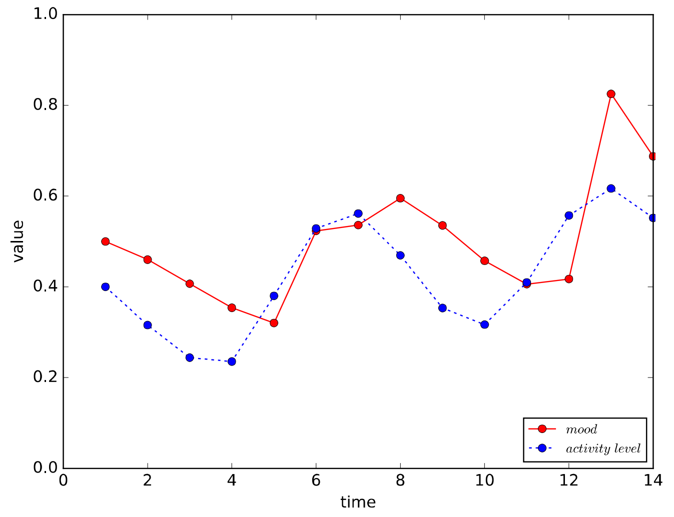
Time Series Learning

Overview

- Previously: we looked at learning algorithms that did not model time explicitly
- Today: we will look at algorithms that consider time explicitly
  - Filtering & smoothing
  - Time series modeling (ARIMA)
  - Recurrent neural networks

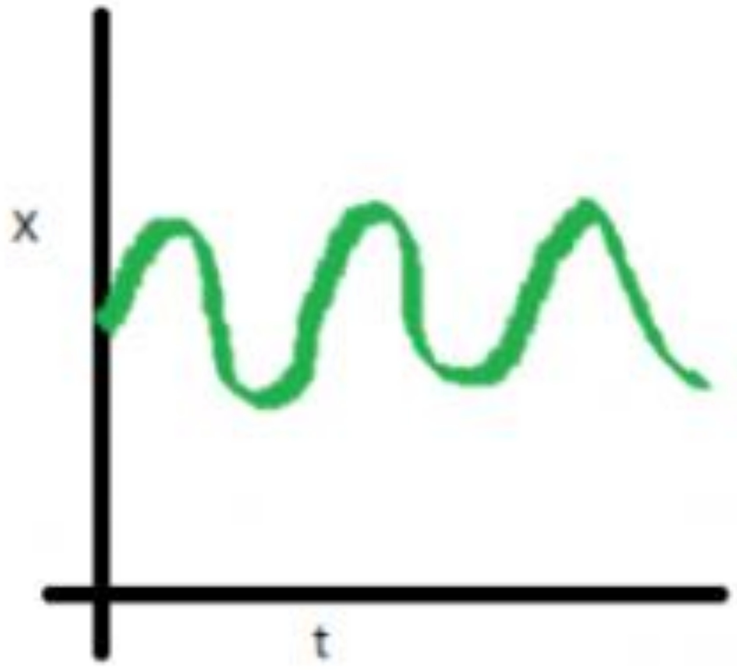


Time Series

- Time series focus on:
  - Understanding periodicity and trends (including temporal pattern changes)
  - Forecasting
- Time series can be decomposed in three components:
  - Periodic variations (daily, weekly, … seasonality)
  - Trend (how the mean evolves over time)
  - Irregular variations (left after we remove the periodic variations and trend)

Stationarity

- A stationary series is one in which the properties – mean, variance and covariance, do not vary with time.

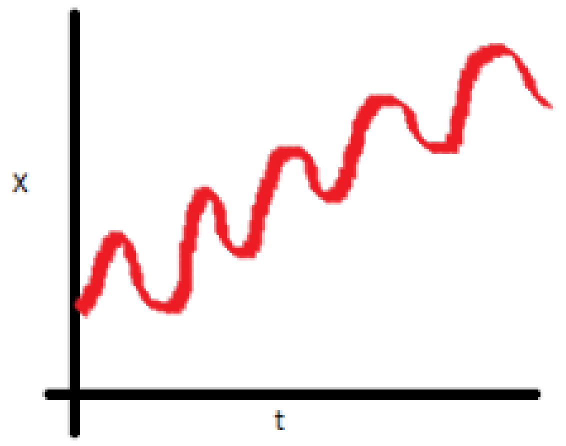


$\Rightarrow$ stationarity

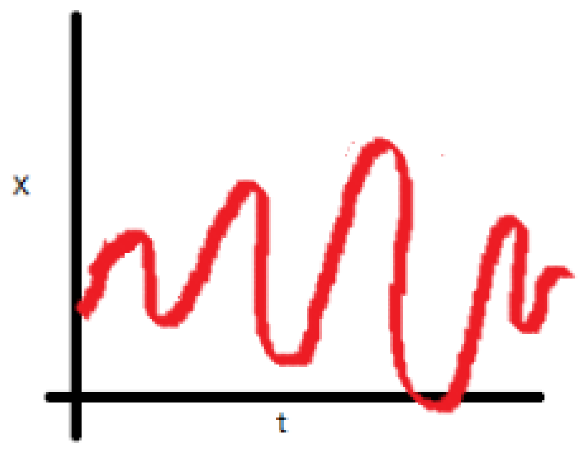
Stationarity

- A stationary series is one in which the properties – mean, variance and covariance, do not vary with time

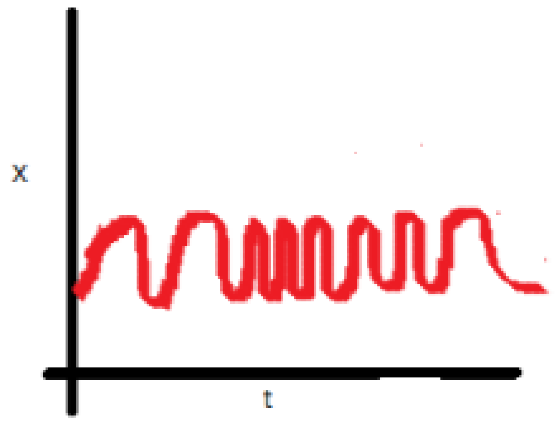
⇒ non stationarity



mean



Variance



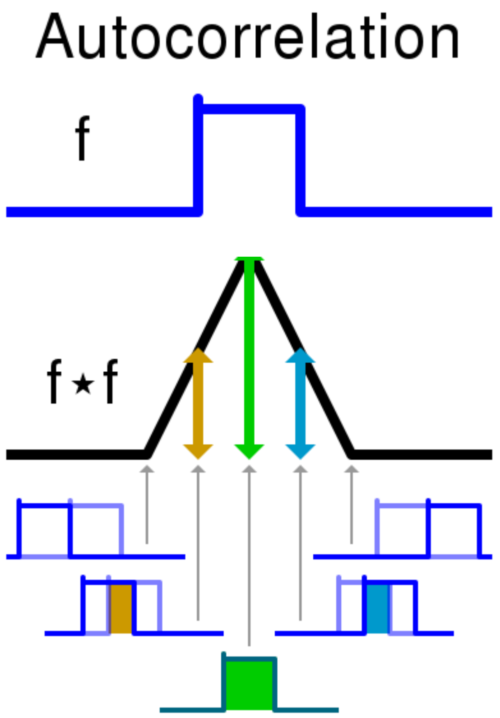
Covariance

https://medium.com/auquan/time-series-analysis-for-financial-data-part-1-stationarity-autocorrelation-and-white-noise-1a1cc2fb23f2

Stationarity

- Important concept: stationarity
  - If trends and periodic variations are removed
  - Then **variance** of the remaining **residuals** is **constant**
- Prerequisite or intermediate step for many algorithms
- Testing criterion: the lagged (λ) **autocorrelation** should remain constant

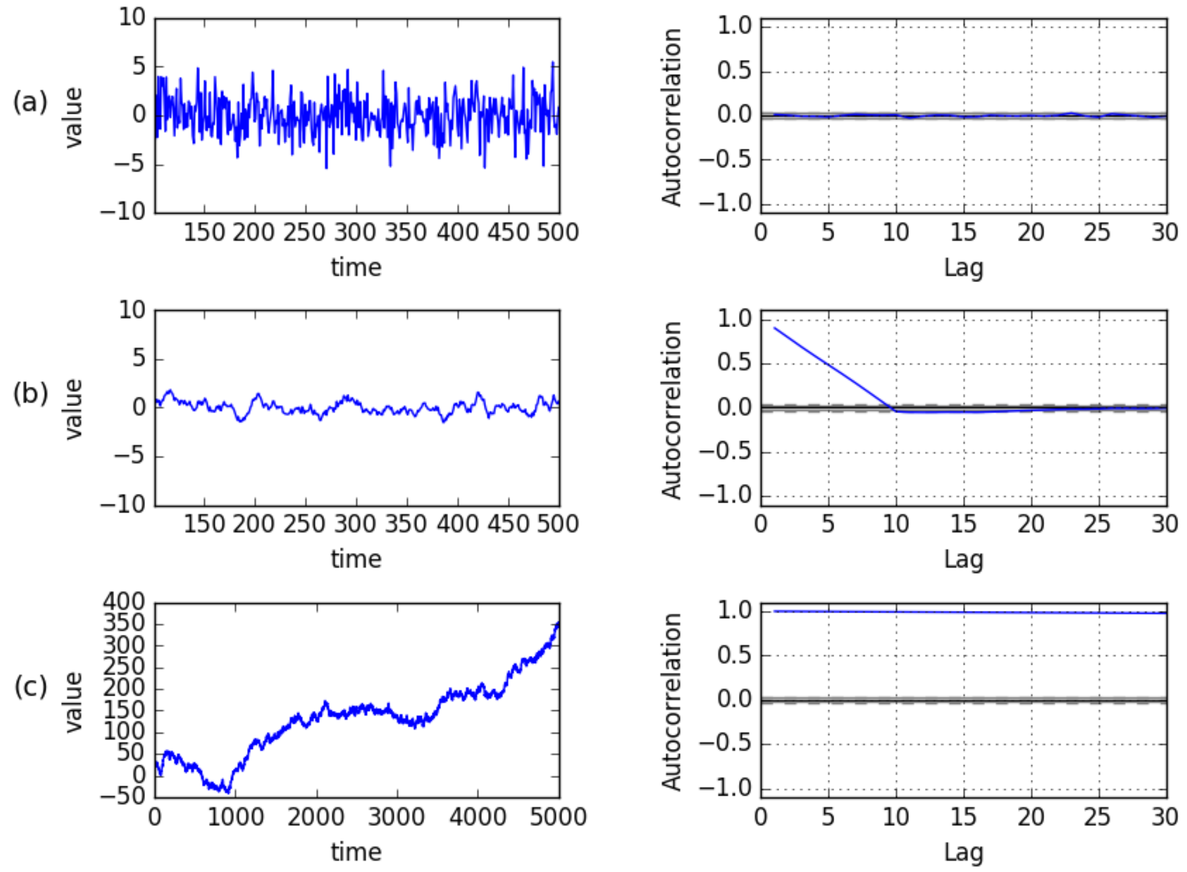
$$r_\lambda = \frac{\sum_{t=1}^{N-\lambda} (x_t - \bar{x})(x_{t+\lambda} - \bar{x})}{\sum_{t=1}^{N} (x_t - \bar{x})^2}$$



