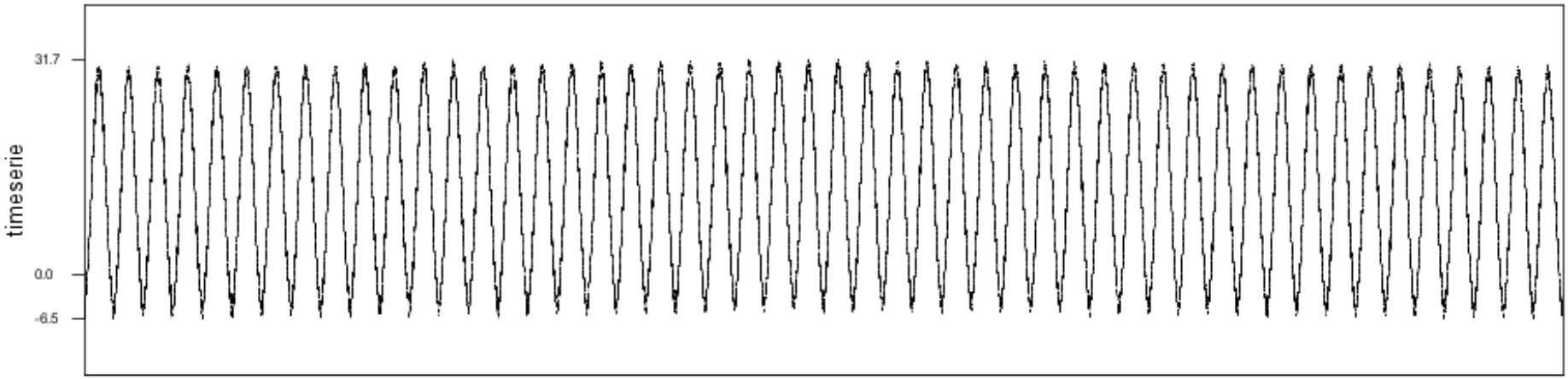
Time Series Analysis

Time Series

- **Motivation**
- **Decomposition Models**
  - Additive models, multiplicative models
- **Global Approaches**
  - Regression
  - With and without seasonal component
- **Local Approaches**
  - Moving Averages Smoothing
  - With and without seasonal component
- **Summary**

Motivation: Temperatures

Example: Temperatures data set (fictive)



- The plot shows the average temperature per day for 50 years.
- Is there any trend visible?
- How to extract seasonal effects?

Decomposition Models

- The time series is given as a sequence of values

$$y_1, \ldots, y_t, \ldots, y_n$$

- We assume that every y_t is a composition of (some of) the following components:
  - g_t trend component
  - s_t seasonal component
  - c_t cyclical variation
  - ϵ_t irregular component (random factors, noise)
- Assume a functional dependency:

$$y_t = f(g_t, s_t, c_t, \epsilon_t)$$

Components of Time Series

Trend Component

- Reflects long-term developments.
- Often assumed to be a monotone function of time.
- Represents the actual component we are interested in.

Cyclic Component

- Reflects mid-term developments.
- Models economical cycles such as booms and recessions.
- Variable cycle length.
- We do not consider this component here.

Remark: Often, both components are combined.

Components of Time Series

Seasonal Component

- Reflects short-term developments.
- Constant cycle length (i. e., 12 months)
- Represents changes that (re)occur rather regularly.

Irregular Component

- Represents everything else that cannot be related to the other components.
- Combines irregular changes, random noise and local fluctuations.
- We assume that the values are small and have an average of zero.

Additive Decomposition

$$y_t = g_t + s_t + \epsilon_t$$

- Pure trend model: $y_t = g_t + \epsilon_t$ (stock market, no season)
- Possible extension: $y_t = g_t + s_t + x_t\beta + \epsilon_t$ (calendar effects)

Multiplicative Decomposition

$$y_t = g_t \cdot s_t \cdot \epsilon_t$$

- Seasonal changes may increase with trend.
- Transform into additive model:

$$\tilde{y}_t = \log y_t + \log s_t + \log \epsilon_t$$

Goal: Estimate the components from a given time series, i. e.

$$\hat{g}_t + \hat{s}_t + \epsilon_t \approx y_t$$

Application: With the estimates, we can compute the

- trend-adjusted series: $y_t - \hat{g}_t$
- season-adjusted series: $y_t - \hat{s}_t$
- We only consider additive models here.

$\Rightarrow$ Additional assumptions necessary in order to find ways to infer the desired components.

Overview

- **Global approach:** There is a fix functional dependence throughout the entire time range. ($\Rightarrow$ regression models)
- **Local approach:** We do not postulate a global model and rather use local estimations to describe the respective components.
- **Seasonal effects:** We have to decide beforehand whether to assume a seasonal component or not.

	Global	Local
without Season	Regression	Smoothing Averages
with Season	Dummy Variables	Smoothing Averages

Global Approach (without Season)

Model: $y_t = g_t + \epsilon_t$

Assumptions:

- No seasonal component: $s_t = 0$
- Depending on g_t, use regression analysis to estimate the parameter(s) to define the trend component.
  - linear trend: $g_t = \beta_0 + \beta_1 t$
  - quadratic trend: $g_t = \beta_0 + \beta_1 t + \beta_2 t^2$
  - polynomial trend: $g_t = \beta_0 + \beta_1 t + \cdots + \beta_q t^q$
  - exponential trend: $g_t = \beta_0 \exp(\beta_1 t)$
  - logistic trend: $g_t = \frac{\beta_0}{\beta_1 + \exp(-\beta_2 t)}$

Global Approach (with Season)

Model: $y_t = s_t + \epsilon_t$ (no trend)

Assumptions:

- No trend component: $g_t = 0$
- Seasonal component does not change from period to period.
- Introduce *dummy variables* for every time span (here: months) that serve as indicator functions to determine to which month a specific t belongs:

$$s_m(t) = \begin{cases} 1, & \text{if } t \text{ belongs to month } m \\ 0, & \text{otherwise} \end{cases}$$

- The seasonal component is then set up as $s_t = \sum_{m=1}^{12} \beta_m s_m(t)$.
- Determine the *monthly effects* β_m with normal least squares method.

Global Approach (with Season)

Model: $y_t = g_t + s_t + \epsilon_t$

Assumptions:

- Estimate $\hat{g}_t$ while temporarily ignoring s_t.
- Estimate s_t from the trend-adjusted $\tilde{y}_t = y_t - \hat{g}_t$.

Model: $y_t = \alpha_1 t + \cdots + \alpha_q t^q + \cdots + \beta_1 s_1(t) + \cdots + \beta_{12} s_{12}(t) + \epsilon_t$

Assumptions:

- Seasonal component does not change from period to period.
- Model the seasonal effects with trigonometric functions:

$$s_t = \beta_0 + \sum_{m=1}^{6} \beta_m \cos\Big(2\pi \frac{m}{12} t\Big) + \sum_{m=1}^{5} \gamma_m \sin\Big(2\pi \frac{m}{12} t\Big)$$

- Determine $\alpha_1, \ldots, \alpha_q, \beta_0, \ldots, \beta_6$ and $\gamma_1, \ldots, \gamma_5$ with normal least squares method.

Local Approach (without Season)

General Idea: Smooth the time series.

- Estimate the trend component g_t at time t as the average of the values around time t.

For a given time series $y_1, \ldots, y_n$, the **Smoothing Average** $y_t^\star$ of order r is defined as follows:

$$y_t^\star = \begin{cases} \frac{1}{2k+1} \cdot \sum_{j=-k}^{k} y_{t+j}, & \text{if } r = 2k+1 \\ \\ \frac{1}{2k} \cdot (\frac{1}{2} y_{t-k} + \sum_{j=-k+1}^{k-1} y_{t+j} + \frac{1}{2} y_{t+k}), & \text{if } r = 2k \end{cases}$$

Local Approach (without Season)

Model: $y_t = g_t + \epsilon_t$

Assumptions:

- In every time frame of width $2k + 1$ the time series can be assumed to be linear.
- ϵ_t averages to zero.
- Then we use the smoothing average to estimate the trend component:

$$\hat{g}_t = y_t^\star$$

Local Approach (with Season)

Model: $y_t = g_t + s_t + \epsilon_t$

Assumptions:

- Seasonal component has period length p (repeats after p points):

$$s_t = s_{t+p}, \quad t = 1, \ldots, n-p$$

- Sum of seasonal values is zero: $\sum_{j=1}^{p} s_j = 0$

- Trend component is linear in time frames of width p (if p is odd) or $p+1$ (if p is even).

- Irregular component averages to zero.

Local Approach (with Season)

Let $k = \frac{p-1}{2}$ (for odd p) or $k = \frac{p}{2}$ (for even p).

Then:

- Estimate the trend component with smoothing average:

$$\hat{g}_t = y_t^\star, \qquad k+1 \leq t \leq n-k$$

- Estimate the seasonal components $s_1, \ldots, s_p$ as follows:

$$\hat{s}_i = \tilde{s}_i - \frac{1}{p}\sum_{j=1}^{p} \tilde{s}_j \quad \text{with} \quad \tilde{s}_t \frac{1}{m_i - l_i + 1} \sum_{j=l}^{m_i} (y_{i+jp} - y_{i+jp}^\star), \qquad 1 \leq i \leq p$$

