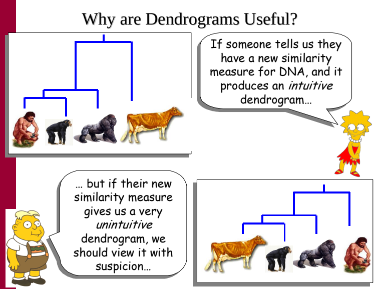
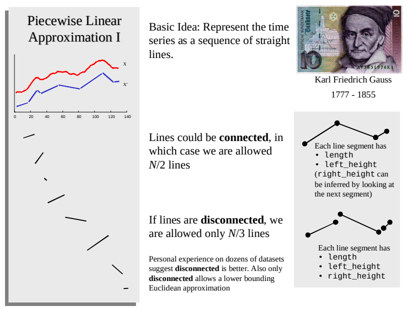
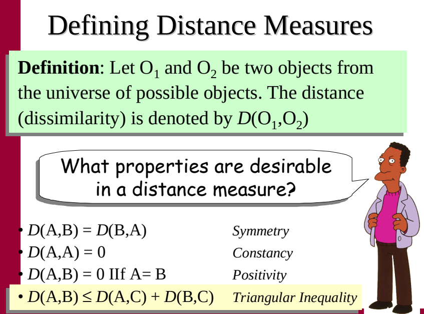




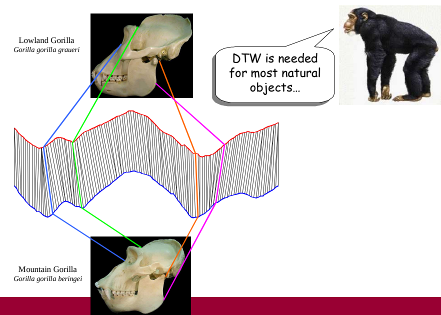
Indexování a dobývání znalostí z časových řad (*time series*)

Eamonn Keogh

eamonn@cs.ucr.edu



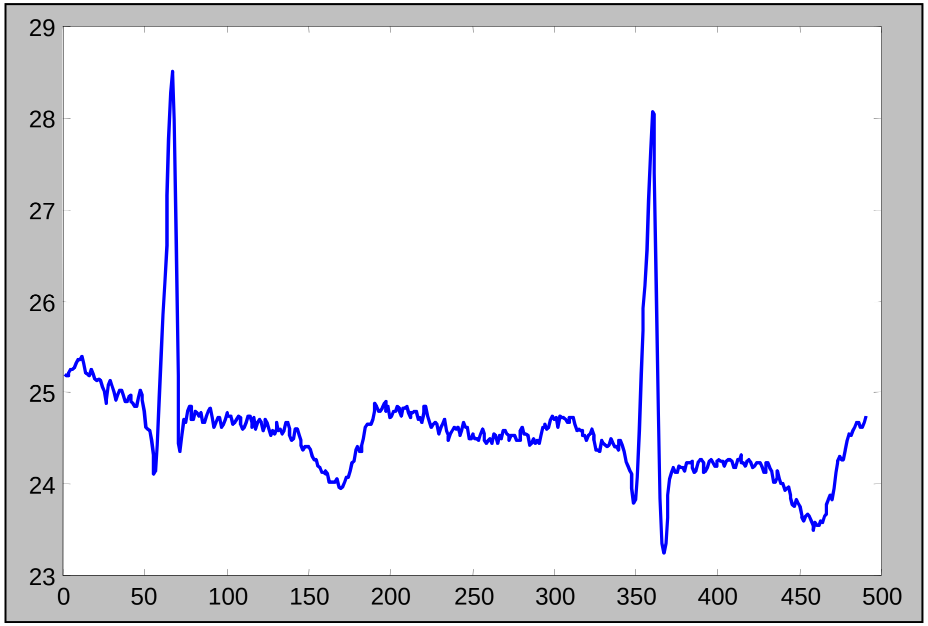
- Tato značka na slídu upozorňuje na doplňkoveé informace (tato část látky není povinná!)



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

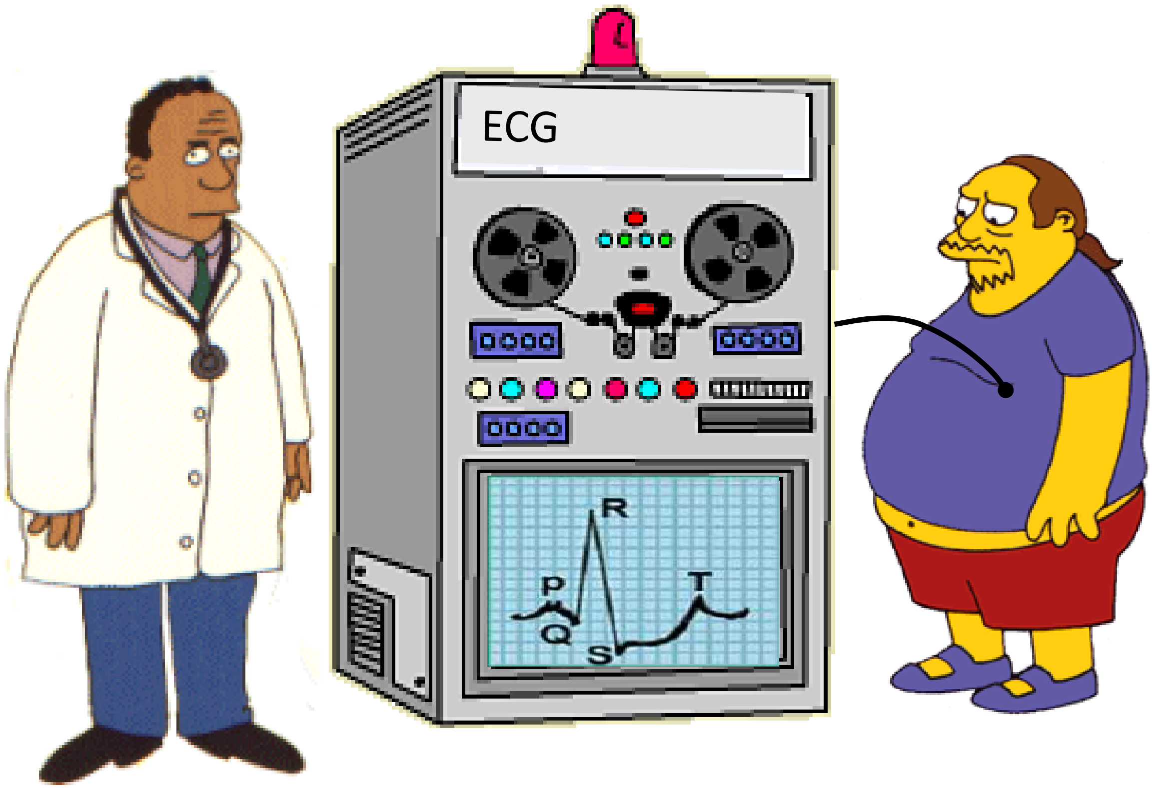
Co jsou časové řady?

Časové řady jsou posloupnosti dat (vektorů dat), které jsou výsledkem pozorování/měření uskutečněných postupně v čase.



Většinu technik (míry podobnosti, indexování, postupy pro redukci dimenzi-onality), se kterými se setkáme v této přednášce, lze použít i pro další typy dat.

Motivační příklad …



Pacient si stěžuje na bolest na prsou. Jeho EKG vypadá neobvykle …

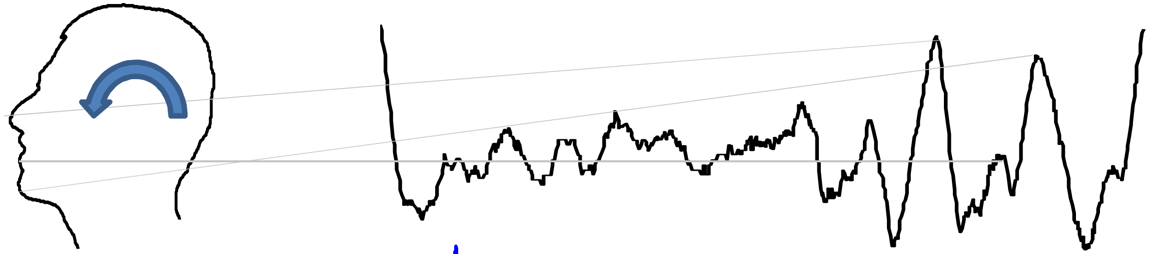
Lékař by rád nalezl v archivu nějaký podobný případ, jehož EKG vypadá **podobně**. Doufá totiž, že to by ho mohlo varovat před možnou chybou, inspirovat při návrhu řešení ...

2 otázky:

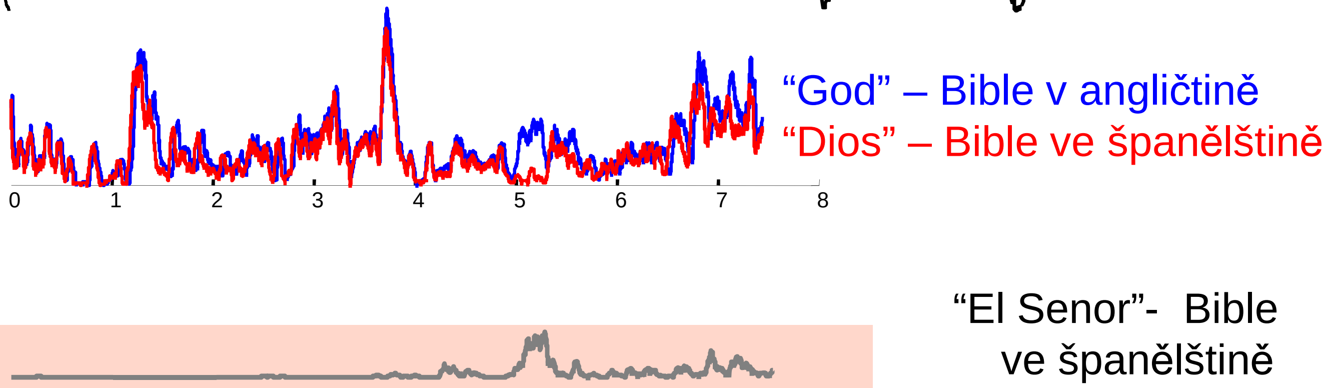
- **Jak poznat podobné křivky? – definice míry vzdálenosti**
- **Jak realizovat vyhledávání RYCHLE?**

Jaké další typy dat lze zpracovávat podobně?

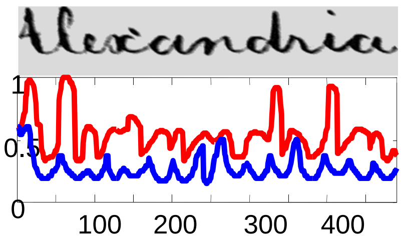
a) Obrázky, videa, …



b) Texty – počet výskytů slov na stránkach

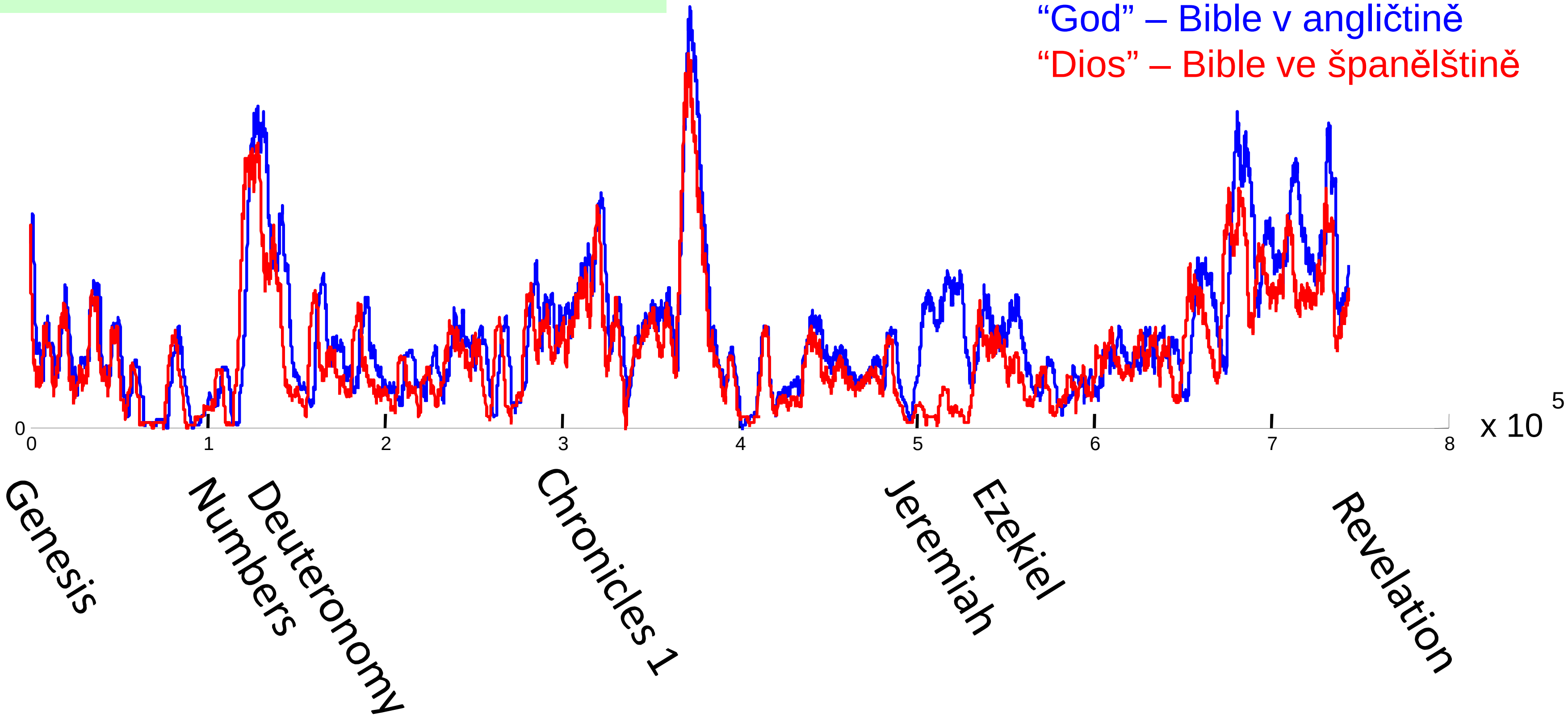


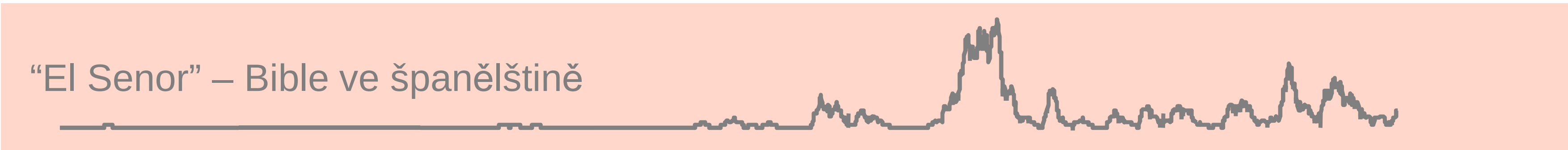
c) rukopis (roztaženo v čase)



Textová data jako časové řady…

Lokální frekvence slov (po stránkách)



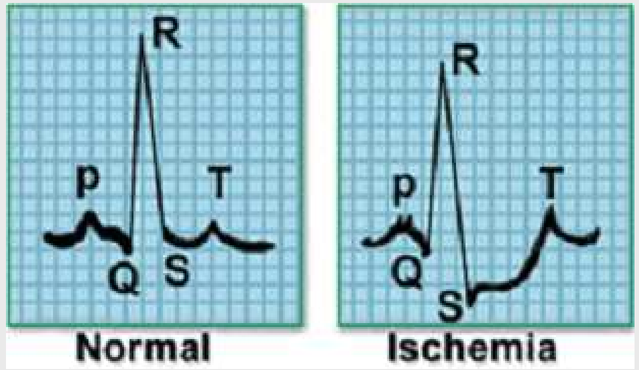


Jaké **typy úloh** jsou pro časové řady důležité?