Note: x_t represents the value of one attribute

Stationarity

- Autocorrelation represents in how far there is a correlation between a time series and a shifted version of itself (with λ steps) => low autocorrelation
- Stationary if the lagged **autocorrelation** remain constant



random

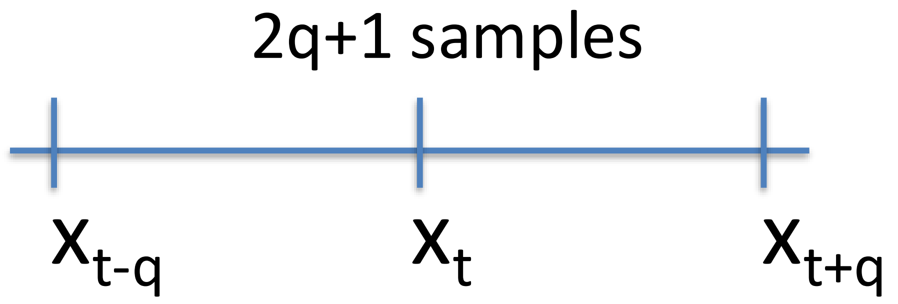
some memory over past time points

cumulative sum of (a) – non-stationary

Filtering & smoothing

- Let us assume our time series of values x_t with a fixed step size Δt
- We can apply a filter to our data, taking *q* points in the future and past into account:

$$z_t = \sum_{r=-q}^{q} a_r x_{t+r}$$



- This generates a new time series z_t

Filtering & smoothing

- What could a_r look like?
  - If $a_r = (2q+1)^{-1}$ it is just the *moving average*
  - If measurements closer to *t* are more important we can use *a triangular shape*:

$$a_r = \begin{cases} \frac{q-|r|}{q^2} & -q \le r \le q \\ 0 & otherwise \end{cases}$$

  - Or exponential smoothing: $z_t = \alpha * \overset{\text{original}}{x_t} + (1-\alpha) * z_{t-1}$ (only past time points mostly)

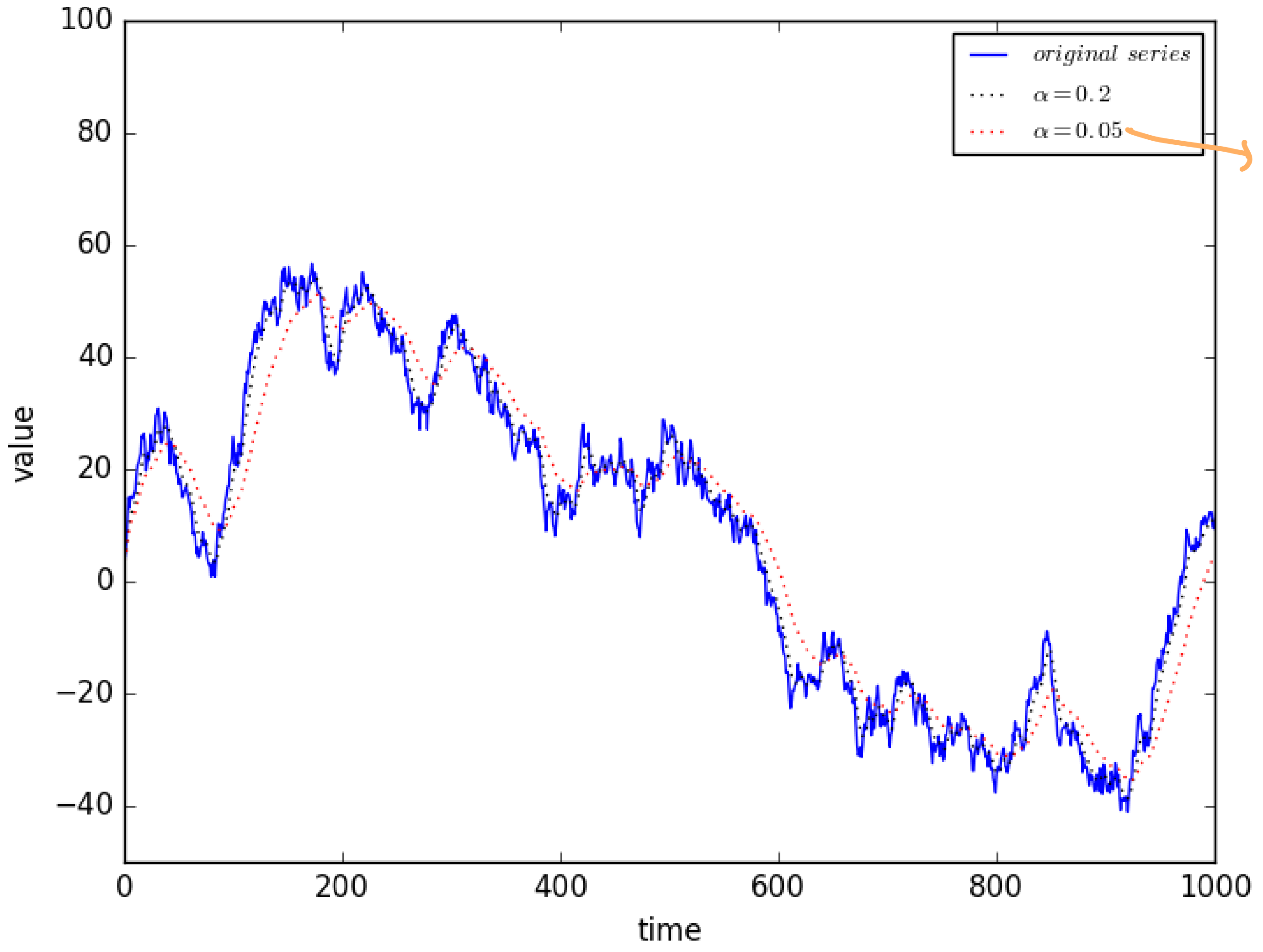
$$a_r = \alpha(1-\alpha)^{|r|}$$

α ~ 1 : no smoothing

α ~ 0 : heavy smoothing

Filtering & smoothing

- Example exponential smoothing



smoothing ↑

Filtering & smoothing: differencing

- Now how can we remove a trend?
- Let us take a filter again, but a simple one:

$z_t = x_t - x_{t-1} = \nabla x_t$

- This takes the difference between the current and previous measurement
  - A long term trend has more or less the same influence on the previous and current time point
- We can apply this operator *d* times (e.g. *d*=2): $\nabla^2 x_t = \nabla x_t - \nabla x_{t-1}$

Filtering & smoothing

- But x_{t-1} might not be a good estimation of the trend
- We can alternatively use an exponential smoothing z_t and take $x_t - z_t$

Filtering & smoothing

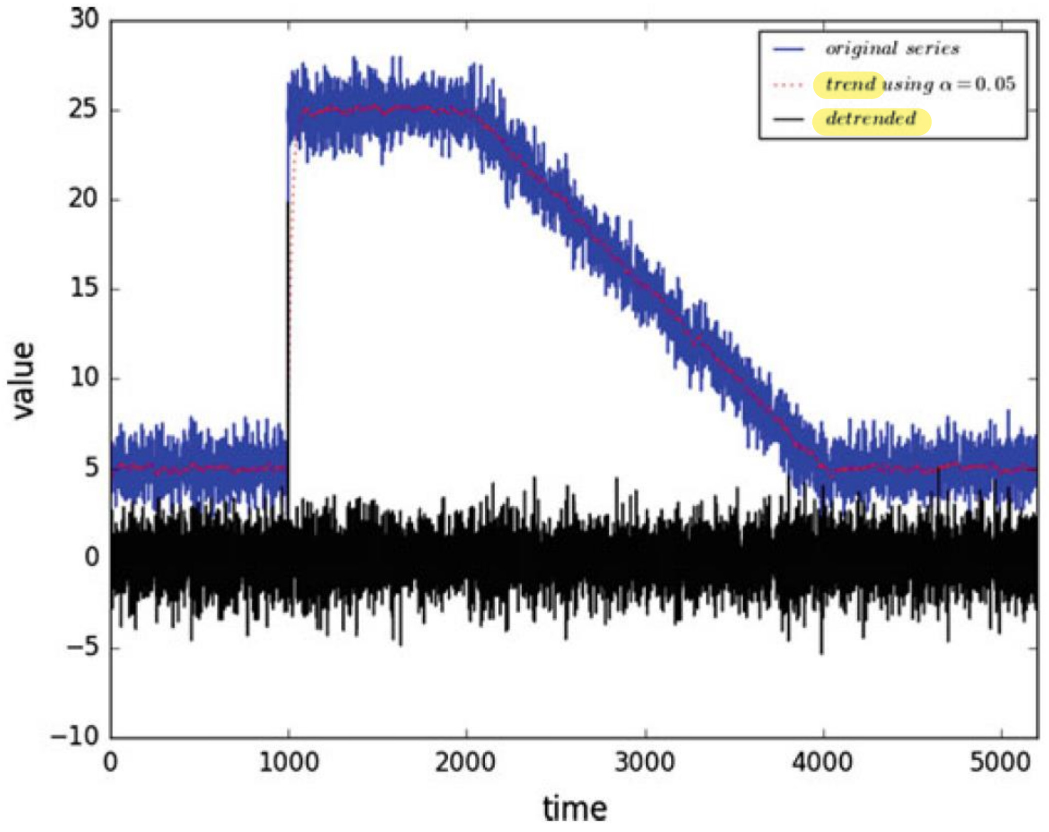


Fig. 8.4 *Black* solid line = example time series, *red* dashed = trends through exponential smoothing, *blue* dotted line = detrended time series

Detrending w/ exponential smoothing: $x_t - z_t$ where $z_t = \alpha * x_t + (1-\alpha) * z_{t-1}$