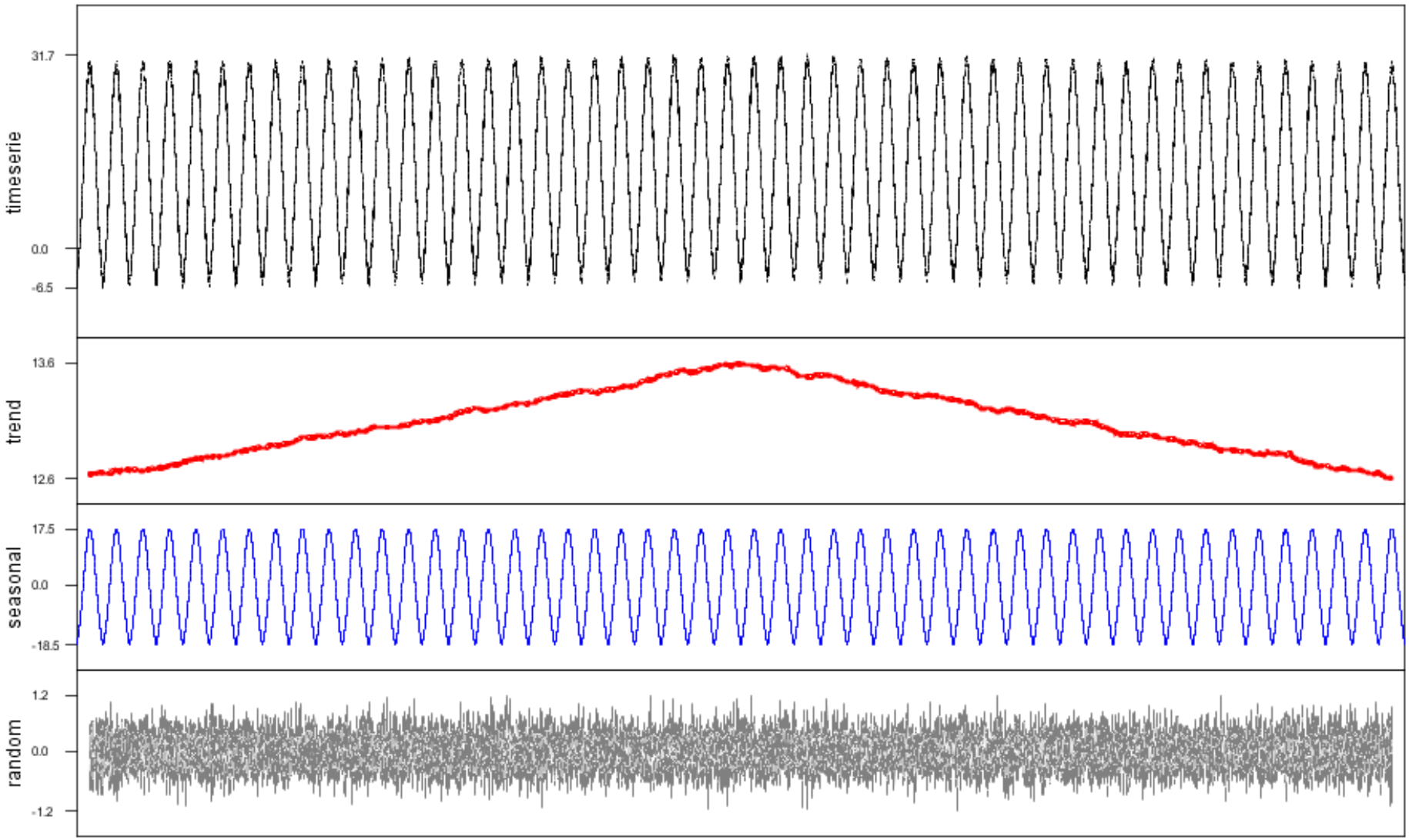
where

$$m_i = \max \{m \in \mathbb{N}_0 \mid i + mp \leq n - k\}$$

and

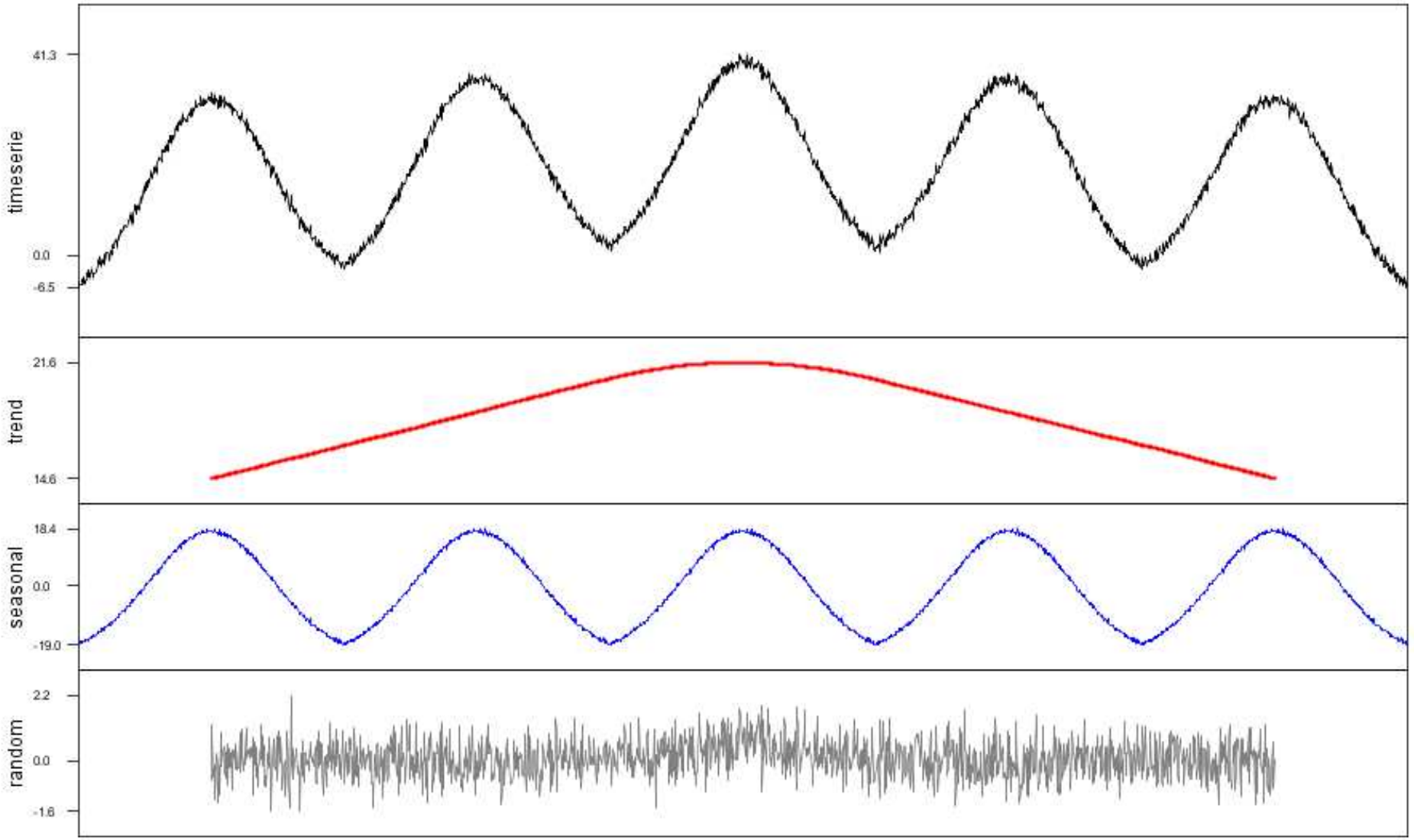
$$l_i = \min \{l \in \mathbb{N}_0 \mid i + lp \geq k + 1\}$$

Example (from motivation)



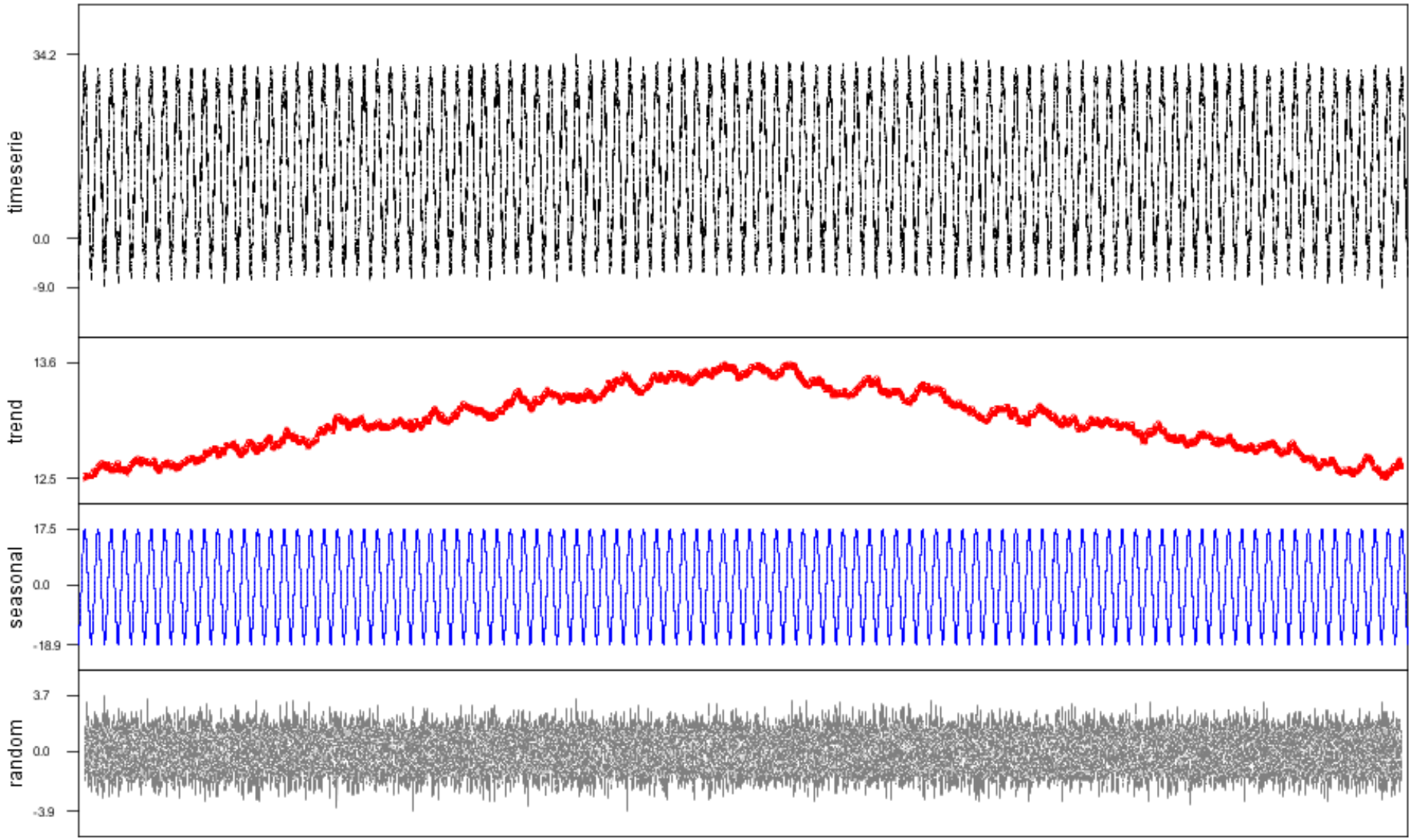
- We can extract an increase and decrease of 1 degree during 50 years even though the amount of noise is more than twice as large than the actual trend.

Example



5 years period, trend ±8 degrees, noise amount ±2 degrees

Example



100 years period, trend ±1 degree, noise amount ±3 degrees

Summary

- **Definition of the problem domain**
  - Consider a time series to be composed of subcomponents.
  - Additive and multiplicative models.
- **Global and local approaches**
  - With and without seasonal components.
- **Robust to noise**
  - Noise can be higher than the trend component itself.

Frequent Patterns in Temporal Data

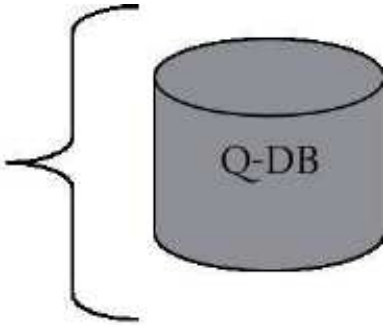
Contents

- Frequent pattern in temporal data
  - Motivation / Problem
  - Other common methods
  - Algorithms / Example

Quality Surveillance of Vehicles

Diesel, automatic, AMG, ...

SSL1 on 1.4.2006, SSL2 on 11.11.2006, ...



