Algorithmic Data Analysis

Esther Galbrun
Autumn 2022
Part V

Mining Time-Series
Data Preparation
Interpolation

Time-series might contain missing values. When data is collected from independent sensors, the values might not be synchronized.

**Linear interpolation** can be used to produce a time-series with equally spaced, synchronized values, easier to manipulate.

If \( x_i \) and \( x_j \) are values at timestamps \( t_i \) and \( t_j \) respectively, the value for timestamp \( t \), such that \( t_i \leq t \leq t_j \), is estimated as:

\[
x = x_i + \frac{t - t_i}{t_j - t_i} \cdot (x_j - x_i)
\]
Interpolation

Time-series might contain missing values
When data is collected from independent sensors, the values might not be synchronized

**Linear interpolation** can be used to produce a time-series with equally spaced, synchronized values, easier to manipulate

More complex methods such as *polynomial interpolation* or *spline interpolation* can also be used
They require more data points for the estimation and often do not provide significantly improved results
Interpolation

Time-series might contain missing values
When data is collected from independent sensors, the values might not be synchronized

Linear interpolation can be used to produce a time-series with equally spaced, synchronized values, easier to manipulate

The result is a time-series $S_x = \langle x_1, x_2, \ldots, x_n \rangle$, with values at each of $n$ equally spaced timestamps $t_1, \ldots, t_n$

i.e. such that $t_i = t_1 + (i - 1) \delta$
for $i = 2, \ldots, n$ and some time step $\delta$
Example

Linear interpolation, for equally spaced time points

\(\langle 0, 5, 10, 15, 20, 25 \rangle\)
Noise removal

Sensors used to collect data can be noise-prone.

Noise removal aims to remove short-term fluctuations.

The distinction between noise and interesting outliers can be difficult to make, in general.

Outliers result from fluctuations during data generation.

Noise are caused by artifacts of the data collection process.

Noise removal approaches include binning and smoothing.
Consider a time-series $S_X = \langle x_1, x_2, \ldots, x_n \rangle$, with values at each of $n$ equally spaced timestamps $t_1, \ldots, t_n$

**Binning**, a.k.a. *piecewise aggregate approximation (PAA)*, divides the time-series into time intervals of size $k$, i.e. into intervals $[t_1, t_k]$, $[t_{k+1}, t_{2k}]$, $\ldots$, $[t(\lfloor n/k \rfloor - 1)k+1, t_{\lfloor n/k \rfloor k}]$

Binned values are averages of values within each interval

$$y_i = \frac{1}{k} \sum_{r=1}^{k} x_{(i-1)k+r} \quad \text{for } i = 1, \ldots, \lfloor n/k \rfloor$$

Instead of average, it is possible to take the median, which is more robust to the presence of outlier values

Binning is lossy, reduces the number of points by a factor of $k$
IBM stock prices from Sept. 2013 to Sept. 2014

Original time-series
Example

**IBM** stock prices from Sept. 2013 to Sept. 2014

Binning, *k* = 3
Example

IBM stock prices from Sept. 2013 to Sept. 2014

Binning, $k = 5$
Data preparation

\[
\langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10 \rangle
\]

Binning, window of width 4

\[
\langle 22, 11.5, 11, 6 \rangle
\]
Moving-average smoothing

Consider a time-series $S_x = \langle x_1, x_2, \ldots, x_n \rangle$, with values at each of $n$ equally spaced timestamps $t_1, \ldots, t_n$

Moving-average smoothing uses overlapping bins of size $k$, i.e. intervals $[t_1, t_k], [t_2, t_{k+1}], \ldots, [t_{n-k+1}, t_n]$

Smoothed values are averages of values within each interval

$$y_i = \frac{1}{k} \sum_{r=0}^{k-1} x_{i+r} \quad \text{for } i = 1, \ldots, n - k + 1$$

In real-time applications, the smoothed value becomes available after the last timestamp in the interval, creating a lag

Wider intervals lead to increased smoothing and lag
Moving-average smoothing

Consider a time-series \( S_X = \langle x_1, x_2, \ldots, x_n \rangle \), with values at each of \( n \) equally spaced timestamps \( t_1, \ldots, t_n \)

Moving-average smoothing uses overlapping bins of size \( k \), i.e. intervals \([t_1, t_k], [t_2, t_{k+1}], \ldots, [t_{n-k+1}, t_n]\)

Smoothed values are averages of values within each interval

\[
y_i = \frac{1}{k} \sum_{r=0}^{k-1} x_{i+r} \quad \text{for } i = 1, \ldots, n - k + 1
\]

Because of the lag, the smoothed time-series might contain uptrends where the original data contains downtrends and vice-versa, causing misleading interpretations of recent trends
Example

