Algorithmic Data Analysis

Esther Galbrun

Spring 2024

UNIVERSITY OF
EASTERN FINLAND

Last updated on 2023-12-28 11:19

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

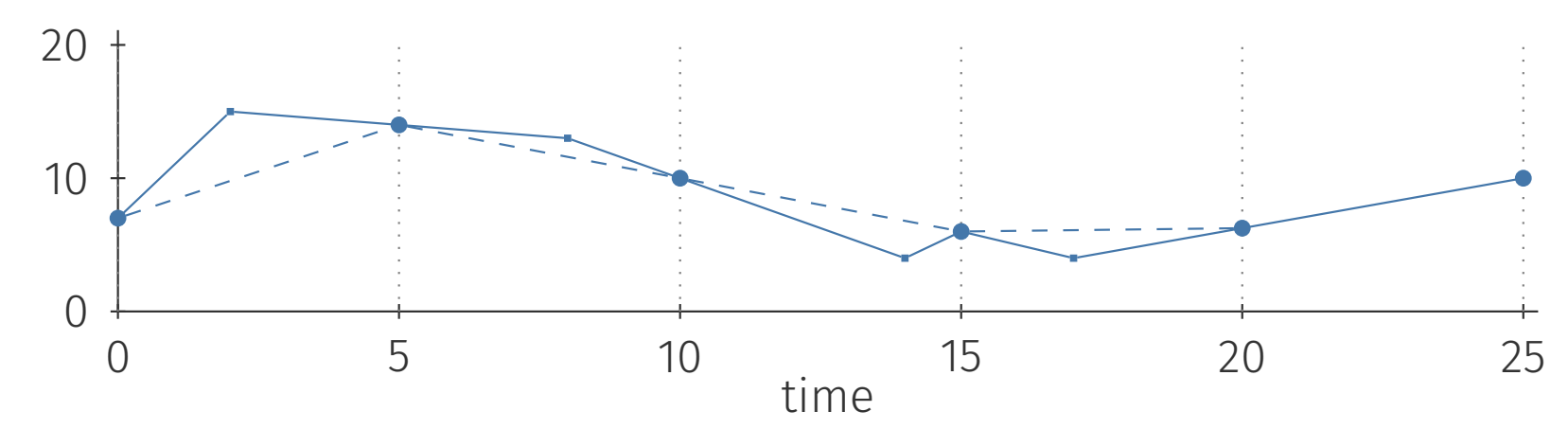
Time-series might contain missing values
When data is collected from independent sensors, the values might not be synchronized

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

The result is a time-series $\mathcal{S}_X = \langle x_1, x_2, \ldots, x_n \rangle$, 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 δ

Example



$\langle (0,7), (2,15), (8,13), (14,4), (15,6), (17,4), (25,10) \rangle$

Linear interpolation, for equally spaced time points
$\langle 0, 5, 10, 15, 20, 25 \rangle$

$\langle (0,7), (5,14), (10,10), (15,6), (20,6.25), (25,10) \rangle$

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*

Binning

Consider a time-series $\mathcal{S}_X = \langle x_1, x_2, \ldots, x_n \rangle$, 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

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

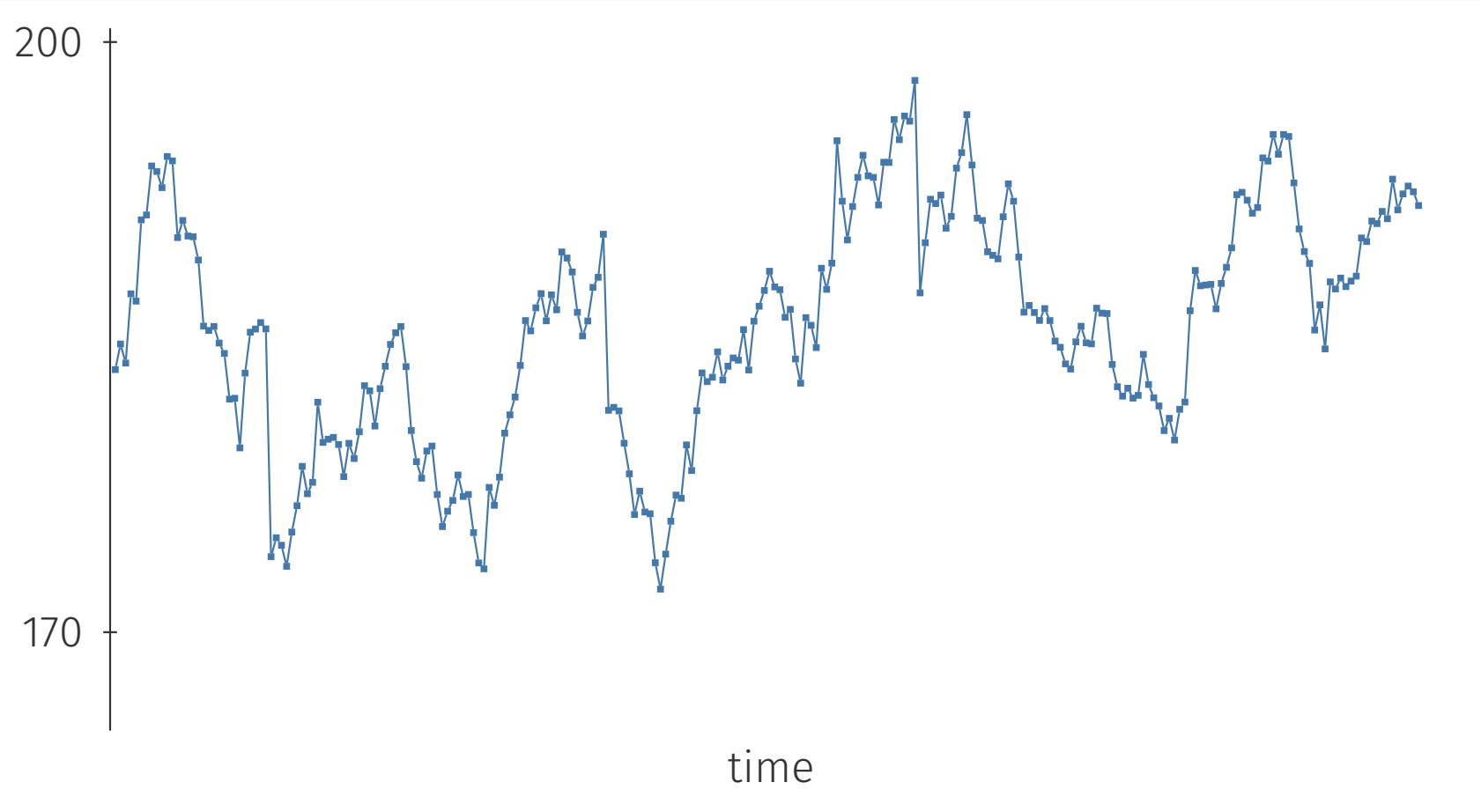
Instead of average, it is possible to take the median, which is more robust to the presence of outlier values

Binning is lossy, reduces the number of points by a factor of k

Example

IBM stock prices from Sept. 2013 to Sept. 2014

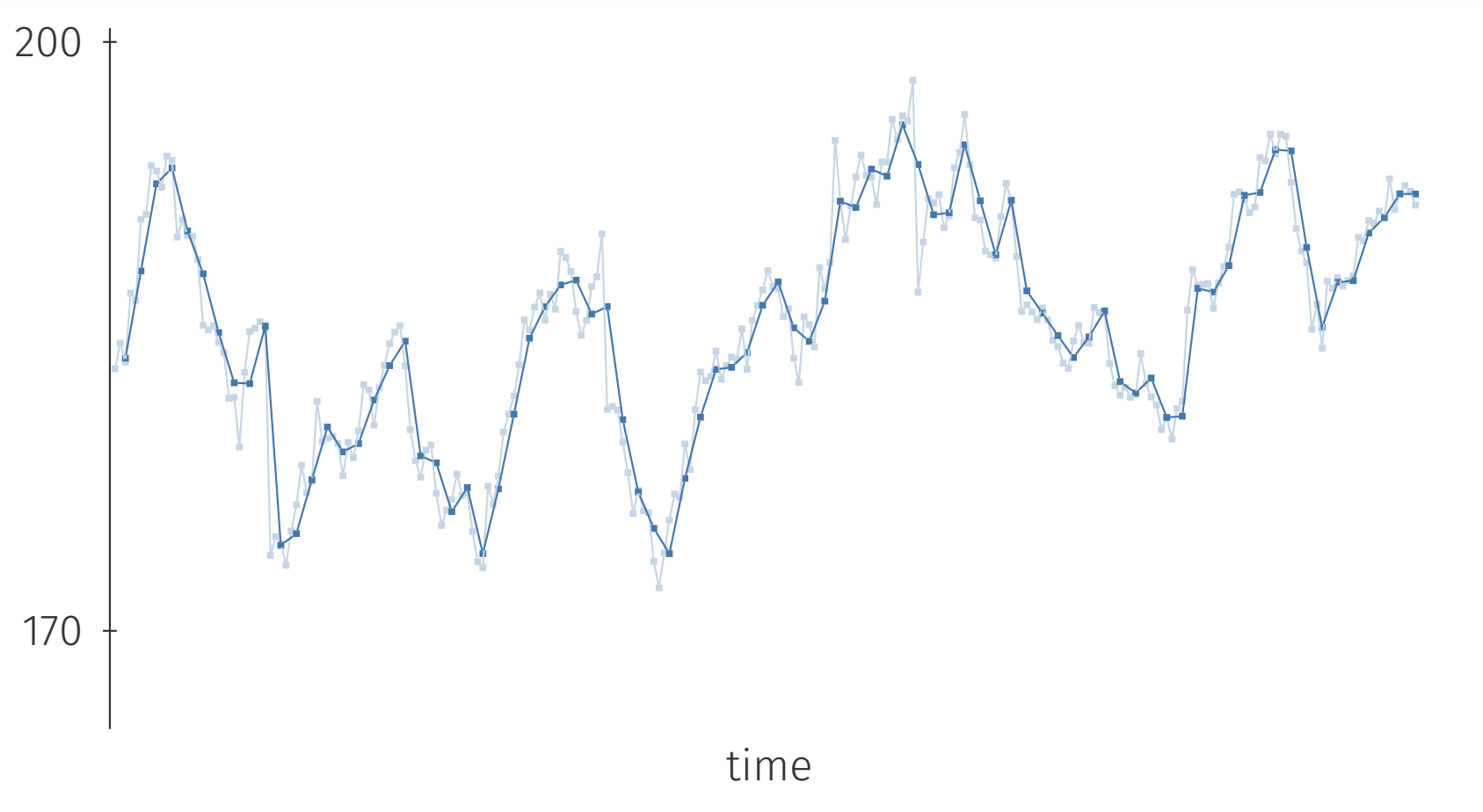
Original time-series



Example

IBM stock prices from Sept. 2013 to Sept. 2014

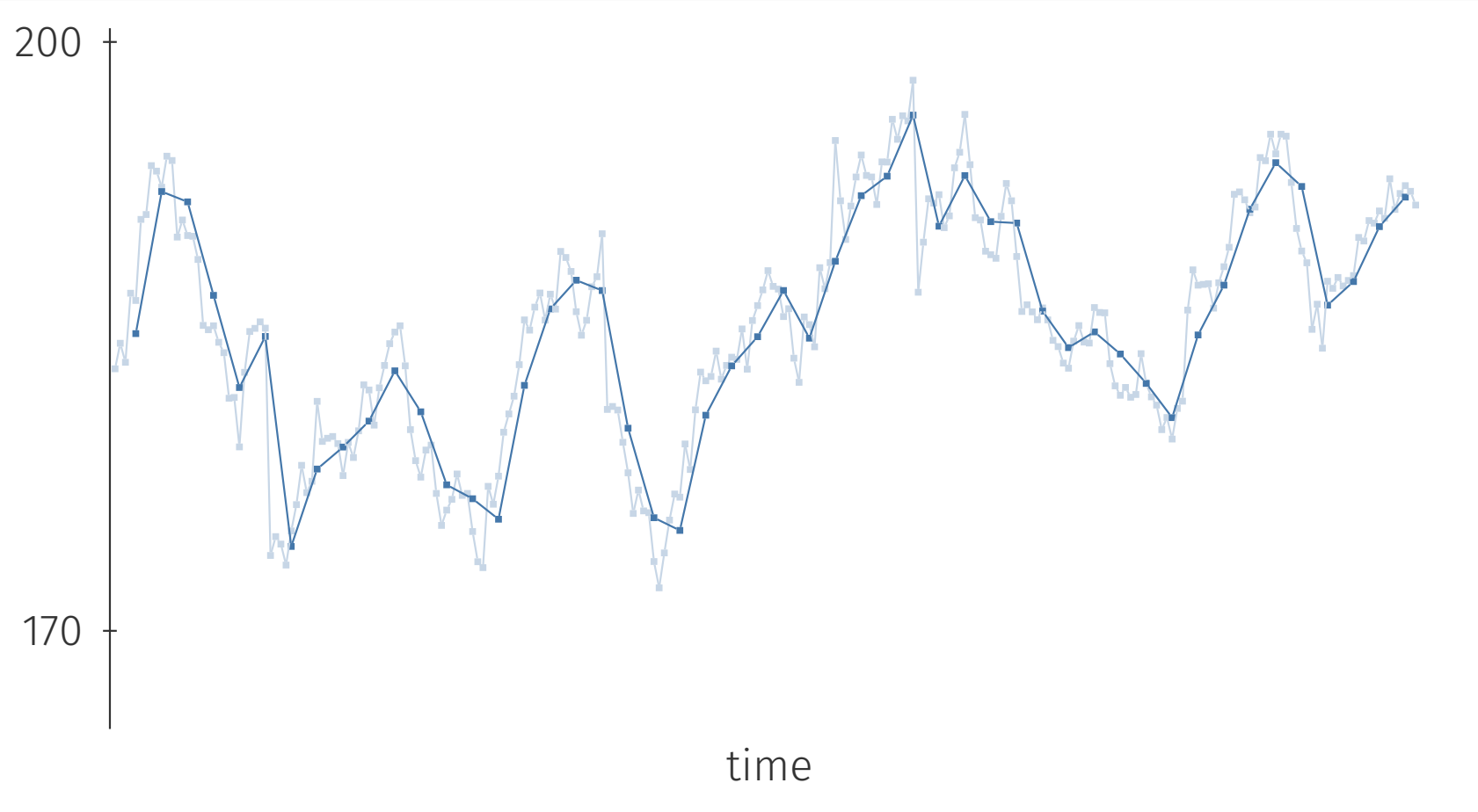
Binning, $k = 3$



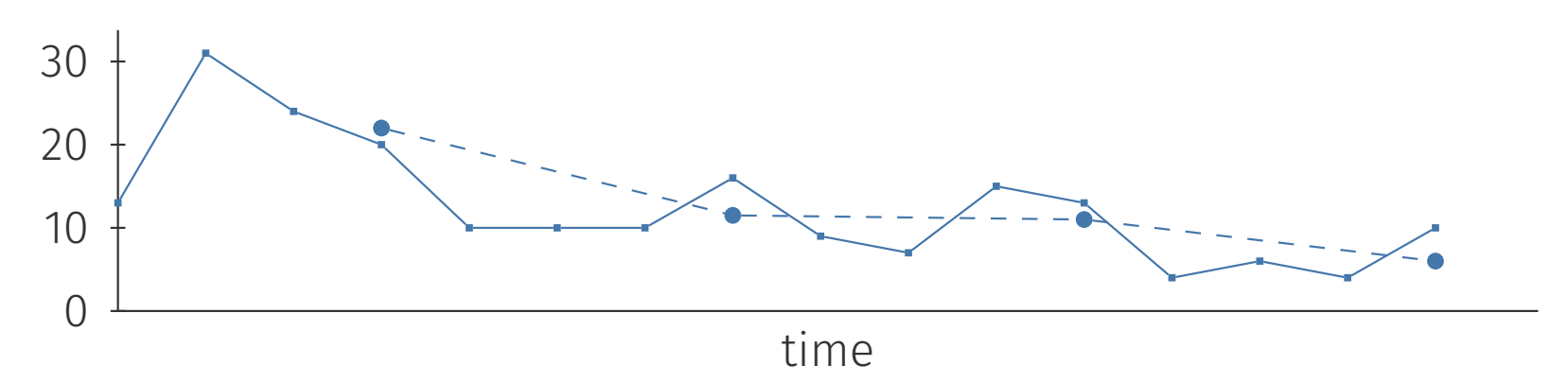
Example

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 $\mathcal{S}_X = \langle x_1, x_2, \ldots, x_n \rangle$, 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 $\mathcal{S}_X = \langle x_1, x_2, \ldots, x_n \rangle$, 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

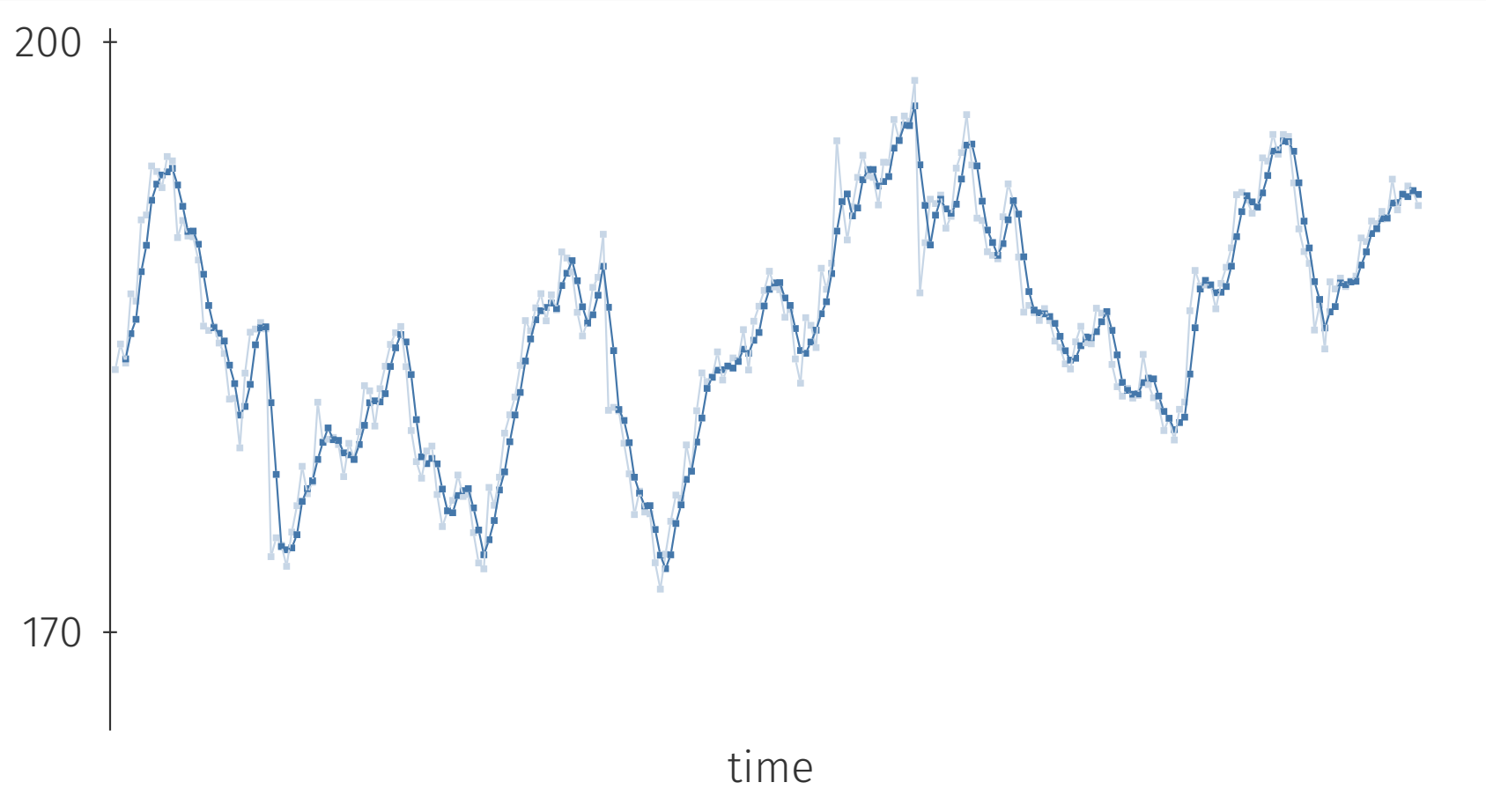
$$y_i = \frac{1}{k} \sum_{r=0}^{k-1} x_{i+r} \quad \text{for } i = 1, \ldots, n-k+1$$

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

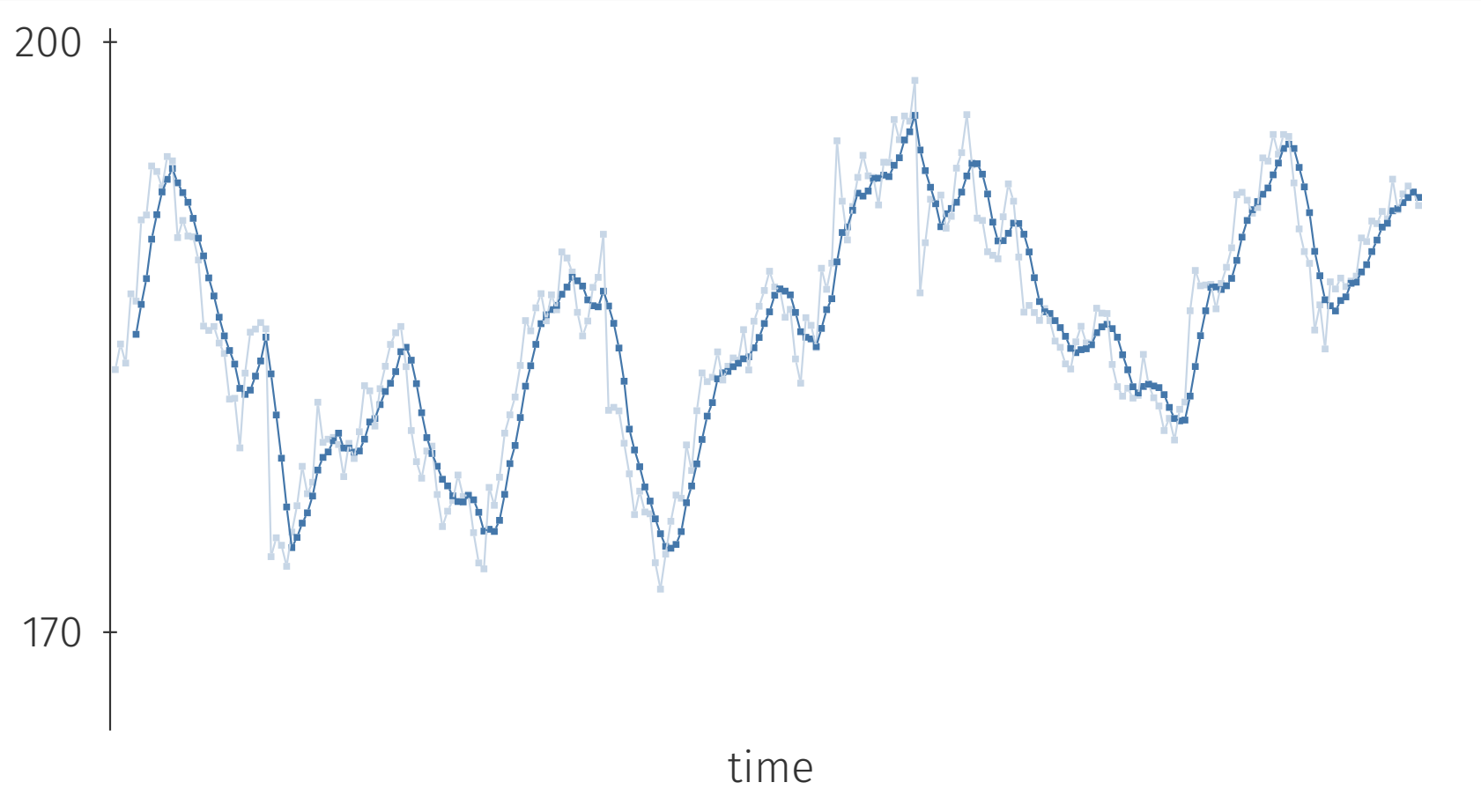
Moving-average smoothing, $k = 3$



Example

IBM stock prices from Sept. 2013 to Sept. 2014

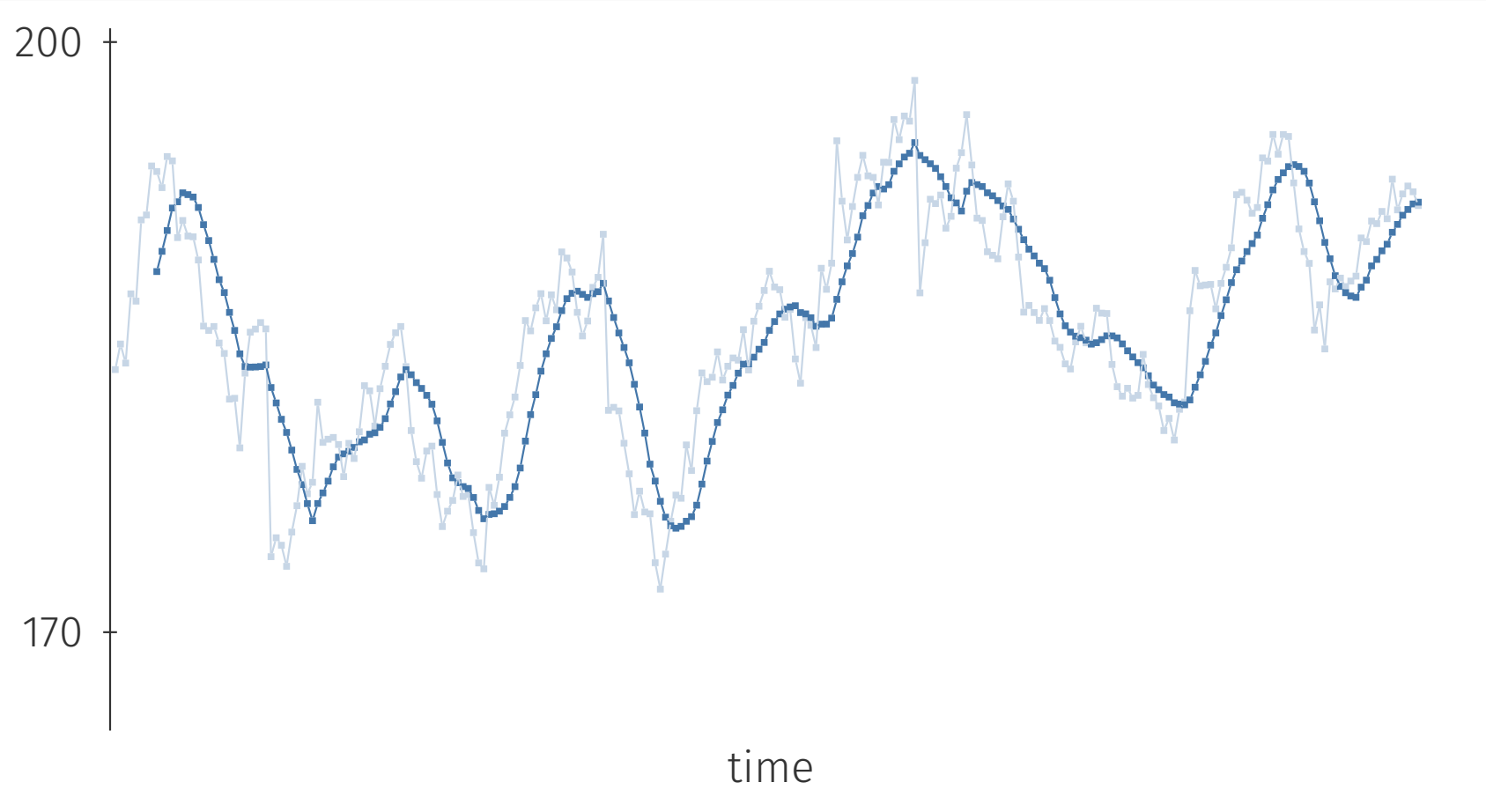
Moving-average smoothing, $k = 5$



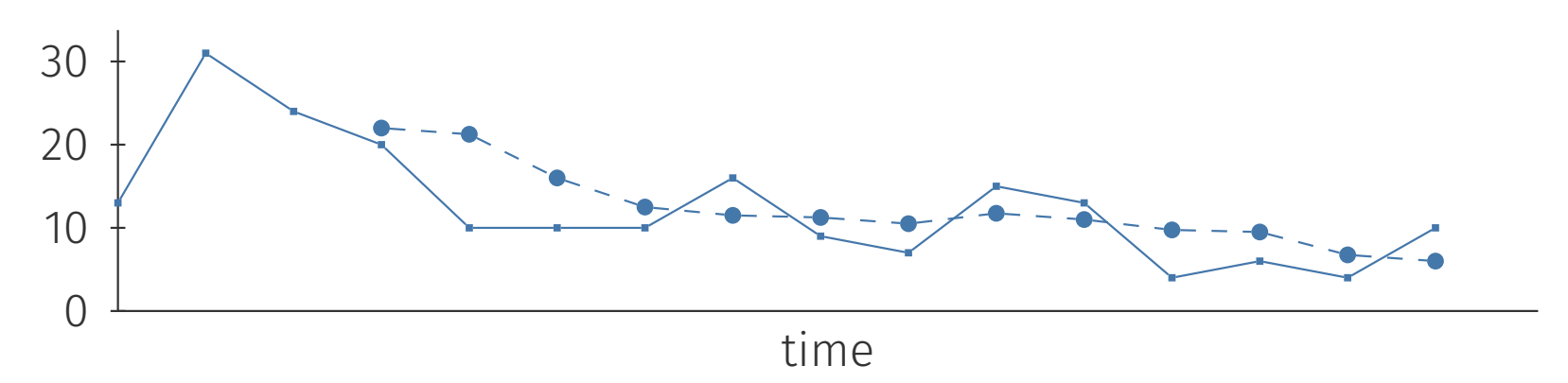
Example

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

$\langle 22, 21.25, 16, 12.5, 11.5, 11.25, 10.5, 11.75, 11, 9.75, 9.5, 6.75, 6 \rangle$

Exponential smoothing

Consider a time-series $\mathcal{S}_X = \langle x_1, x_2, \ldots, x_n \rangle$, 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 $y_1 = x_1$

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

Exponential smoothing

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

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

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

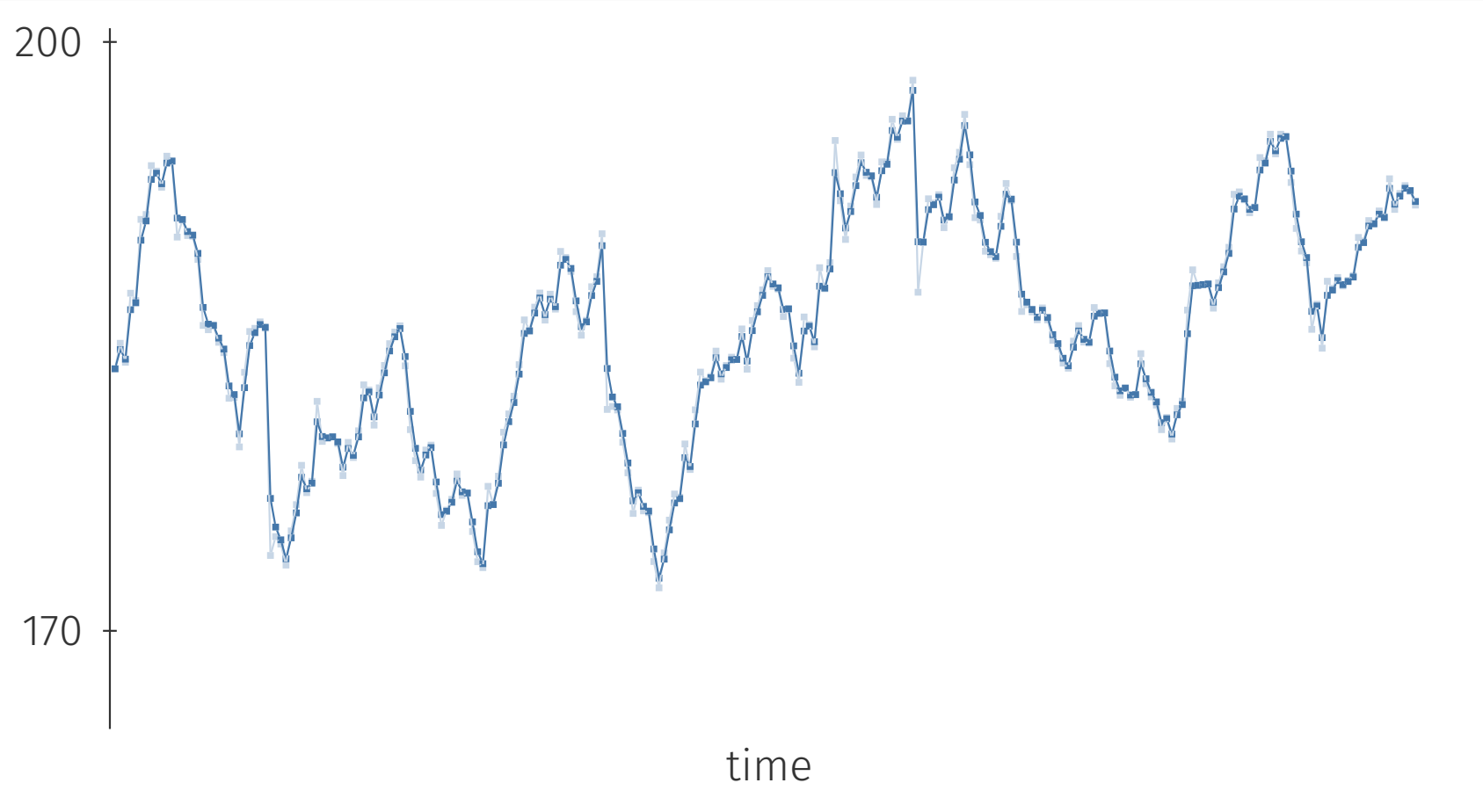
The smoothed values can be expressed as an exponentially decayed sum of the original values, giving more importance to recent values

The smoothing parameter α 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

Example

IBM stock prices from Sept. 2013 to Sept. 2014

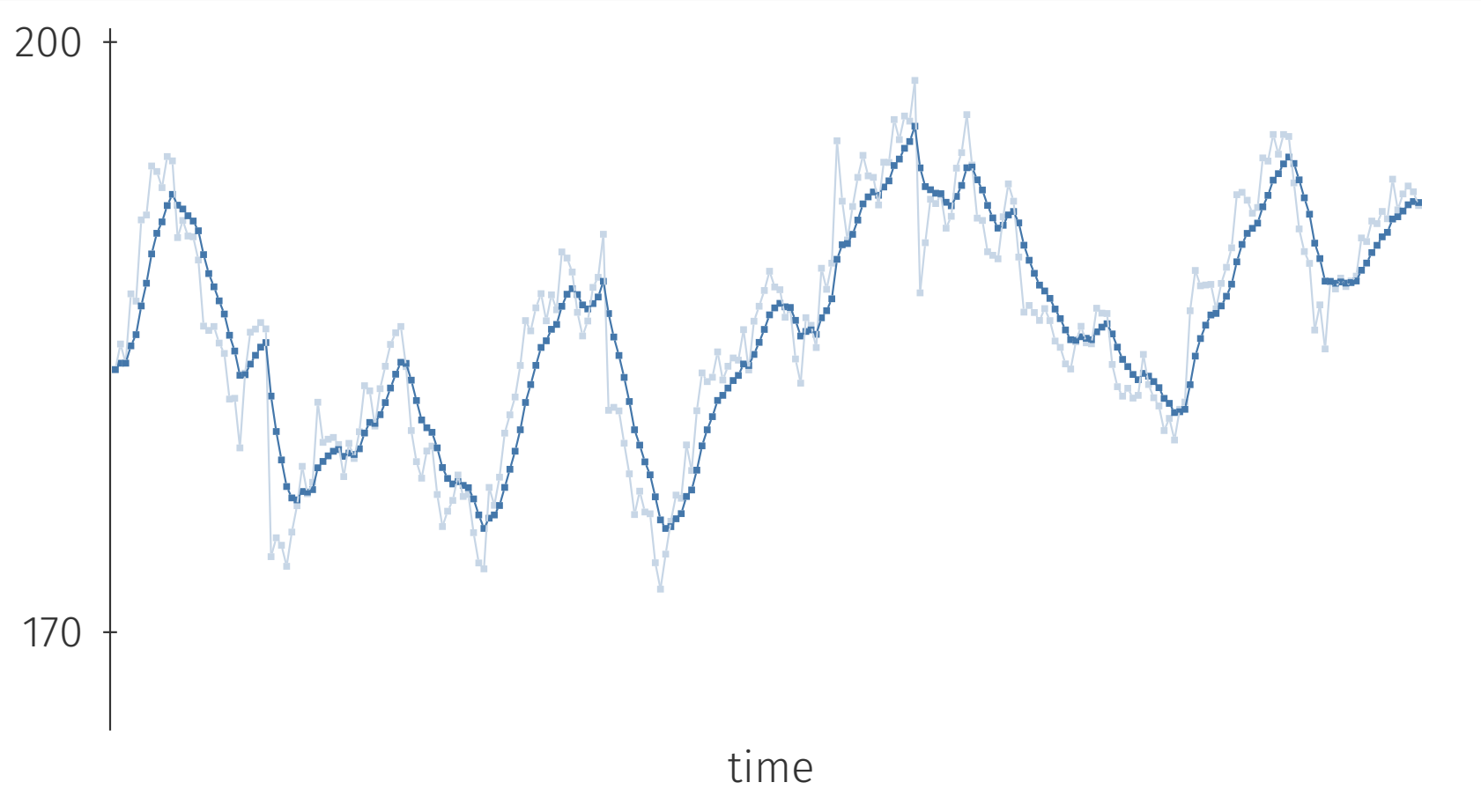
Exponential smoothing, $\alpha = 0.75$



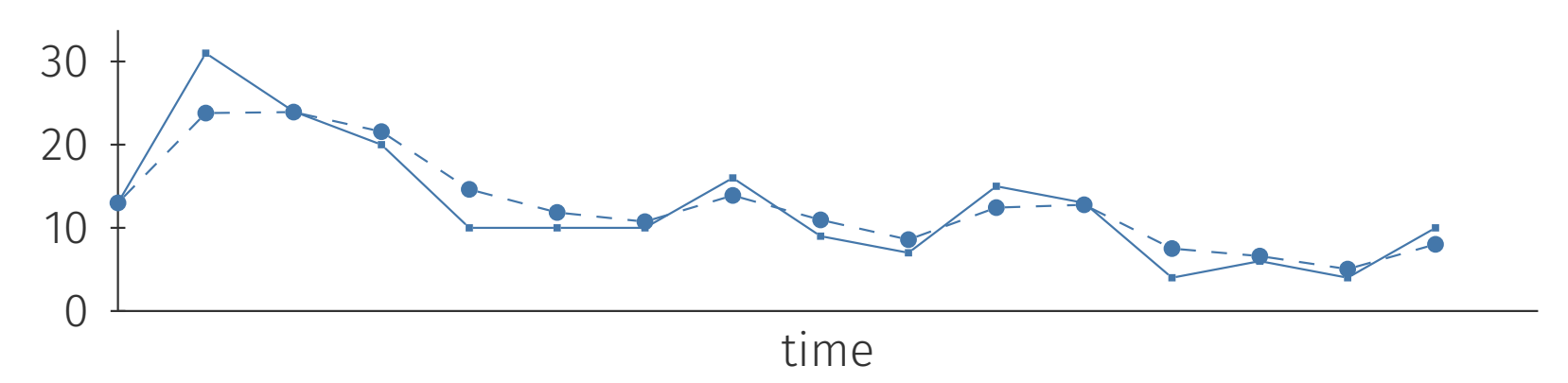
Example

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 $\mathcal{S}_X = \langle x_1, x_2, \ldots, x_n \rangle$, 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 $\mathcal{S}_X = \langle x_1, x_2, \ldots, x_n \rangle$, with mean μ and standard deviation σ, **standardization** maps the original time-series values to new values