A time series can be decomposed

Original Data

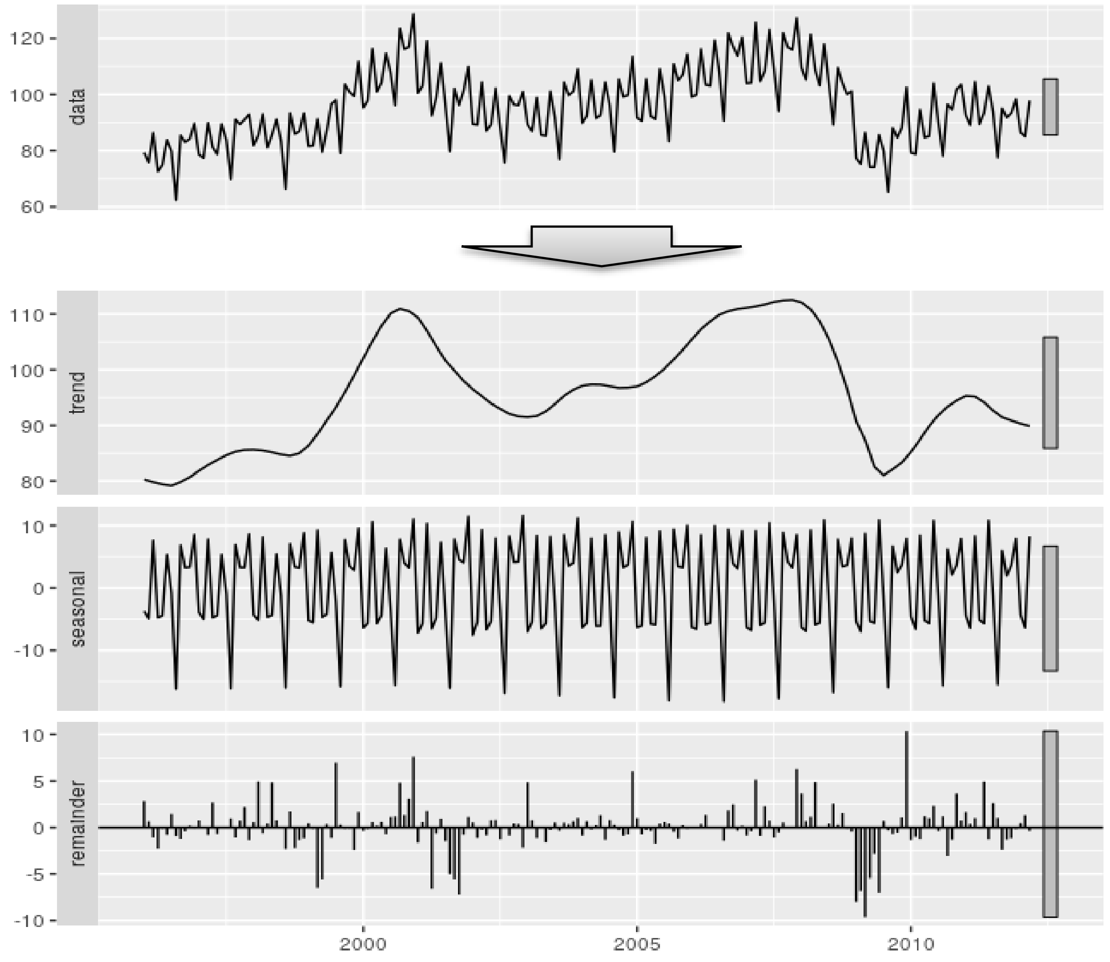
Trend

+

Seasonal

+

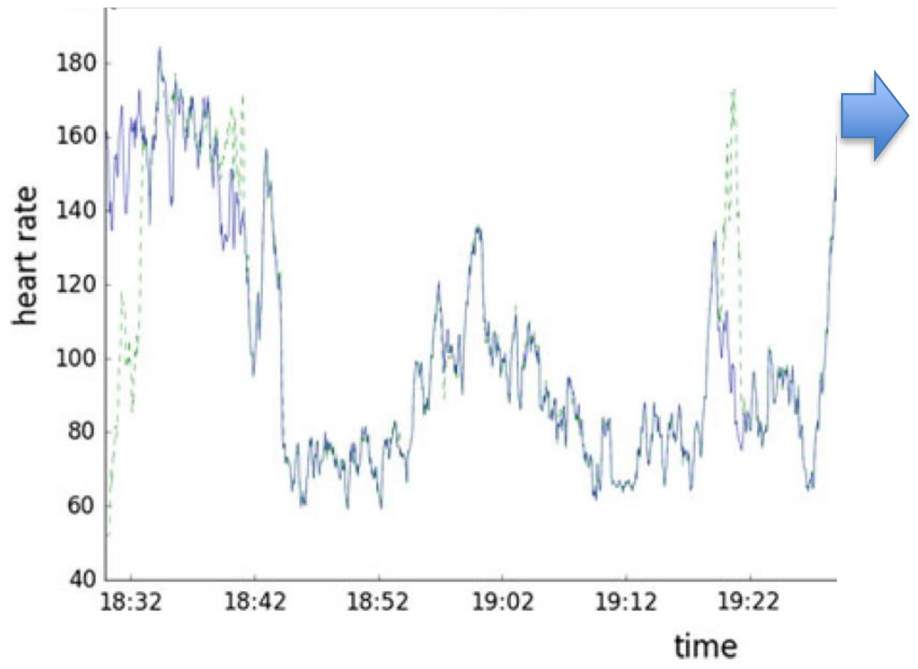
Remainder



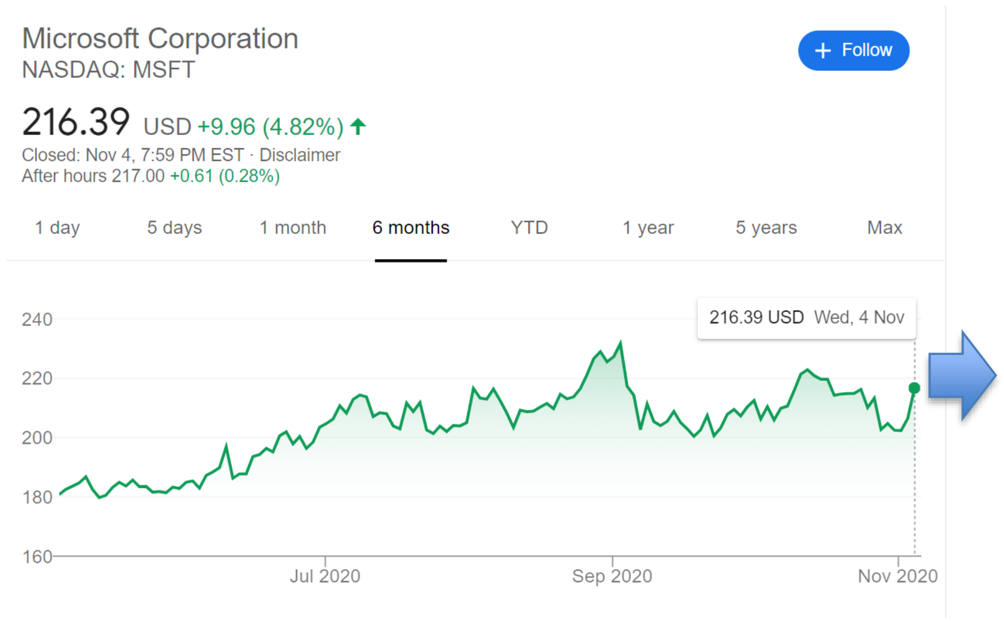
Year

Forecasting: Principles and Practice, Rob J Hyndman and George Athanasopoulos
https://otexts.com/fpp2/components.html

Time series prediction?



Forecasting my next heart rate?


Microsoft Corporation
NASDAQ: MSFT

216.39 USD +9.96 (4.82%)

Closed: Nov 4, 7:59 PM EST · Disclaimer
After hours 217.00 +0.61 (0.28%)

Forecasting stock prices?

ARIMA

difference

- **ARIMA**: Auto Regressive Integrated Moving Average
  - A class of models that ‘explains’ a given time series based on
    - AR → its **own past values**, that is, its own lags
    - MA → the **lagged forecast errors**,
  - so that equation can be used to forecast future values

ARIMA

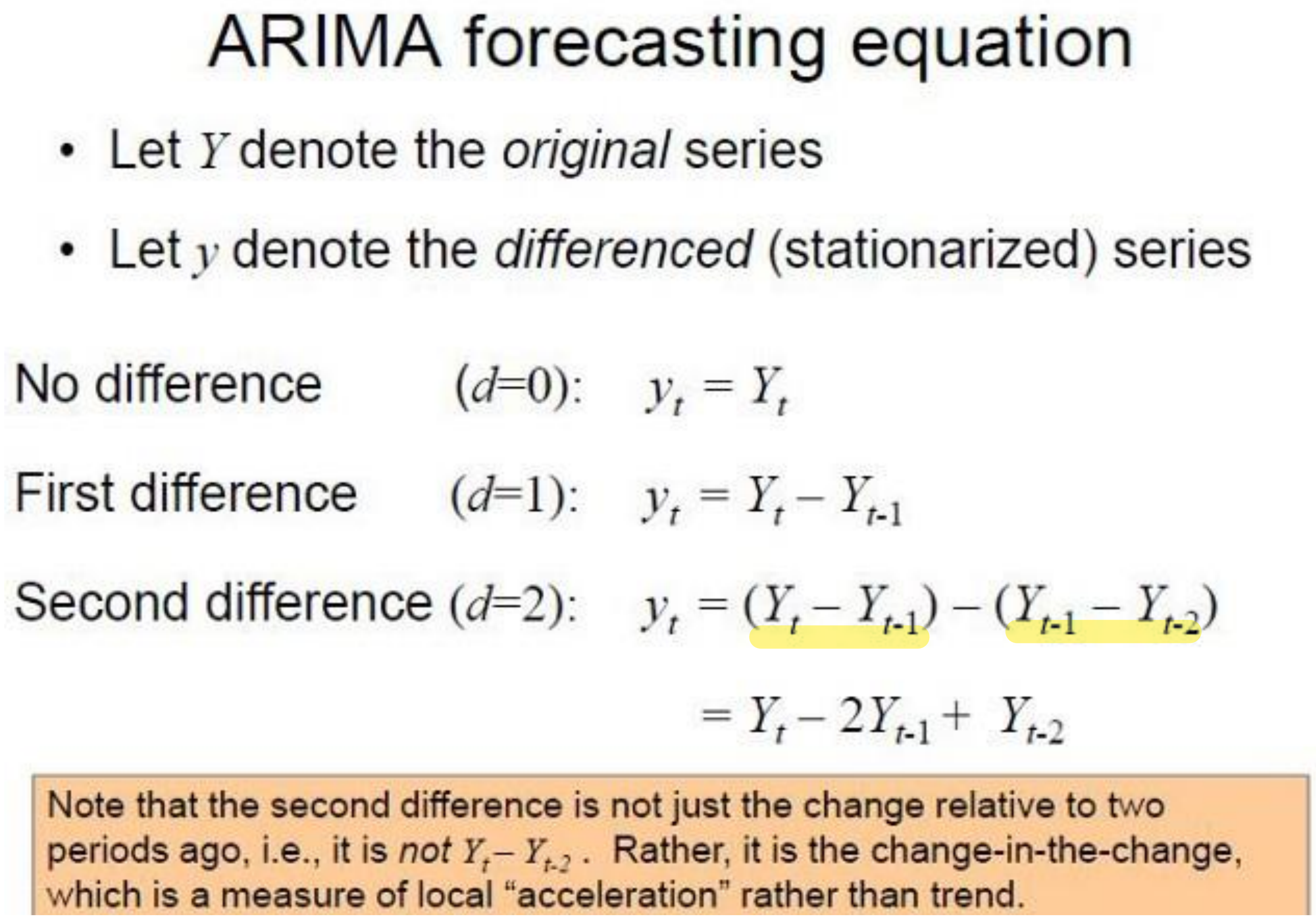
- **Auto Regressive (AR only) model**
  - Current value is explained by its **own past values**, that is, its own lags

