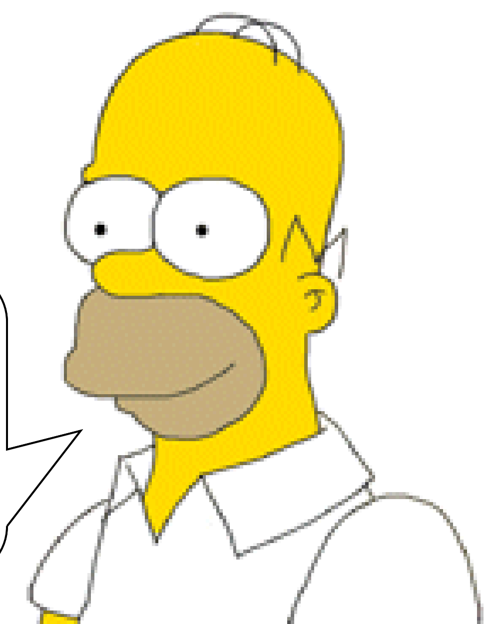
- Dynamic Time Warping gives **much better** results than Euclidean distance on virtually all problems.

- Dynamic Time Warping is very very slow to calculate!

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

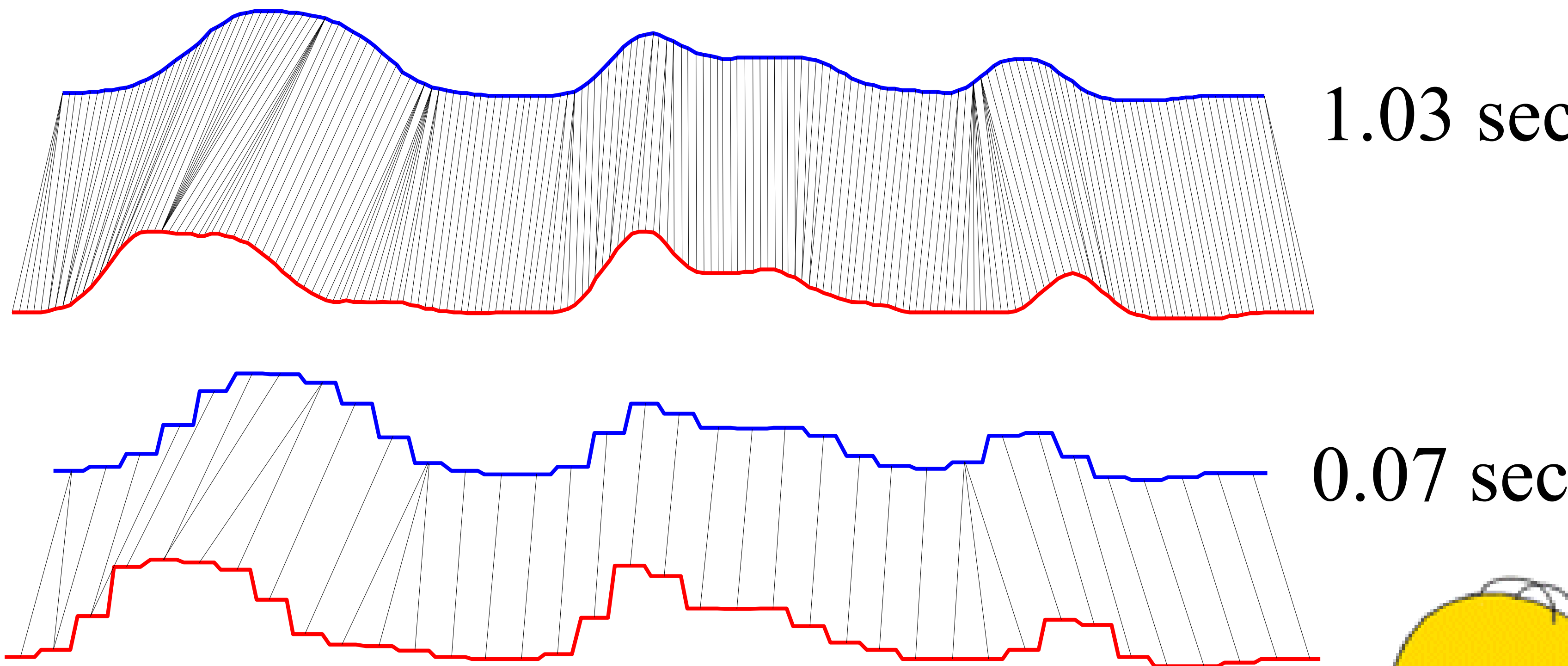
Fast Approximations to Dynamic Time Warp Distance I





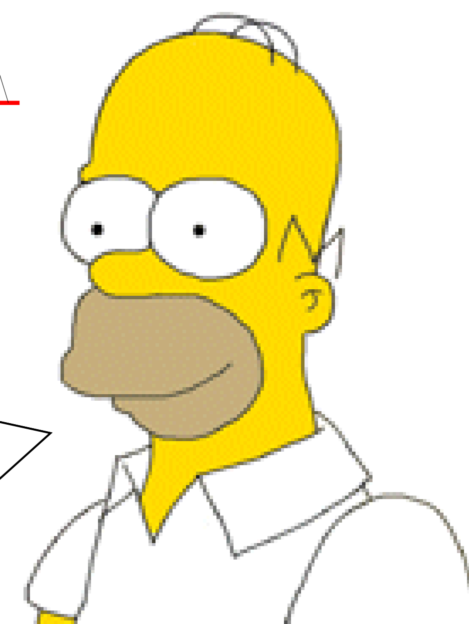
Simple Idea: Approximate the time series with some compressed or downsampled representation, and do DTW on the new representation. How well does this work...

Fast Approximations to Dynamic Time Warp Distance II



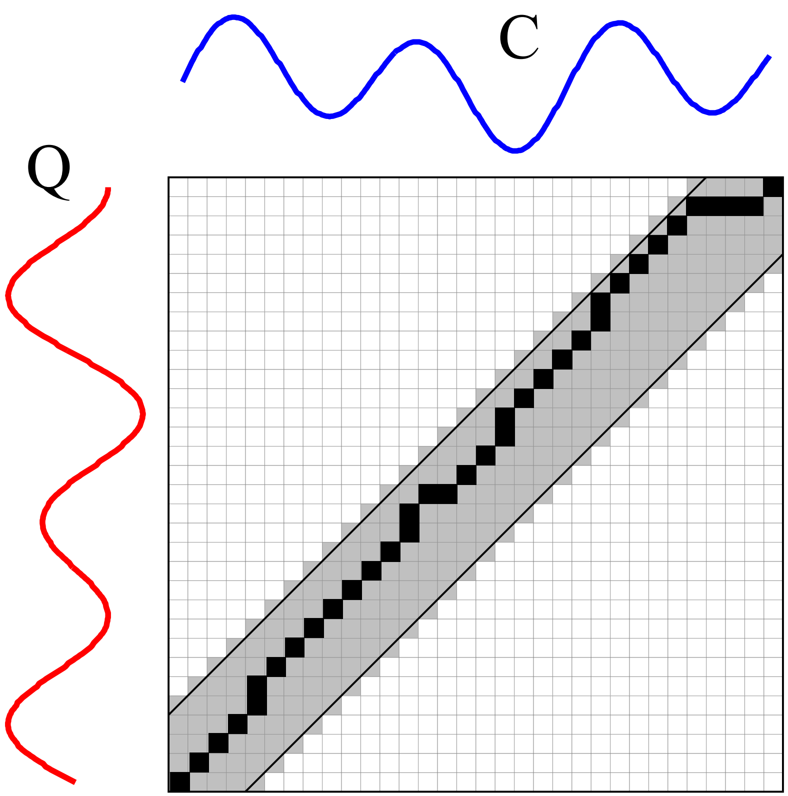
… there is strong visual evidence to suggests it works well