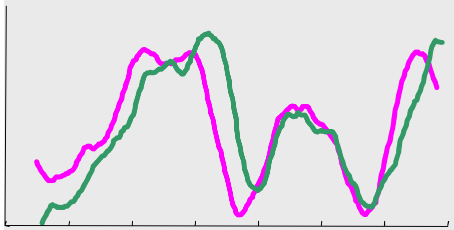
Ve všech potřebujeme charakterizaci ***podobnosti* (Simmilarity)**

Shlukování

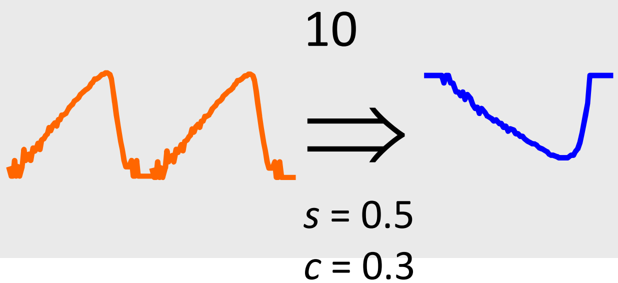
Klasifikace



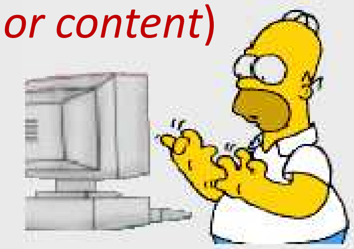
Hledání motivu

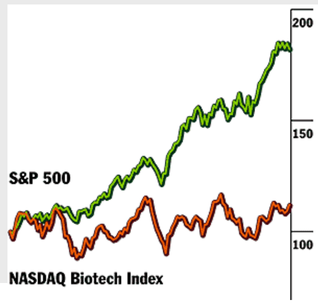


Hledání pravidel

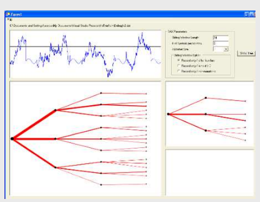


Dotaz příkladem (*query by example or content*)





Visualizace



Hledaní něčeho neobvyklého

Co je těžkého na práci s časovými řadami? Část I

1. **Velký objem dat** → vysoké požadavky na efektivní reprezentaci i implementaci používaných algoritmů (např. tak, aby se minimalizovaly přístupy na externí paměť).

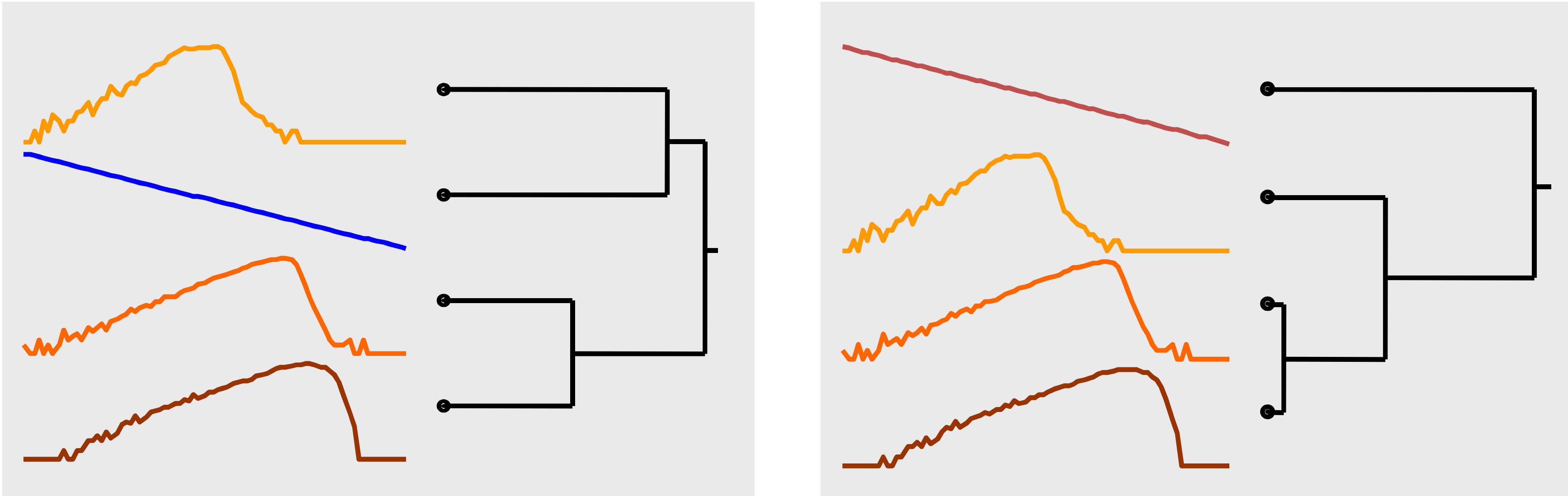
- 1 hodina dat z EKG: 1 Gigabyte.
- Typický Weblog: 5 Gigabytes/týden.
- Space Shuttle Database: 200 Gigabytes ….

2. Problémy spojené se **slučováním dat z více zdrojů (merge data from various sources)**:

- Různé formáty.
- Rozdílné vzorkovací frekvence.
- Šum, chybějící hodnoty, ….

Co je těžkého na práci s časovými řadami? Část II

Odpověď: Hodnocení podobnosti je subjektivní!



Hodnocení podobnosti závisí na uživateli, na zdroji dat i na řešené úloze. Tyto aspekty je nutné respektovat při definici odpovídající míry podobnosti.

Definice míry vzdálenosti (*Distance Measures*)

Definice: Nechť O_1 a O_2 jsou dva objekty z množiny všech možných objektů (univerzum). Jejich vzájemná vzdálenost (nepodobnost - *dissimilarity*) se označuje $\boldsymbol{D(O_1,O_2)}$

Jaké vlastnosti by měla mít míra vzdálenosti (nepodobnosti)? Má být

- $D(A,B) = D(B,A)$ *symetrická*
- $D(A,A) = 0$ *konstantní*
- $D(A,B) = 0$ iff $A= B$ *kladná*
- $D(A,B) \leq D(A,C) + D(B,C)$ *splňovat trojúhelníkovou nerovnost*

Proč požadujeme trojúhelníhovou nerovnost?

Požadují ji prakticky všechny metody na indexování dat.
Důvod? Předpokladejme, že k danému bodu **Q** máme vybrat mezi 3 body **a, b** a **c** ten, který je od **Q** nejméně vzdálen.

Nejdřív zjistíme, že bod **a** je od **Q** vzdálen **2** jednotky: **a** se stane *zatím-nejlepší*. Pro další bod **b** vypočteme vzdálenost od **Q** **7.81** jd.

Teď už nemusíme zjišťovat vzdálenost mezi Q a **c** – stačí využít trojúhelníkovou nerovnost !

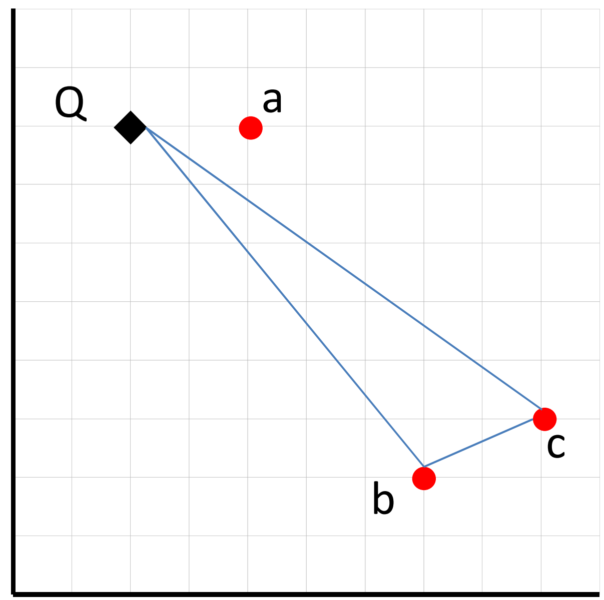
Víme, že $D(\mathrm{Q},\mathbf{b}) \leq D(\mathrm{Q},\mathbf{c}) + D(\mathbf{b},\mathbf{c})$

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

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

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

Dolní odhad pro vzdálenost mezi **Q** a **c** je 5.51 jd. Není tedy lepší než náš *zatím-nejlepší* bod **a**.



	a	b	c
a		6.70	7.07
b			2.30
c			

Eucleidova míra vzdálensoti

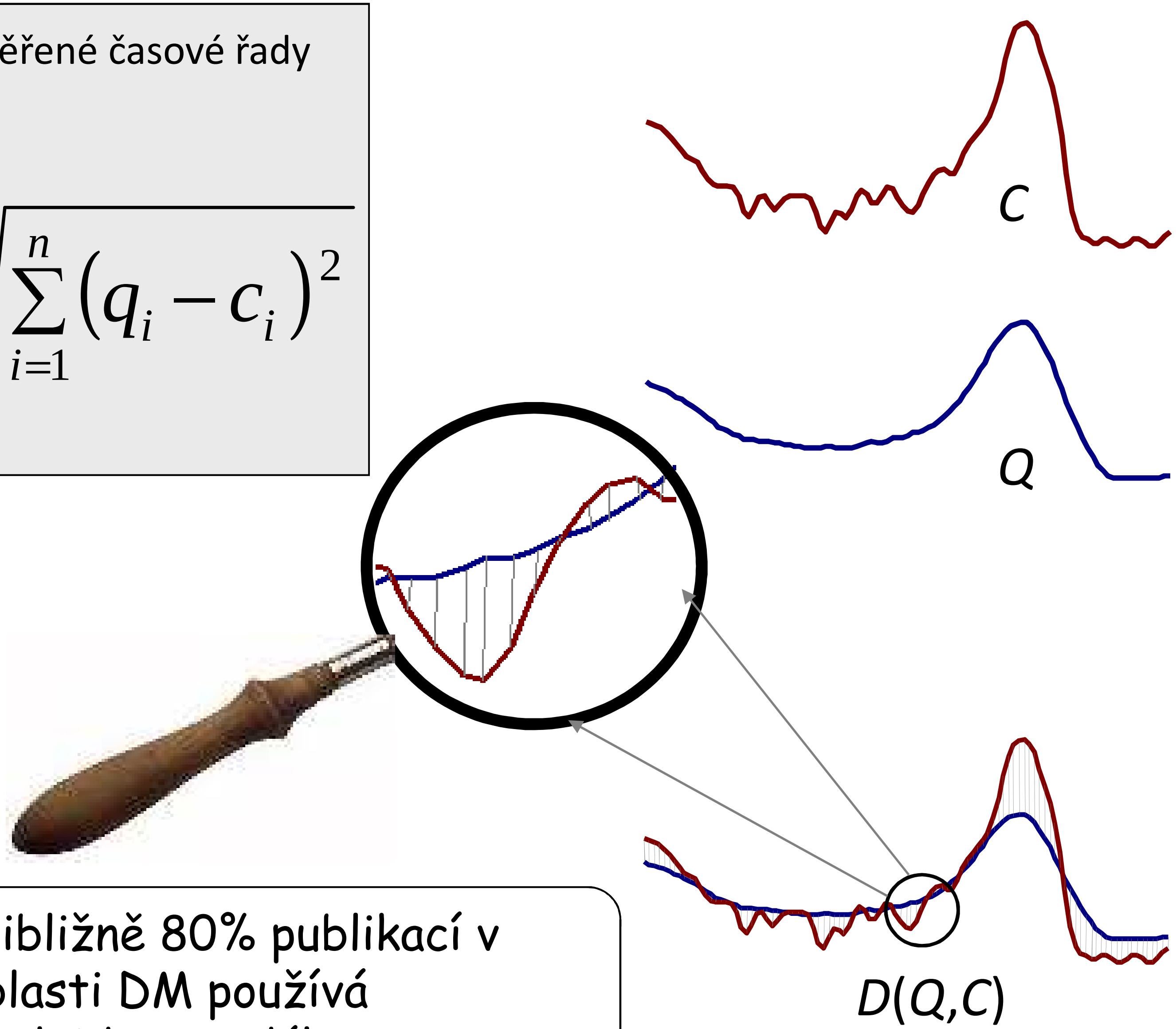
Je pro dvě synchronně měřené časové řady

$Q = q_1 \ldots q_n$

$C = c_1 \ldots c_n$

definována takto:

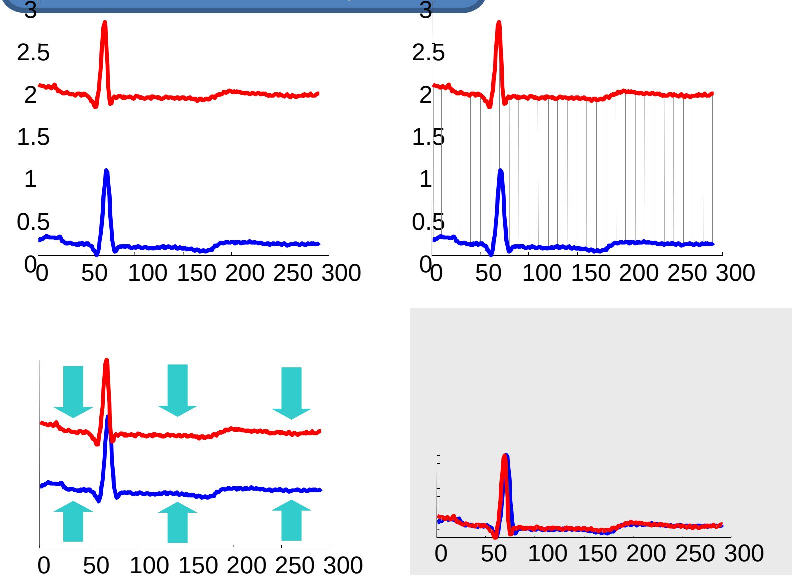
$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$



Předzpracování pomocí lineární transformace

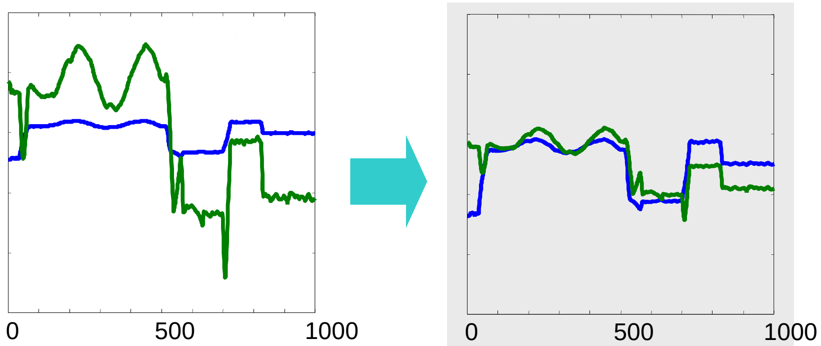
T_I: vyvážení (*Offset Translation*)

$Q = Q$ - průměr(Q)



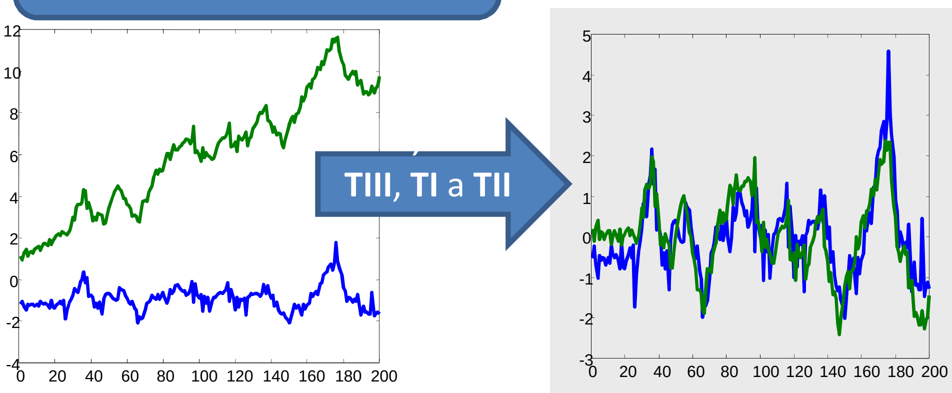
T_{II}: úprava amplitud (Amplitude Scaling)

$Q = (Q$ - průměr(Q)) / st_odchylka(Q)



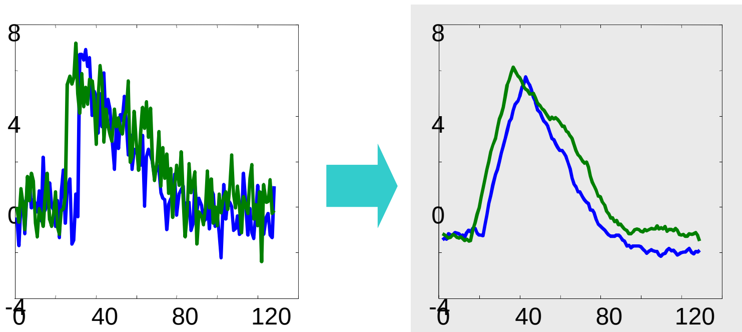
T_{III}: odstranění lin. trendu

Původní signál ***s*1** je proložen přímkou ***l*1** a výsledným signálem se stane jejich rozdíl (***s*1 – *l*1**) .