Quality Surveillance of Vehicles

Standheizung — Park heating

Automatik — Automatic

SSL 1 | SSL 2 | SSL 3

Kombi — Estate car

SSL 3

Automatik — Automatic

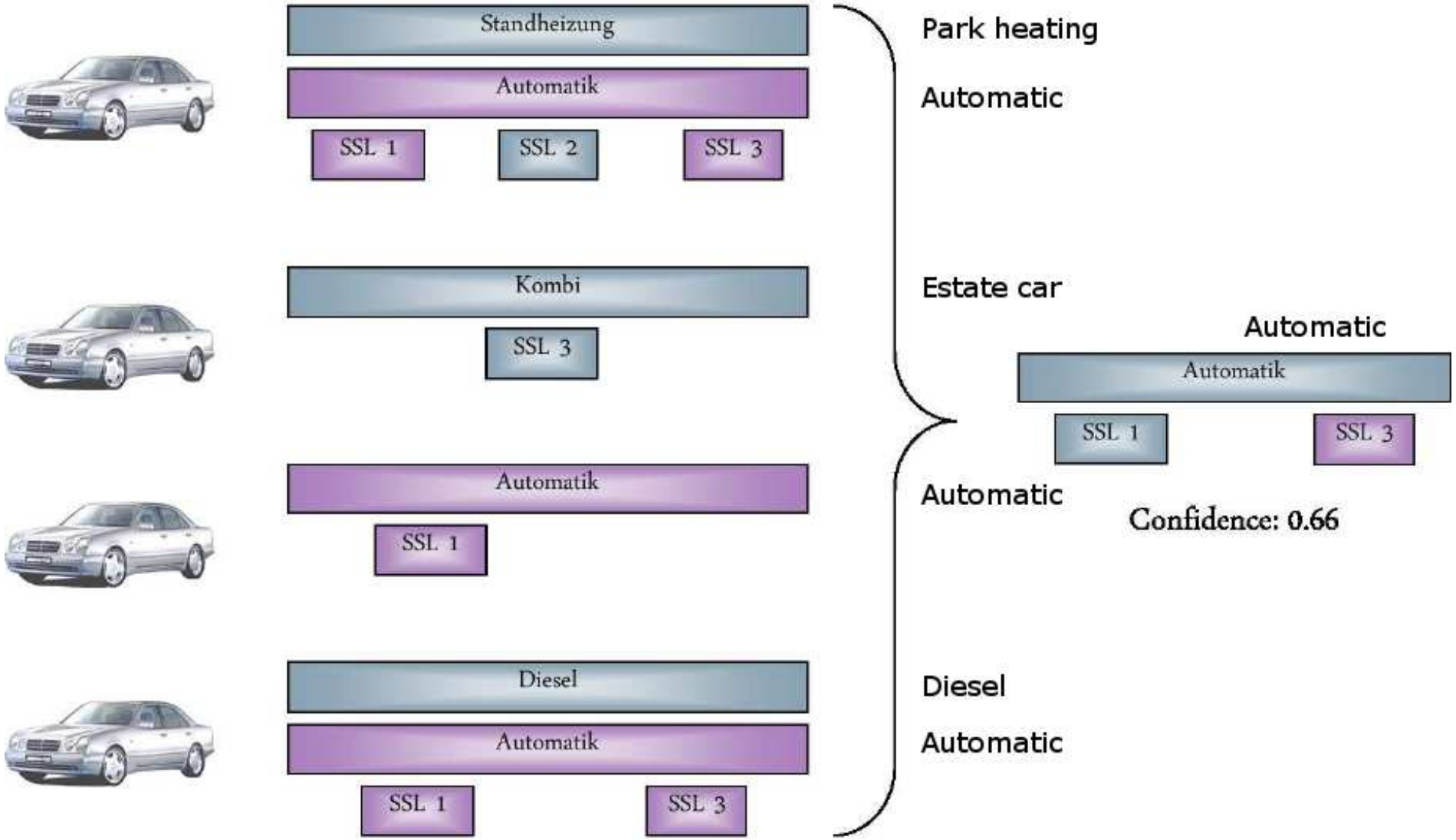
SSL 1

Diesel — Diesel

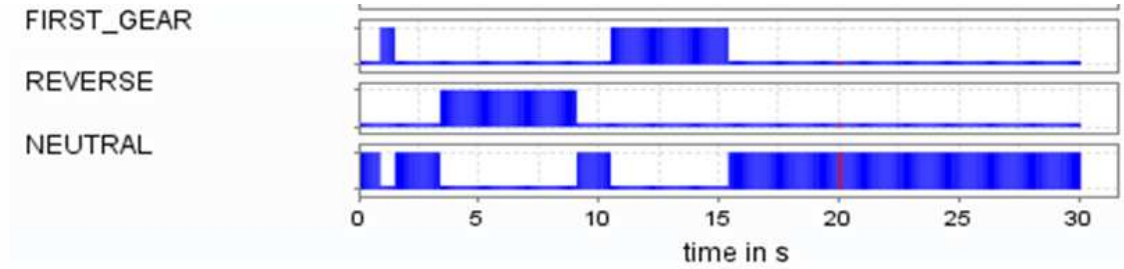
Automatik — Automatic

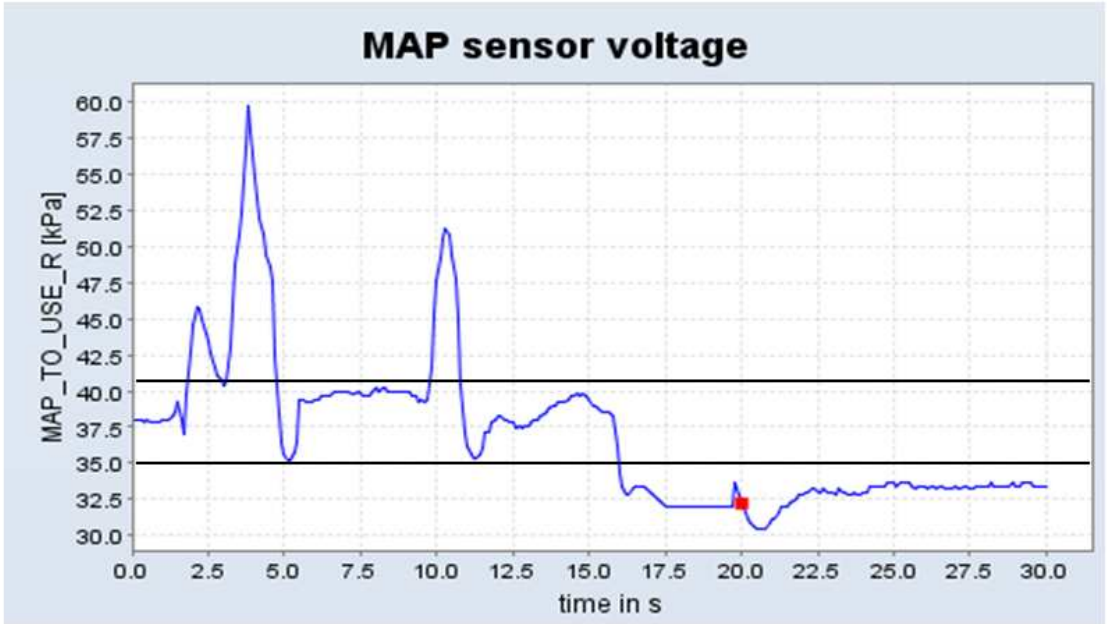
SSL 1 | SSL 3

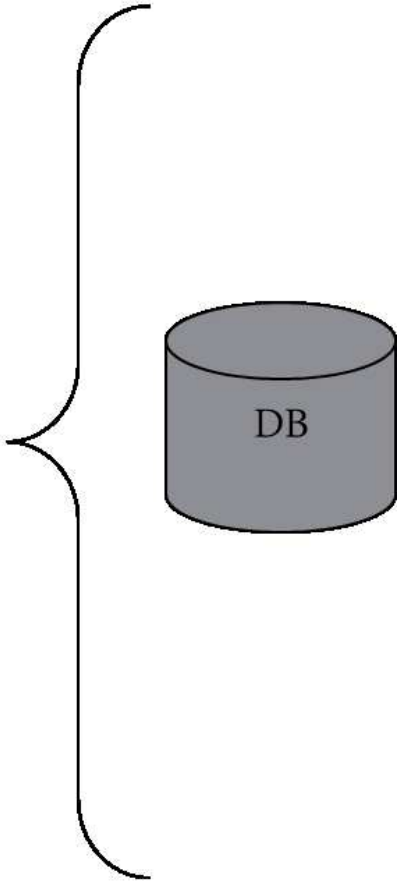
Quality Surveillance of Vehicles



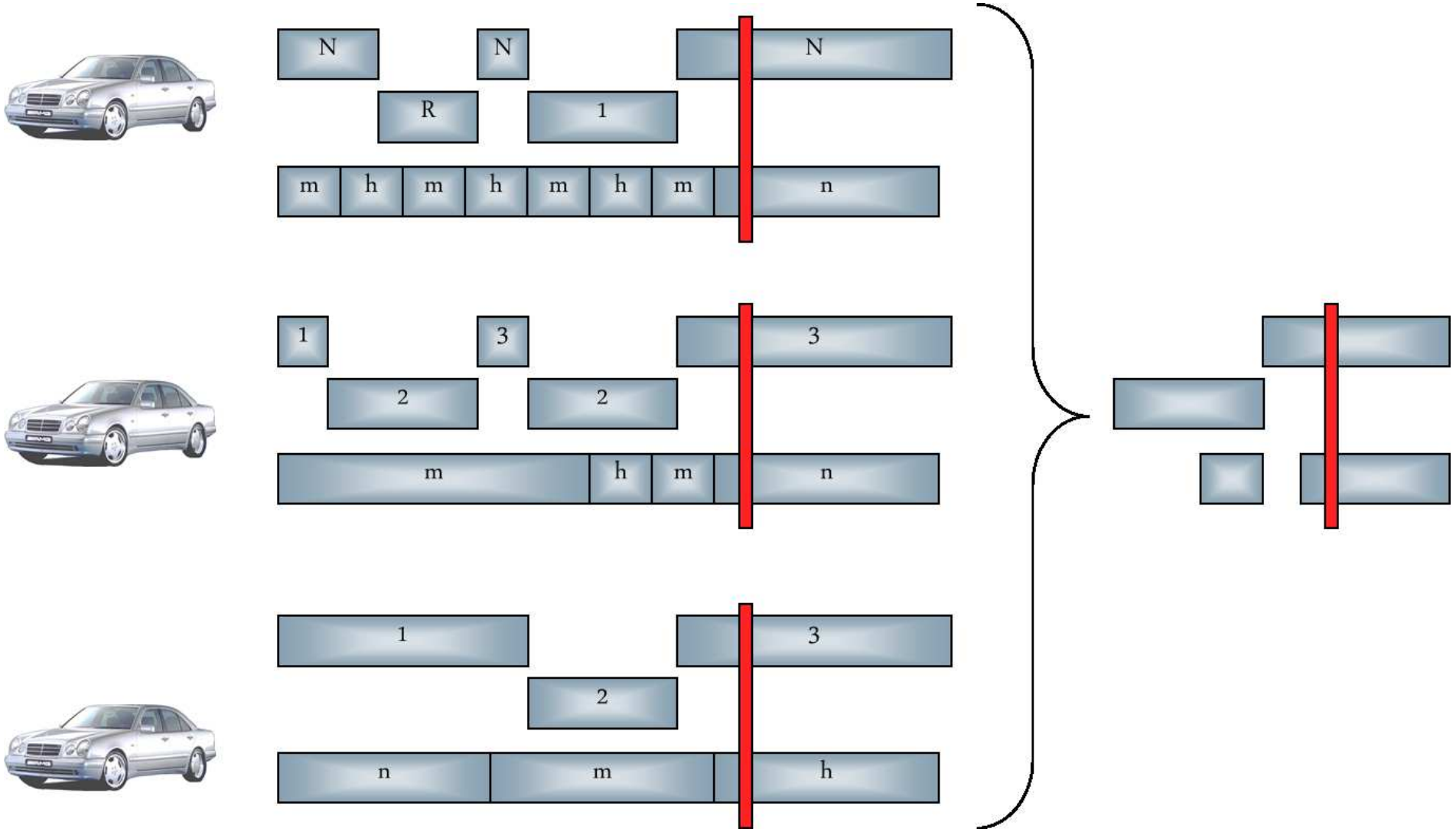
Pilot Series Vehicles



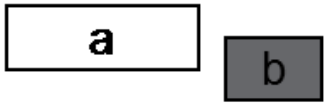
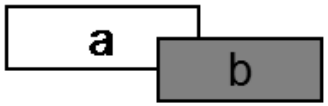
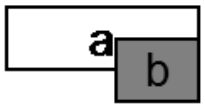
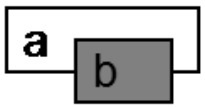
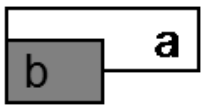
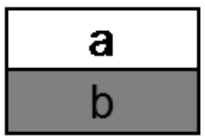




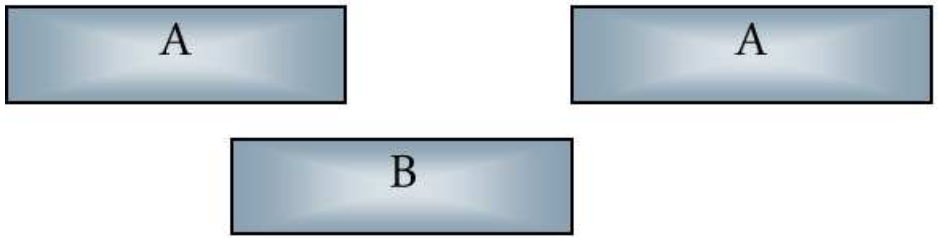
Pilot Series Vehicles



What is a Temporal Pattern?