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

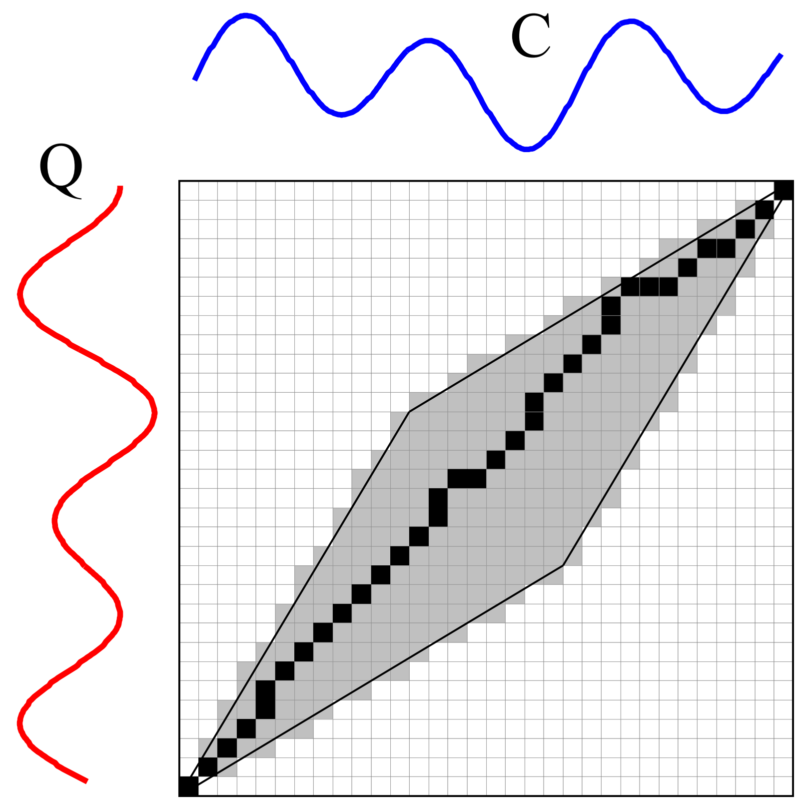


Global Constraints

- Slightly speed up the calculations
- Prevent pathological warpings



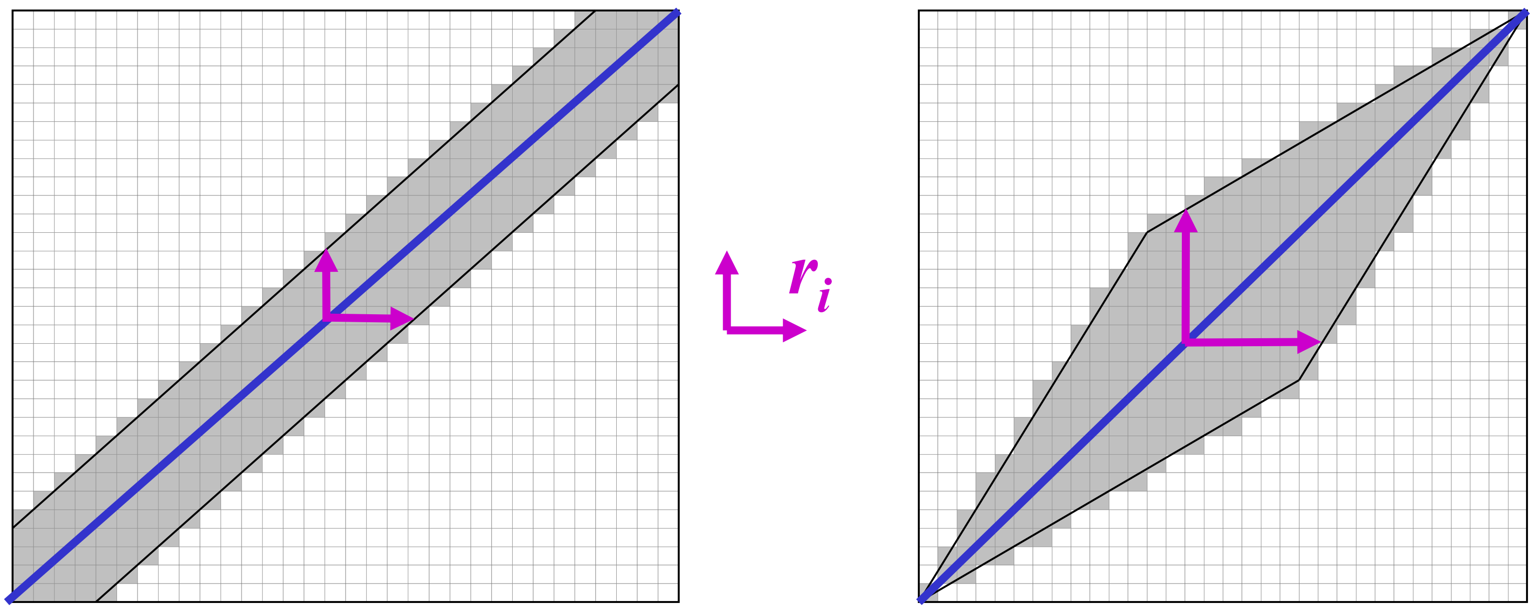
Sakoe-Chiba Band



Itakura Parallelogram

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

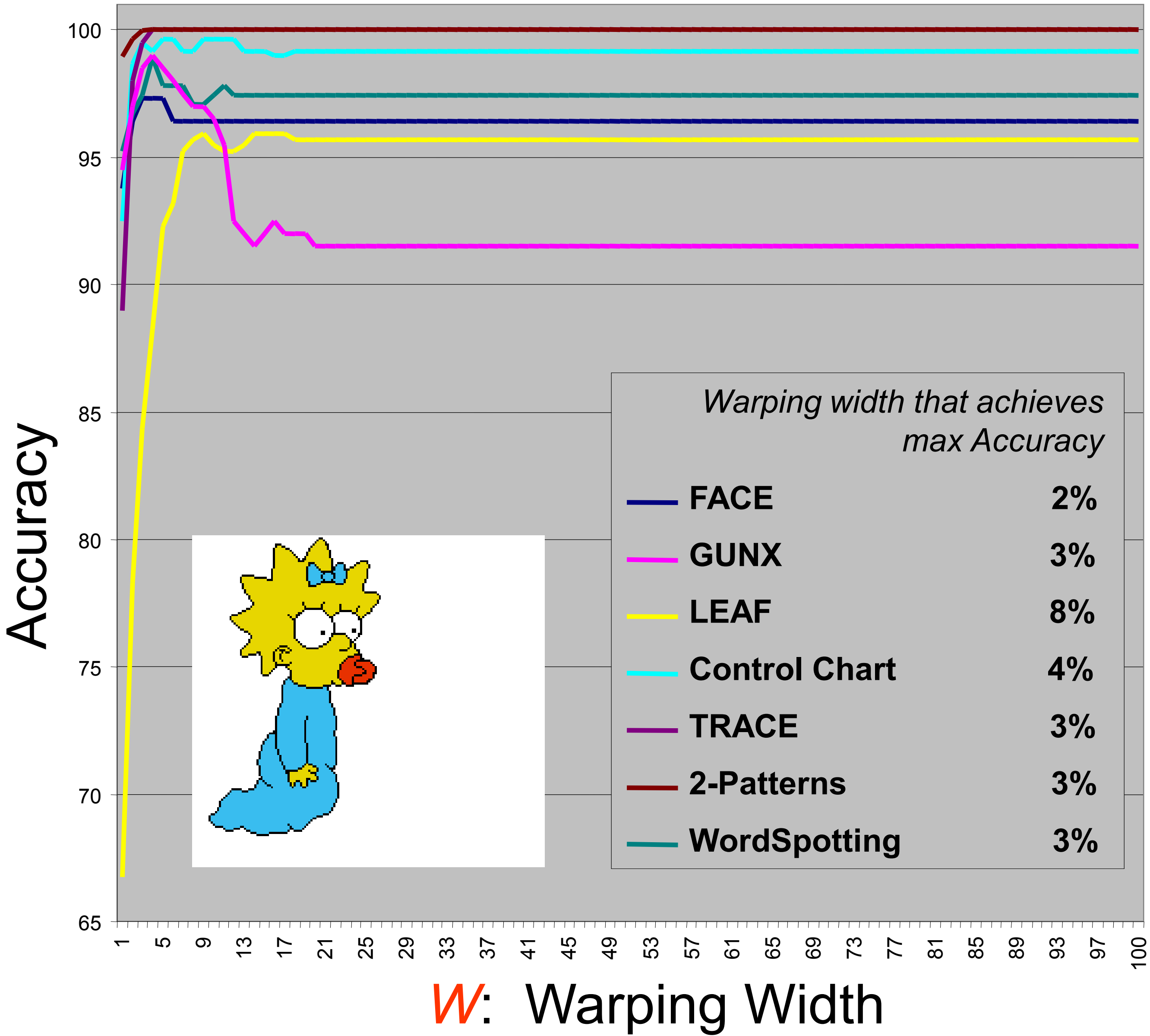
Where r is a term defining allowed range of warping for a given point in a sequence.