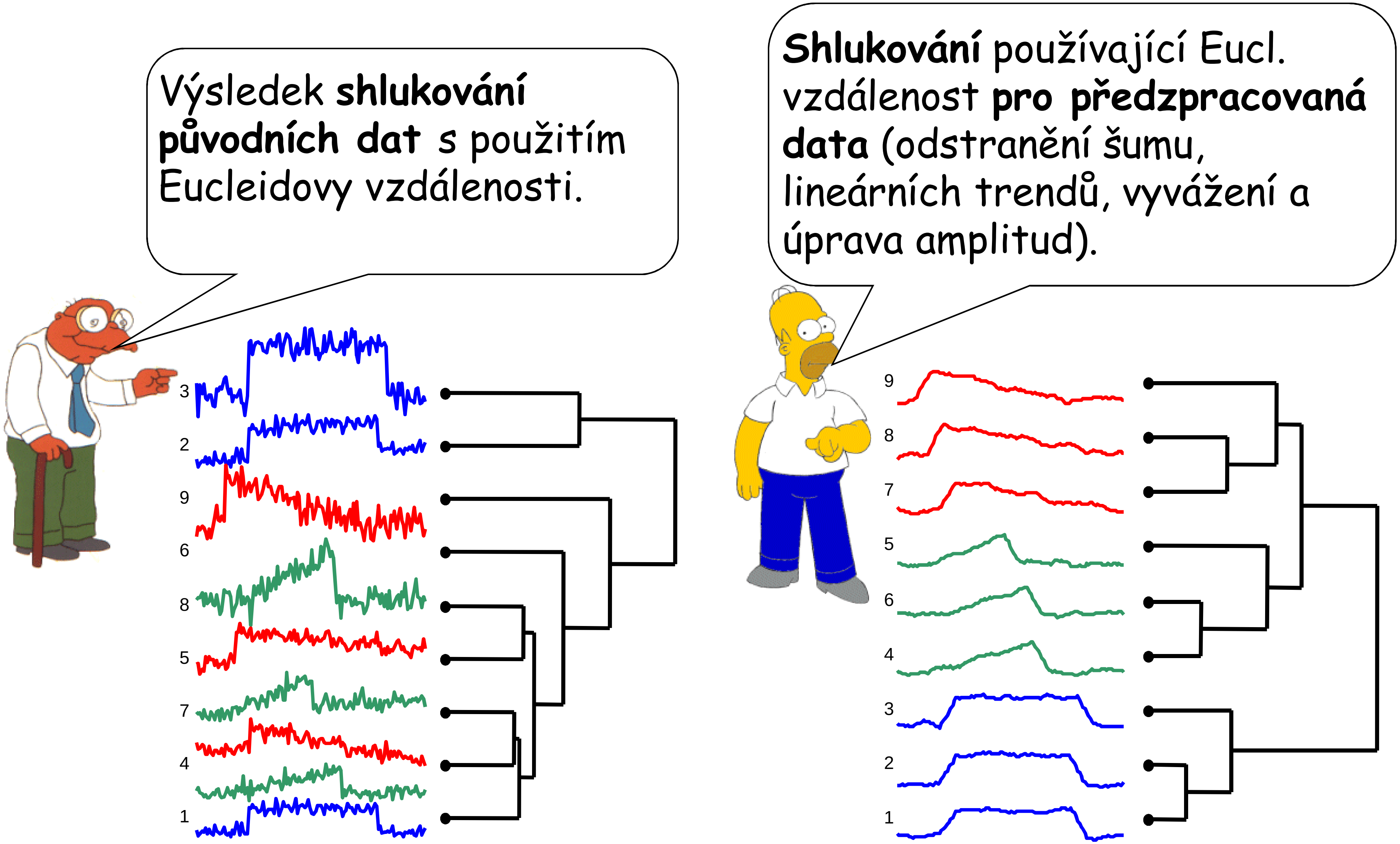


T_{IV}: odstranění šumu

Vyhlazení (*smoothing*): Každá hodnota je nahrazena průměrem okolních hodnot.

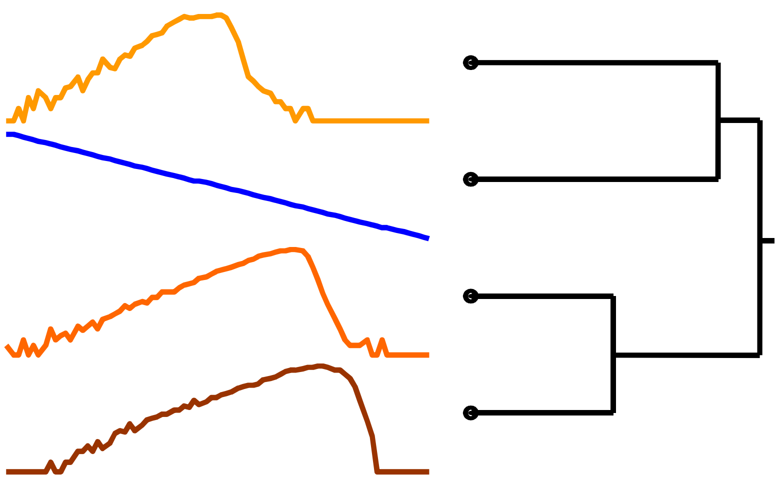


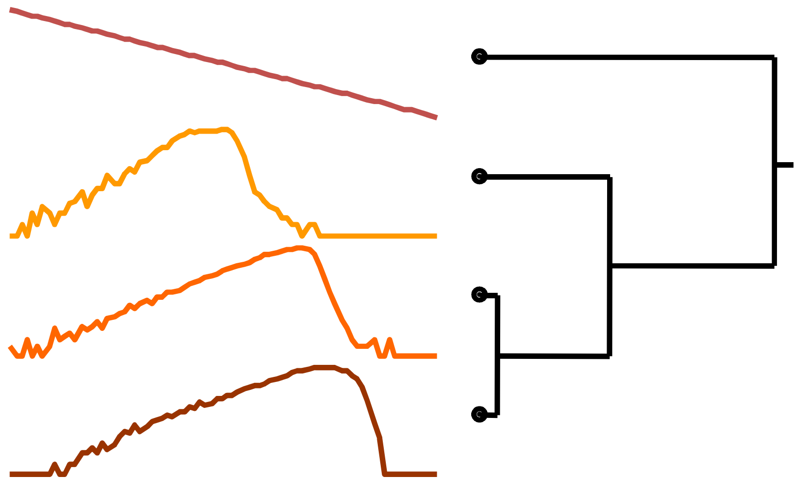
Rychlý experiment demonstrující **užitečnost předzpracování**

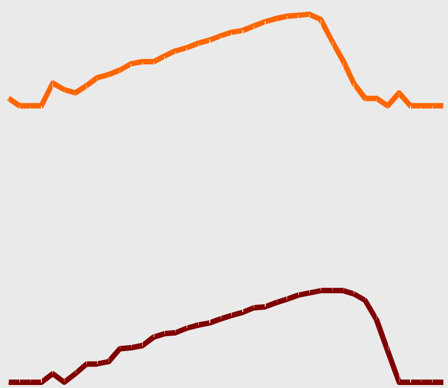


Dynamic Time Warping

Dynamické borcení času (**DTW**)

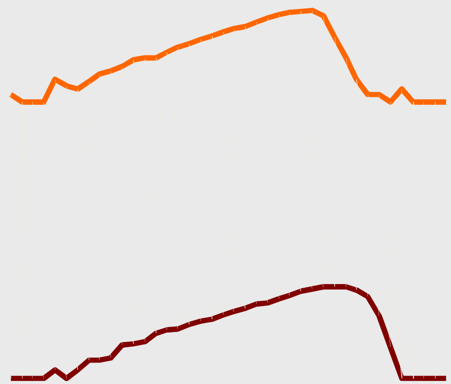






Pevná časová osa (*Fixed Time Axis*)

Předpokládá, že posloupnosti jsou srovnávány „bod po bodu“ ve stejných intervalech.

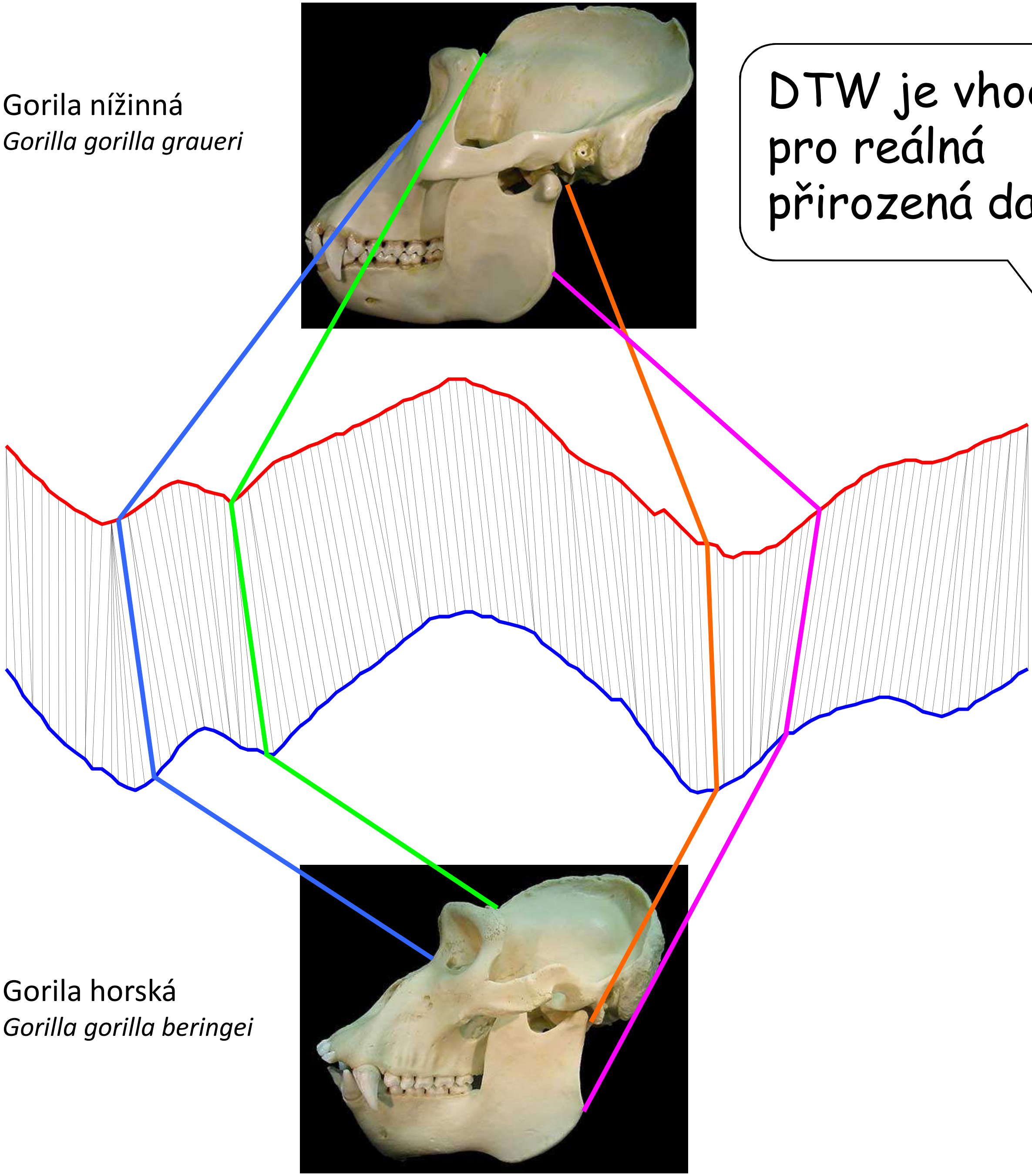


“Zborcená” časová osa

Umožňuje nelineární skoky při srovnávání bodů dvou signálů.

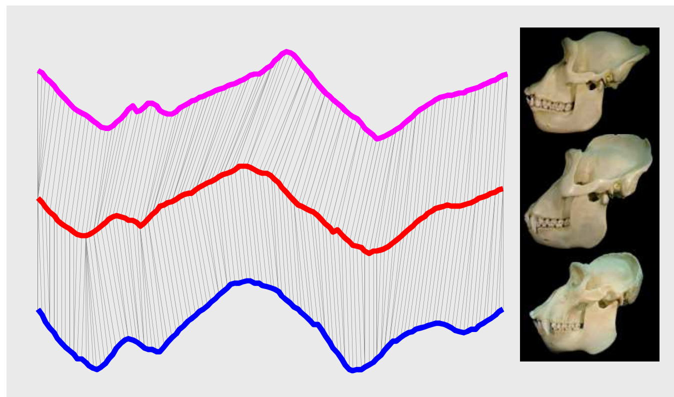
Poznámka: Nejdřív ukážeme pár příkladů, pak se vrátíme k metodě výpočtu.

Gorila nížinná
Gorilla gorilla graueri



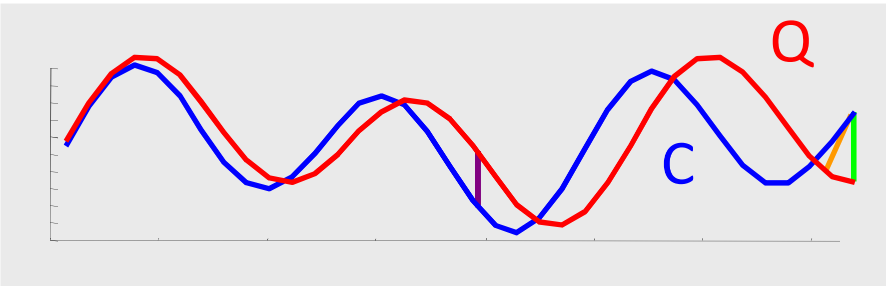
Gorila horská
Gorilla gorilla beringei

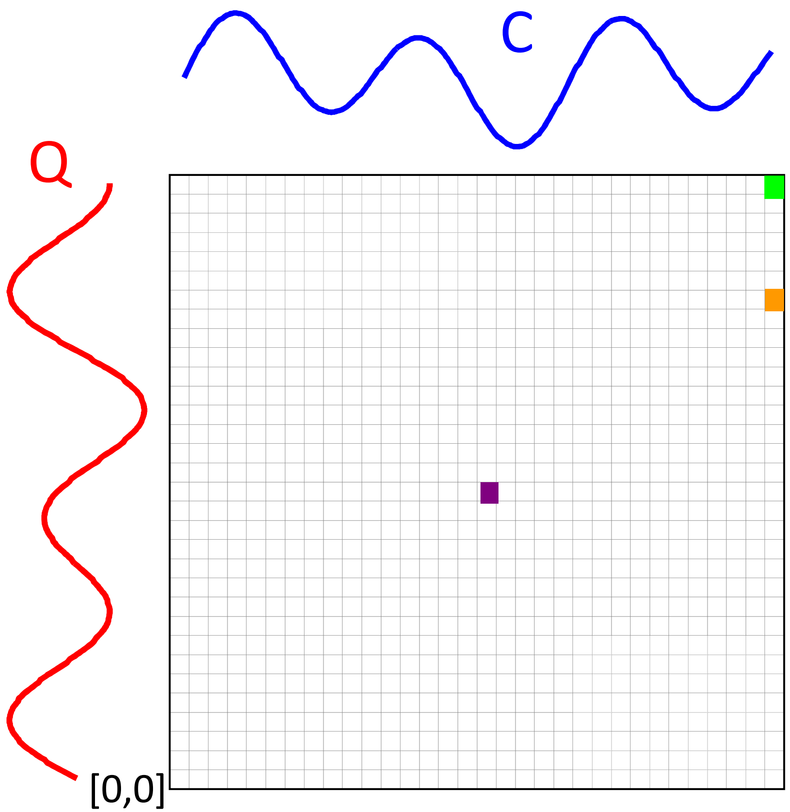
DTW je vhodné pro reálná přirozená data...