IBM stock prices from Sept. 2013 to Sept. 2014

Moving-average smoothing, $k = 3$
Example

IBM stock prices from Sept. 2013 to Sept. 2014
Moving-average smoothing, $k = 5$
Example

IBM stock prices from Sept. 2013 to Sept. 2014
Moving-average smoothing, $k = 9$
Data preparation

\(\langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10\rangle\)

Moving average smoothing, window of width 4

\(\langle 22, 21.25, 16, 12.5, 11.5, 11.25, 10.5, 11.75, 11, 9.75, 9.5, 6.75, 6\rangle\)
Consider a time-series $S_X = \langle x_1, x_2, \ldots, x_n \rangle$, with values at each of $n$ equally spaced timestamps $t_1, \ldots, t_n$

In **exponential smoothing**, the current smoothed value is defined as a linear combination of the current original value and the previous smoothed value.

For smoothing parameter $\alpha \in [0, 1]$ and letting $y_1 = x_1$

$$y_i = \alpha \cdot x_i + (1 - \alpha) \cdot y_{i-1} \quad \text{for } i = 2, \ldots, n$$
Consider a time-series $S_X = \langle x_1, x_2, \ldots, x_n \rangle$, with values at each of $n$ equally spaced timestamps $t_1, \ldots, t_n$

For smoothing parameter $\alpha \in [0, 1]$ and letting $y_1 = x_1$

$$y_i = \alpha \cdot x_i + (1 - \alpha) \cdot y_{i-1} \quad \text{for } i = 2, \ldots, n$$

The smoothed values can be expressed as an exponentially decayed sum of the original values, giving more importance to recent values.

The smoothing parameter $\alpha$ regulates the decay factor. Setting $\alpha = 1$ means there is no smoothing, the resulting series is identical to the original. Setting $\alpha = 0$ results in smoothing the whole series to the constant value of $x_1$.
IBM stock prices from Sept. 2013 to Sept. 2014
Exponential smoothing, $\alpha = 0.75$
Example

IBM stock prices from Sept. 2013 to Sept. 2014

Exponential smoothing, $\alpha = 0.25$
Data preparation

\[\langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10 \rangle\]

Exponential smoothing, \(\alpha = 0.6\)

\[\langle 13.00, 23.80, 23.92, 21.57, 14.63, 11.85, 10.74, 13.90, 10.96, 8.58, 12.43, 12.77, 7.51, 6.60, 5.04, 8.02 \rangle\]
Normalization

When multiple time-series containing values that are measured on different scales are analysed simultaneously, they might need to be normalized to allow meaningfully comparing relative trends rather than absolute values.

Given a time-series \( S_X = \langle x_1, x_2, \ldots, x_n \rangle \), taking values in a bounded range \([v_{\text{min}}, v_{\text{max}}]\), range-based normalization maps the original time-series values to new values in the range \([0, 1]\)

\[
y_i = \frac{x_i - v_{\text{min}}}{v_{\text{max}} - v_{\text{min}}}
\]
Normalization

When multiple time-series containing values that are measured on different scales are analysed simultaneously, they might need to be normalized to allow meaningfully comparing relative trends rather than absolute values.

Given a time-series $S_X = \langle x_1, x_2, \ldots, x_n \rangle$, with mean $\mu$ and standard deviation $\sigma$, **standardization** maps the original time-series values to new values:

$$ y_i = \frac{x_i - \mu}{\sigma} $$

No guaranteed specific range for the resulting values.
IBM, Cisco and Apple stock prices from Sept. 2013 to Sept. 2014

Original time-series
IBM, Cisco and Apple stock prices from Sept. 2013 to Sept. 2014
Range-based normalized time-series
IBM, Cisco and Apple stock prices from Sept. 2013 to Sept. 2014

Standardized time-series
Data preparation

\[ \langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10 \rangle \]

Range-based normalization

\[ \langle 0.33, 1.00, 0.74, 0.59, 0.22, 0.22, 0.22, 0.44, 0.19, 0.11, 0.41, 0.33, 0.00, 0.07, 0.00, 0.22 \rangle \]
Data preparation

\[ \langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10 \rangle \]

Standardization

\[ \langle 0.05, 2.59, 1.60, 1.04, -0.37, -0.37, -0.37, 0.48, -0.51, -0.79, 0.34, 0.05, -1.22, -0.93, -1.22, -0.37 \rangle \]
Discretization

Time-series can be converted into discrete sequences by discretizing the behavioral attributes

Transform time-point values into interval-based representation

**value abstraction** (absolute)
- dividing the range of a variable into bins
  - e.g. \{low, medium, high\}

**trend abstraction** (relative)
- looking at the local behavior of the variable
  - e.g. \{decreasing, stable, increasing\}
Discretization

Symbolic aggregate approximation (SAX)