Relation a to b		Inverse Relation b to a
a *before* b		b *after* a
a *meets* b		b *is-met-by* a
a *overlaps* b		b *is-overlaped-by* a
a *is-finished-by* b		b *finishes* a
a *contains* b		b *during* a
a *is-started-by* b		b *starts* a
a *equals* b		b *equals* a

What is a Temporal Pattern?

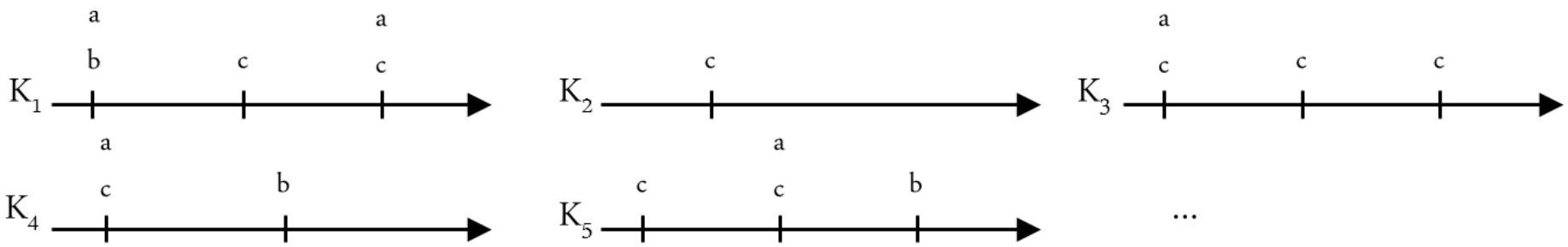


	A	B	A
A			
B			
A			

Content

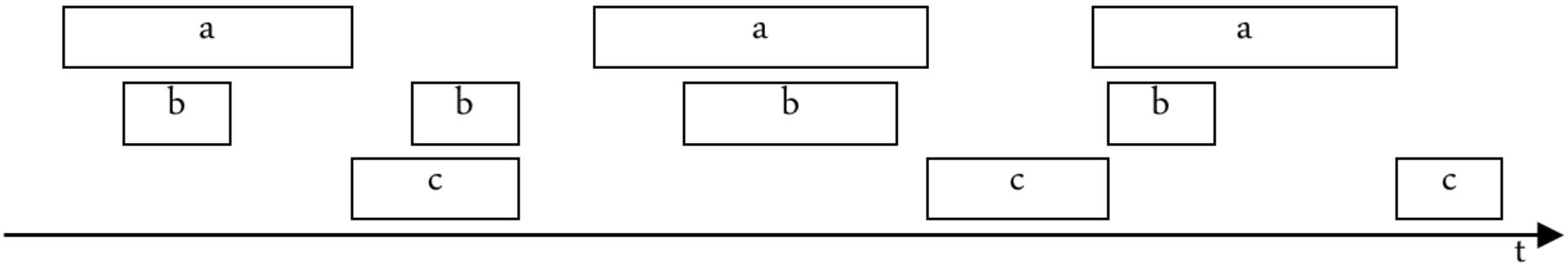
- Frequent pattern in temporal data
  - Motivation / Problem
  - **Other common methods**
  - Algorithms / Example

Agrawal 1995



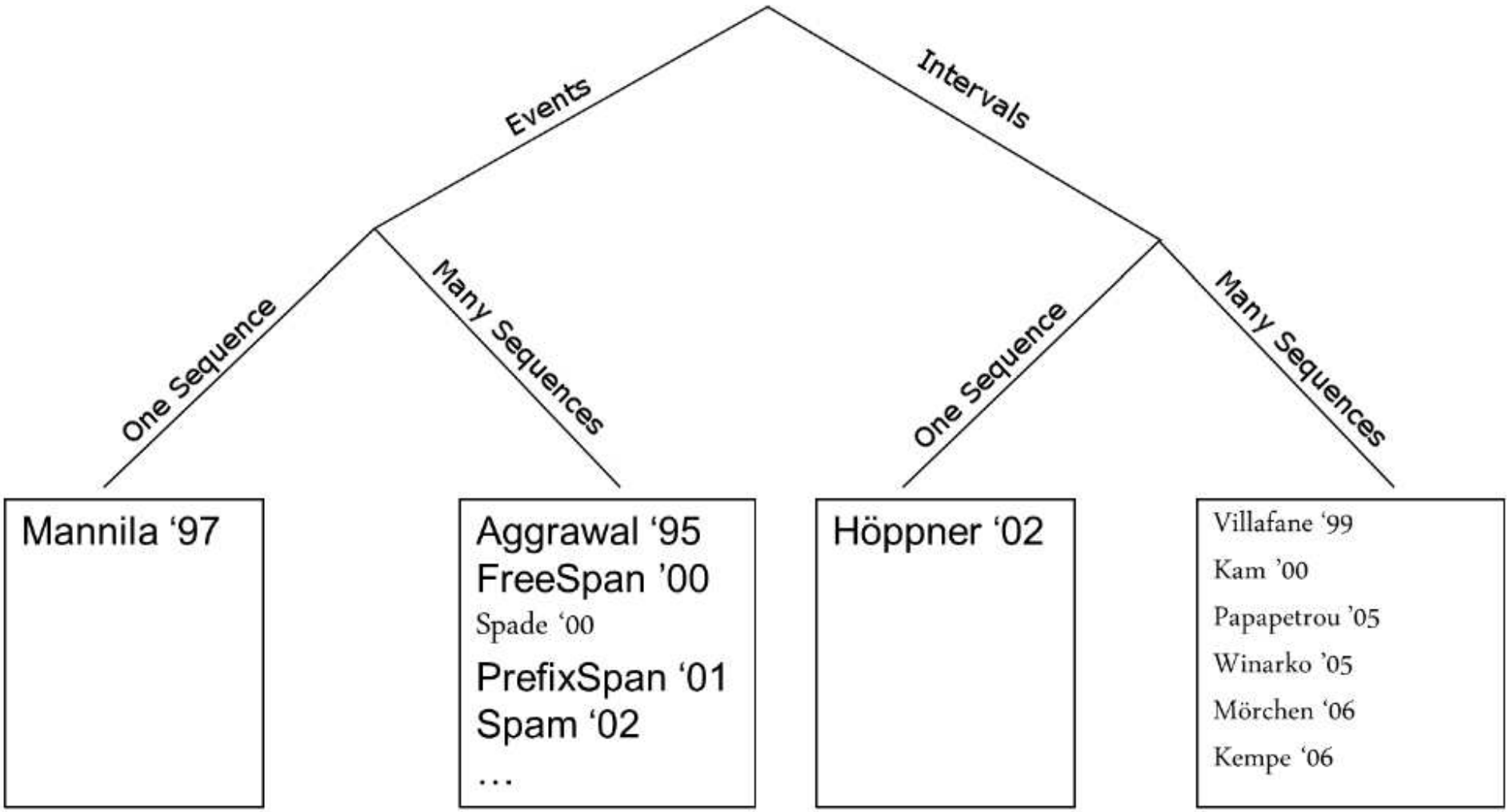
- Many customers with a history of item sets (transaction)
- Searches for sequences
  - z.B. $\{a, b\} \rightarrow \{c\} \rightarrow \{b, c\}$
- No rules
- No time window
- Support: Count!
  - Support counter of a sequence is incremented, if it occurs for a customer

Höppner 2002



- On a single time line, events have a duration
- Searches for patterns
  - a contains b and meets with c
- Rules are induced from patterns
- Uses a time frame that needs to contain the pattern
- Support: *Temporal Support*

Related Methods



Content

- Frequent patterns in temporal data
  - Motivation / Problem
  - Other common methods
  - **Algorithms / Example**

Partial Apriori Criterion

	A	B A
A	e	o b
B	io	e m
A	a	im e

- All suffixes of a frequent pattern are frequent.

Candidate Generation

Pattern P	l_p	$l_1 \ldots l_{k-1}$
l_p	e	C
l_1		
$\vdots$	B	A
l_{k-1}		

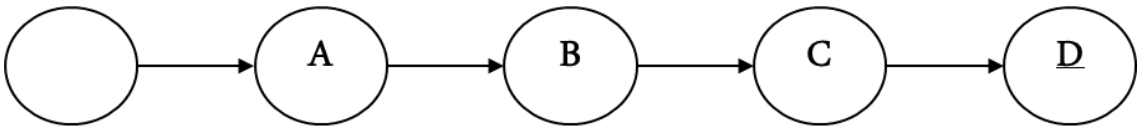
Pattern Q	l_q	$l_1 \ldots l_{k-1}$
l_q	e	E
l_1		
$\vdots$	D	A
l_{k-1}		

Pattern R	l_p	l_q	$l_1 \ldots l_{k-1}$
l_p	e	?	C
l_q	?	e	E
l_1			
$\vdots$	B	D	A
l_{k-1}			