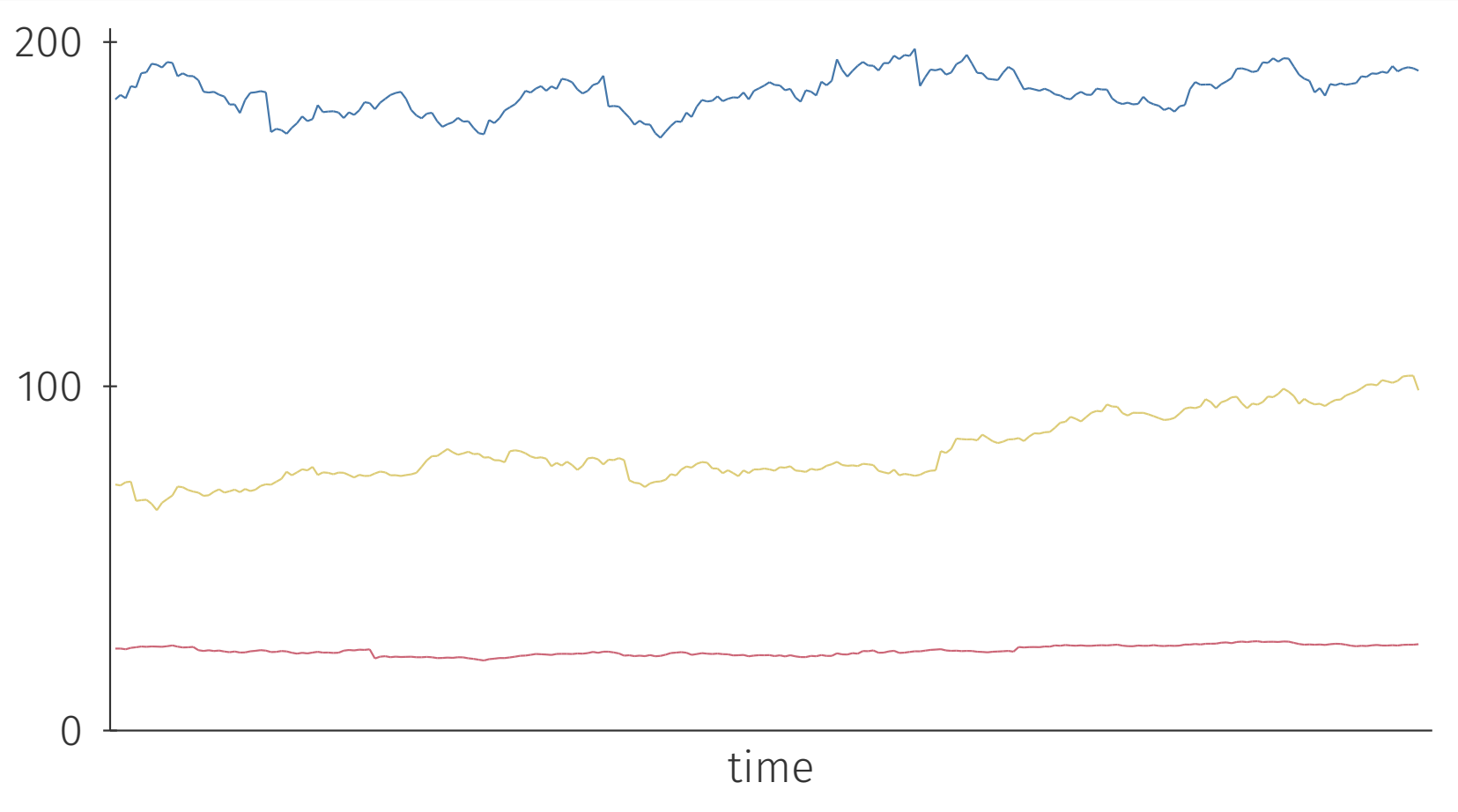
$$y_i = \frac{x_i - \mu}{\sigma}$$

! No guaranteed specific range for the resulting values

Example

IBM, Cisco and Apple stock prices from Sept. 2013 to Sept. 2014

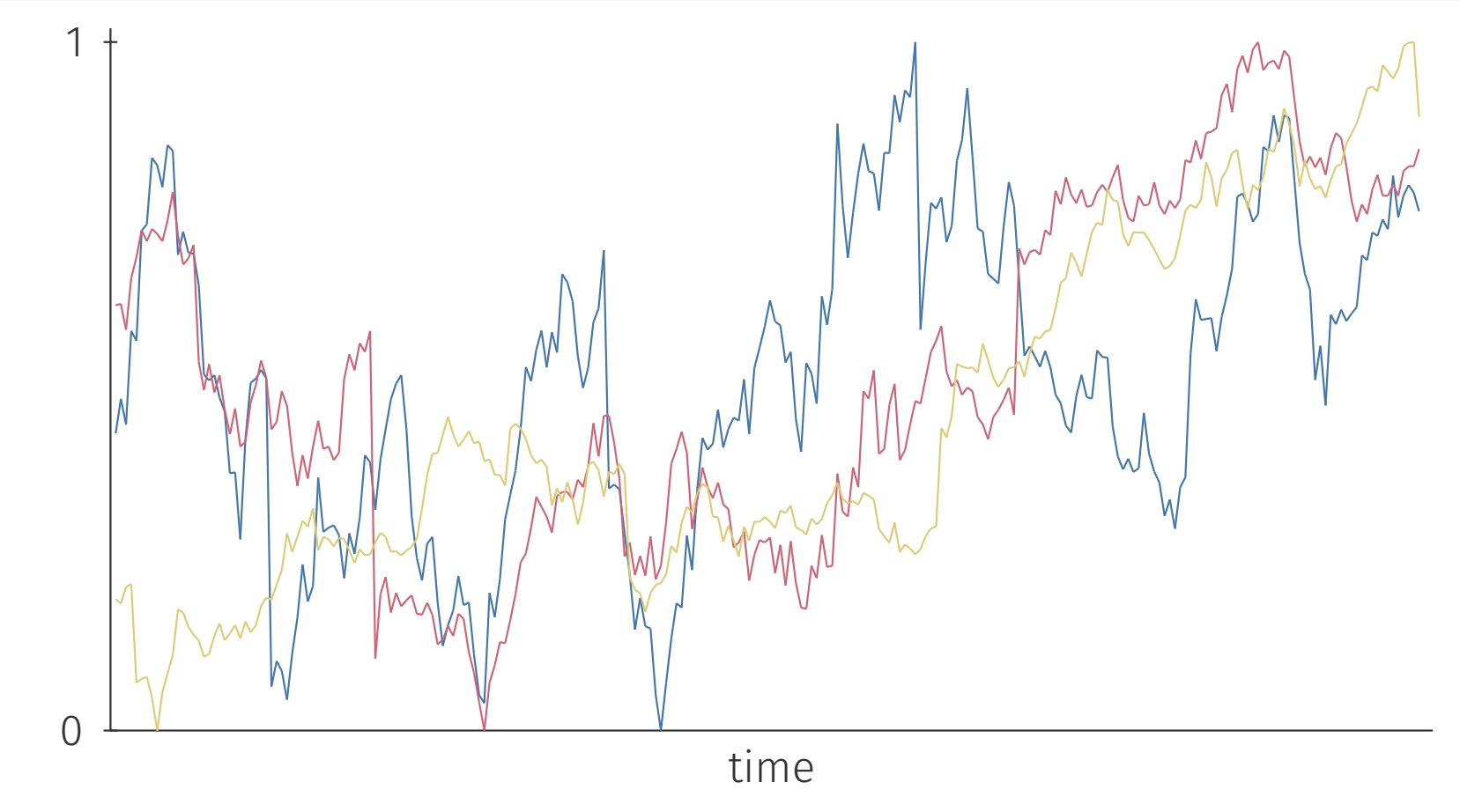
Original time-series



Example

IBM, Cisco and Apple stock prices from Sept. 2013 to Sept. 2014

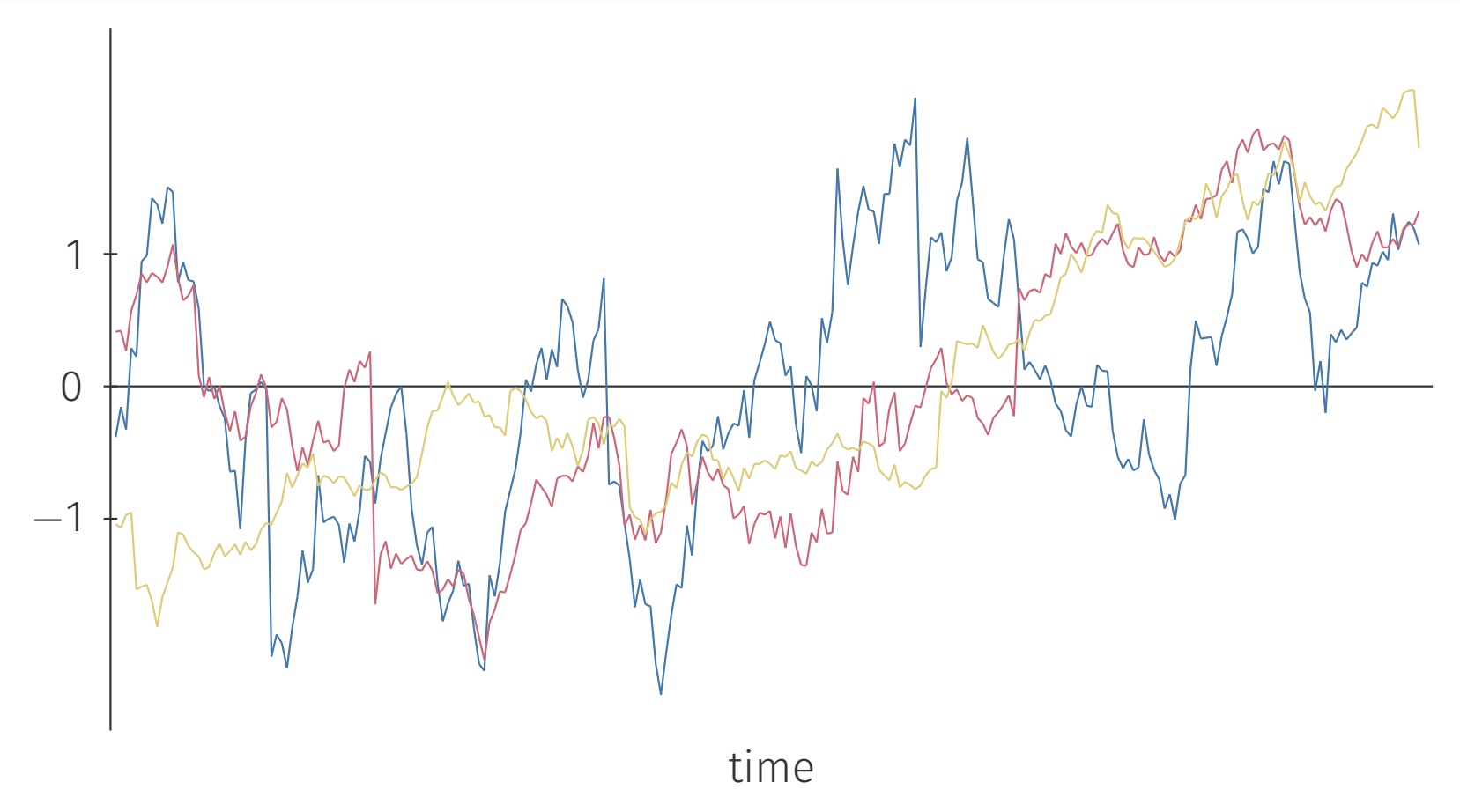
Range-based normalized time-series



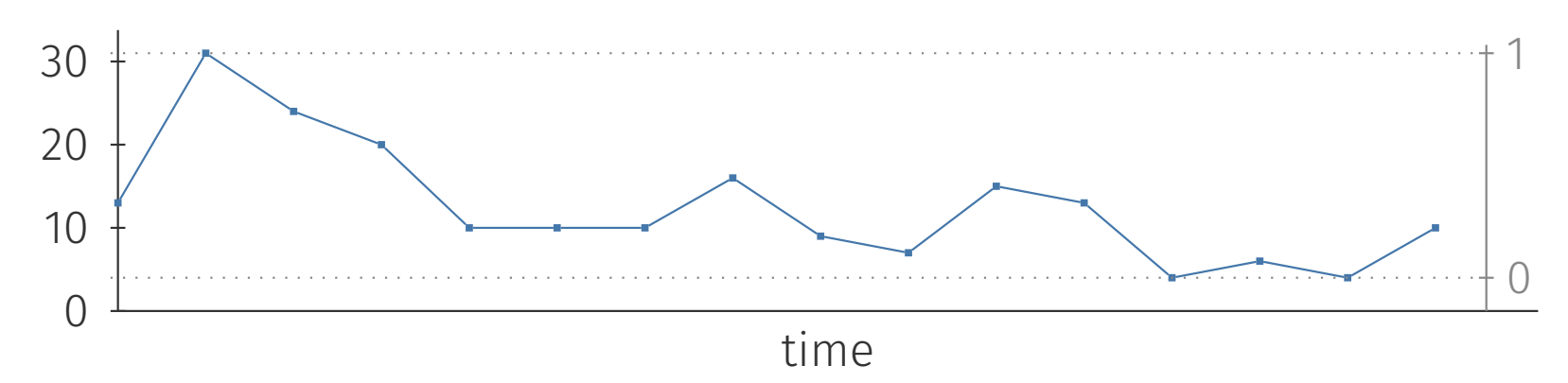
Example

IBM, Cisco and Apple stock prices from Sept. 2013 to Sept. 2014

Standardized time-series



Data preparation

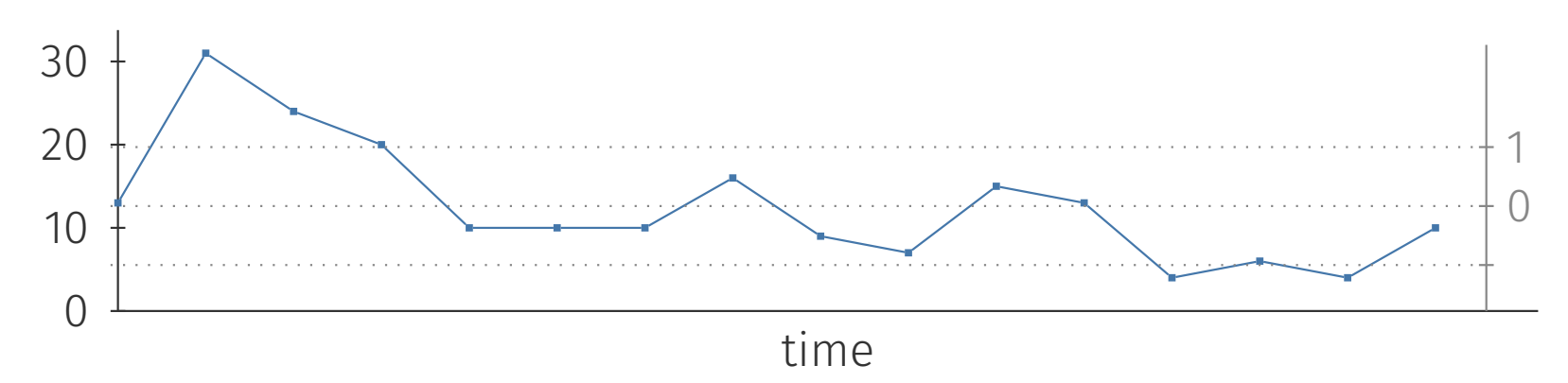


$\langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10 \rangle$

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

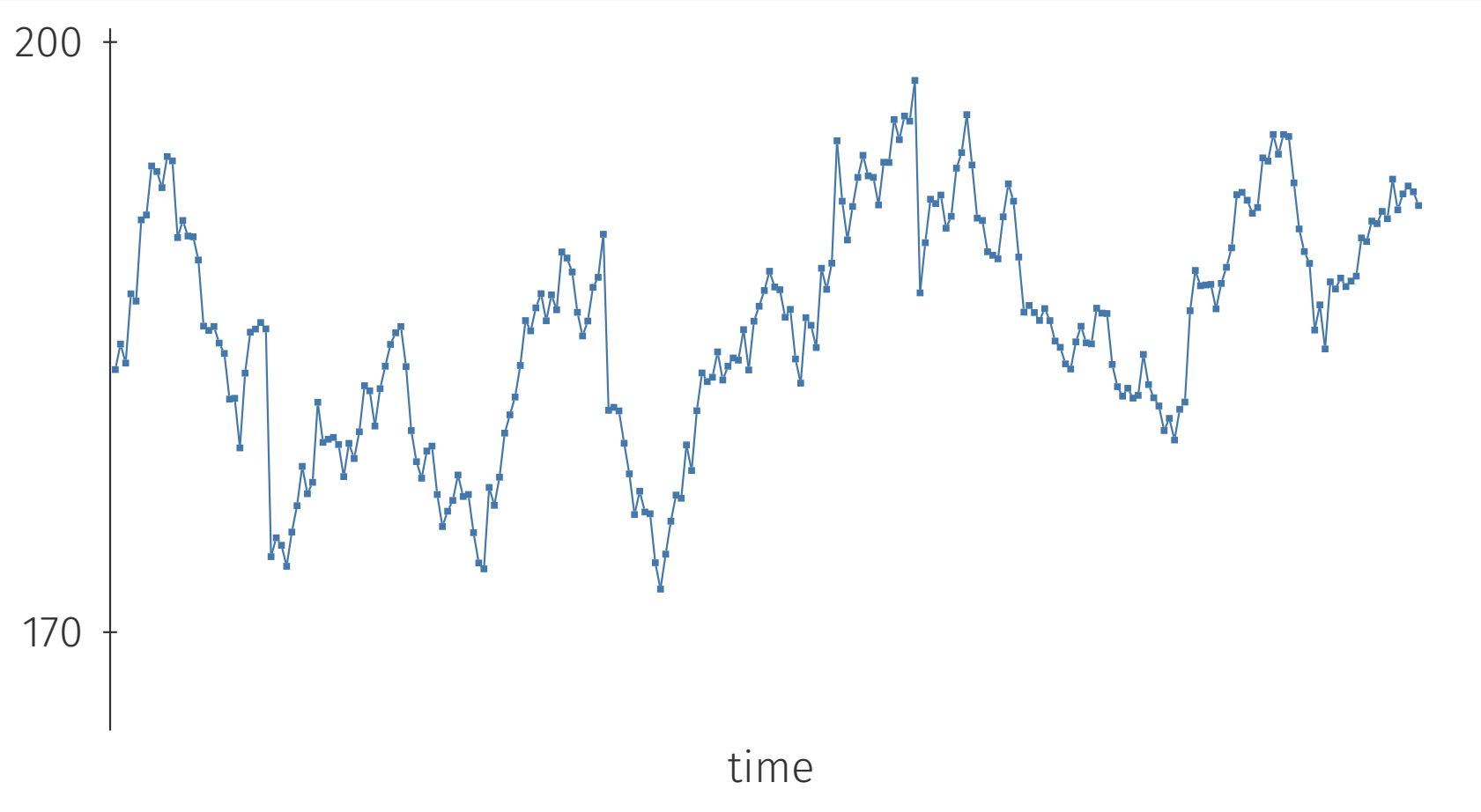
Select breakpoints such that symbols have approximately equal occurrence frequencies
Use Gaussian distribution assumption for long time-series or in the streaming setting

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

Example

IBM stock prices from Sept. 2013 to Sept. 2014

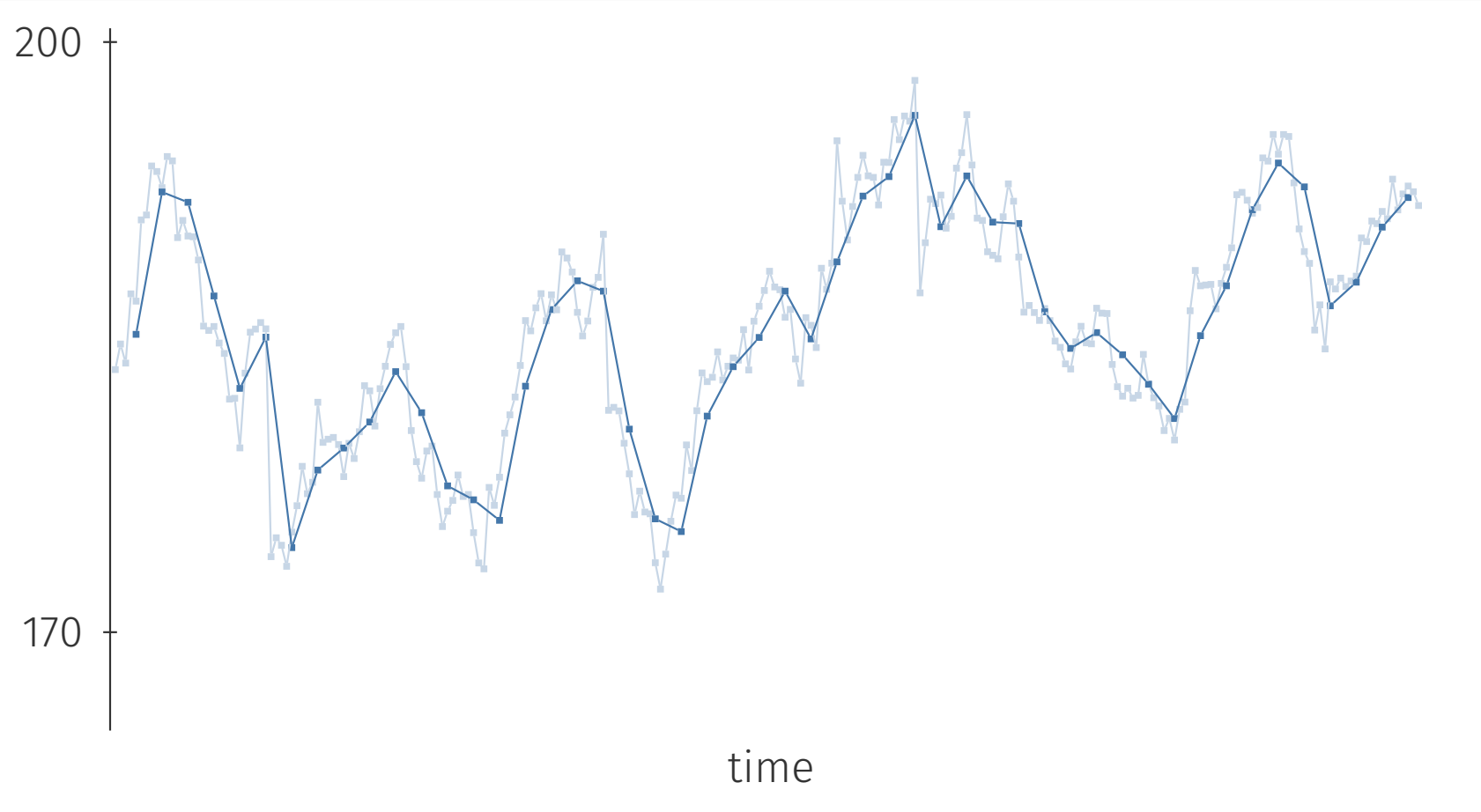
Original time-series



Example

IBM stock prices from Sept. 2013 to Sept. 2014

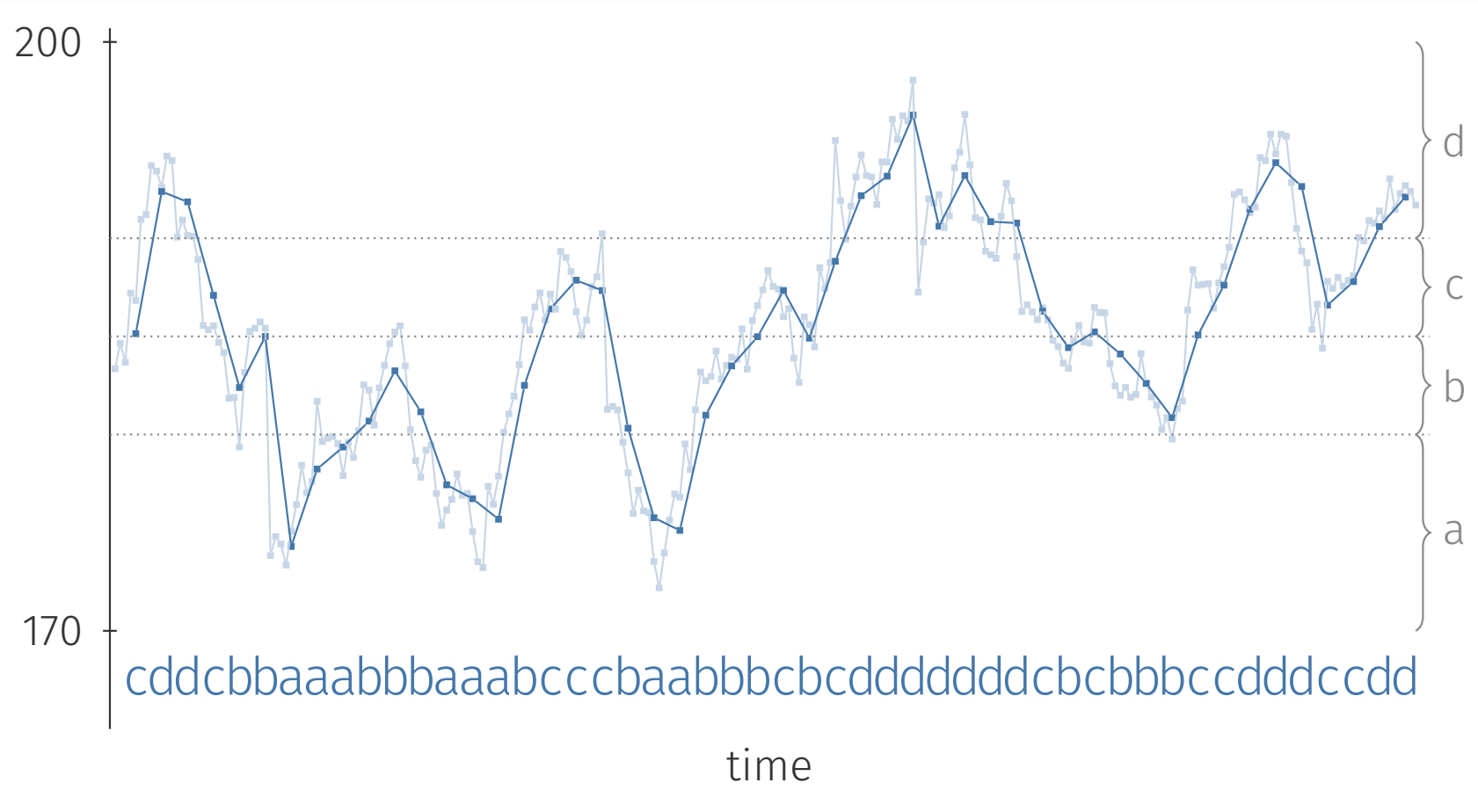
Binning, $k = 5$



Example

IBM stock prices from Sept. 2013 to Sept. 2014

Discretizing



Transforms

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

Transforms: an analogy

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?

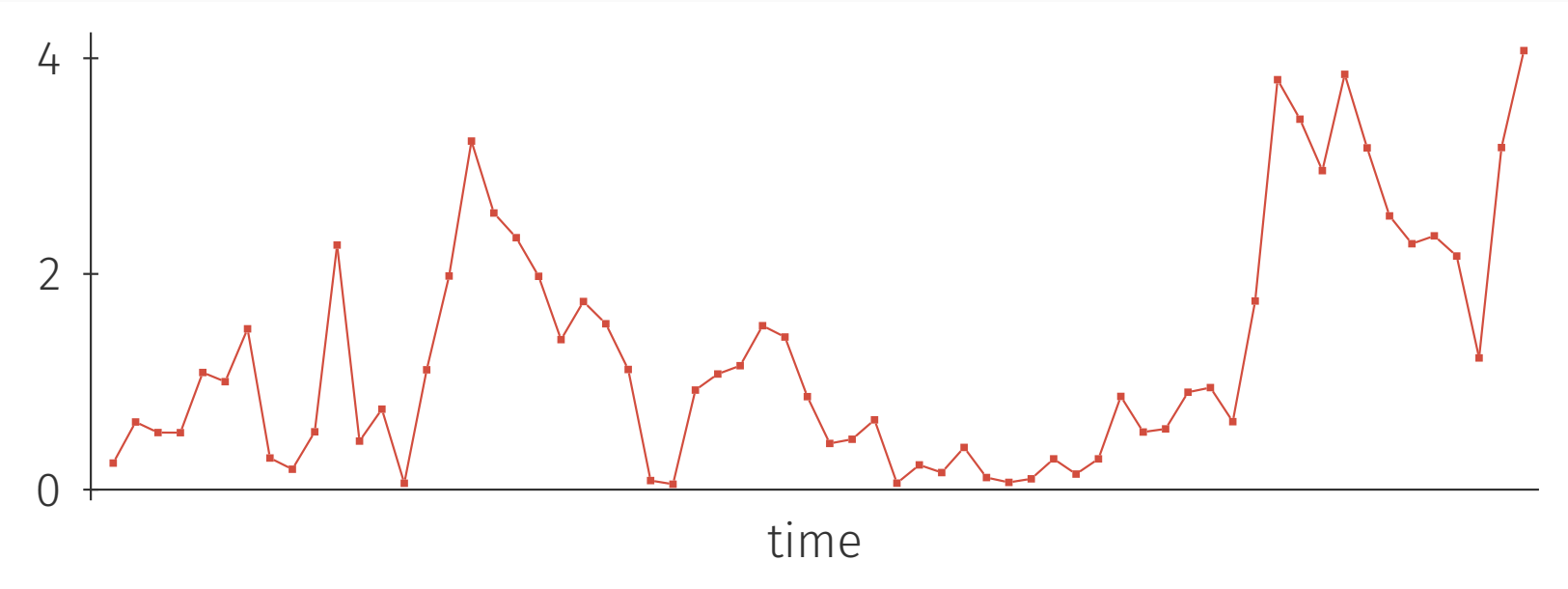
Simply blend the ingredients back together

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

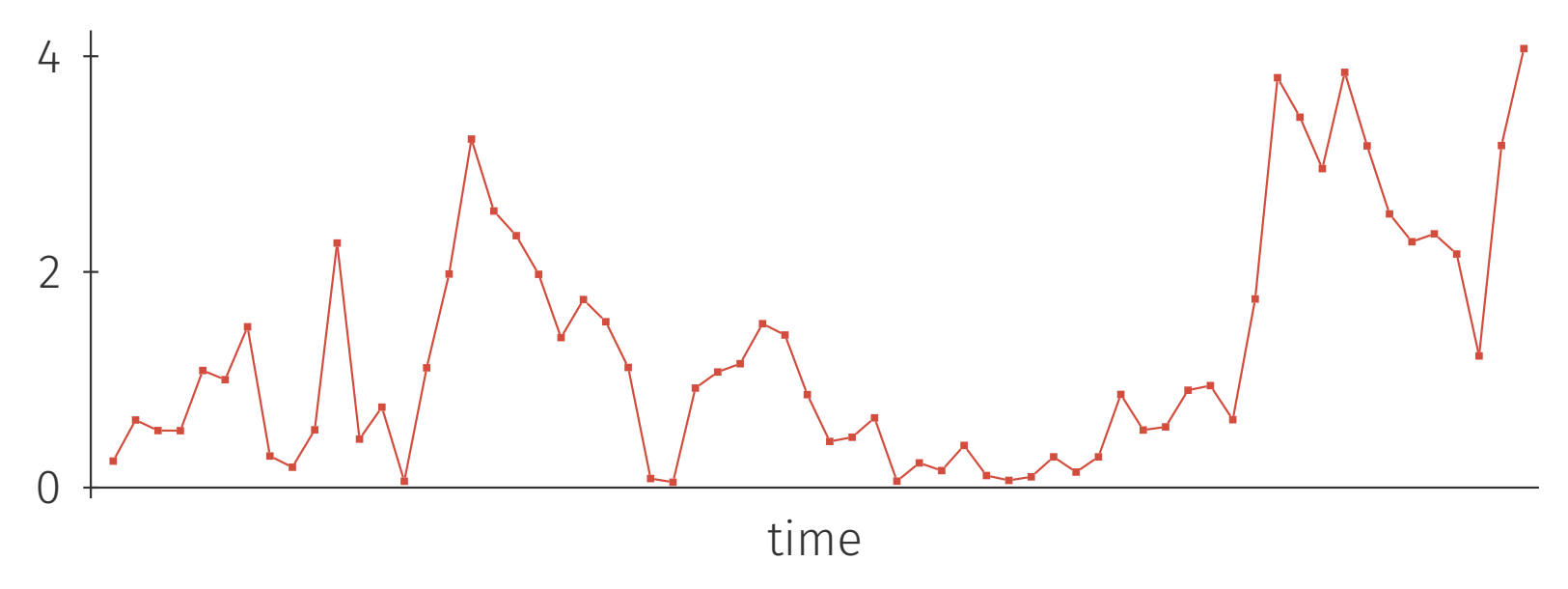
Discrete wavelet transform (DWT)

Annual copper prices during the early 19th century



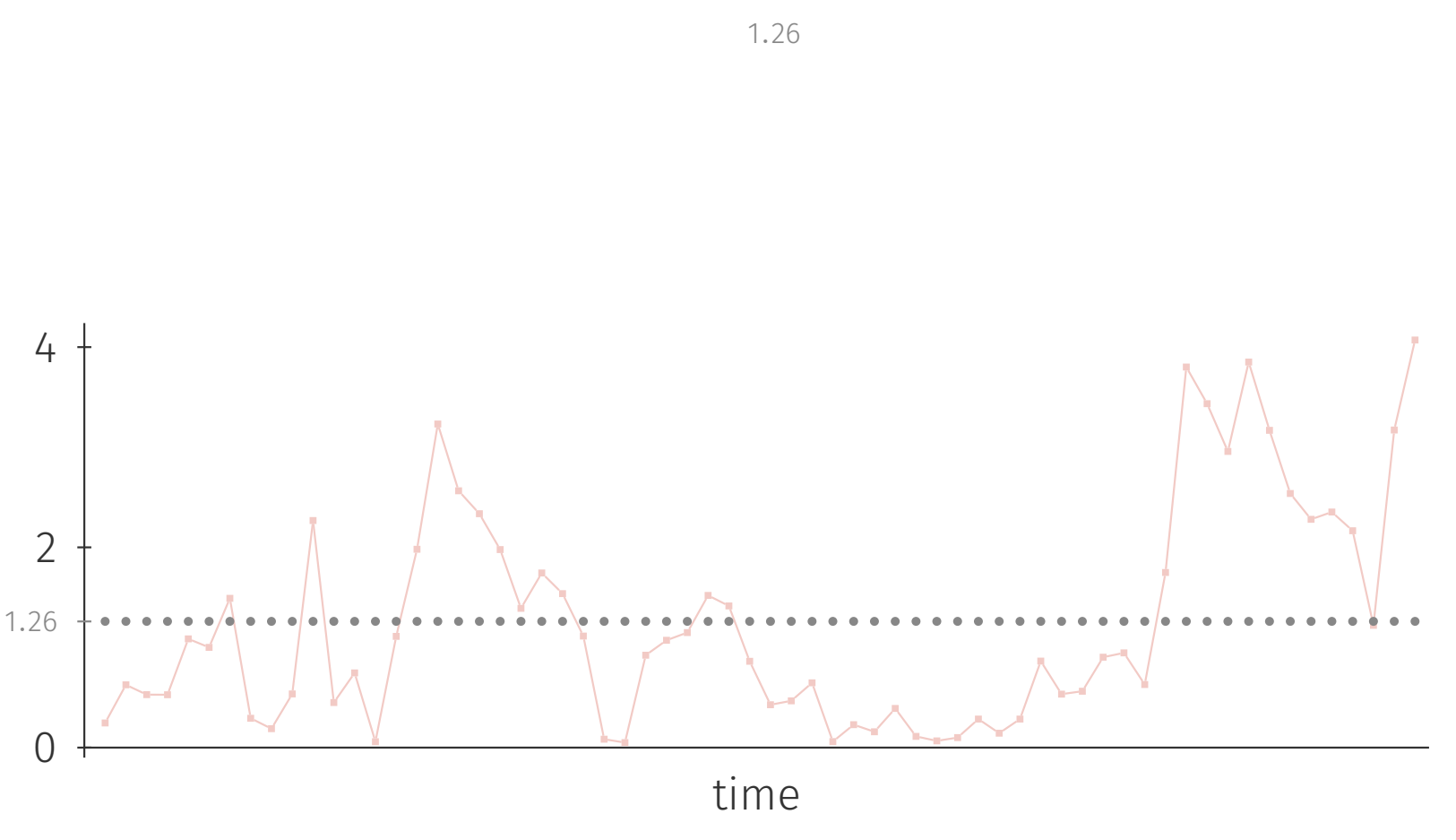
Discrete wavelet transform (DWT)

Adjacent values in the time-series are often very similar, storing all the values is wasteful, redundant



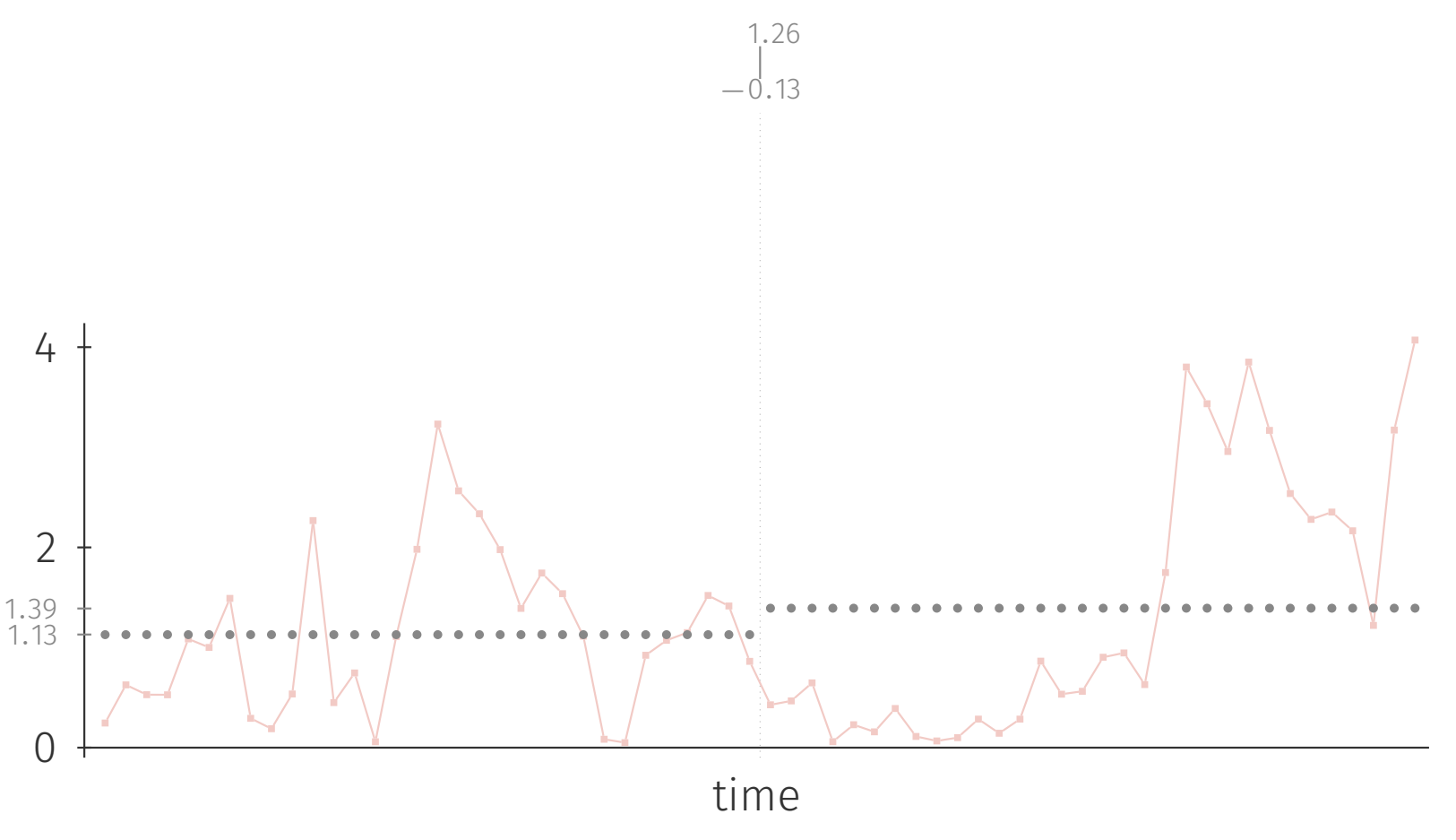
Discrete wavelet transform (DWT)

The average value alone provides a very crude representation of the time-series, without any information about variations



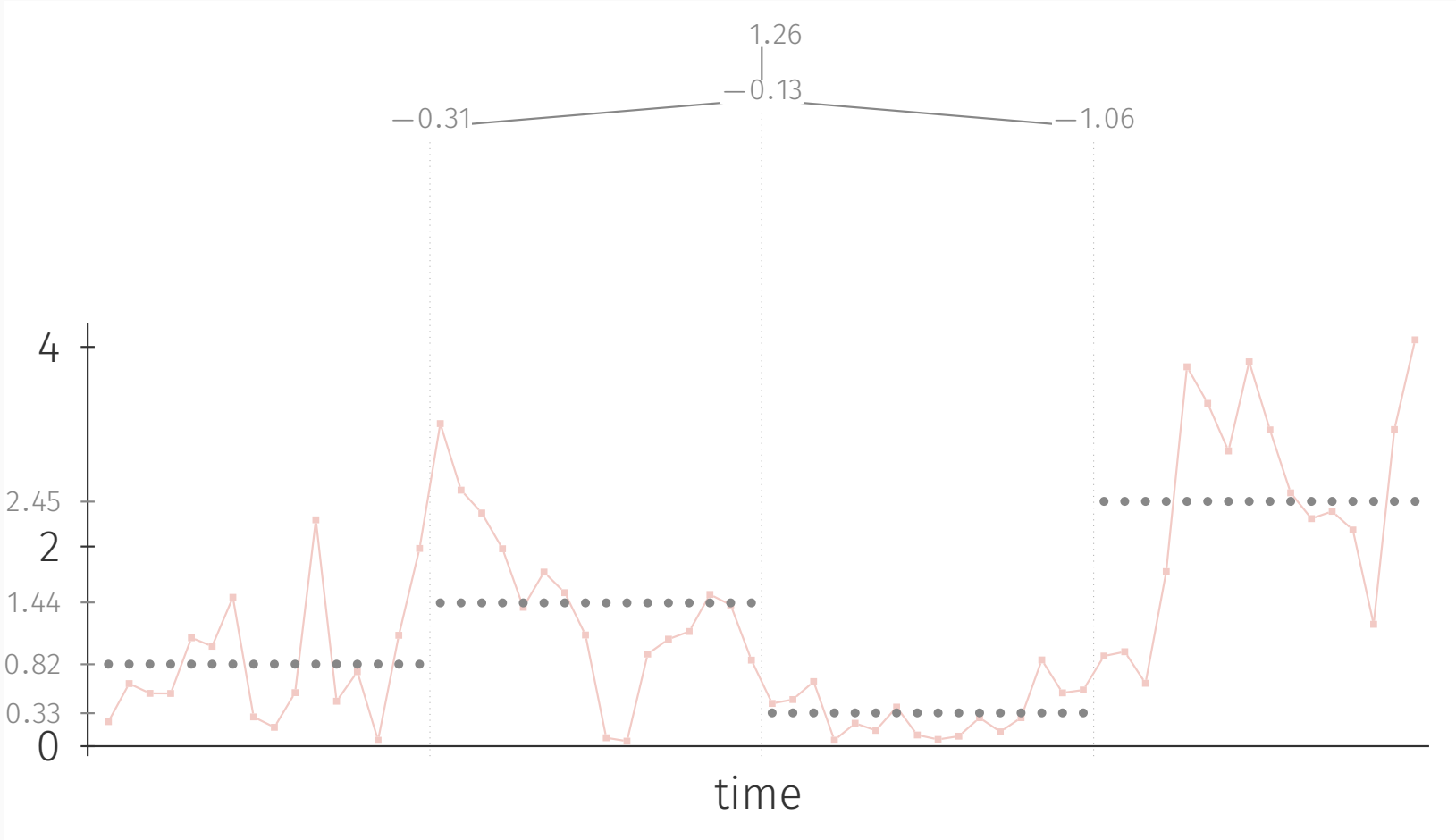
Discrete wavelet transform (DWT)

Adding the difference between first and second half allows to reconstruct the average values during both halves



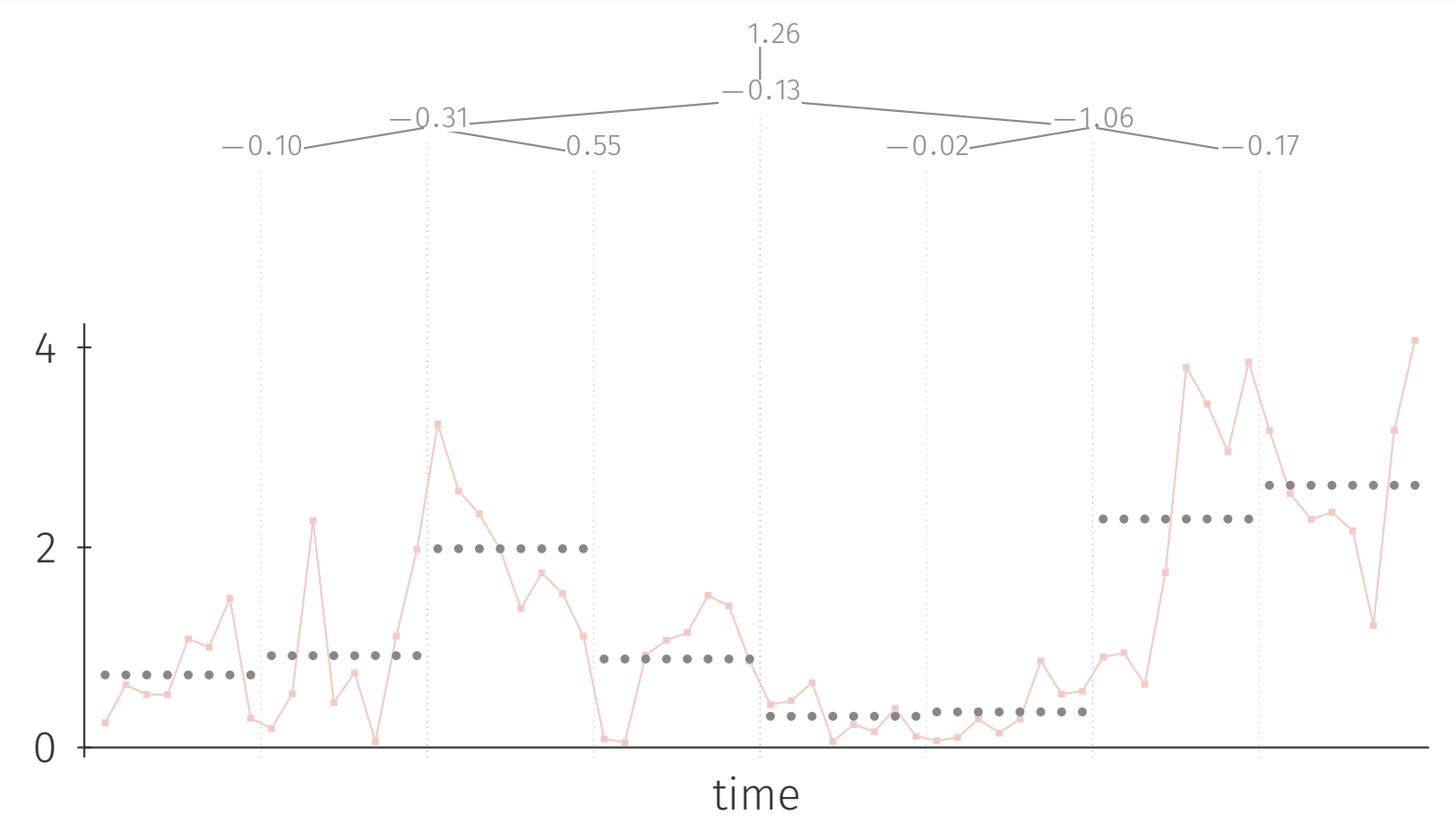
Discrete wavelet transform (DWT)

Adding the difference between first and second quarter as well as between third and fourth quarter...