Postup výpočtu DTW I

Vytvoříme matici **|Q|** krát **|C|**, všech hodnot vzdáleností mezi indexovanými body obou křivek a hledáme "nejlevnější" cestu z [0,0] do [*n*,*n*] .





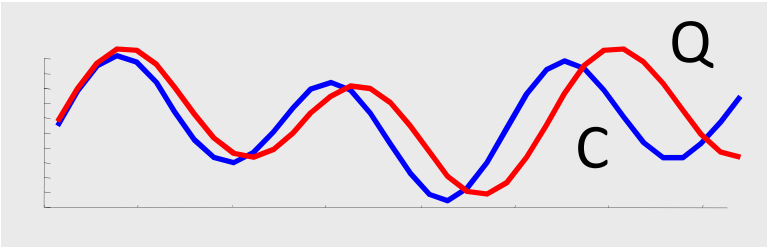
Jaký je odhad složitosti výpočtu
(počet možných cest WPs) v matici
n x ***n*** **?**

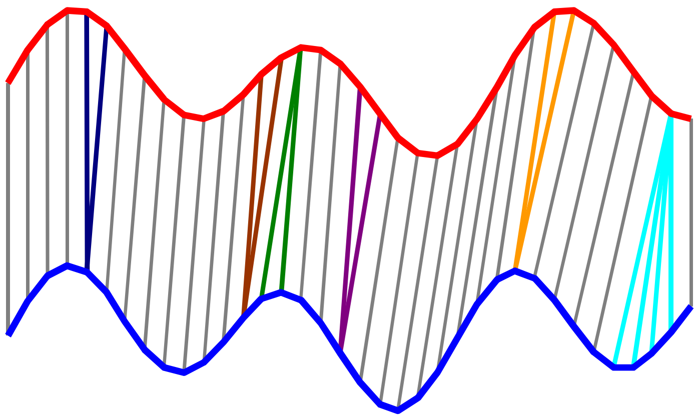
Hrubý horní odhad je $\mathbf{3}^{n}$

Postup výpočtu DTW II.

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

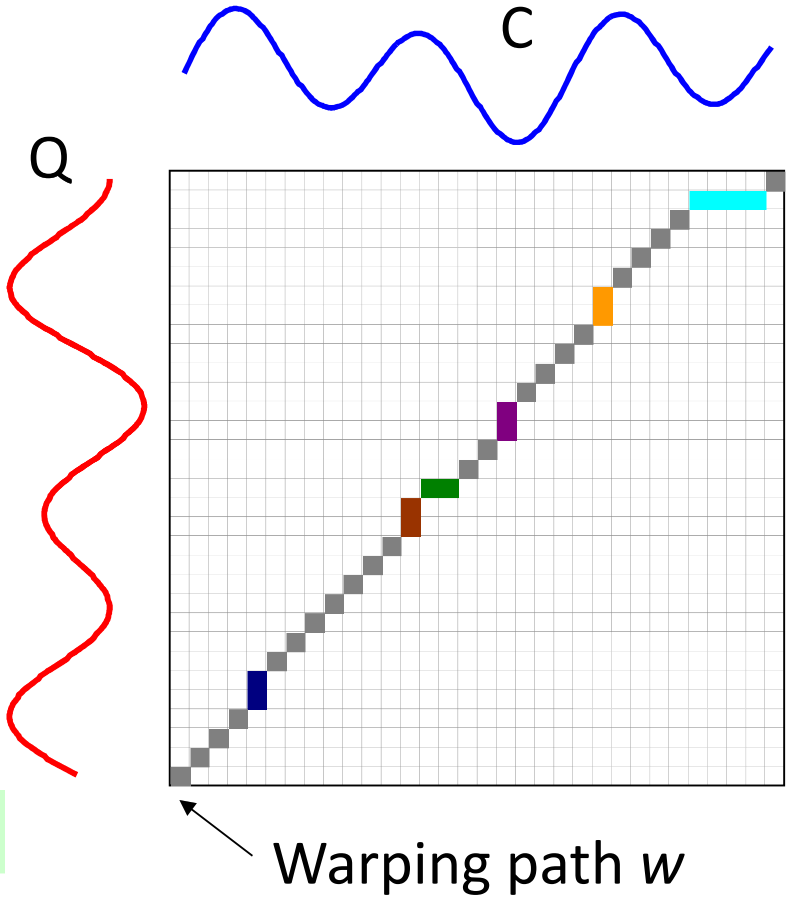
$$DTW(Q,C) = \min\left\{ \sqrt{\sum_{k=1}^{K} w_k} \Big/ K \right.$$



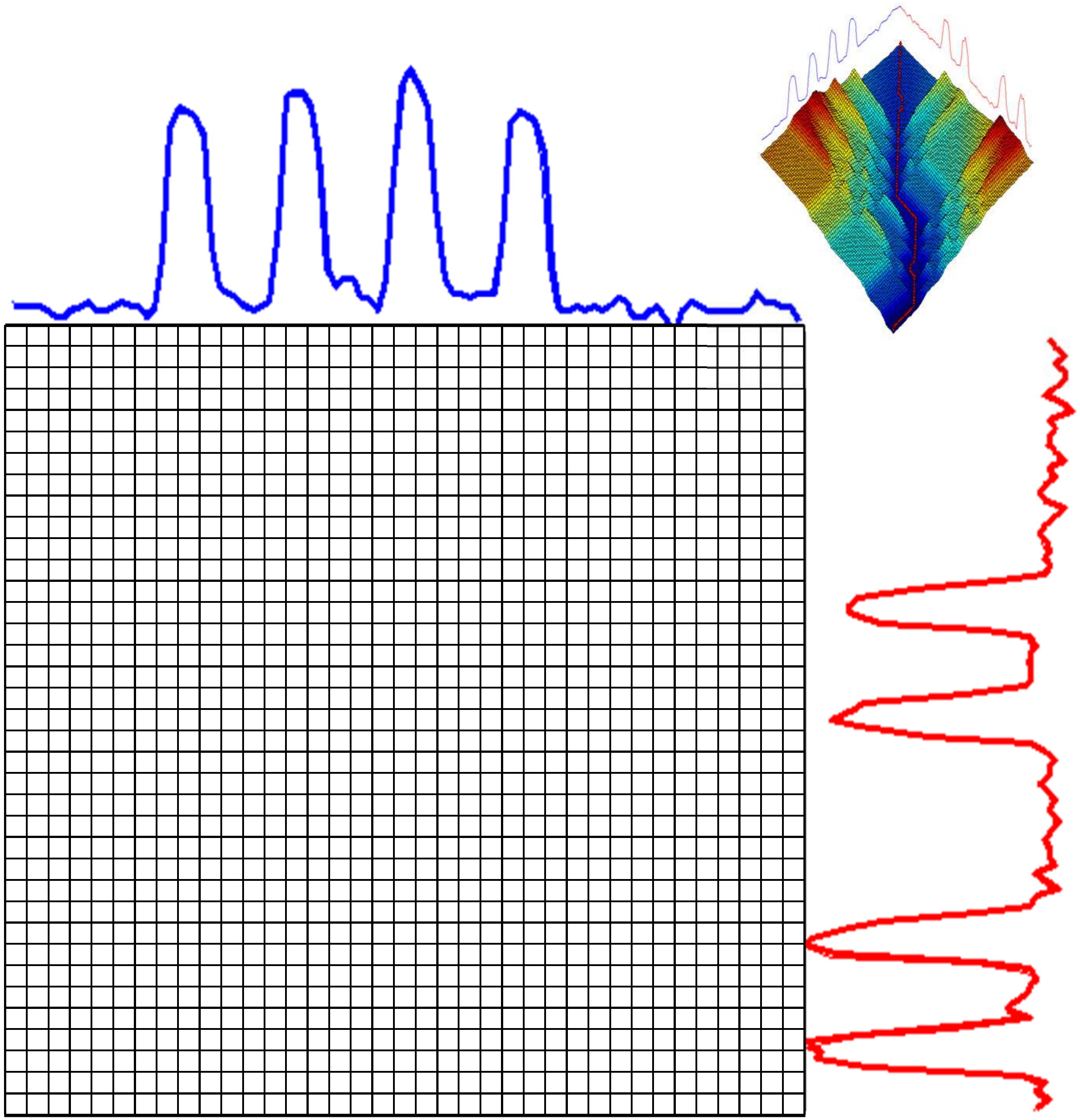


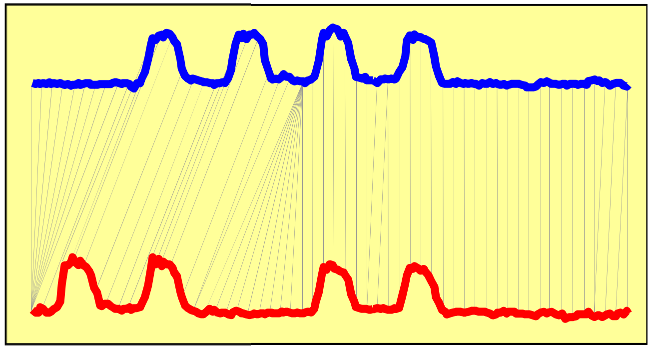
This recursive function gives us the minimum cost path

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



* Example: visualize the process on a real world problem I

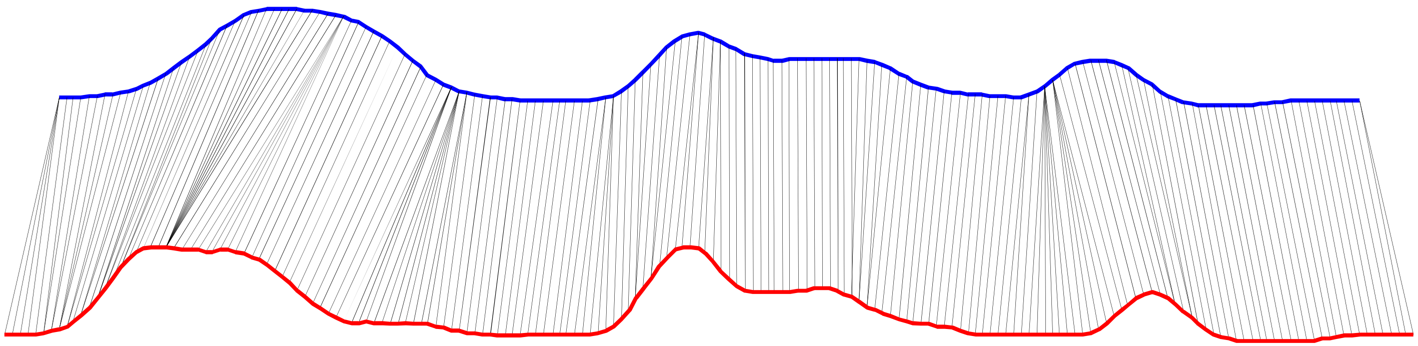




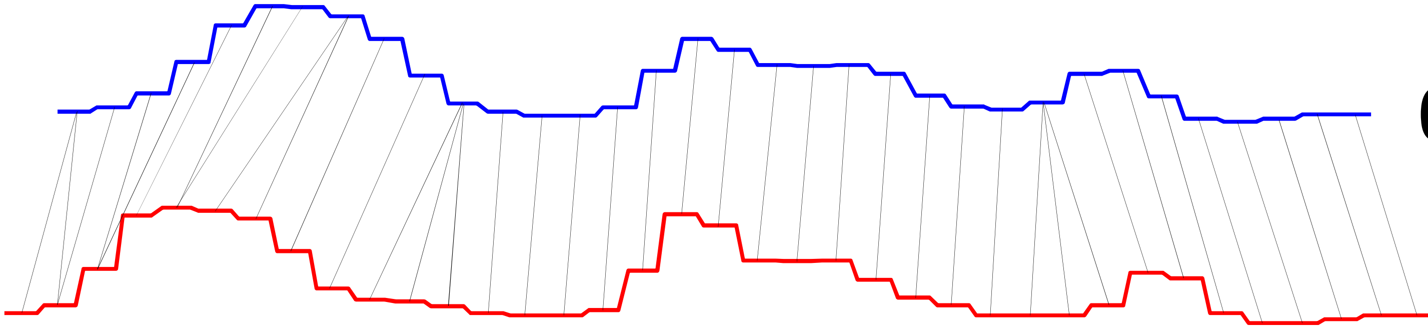
This example shows 2 one-week periods from the power demand time series.

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.

Are **Fast Approximations** to Dynamic Time Warp Distance Usefull?



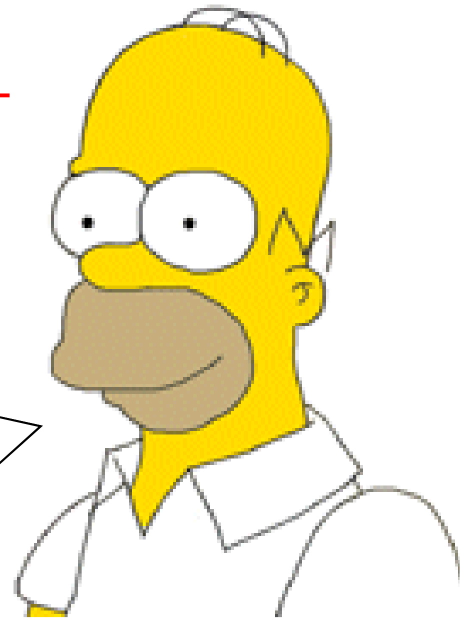
1.03 sec



0.07 sec

… there is strong visual evidence to suggests it works well

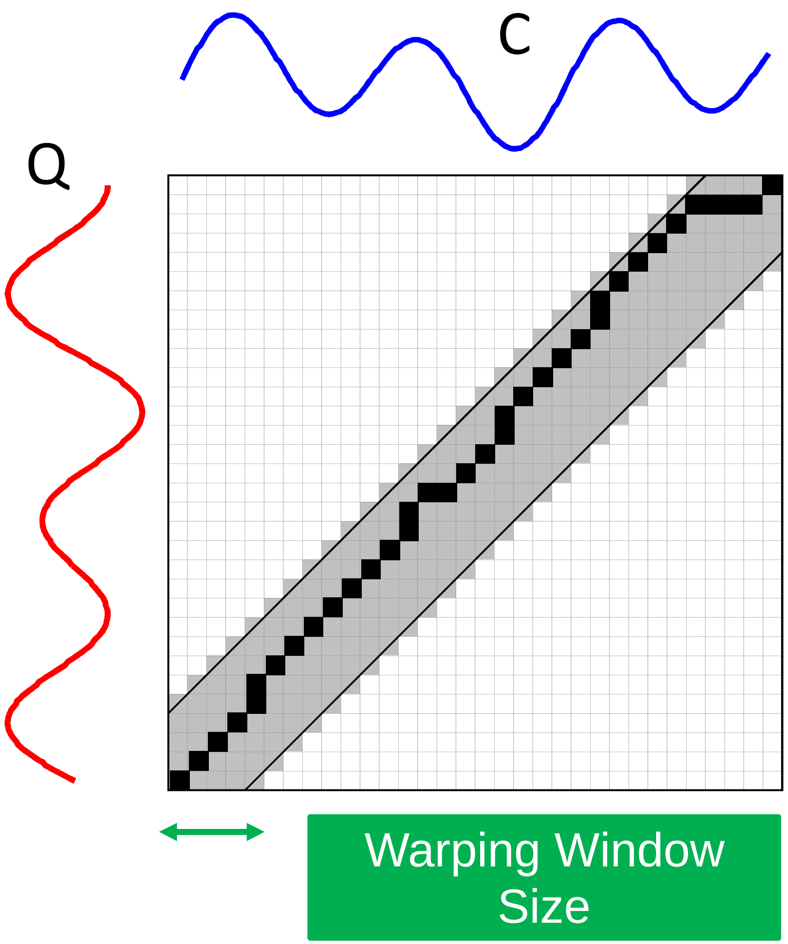
There is good experimental evidence for the utility of the approach on clustering, classification, etc