Allens Interval Relations

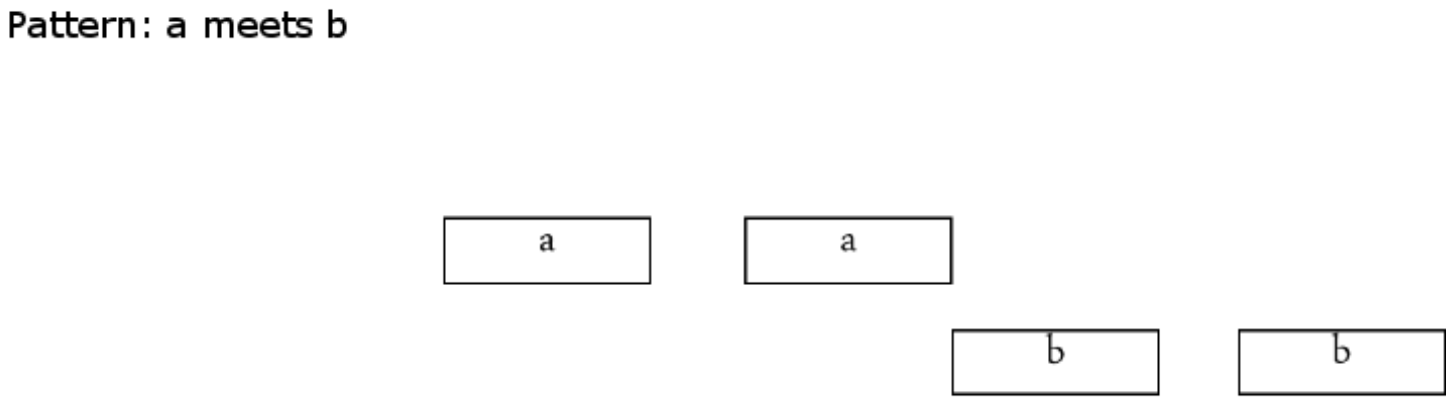
Support Evaluation

	A	B	C	D
A	e	**e**	**c**	**c**
B	e	e	**c**	**c**
C	d	d	e	**b**
D	d	d	a	e



- Exploit the property of normalised patterns by finite automata

Finite automata get stuck

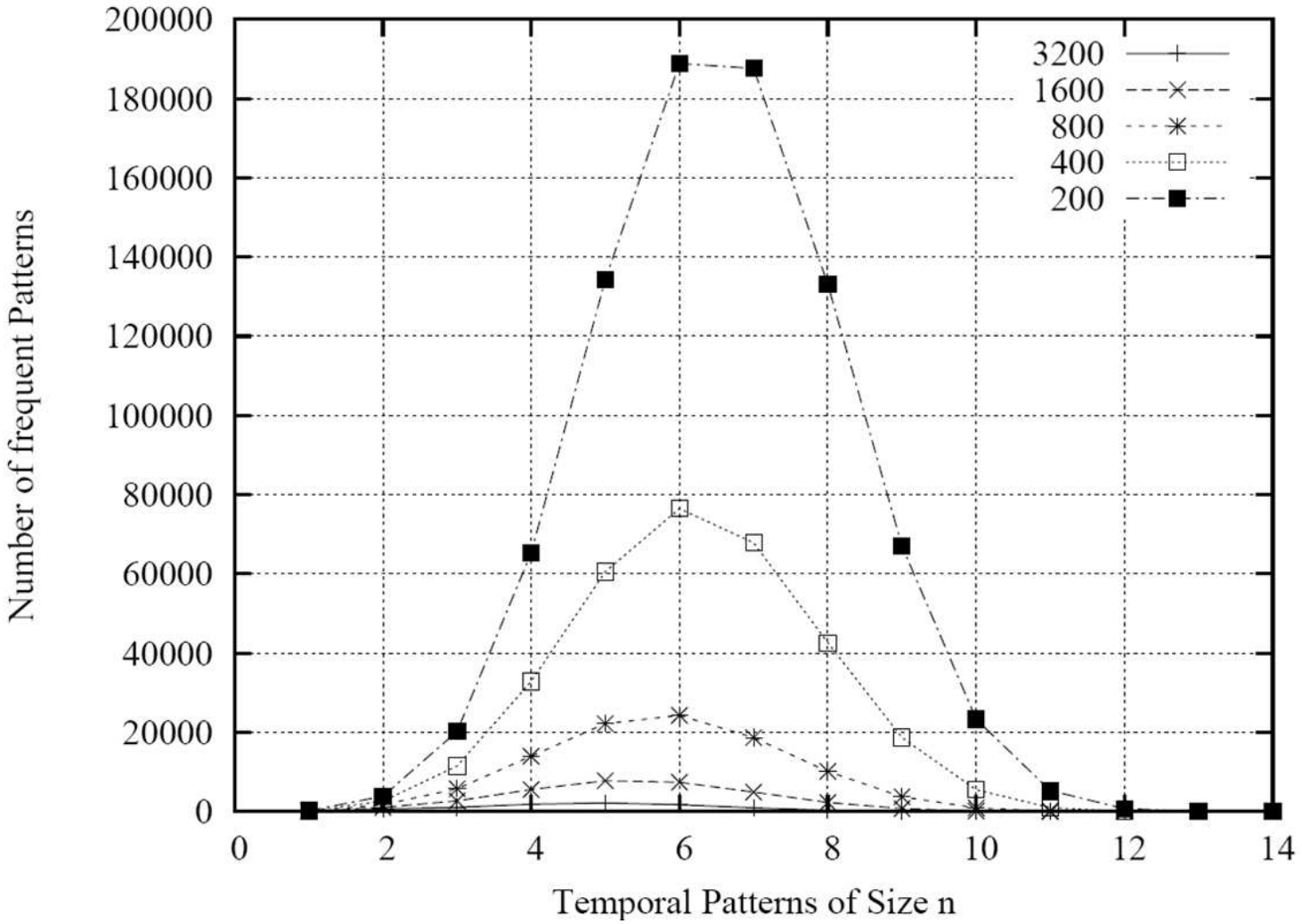


- A finite automaton gets stuck after accepting the fist 'a'.
- Solution to this problem: Create copies and filter found occurences

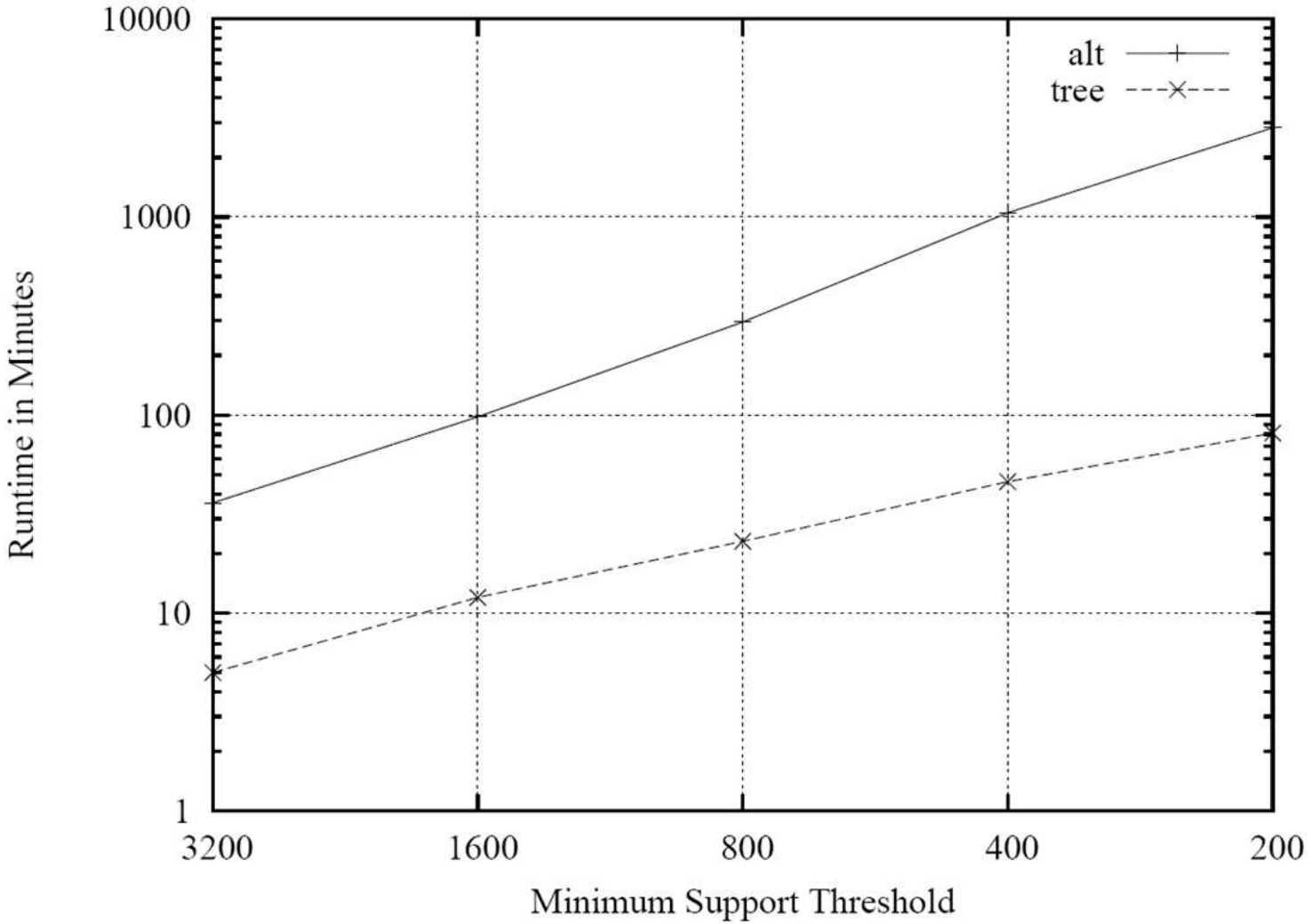
Example: Quality surveillance of Vehicles

- 101 250 vehicles
  - Workshop stops
  - Vehicle configuration
  - 1.4 Mio. temporal intervals

Number of Frequent Patterns



Runtime


10000
1000
100
10
1
Runtime in Minutes
alt
tree
3200
1600
800
400
200
Minimum Support Threshold

Efficiently Finding Motifs in Time Series

Content

- Efficiently Finding Motifs in Time Series
  - Data Mining in Time Series
  - memory-efficient Representationen
  - Symbolic Aggregat-Approximation (SAX)
  - Finding Motifs in Time Series with SAX
  - Example

Data Mining in Time Series

- Main Task: Find useful information in time series
- typical problems: Clustering, Classification, Discovery of frequent patterns and rules, visualisisation, anomaly detection
- Problems are reduced to finding repeated, similar subsequences because of the amount of data
- requires: Similarity measure to compare subsequences
- e.g. euclidean distance

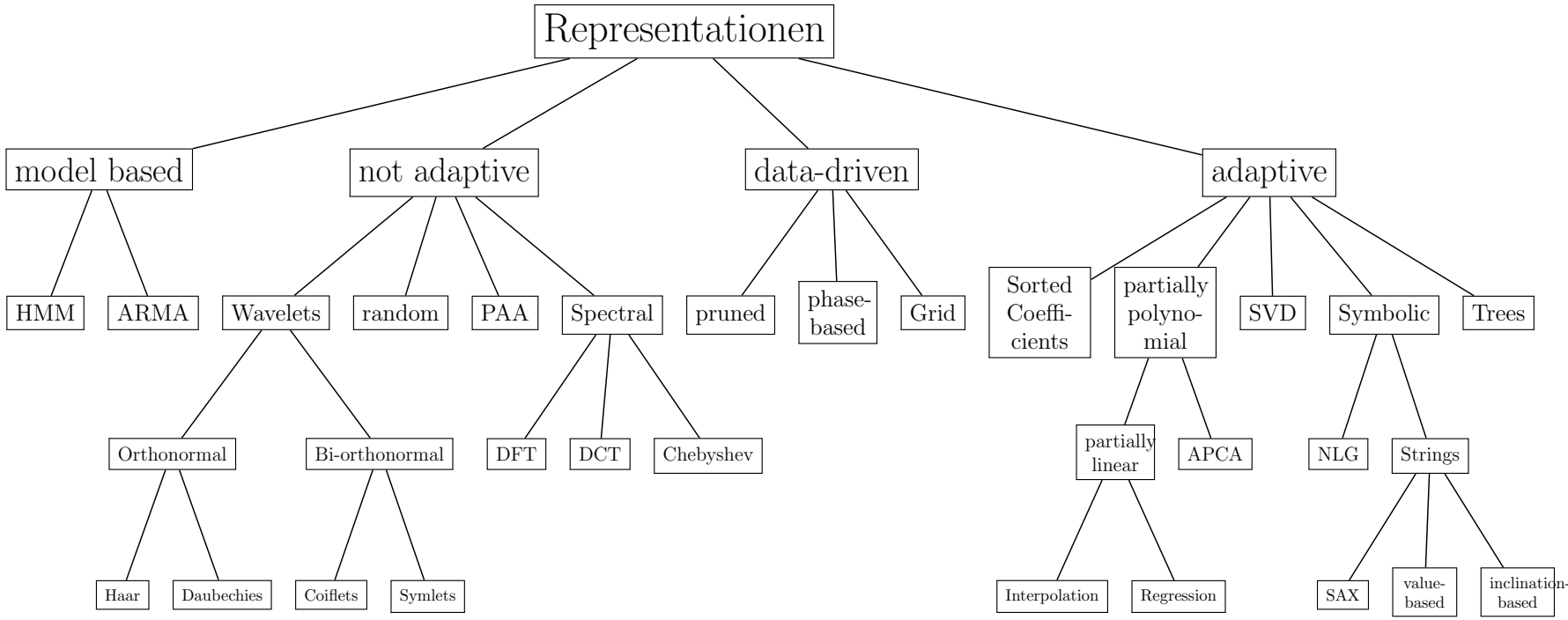
$$d(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

  between 2 standard normal distributed subsequences $Q = (q_1, \ldots, q_n)^T$ and $C = (c_1, \ldots, c_n)^T$
- Problem: many comparisons and memory capacity often too low to load all the required data