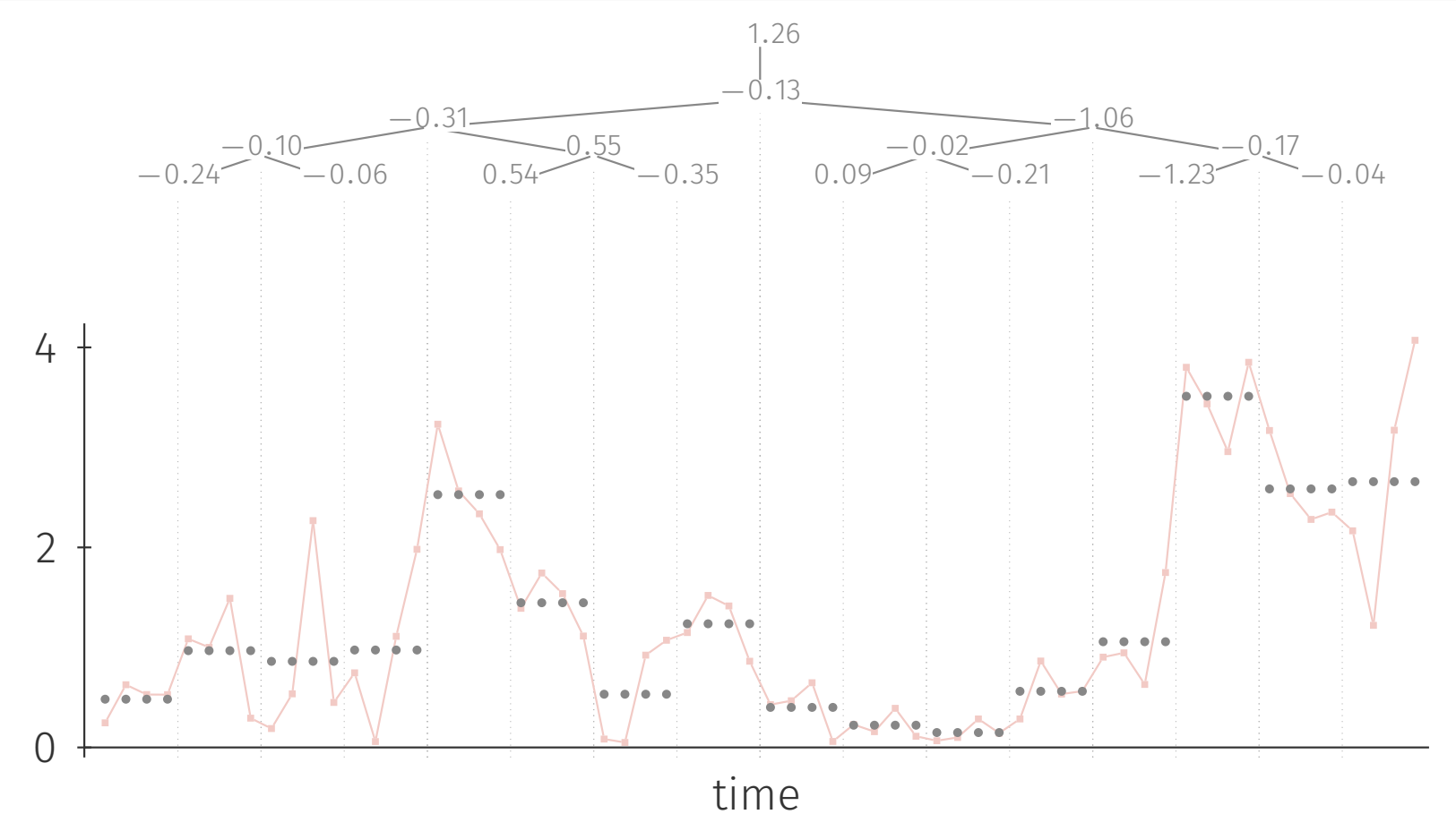
Discrete wavelet transform (DWT)

This process can be applied recursively...



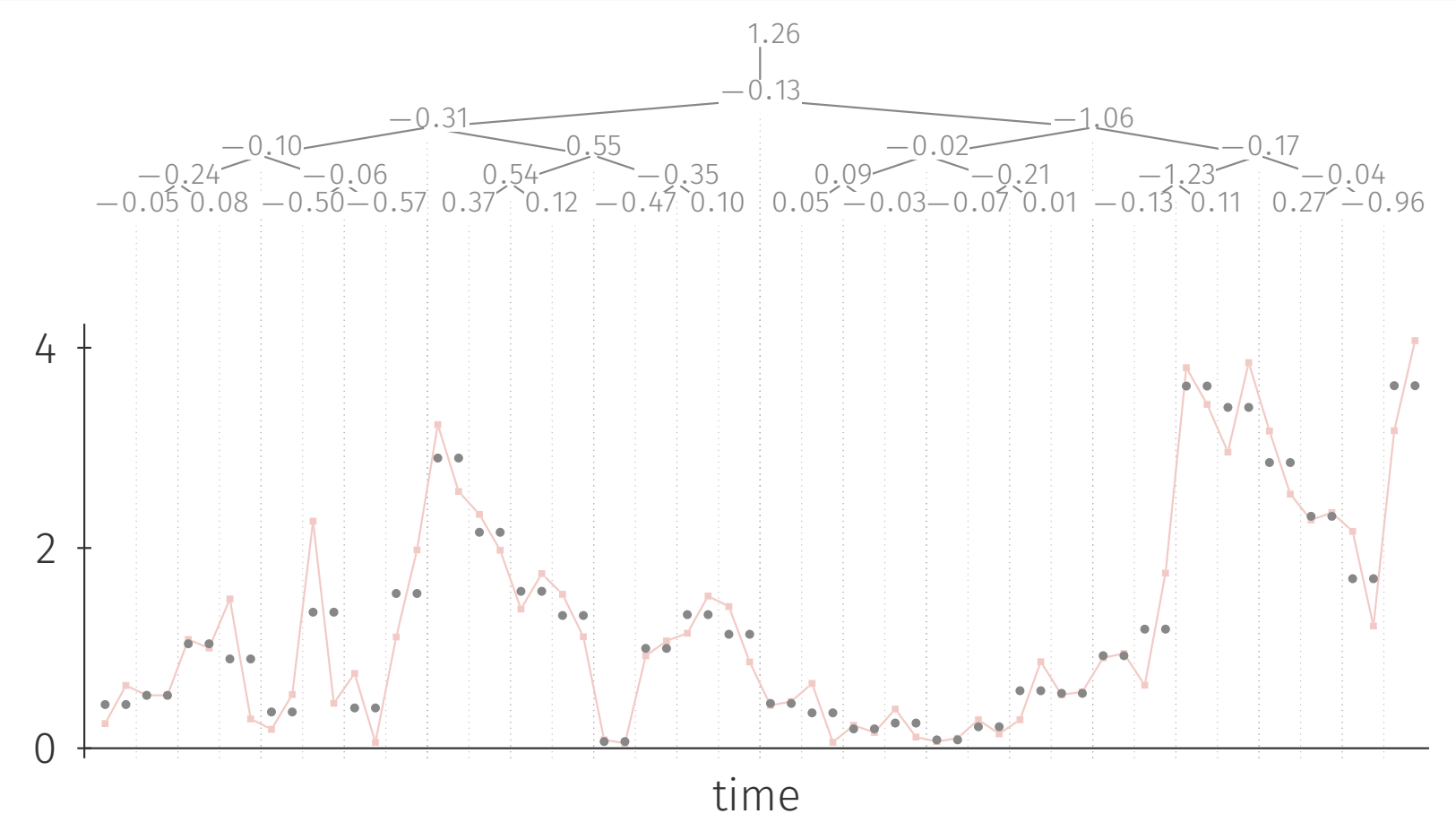
Discrete wavelet transform (DWT)

This process can be applied recursively...



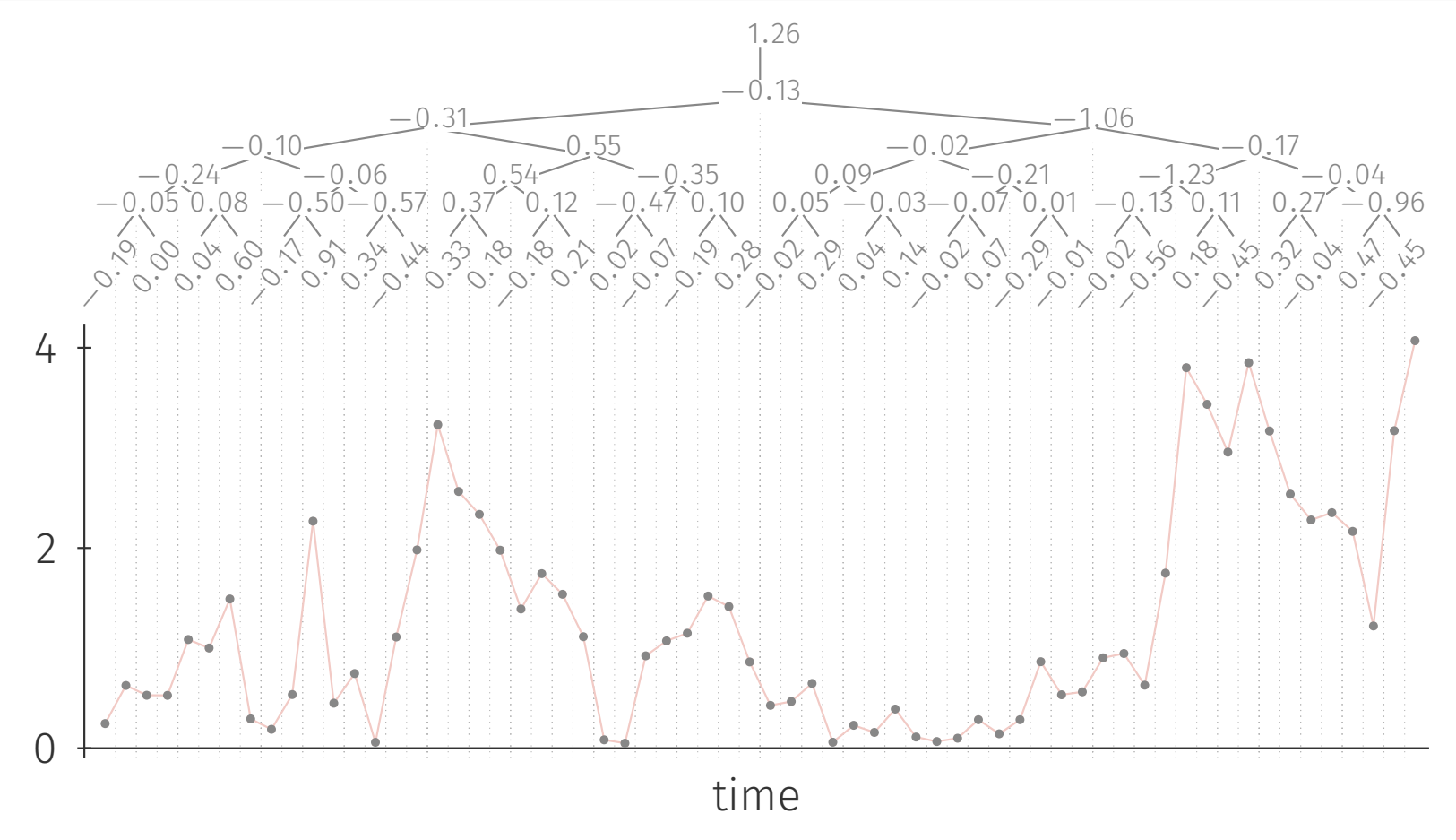
Discrete wavelet transform (DWT)

This process can be applied recursively...



Discrete wavelet transform (DWT)

This process can be applied recursively...



Discrete wavelet transform (DWT)

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

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

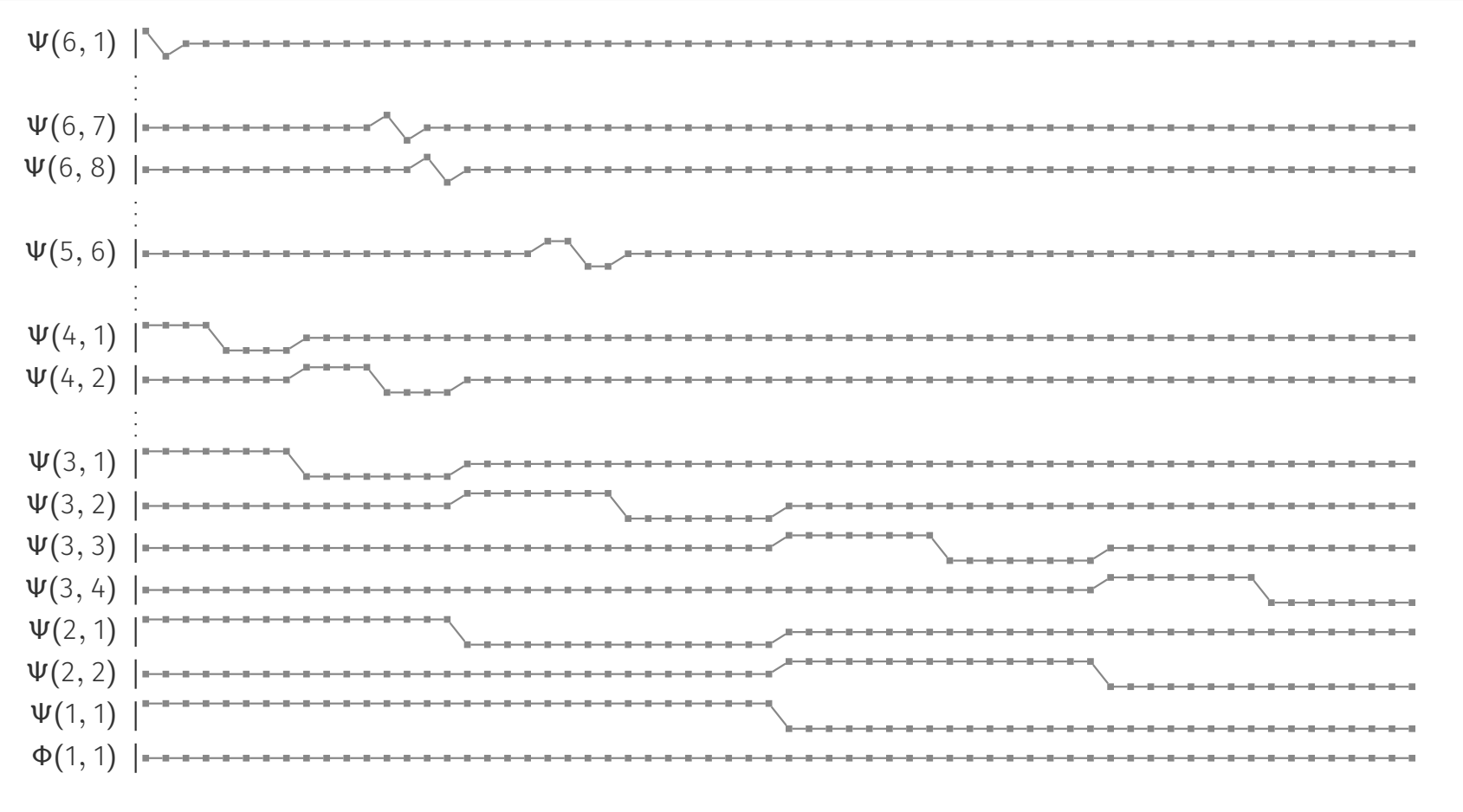
Let $\Phi(k, i)$ be the average value of this segment

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

$\Phi(1, 1)$ is the global average

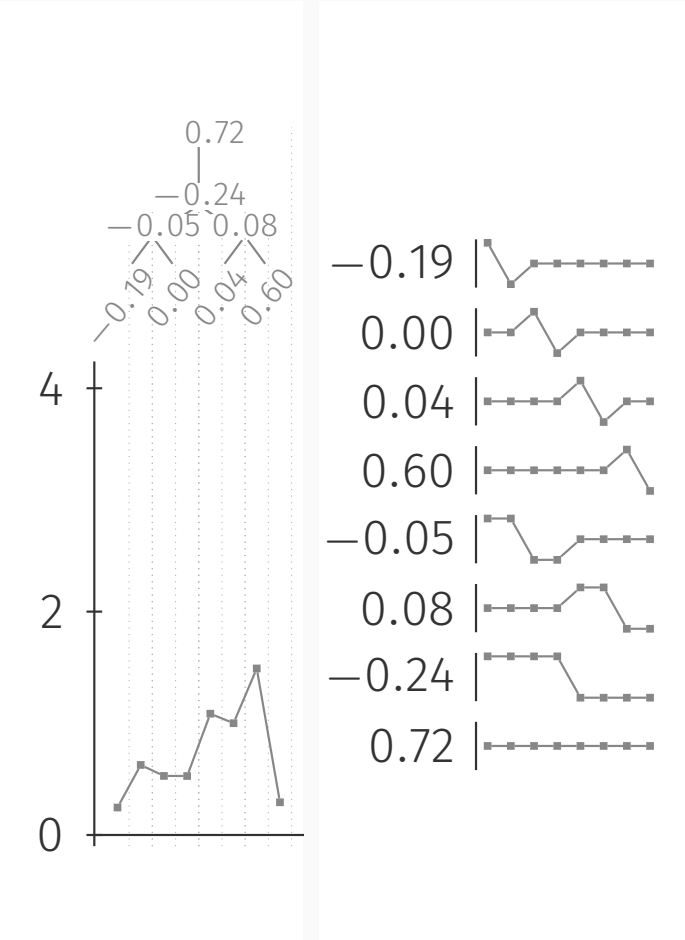
Discrete wavelet transform (DWT)

This process decomposes the time-series into a collection of *wavelets* with different widths, offsets and weights



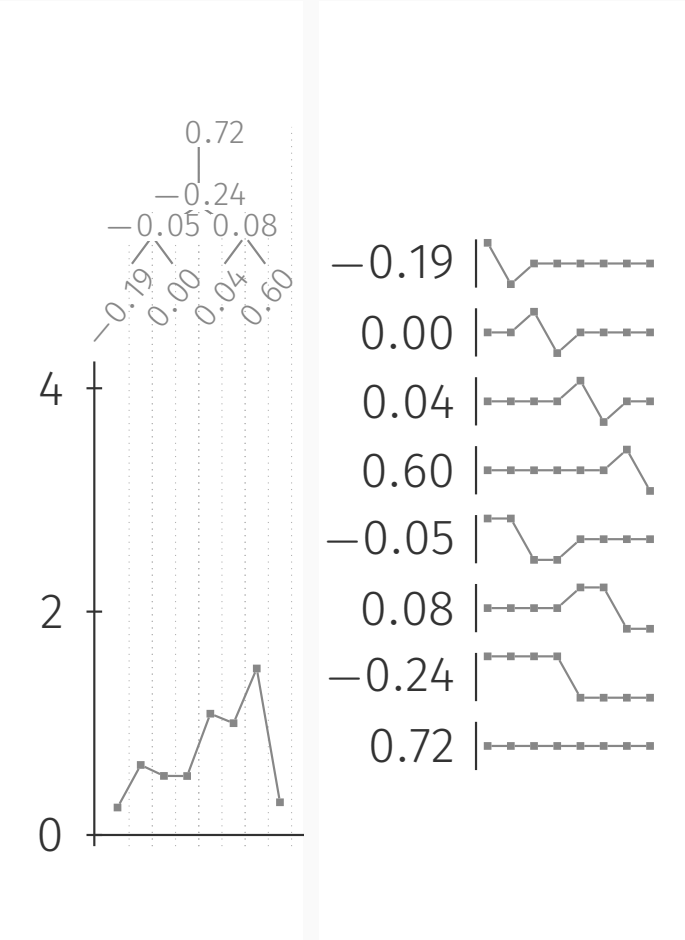
Discrete wavelet transform (DWT)

This process decomposes the time-series into a collection of *wavelets* with different widths, offsets and weights



Discrete wavelet transform (DWT)

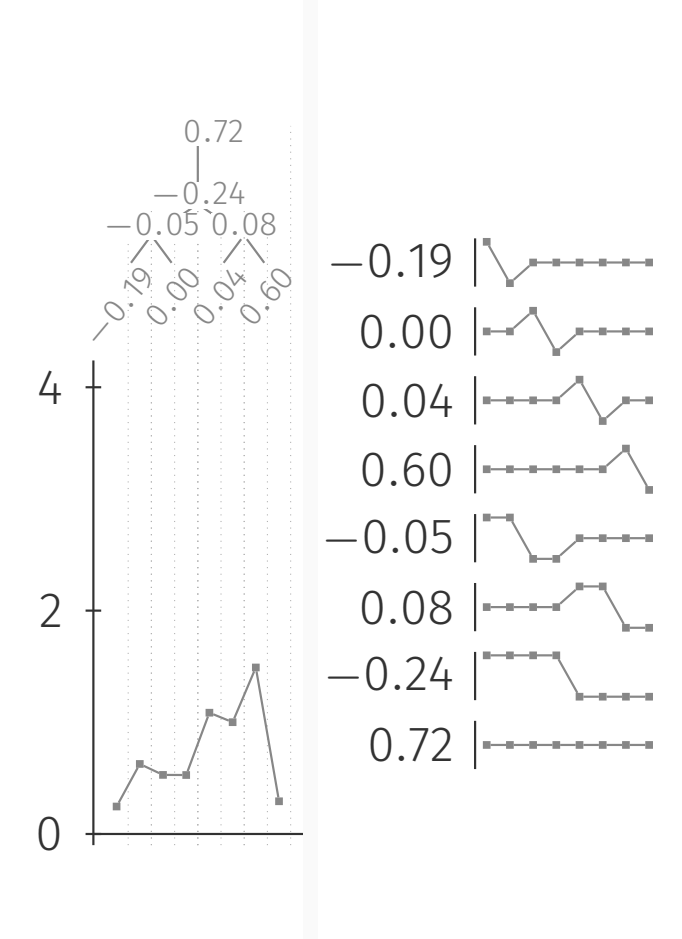
The original time-series can be reconstructed by summing all the weighted wavelets



Discrete wavelet transform (DWT)

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

Vector $\boldsymbol{a}$ contains the weights for the different wavelets



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

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 $\boldsymbol{a}^T \cdot W$

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

$a_i \left\| \boldsymbol{w}^{(i)} \right\|$ are the normalized weights
$\boldsymbol{w}^{(i)} / \left\| \boldsymbol{w}^{(i)} \right\|$ are the normalized basis vectors

Discrete wavelet transform (DWT)

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

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

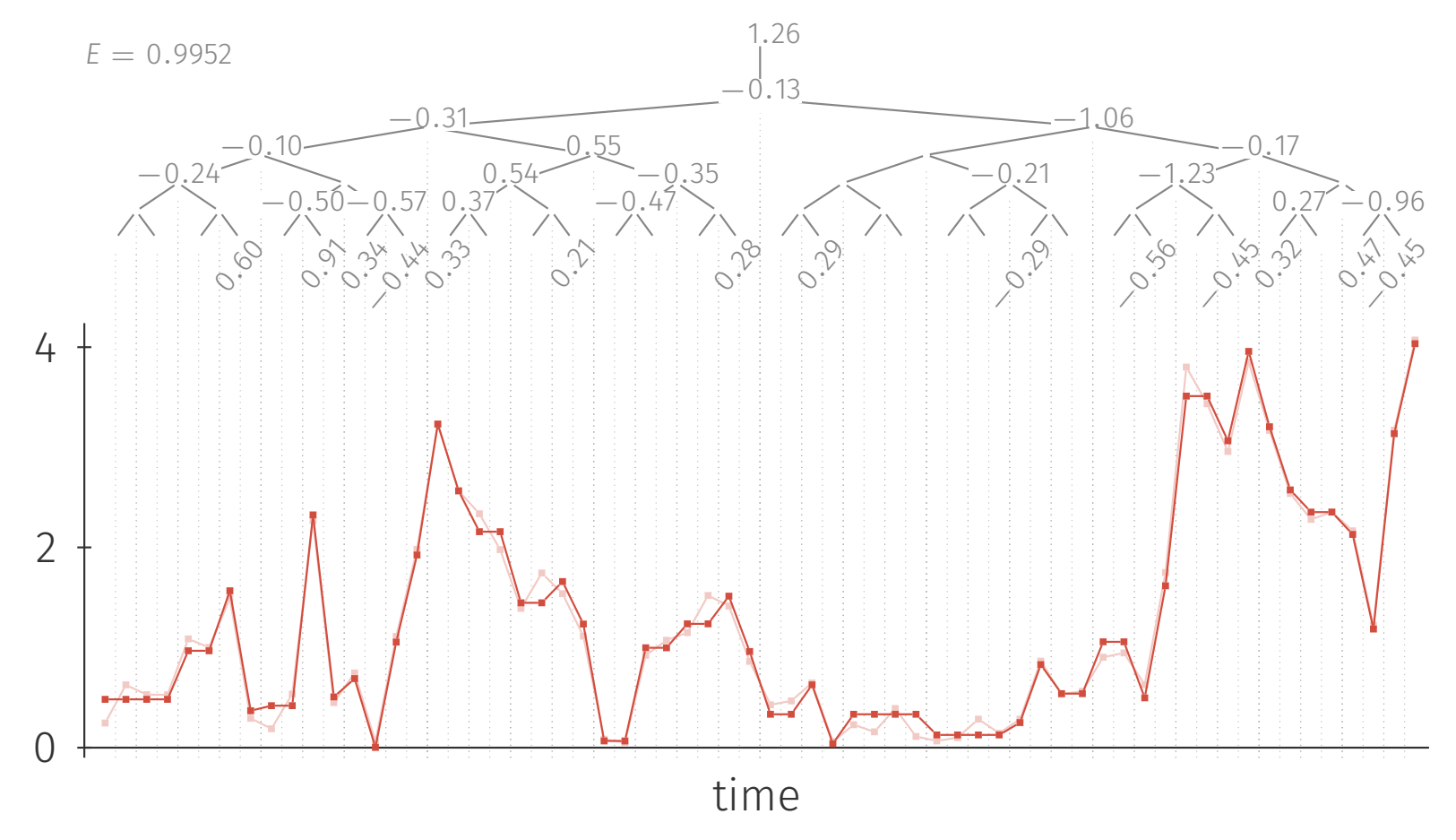
Dropping some weights reduces the dimensionality of the representation

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

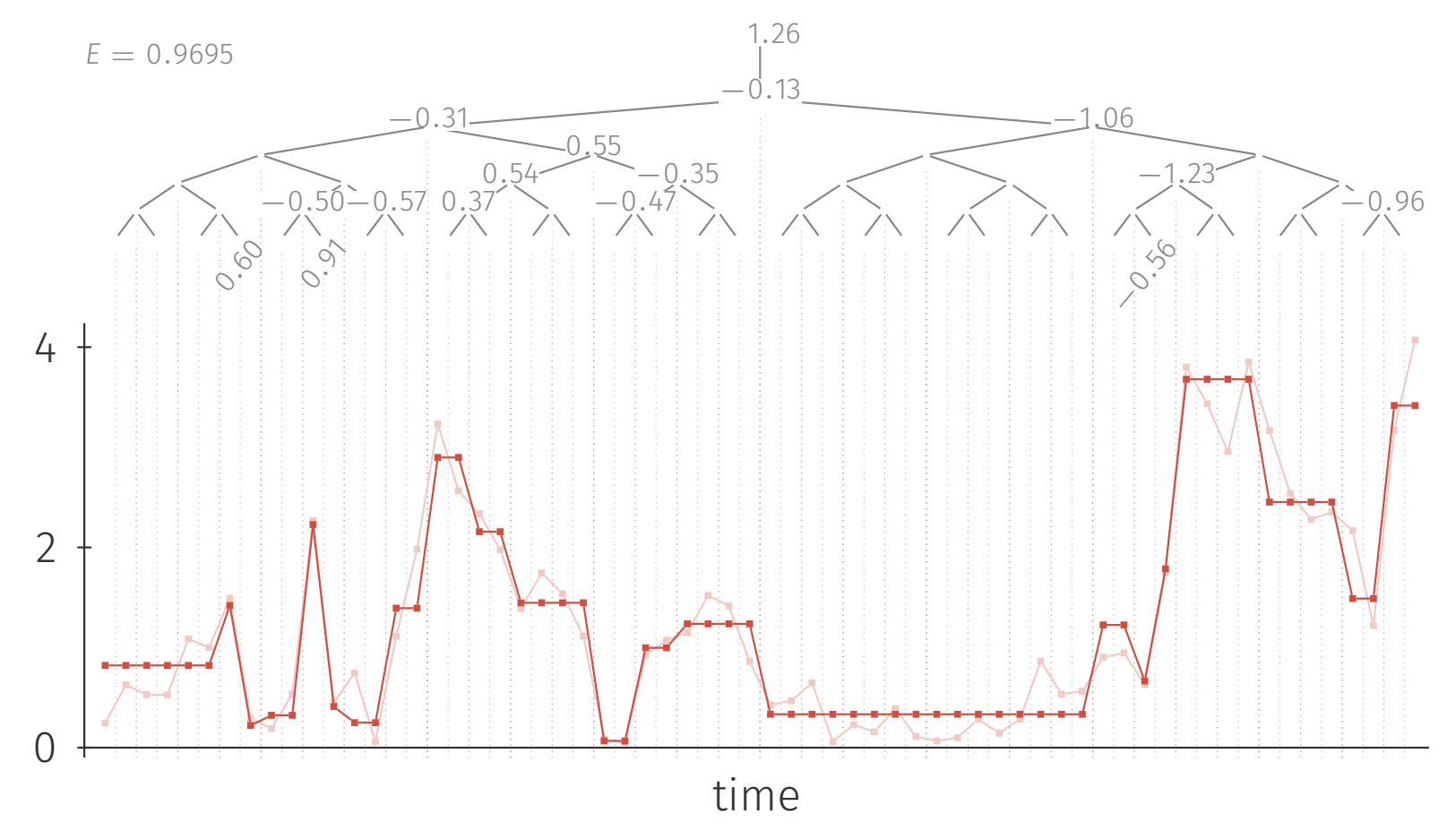
Discrete wavelet transform (DWT)

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



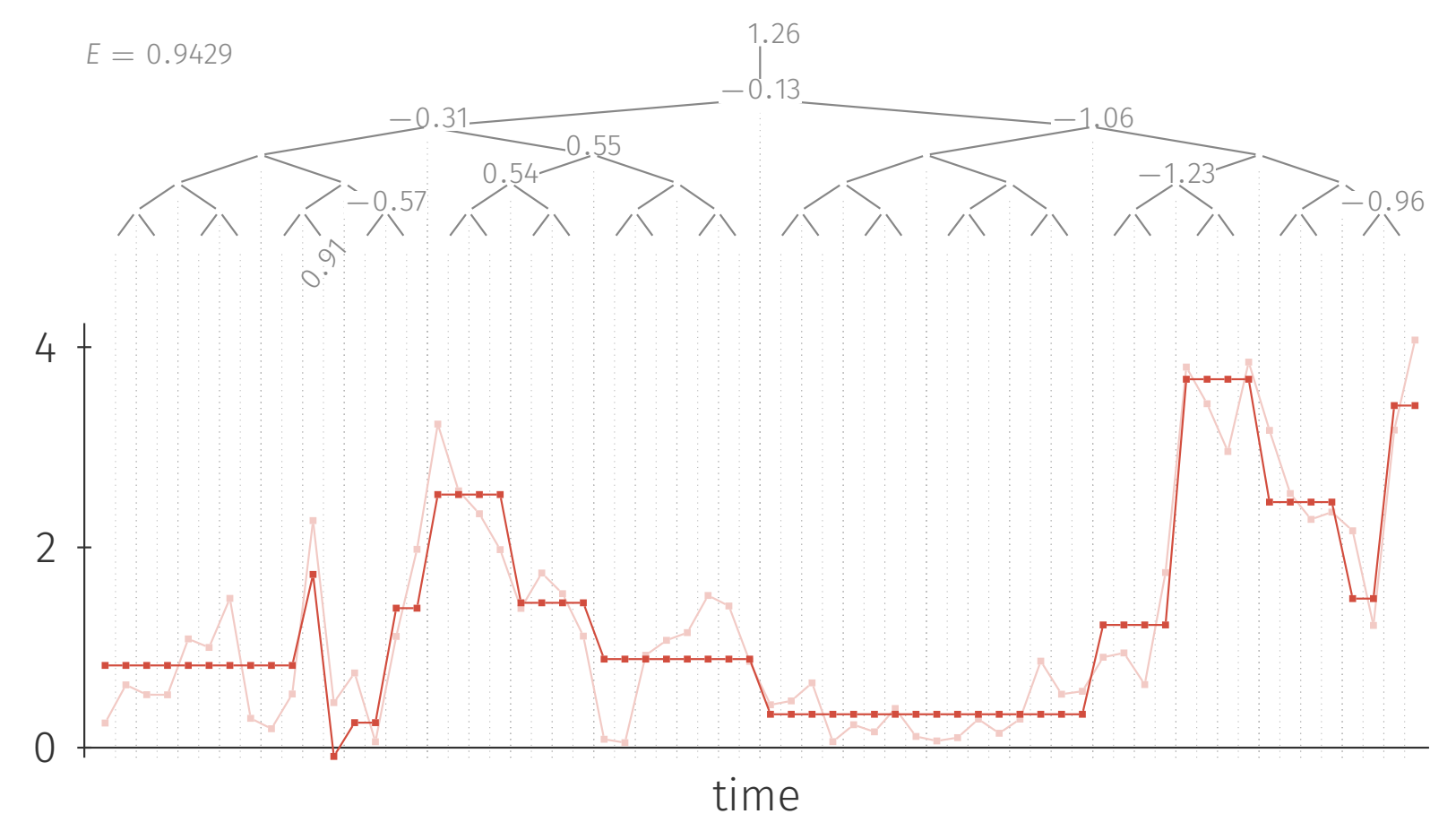
Discrete wavelet transform (DWT)