1. Average values over successive equally spaced windows, i.e. compute piecewise aggregate approximation (PAA)
2. Convert the resulting continuous values to a small number of discrete values

Select breakpoints such that symbols have approximately equal occurrence frequencies
Use Gaussian distribution assumption for long time-series or in the streaming setting

SAX is a lower-bounding approach, i.e. it allows distance measures to be defined on the symbolic representation that lower-bounds the distance in the original representation
IBM stock prices from Sept. 2013 to Sept. 2014

Original time-series
IBM stock prices from Sept. 2013 to Sept. 2014

Binning, $k = 5$
IBM stock prices from Sept. 2013 to Sept. 2014

Discretizing

Example
Transforms
Transforms map the data into a different representation space
More convenient representation for evaluating similarity
The dimensionality of the data can be reduced while retaining most of the information by selecting a subset of the dimensions of the new representation space
If we want to analyse different vegetable soups
A drop by drop comparison of soups is difficult
Instead we convert the soup to its recipe, i.e. proportions of the different vegetables it contains
Recipes are easier to analyse, modify and compare

- Simplify a soup by keeping only the main vegetables
- Compare soups by comparing the proportions of different vegetables in their respective recipes
Transforms: an analogy

How to find the recipe?
Imagine we have filters corresponding to different vegetables i.e. we have a potato filter, a carrot filter, an onion filter, etc.
Pour a soup into a filter to extract the associated vegetable

How to reconstruct the soup?
Simply blend the ingredients back together
Transforms: an analogy

• Filters must be *complete*, there must be a dedicated filter for every ingredient possibly involved.

• Filters must be *independent*, modifying the amount of one vegetable should affect the result of the associated filter but not the results of other filters.

• Separating and combining the ingredients in any order must always give the same result.
Annual copper prices during the early 19th century
Discrete wavelet transform (DWT)

Adjacent values in the time-series are often very similar, storing all the values is wasteful, redundant
The average value alone provides a very crude representation of the time-series, without any information about variations.
Discrete wavelet transform (DWT)

Adding the difference between first and second half allows to reconstruct the average values during both halves.
Adding the difference between first and second quarter as well as between third and fourth quarter...
Discrete wavelet transform (DWT)

This process can be applied recursively...
Discrete wavelet transform (DWT)

This process can be applied recursively...

![Chart showing a recursive application of DWT with time-series data points and coefficients.](chart.png)
Discrete wavelet transform (DWT)

This process can be applied recursively...
Discrete wavelet transform (DWT)

This process can be applied recursively...
For simplicity, assume the length $n$ of the series is a power of 2.

The decomposition defines $2^{k-1}$ weights of order $k$, for $k = 1, \ldots, \log_2(n)$.

Let $\Psi(k, i)$ be the $i^{th}$ weight of order $k$, corresponding to the segment of the time-series between positions

\[
\frac{(i - 1) \cdot n}{2^{k-1}} + 1 \quad \text{and} \quad \frac{i \cdot n}{2^{k-1}}
\]

Let $\Phi(k, i)$ be the average value of this segment

\[
\Psi(k, i) = \frac{\Phi(k + 1, 2i - 1) - \Phi(k + 1, 2i)}{2}
\]

$\Phi(1, 1)$ is the global average.
Discrete wavelet transform (DWT)

This process decomposes the time-series into a collection of wavelets with different widths, offsets and weights.
Discrete wavelet transform (DWT)

This process decomposes the time-series into a collection of wavelets with different widths, offsets and weights.
Discrete wavelet transform (DWT)

The original time-series can be reconstructed by summing all the weighted wavelets.
Discrete wavelet transform (DWT)

Each row of matrix $W$ contains a basis vector, i.e. a wavelet.

Vector $a$ contains the weights for the different wavelets.

\[ a = \begin{bmatrix} -0.19 \\ 0.00 \\ 0.04 \\ 0.60 \\ -0.05 \\ 0.08 \\ -0.24 \\ 0.72 \end{bmatrix} \]

\[ W = \begin{bmatrix} -0.19 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.00 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0.04 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0.60 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ -0.05 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0.08 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ -0.24 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 0.72 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \]
Discrete wavelet transform (DWT)

The original time-series can be reconstructed as $a^T \cdot W$

$$a = \begin{bmatrix} -0.19 \\ 0.00 \\ 0.04 \\ 0.60 \\ -0.05 \\ 0.08 \\ -0.24 \\ 0.72 \end{bmatrix} \quad W = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$a^T \cdot W = \begin{bmatrix} 0.24 & 0.62 & 0.53 & 0.53 & 1.08 & 1.00 & 1.48 & 0.28 \end{bmatrix}$$
Discrete wavelet transform (DWT)

The original time-series can be reconstructed as \( a^T \cdot W \)

\[
S = a^T \cdot W = \sum_{i=1}^{n} a_i w^{(i)} = \sum_{i=1}^{n} a_i \frac{w^{(i)}}{||w^{(i)}||}
\]