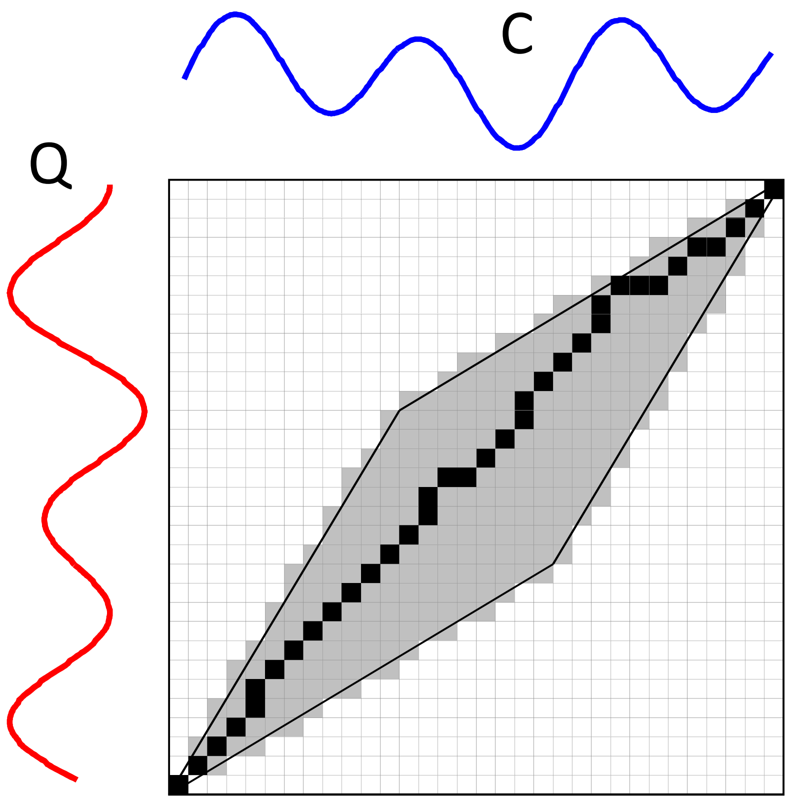
Global Constraints

- Slightly speed up the calculations
- Prevent pathological warpings

Sakoe-Chiba Band



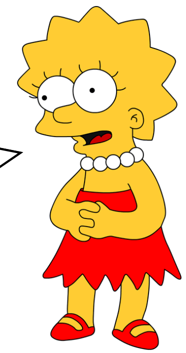
Itakura Parallelogram



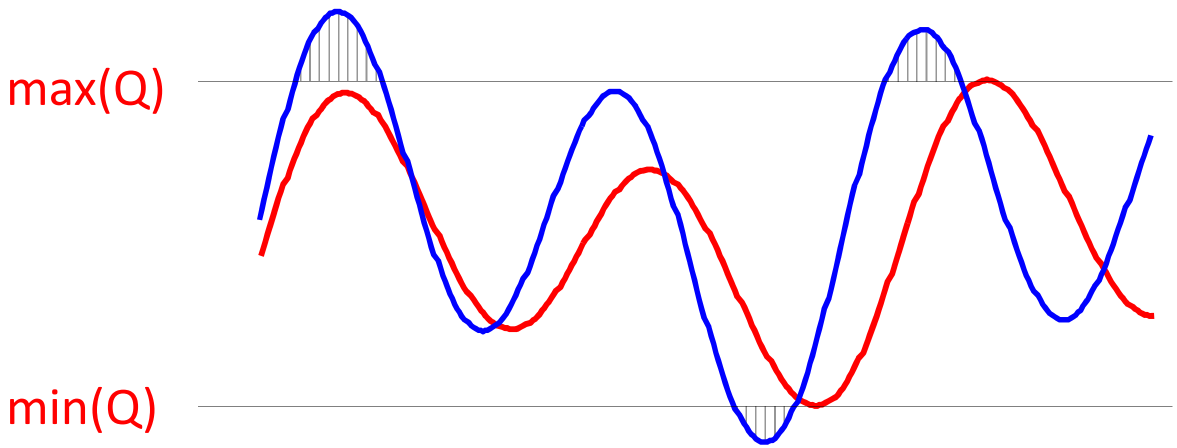
How to speed up calculation of distance?
Lower Bounding

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

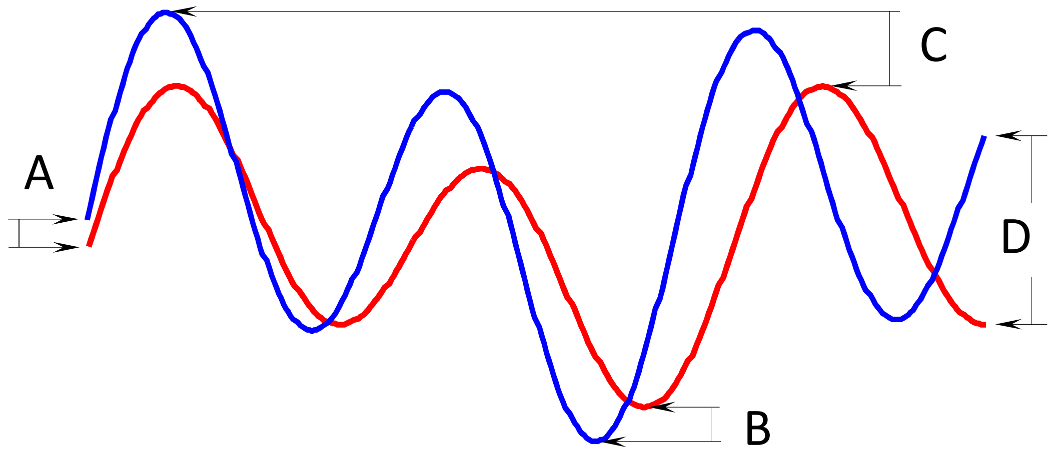


LB_Yi

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

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

* Lower Bound of Kim

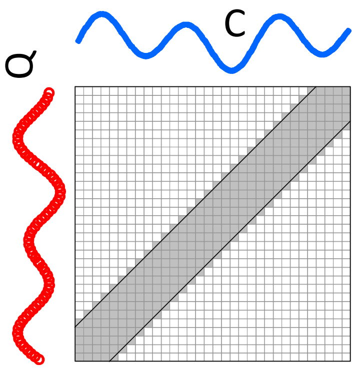


LB_Kim

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

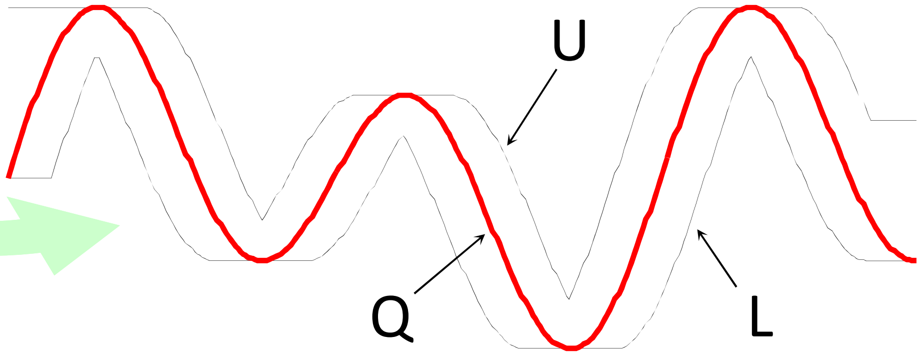
The squared difference between the two sequence's first (A), last (D), **minimum** (B) and **maximum points** (C) is returned as the lower bound

* Lower Bound of Keogh



Sakoe-Chiba Band

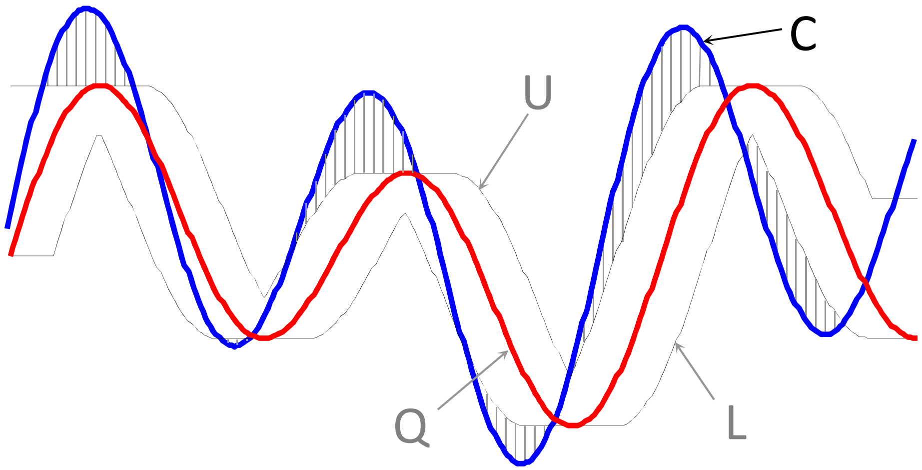
$U_i = \max(q_{i-r} : q_{i+r})$
$L_i \ = \min(q_{i-r} : q_{i+r})$



LB_Keogh

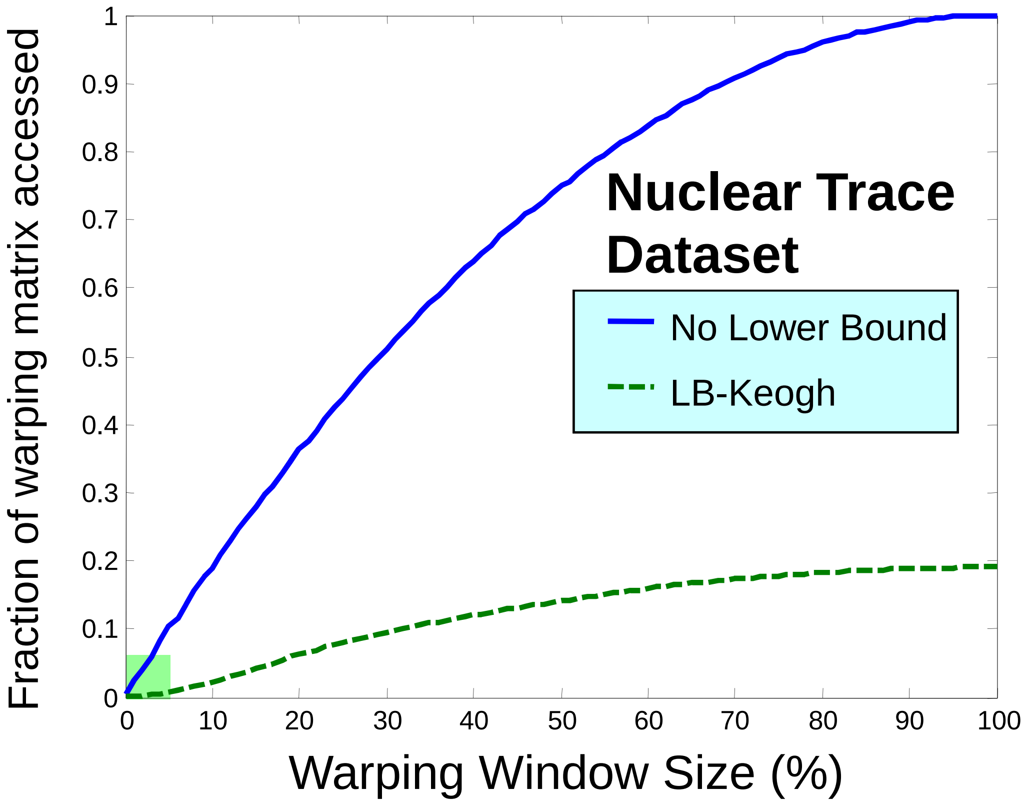
Envelope-Based Lower Bound

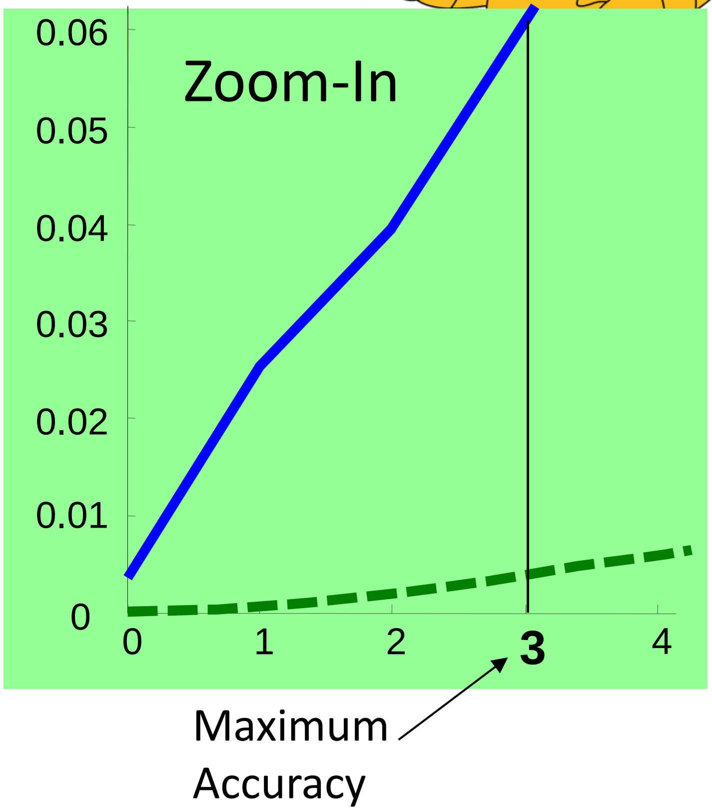
$$LB\_Keogh(Q,C) = \sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2 & if\ q_i > U_i \\ (q_i - L_i)^2 & if\ q_i < L_i \\ 0 & otherwise \end{cases}$$

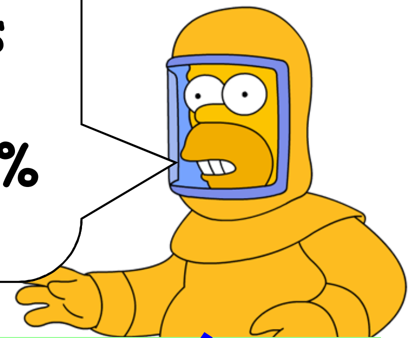


* How Useful are Lower Bounds?

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!