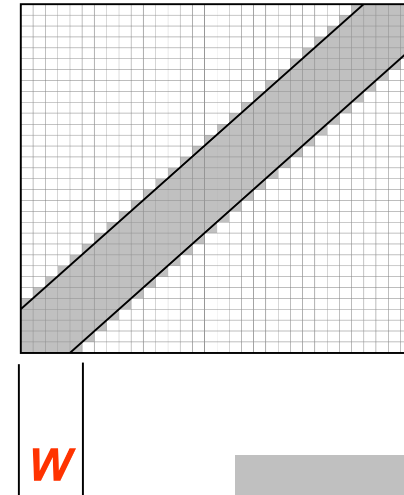


Sakoe-Chiba Band

Itakura Parallelogram

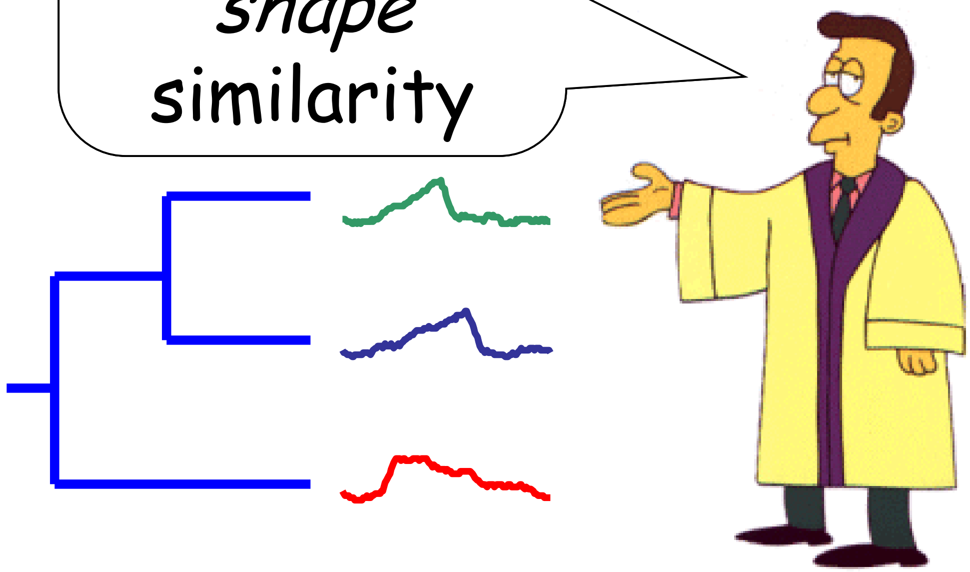
Accuracy vs. Width of Warping Window



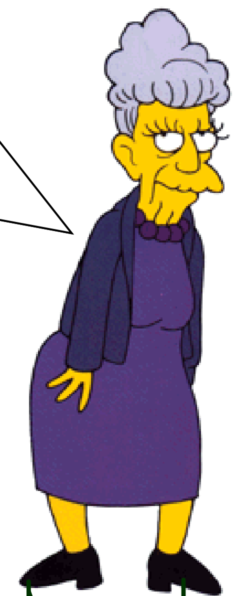


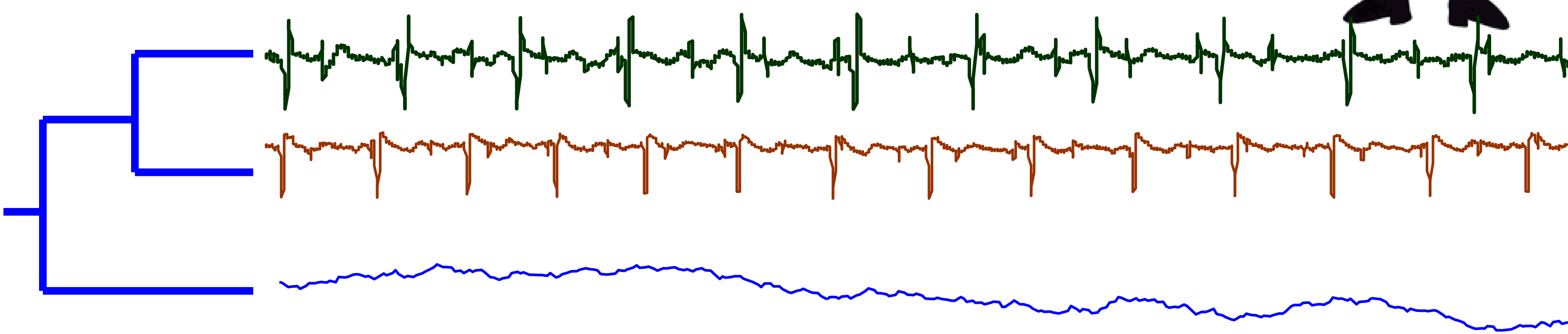
Two Kinds of Similarity

We are done with *shape* similarity

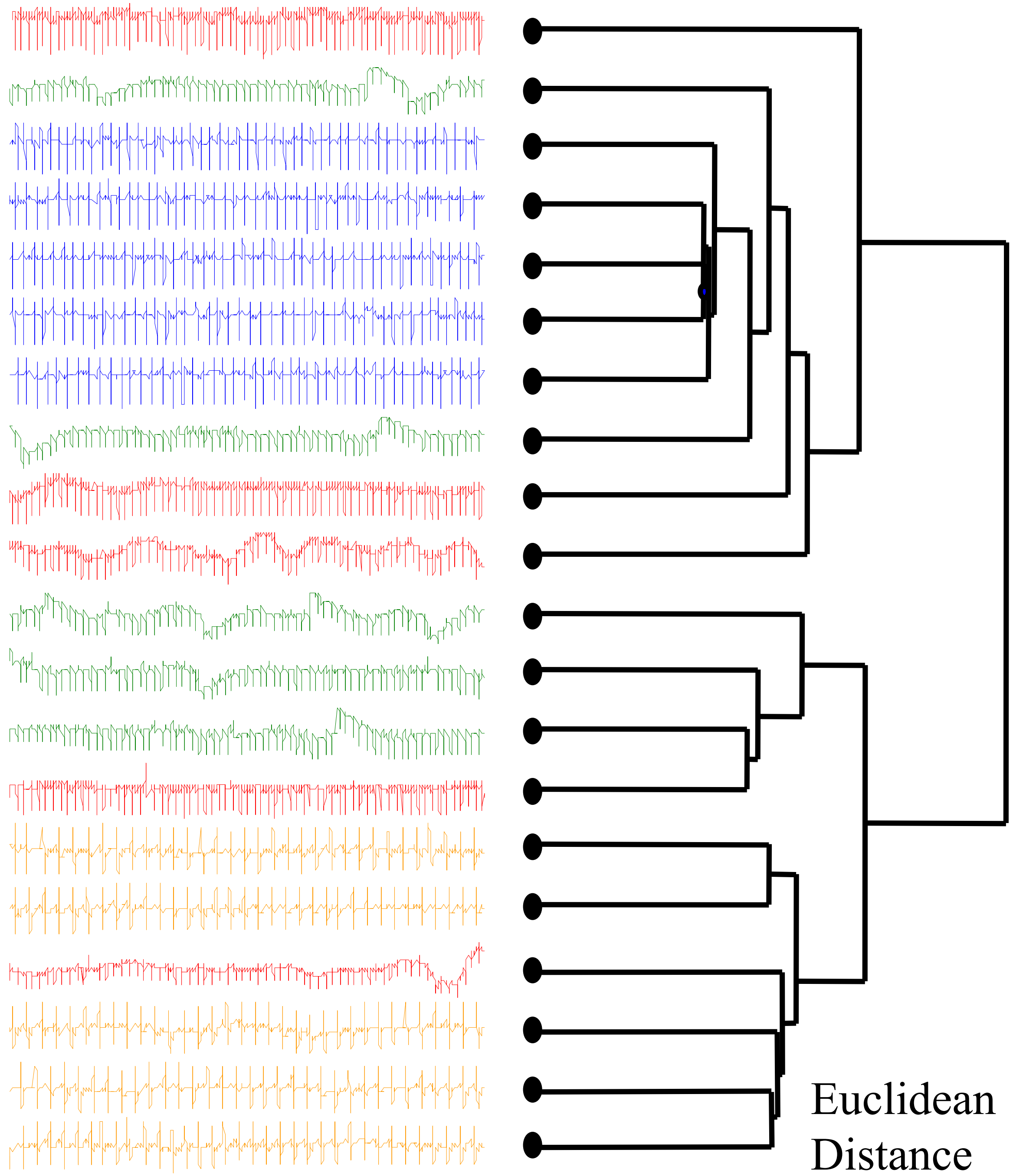


Let us consider similarity at the *structural* level for the next 10 minutes



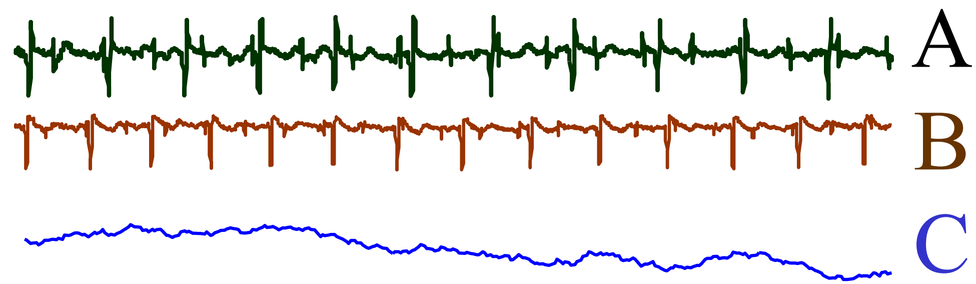



For long time series, *shape* based similarity will give very poor results. We need to measure similarly based on high level *structure*


Euclidean Distance

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**?

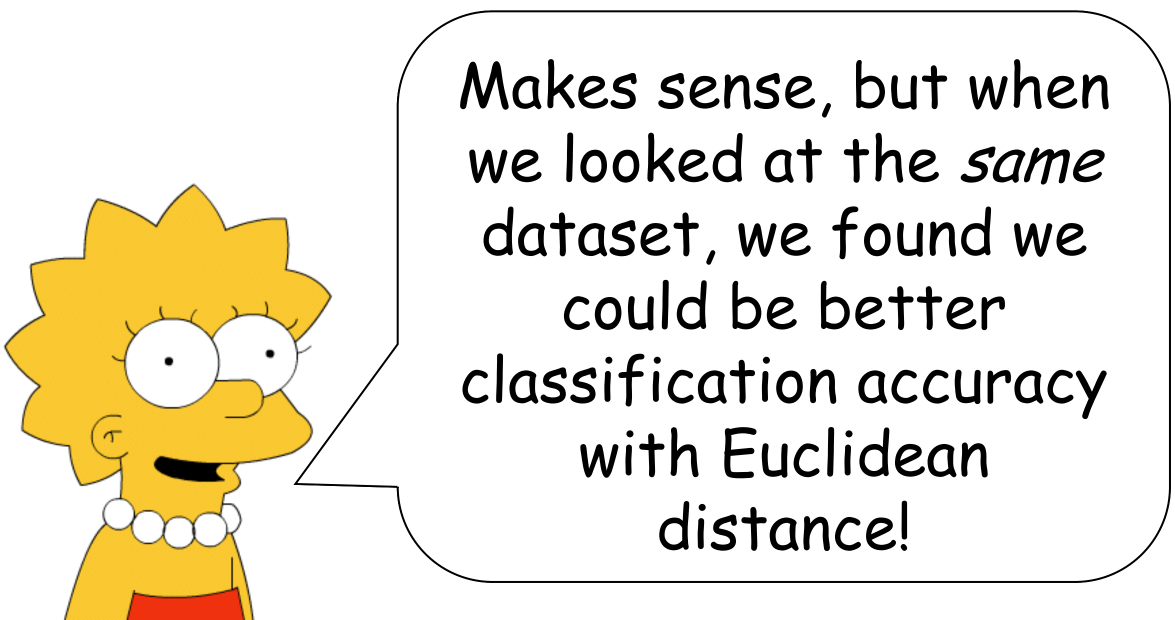
Feature-based Classification of Time-series Data

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?