\( a_i ||w^{(i)}|| \) are the normalized weights
\( w^{(i)}/||w^{(i)}|| \) are the normalized basis vectors
Discrete wavelet transform (DWT)

The original time-series can be reconstructed as $a^T \cdot W$

$$S = a^T \cdot W = \sum_{i=1}^{n} a_i w^{(i)} = \sum_{i=1}^{n} a_i \frac{w^{(i)}}{\|w^{(i)}\|}$$

Dropping some weights reduces the dimensionality of the representation
The sum of squared normalized weights is the energy retained in the approximated time-series
Retaining the weights with largest normalized values allows to minimize the reconstruction error
Discrete wavelet transform (DWT)

Dropping the smallest normalized weights provides a compact representation with minimum reconstruction error

\[ E = 0.9952 \]
Discrete wavelet transform (DWT)

Dropping the smallest normalized weights provides a compact representation with minimum reconstruction error.

\[ E = 0.9695 \]
Discrete wavelet transform (DWT)

Dropping the smallest normalized weights provides a compact representation with minimum reconstruction error.

\[ E = 0.9429 \]

\[ 0.97 \]

\[ 0.57 \]

\[ 0.54 \]

\[ 0.55 \]

\[ 1.26 \]

\[ -0.13 \]

\[ -1.06 \]

\[ -1.23 \]

\[ -0.96 \]
Discrete wavelet transform (DWT)

Dropping the smallest normalized weights provides a compact representation with minimum reconstruction error

\[ E = 0.8898 \]
Discrete wavelet transform (DWT)

Dropping the smallest normalized weights provides a compact representation with minimum reconstruction error.
Discrete wavelet transform (DWT)

Dropping the smallest normalized weights provides a compact representation with minimum reconstruction error.
Discrete wavelet transform (DWT)

Dropping the smallest normalized weights provides a compact representation with minimum reconstruction error

\[ E = 0.7739 \]
Discrete wavelet transform (DWT)

Dropping the smallest normalized weights provides a compact representation with minimum reconstruction error.

\[ E = 0.5719 \]
Example

\[\langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10 \rangle\]

Discrete wavelet transform (DWT), keeping 1/4 of dimensions
### Example

\[ a = \begin{bmatrix} -9.0 & 2.0 & 0.0 & -3.0 & 1.0 & 1.0 & -1.0 & -3.0 & 0.0 & -1.5 & -3.0 & -1.0 & 5.25 & 2.5 & 4.125 & 12.625 \end{bmatrix} \]

\[ W = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]

\[ E = \begin{bmatrix} 162.0 & 8.0 & 0.0 \end{bmatrix} \]
Example

Discrete wavelet transform (DWT), keeping $\frac{1}{4}$ of dimensions

$\langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10 \rangle$

energy retained $= 95.55\%$
Discrete Fourier transform (DFT)

Given a time-series \( S_X = \langle x_0, x_1, \ldots, x_{n-1} \rangle \)

The discrete Fourier transform decomposes the time-series into a collection of *sinusoids* with associated coefficients. Each Fourier coefficient \( f_k \) is a complex value. The original time-series can be reconstructed by summing all the weighted sinusoids.

\[
f_k = \sum_{r=0}^{n-1} x_r \cdot (\cos(2\pi rk/n) - i \sin(2\pi rk/n)) \quad \text{for } k = 0, \ldots, n - 1
\]

\[
x_r = \frac{1}{n} \sum_{k=0}^{n-1} f_k \cdot (\cos(2\pi rk/n) - i \sin(2\pi rk/n)) \quad \text{for } r = 0, \ldots, n - 1
\]

\( i \) denotes the imaginary number, \( i^2 = -1 \)
Each Fourier coefficient is a complex value $f_k = a_k + ib_k$

The Fourier coefficients are such that $a_{n-k} = a_k$ and $b_{n-k} = -b_k$ for $k > 0$

Therefore, the imaginary parts in the reconstructed series cancel out

Furthermore, the $n/2$ first complex coefficients need to be retained to reconstruct the original series exactly

Dropping the coefficients with low energy $a_k^2 + b_k^2$ provides a compact approximate representation
The discrete Fourier transform satisfies the \textit{additivity property}, i.e. the Fourier coefficients of the sum of two series are the sum of their coefficients

\[
\sum_{r=0}^{n-1} x_r^2 = \frac{1}{n} \sum_{k=0}^{k-1} a_k^2 + b_k^2
\]

This allows to compute the scaled Euclidean distance between two series by computing the Euclidean distance between their coefficients
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018
Weekly average **temperature** in Kuopio from 2014 to 2018
Weekly average temperature in Kuopio from 2014 to 2018
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018