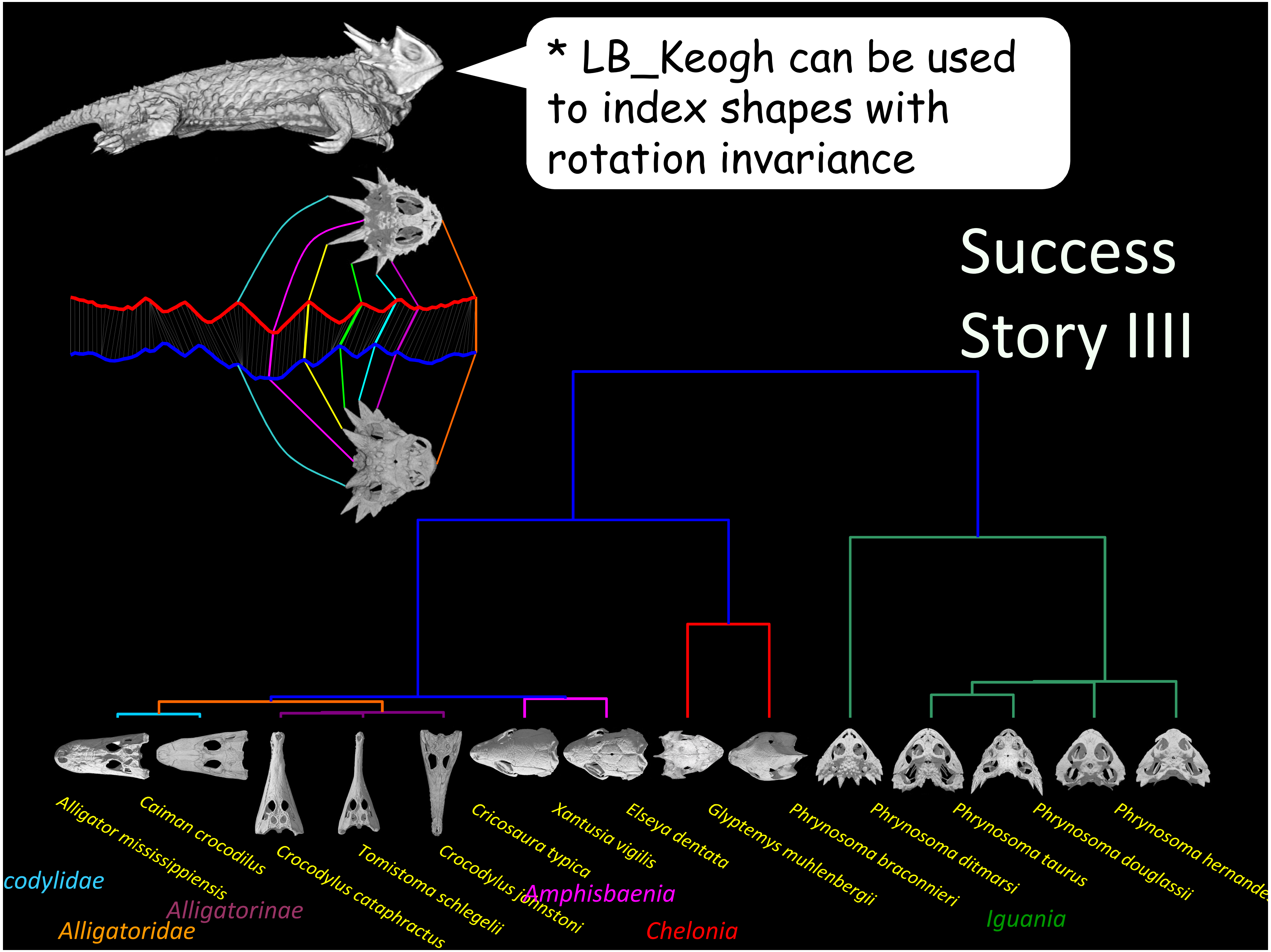


* Frequent myths about DTW …

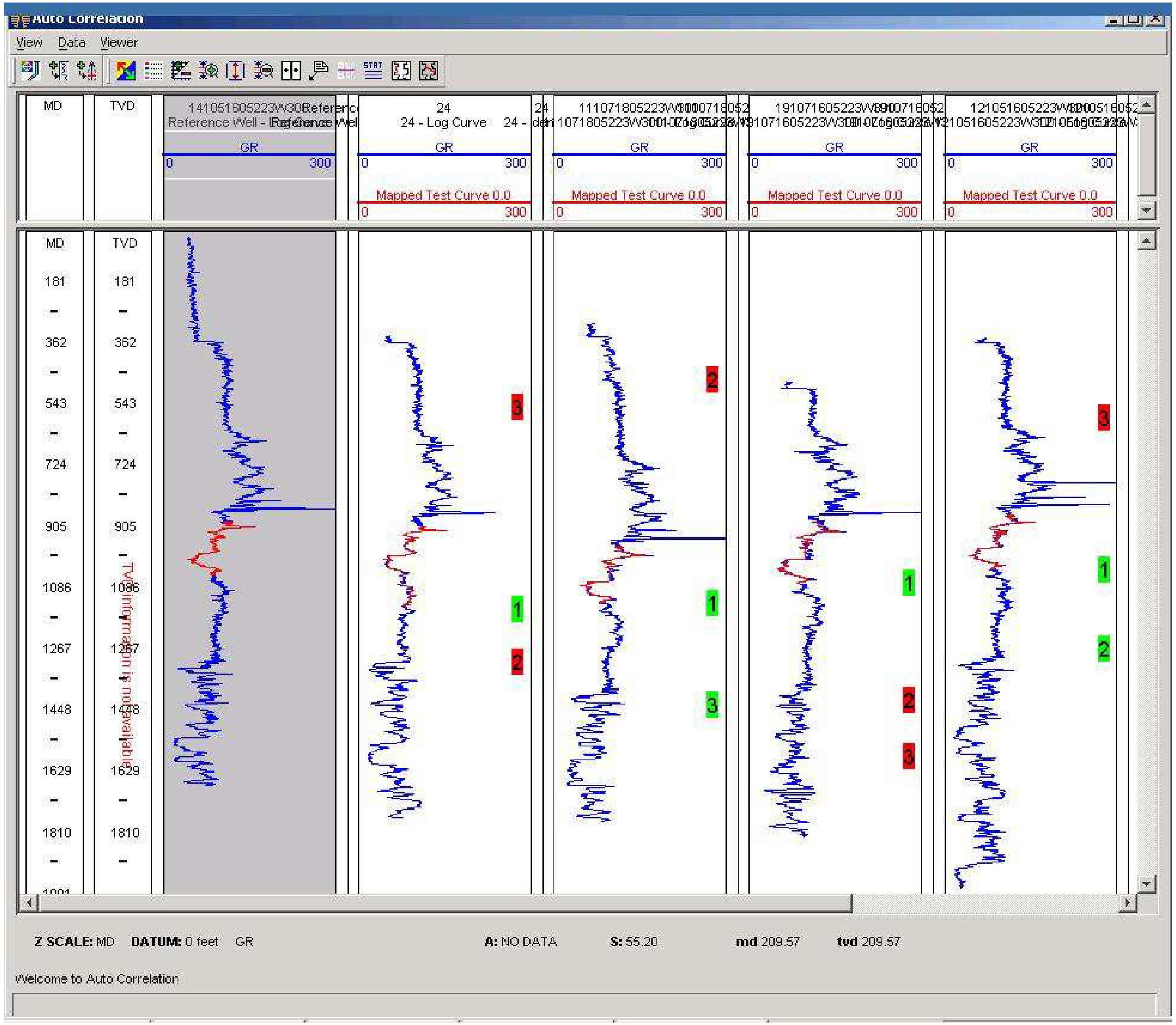
- *"DTW incurs a heavy CPU cost"*[1]
- *"DTW is limited to only small time series datasets"*[2]
- *"(DTW) quadratic cost makes its application on databases of long time series very expensive"*[3]
- *"(DTW suffers from) serious performance degradation in large databases"*[4]

This is simply not true!

…DTW can be close to linear for data mining problems!


* LB_Keogh can be used to index shapes with rotation invariance
Success
Story IIII
Alligator mississippiensis
Caiman crocodilus
Crocodylus cataphractus
Tomistoma schlegelii
Crocodylus johnstoni
Cricosaura typica
Xantusia vigilis
Elseya dentata
Glyptemys muhlenbergii
Phrynosoma braconnieri
Phrynosoma ditmarsi
Phrynosoma taurus
Phrynosoma douglassii
Phrynosoma hernande
codylidae
Alligatorinae
Alligatoridae
Amphisbaenia
Chelonia
Iguania

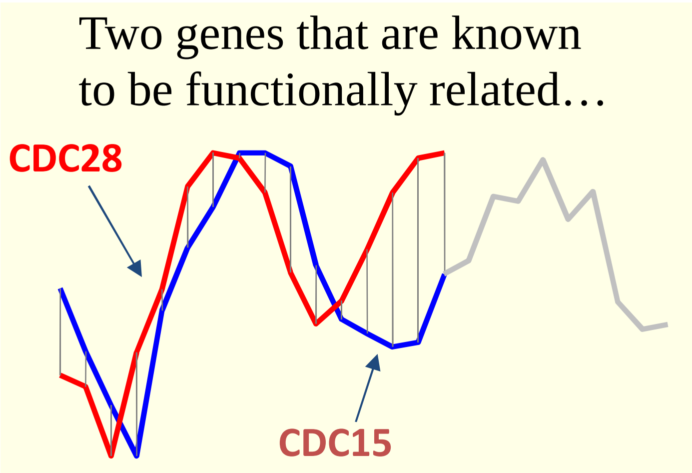
* Success Story

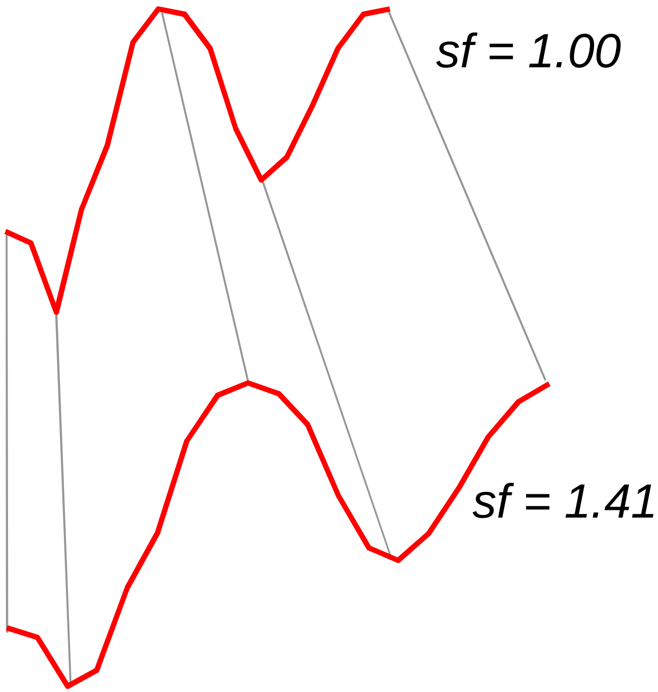


The lower bounding technique is being used by ChevronTexaco for comparing seismic data

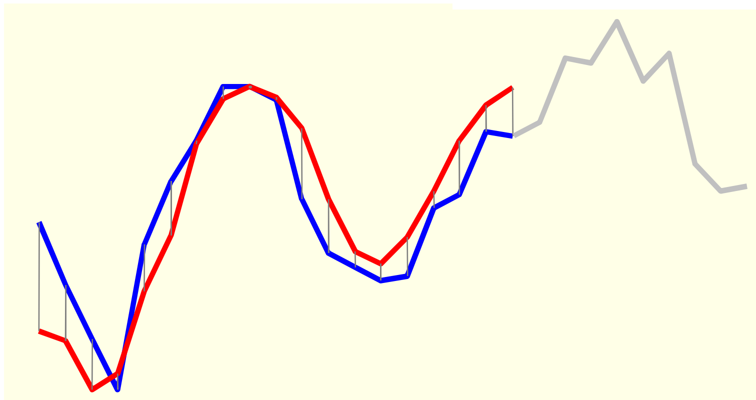
Thanks of Steve Zoraster for the figure

Uniform Scaling I



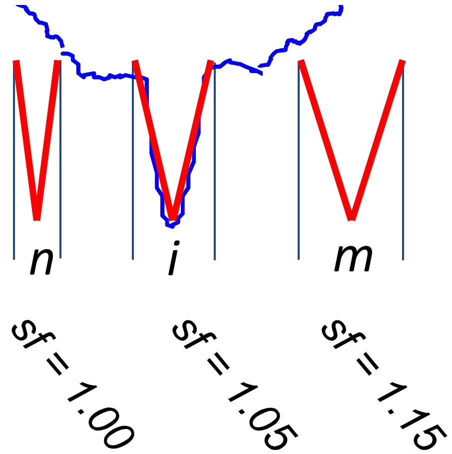


Sometimes global or *uniform scaling* is as important as DTW



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



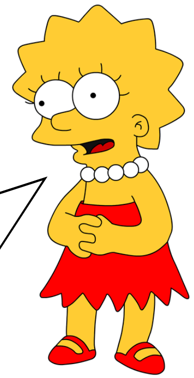

```
Algorithm: Test_All_Scalings(Q,C)
        best_match_val     = inf;
        best_scaling_factor       = null;
         for p = n to m
      QP = rescale(Q,p);
       distance = squared_Euclidean_distance
       if distance <  best_match_val
          best_match_val = distance;
          best_scaling_factor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

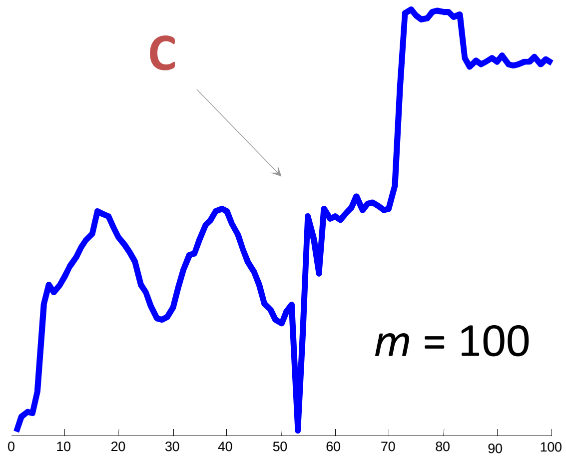
Here is the code to **Test_All_Scalings**, the time complexly is only $O((m-n) * n)$, but we may have to do this many times…

* Lower Bounding Revisited!

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

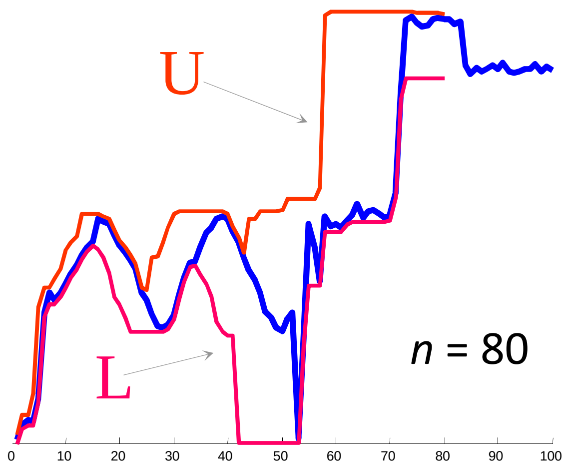
```
Algorithm: Lower_Bounding_Sequential_Scan(Q,C)
overall_best_time_series = null;
overall_best_match_val   = inf;
for i = 1 to number_of_time_series_in_(C)
  if  lower_bound_distance(Q,Ci) < overall_best_match_val
    [dist, scale] = Test_All_Scalings(Q,C_i)
     if dist <  overall_best_match_val
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

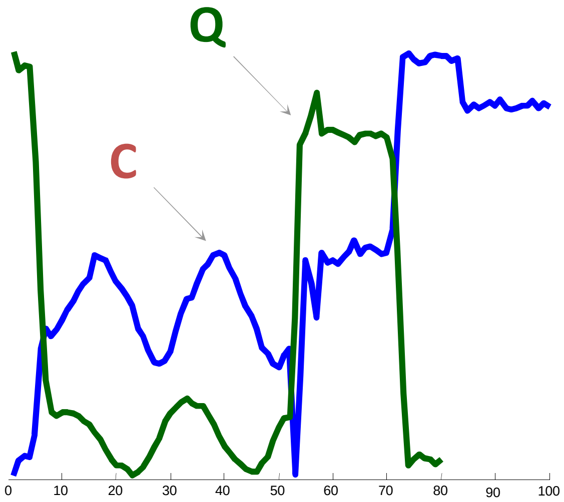


We can build envelopes around all candidates time series C_i, in our database, just like we did for DTW, except the definition of the envelopes is different.

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

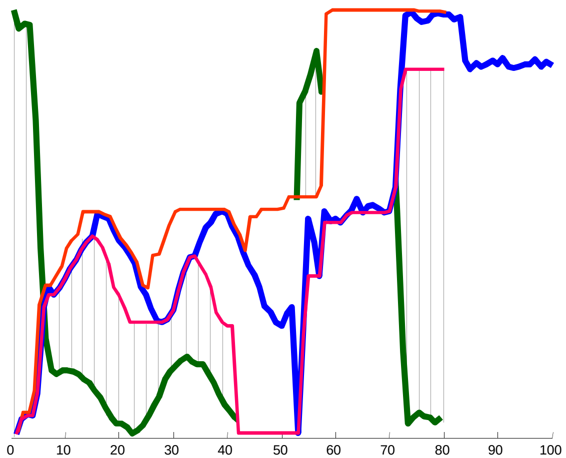
$$L_i = \min\left(c_{\lfloor (i-1)*m/n \rfloor + 1}, \ldots, c_{\lfloor i*m/n \rfloor} \right)$$

*



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!



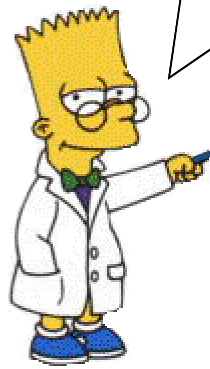
LB_Keogh

Envelope-Based Lower Bound

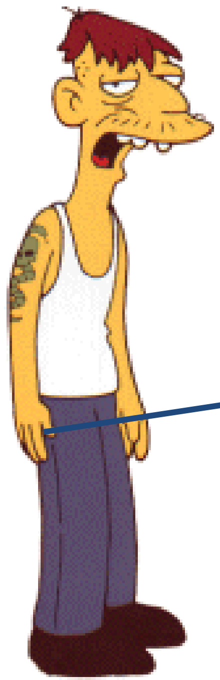
$$LB\_Keogh(Q,C) = \sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2 & if\ q_i > U_i \\ (q_i - L_i)^2 & if\ q_i < L_i \\ 0 & otherwise \end{cases}$$

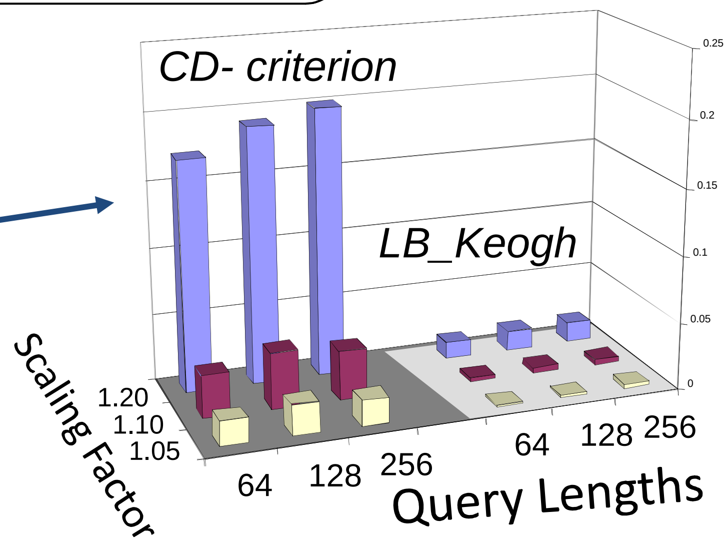
An experiment to test the utility of lower bounding uniform scaling, over different scaling factors (Y-axis) and scaling lengths (X-axis). The dataset was a “mixed bag” of 10,000 assorted time series.