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \epsilon_1$$

- **Moving Average (MA only) model**
  - Current value is explained by the **lagged forecast errors**

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

ARIMA



ARIMA forecasting equation

- Let Y denote the *original* series
- Let y denote the *differenced* (stationarized) series

No difference $(d=0)$: $y_t = Y_t$

First difference $(d=1)$: $y_t = Y_t - Y_{t-1}$

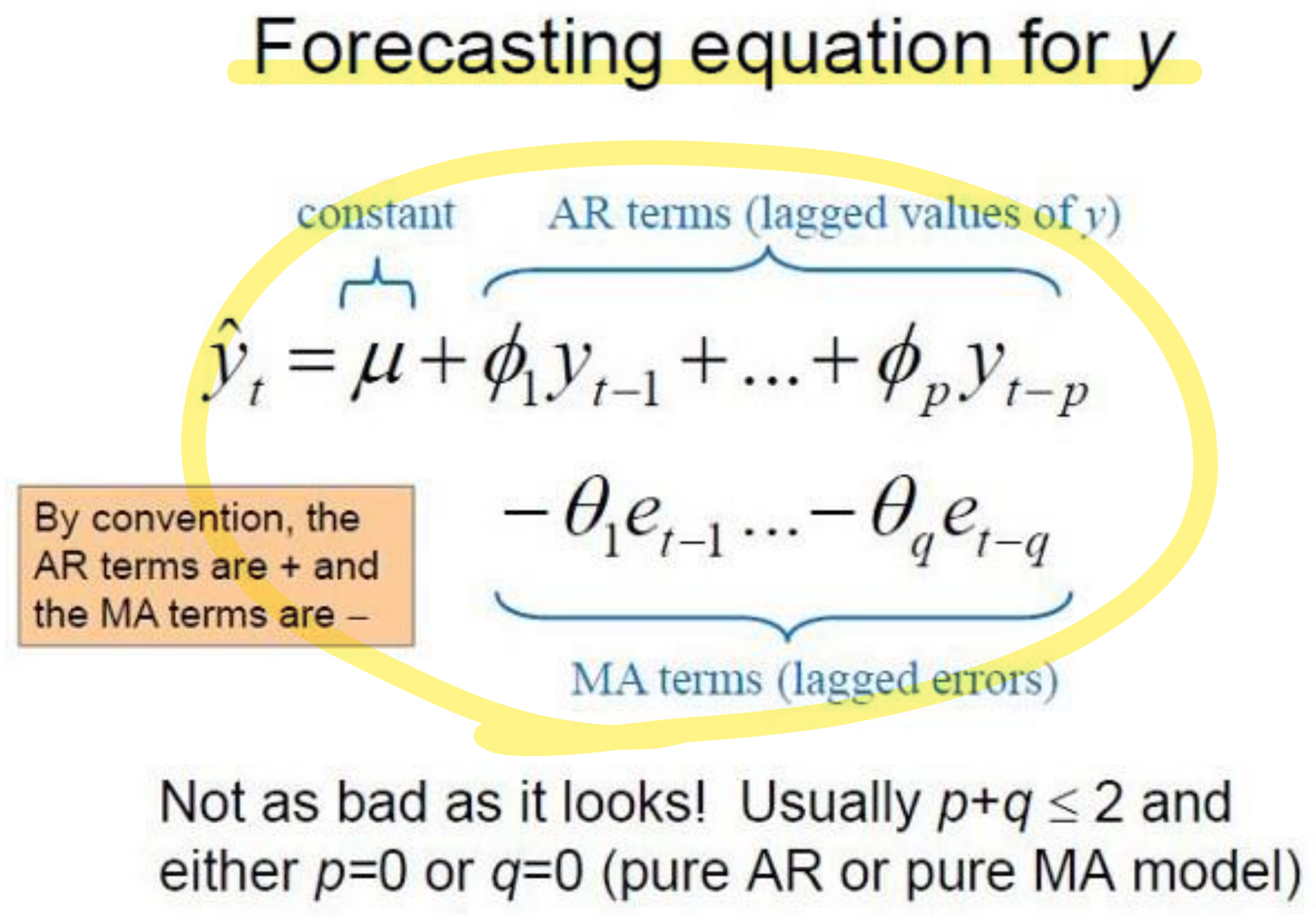
Second difference $(d=2)$: $y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2})$

$$= Y_t - 2Y_{t-1} + Y_{t-2}$$

Note that the second difference is not just the change relative to two periods ago, i.e., it is *not* $Y_t - Y_{t-2}$. Rather, it is the change-in-the-change, which is a measure of local "acceleration" rather than trend.

https://towardsdatascience.com/unboxing-arima-models-1dc09d2746f8

ARIMA



Forecasting equation for *y*

$$\hat{y}_t = \overbrace{\mu}^{\text{constant}} + \overbrace{\phi_1 y_{t-1} + \ldots + \phi_p y_{t-p}}^{\text{AR terms (lagged values of } y)}$$

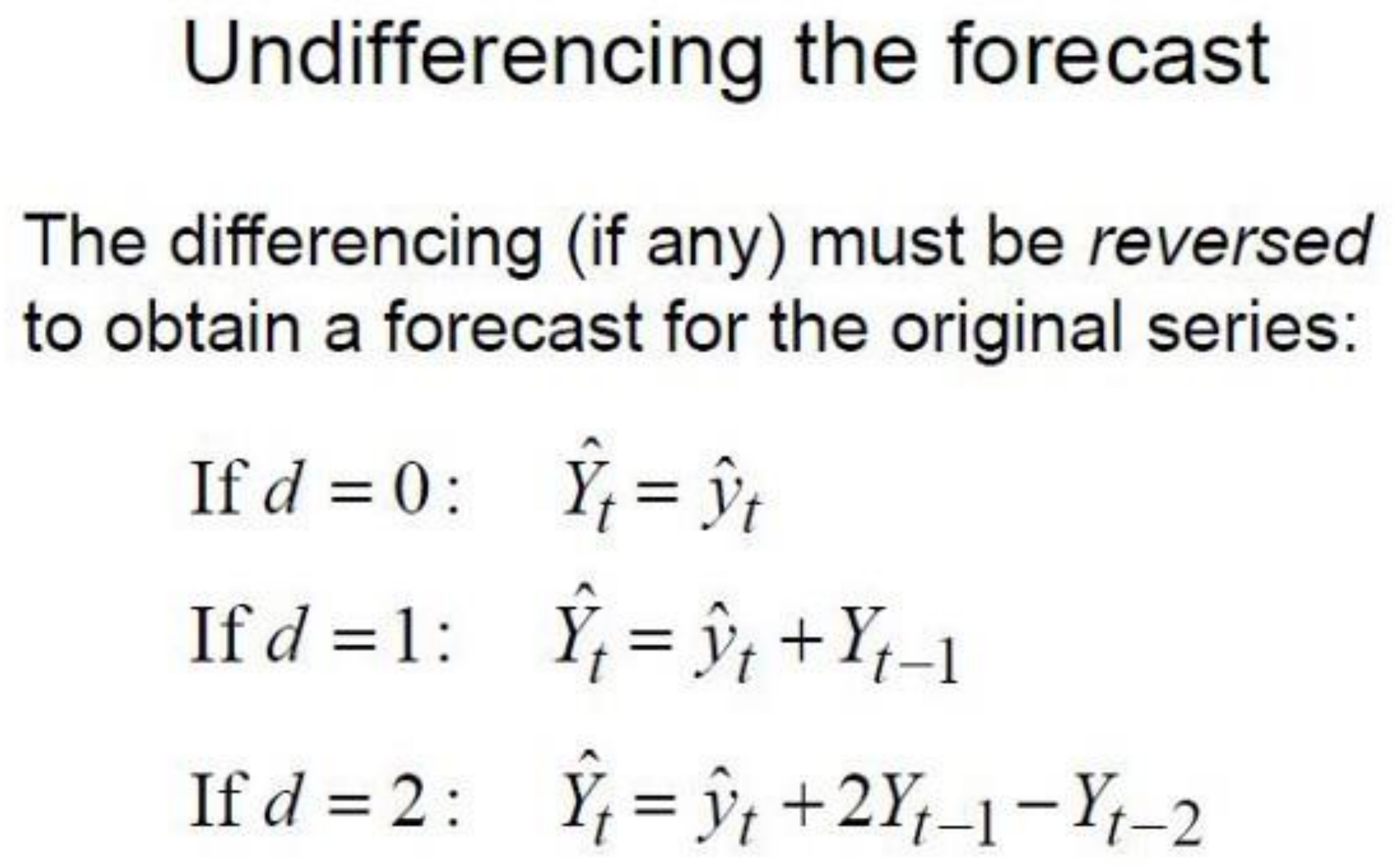
$$\underbrace{-\theta_1 e_{t-1} \ldots - \theta_q e_{t-q}}_{\text{MA terms (lagged errors)}}$$

By convention, the AR terms are + and the MA terms are –

Not as bad as it looks! Usually $p+q \leq 2$ and either $p=0$ or $q=0$ (pure AR or pure MA model)

https://towardsdatascience.com/unboxing-arima-models-1dc09d2746f8

ARIMA



Undifferencing the forecast

The differencing (if any) must be *reversed* to obtain a forecast for the original series:

If $d = 0$: $\hat{Y}_t = \hat{y}_t$

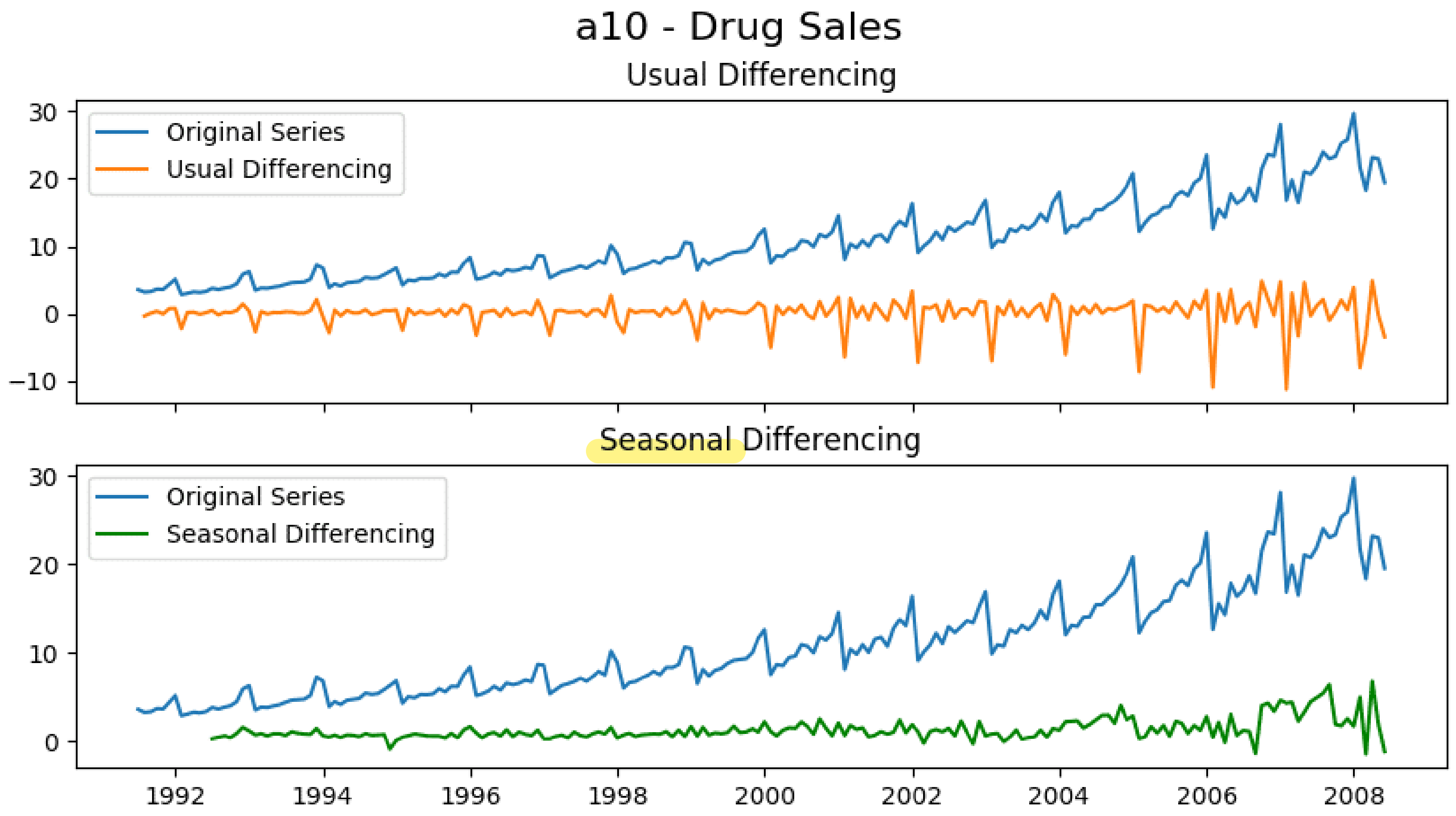
If $d = 1$: $\hat{Y}_t = \hat{y}_t + Y_{t-1}$

If $d = 2$: $\hat{Y}_t = \hat{y}_t + 2Y_{t-1} - Y_{t-2}$

https://towardsdatascience.com/unboxing-arima-models-1dc09d2746f8

ARIMA

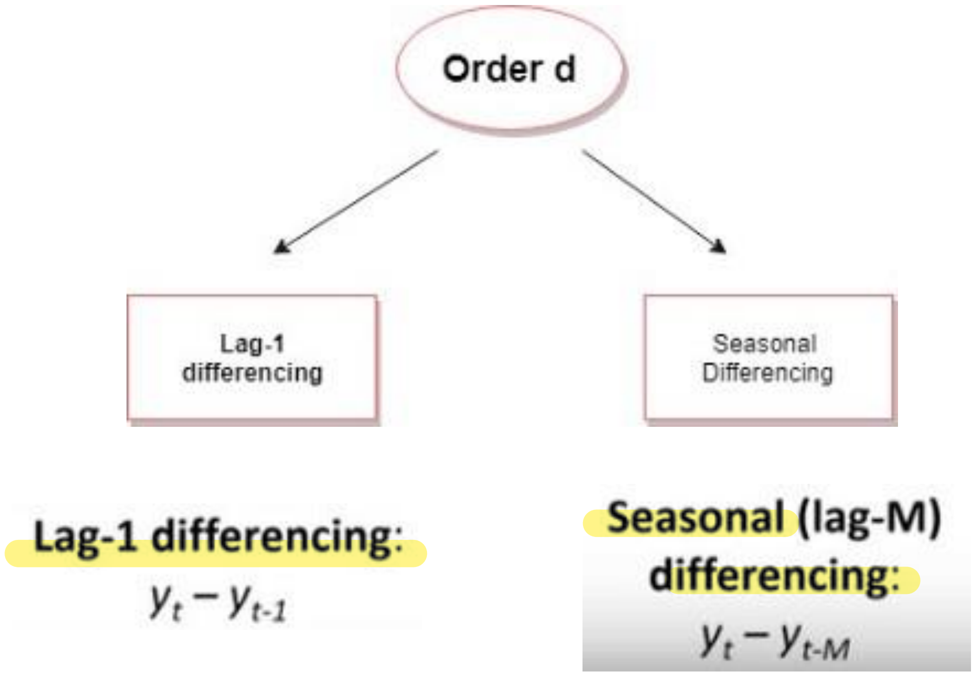
- **Seasonal data requires** "seasonal differencing": $Y_t - Y_{t-m}$



https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

ARIMA

- Trend vs. seasonal effects ➔ seasonal time series needs a special care: Seasonal ARIMA


Order d
Lag-1 differencing
Seasonal Differencing
Lag-1 differencing: $y_t - y_{t-1}$
Seasonal (lag-M) differencing: $y_t - y_{t-M}$

Regular ARIMA **Seasonal ARIMA (SARIMA)**

https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

ARIMA - example

- Let’s analyze a fragment of 4000 accelerometer data measurements (≈20 s) which we evenly space at a 10ms level to accommodate for a proper time series application

ARIMA – p & q

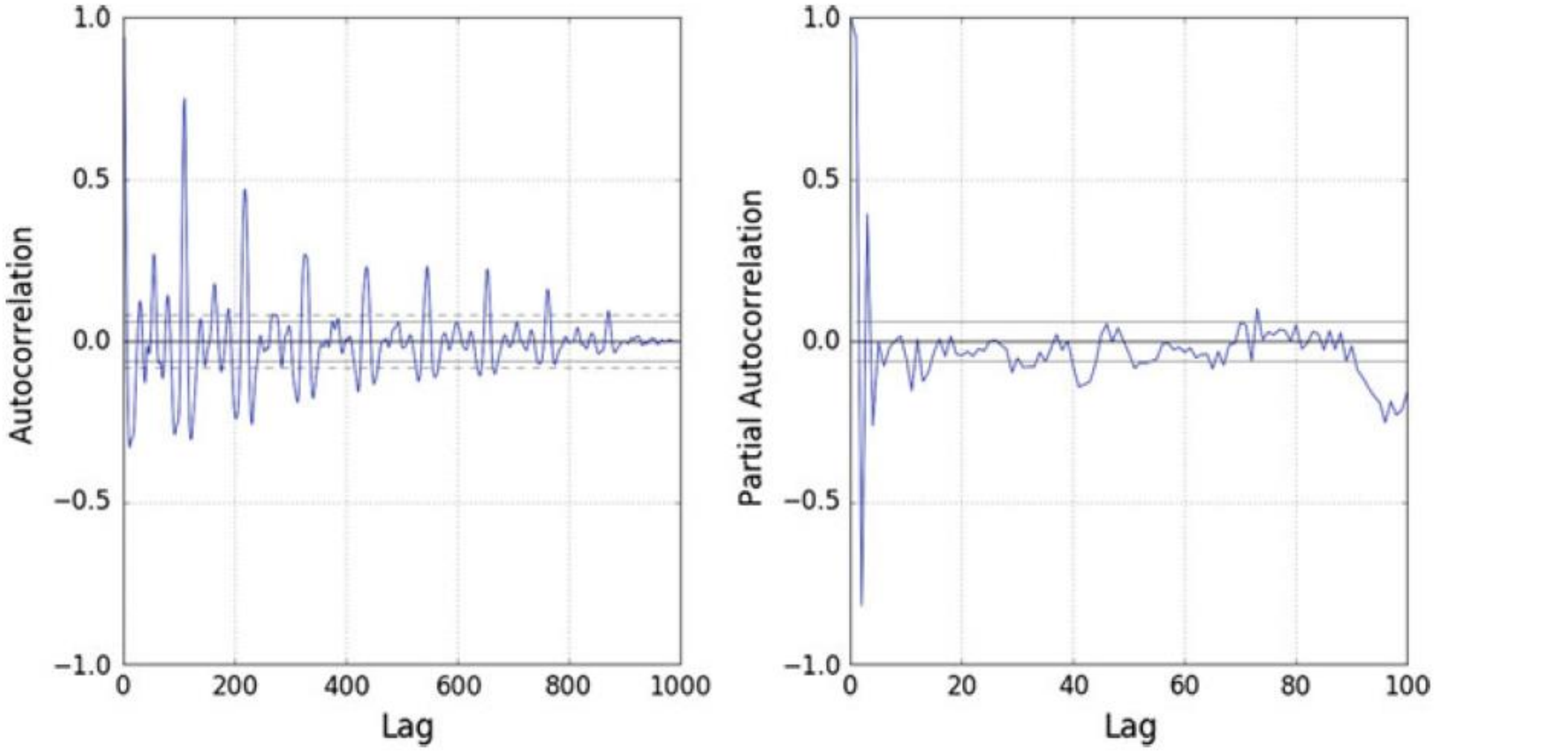
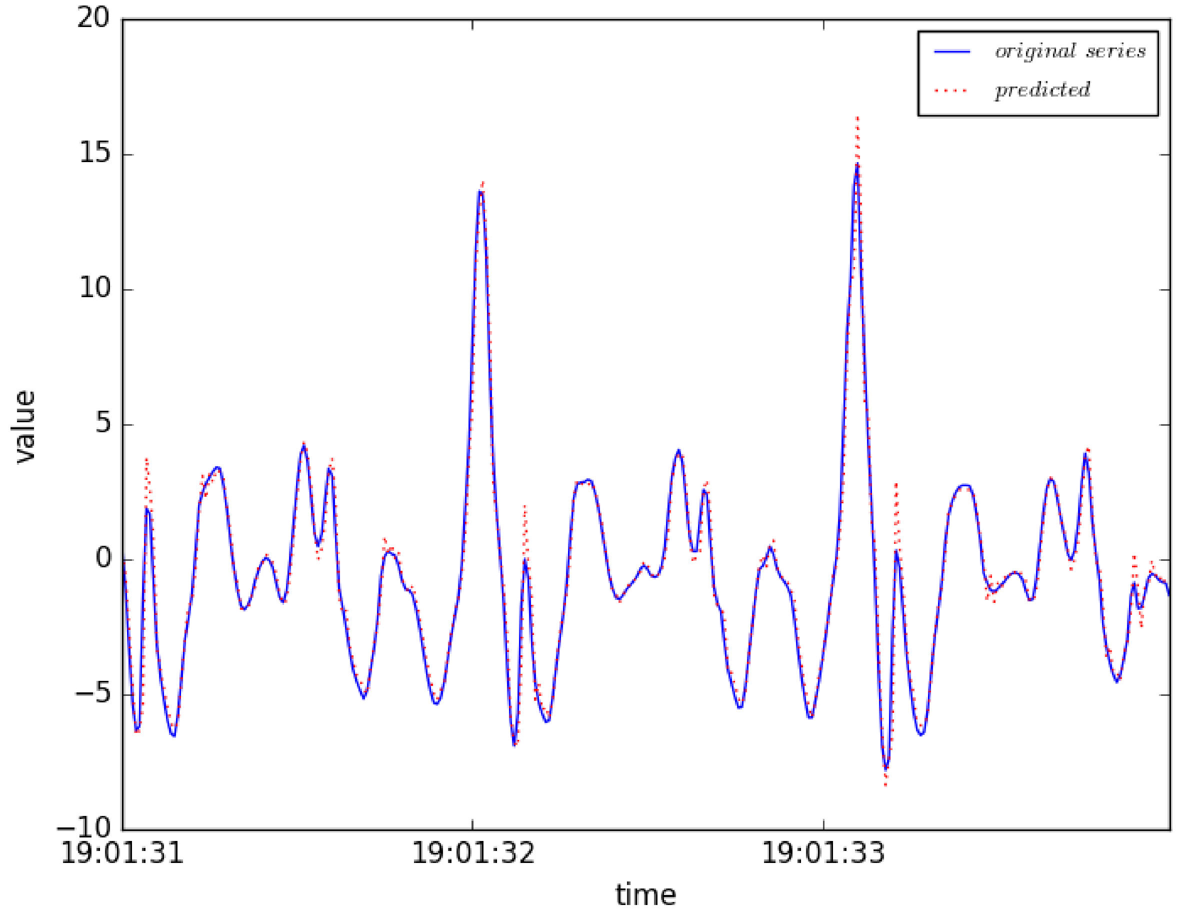