$E = 0.8916$
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018

\[ E = 0.8969 \]
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018

$E = 0.9018$
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018

$E = 0.9617 \ (1/4 \text{ of dims})$
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018

$E = 0.9859 \ (1/2 \ of \ dims)$
Discrete Fourier transform (DFT)

Annual copper prices during the early 19\textsuperscript{th} century
Discrete Fourier transform (DFT)

Annual **copper** prices during the early 19\textsuperscript{th} century

\[ E = 0.5832 \]
Annual *copper* prices during the early 19\textsuperscript{th} century
Discrete Fourier transform (DFT)

Annual **copper** prices during the early 19\textsuperscript{th} century

\[ E = 0.6740 \]
Discrete Fourier transform (DFT)

Annual copper prices during the early 19\textsuperscript{th} century

\[ E = 0.6999 \]
Annual copper prices during the early 19\textsuperscript{th} century

\[ E = 0.7226 \]
Discrete Fourier transform (DFT)

Annual \textit{copper} prices during the early 19\textsuperscript{th} century

\begin{itemize}
  \item $E = 0.8914 \, (1/4 \text{ of dims})$
\end{itemize}
Discrete Fourier transform (DFT)

Annual copper prices during the early 19th century

\[ E = 0.9725 \text{ (}1/2\text{ of dims)} \]

UEF//School of Computing   ADA:Mining Time-Series
Models for time-series
Models for univariate time-series

Given a univariate time-series $S_X = \langle x_1, x_2, \ldots, x_n \rangle$, with $x_i \in \mathbb{R}$, the aim is to predict $x_{n+1}$
A stationary process is a stochastic process whose unconditional joint probability distribution does not change when shifted in time.

In a **strictly stationary time-series**, the probabilistic distribution of the values in any time interval \([a, b]\) is identical to that in the shifted interval \([a + \tau, b + \tau]\) for any value of the time shift \(\tau\).

Parameters, e.g. mean and variance, do not change over time. Estimated parameters are good predictors of future behavior.

White noise is the simplest example of a stationary process.
A stationary process is a stochastic process whose unconditional joint probability distribution does not change when shifted in time.

In a **strictly stationary time-series**, the probabilistic distribution of the values in any time interval \([a, b]\) is identical to that in the shifted interval \([a + \tau, b + \tau]\) for any value of the time shift \(\tau\).

In a **weakly stationary time-series**, the mean and autocovariance are constant in time.
Differencing

In some cases, the original time-series is not stationary but the difference between successive values is

Converting an original sequence into a sequence of differences is called **differencing**, e.g. first-order differencing of $S_X$

$$S_{X'} = \langle x'_1, x'_2, \ldots, x'_{n-1} \rangle,$$ where $x'_i = x_i - x_{i-1}$

Higher order differencing can also be used e.g. second-order differencing of $S_X$

$$S_{X''} = \langle x''_1, x''_2, \ldots, x''_{n-2} \rangle,$$ where $x''_i = x'_i - x'_{i-1}$

$$= x_i - 2x_{i-1} + x_{i-2}$$

For geometrically increasing series, the logarithm function is applied before differencing.
Example

IBM stock prices from Sept. 2013 to Sept. 2014

Original time-series

![Chart showing IBM stock prices from Sept. 2013 to Sept. 2014. The chart illustrates the fluctuation of stock prices over time with time on the x-axis and stock price on the y-axis.](chart.png)
Example

IBM stock prices from Sept. 2013 to Sept. 2014

First-order differenced time-series
IBM stock prices from Sept. 2013 to Sept. 2014
Second-order differenced time-series
Monthly mean CO₂ ppm. in Hawaii (US) from 1965 to 1980

Original time-series
Example

Monthly mean CO$_2$ ppm. in Hawaii (US) from 1965 to 1980
First-order differenced time-series
Nail prices from 1800 to 1996

Original time-series
Example

Nail prices from 1800 to 1996

First-order differenced time-series
Example

Nail prices from 1800 to 1996

Logarithmic time-series
Nail prices from 1800 to 1996
First-order differenced logarithmic time-series
Autocovariance

The **covariance** between two real-valued random variables $X$ and $Y$ is

$$
cov(X, Y) = E[(X - E[X])(Y - E[Y])]$$

The **autocovariance** at lag $\tau$ of time-series $X = x_1, x_2, \ldots, x_n$ is the covariance between the time-series and itself shifted by $\tau$

The **autocorrelation** at lag $\tau$ of time-series $X$ is the normalized covariance $\text{cov}_t(X_t, X_{t+\tau})/\text{var}_t(X_t)$ computed as

$$
R_{\tau}(X) = \frac{(X_t - \mu_X) \cdot (X_{t+\tau} - \mu_X)}{n \cdot (X_t - \mu_X)^2}
$$
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014

lag = 0

UEF//School of Computing
ADA: Mining Time-Series
IBM stock prices from Sept. 2013 to Sept. 2014