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



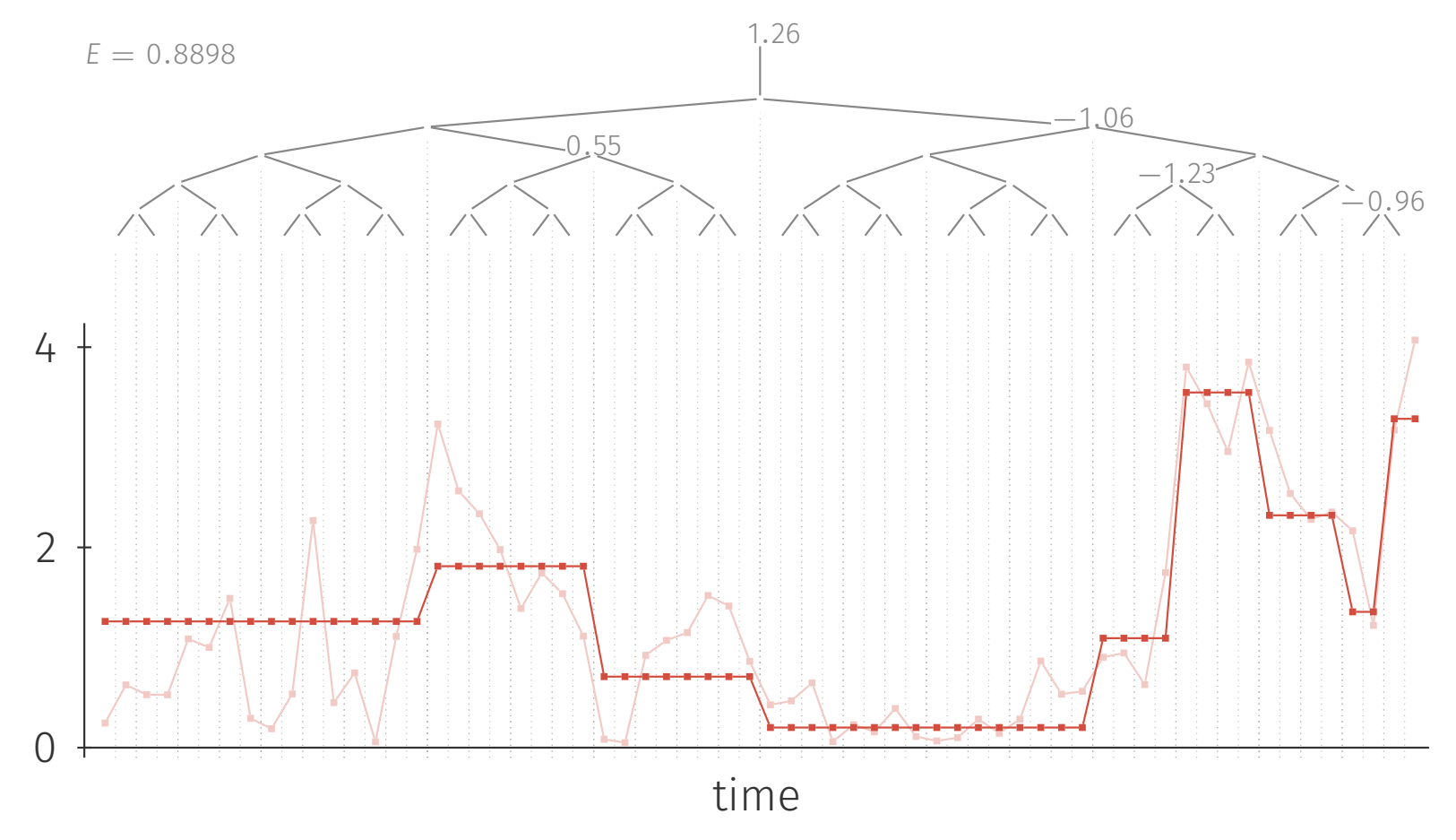
Discrete wavelet transform (DWT)

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



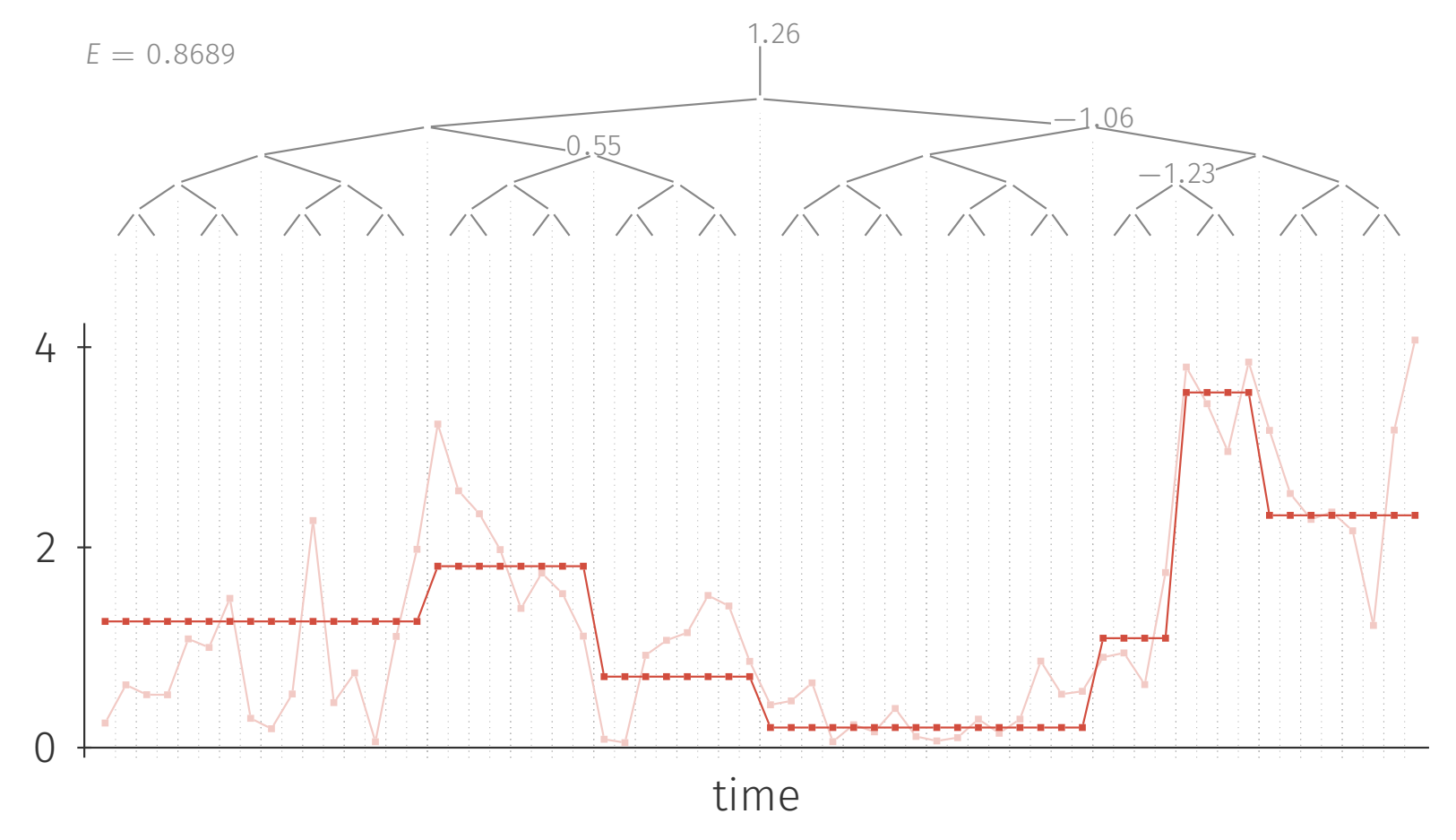
Discrete wavelet transform (DWT)

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



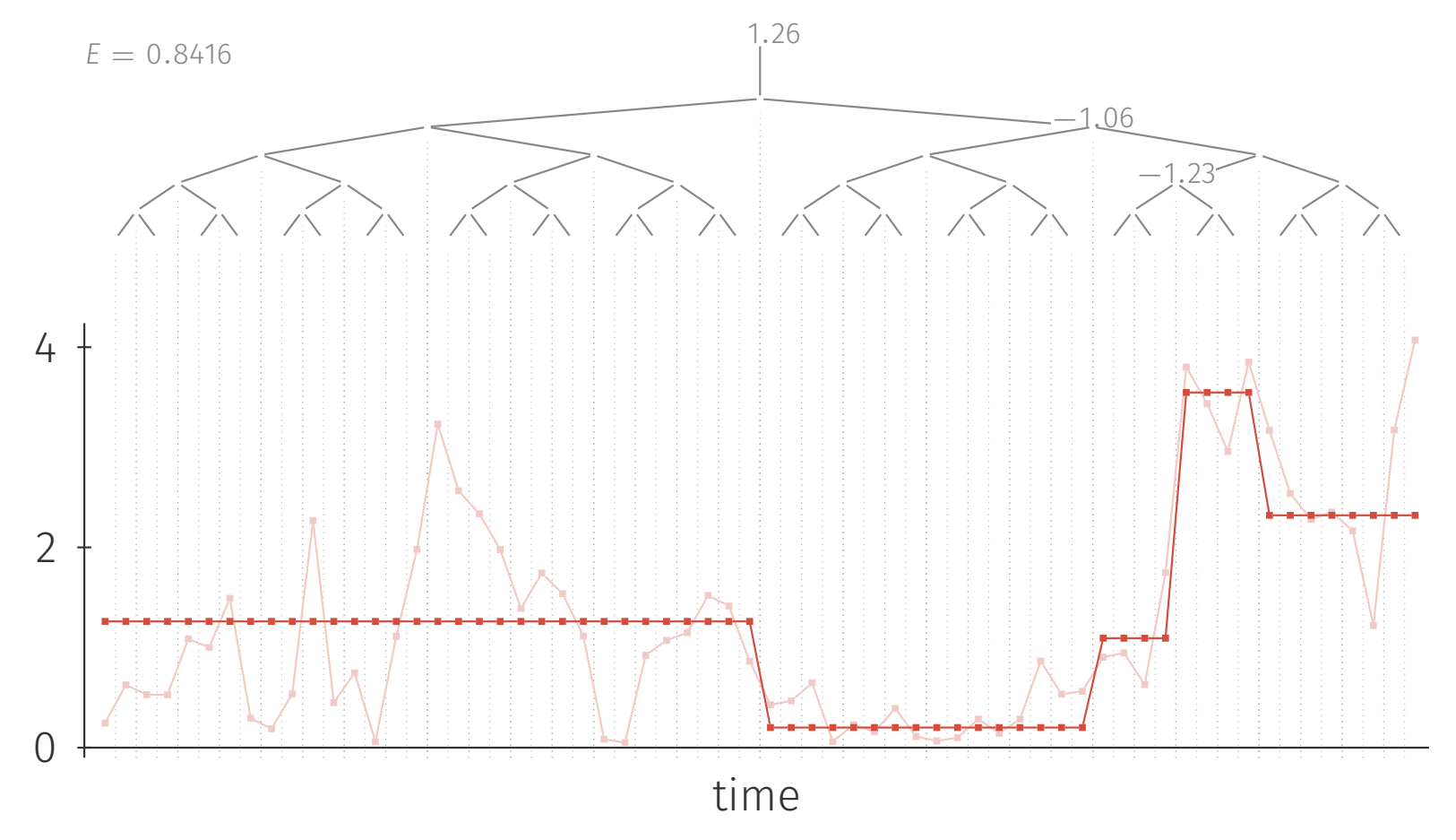
Discrete wavelet transform (DWT)

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



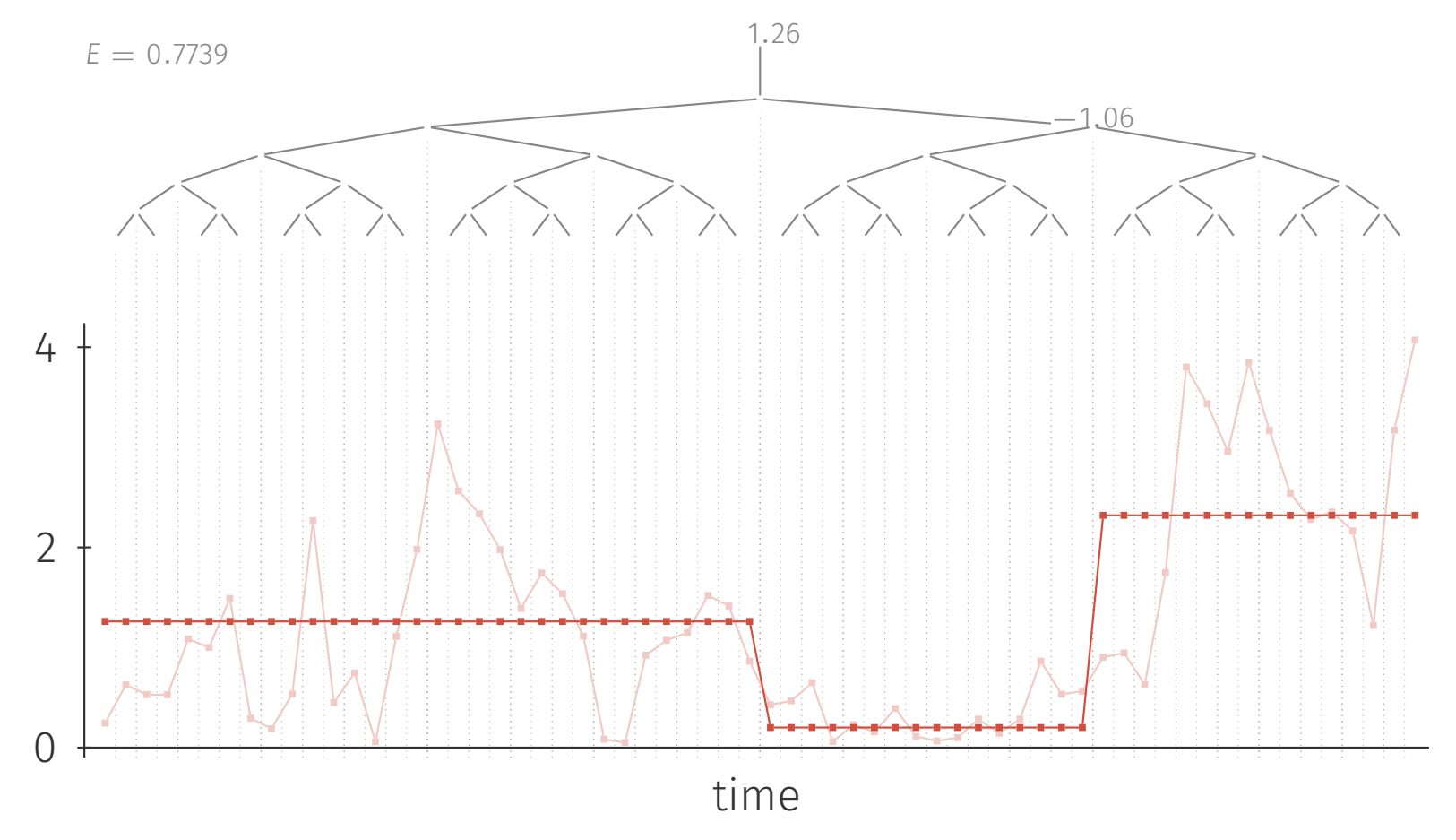
Discrete wavelet transform (DWT)

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



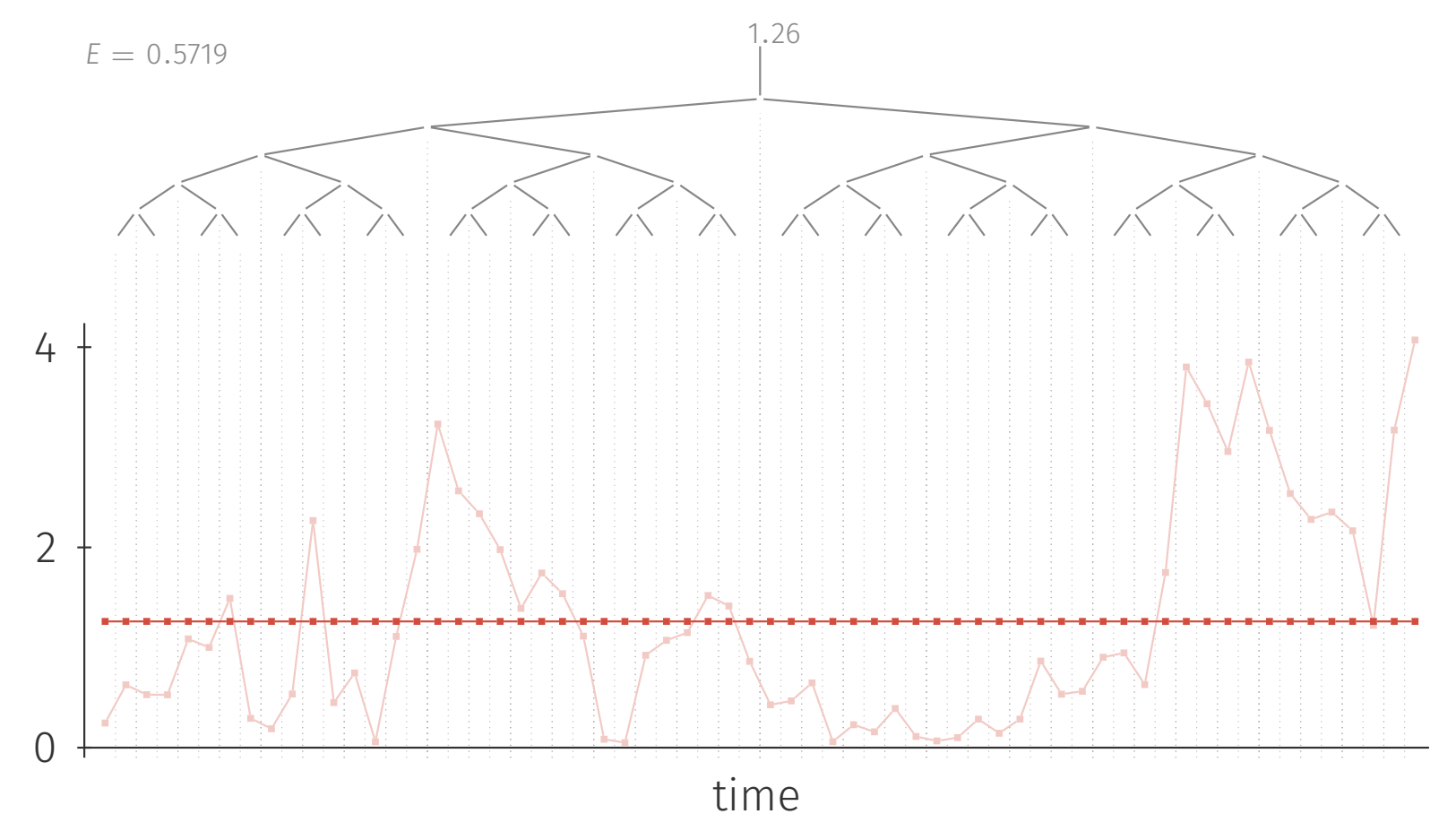
Discrete wavelet transform (DWT)

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

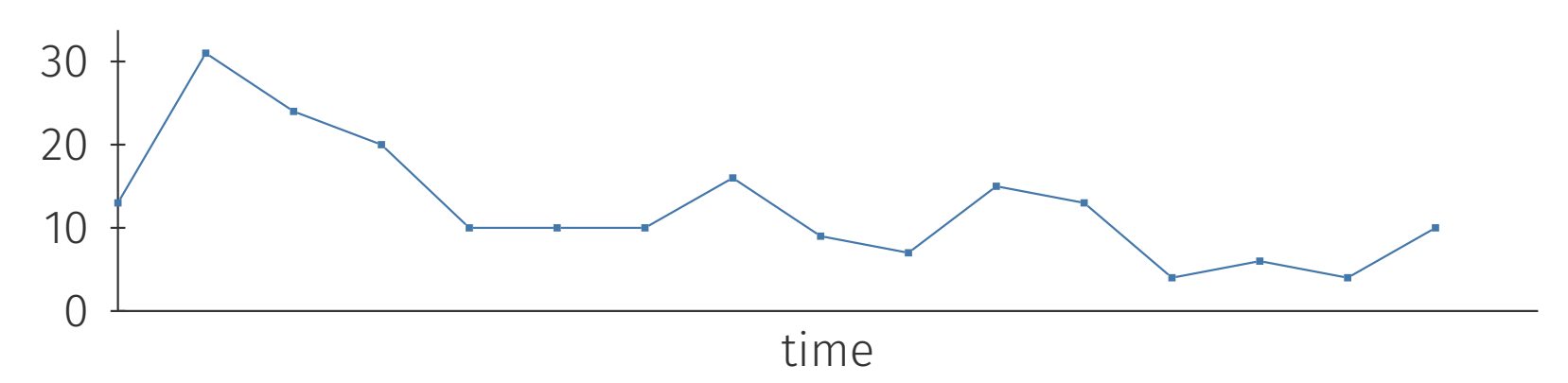


Discrete wavelet transform (DWT)

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



Example



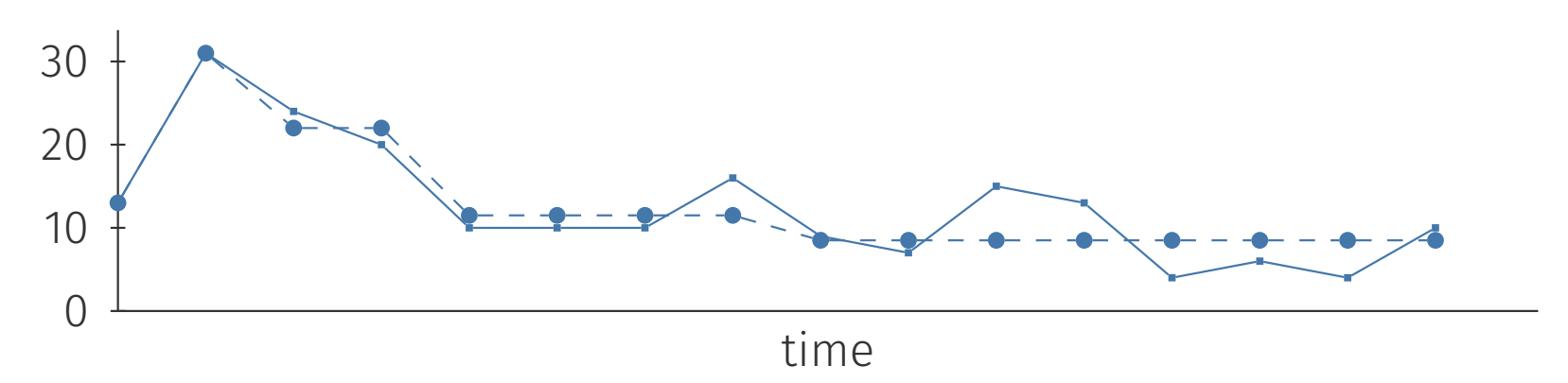
$\langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10 \rangle$

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

Example

$$a = \begin{bmatrix} -9.0 \\ 2.0 \\ 0.0 \\ -3.0 \\ 1.0 \\ 1.0 \\ -1.0 \\ -3.0 \\ 0.0 \\ -1.5 \\ -3.0 \\ -1.0 \\ 5.25 \\ 2.5 \\ 4.125 \\ 12.625 \end{bmatrix} \quad W = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad E = \begin{bmatrix} \mathbf{162.0} \\ 8.0 \\ 0.0 \\ 18.0 \\ 2.0 \\ 2.0 \\ 2.0 \\ 18.0 \\ 0.0 \\ 9.0 \\ 36.0 \\ 4.0 \\ \mathbf{220.5} \\ 50.0 \\ \mathbf{272.25} \\ \mathbf{2550.25} \end{bmatrix}$$

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

$$\langle 13.0, 31.0, 22.0, 22.0, 11.5, 11.5, 11.5, 11.5,$$
$$8.5, 8.5, 8.5, 8.5, 8.5, 8.5, 8.5, 8.5 \rangle$$

energy retained $= 95.55\%$

Discrete Fourier transform (DFT)

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

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

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

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

i denotes the imaginary number, $\mathrm{i}^2 = -1$

Discrete Fourier transform (DFT)

Each Fourier coefficient is a complex value $f_k = a_k + \mathrm{i}b_k$
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

Discrete Fourier transform (DFT)

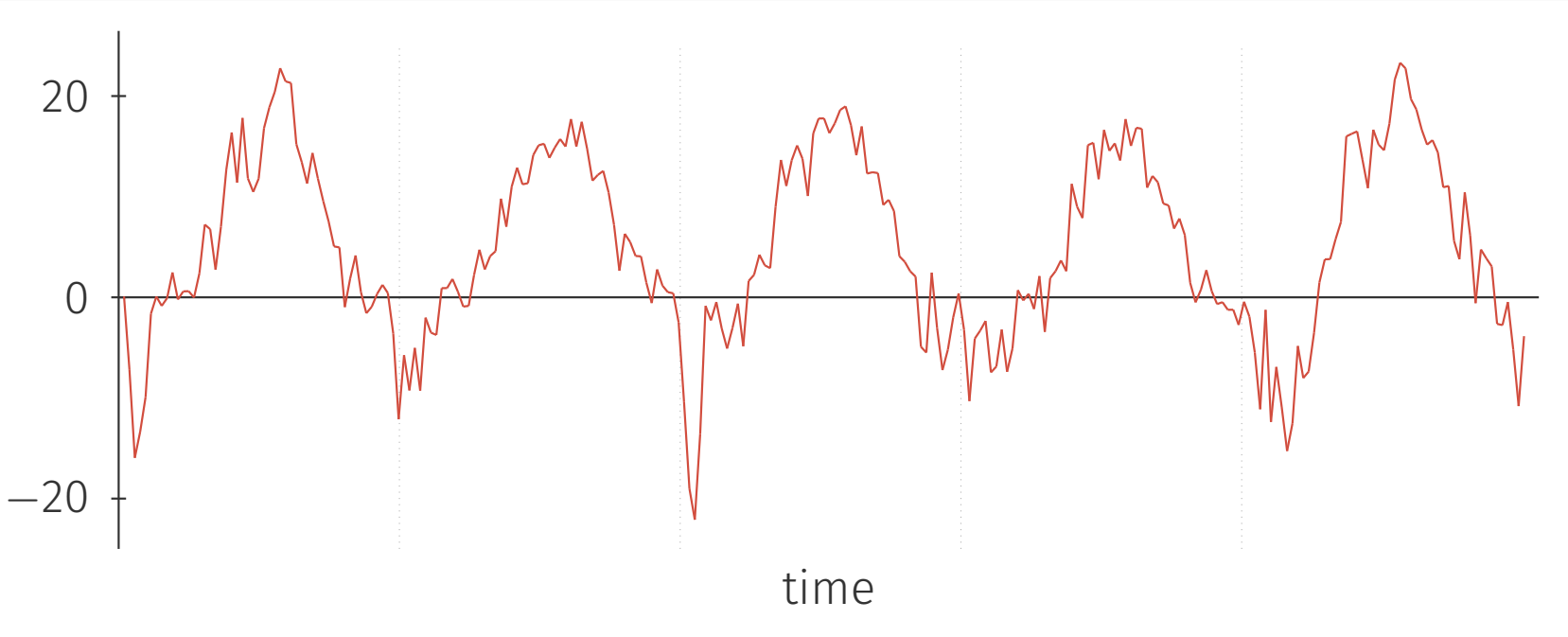
The discrete Fourier transform satisfies the *additivity property*, i.e. the Fourier coefficients of the sum of two series are the sum of their coefficients

It also satisfies *Parseval's theorem*, i.e. $$\sum_{r=0}^{n-1} x_r^2 = \frac{1}{n} \sum_{k=0}^{k-1} a_k^2 + b_k^2$$

This allows to compute the scaled Euclidean distance between two series by computing the Euclidean distance between their coefficients

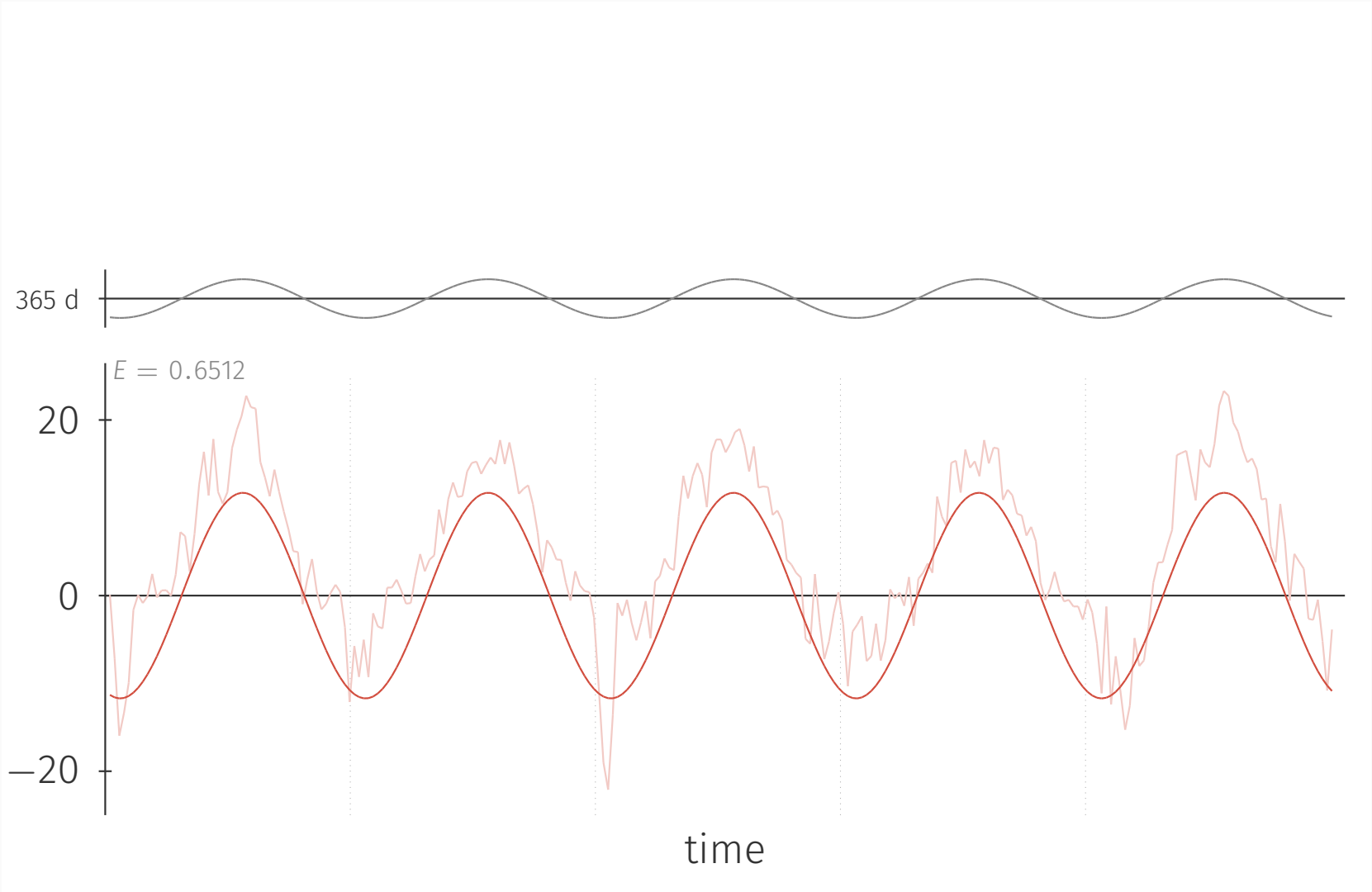
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018



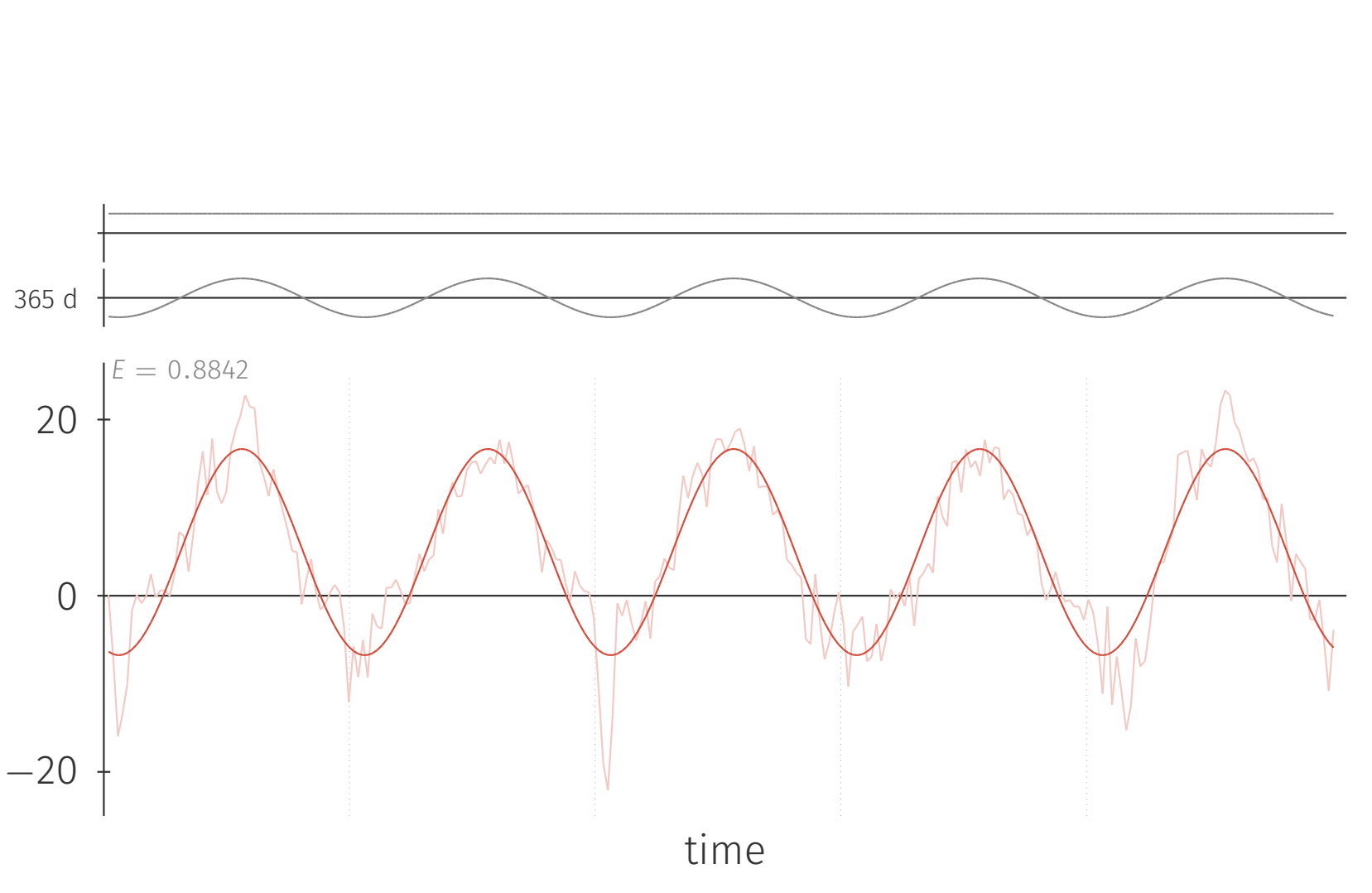
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018



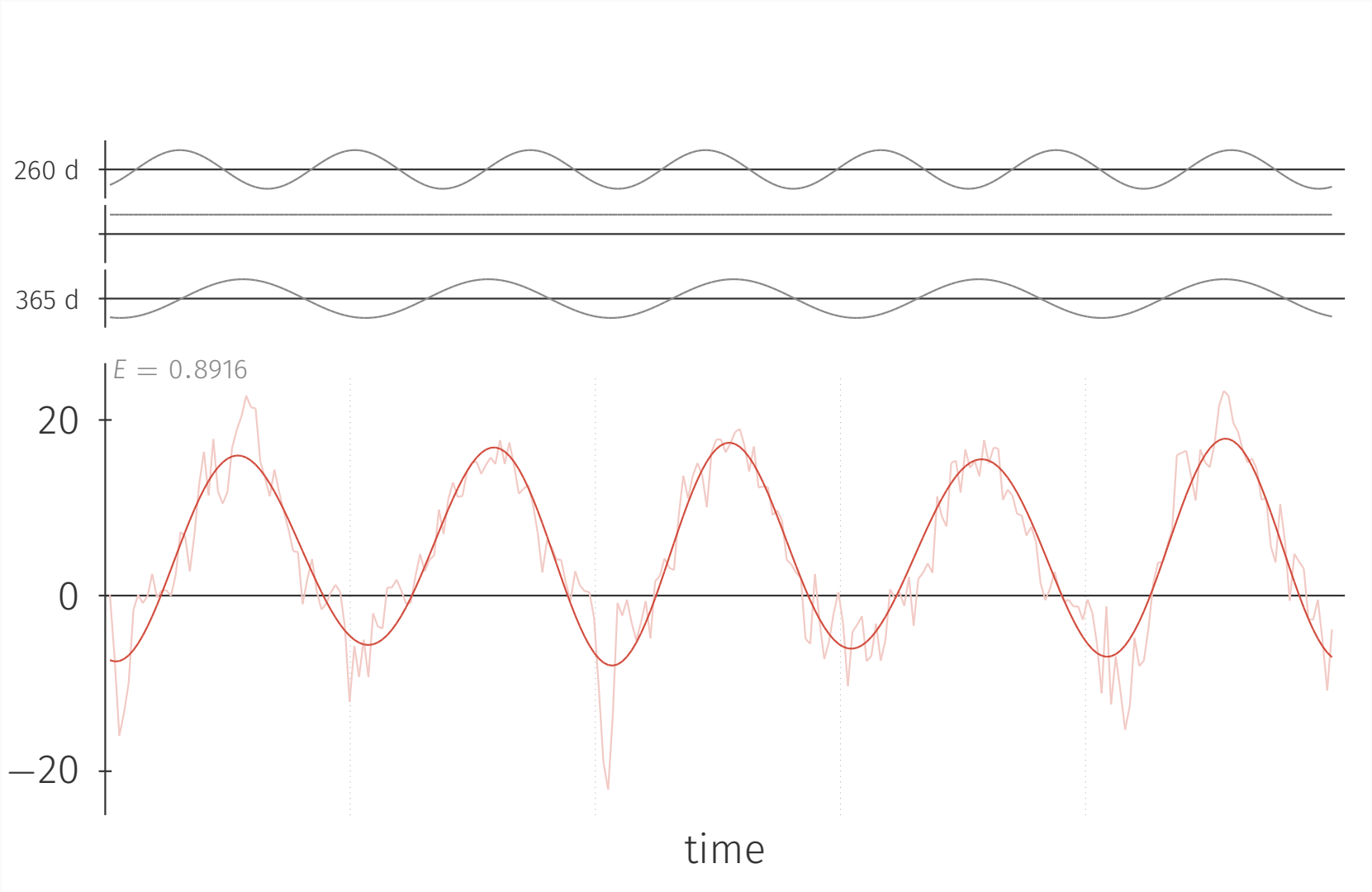
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018



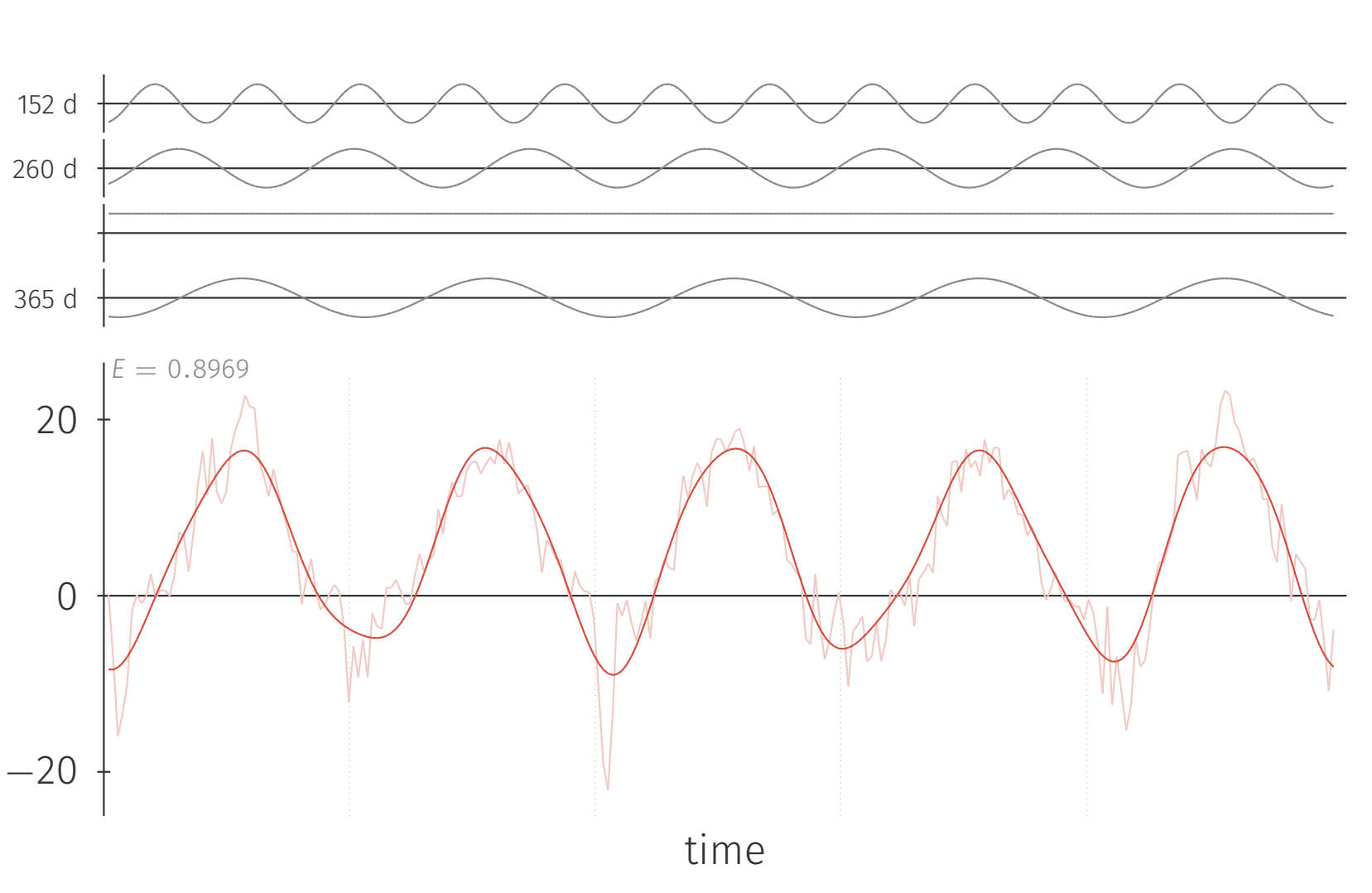
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018