Fig. 8.7 Autocorrelation Function (ACF) for a set of 2000 raw data measurements (≈10 s) and Partial Autocorrelation Function (PACF) for the same set of raw data

ARIMA - example (1)

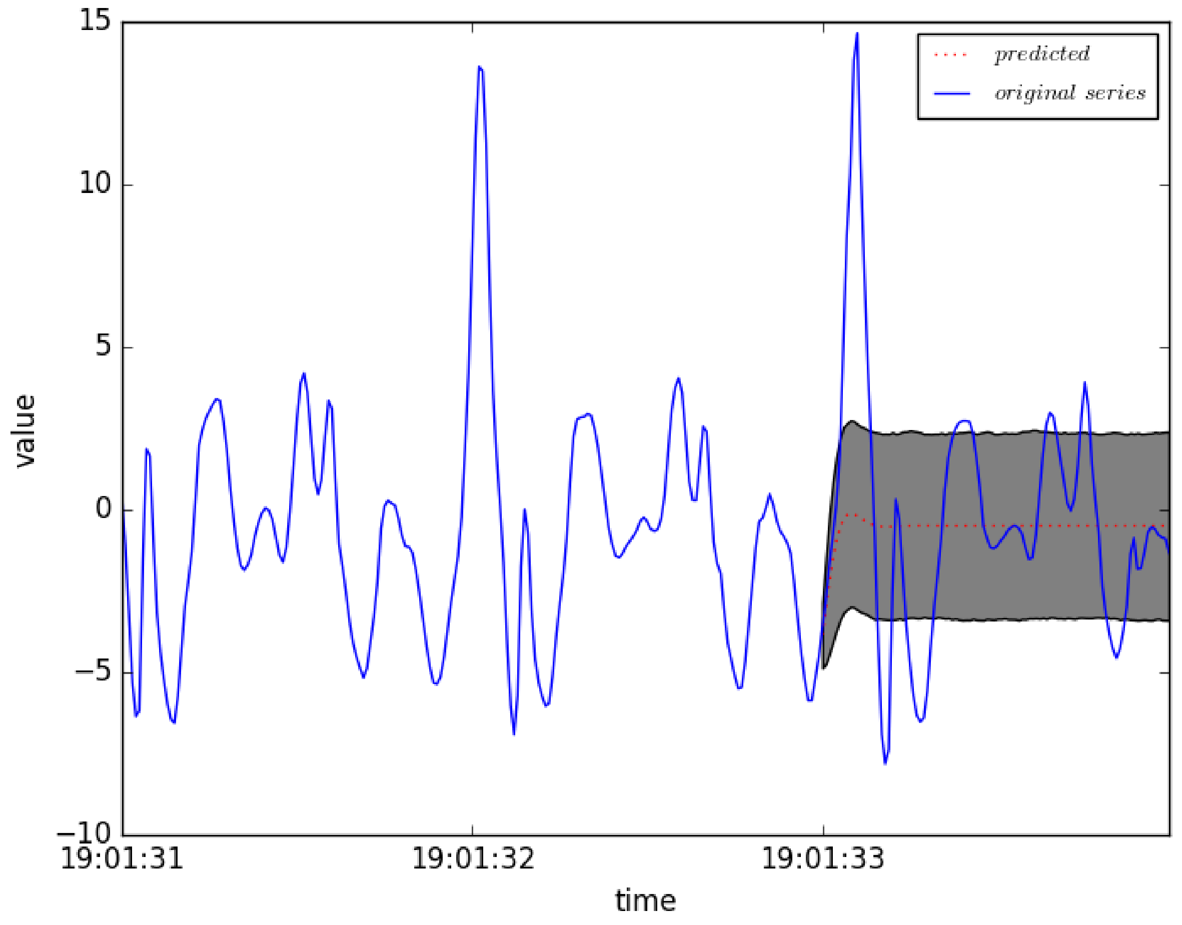
- One step ahead prediction (p=3, q=2)



The *blue line* represents 500 measurements of the original data, the *red line* the one-step ahead-prediction

ARIMA - example (2)

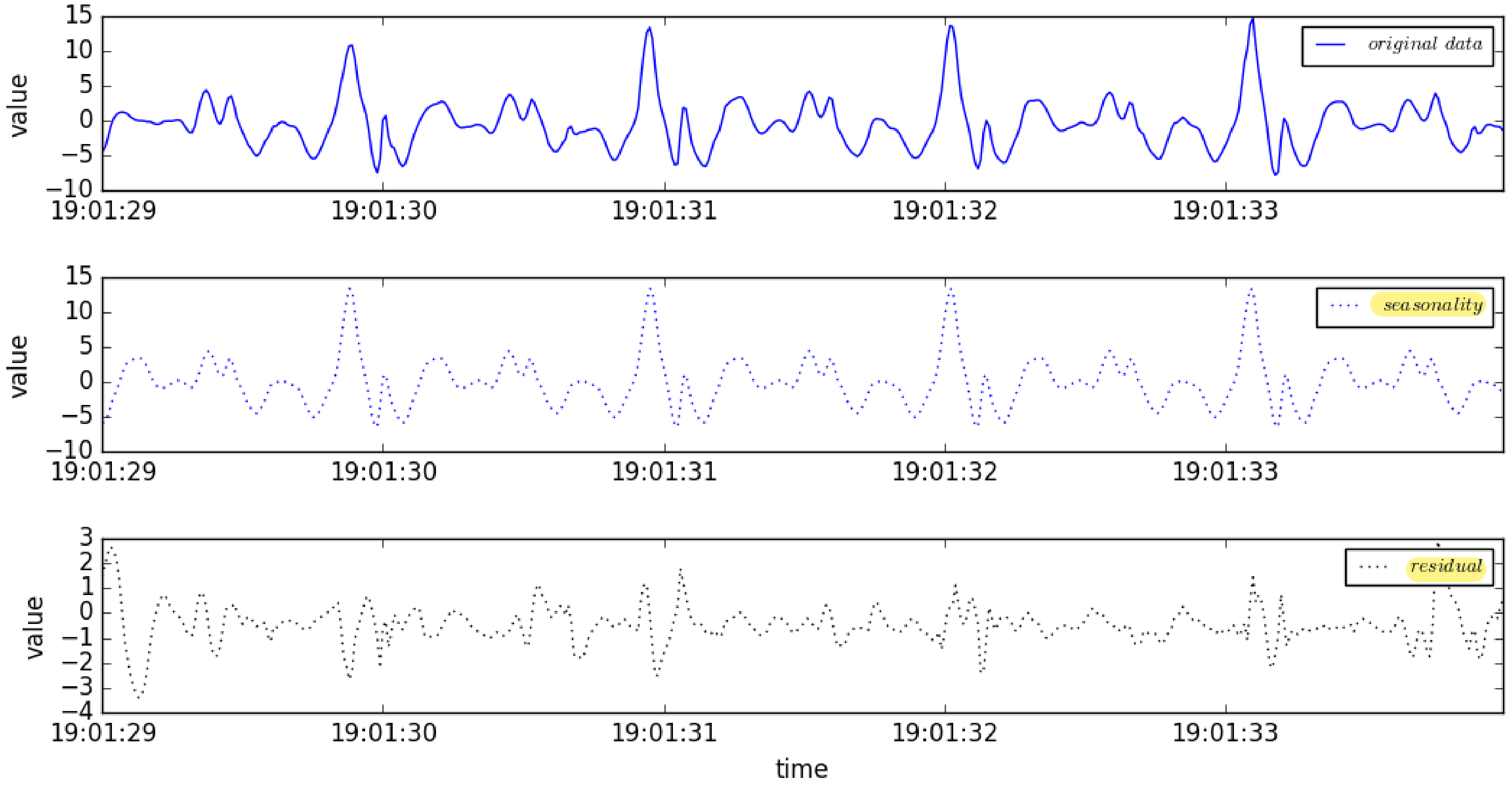
- Multiple steps ahead prediction (p=3, q=2)



The blue line indicates the original time series that was used to estimate the ARMA model.
The red line is the “long-term” prediction, the shaded area its uncertainty