This is the time taken by brute force search



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





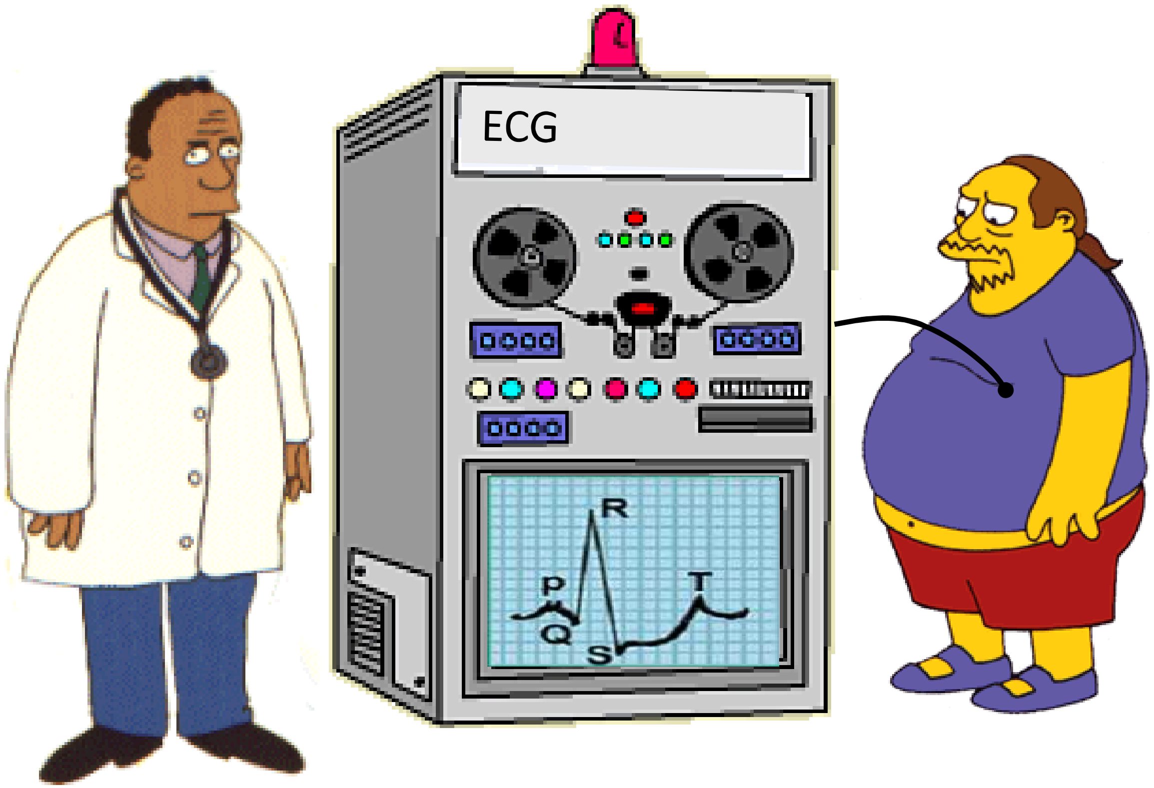
*

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

Motivating example revisited…



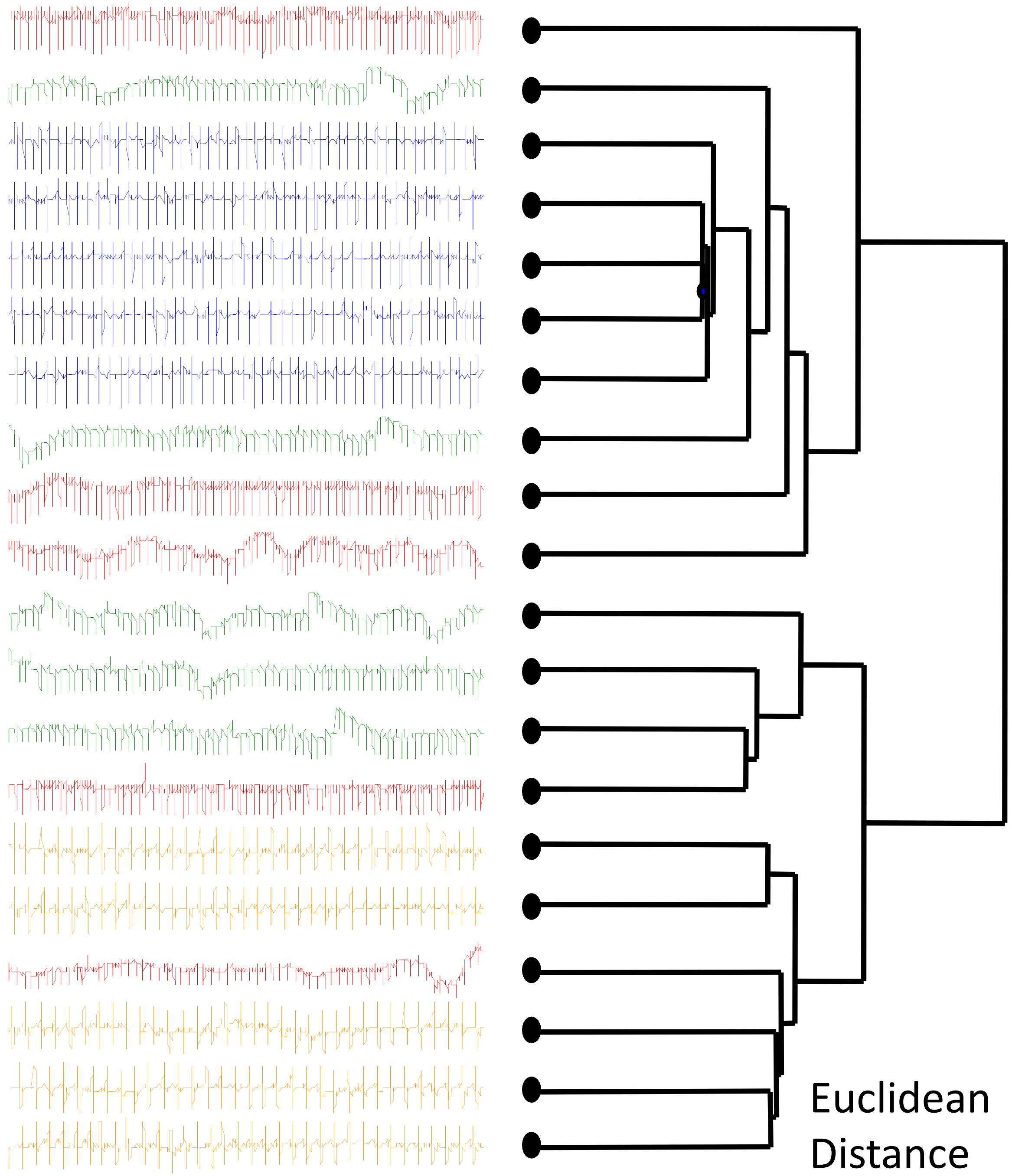
You go to the doctor because of chest pains. Your ECG looks strange…

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

Two questions:

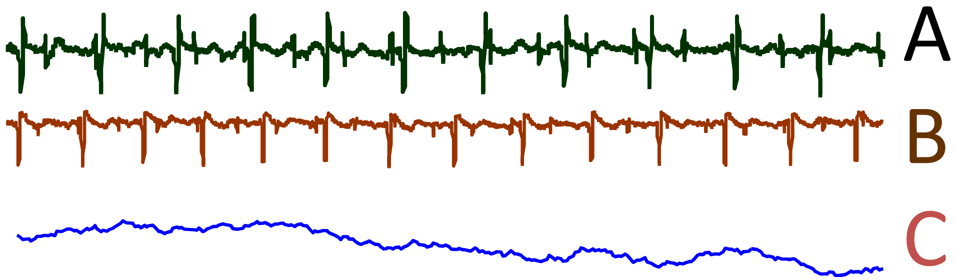
- How do we define similar?
- **How do we search quickly?**


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


Euclidean
Distance

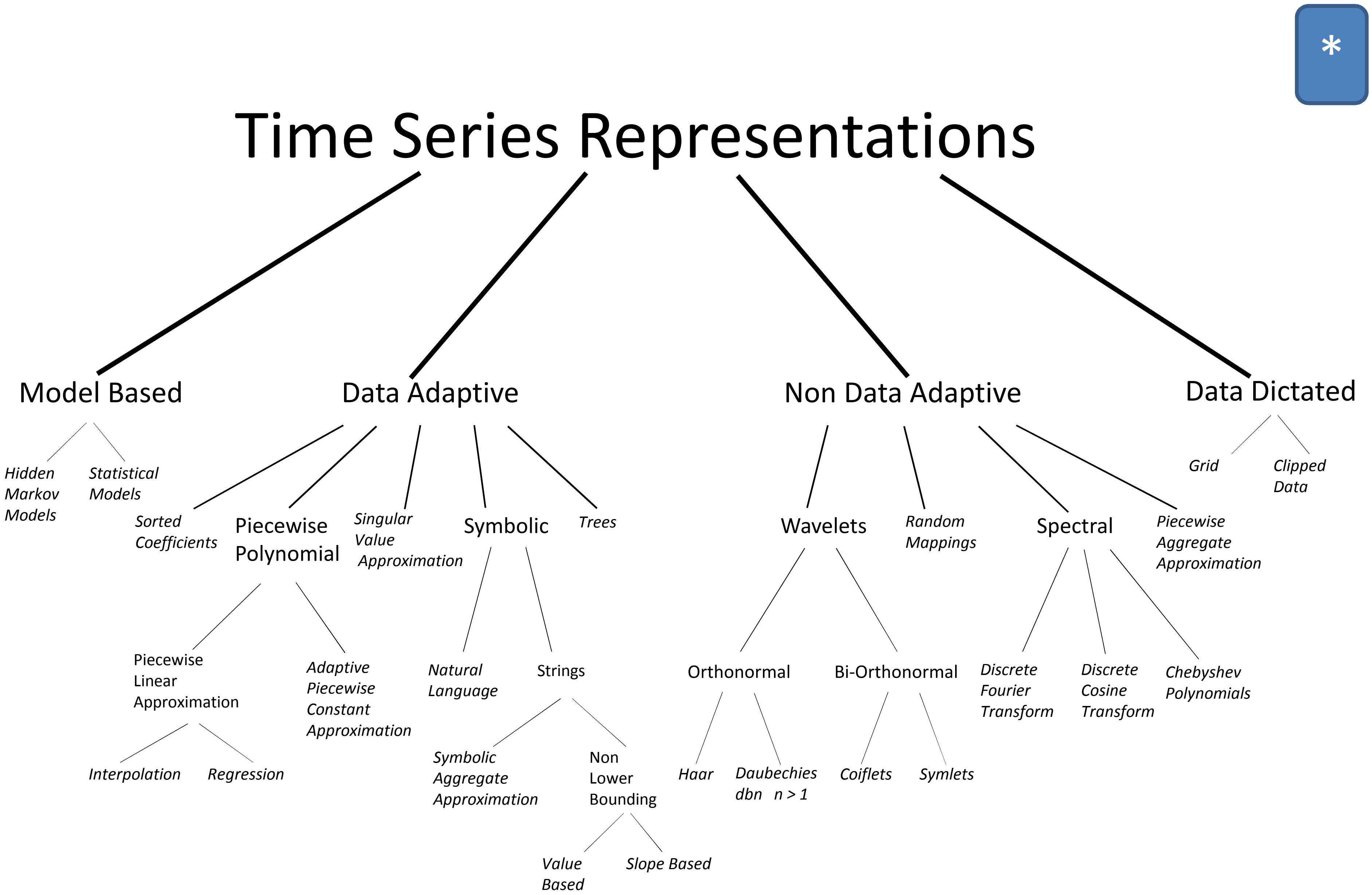
Structure or Model Based Similarity

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



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

Time Series / Feature	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
…	…	…	…


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

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

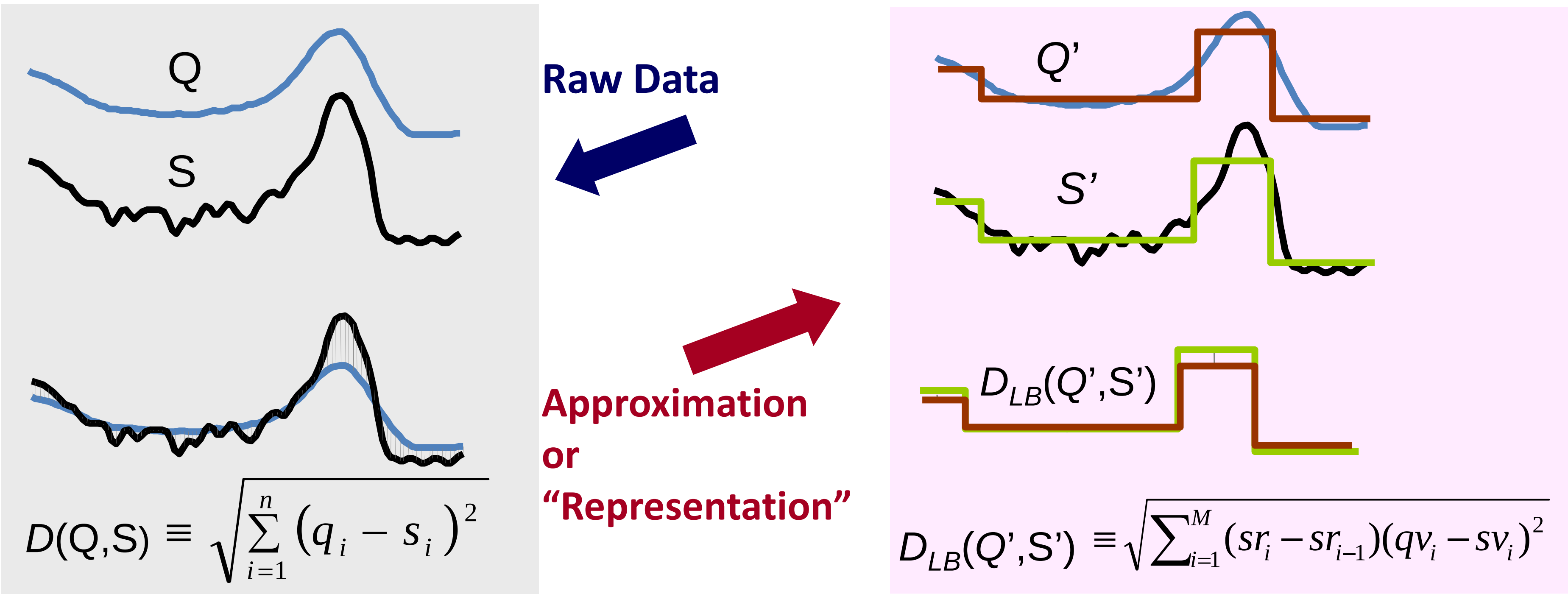
- Approximately solve the problem at hand in main memory

- 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?