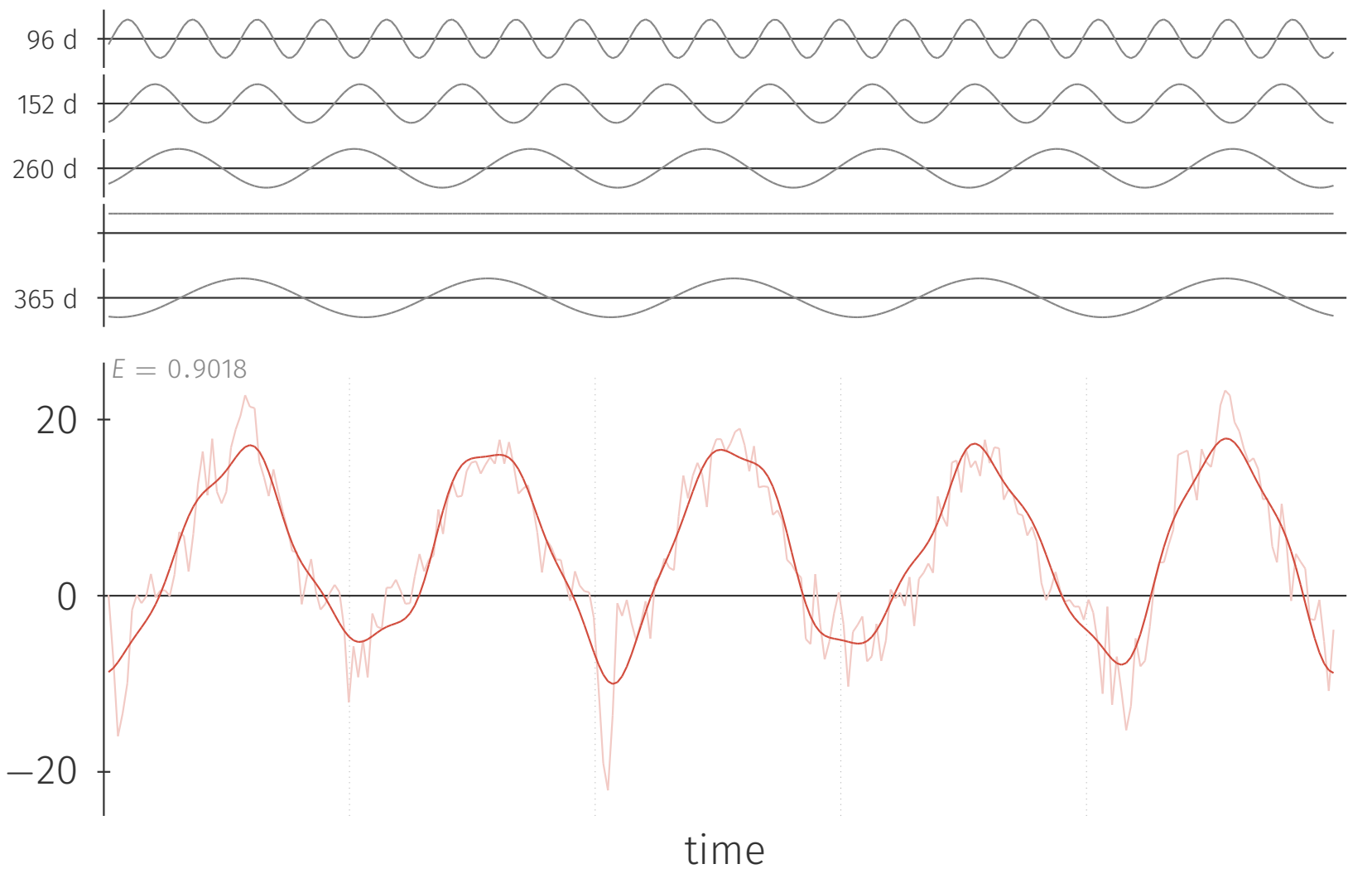
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018



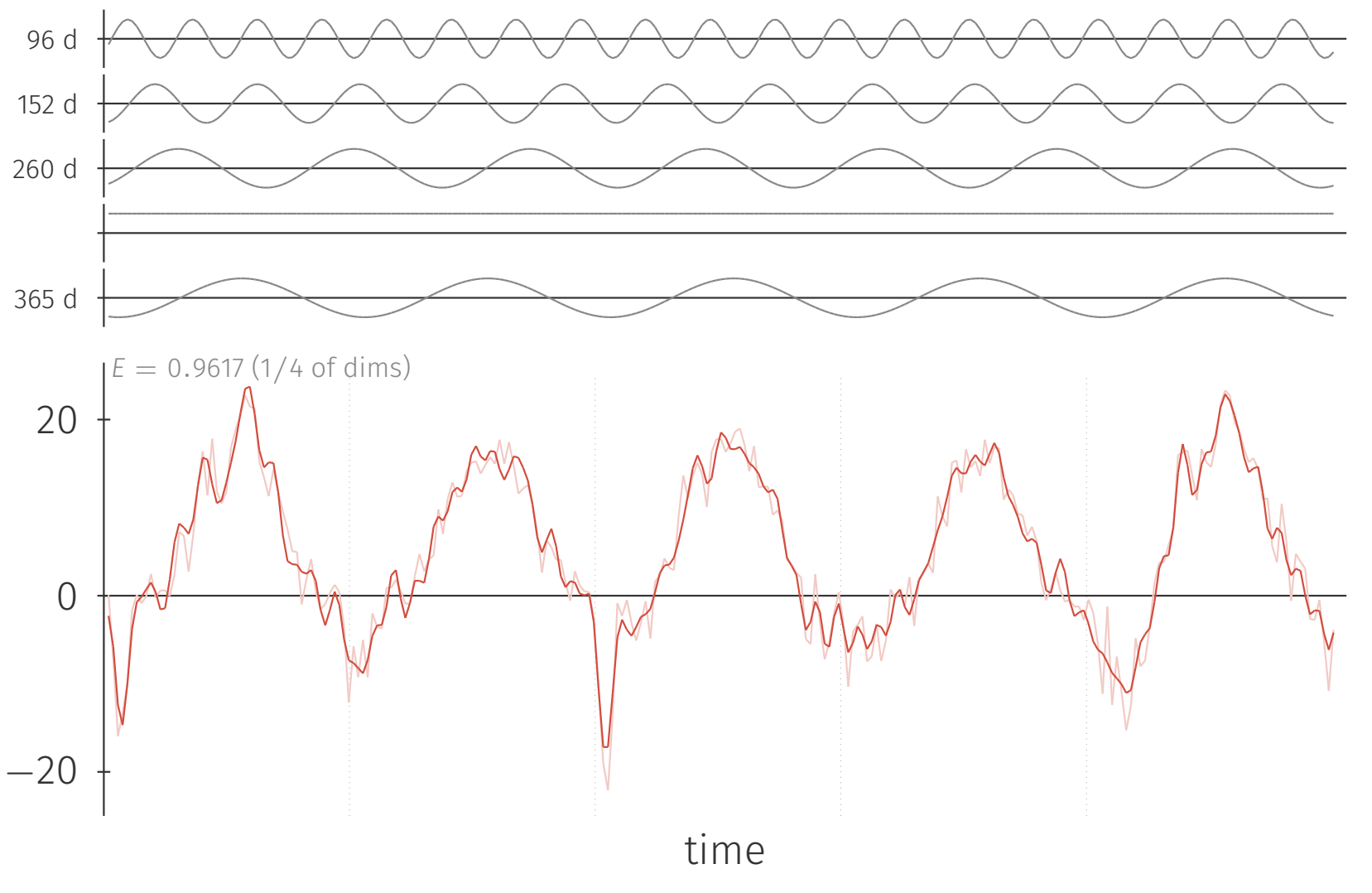
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018



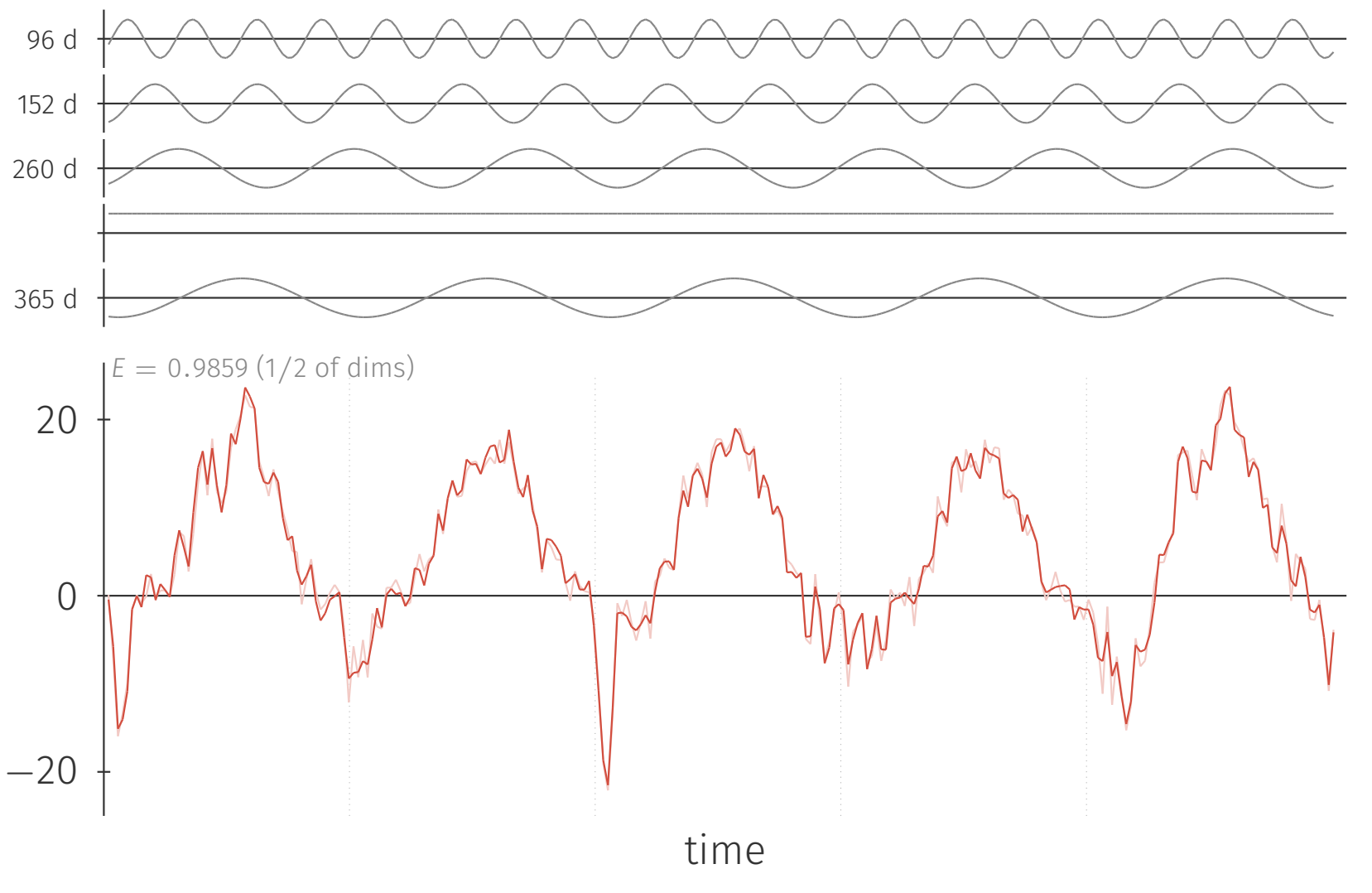
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018



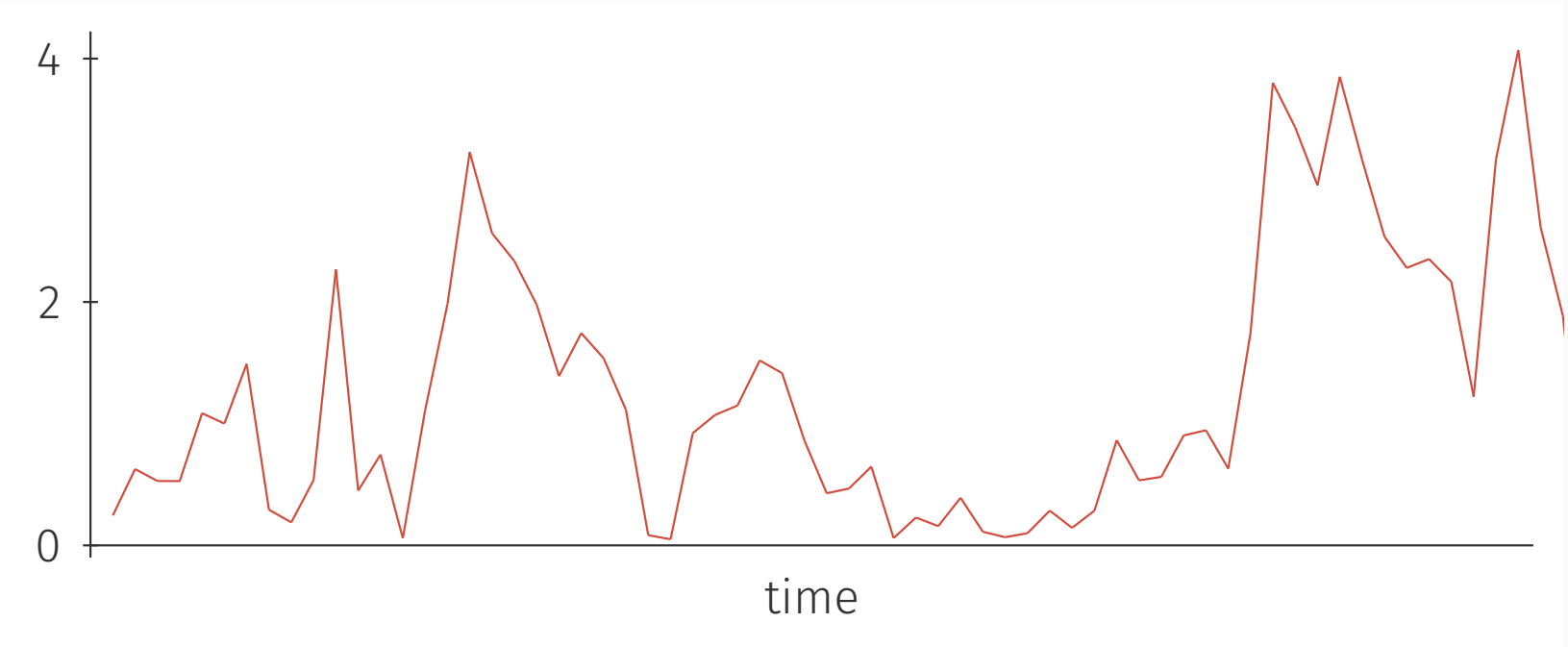
Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018



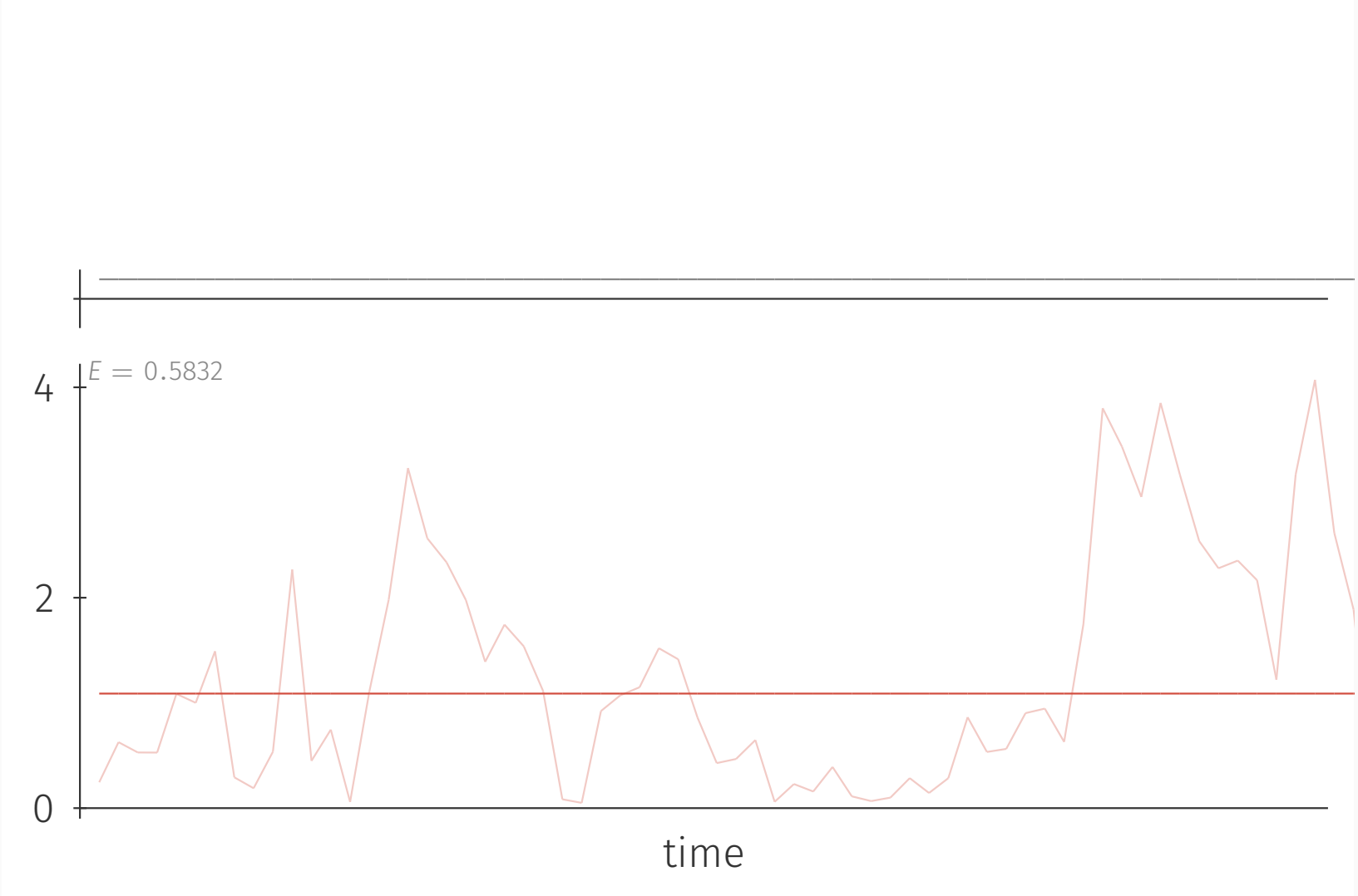
Discrete Fourier transform (DFT)

Annual copper prices during the early 19th century



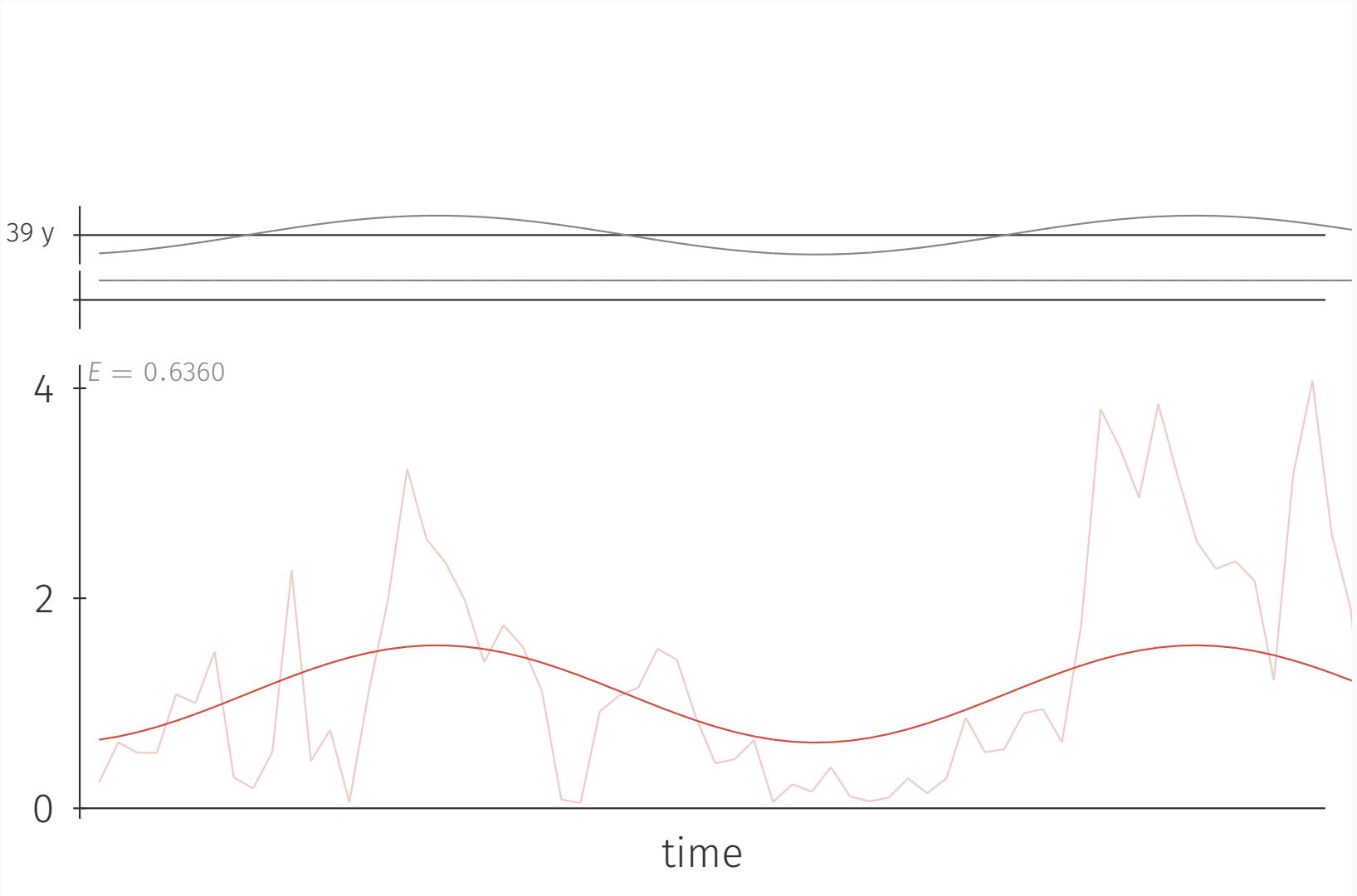
Discrete Fourier transform (DFT)

Annual copper prices during the early 19th century



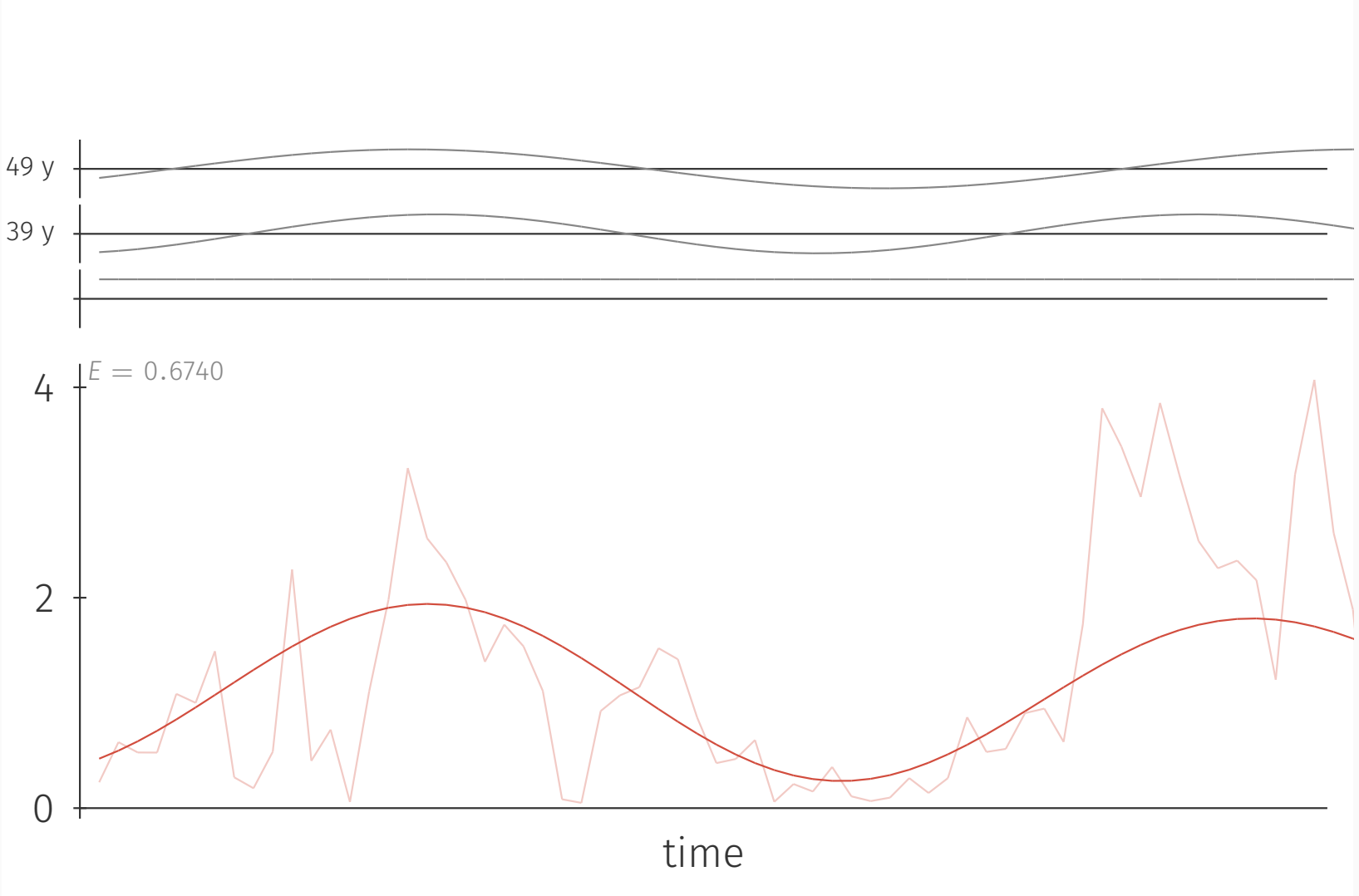
Discrete Fourier transform (DFT)

Annual copper prices during the early 19th century



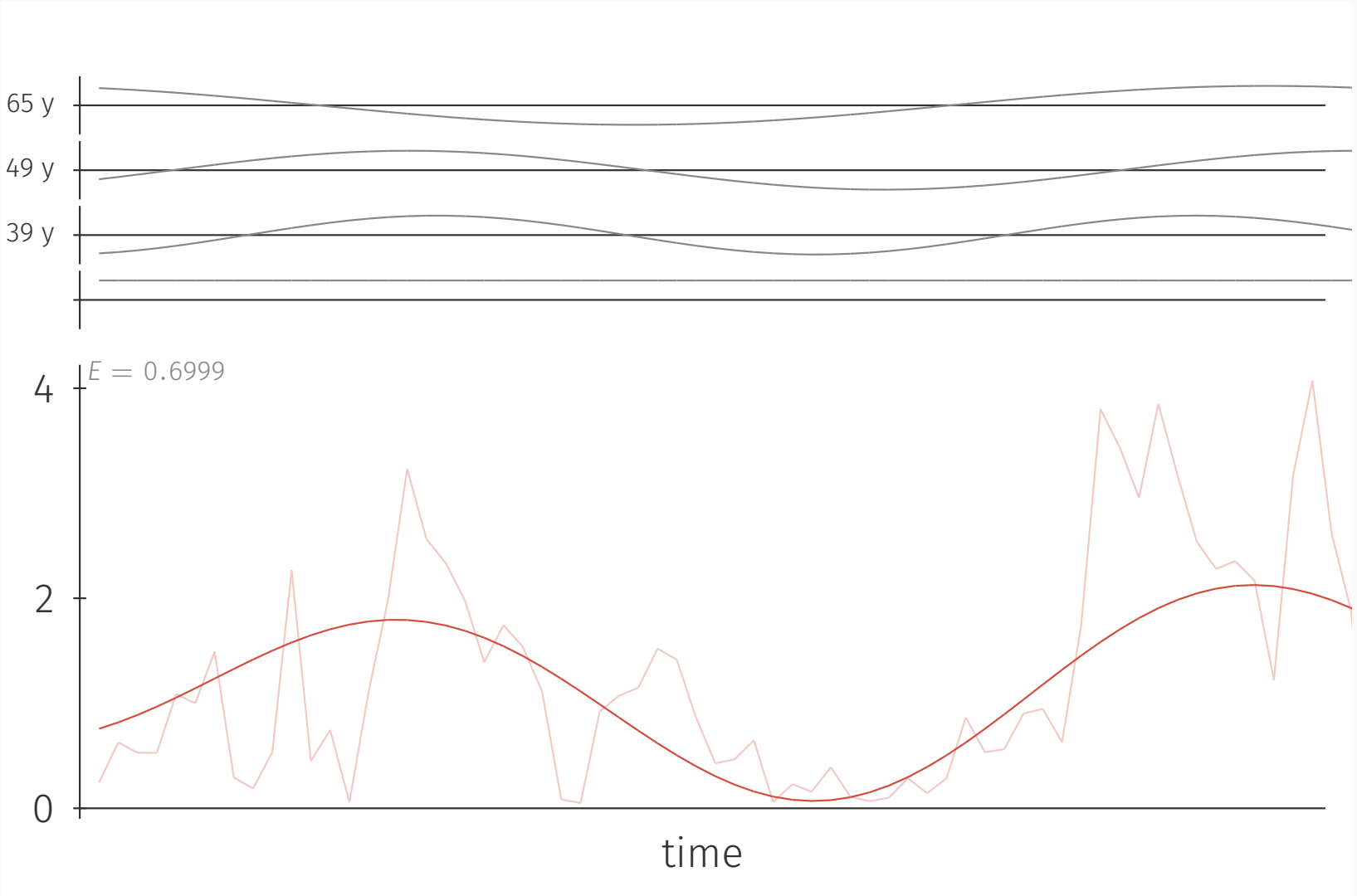
Discrete Fourier transform (DFT)

Annual copper prices during the early 19th century



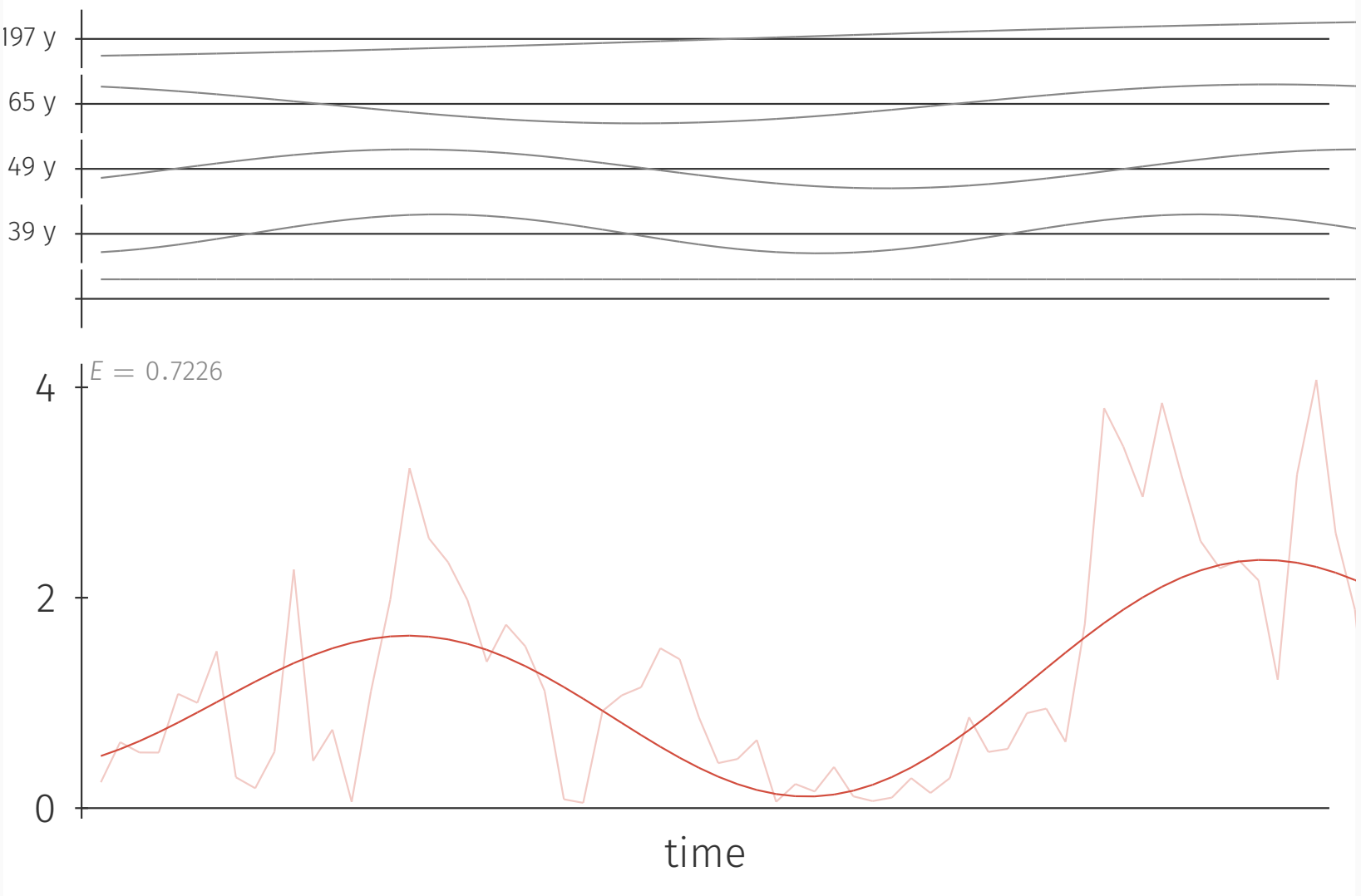
Discrete Fourier transform (DFT)

Annual copper prices during the early 19th century



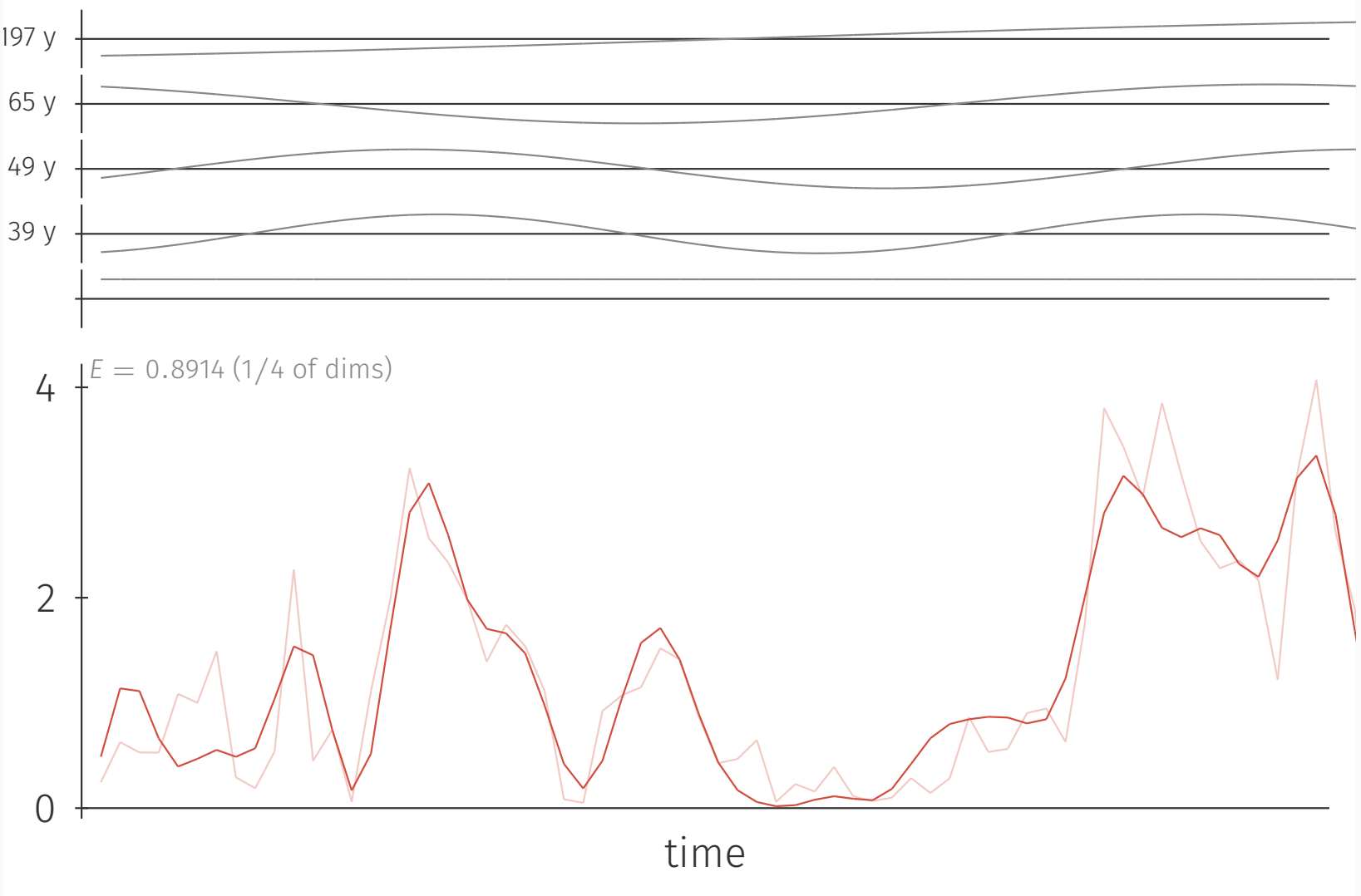
Discrete Fourier transform (DFT)

Annual copper prices during the early 19th century



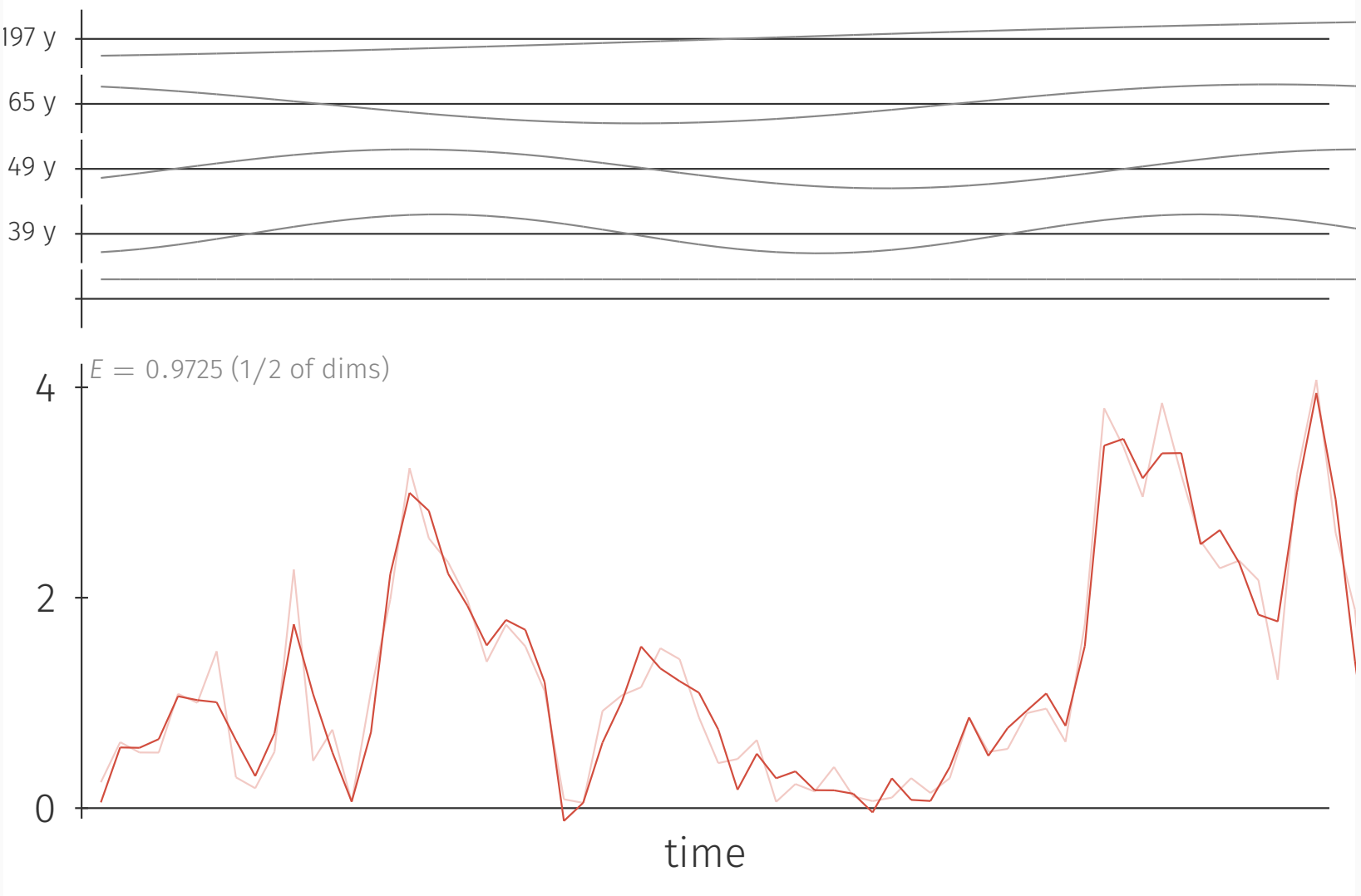
Discrete Fourier transform (DFT)