Autocorrelation

lag = 0  lag = 5
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014

lag = 0  
lag = 5  
lag = 10

UEF//School of Computing  ADA: Mining Time-Series
Autocorrelation

**IBM** stock prices from Sept. 2013 to Sept. 2014

- lag = 0
- lag = 5
- lag = 10
- lag = 50
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014

lag = 0

lag = 5

lag = 10

lag = 50

lag = 100
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014

lag = 0  lag = 5  lag = 10  lag = 50  lag = 100  lag = 150

UEF/School of Computing  ADA:Mining Time-Series
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014

lag = 0  lag = 5  lag = 10  lag = 50  lag = 100  lag = 150

UF/\School of Computing  ADA:Mining Time-Series
Auto-regressive model AR($p$)

The value $x_i$ at timestamp $t_i$ is defined as a linear combination of the values in the immediately preceding window of length $p$

$$x_i = \sum_{k=1}^{p} a_k \cdot x_{i-k} + c + \epsilon_i$$

The deviation from predicted value, $\epsilon_i$, can be viewed as white noise or a shock.

The regression coefficients $a_1, \ldots, a_p$ and $c$ need to be learnt from training data.
Auto-regressive models

**Auto-regressive model AR(p)** \( x_i = \sum_{k=1}^{p} a_k \cdot x_{i-k} + c + \epsilon_i \)

The regression coefficients \( a_1, \ldots, a_p \) and \( c \) need to be learnt from training data.

A linear equation between the coefficients is created for the value at each timestamp in the past history of the time-series and its associated preceding window.

When the number of timestamps available is larger than \( p \), this results in an over-determined system of equations. The coefficients minimizing the square error of this system of equations are approximated with *least-squares regression*. 
Auto-regressive models

Moving-average model MA($q$)

The value $x_i$ at timestamp $t_i$ is defined as a linear combination of the shocks in the immediately preceding window of length $q$

$$x_i = \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i$$

Parameter $c$ is the mean of the time-series

The regression coefficients $b_1, \ldots, b_q$ need to be learnt from training data
Auto-regressive models

Moving-average model $MA(q)$

\[ x_i = \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i \]

The regression coefficients $b_1, \ldots, b_q$ need to be learnt from training data.

The auto-regressive model relates the current value to the history of previous values.

The moving-average model relates the current value to the history of deviations from previous forecasted values.

This circularity means the system of equations is non-linear.

The solution is found using iterative non-linear methods.
Auto-regressive models

Auto-regressive model $\text{AR}(p)$
$$x_i = \sum_{k=1}^{p} a_k \cdot x_{i-k} + c + \epsilon_i$$

Moving-average model $\text{MA}(q)$
$$x_i = \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i$$

Auto-regressive moving-average model $\text{ARMA}(p, q)$
Combine $p$ auto-regressive terms and $q$ moving-average terms to capture the impact of both autocorrelation and shocks
$$x_i = \sum_{k=1}^{p} a_k \cdot x_{i-k} + \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i$$
Auto-regressive models

Auto-regressive model AR\((p)\) \(x_i = \sum_{k=1}^{p} a_k \cdot x_{i-k} + c + \epsilon_i\)

Moving-average model MA\((q)\) \(x_i = \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i\)

ARMA\((p, q)\) \(x_i = \sum_{k=1}^{p} a_k \cdot x_{i-k} + \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i\)

Auto-regressive integrated moving-average model ARIMA\((p, d, q)\)
ARMA model applied to the \(d\)-order differenced time-series

\(x'_i = \sum_{k=1}^{p} a_k \cdot x'_{i-k} + \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i\)
Auto-regressive models

Auto-regressive model AR($p$)  
\[
x_i = \sum_{k=1}^{p} a_k \cdot x_{i-k} + c + \epsilon_i
\]

Moving-average model MA($q$)  
\[
x_i = \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i
\]

ARMA($p$, $q$)  
\[
x_i = \sum_{k=1}^{p} a_k \cdot x_{i-k} + \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i
\]

ARIMA($p$, $d$, $q$)  
\[
x'_i = \sum_{k=1}^{p} a_k \cdot x'_{i-k} + \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i
\]
ACF and PACF

The **autocorrelation** at lag $\tau$ of time-series $X$ is the normalized covariance

$$ACF_X(\tau) = R_\tau(X) = \frac{t(X_t, X_{t+\tau})}{\text{var}_t(X_t)}$$

The **partial autocorrelation** at lag $\tau$ of time-series $X$, $\text{PACF}_X(\tau)$, is the autocorrelation at lag $\tau$ that is not accounted for by shorter lags.
Box–Jenkins modelling procedure