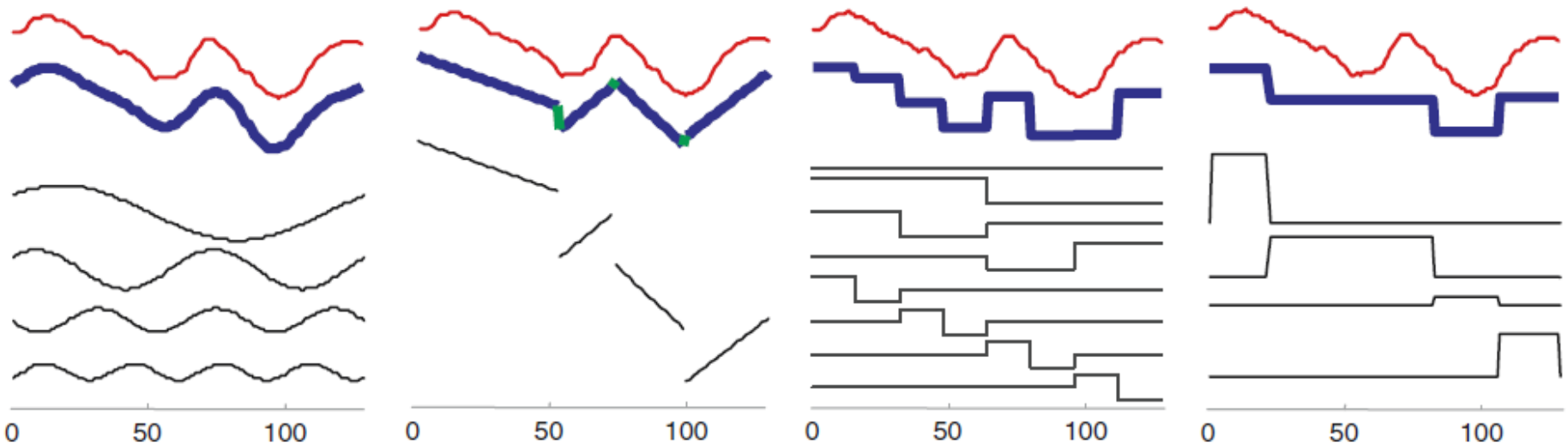
Memory Efficient Representation

- Problem: Many, slow accesses to data
- Solution: Approximation of time series, which fits into memory and keeps relevant (or interesting) features
- e.g. discrete fourier transformation (DFT), discrete wavelet transformation (DWT), partially linear approximation and adaptive, partially constant approximation (APCA), singular value decomposition (SVD)
- here: symbolic representationen
- Advantage: Algorithms from information retrieval and bioinformatics can be used (Hashing, markovian models, ...)

Time Series REpresentation

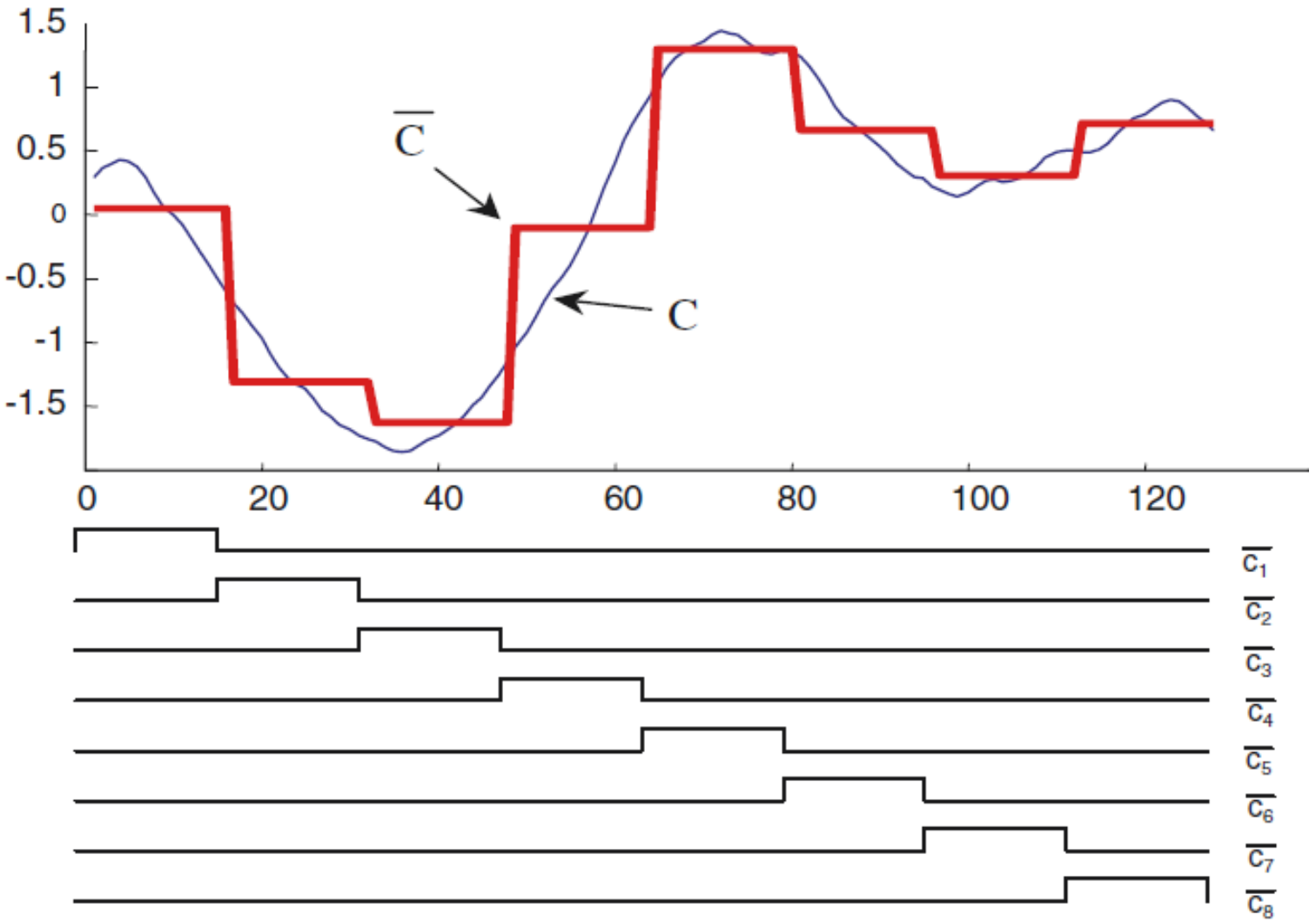


Most Commonly Used Representation



DFT PLA Haar wavelet APCA

Partially Aggregated Approximation (PAA)



Reduction from 128 to 8 Data points

Symbolic Aggregate Approximation (SAX)

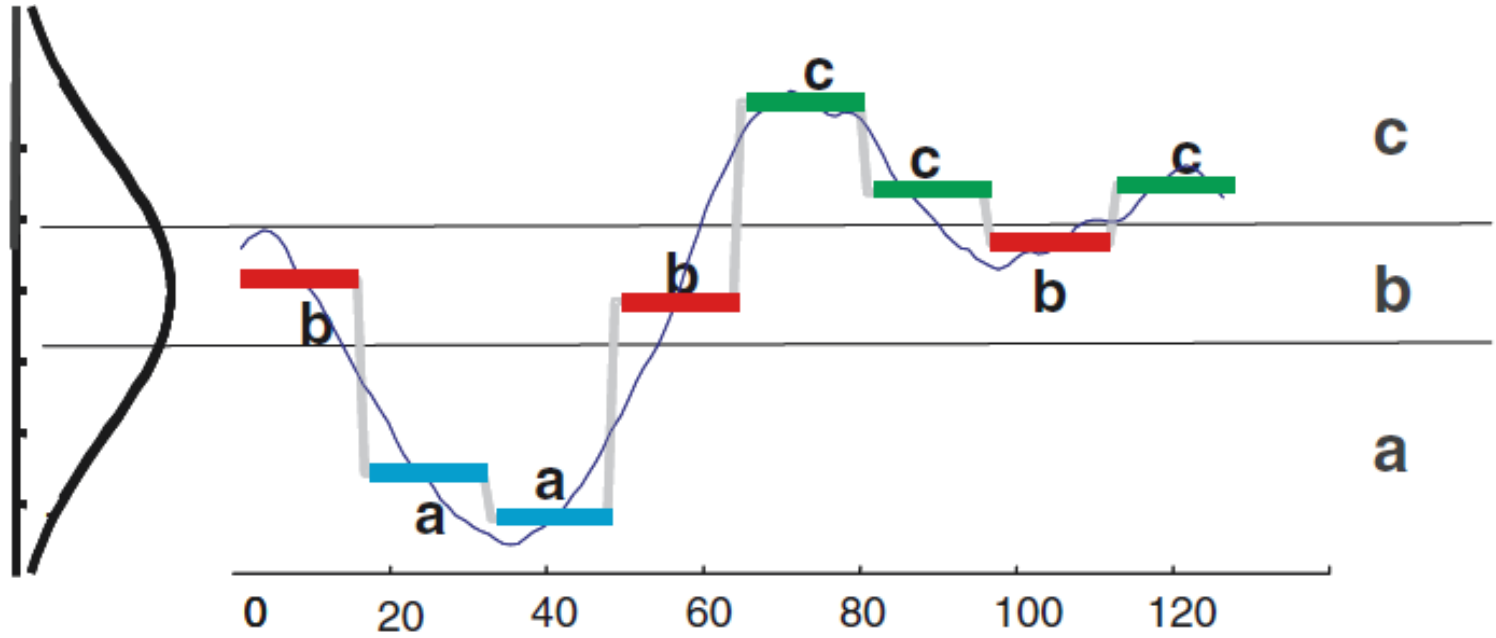
- for every sequence of length n a word with length w is defined (over an alpabet $A = \{\alpha_1, \ldots, \alpha_a\}$ with $|A| = a$
- simple Algorithm:
  1. Split up subsequence into w equally-sized intervals
  2. PAA: For each interval find a representativ (e.g. mean value)
     $C = (c_1, \ldots, c_n)^T$ is mapped to$\bar{C} = (\bar{c}_1, \ldots, \bar{c}_w)$ durch
     $$\bar{c}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j$$
  3. Map mean value $\bar{c}_i$ of $\bar{C}$ to one of the a letters by
     $$\hat{a}_i = \alpha_j \Leftrightarrow \beta_{j-1} \leq \bar{c}_i \leq \beta_j$$
- Assumption: Range of values of the PAA sequence is normally distributed and every occurence of a letter is equally likely
- Mapping $\bar{c}_i \mapsto b \in A$ by "sites of fracture" $\beta_1, \ldots, \beta_{a-1}$