Distance Measure

“Viterbi-Like” Algorithm

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

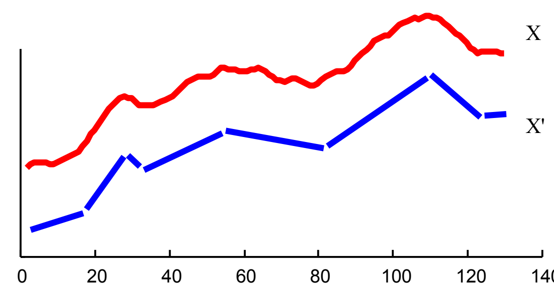
There tends to be lots of parameters to tune…

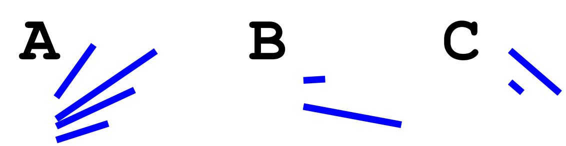
	A	B	C
A	0.1	0.4	0.5
B	0.4	0.2	0.2
C	0.5	0.2	0.3

Features

The parameters of a Markov Model

The time series is first converted to a piecewise linear model



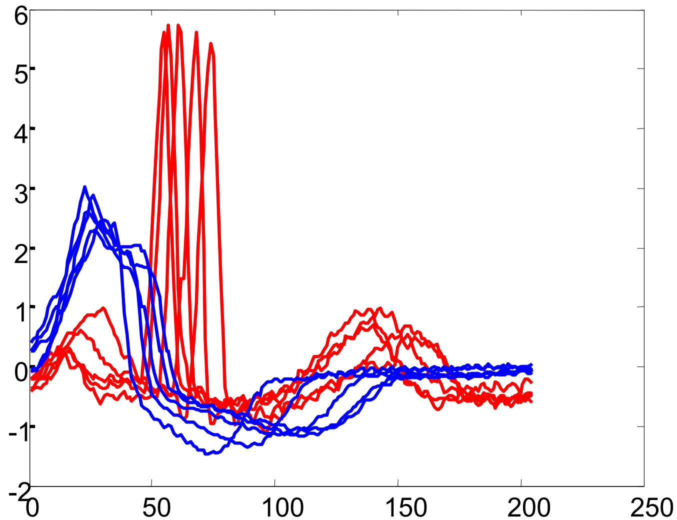


Deformable Markov Model Templates for Time Series Pattern Matching

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

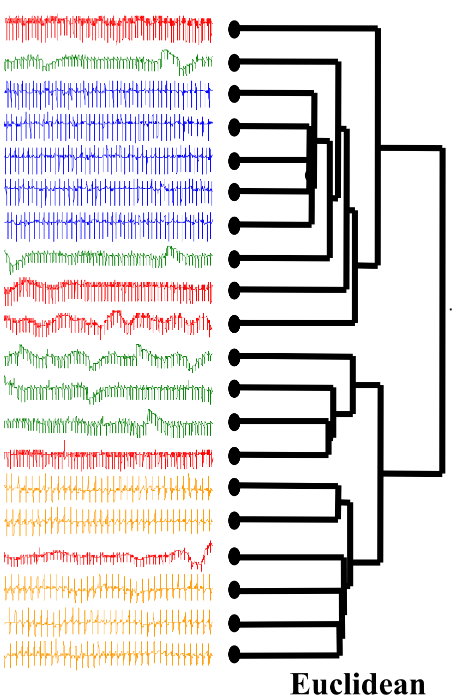
(**In general**) Li, Chen, Li, Ma, and Vitányi: (**For time series**) Keogh, Lonardi and Ratanamahatana

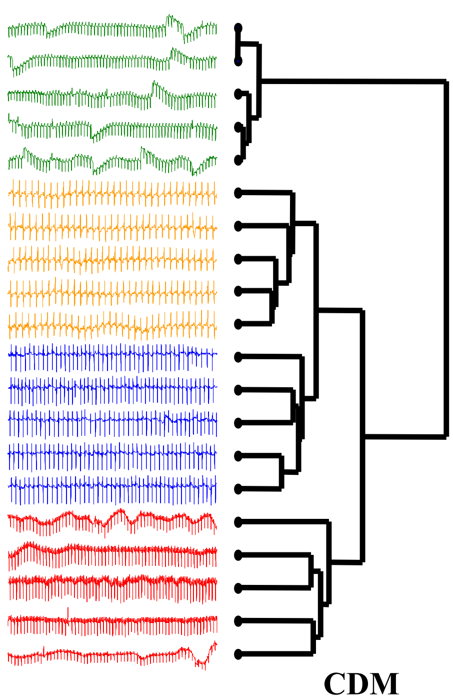
- features?
- distance measure/ learning algorithm?