Annual copper prices during the early 19th century



Discrete Fourier transform (DFT)

Annual copper prices during the early 19th century



Models for time-series

Models for univariate time-series

Given a univariate time-series $\mathcal{S}_X = \langle x_1, x_2, \ldots, x_n \rangle$, with $x_i \in \mathbb{R}$, the aim is to predict x_{n+1}

Stationarity

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 τ

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

Stationarity

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 τ

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 $\mathcal{S}_X$

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

Higher order differencing can also be used
e.g. second-order differencing of $\mathcal{S}_X$

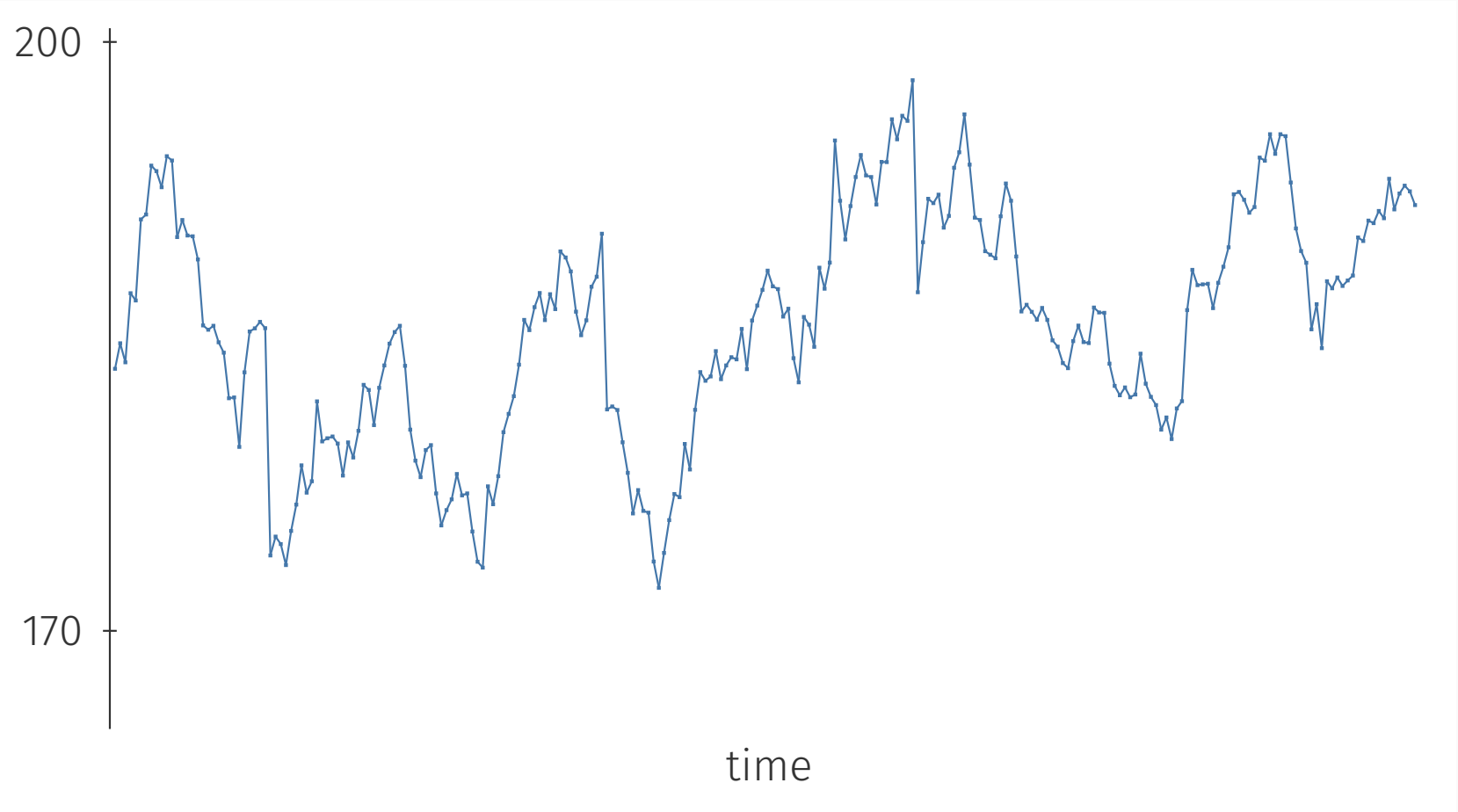
$$\mathcal{S}_{X''} = \langle x''_1, x''_2, \ldots, x''_{n-2} \rangle, \text{ where } x''_i = x'_i - x'_{i-1}$$
$$= x_i - 2x_{i-1} + x_{i-2}$$

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

Example

IBM stock prices from Sept. 2013 to Sept. 2014

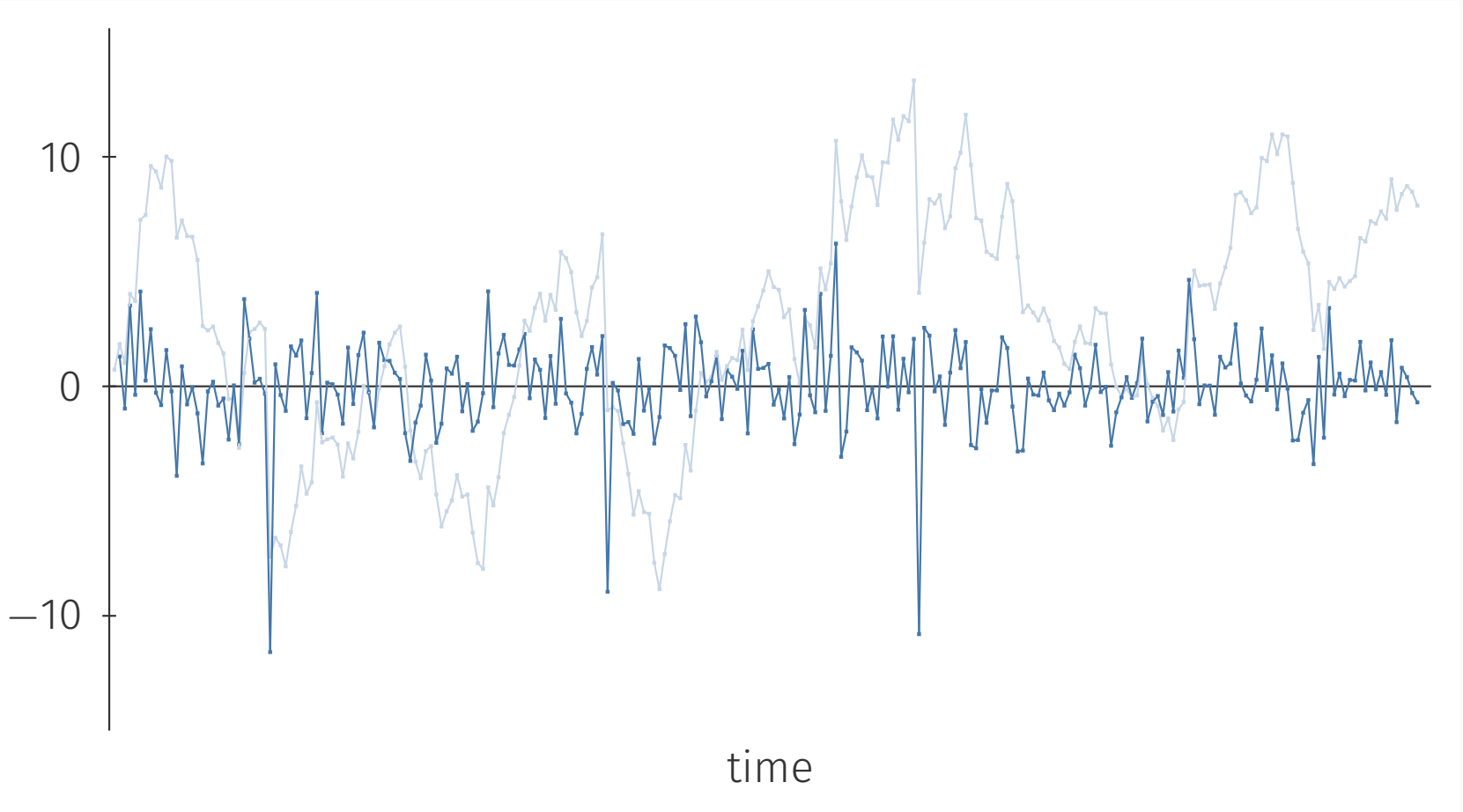
Original time-series



Example

IBM stock prices from Sept. 2013 to Sept. 2014

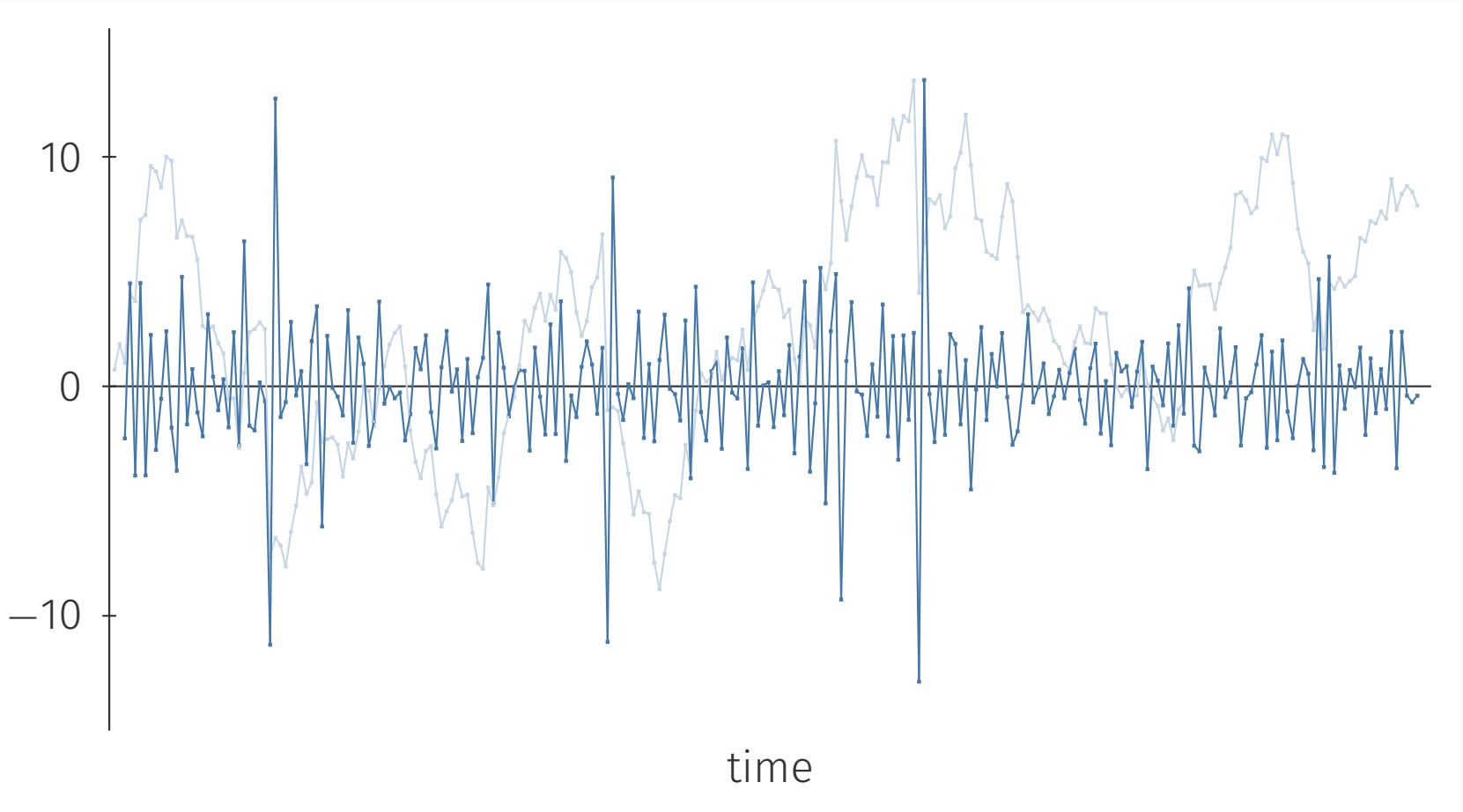
First-order differenced time-series



Example

IBM stock prices from Sept. 2013 to Sept. 2014

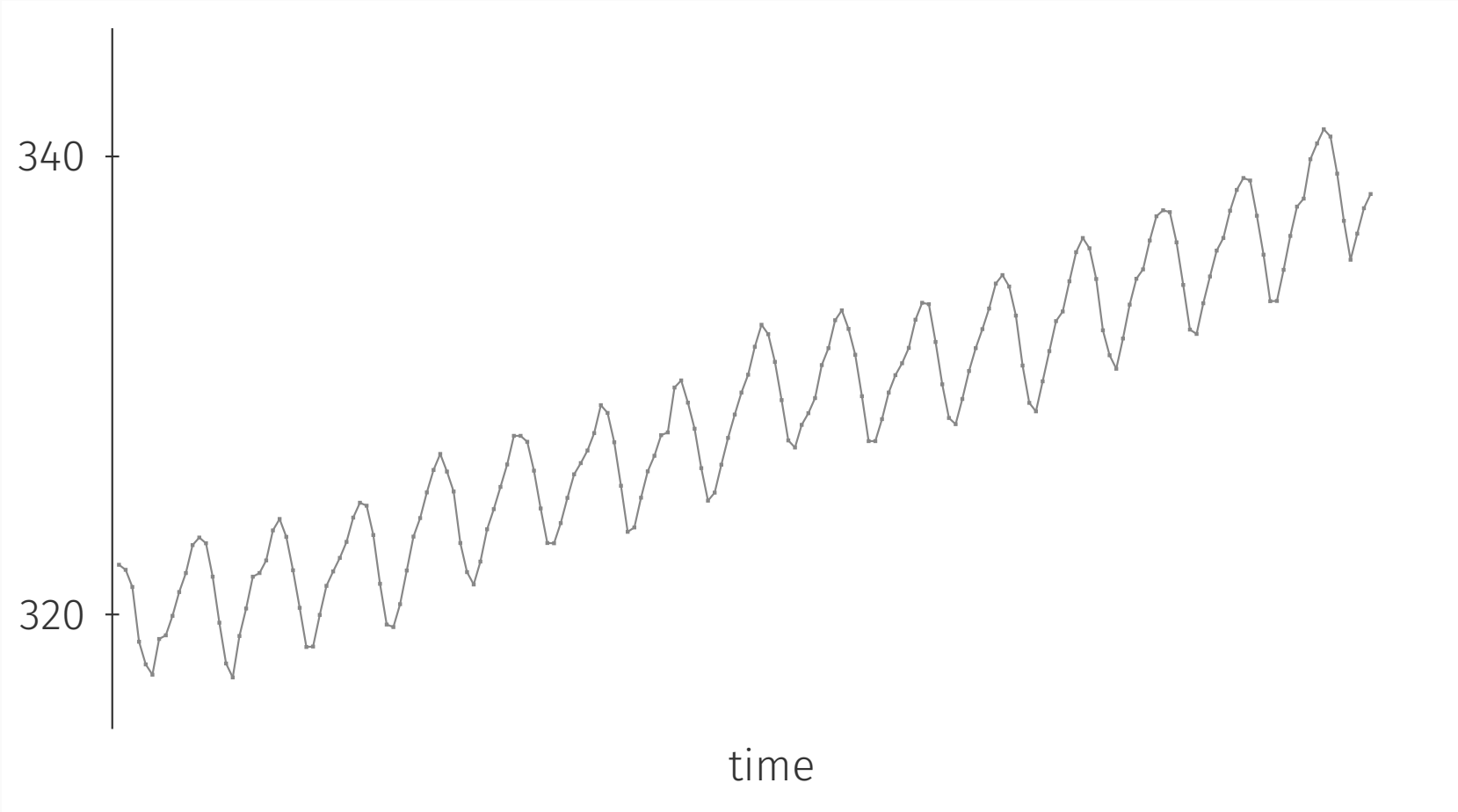
Second-order differenced time-series



Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980

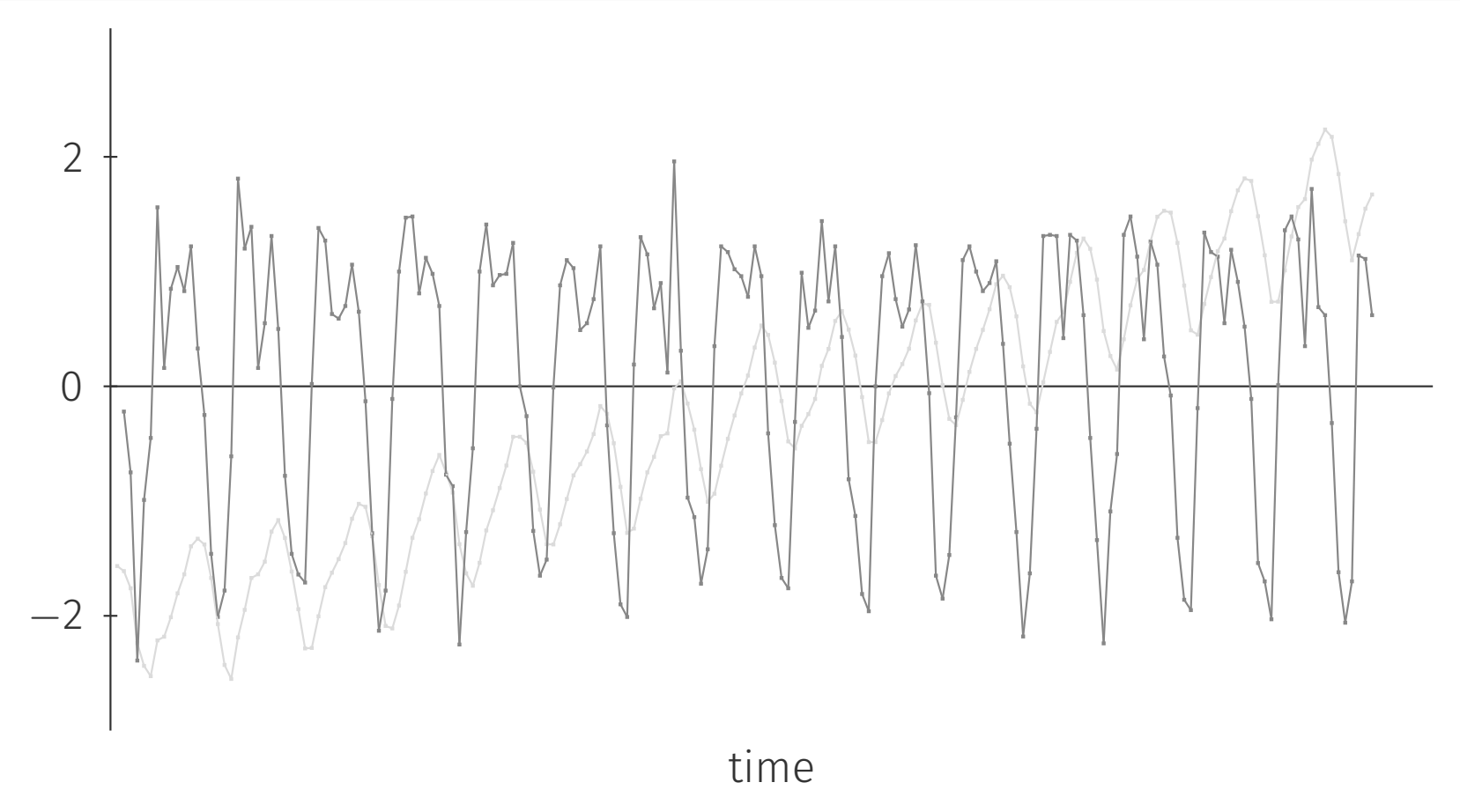
Original time-series



Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980

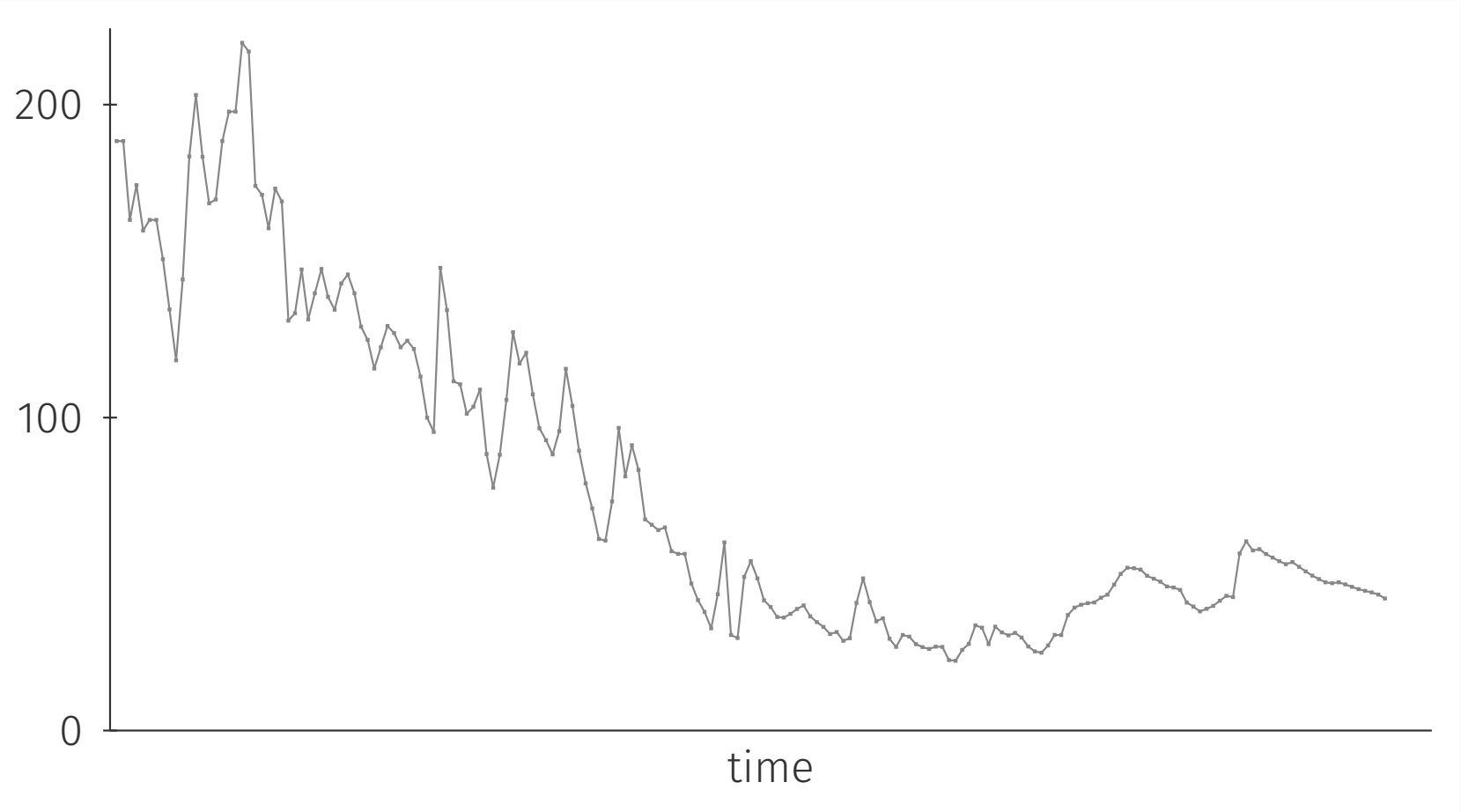
First-order differenced time-series



Example

Nail prices from 1800 to 1996

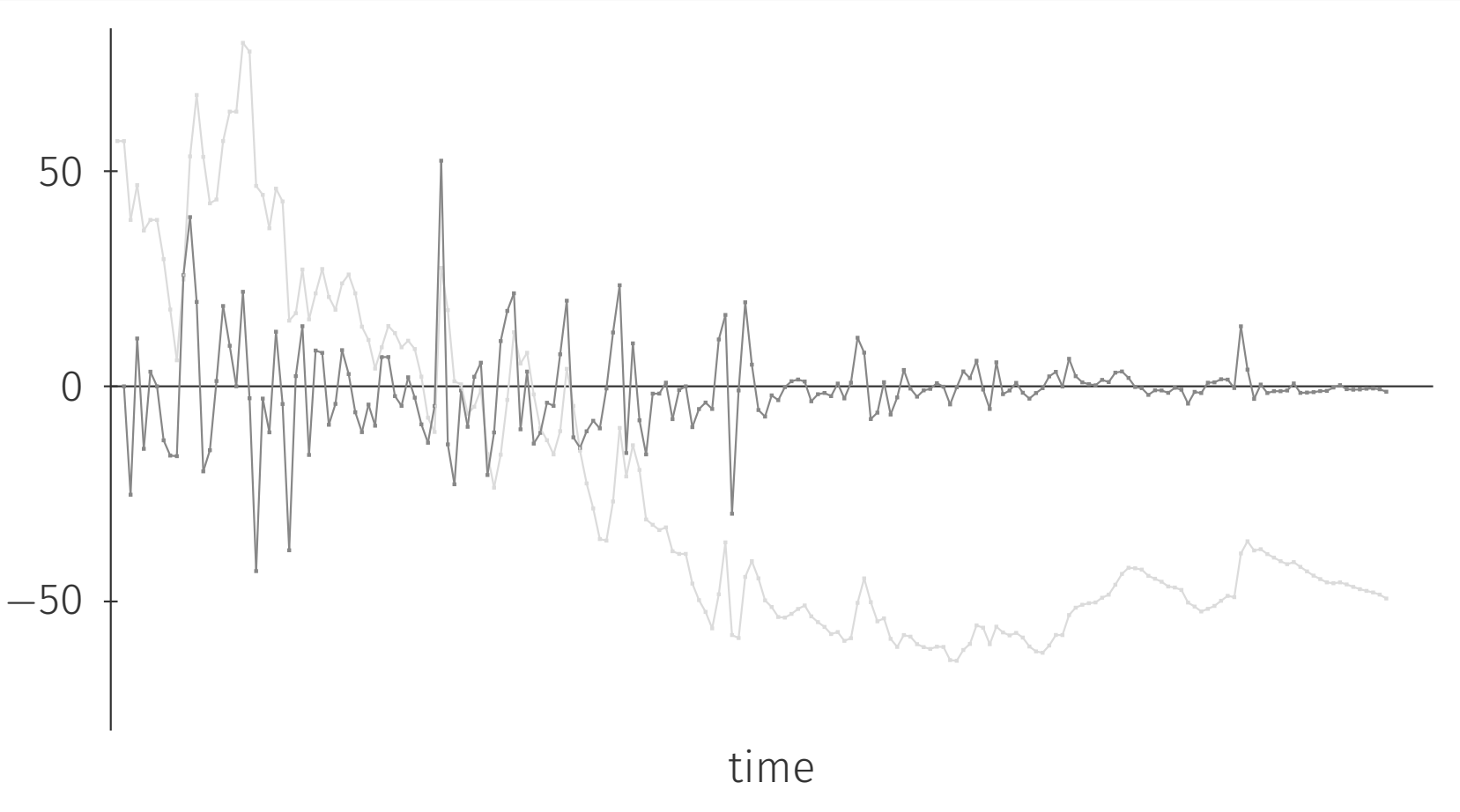
Original time-series



Example

Nail prices from 1800 to 1996

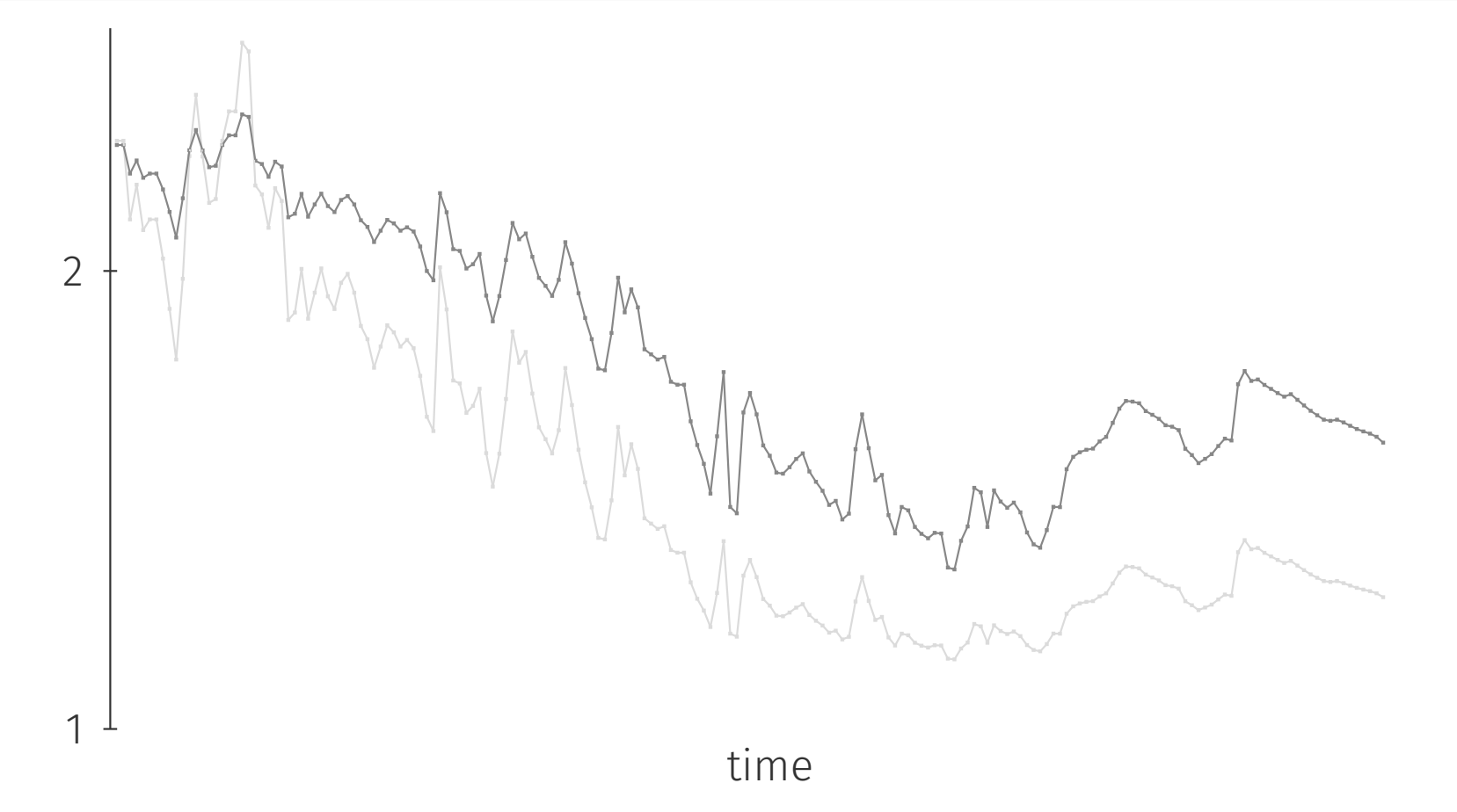
First-order differenced time-series



Example

Nail prices from 1800 to 1996

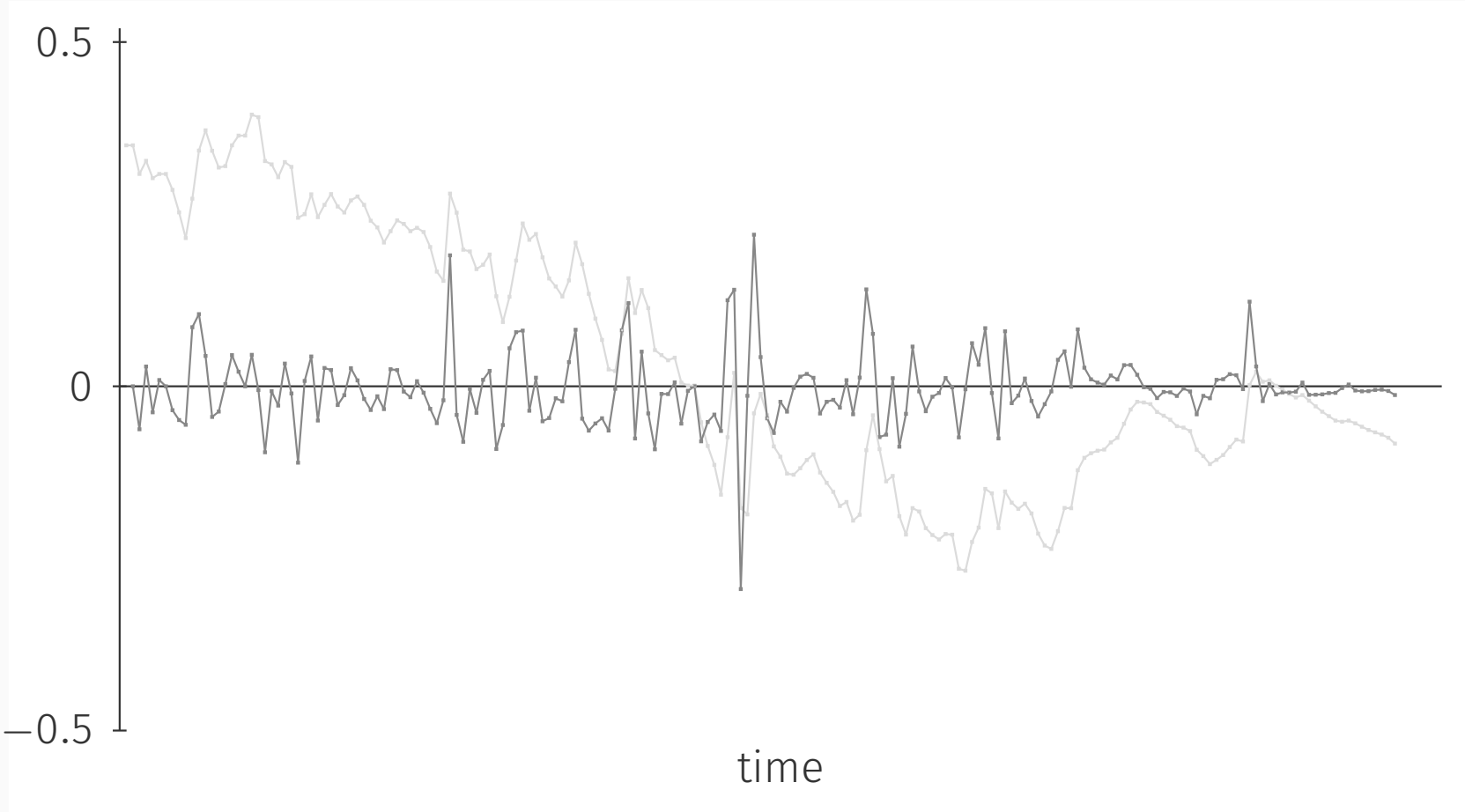
Logarithmic time-series



Example

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

$$\mathrm{cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$$

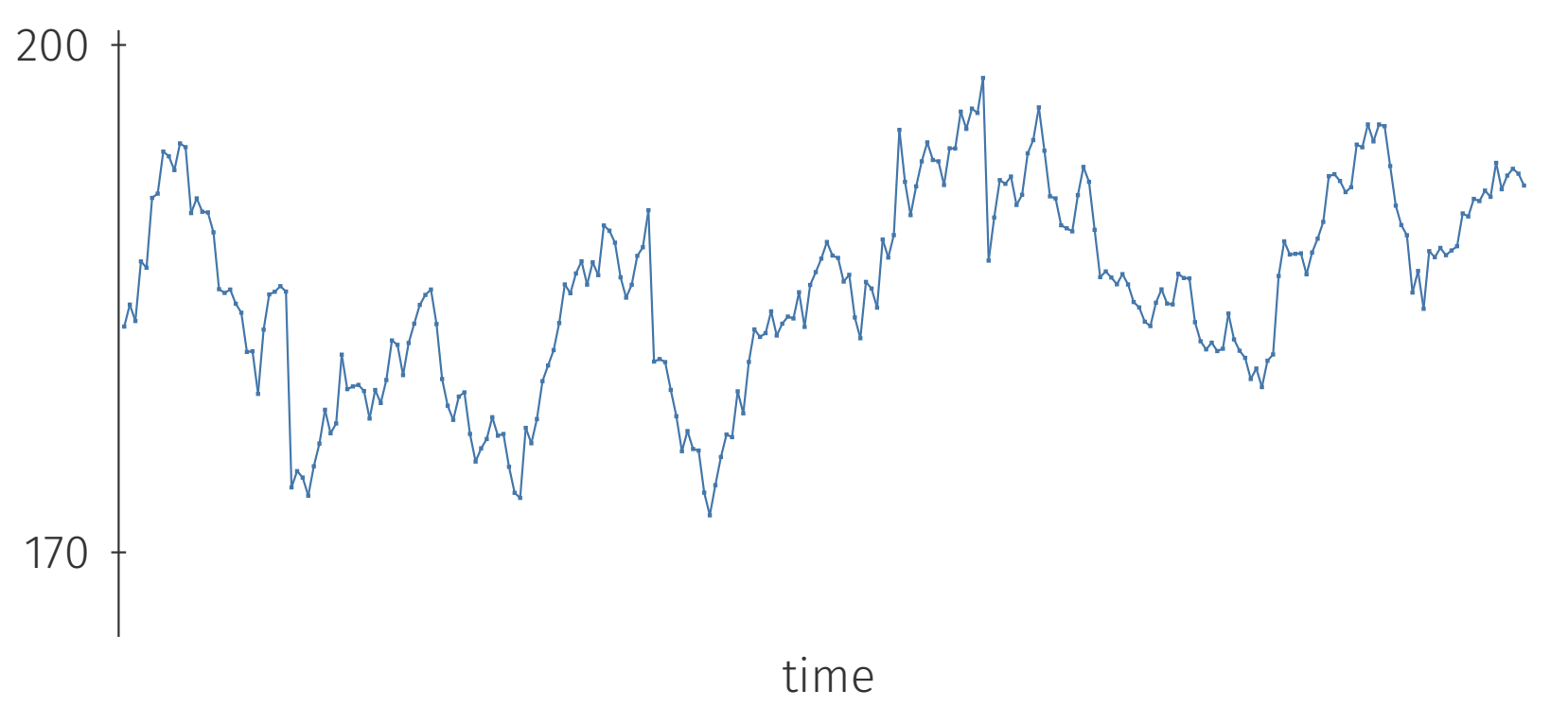
The **autocovariance** at lag τ of time-series $X = x_1, x_2, \ldots, x_n$ is the covariance between the time-series and itself shifted by τ

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

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

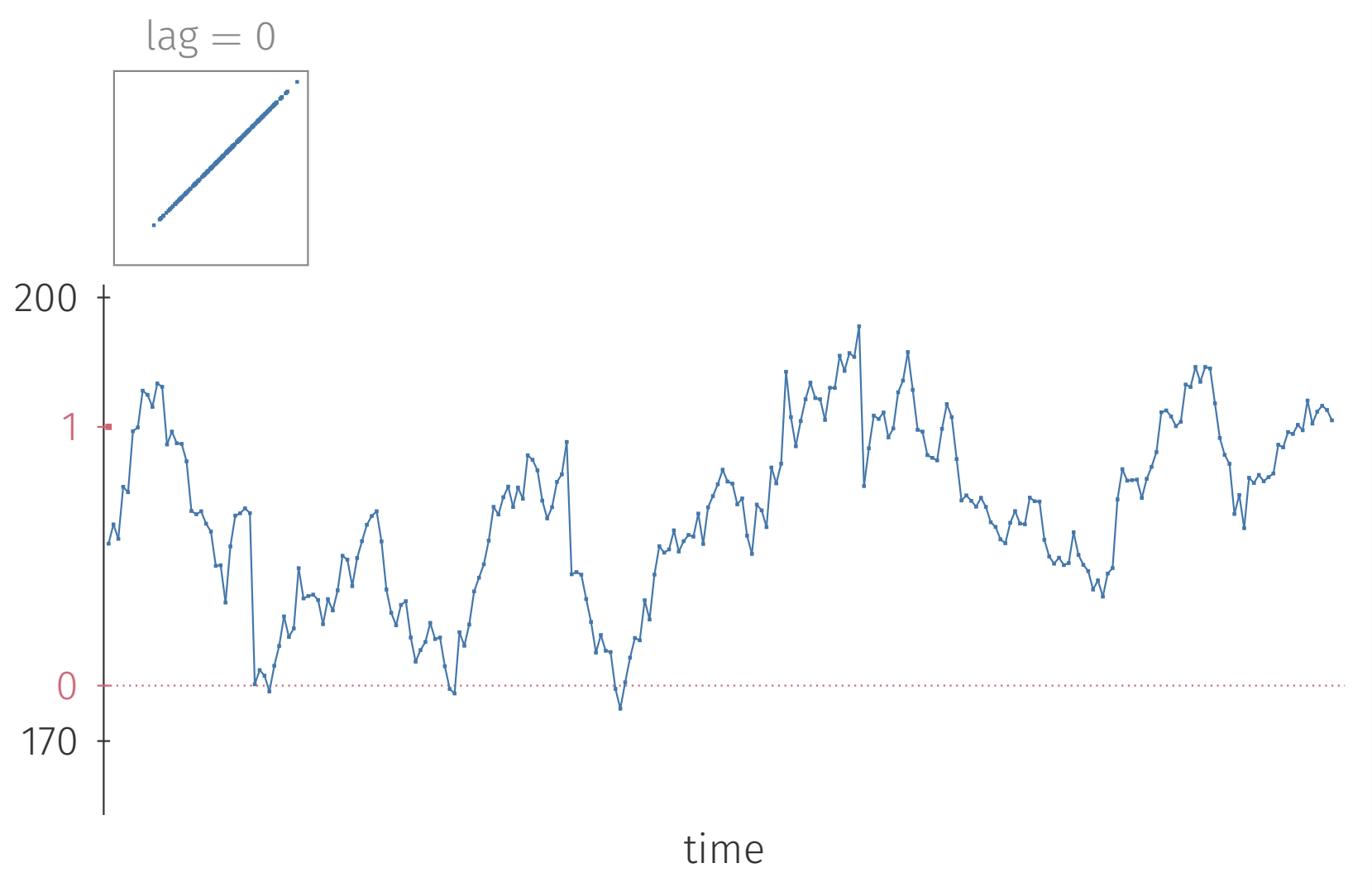
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014