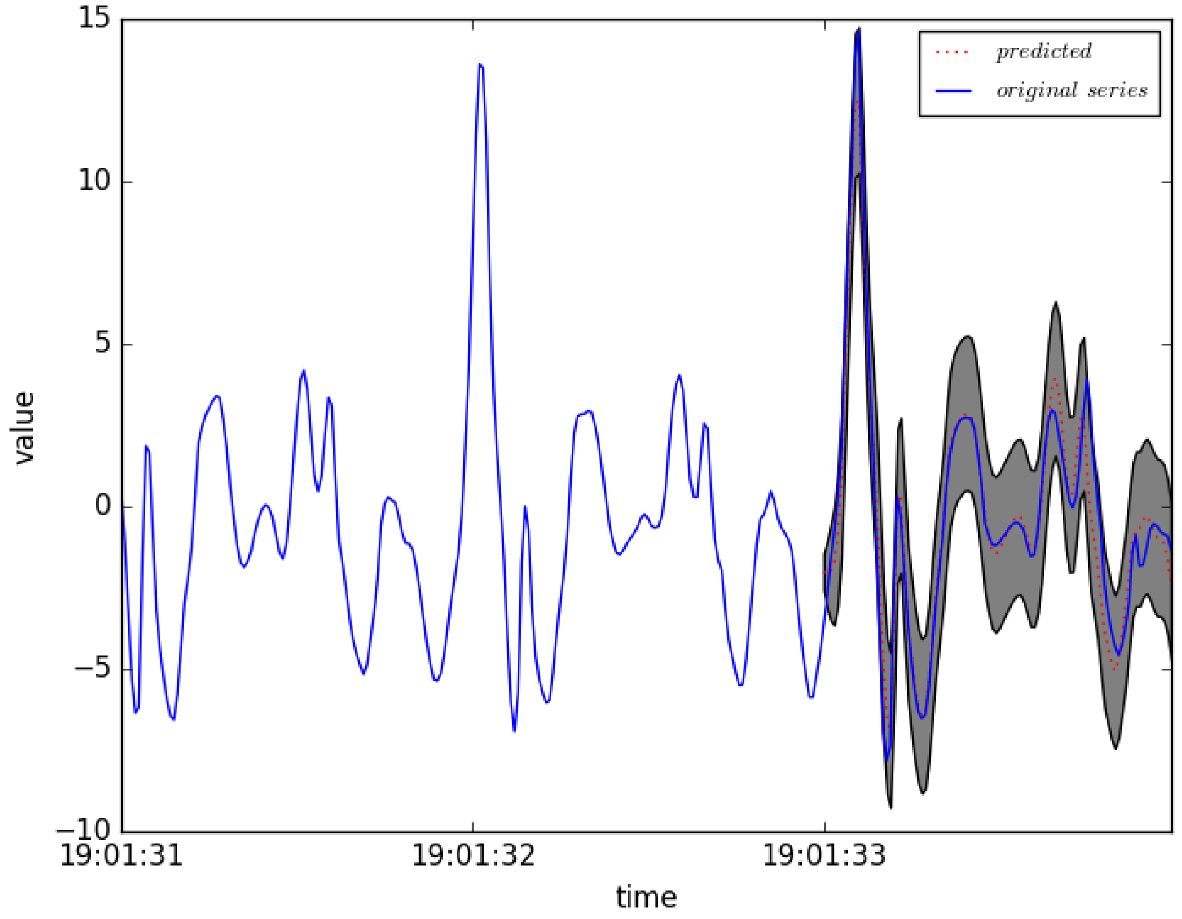
ARIMA - example (3)

- Seasonality decomposition

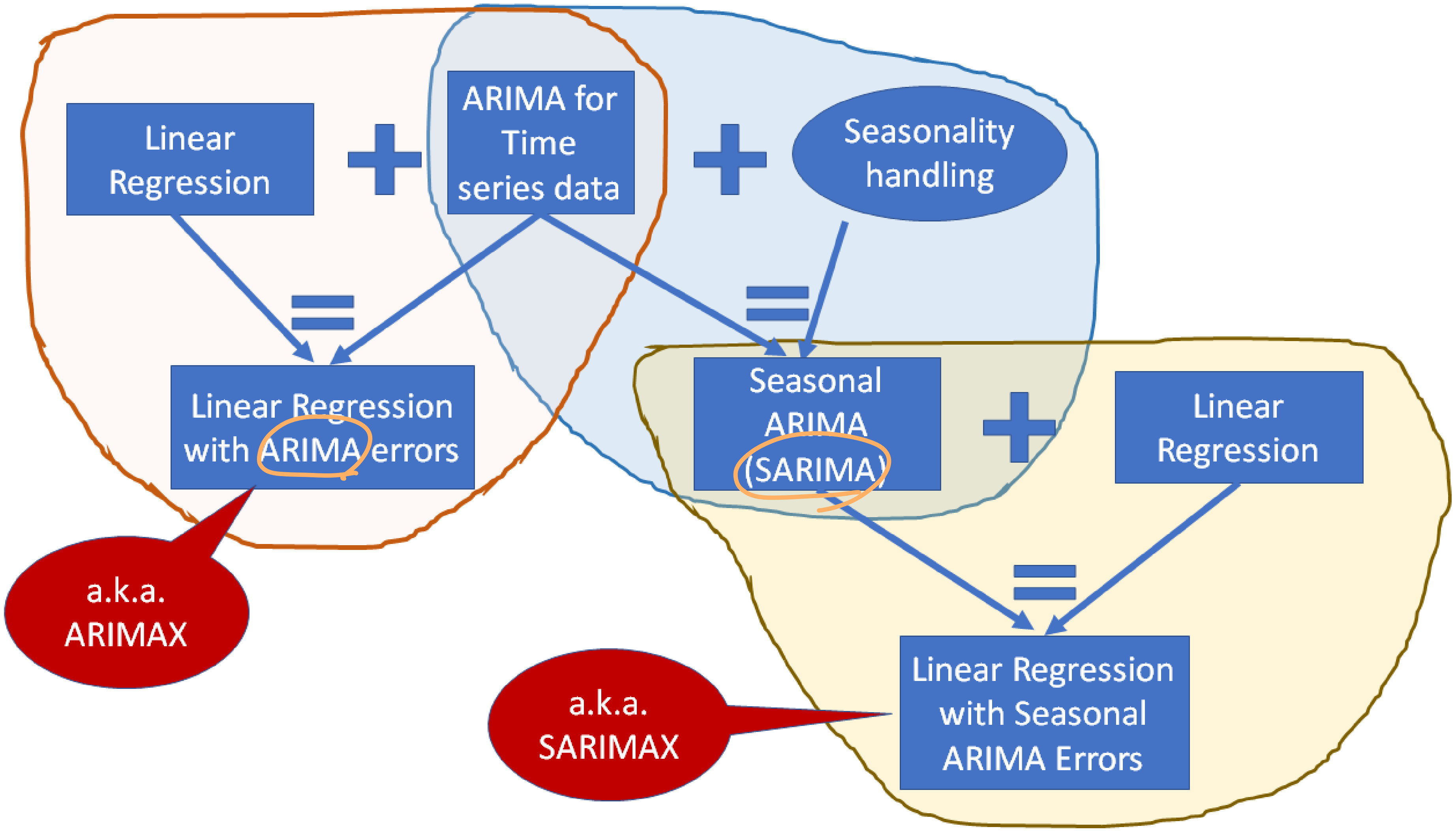


ARIMA - example (4)

- Multiple steps ahead prediction with seasonality (p=3, q=2) w/ **S-ARIMA**



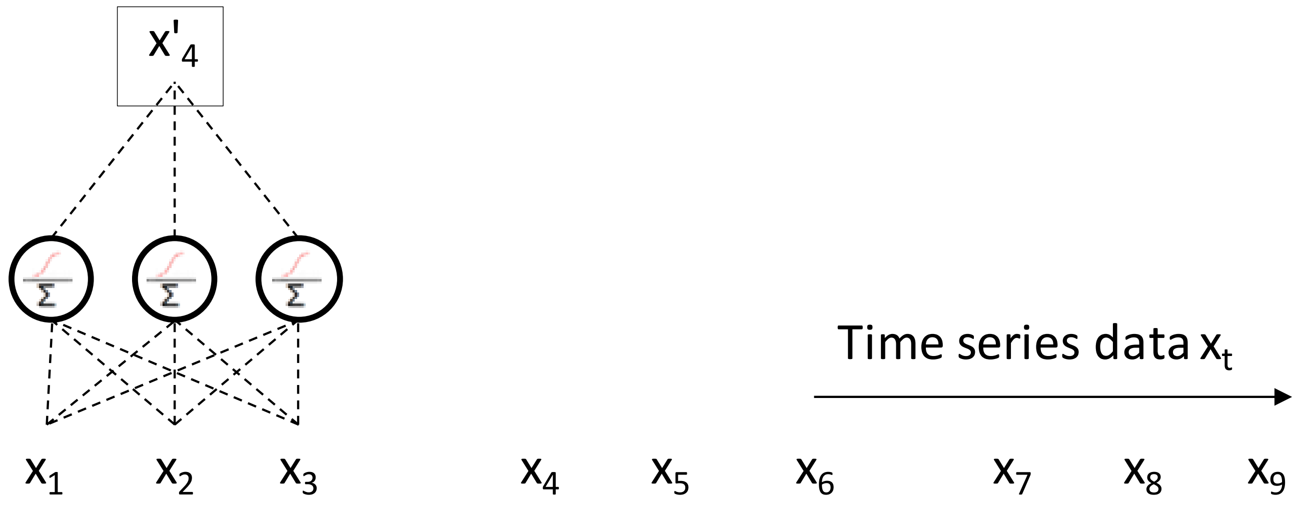
ARIMAX: with Explanatory Variables



https://towardsdatascience.com/regression-with-arima-errors-3fc06f383d73

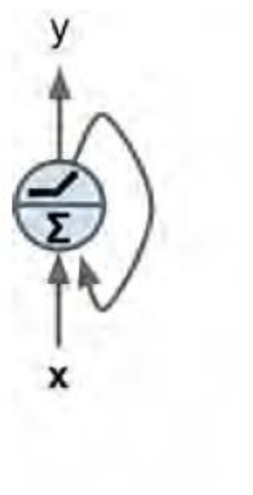
Using a neural network for time-series prediction?

