Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series

Kevin Wellenzohn[1] Michael H. Böhlen[1]
Anton Dignös[2] Johann Gamper[2] Hannes Mitterer[2]

[1]Department of Computer Science
University of Zurich

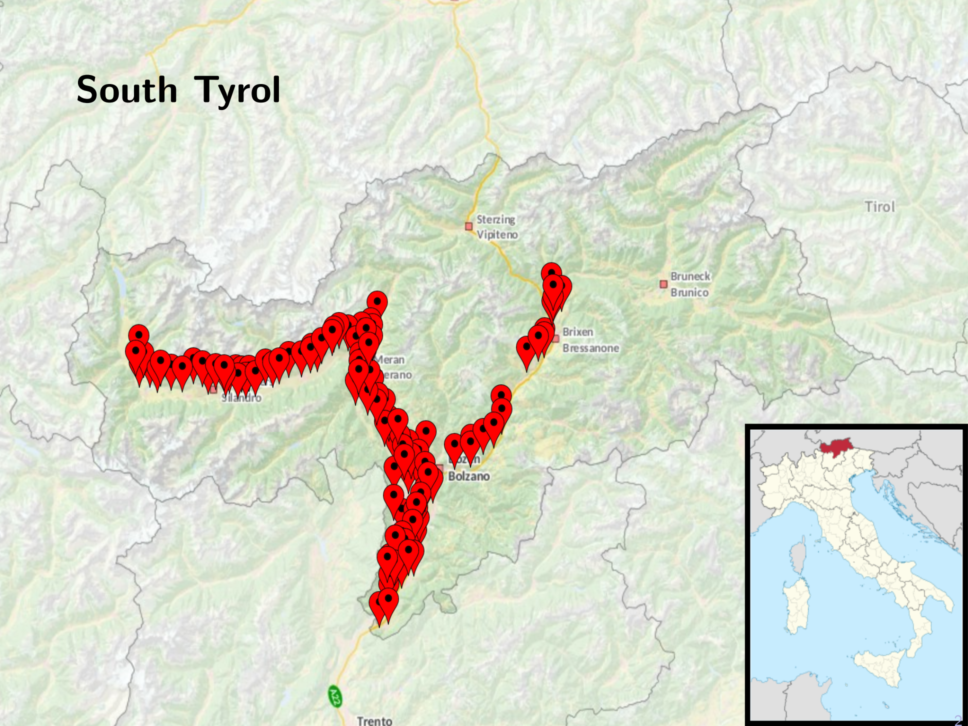
[2]Faculty of Computer Science
Free University of Bolzano

March 24, 2017

University of Zurich UZH

unibz
Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

South Tyrol


Sterzing
Vipiteno
Tirol
Bruneck
Brunico
Brixen
Bressanone
Silandro
Bolzano
A22
Trento

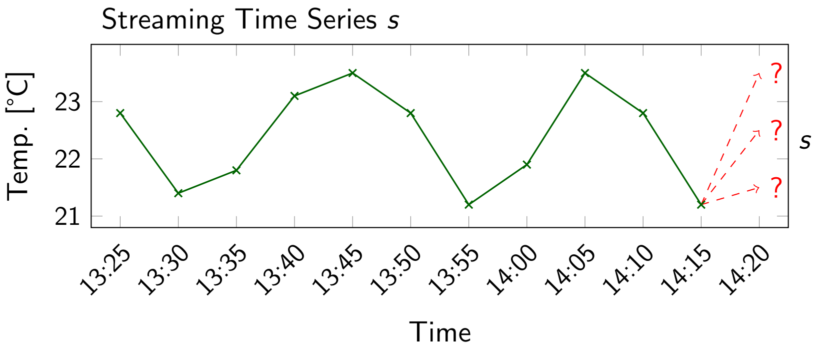
Overview

Problem. Streaming time series often have **missing values**, e.g. due to sensor failures or transmission delays!

Goal. Accurately **impute** (i.e. recover) the latest measurement by exploiting the **correlation** among streams.

Challenge. Streaming time series are often **non-linearly correlated**, e.g. due to **phase shifts**.

Example



- The latest value at time 14:20 is **missing** and needs to be **imputed** (i.e. recovered).

Approach

Top-k Case Matching (TKCM)

Intuition. Impute a missing value in time series s with past values from s when a set of correlated **reference time series** exhibited similar **patterns**.