Distance Measure

Co-Compressibility

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



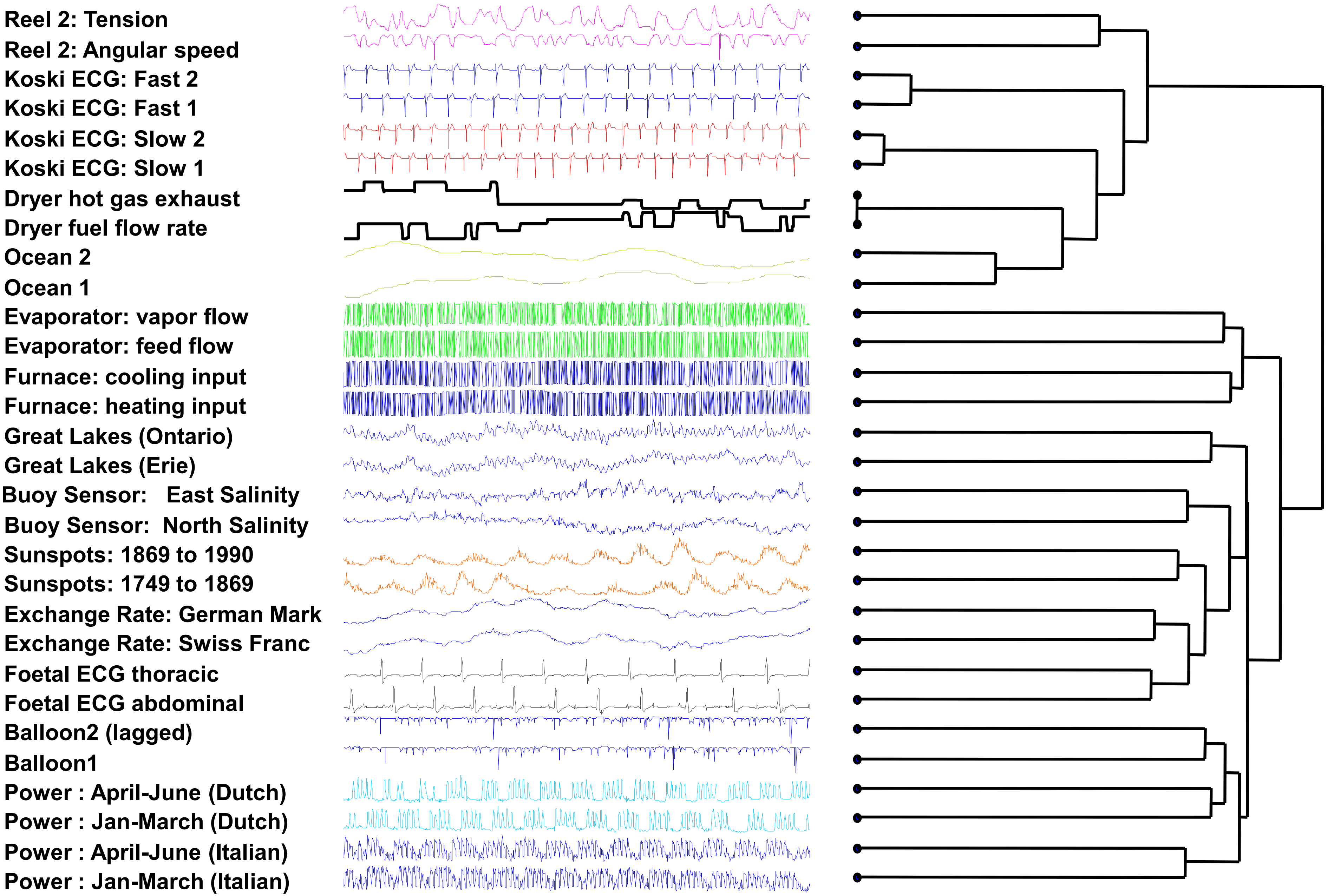


Features

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

- 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 =>
    extract features

Anomaly (interestingness) detection

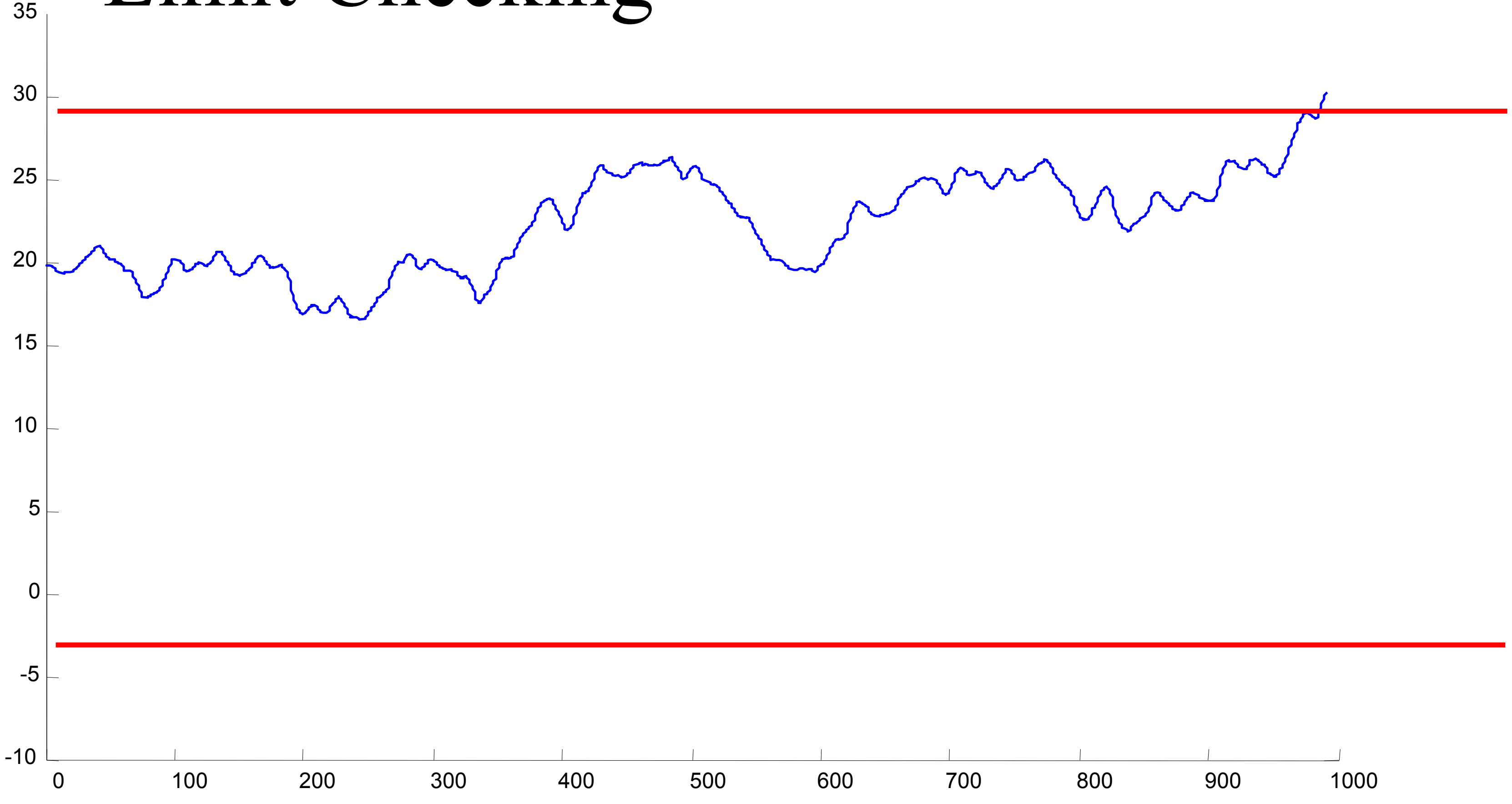
We would like to be able to discover surprising (unusual, interesting, anomalous) patterns in time series.

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.