"Sites of Fracture" of a Normal Distribution

$\lvert A\rvert$	3	4	5	6	7	8	9	10
β_1	$-.43$	$-.67$	$-.84$	$-.97$	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	0.25	0.43	0.57	0.67	0.76	0.84
β_3		0.67	0.25	0	$-.18$	$-.32$	$-.43$	$-.52$
β_4			0.84	0.43	0.18	0	$-.14$	$-.25$
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

- sites of fracture split normal distribution into equally probable regions

Example: SAX



- here: $n = 128, w = 8, a = 3$
- Result: **baabccbc**

Distance Measure for SAX

- PAA: lower bound to euclidean distance by

$$d_r(\bar{Q}, \bar{C}) = \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} (\bar{q}_i - \bar{c}_i)^2}$$

- SAX:

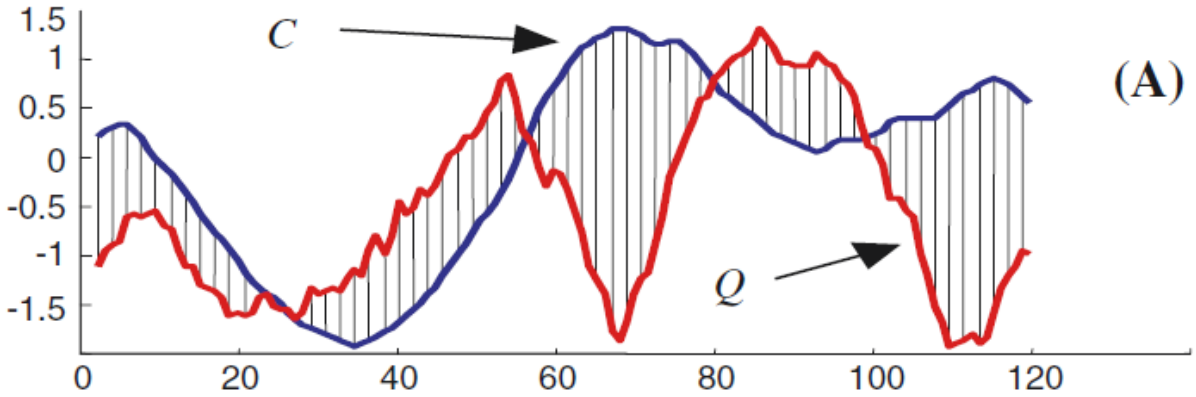
$$d^*(\hat{Q}, \hat{C}) = \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} d_a^*(\hat{q}_i, \hat{c}_i)^2}$$

- Distance measure d_a^* should be defined by a lookup table, e.g. for $a = 4$

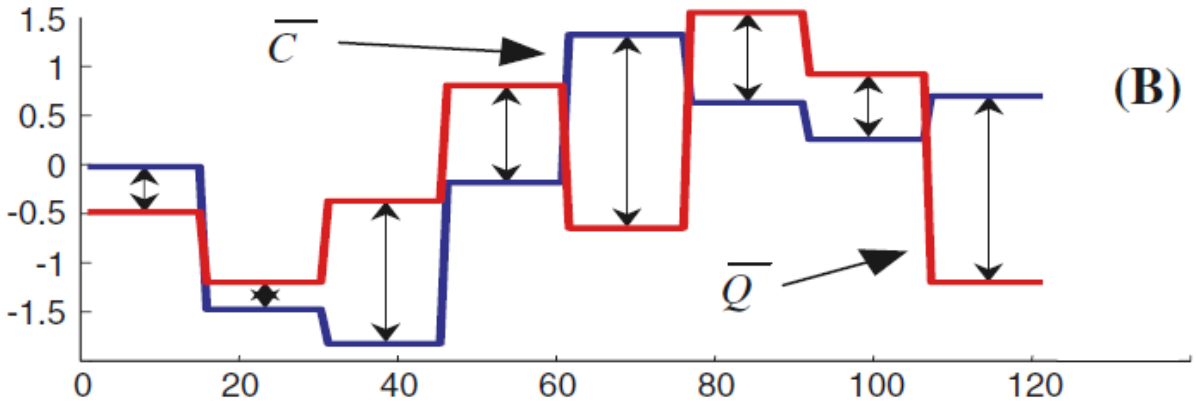
	a	**b**	**c**	**d**
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.340	0.67	0	0

$$d_a^*(r, c) = \begin{cases} 0 & \text{falls } |r - c| \leq 1, \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)} & \text{sonst} \end{cases}$$

Comparison of Distance Measures



(A)



(B)

$$\hat{C} = \mathbf{b\ a\ a\ b\ c\ c\ b\ c}$$

$$\hat{Q} = \mathbf{b\ a\ b\ c\ a\ c\ c\ a}$$

(C)

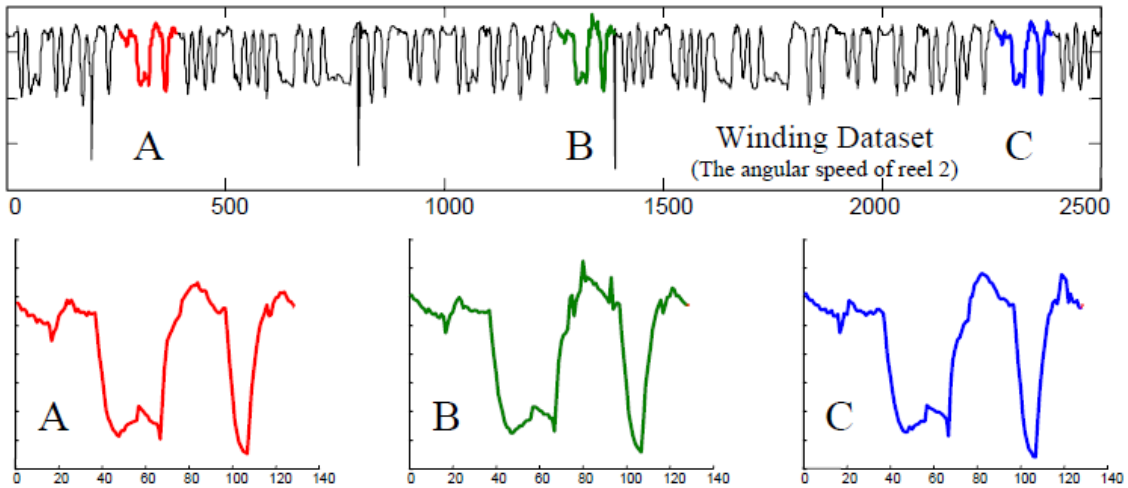
SAX-Advantage: Lower Bound

- $d^*(\hat{Q}, \hat{C})$ is *lower bound* of the euclidean distance $d(Q, C)$ of the original sequences Q and C

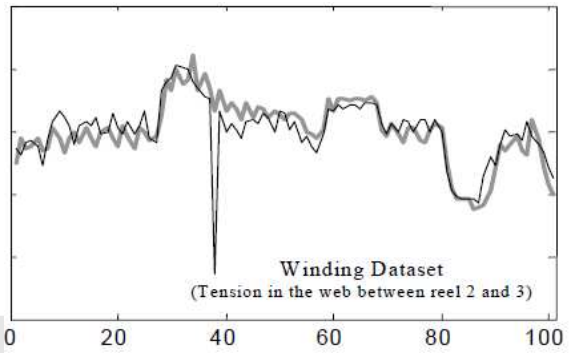
$$d^*(\hat{Q}, \hat{C}) \leq d(Q, C)$$

- if $\hat{Q}$ and $\hat{C}$ are dissimilar then Q and C are dissimilar as well
- SAX-based algorithms produce identical results compared to algorithms that work with original data
- "only" similar SAX words should be compared in the original feature space
- thus only few accesses to original data

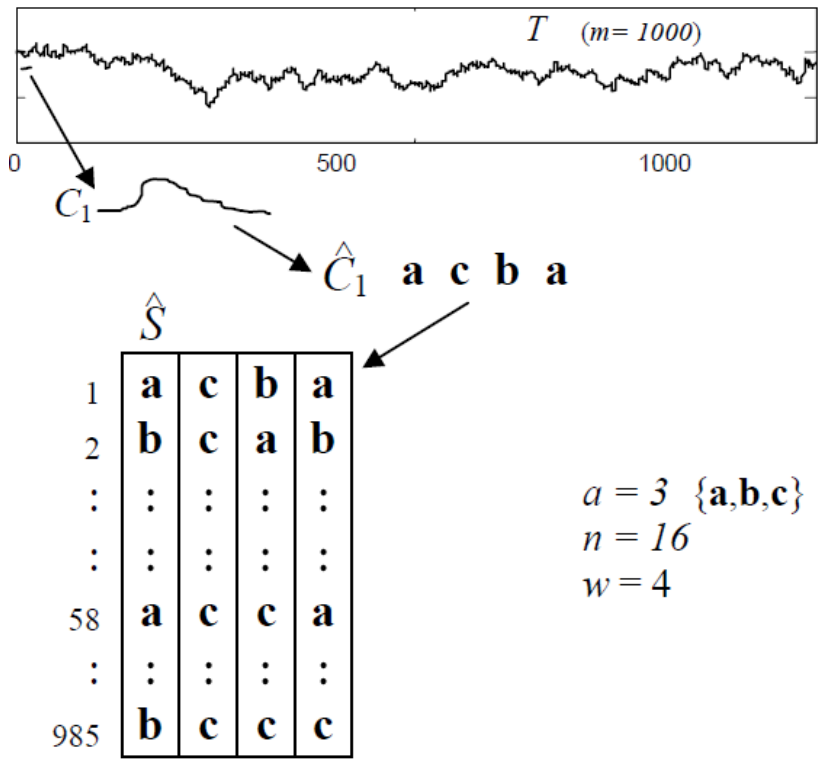
Finding Motifs in Time Series



- Motifs: Primitive, frequent (similar) patterns, prototypes
- Challenges:
  - Motifs are unknown beforehand
  - exhaustive search is too expensive with a complexity of $O(n^2)$
  - Outliers influence euclidean distance



Creating the SAX Matrix



- Find all motifs of a time series of length m by sliding windows
- Window lengths n leads to $(m-n+1)$ subsequences
- Transform every subsequence into a SAX word of length w
- Store in row matrix (so called SAX matrix)
- Matrix has w columns and $(m-n+1)$ rows