Top-k Case Matching (TKCM)

Intuition. Impute a missing value in time series s with past values from s when a set of correlated **reference time series** exhibited similar **patterns**.

Imputation Steps:

1. Draw query pattern over most recent values

Top-k Case Matching (TKCM)

Intuition. Impute a missing value in time series s with past values from s when a set of correlated **reference time series** exhibited similar **patterns**.

Imputation Steps:

1. Draw query pattern over most recent values
2. Find k most similar non-overlapping patterns

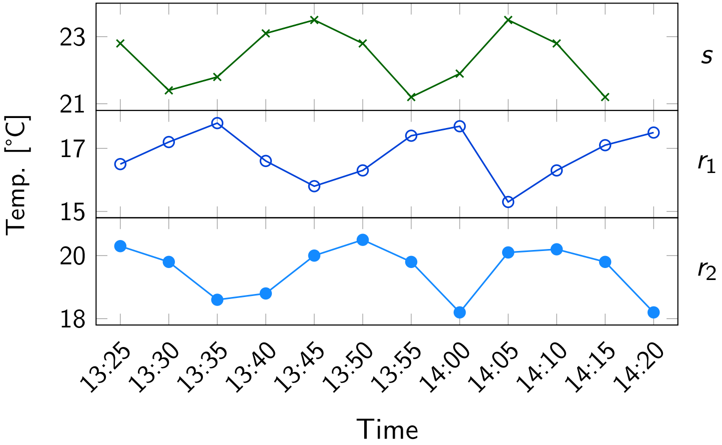
Top-k Case Matching (TKCM)

Intuition. Impute a missing value in time series s with past values from s when a set of correlated **reference time series** exhibited similar **patterns**.

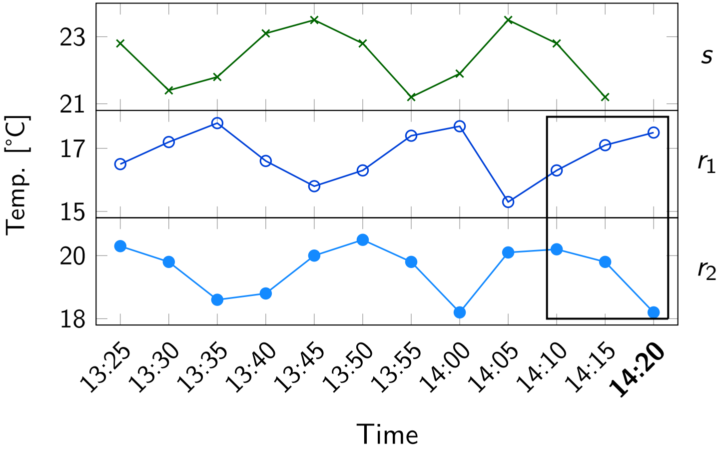
Imputation Steps:

1. Draw query pattern over most recent values
2. Find k most similar non-overlapping patterns
3. Impute missing value using the k most-similar patterns