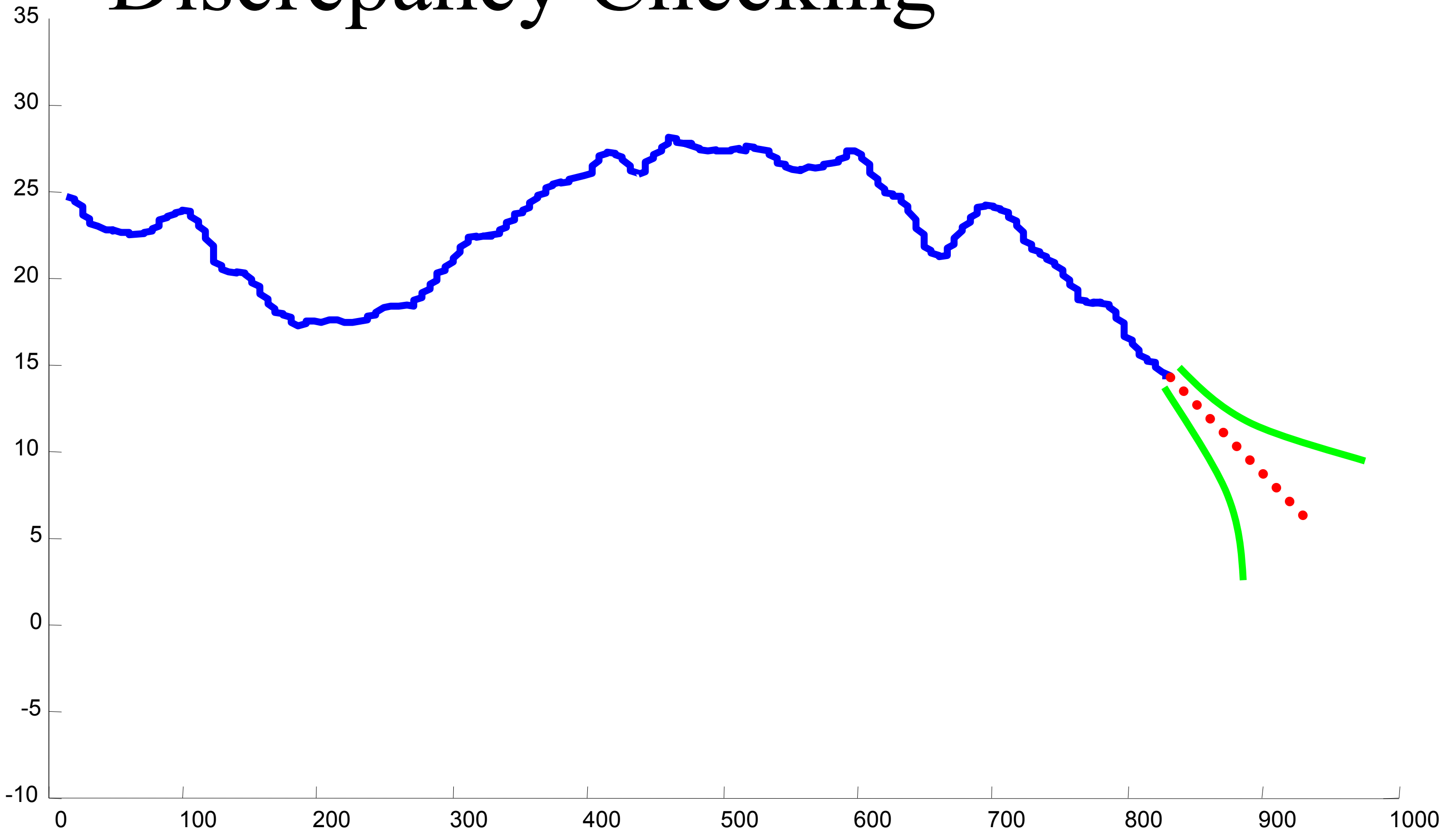
Simple Approaches I

Limit Checking



Simple Approaches II

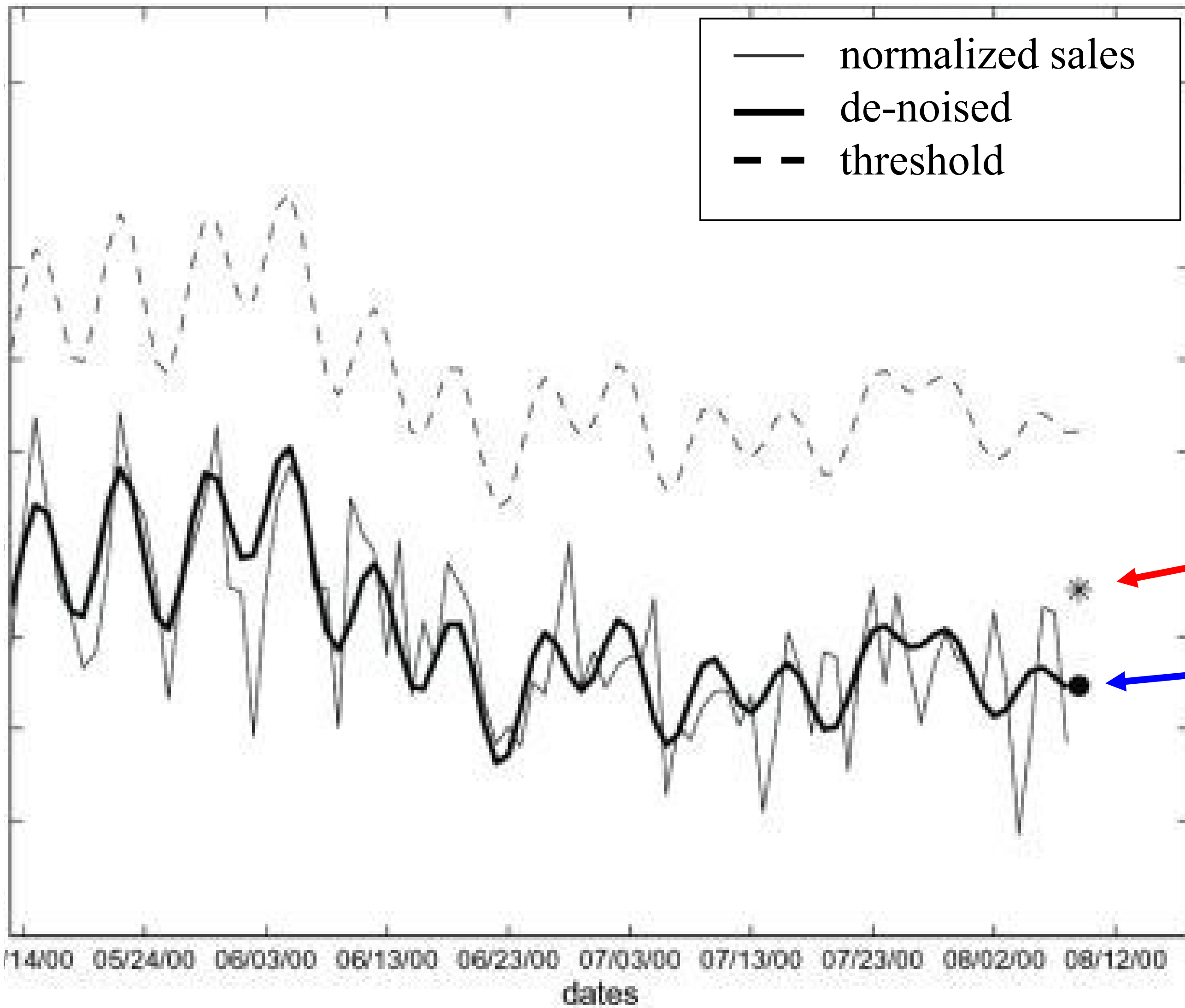
Discrepancy Checking



Discrepancy Checking: Example

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

Goldenberg, Shmueli, Caruana, and Fienberg



Actual value

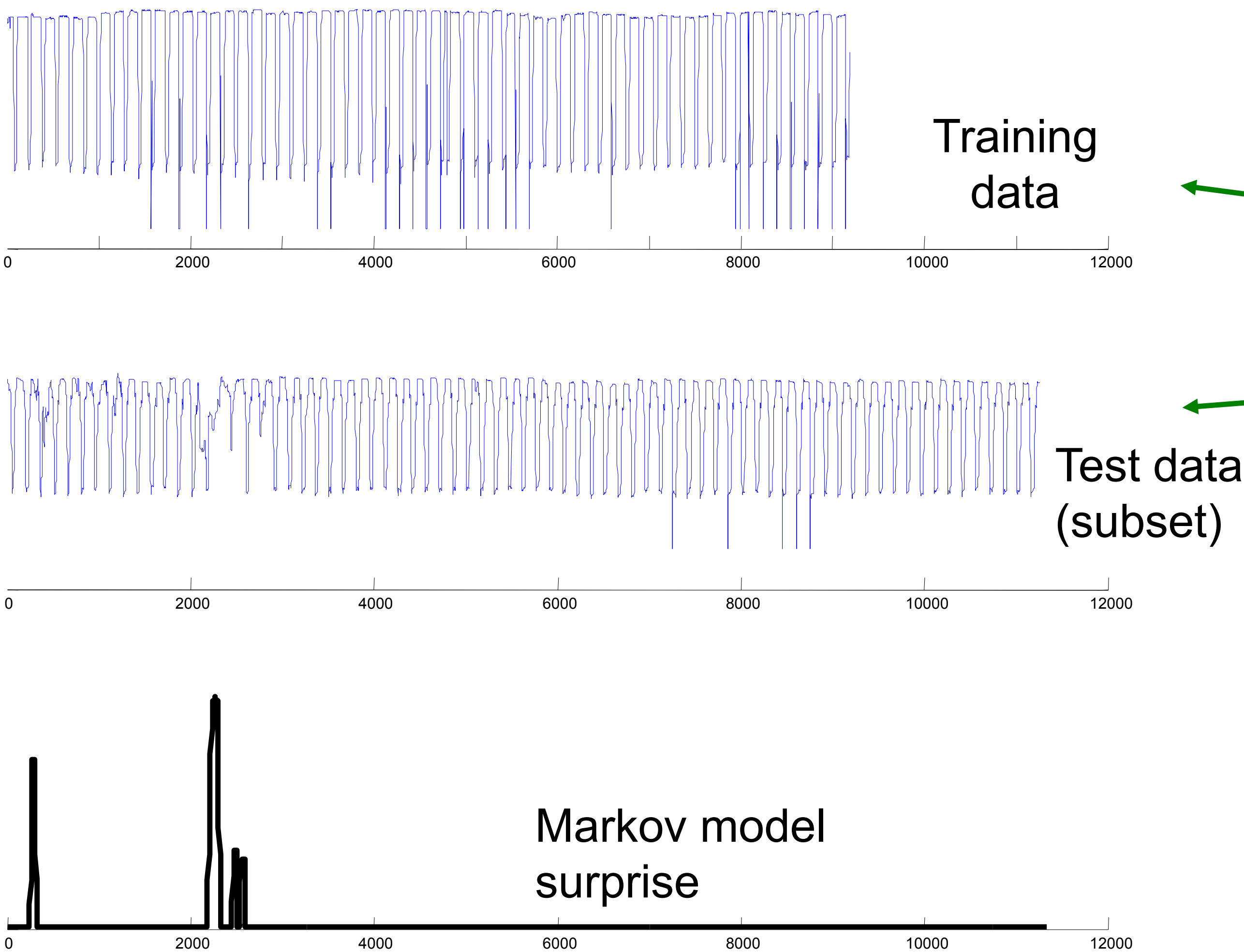
Predicted value

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*.

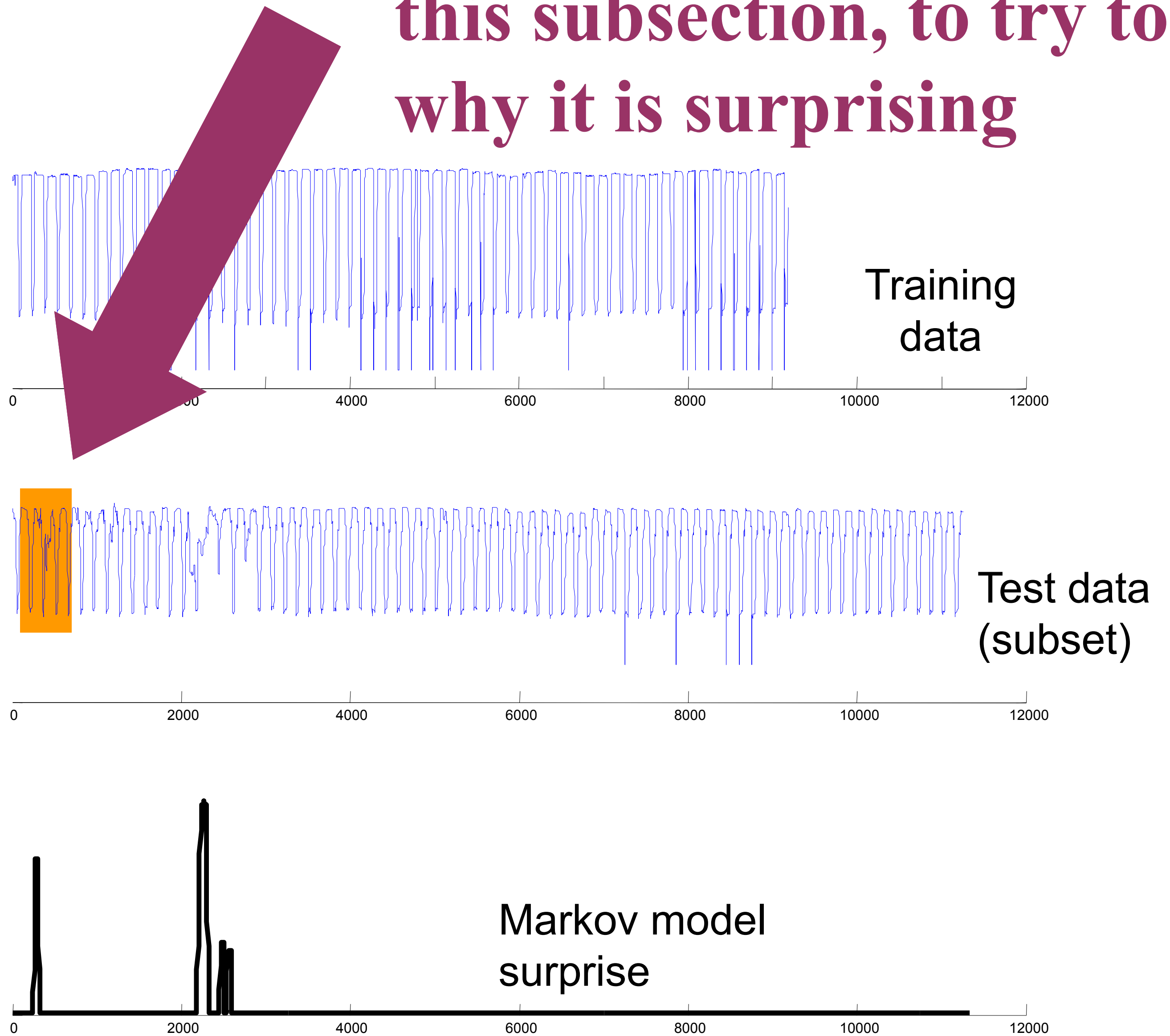
- Time series can be easily converter to text
    - Discretization of numerical values