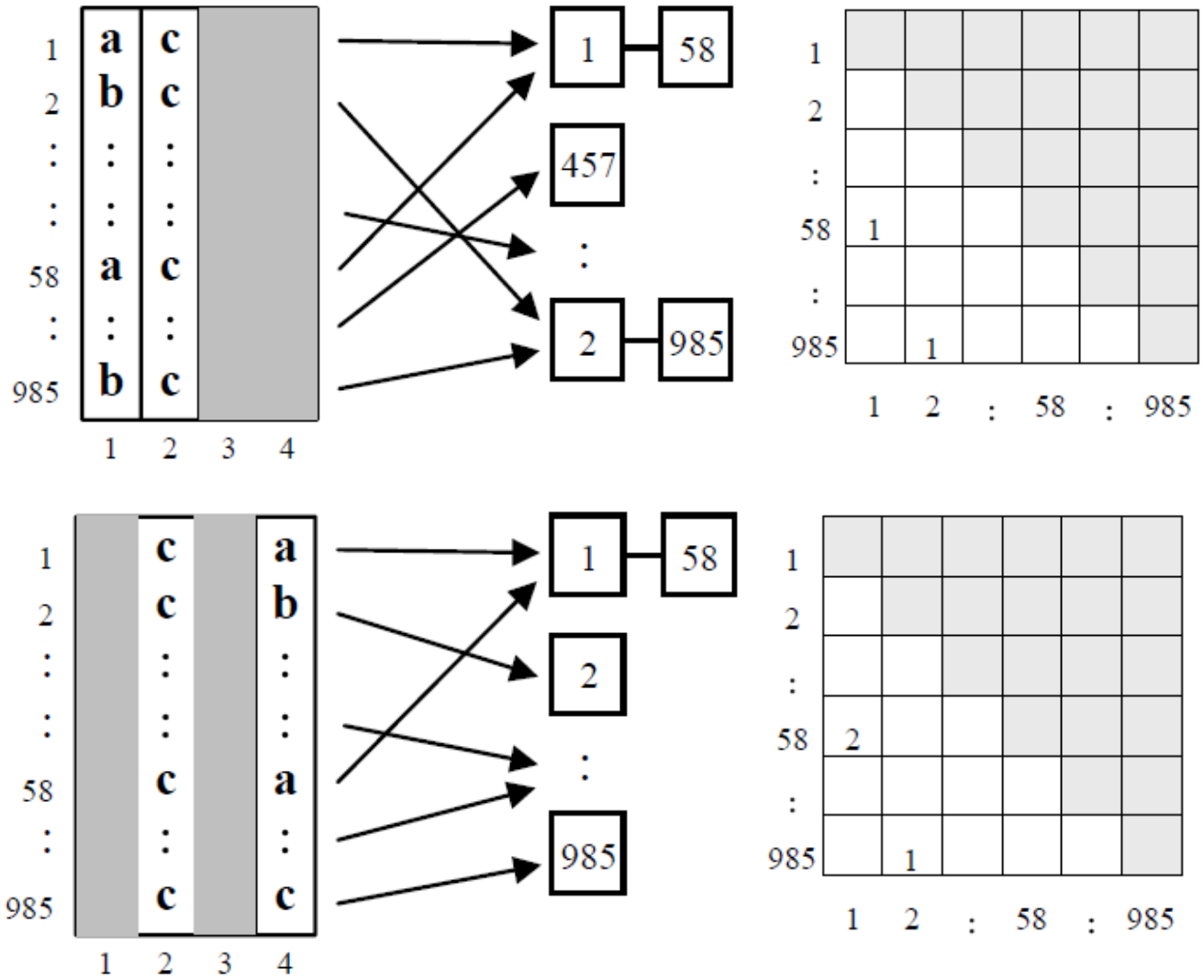
Random Projection

- Guess motif positions by so-called random projection
- pair-wise comparison of SAX words
- Collision matrix M with $(m - n + 1)^2$ cells for every comparison
- Implement M efficiently by a hash table
- At first $M(i, j) = 0$ for $1 \leq i, j \leq m - n + 1$
- Idea: Compare characters of two words in a SAX matrix with each other
- Better assumption: "don't care symbols" in sequences with unknown location
- E.G. noisy Motif or compression/expansion of a sequence

Random Projection

- Thus: SAX matrix is projected onto $1 \leq k < w$ randomly chosen columns
- Compare all rows of the projection
- if two projected SAX words in row i and j are identical, increment $M(i, j)$
- Projection is repeated t times, because some motifs will share an entry in M after some iterations
- It is unlikely that many random sequences will collide with an alredy found motif
- user-edefined threshhold s with $1 \leq s \leq k$ for collision entries in M
- All $M(i, j) \geq s$ are candidates for motifs
- But: the local neighbourhood of a sequence i contains many (so-called trivial) matches
- These are filtered at the end!

Random Projection



- first two iterations of a random projection

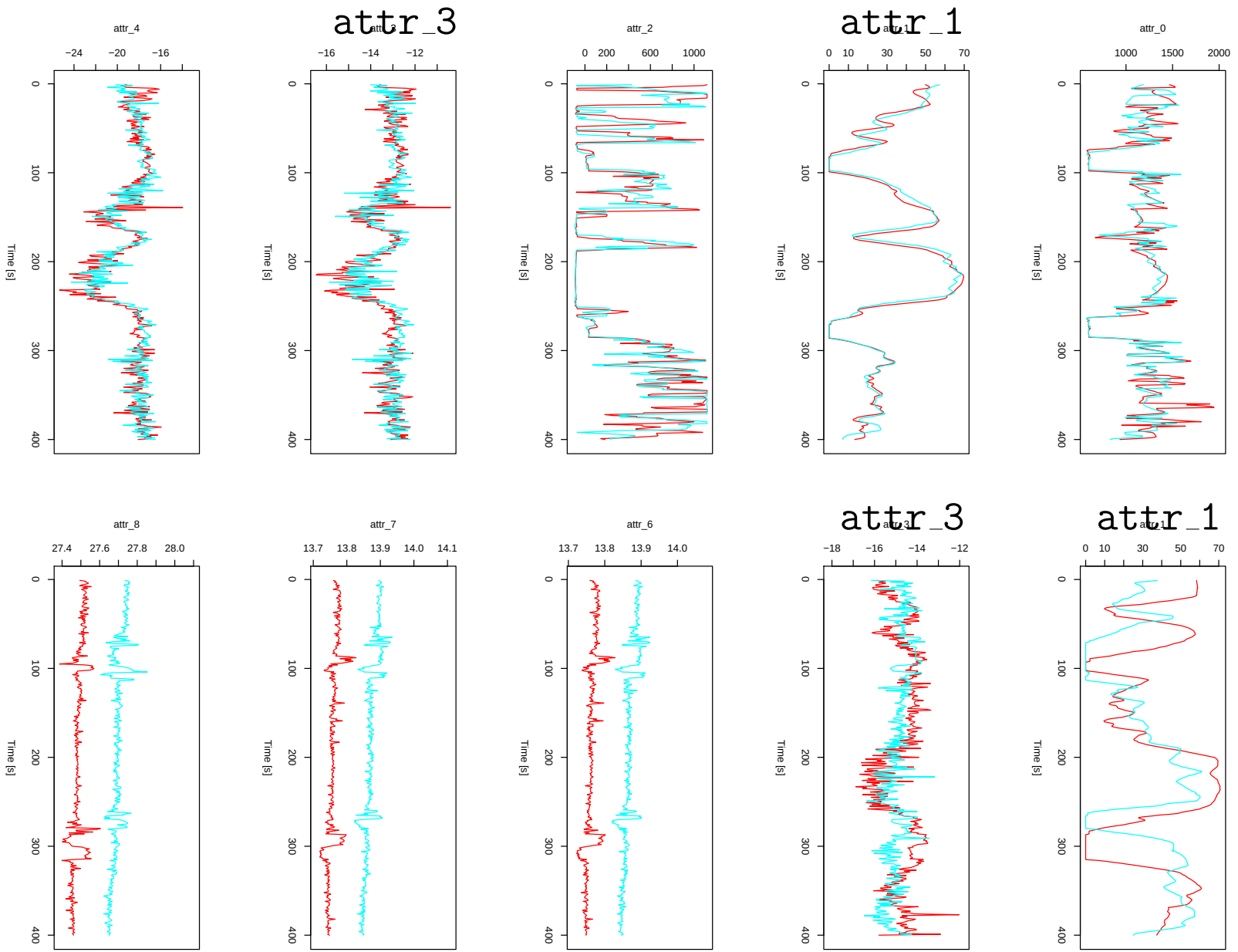
Subdimensional Motifs

- Random projections for motifs in univariate SAX time series can also be used in a multi-variate way
- Idea: increment the collision matrix M *for each attribute* $j \in \{1, \ldots, p\}$ for each projected SAX word
- Problem: relevant dimensions of potentional subdimensional motifs are unknown
- Solution:
  - Estimate a distribution $P(d_j)$ over distances between non-trivial matches by drawing a sample
  - Determine the distances $d_1^*, \ldots, d_p^*$ for each entry $M(i, j) \geq s$
  - if $P(d_j \leq d_j^*) < r_j^{\text{rel}}$ (user-specified *dimension relevance*), then every jth attribute is relevant

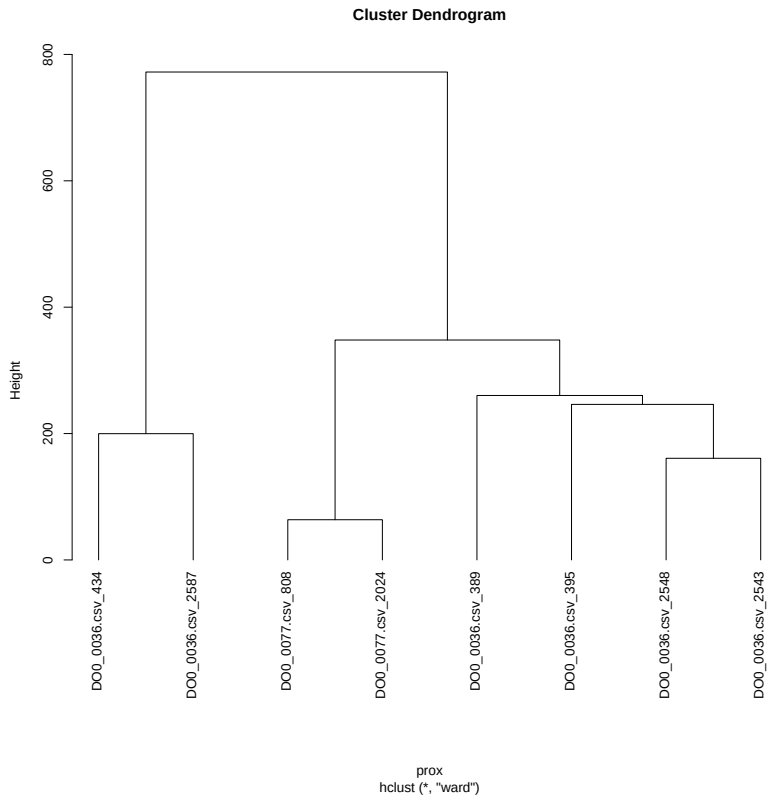
Example

- Expert identifies $p = 9$ of a total of 130 channels as important
- Motif lasts at least $n = 400\,\text{ms}$
- 10 time series are given to search for subdimensional motifs

Subdimensional Motif in Two Time Series



Clustering of Motivs



- Calculate dissimilarity matrix by pairwise comparison of all found patterns in the 10 time series based on d^*
- Matrix is symetric, positive and contains only zeros on its principal diagonal
- Can be used for grouping the occurences to find motifs that occur in several time series
- Here: hierarchical, agglomerative clustering of all motifs, that contained the attributes `attr_1` `attr_3`