Applying TKCM

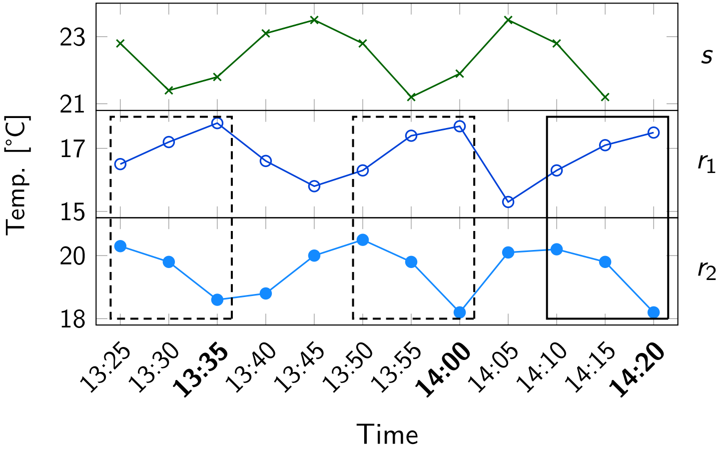


Applying TKCM



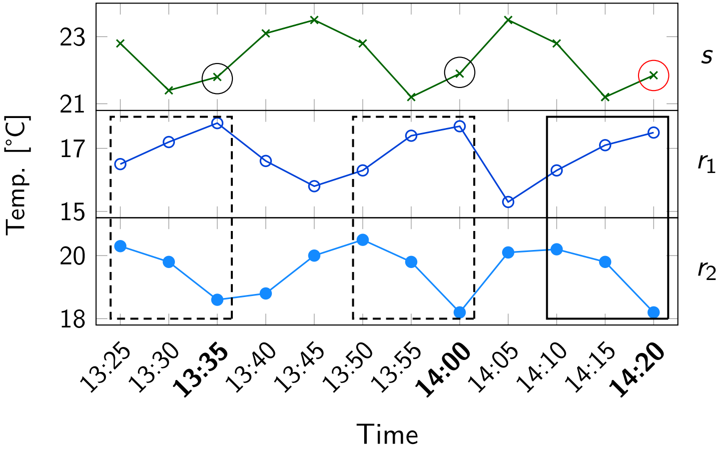
1. Define query pattern $P(14{:}20)$ over $d = 2$ reference time series $\{r_1, r_2\}$ in a time frame of $l = 10$ minutes

Applying TKCM



2. The $k = 2$ most similar non-overlapping patterns are $P(14{:}00)$ and $P(13{:}35)$

Applying TKCM



3. Missing value is imputed as
$\hat{s}(14{:}20) = \frac{1}{2}(s(14{:}00) + s(13{:}35)) = 21.85°C$

Query Pattern

Pattern length $l = 3$

16.3	17.1	17.5	r_1
20.2	19.9	18.2	r_2
14:10	14:15	14:20	

reference time series $d = 2$

- With $l > 1$, TKCM takes the temporal context into account and captures how time series change over time
- Pattern length l is important to deal with **non-linear correlations**

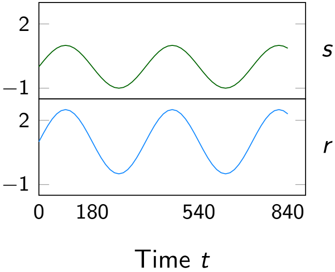
Related Work

1. **Centroid Decomposition (CD)**
   - M. Khayati, M. H. Böhlen, and J. Gamper. Memory-efficient centroid decomposition for long time series. ICDE 2014
   - Singular Value Decomposition (SVD) that expects **linear correlations**
2. **SPIRIT**
   - S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. VLDB 2005
   - Principal Component Analysis (PCA) that expects **linear correlations**
3. **MUSCLES**
   - B. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. ICDE 2000
   - Multi-variate linear regression that expects **linear correlations**

Linear vs. Non-Linear Correlations

Linear Correlations

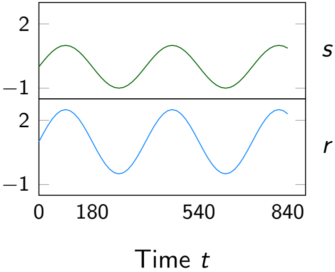
$s(t) = \text{sind}(t)$
$r(t) = 1.5 \times \text{sind}(t) + 1$

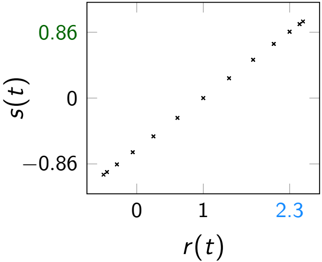


- Time series s and r have different **amplitude** and **offset**

Linear Correlations

$s(t) = \text{sind}(t)$
$r(t) = 1.5 \times \text{sind}(t) + 1$

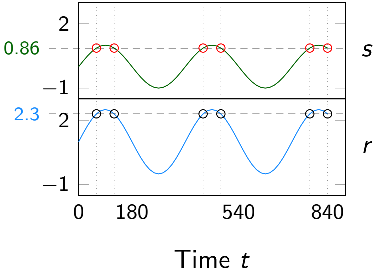


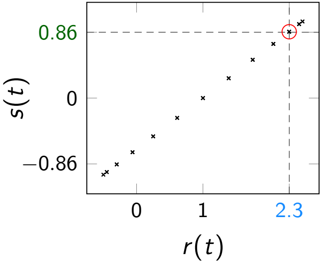


- Time series s and r have different **amplitude** and **offset**
- They are **linearly correlated** and their Pearson Correlation Coefficient is 1!