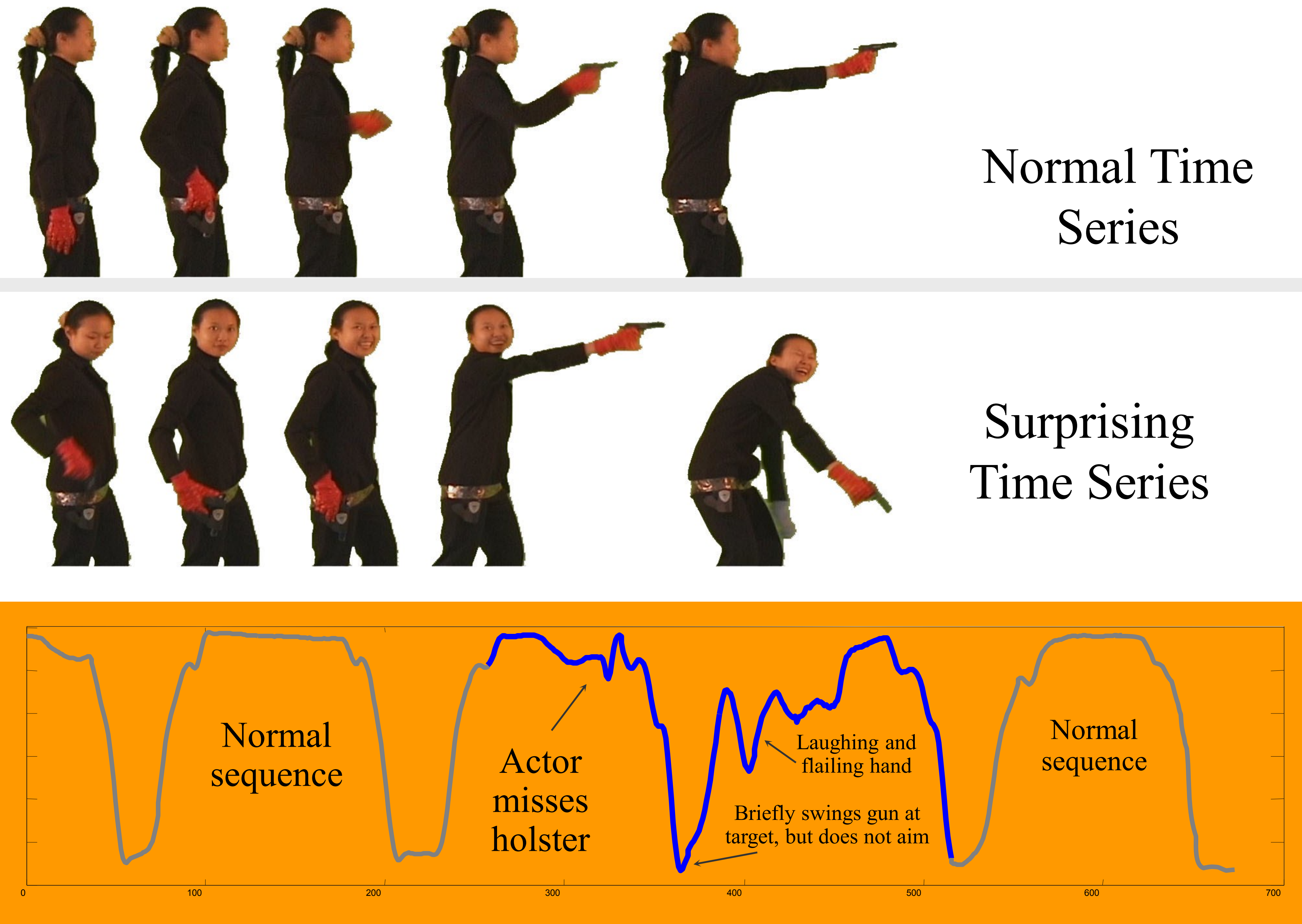
We can use Markov models to find surprises in time series…



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

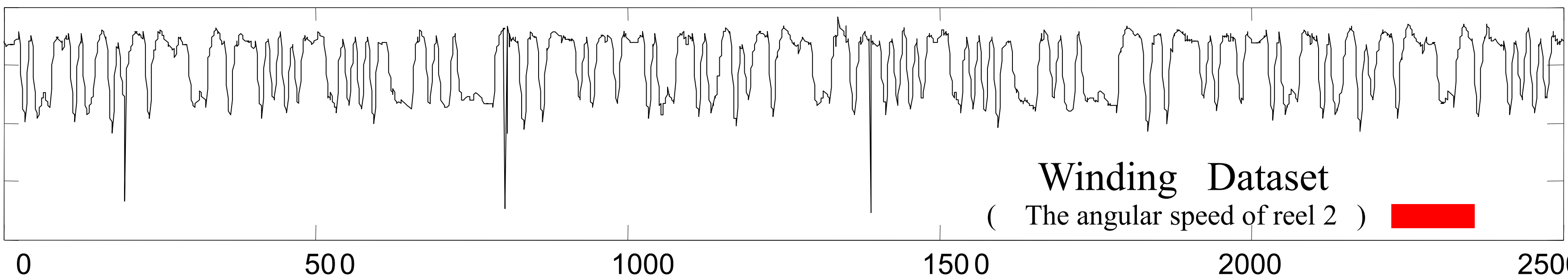
Anomaly (interestingness) detection

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

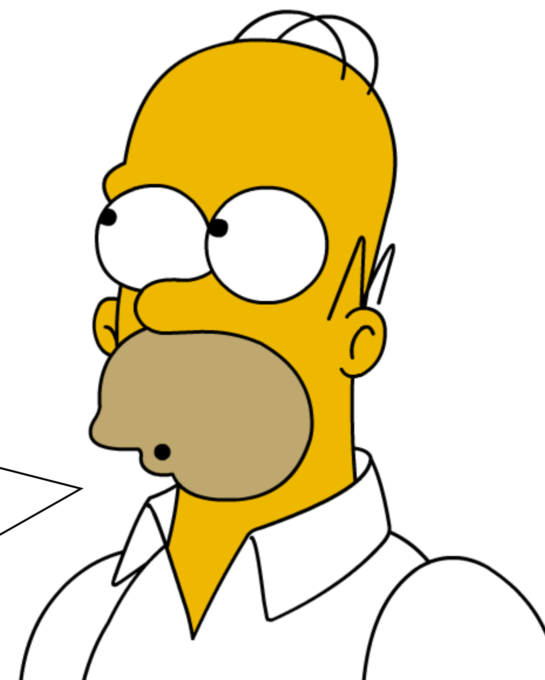
How can we find interesting patterns…

- 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)

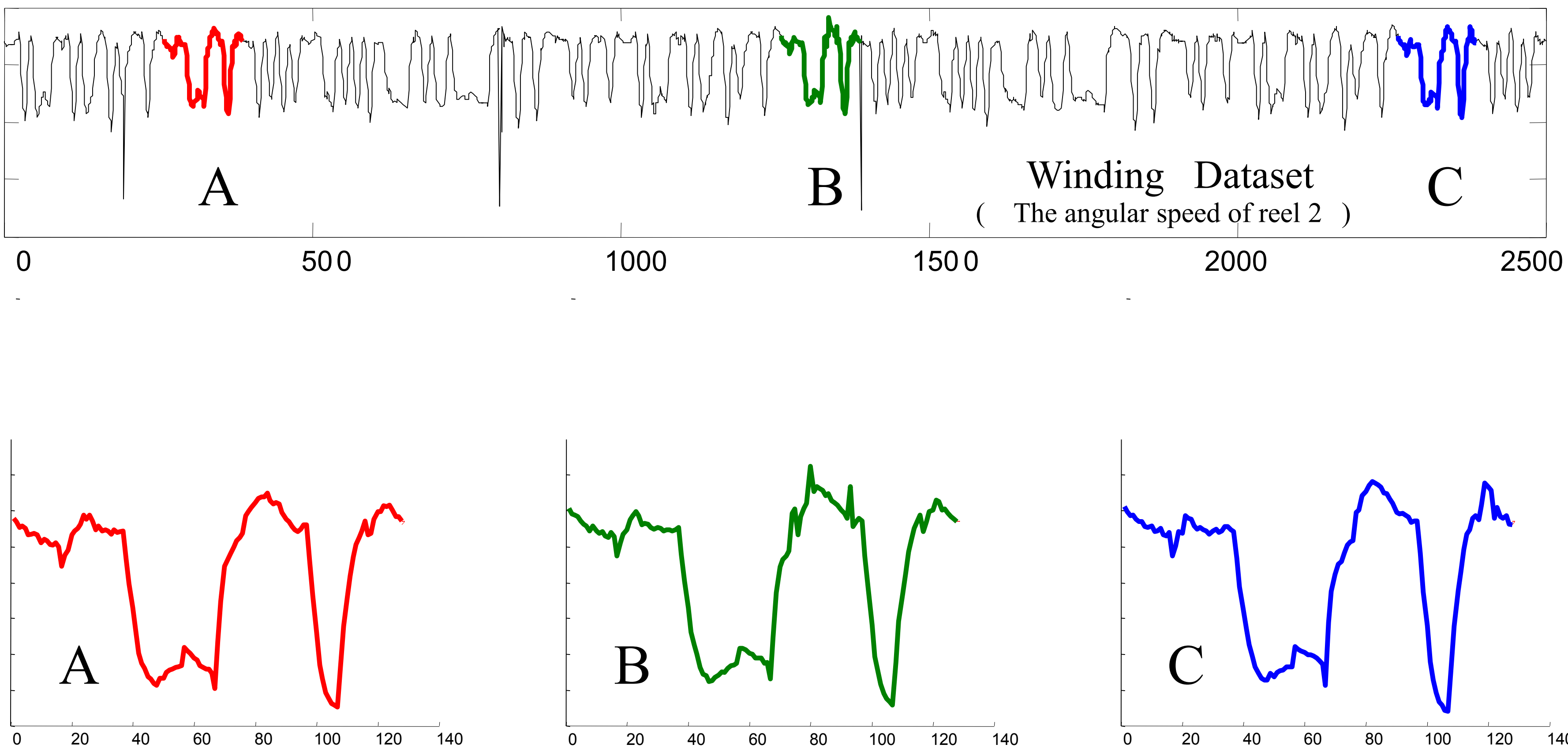


Are there any repeated patterns, of about this length in the above time series?