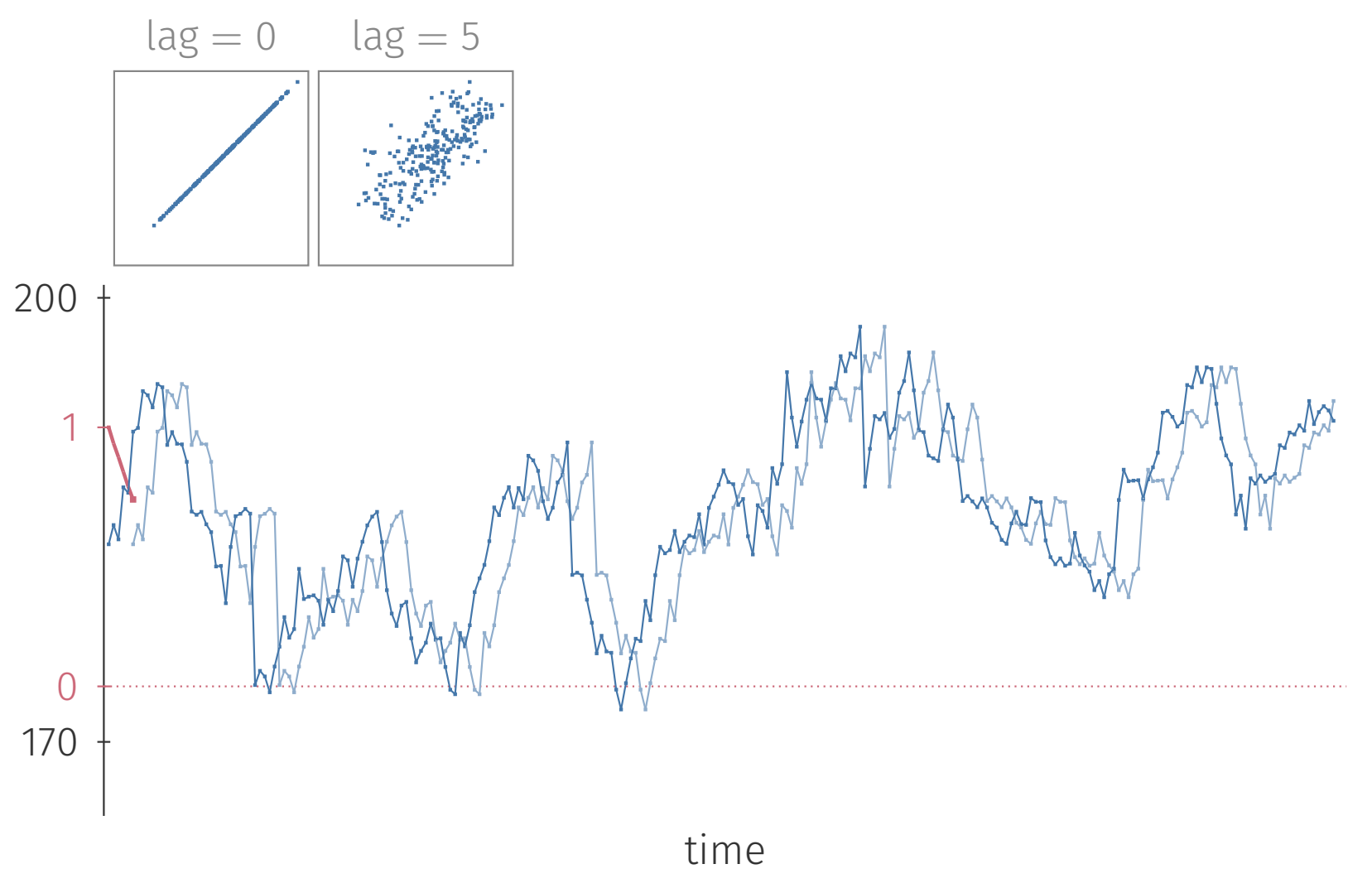
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014



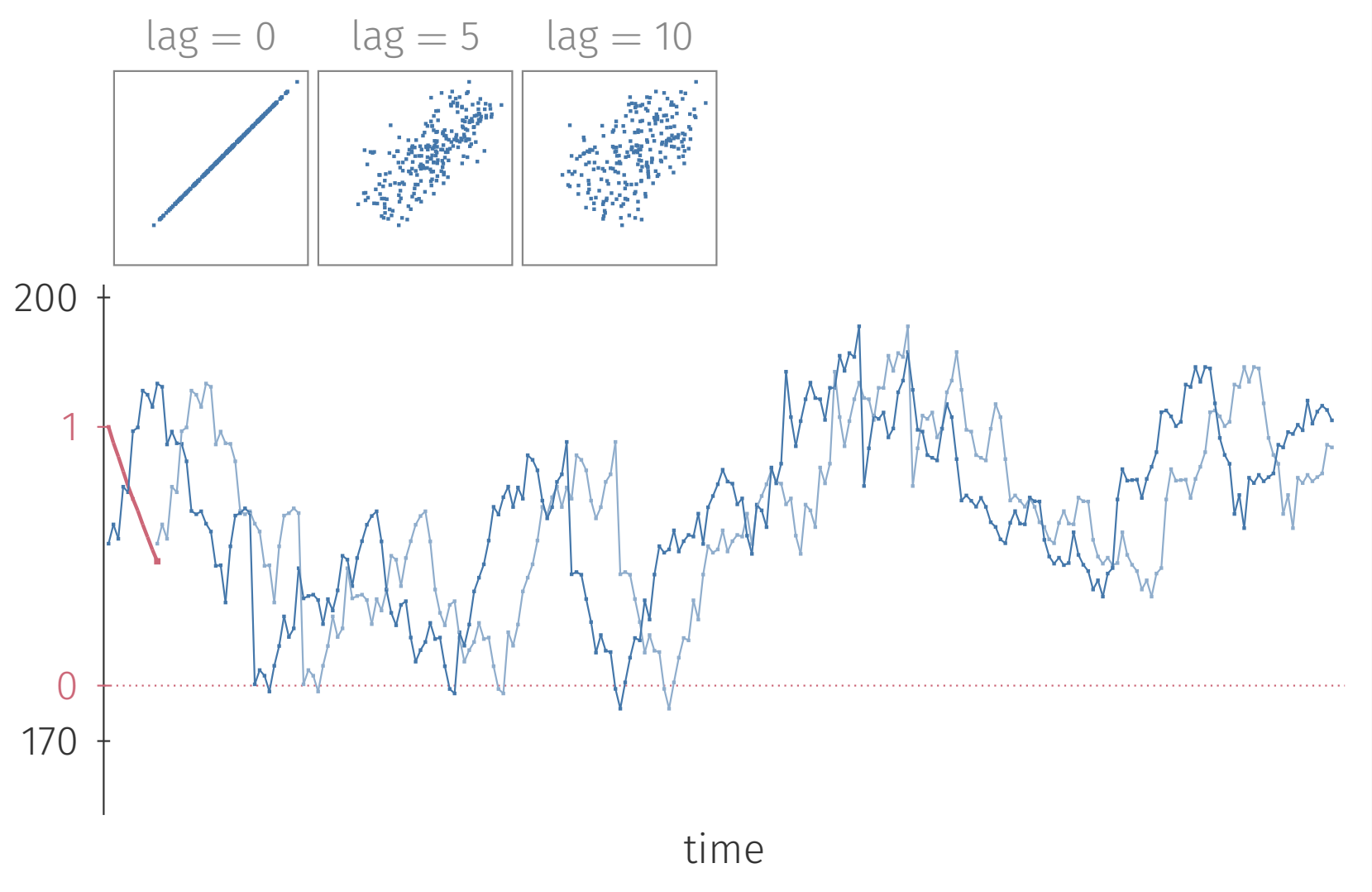
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014



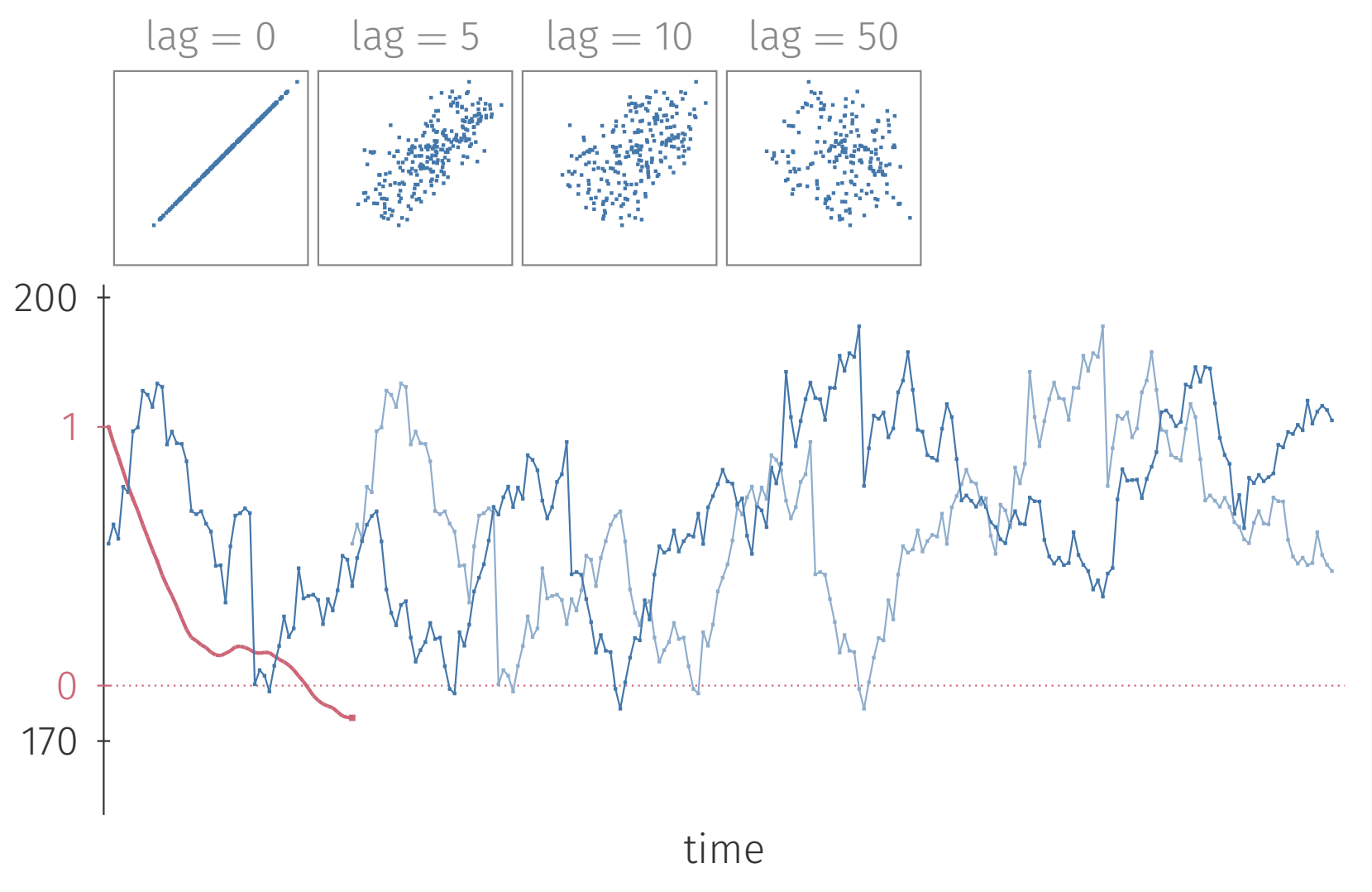
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014



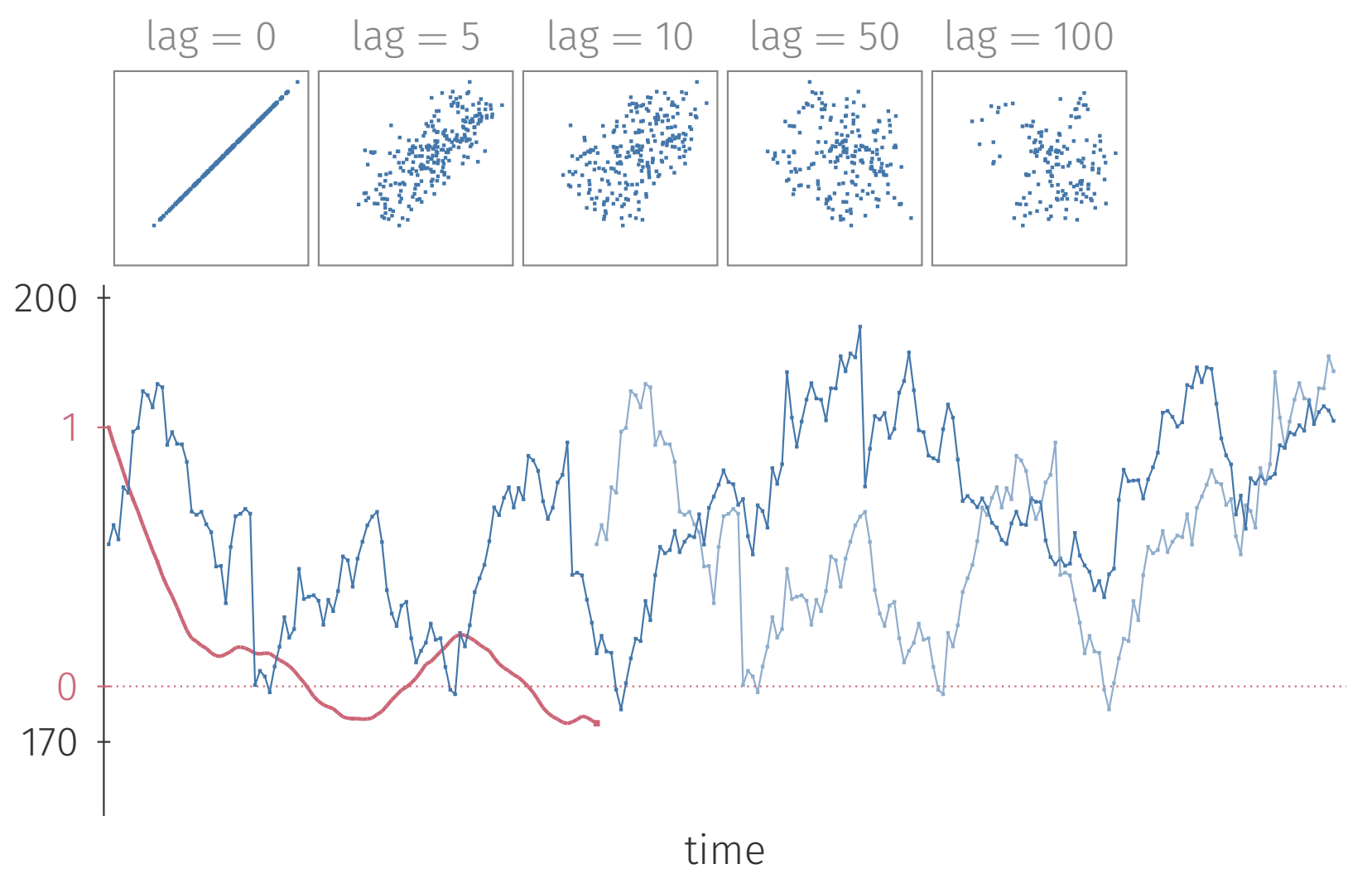
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014



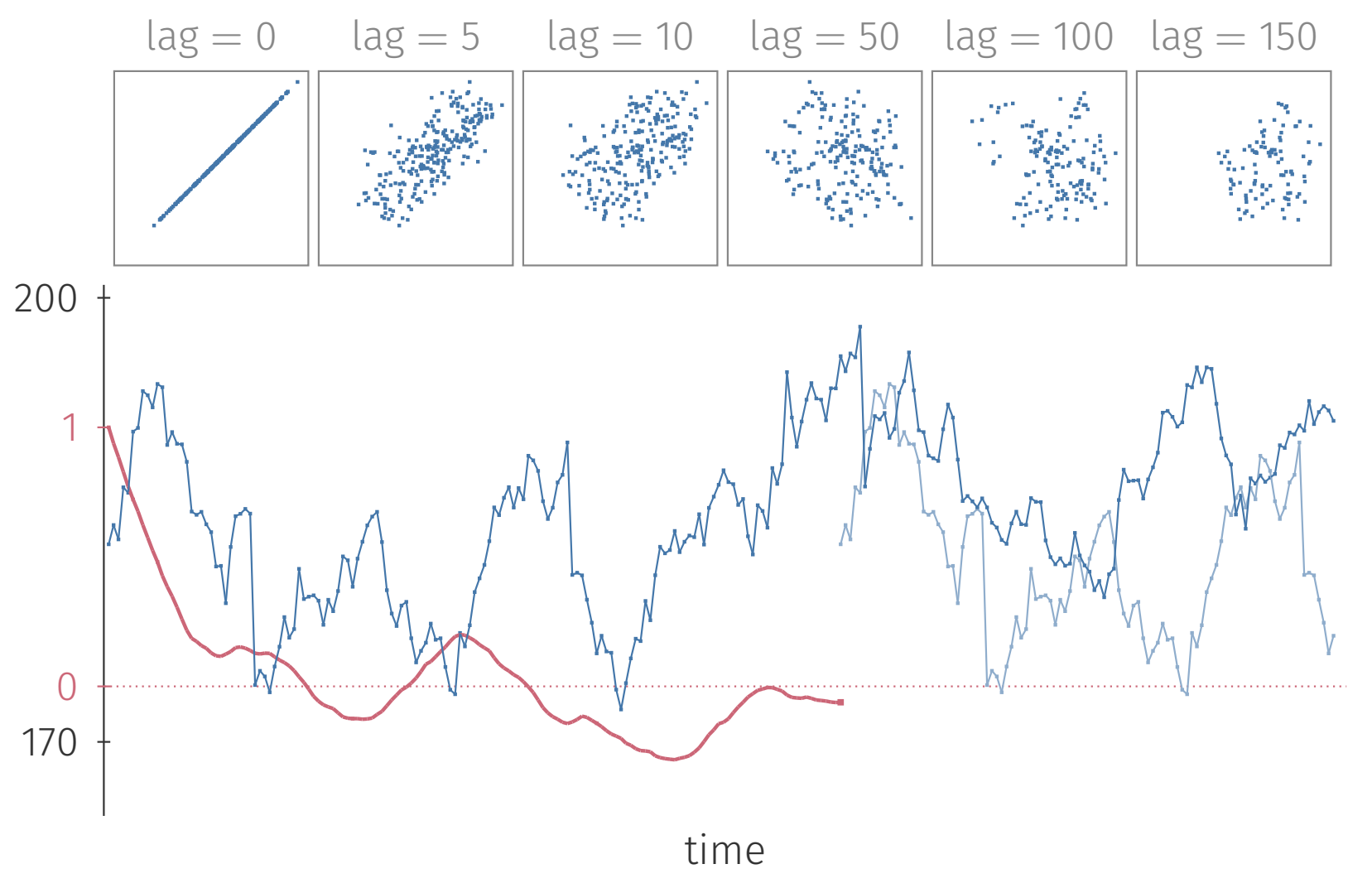
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014



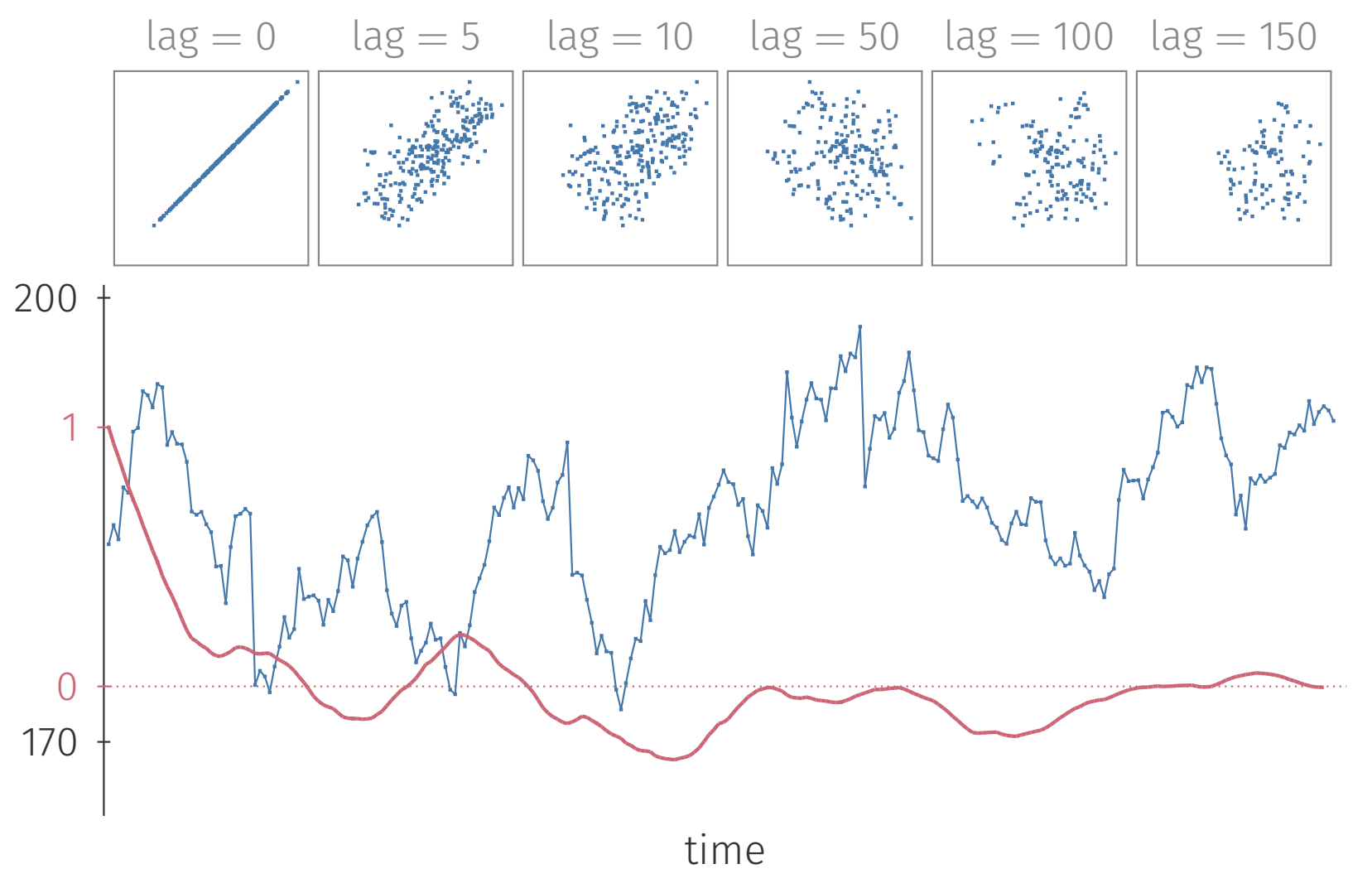
Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014



Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014



Auto-regressive models

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, ϵ_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

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

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*

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

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

Auto-regressive moving-average model 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 τ of time-series X is the normalized covariance

$$\text{ACF}_X(\tau) = \text{R}_\tau(X) = \frac{_t(X_t, X_{t+\tau})}{\text{var}_t(X_t)}$$

The **partial autocorrelation** at lag τ of time-series X, $\text{PACF}_X(\tau)$, is the autocorrelation at lag τ 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

Coefficient of determination

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

$$R^2 = 1 - \frac{\text{mean}(\epsilon_i^2)}{\text{var}(X_i^2)}$$

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

Models for multivariate time-series

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

Models for multivariate time-series

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

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

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

C captures the correlations across variables, not across time

Models for multivariate time-series

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

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

$$C = Q \Lambda Q^T$$

where the m columns of Q contain the orthogonal eigenvectors and the diagonal of Λ 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 $\mathcal{S}_X = \langle \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(n)} \rangle$, with $\boldsymbol{x}^{(i)} \in \mathbb{R}^m$, the aim is to predict $\boldsymbol{x}^{(n+1)}$

3. Compute the p-dimensional hidden time-series $\mathcal{S}_H$

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

The p variables of $\mathcal{S}_H$ are uncorrelated

Models for multivariate time-series

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

4. Build a model for each dimension j of $\mathcal{S}_H$
   Use them to predict the hidden values $h_j^{(n+1)}$

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

Predict $h_j^{(n+1)}$ using the model built for hidden variable j and corresponding hidden values $h_j^{(n)}, h_j^{(n-1)}, \ldots$

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

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

Return $\boldsymbol{x}^{(n+1)} = \boldsymbol{h}^{(n+1)} P^T$ as the predicted vector

Models for multivariate time-series

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

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 $\mathcal{S}_H$
4. Build a model for each dimension j of $\mathcal{S}_H$
   Use them to predict the hidden values $h_j^{(n+1)}$
5. Transform $\boldsymbol{h}^{(n+1)}$ back to the original m-dimensional representation, the prediction $\boldsymbol{x}^{(n+1)}$ 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

Periodicity

Time-series might exhibit regularly recurrent, cyclic, behavior i.e. display **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 $\mathcal{S}_X = \langle x_0, x_1, \ldots, x_{n-1} \rangle$

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

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

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

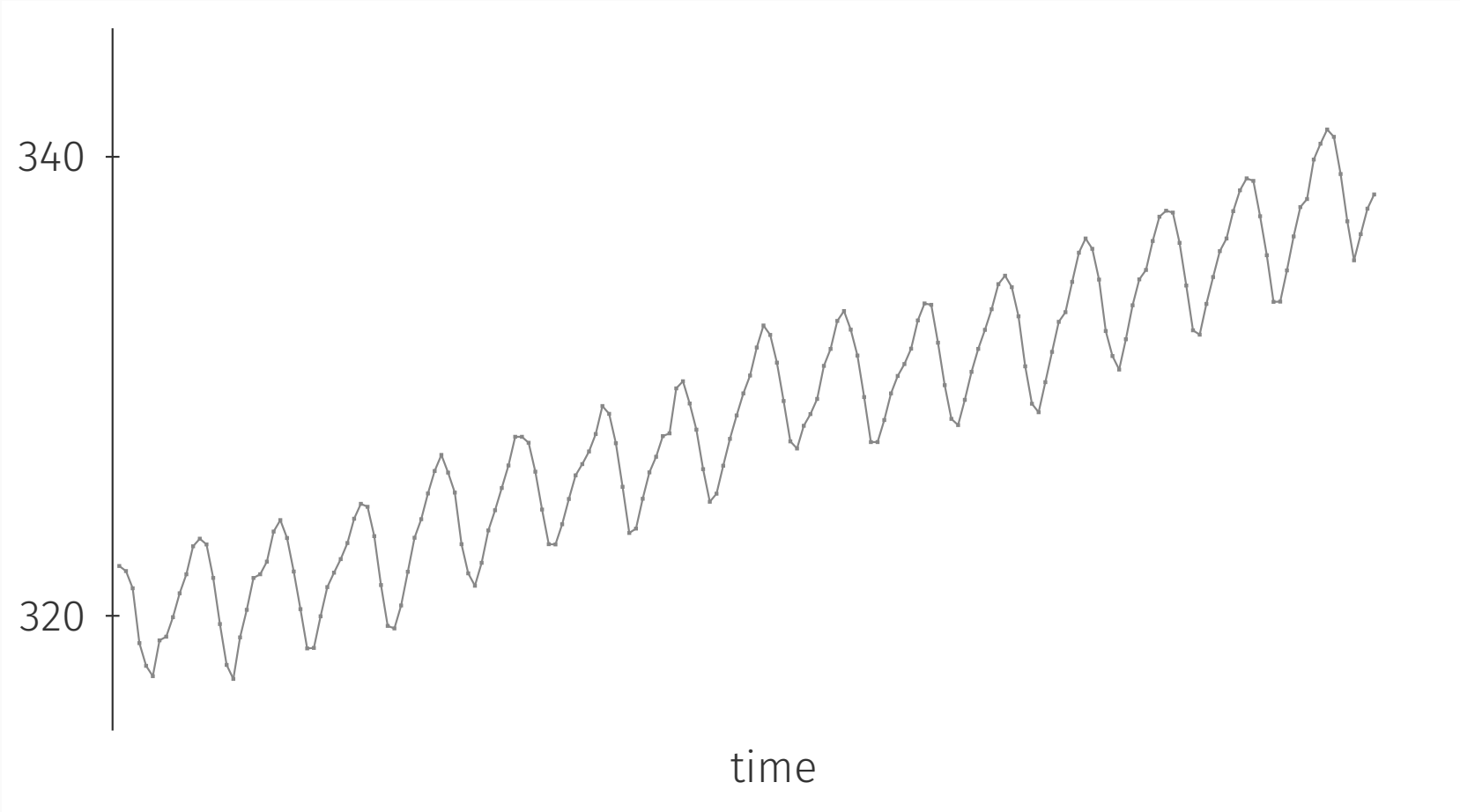
If a component has a high amplitude compared to the others, the entire series will be dominated by its periodic behavior

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

Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980

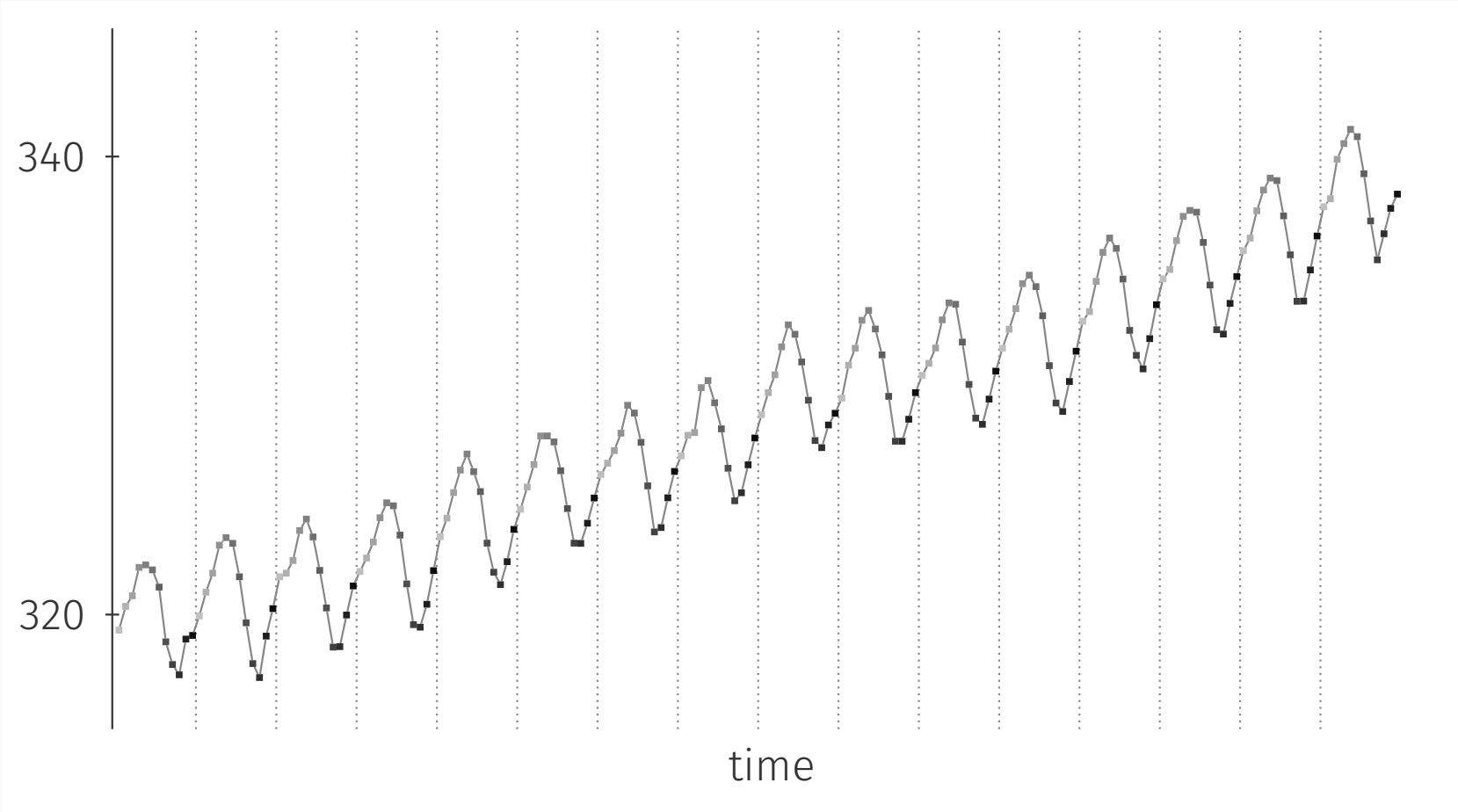
Original time-series



Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980

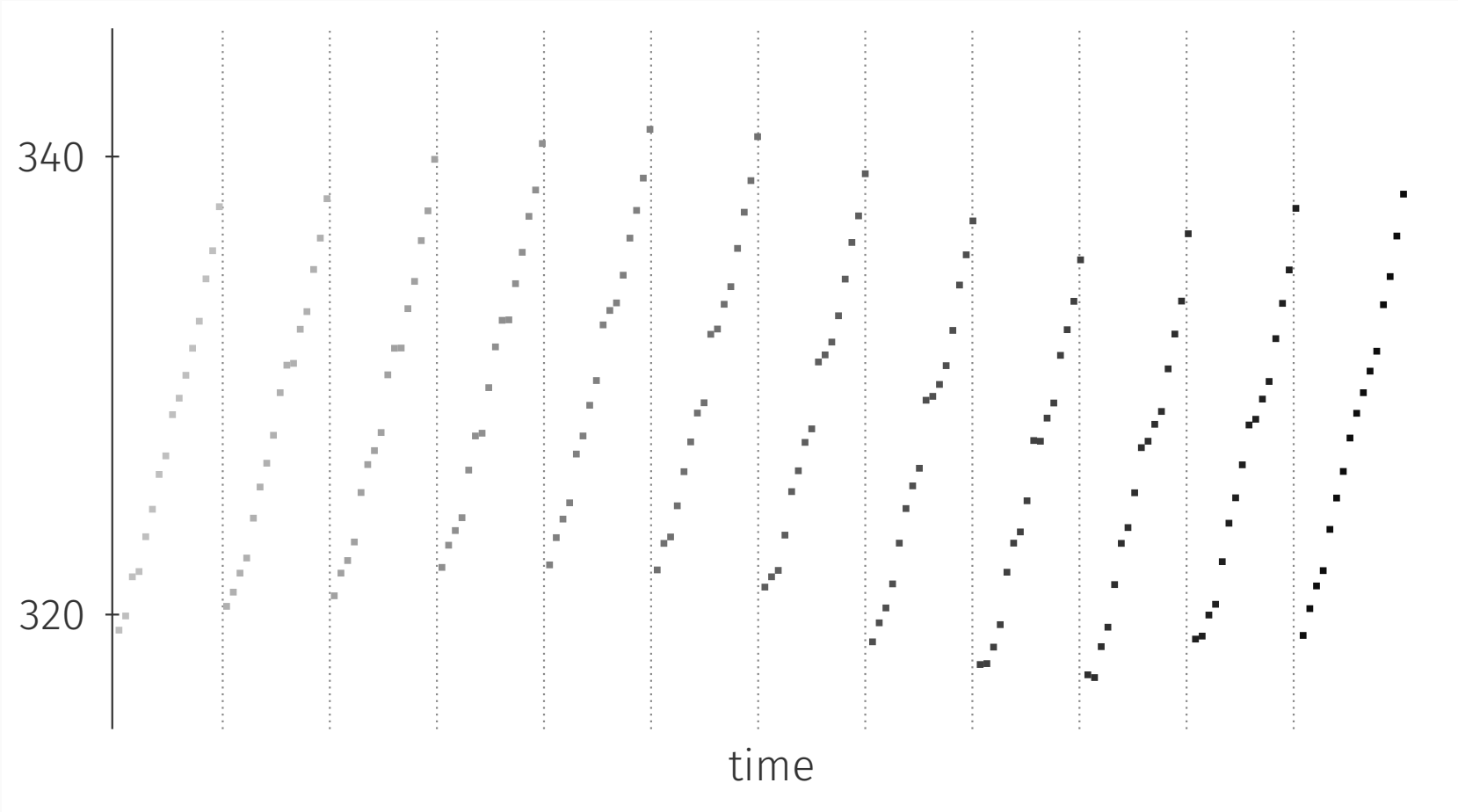
Folding the time-series, $p = 12$



Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980

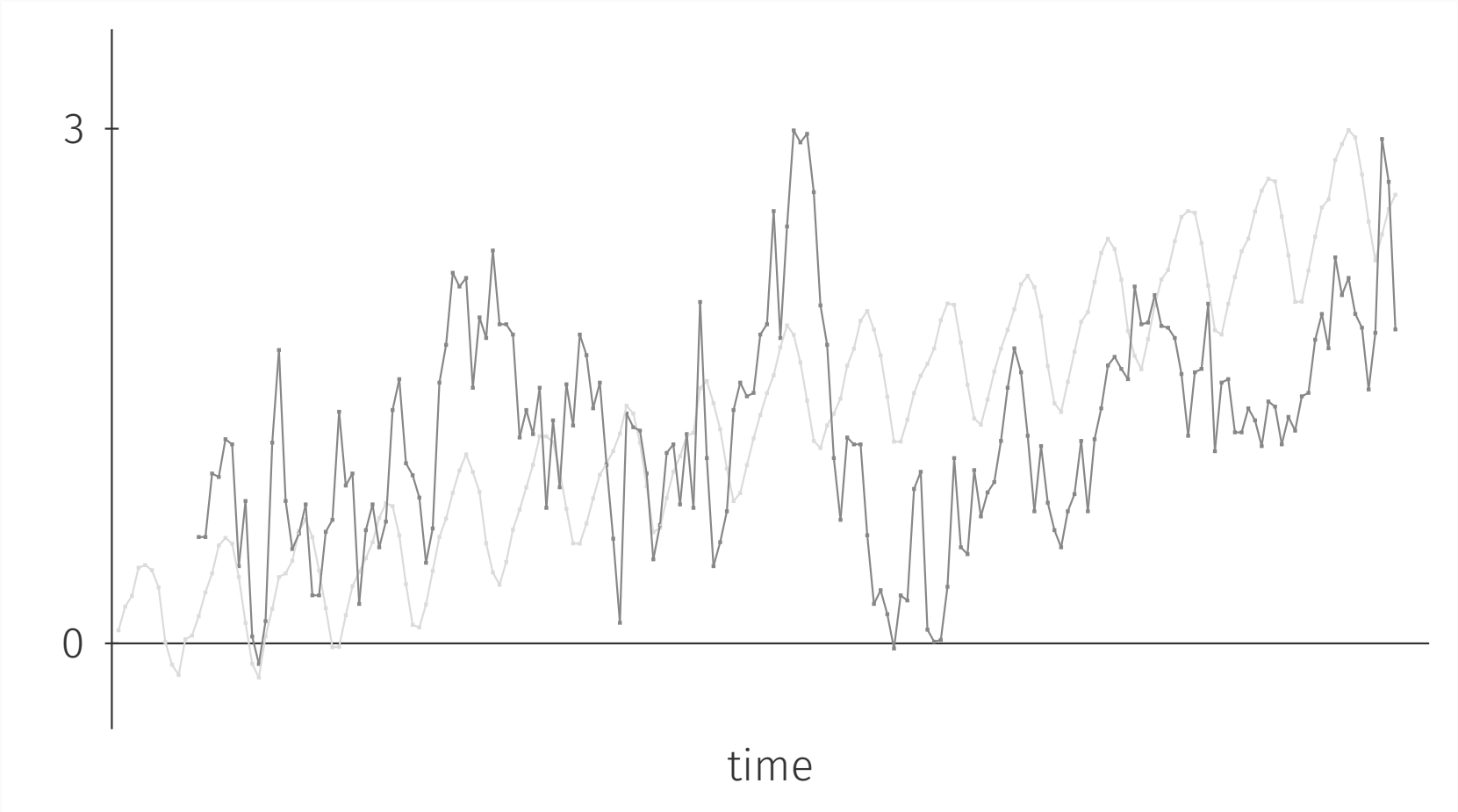
Folding the time-series, $p = 12$



Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980

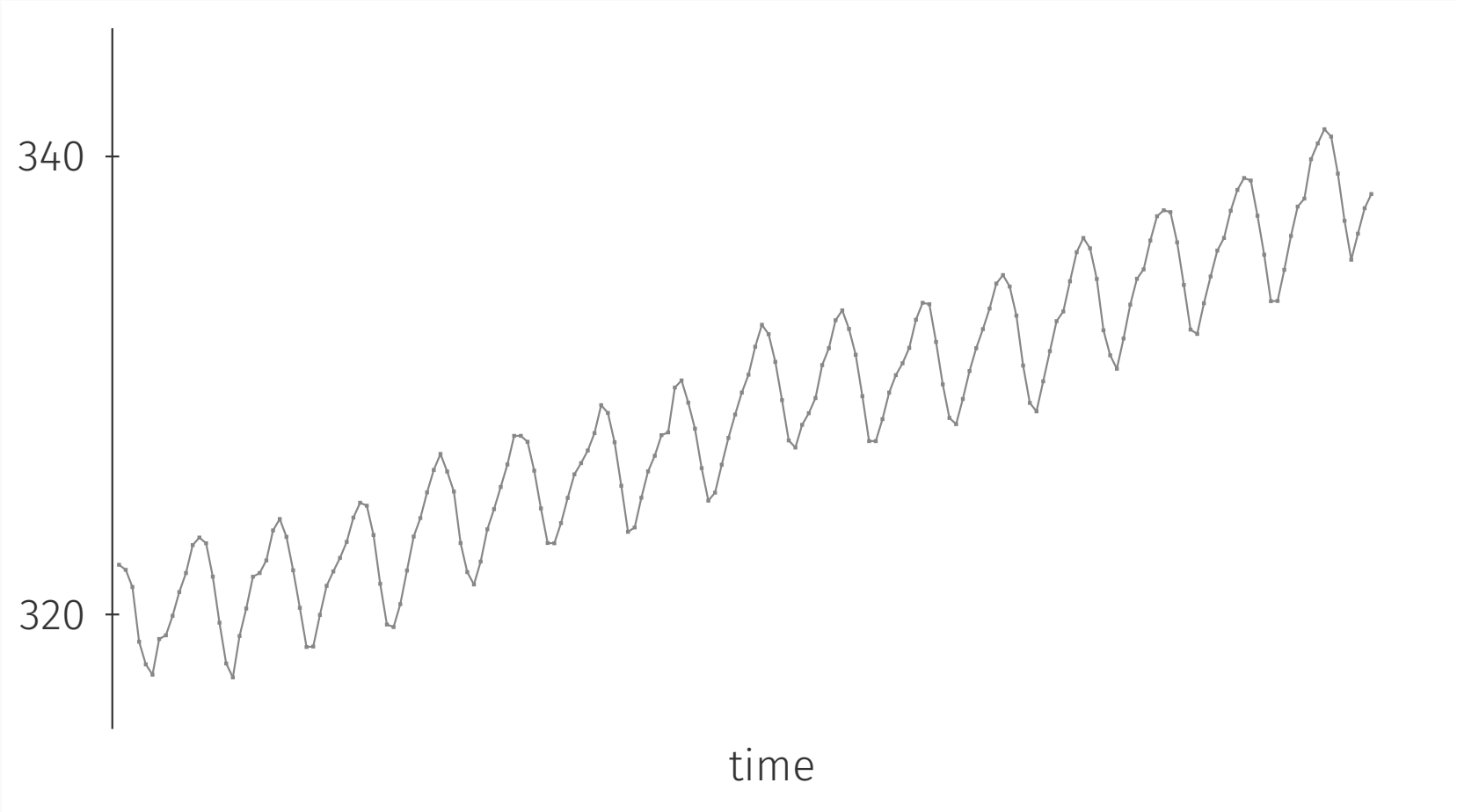
Seasonal differenced time-series, $p = 12$



Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980

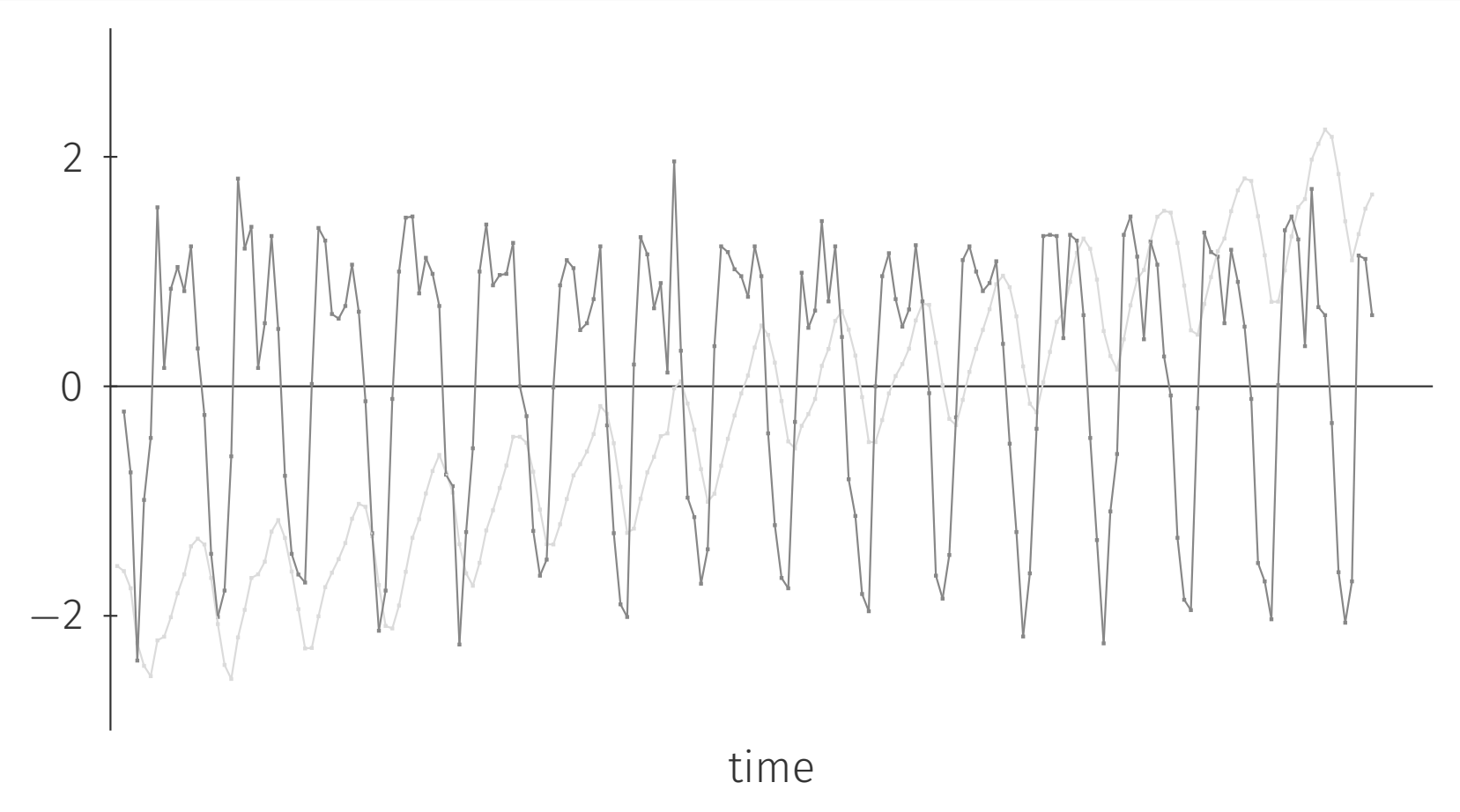
Original time-series



Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980

First-order differenced time-series

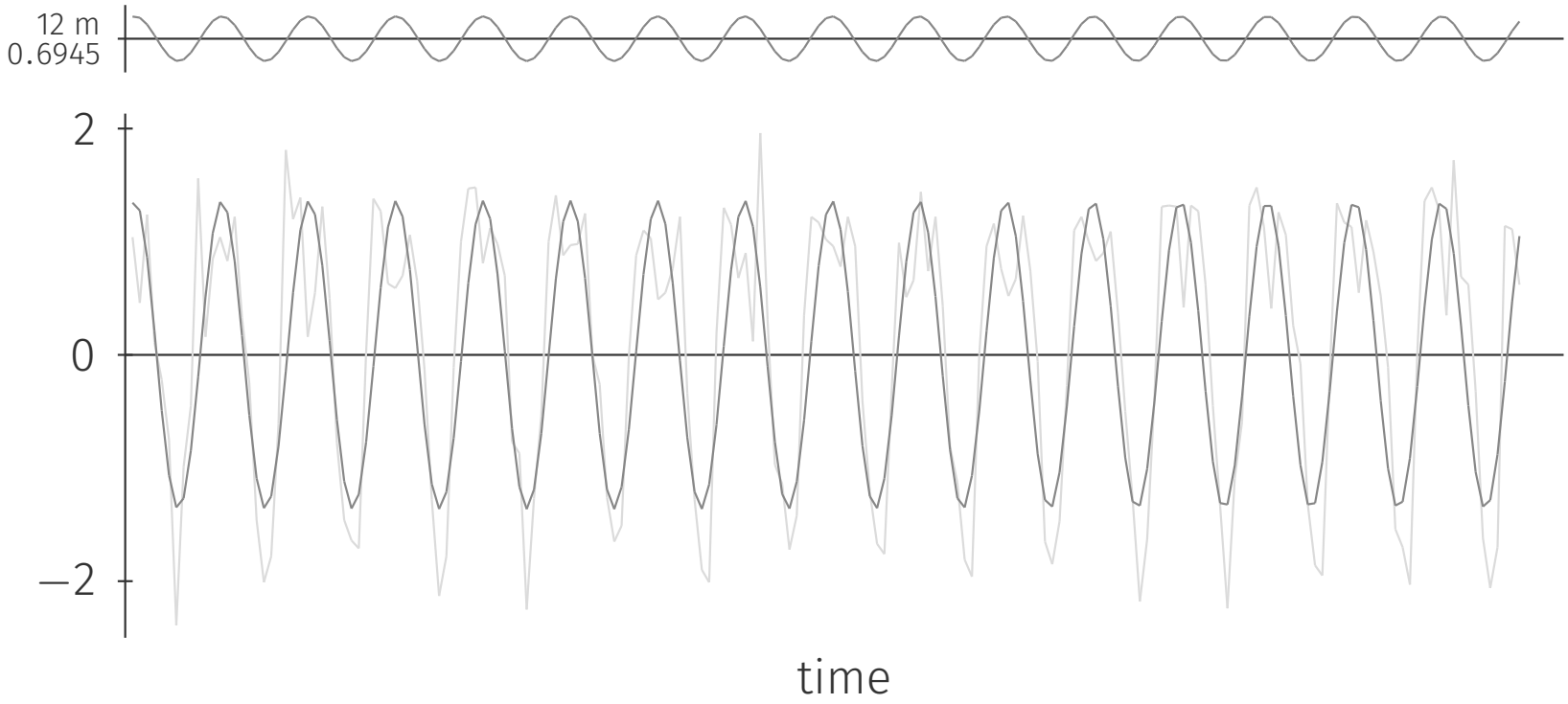


Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980

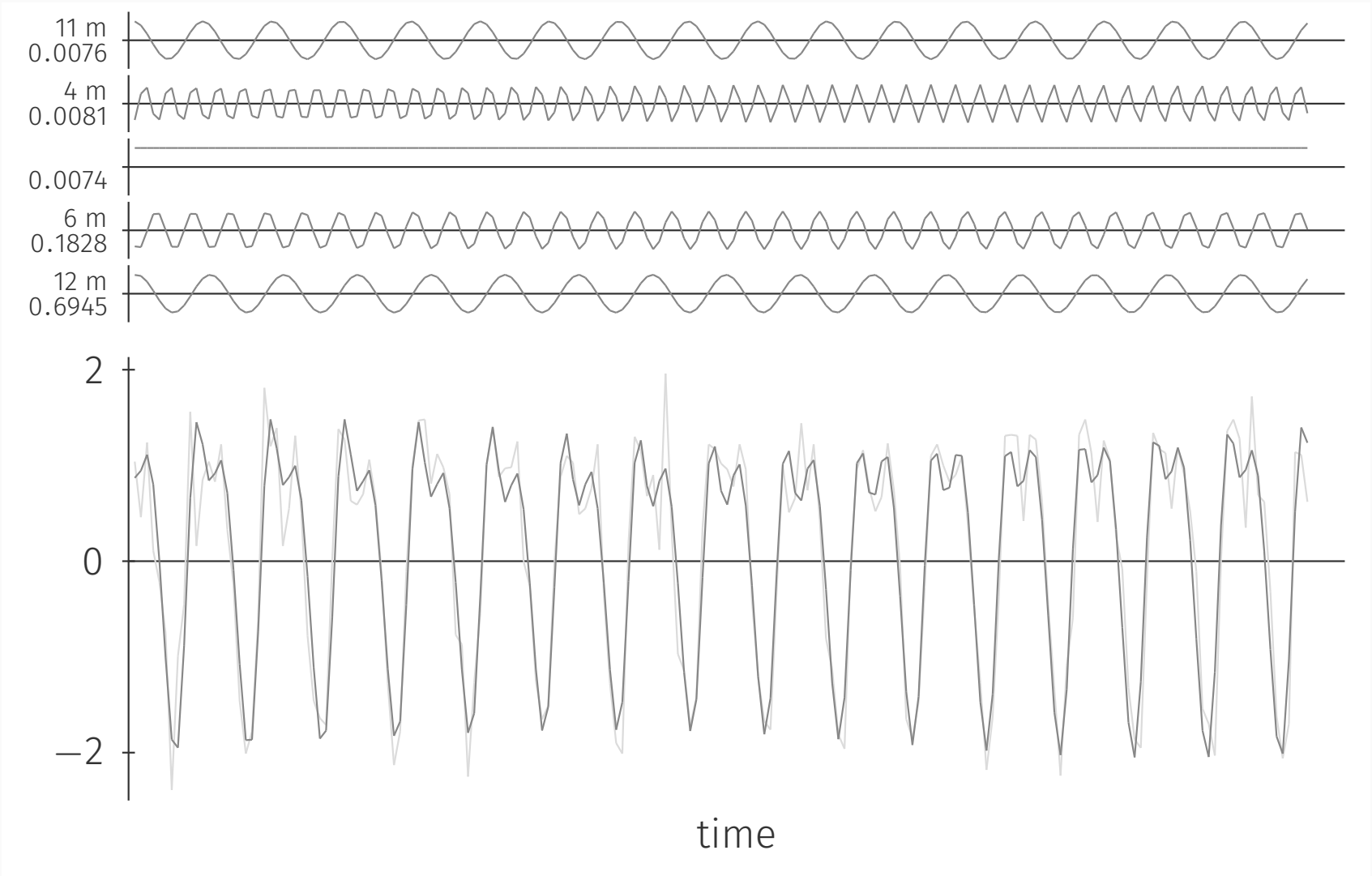
Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980



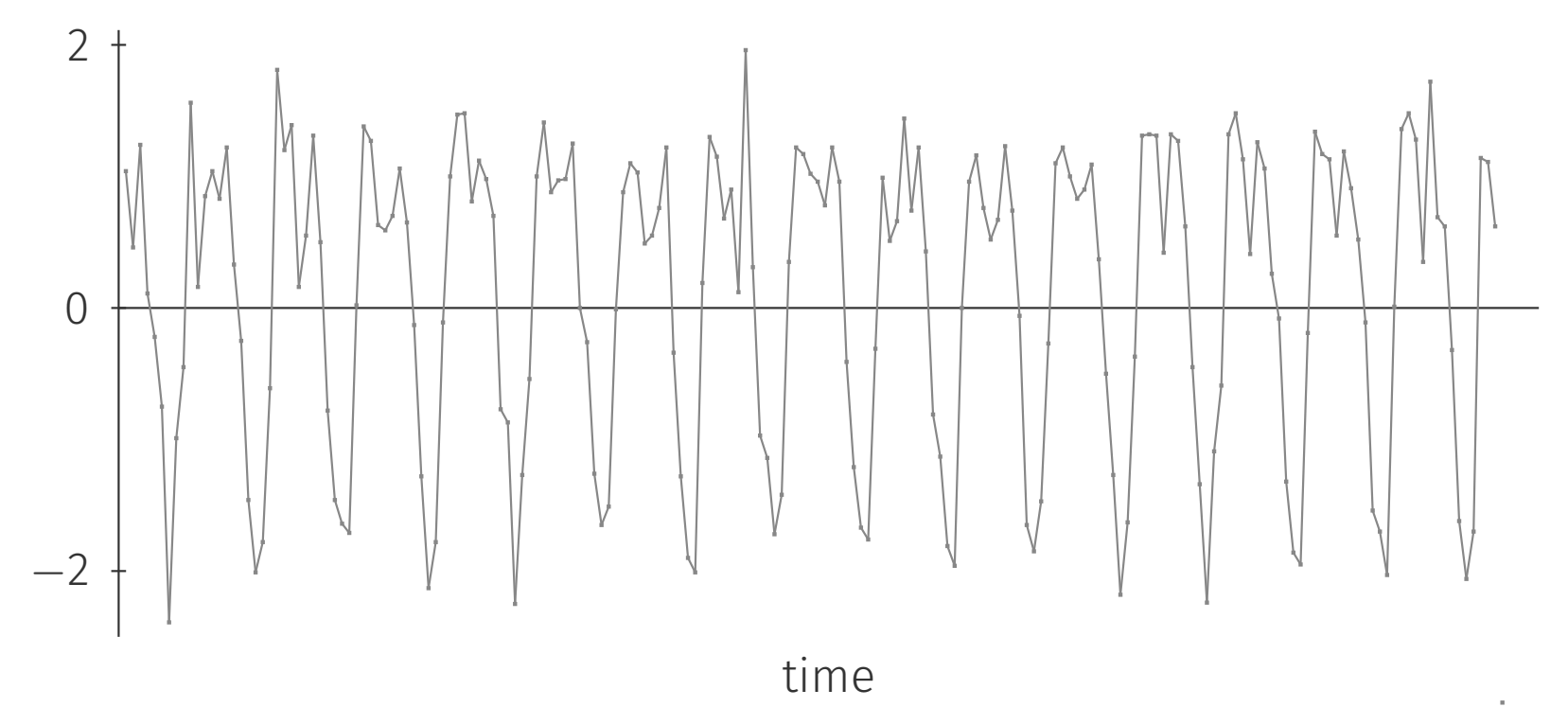
Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980



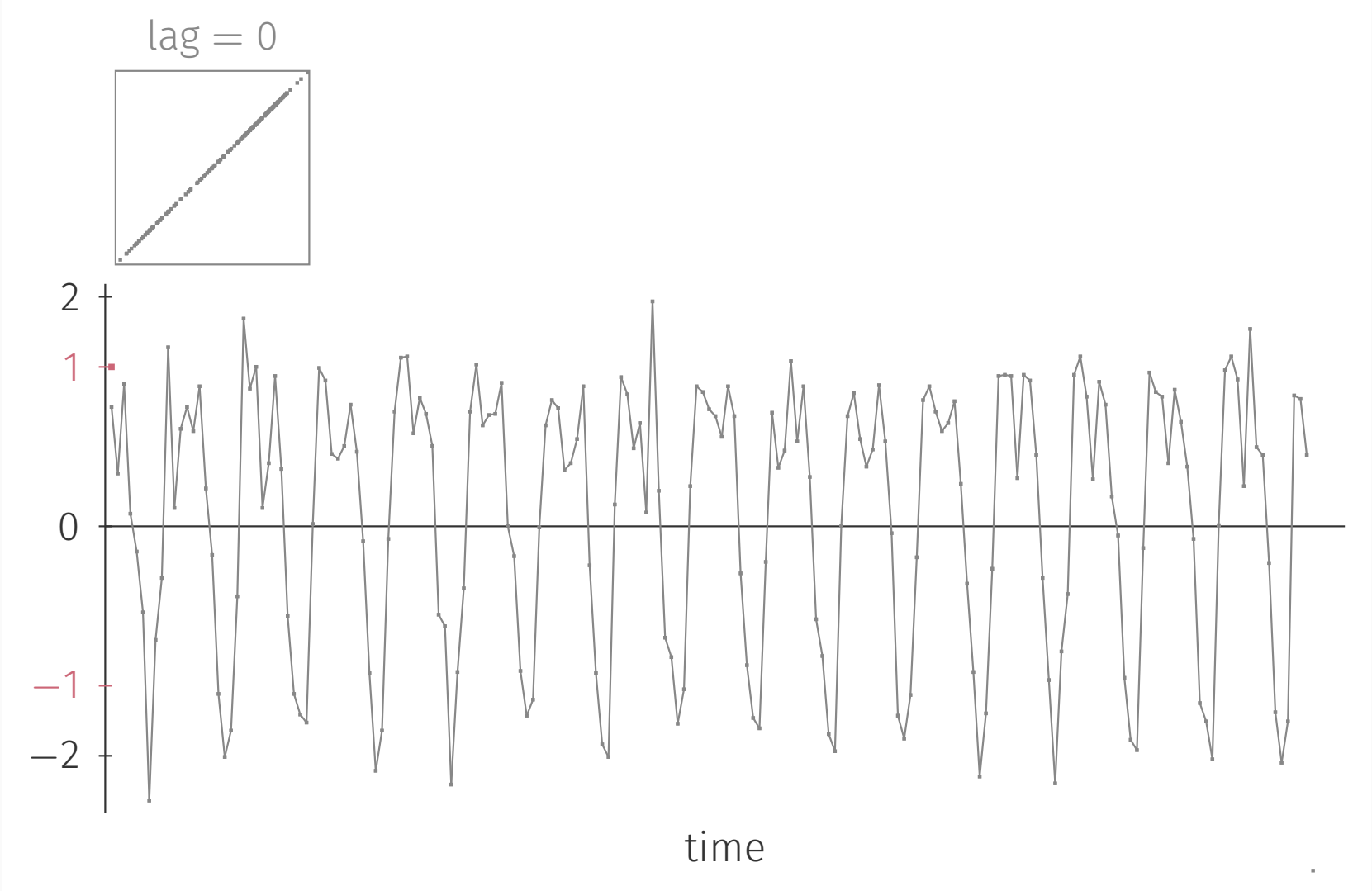
Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980



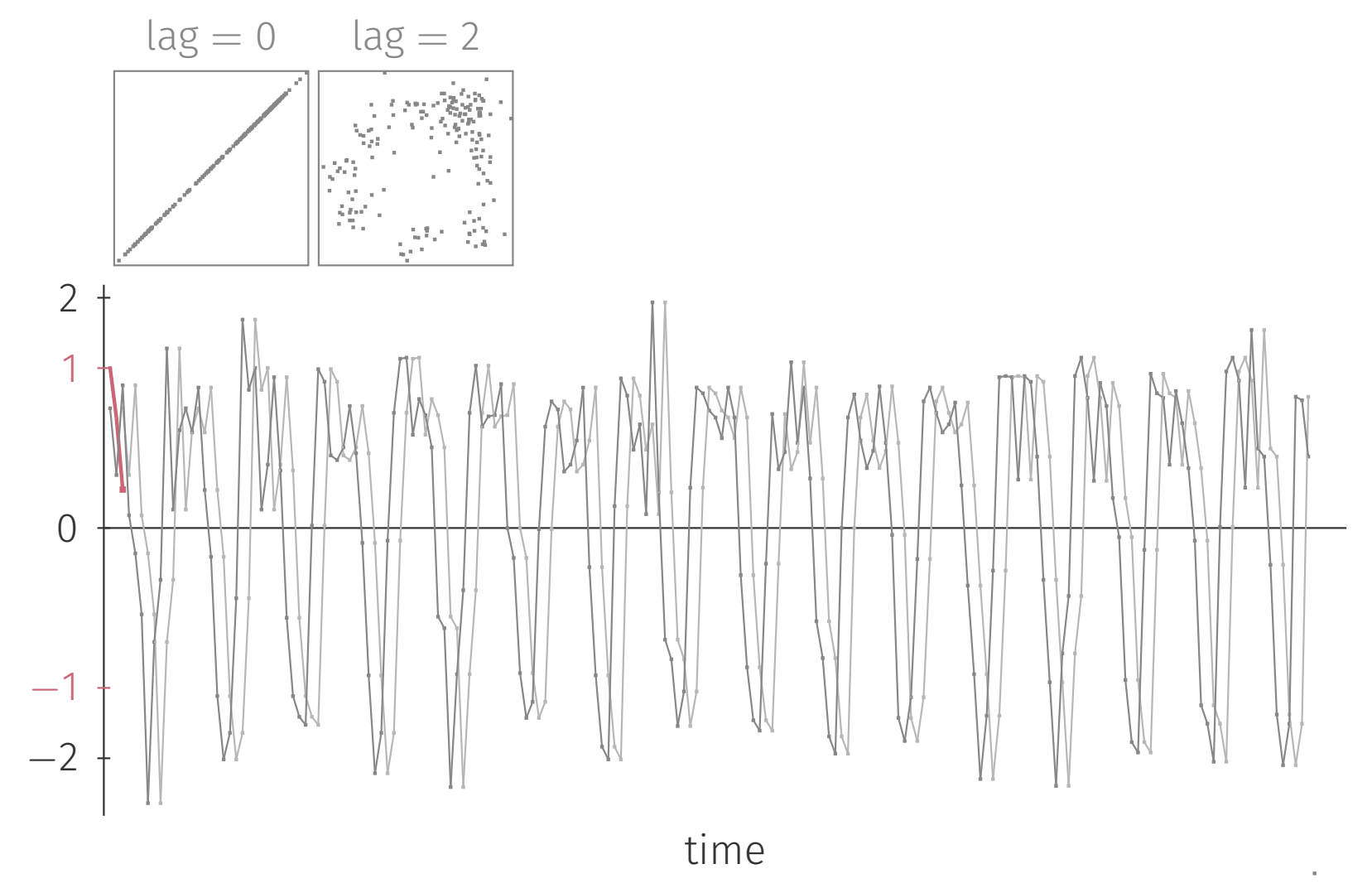
Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980



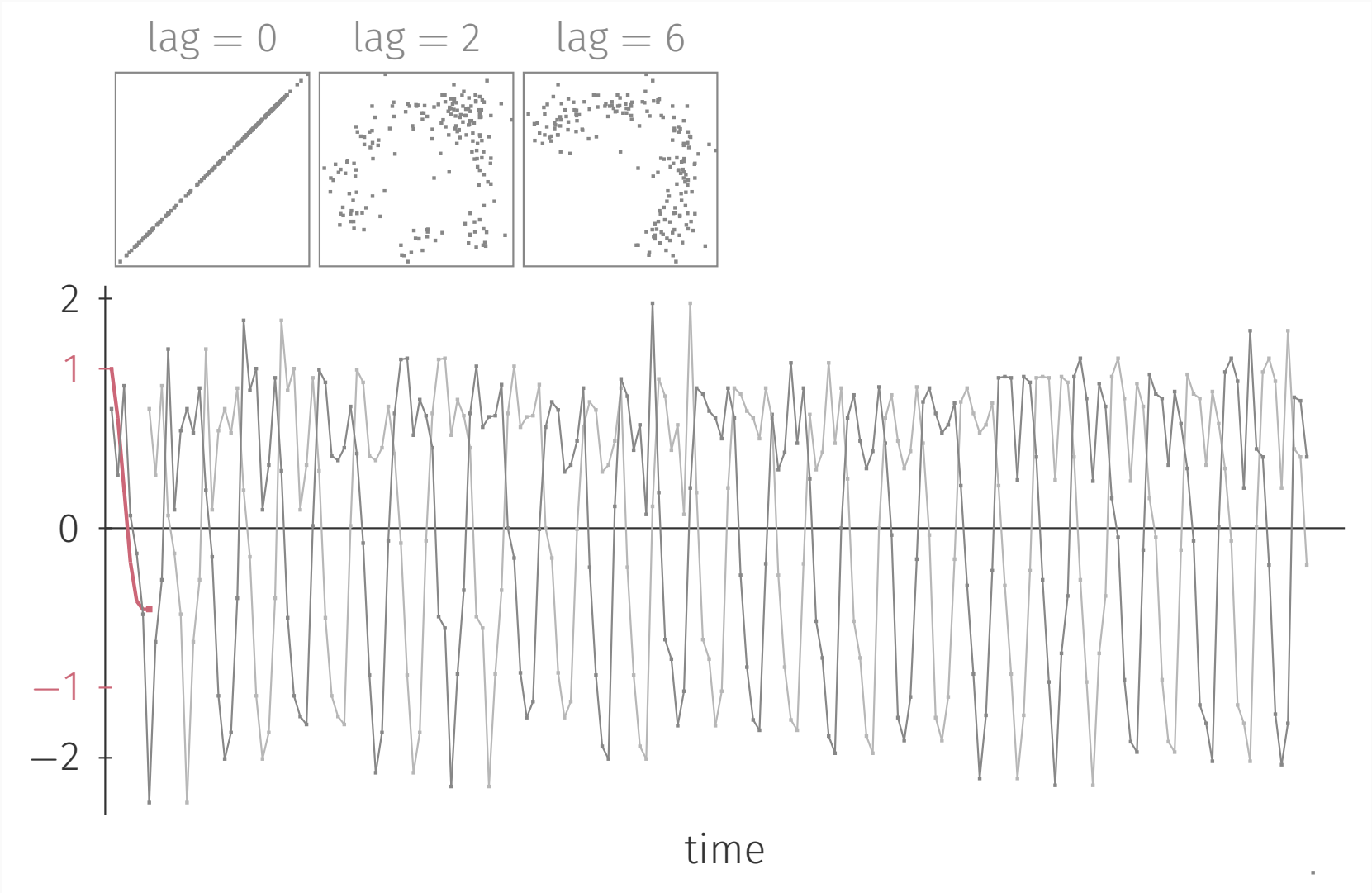
Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980



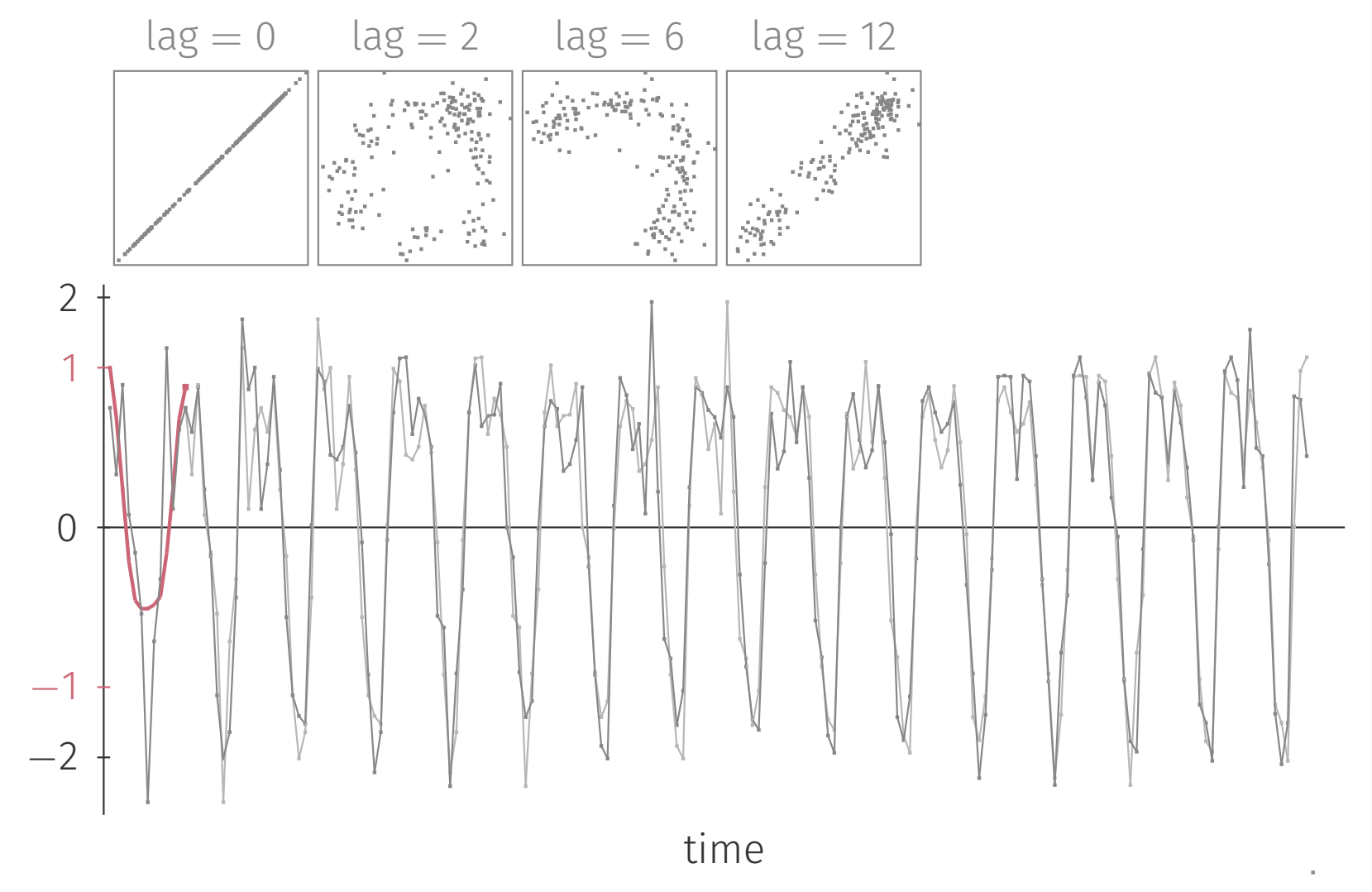
Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980



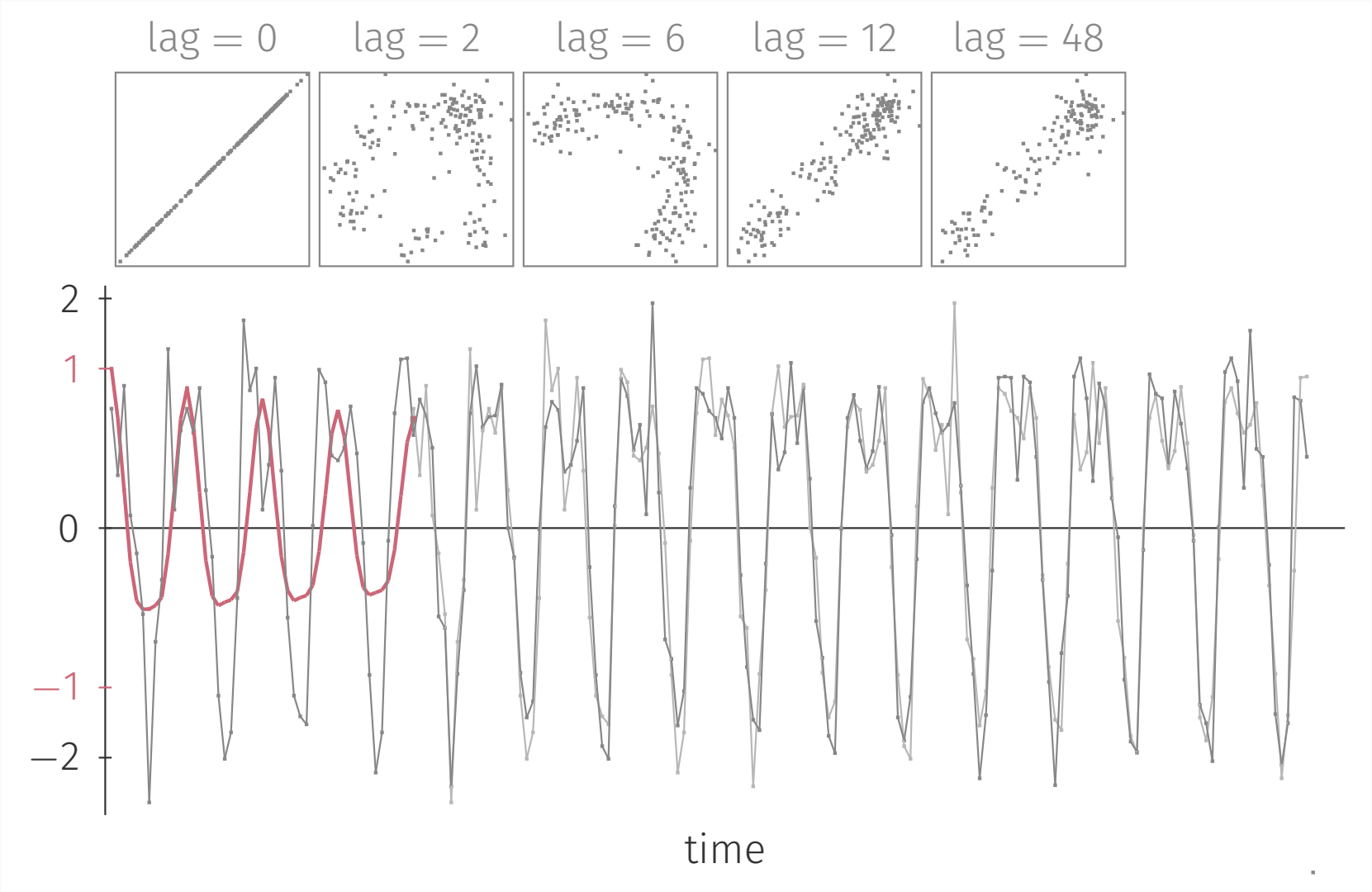
Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980



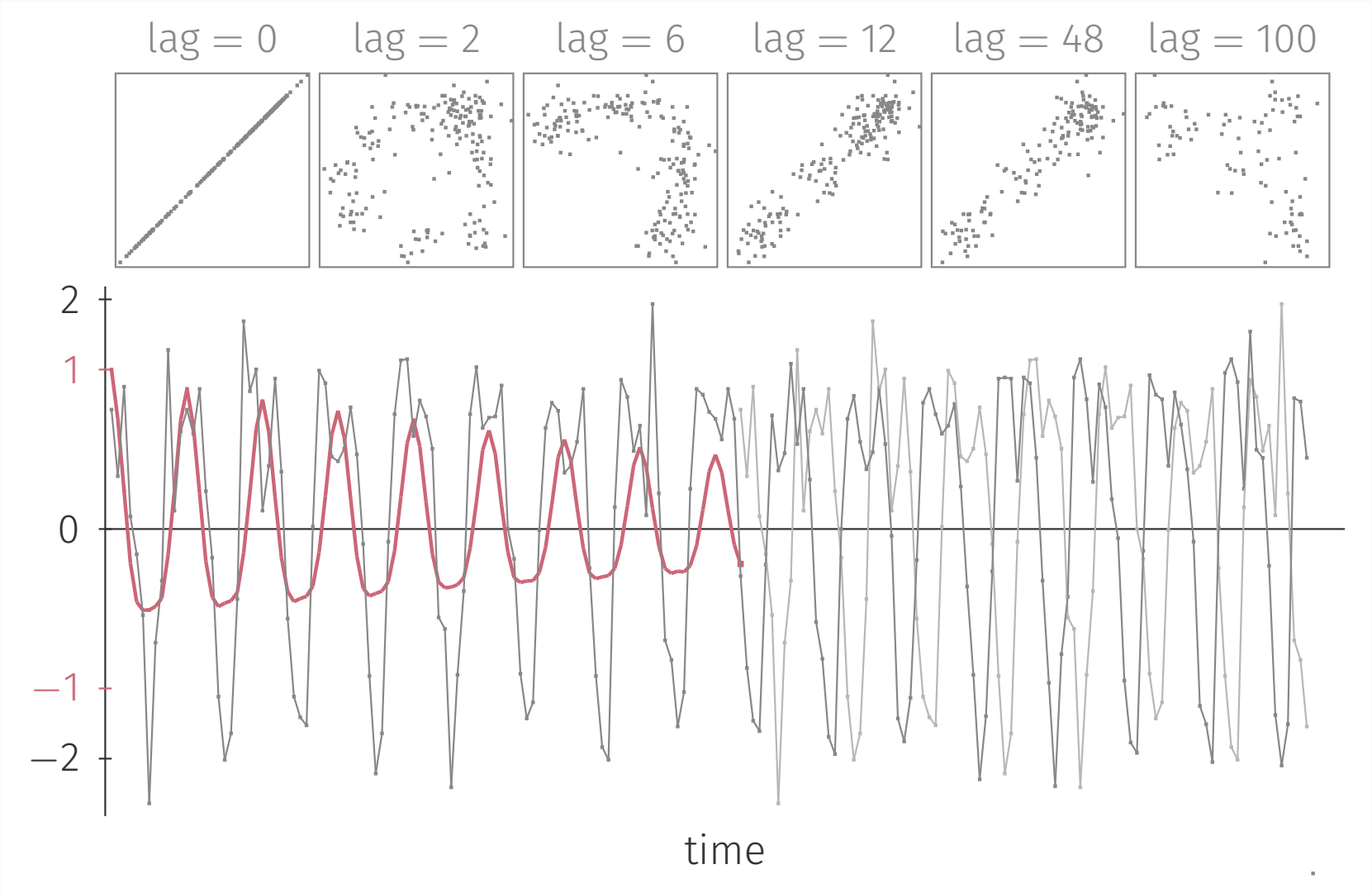
Example

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



Example

Monthly mean CO_2 ppm. in Hawaii (US) from 1965 to 1980