Linear Correlations

$s(t) = \text{sind}(t)$
$r(t) = 1.5 \times \text{sind}(t) + 1$

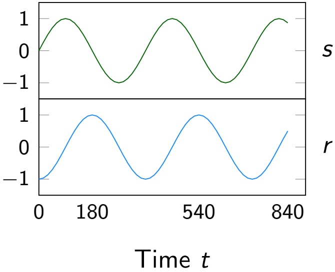


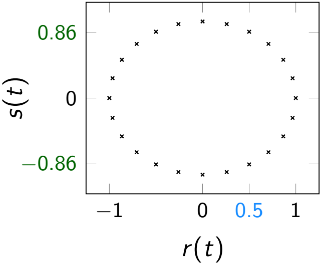


- Time series s and r have different **amplitude** and **offset**
- They are **linearly correlated** and their Pearson Correlation Coefficient is 1!

Non-Linear Correlations

$$s(t) = \text{sind}(t)$$
$$r(t) = \text{sind}(t - 90)$$

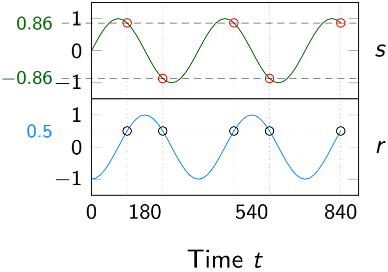


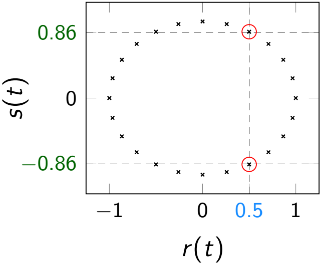


- Time series s and r are **phase-shifted** by 90 degrees
- They are **non-linearly correlated** and their Pearson Correlation Coefficient is 0!

Non-Linear Correlations

$$s(t) = \text{sind}(t)$$
$$r(t) = \text{sind}(t - 90)$$

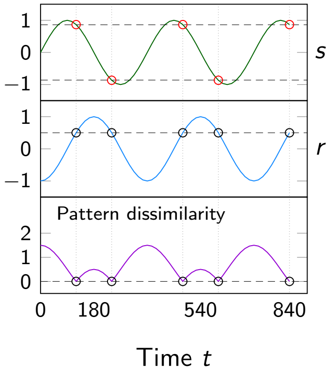




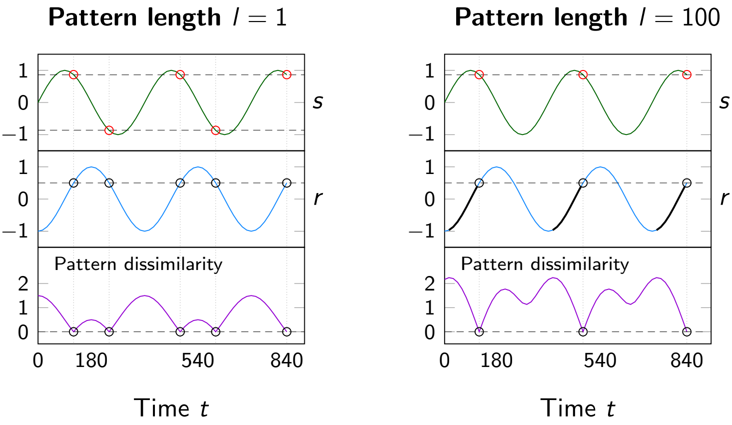
- Time series s and r are **phase-shifted** by 90 degrees
- They are **non-linearly correlated** and their Pearson Correlation Coefficient is 0!

Pattern Length l and Non-Linear Correlations

Pattern length $l = 1$



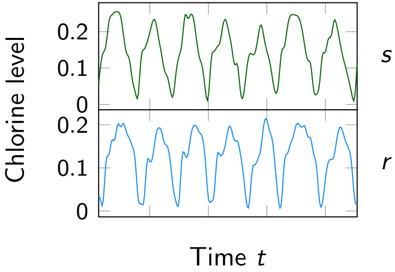
Pattern Length l and Non-Linear Correlations