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*.

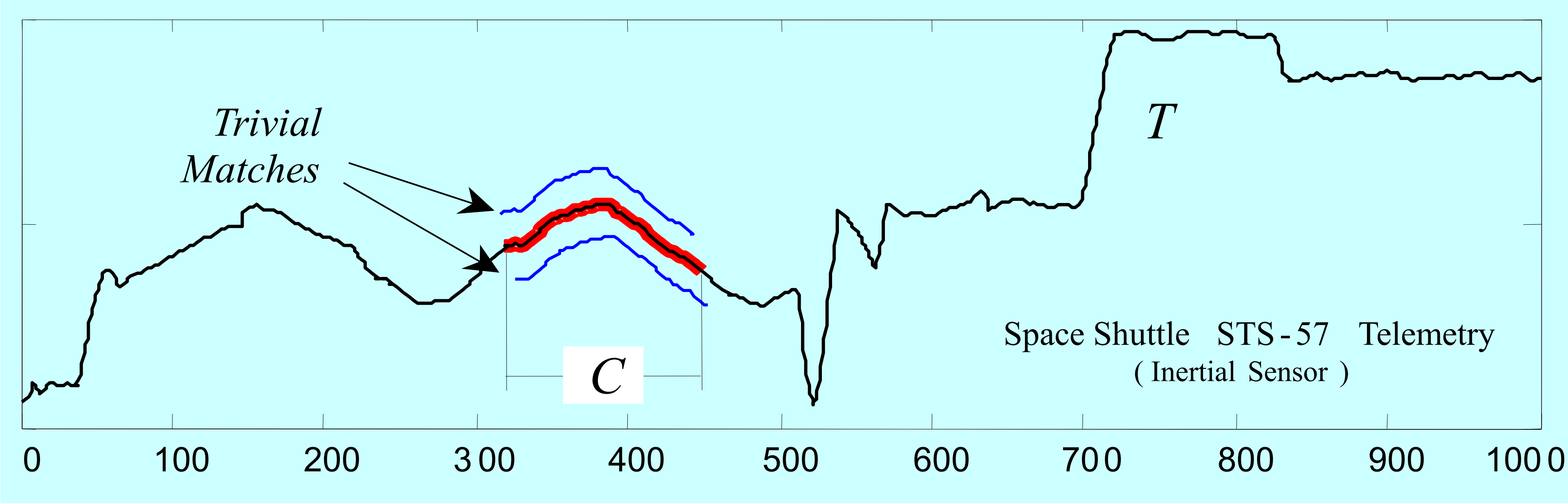
· Several time series **classification algorithms** work by constructing typical prototypes of each class. These prototypes may be considered motifs.

· 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)



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?

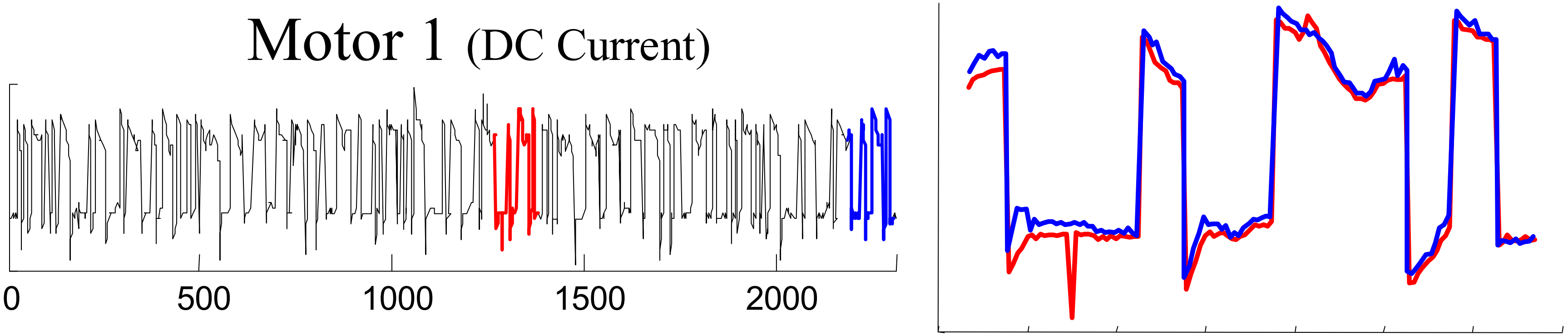
The obvious brute force search algorithm is just too slow…

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.

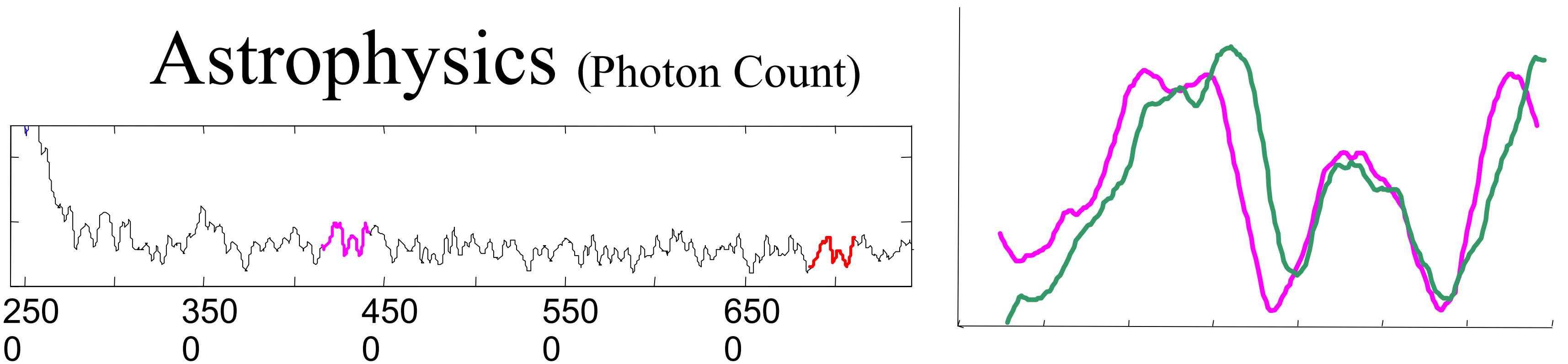
*** J Buhler and M Tompa. *Finding motifs using random projections*. In RECOMB'01. 2001.**

Some Examples of Real Motifs

Motor 1 (DC Current)



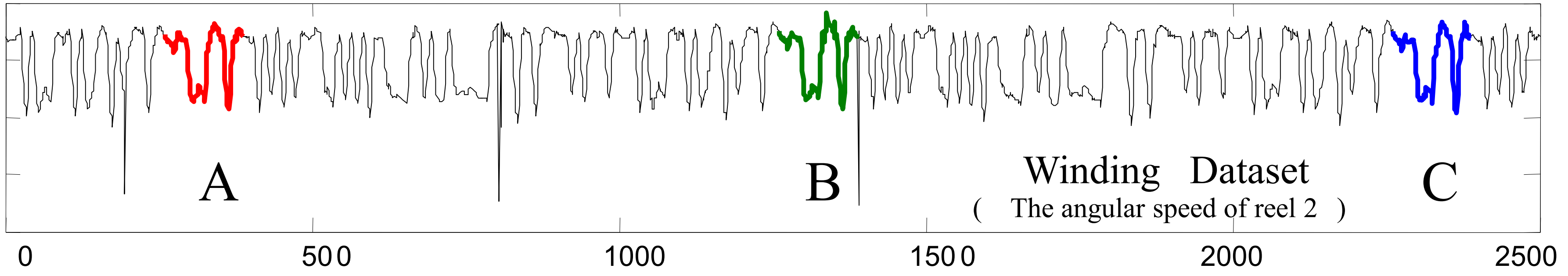
Astrophysics (Photon Count)



Motifs Discovery Challenges

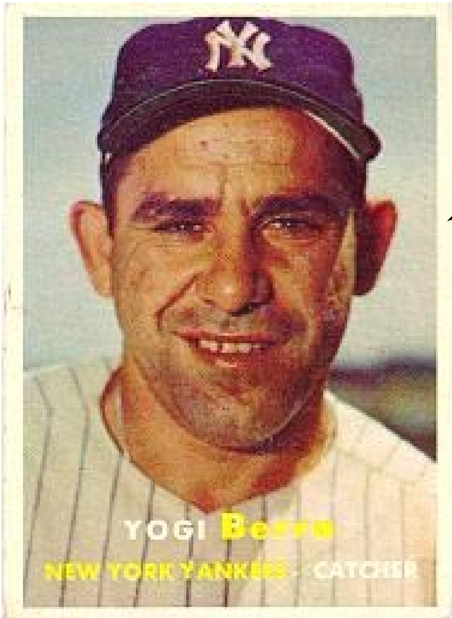
How can we find motifs…

- 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)