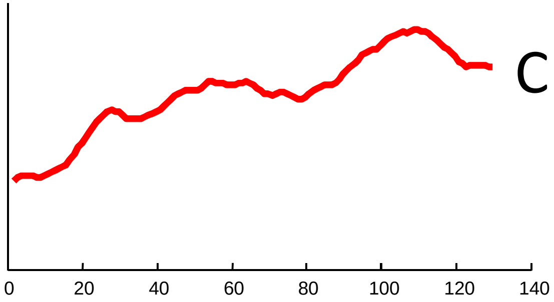
• Recall that we have seen lower bounding for **distance measures** (DTW and uniform scaling) Lower bounding for **representations** is a similar idea…

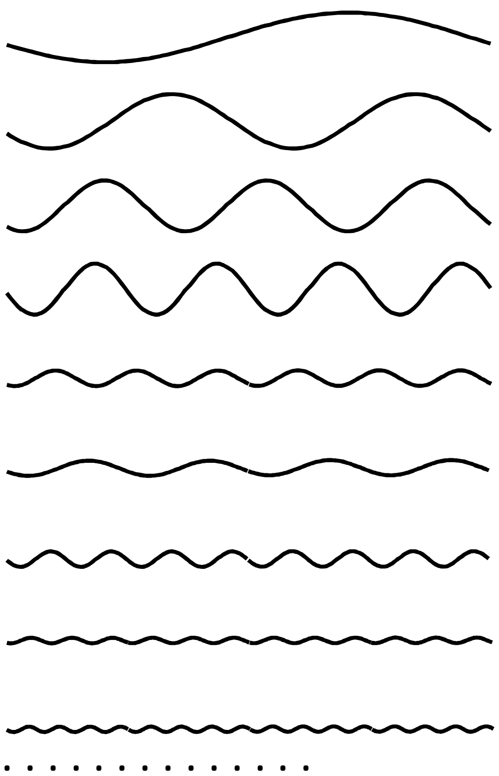


Lower bounding means that for all Q and S, we have:

$$D_{LB}(Q',S') \leq D(Q,S)$$

An Example of a Dimensionality Reduction Technique II





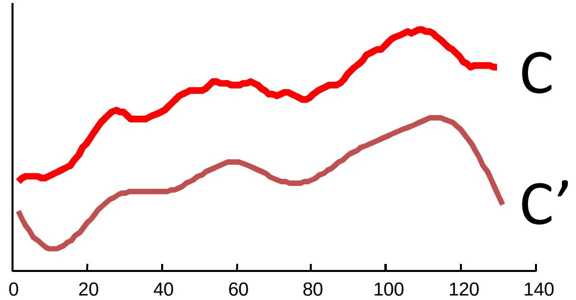
Raw Data	Fourier Coefficients
0.4995	1.5698
0.5264	1.0485
0.5523	0.7160
0.5761	0.8406
0.5973	0.3709
0.6153	0.4670
0.6301	0.2667
0.6420	0.1928
0.6515	0.1635
0.6596	0.1602
0.6672	0.0992
0.6751	0.1282
0.6843	0.1438
0.6954	0.1416
0.7086	0.1400
0.7240	0.1412
0.7412	0.1530
0.7595	0.0795
0.7780	0.1013
0.7956	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
…	…
…	…

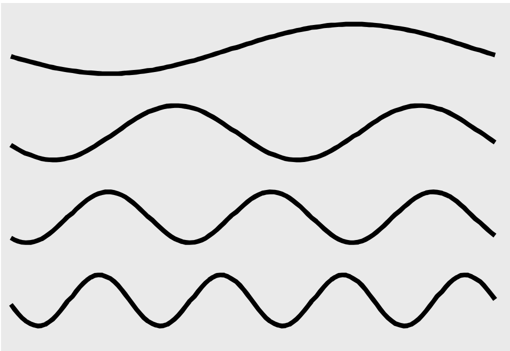
We can decompose the data into 64 pure sine waves using the **D**iscrete **F**ourier **T**ransform (DFT) - just the first few sine waves are shown.

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

of the data.

$$\frac{15}{16}$$

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

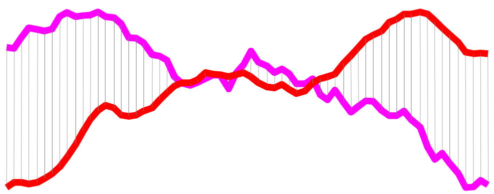
N = 8

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



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

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!

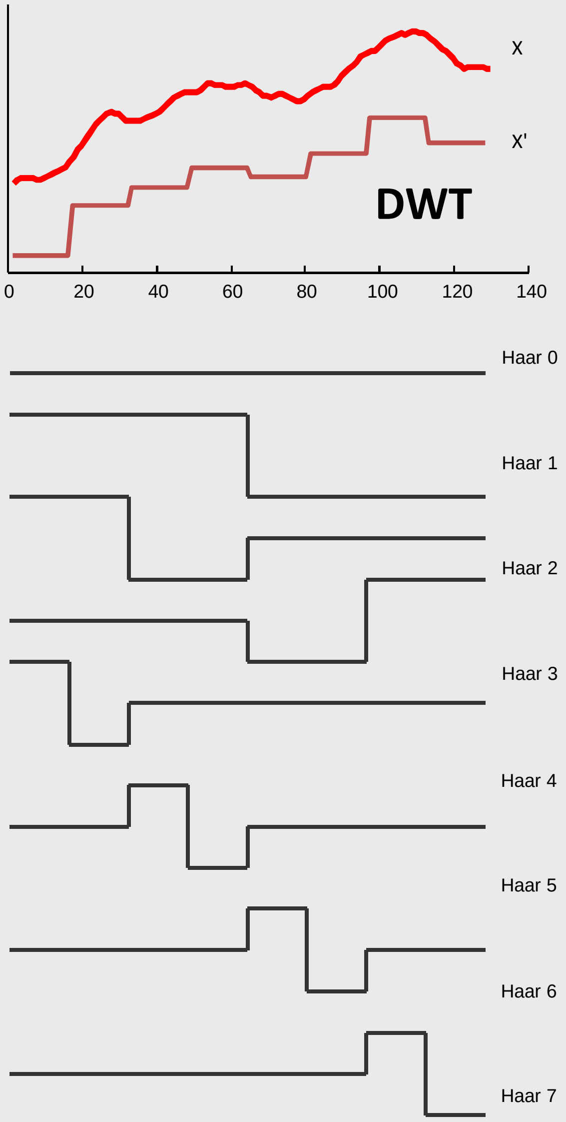
w ta 1	Raw Data 2	Raw Data n
.4995	0.7412	0.8115
.5264	0.7595	0.8247
.5523	0.7780	0.8345
.5761	0.7956	0.8407
.5973	0.8115	0.8431
.6153	0.8247	0.8423
.6301	0.8345	0.8387
.6420	0.8407	0.4995
.6515	0.8431	0.7412
.6596	0.8423	0.7595
.6672	0.8387	0.7780
.6751	0.4995	0.7956
.6843	0.5264	0.5264
.6954	0.5523	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

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.

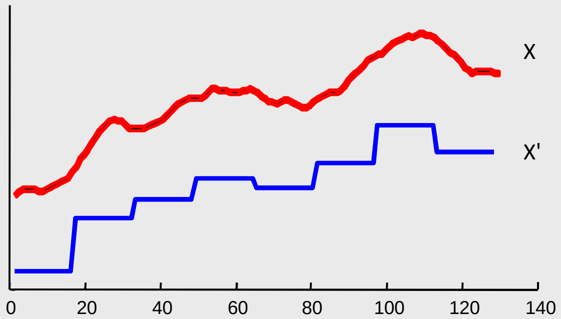
Alfred Haar

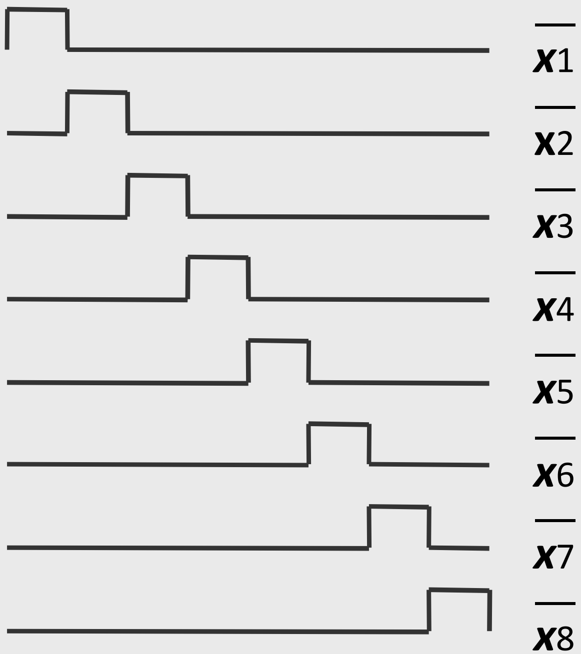
1885-1933

Excellent free Wavelets Primer

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

Piecewise Aggregate Approximation I





*

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

Note that each box is of the same length (***n/N***).

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

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

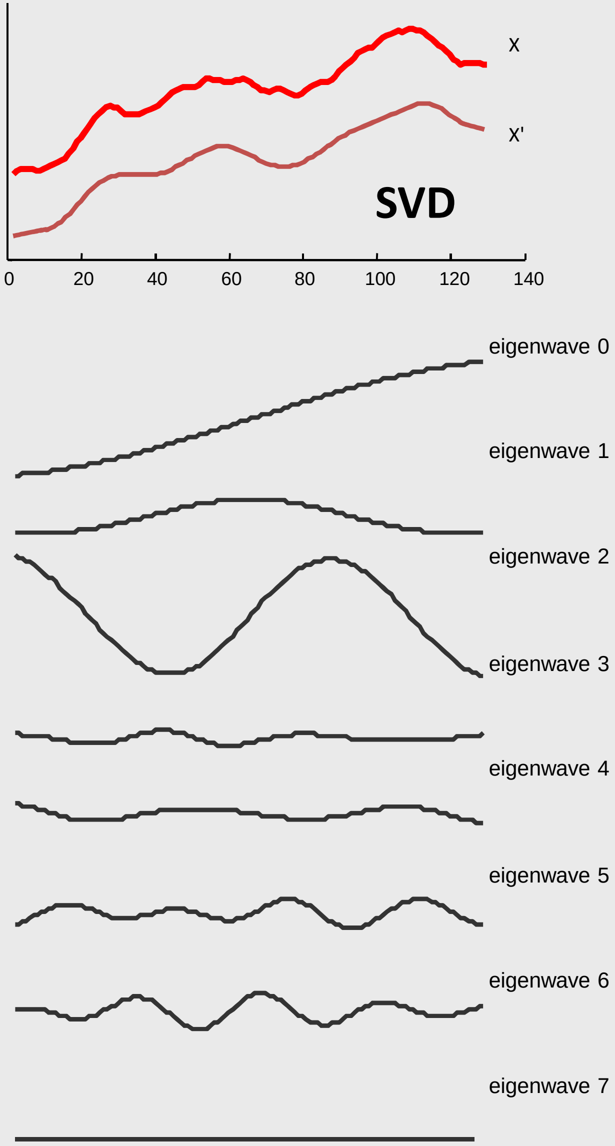
$$DR(\bar{X},\bar{Y}) \equiv \sqrt{\tfrac{n}{N}} \sqrt{\sum_{i=1}^{N} (\bar{x}_i - \bar{y}_i)^2}$$

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

Singular Value Decomposition I



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

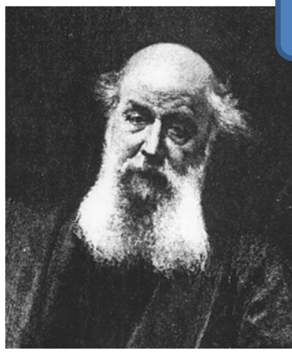
SVD is similar to Fourier and Wavelet approaches is that we represent the data in terms of a linear combination of shapes (in this case *eigenwaves*).

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.

Good free SVD Primer

Singular Value Decomposition - A Primer. Sonia Leach

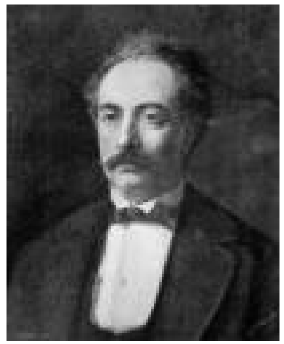


James Joseph Sylvester

1814-1897

Camille Jordan

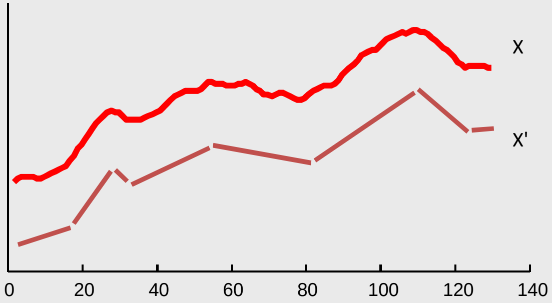
(1838--1921)

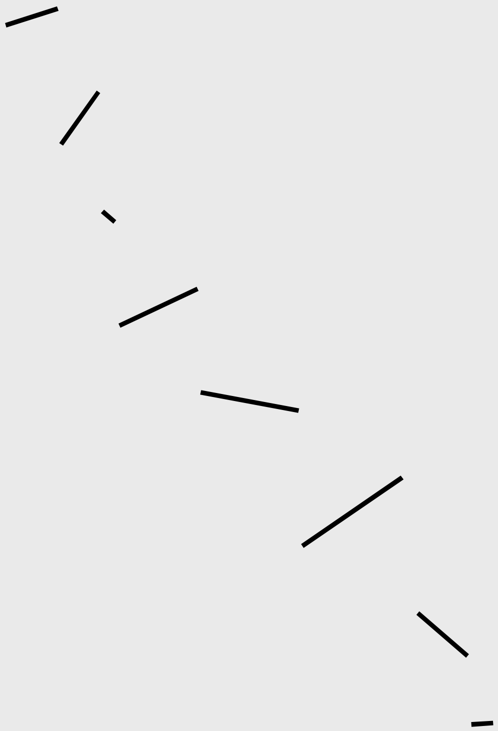


Eugenio Beltrami

1835-1899

Piecewise Linear Approximation





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

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

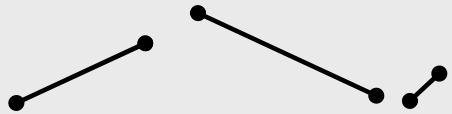
If lines are **disconnected**, we are allowed only $N/3$ lines

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

Karl Friedrich Gauss, 1777 - 1855

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

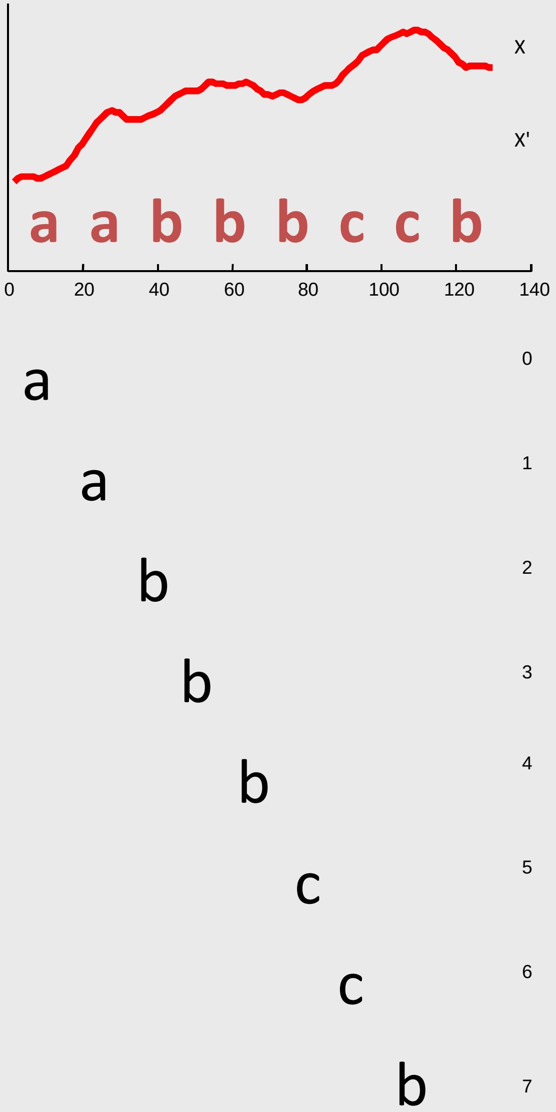
(`right_height` can be inferred by looking at the next segment)



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

- Good ability to compress natural signals.
- Fast linear time algorithms for PLA exist.
- Already widely accepted in some communities (e.g. biomedical)

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.

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**.