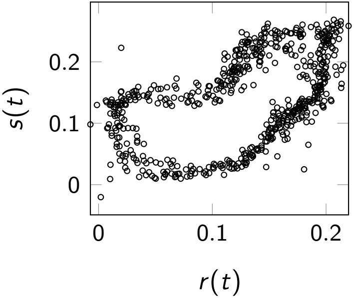


- With $l > 1$ there are less patterns with pattern dissimilarity 0

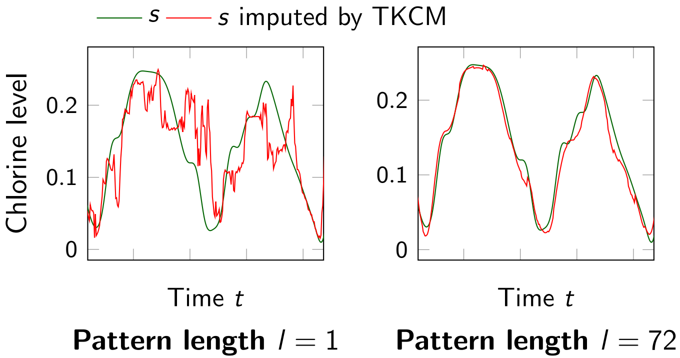
Chlorine Dataset

- Chlorine dataset is **phase-shifted** and hence **non-linearly correlated**





Importance of Pattern Length l



- A larger pattern length decreases the oscillation in the imputed time series

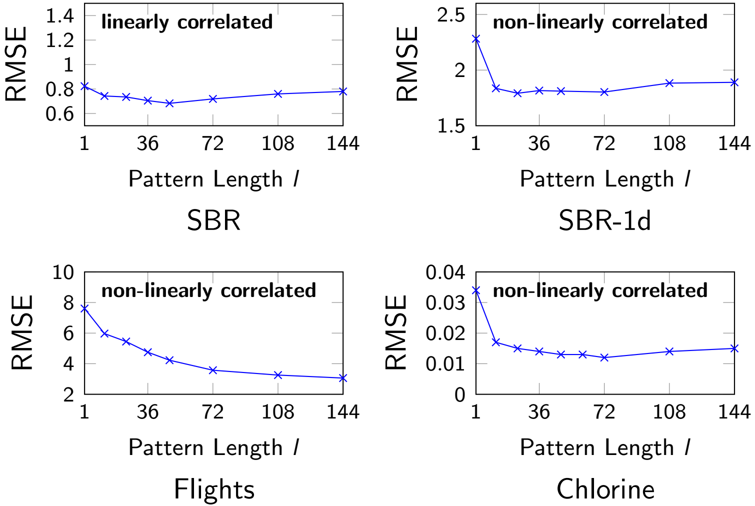
Experiments

Datasets

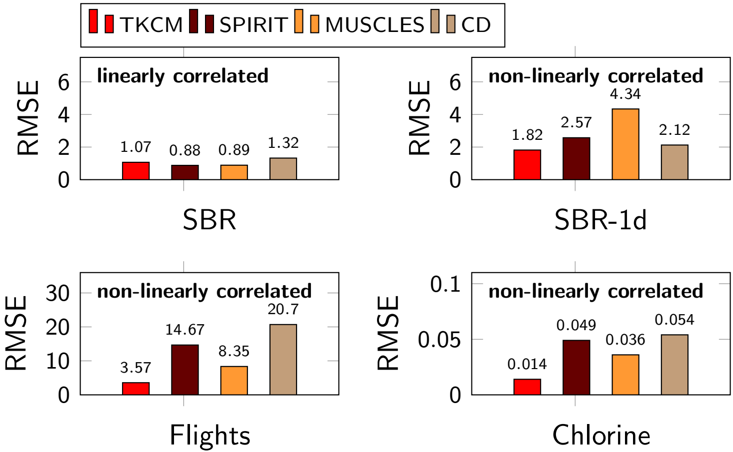
We use 4 datasets:

1. SBR
   - 130 meteorological time series from South Tyrol
   - **linearly correlated**
2. SBR-1d
   - SBR dataset shifted up to 1 day
   - **non-linearly correlated**
3. Flights
   - 8 time series
   - **non-linearly correlated**
4. Chlorine
   - 166 time series
   - **non-linearly correlated**

Pattern Length l



Comparison



- TKCM is more accurate on all **non-linearly correlated** datasets (SBR-1d, Flights, and Chlorine).

Conclusion & Future Work

Conclusion

- TKCM imputes the current missing value in a stream using reference time series
- TKCM exploits **linear** and **non-linear correlations** among time series

Future work

- Automatically choose reference time series
- Improve efficiency of TKCM by pruning candidate patterns

Thanks!