Model identification

1. Use differencing to make the time-series stationary
2. Determine the most suitable model and find appropriate values for $q$ and $p$
   - by looking at ACF and PACF respectively, or
   - by using Akaike’s Information Criterion (AIC)

Model estimation
Estimate the parameters of the model from historical data

Model validation
Check that the model is adequate for the time-series
The effectiveness of the forecasting model can be quantified by measuring the noise in the estimated coefficients. The coefficient of determination measures the ratio between the white noise and the series variance.

\[ R^2 = 1 - \frac{\text{mean} \left( \epsilon_i^2 \right)}{\text{var} \left( x_i^2 \right)} \]

It should be as close to 1 as possible.
Models for multivariate time-series

In practice, time-series often consist of multiple variables. In addition to correlation across time, i.e. individual variables being autocorrelated, there might be significant correlations across the variables.

One approach to build models for this scenario is to use *hidden variables*. The multiple input time-series are transformed into a smaller number of uncorrelated time-series, typically using principal component analysis (PCA).

A model is built for each such time-series individually. The models are used to predict hidden values, which are then mapped back into the original representation.
Given a multivariate time-series \( S_x = \langle x^{(1)}, x^{(2)}, \ldots, x^{(n)} \rangle \), with \( x^{(i)} \in \mathbb{R}^m \), the aim is to predict \( x^{(n+1)} \).
Models for multivariate time-series

Given a multivariate time-series $\mathcal{S}_X = \langle x^{(1)}, x^{(2)}, \ldots, x^{(n)} \rangle$, with $x^{(i)} \in \mathbb{R}^m$, the aim is to predict $x^{(n+1)}$

1. Build the $m \times m$ covariance matrix $C$, where $C_{ij}$ is the covariance between variables $i$ and $j$

$$C_{ij} = (\langle x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)} \rangle, \langle x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(n)} \rangle)$$

$C$ captures the correlations across variables, not across time.
Given a multivariate time-series $S_x = \langle x^{(1)}, x^{(2)}, \ldots, x^{(n)} \rangle$, with $x^{(i)} \in \mathbb{R}^m$, the aim is to predict $x^{(n+1)}$

2. Compute the eigendecomposition of $C$
   Keep the $p$ eigenvectors with largest eigenvalues

$$C = QA^T$$

where the $m$ columns of $Q$ contain the orthogonal eigenvectors and the diagonal of $\Lambda$ contains the corresponding eigenvalues.

Let $P$ be the $m \times p$ matrix obtained by selecting the $p$ columns of $Q$ with largest eigenvalues, for some $p \ll m$
Models for multivariate time-series

Given a multivariate time-series $S_X = \langle x^{(1)}, x^{(2)}, \ldots, x^{(n)} \rangle$, with $x^{(i)} \in \mathbb{R}^m$, the aim is to predict $x^{(n+1)}$

3. Compute the $p$-dimensional hidden time-series $S_H$

$$S_H = \langle h^{(1)}, h^{(2)}, \ldots, h^{(n)} \rangle, \text{ with } h^{(i)} = x^{(i)}P \in \mathbb{R}^p$$

The $p$ variables of $S_H$ are uncorrelated
Models for multivariate time-series

Given a multivariate time-series $S_X = \langle x^{(1)}, x^{(2)}, \ldots, x^{(n)} \rangle$, with $x^{(i)} \in \mathbb{R}^m$, the aim is to predict $x^{(n+1)}$

4. Build a model for each dimension $j$ of $S_H$
   Use them to predict the hidden values $h^{(n+1)}_j$

Build $p$ separate univariate models using for instance the Box–Jenkins modelling procedure

Predict $h^{(n+1)}_j$ using the model built for hidden variable $j$ and corresponding hidden values $h^{(n)}_j, h^{(n-1)}_j, \ldots$
Given a multivariate time-series $S_X = \langle x^{(1)}, x^{(2)}, \ldots, x^{(n)} \rangle$, with $x^{(i)} \in \mathbb{R}^m$, the aim is to predict $x^{(n+1)}$

5. Transform $h^{(n+1)}$ back to the original $m$-dimensional representation, the prediction $x^{(n+1)}$ for timestamp $n + 1$

Return $x^{(n+1)} = h^{(n+1)}P^T$ as the predicted vector
Models for multivariate time-series

Given a multivariate time-series $S_x = \langle x^{(1)}, x^{(2)}, \ldots, x^{(n)} \rangle$, with $x^{(i)} \in \mathbb{R}^m$, the aim is to predict $x^{(n+1)}$

1. Build the $m \times m$ covariance matrix $C$, where $C_{ij}$ is the covariance between variables $i$ and $j$
2. Compute the eigendecomposition of $C$
   Keep the $p$ eigenvectors with largest eigenvalues
3. Compute the $p$-dimensional hidden time-series $S_H$
4. Build a model for each dimension $j$ of $S_H$
   Use them to predict the hidden values $h^{(n+1)}_j$
5. Transform $h^{(n+1)}$ back to the original $m$-dimensional representation, the prediction $x^{(n+1)}$ for timestamp $n + 1$
Models for time-series

Artificial neural networks (ANN) offer a flexible alternative e.g. long short-term Memory (LSTM) recurrent neural networks (RNN) architecture

Have fewer restrictions
Can model non-linear functions
Time-series might exhibit regularly recurrent, cyclic, behavior i.e. display \textit{periodicity} (a.k.a. seasonality)
Periodicity

Time-series might exhibit regularly recurrent, cyclic, behavior i.e. display **periodicity** (a.k.a. seasonality)

Seasonal differencing \( x_i - x_{i-p} \) for some integer \( p > 1 \), i.e. taking the difference between values one period \( p \) apart, can be used to remove the effect of seasonality
Periodicity

Given a time-series $S_X = \langle x_0, x_1, \ldots, x_{n-1} \rangle$

The discrete Fourier transform decomposes the time-series into $n - 1$ periodic sinusoidal components

$$x_r = \frac{1}{n} \sum_{k=0}^{n-1} f_k \cdot (\cos(2\pi r k / n) - i \sin(2\pi r k / n)) \quad \text{for } r = 0, \ldots, n - 1$$

The $k^{th}$ component, corresponding to coefficient $f_k = a_k + i b_k$, has periodicity $n / k$ and amplitude $\sqrt{a_k^2 + b_k^2}$.

If a component has a high amplitude compared to the others, the entire series will be dominated by its periodic behavior.

Only components such that $k \in [\beta, n / \alpha]$ have period at least $\alpha \geq 2$ and appear at least $\beta \geq 2$ in the series.
Monthly mean CO₂ ppm. in Hawaii (US) from 1965 to 1980

Original time-series
Example

Monthly mean CO$_2$ ppm. in Hawaii (US) from 1965 to 1980

Folding the time-series, $p = 12$
Monthly mean CO$_2$ ppm. in Hawaii (US) from 1965 to 1980
Folding the time-series, $p = 12$
Example

Monthly mean $\text{CO}_2$ ppm. in Hawaii (US) from 1965 to 1980
Seasonal differenced time-series, $p = 12$
Example

Monthly mean CO₂ ppm. in Hawaii (US) from 1965 to 1980

Original time-series

![Graph showing the monthly mean CO₂ ppm in Hawaii from 1965 to 1980. The graph illustrates a trend with seasonal variations.](image-url)
Monthly mean CO$_2$ ppm. in Hawaii (US) from 1965 to 1980

First-order differenced time-series
Monthly mean CO\textsubscript{2} ppm. in Hawaii (US) from 1965 to 1980
Monthly mean CO$_2$ ppm. in Hawaii (US) from 1965 to 1980
Example

Monthly mean CO₂ ppm. in Hawaii (US) from 1965 to 1980

*Graph showing time-series data with labeled axes.*
Monthly mean CO$_2$ ppm. in Hawaii (US) from 1965 to 1980
Example

Monthly mean CO$_2$ ppm. in Hawaii (US) from 1965 to 1980

lag = 0

UEF//School of Computing   ADA: Mining Time-Series
Monthly mean CO\textsubscript{2} ppm. in Hawaii (US) from 1965 to 1980
Monthly mean CO₂ ppm. in Hawaii (US) from 1965 to 1980

lag = 0  lag = 2  lag = 6
Example

Monthly mean CO$_2$ ppm. in Hawaii (US) from 1965 to 1980

lag = 0  lag = 2  lag = 6  lag = 12

UEF//School of Computing  ADA:Mining Time-Series
Monthly mean CO₂ ppm. in Hawaii (US) from 1965 to 1980

lag = 0  lag = 2  lag = 6  lag = 12  lag = 48
Example

Monthly mean CO\textsubscript{2} ppm. in Hawaii (US) from 1965 to 1980

<table>
<thead>
<tr>
<th>lag</th>
<th>plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>![lag=0]</td>
</tr>
<tr>
<td>2</td>
<td>![lag=2]</td>
</tr>
<tr>
<td>6</td>
<td>![lag=6]</td>
</tr>
<tr>
<td>12</td>
<td>![lag=12]</td>
</tr>
<tr>
<td>48</td>
<td>![lag=48]</td>
</tr>
<tr>
<td>100</td>
<td>![lag=100]</td>
</tr>
</tbody>
</table>

\[lag = 0\]
\[lag = 2\]
\[lag = 6\]
\[lag = 12\]
\[lag = 48\]
\[lag = 100\]
Monthly mean CO$_2$ ppm. in Hawaii (US) from 1965 to 1980

Example