- But a neural network does not consider temporal characteristics of the data



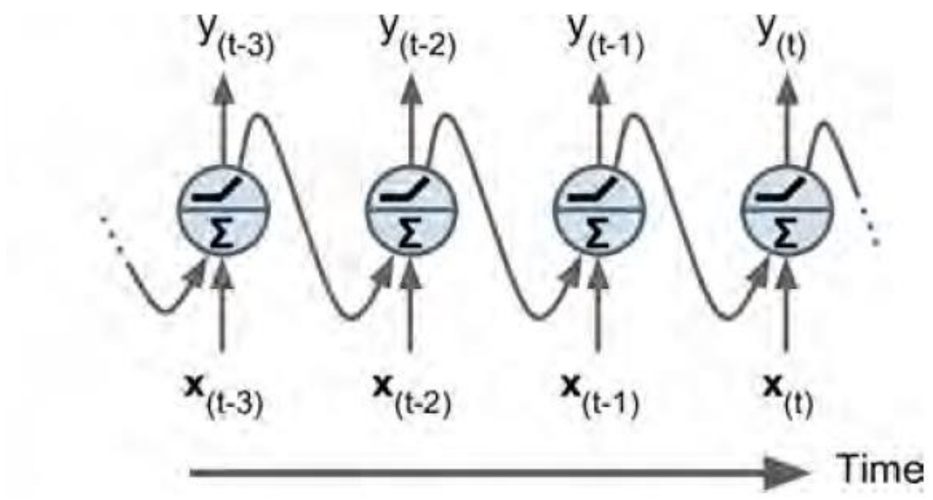
A simple neural net that inputs three items and outputs the next item

Recurrent neural network



Recurrent Neuron Network

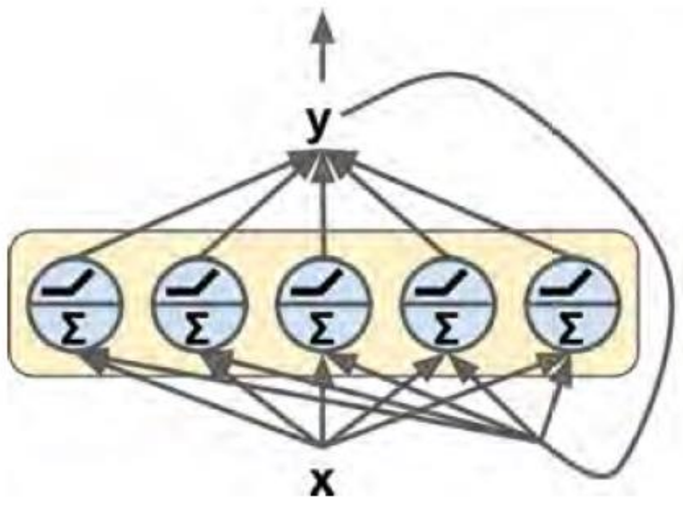
It looks very much like a feedforward neural network, except it also has connections pointing backward (very simple one with only one input & output w/ recurring output as an input)



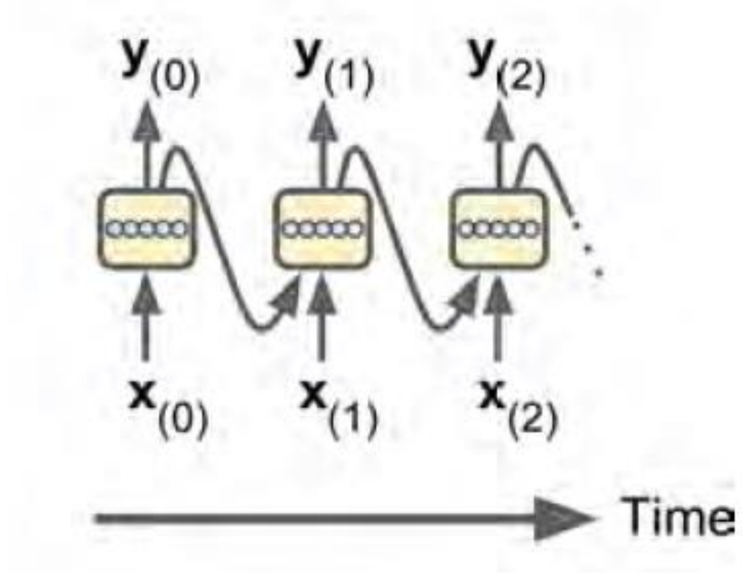
Unrolling the RNN through time

At each *time step t* (also called a *frame*), this *recurrent neuron* receives the inputs $x_{(t)}$ as well as its own output from the previous time step, $y_{(t-1)}$

Recurrent neural network



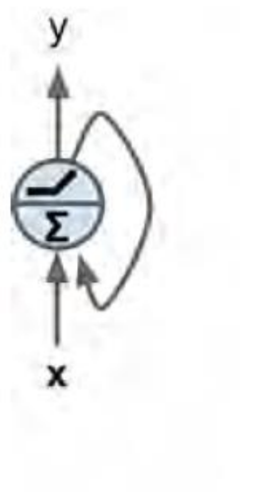
RNN with a layer of recurrent neurons



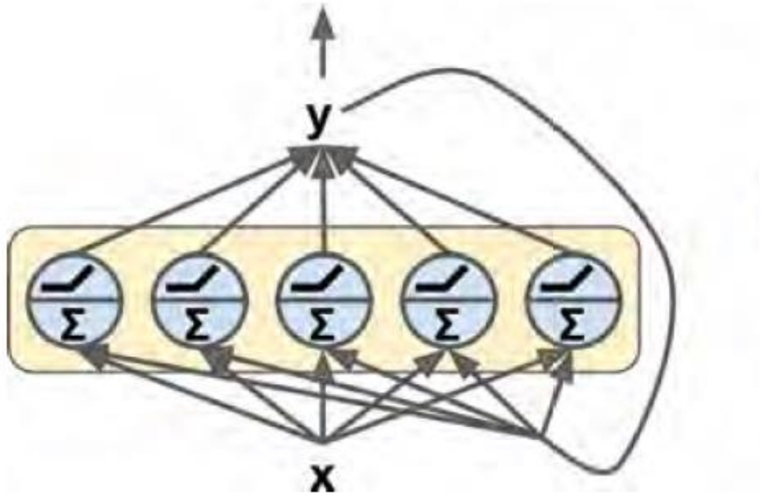
Unrolling the RNN through time

- You can easily create **a layer of recurrent neurons (=# hidden units)**
- At each time step *t*, every neuron receives both the input vector **x**(*t*) and the output vector from the previous time step **y**(*t*–1)
- Note that both the inputs and outputs are vectors now (when there was just a single neuron, the output was a scalar)

RNN in Keras

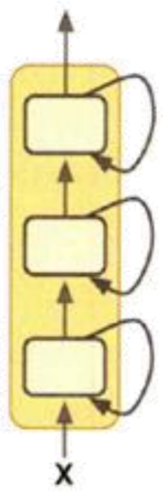


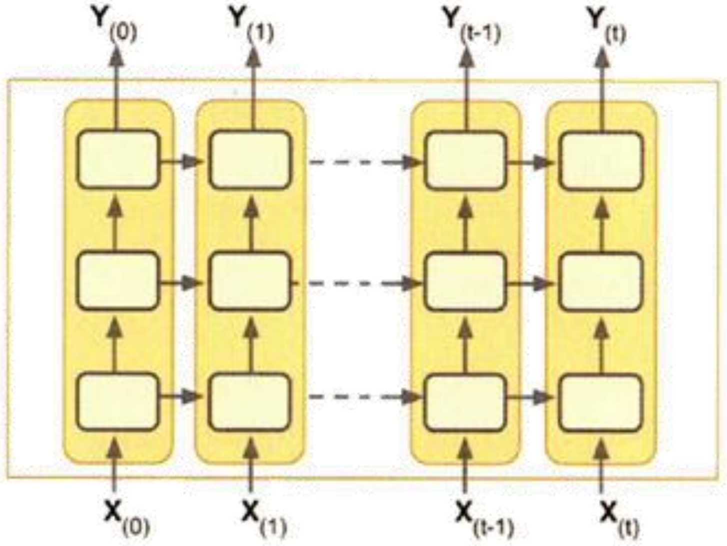
```
model = keras.models.Sequential([
        keras.Layers.SimpleRNN(1, input_shape=[None, 1])
])
```



```
model = keras.models.Sequential([
        keras.Layers.SimpleRNN(5, input_shape=[None, 1])
])
```

Recurrent neural network





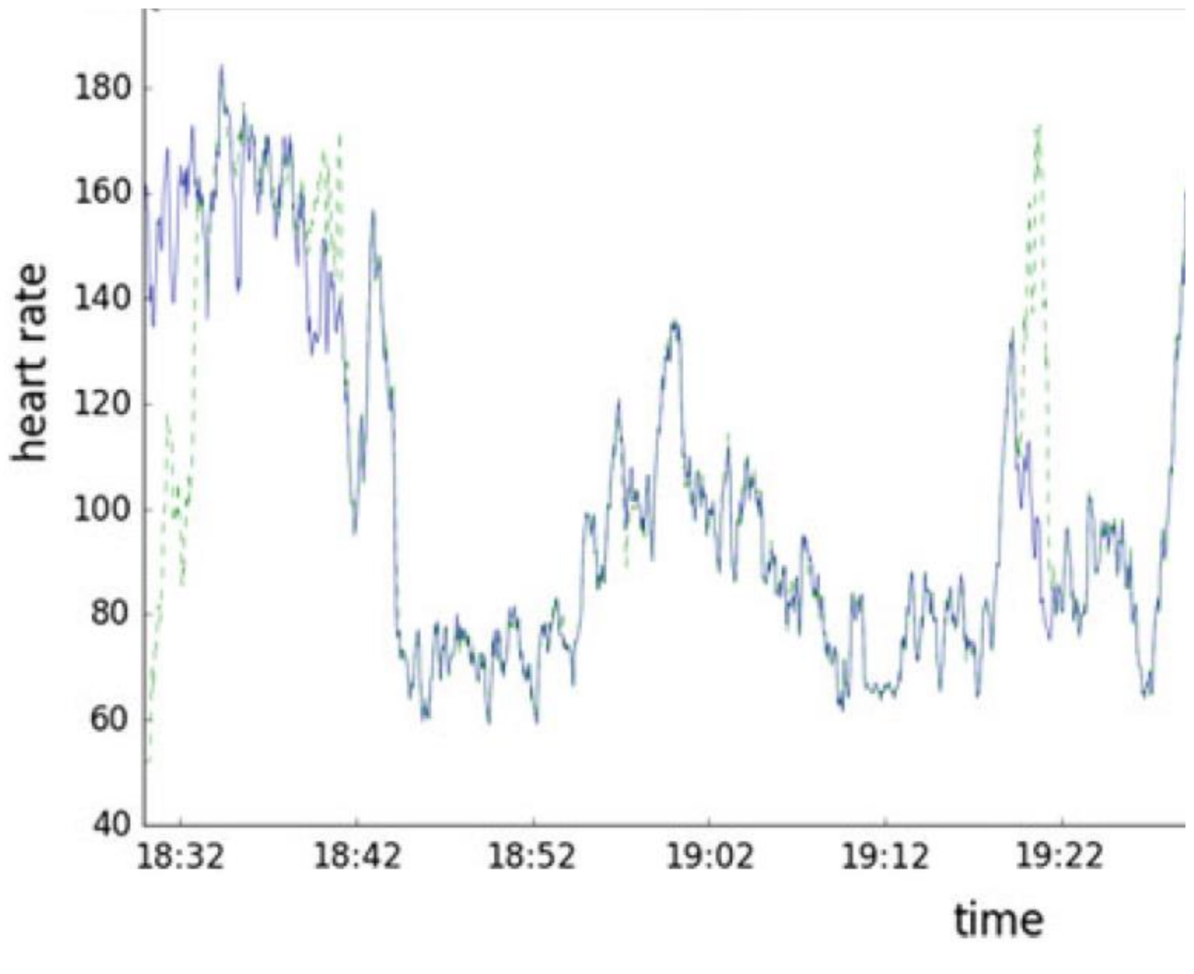
Deep RNN: stacking three layers of simple recurrent neurons

Unrolling the RNN through time

```
nodel = keras.models.Sequential([
        keras.layers.SimpleRNN(1, return_sequences=True, input_shape=[None, 1]),
        keras.layers SimpleRNN(1, return_sequences=True),
        keras.layers.SimpleRNN(1)
])
```

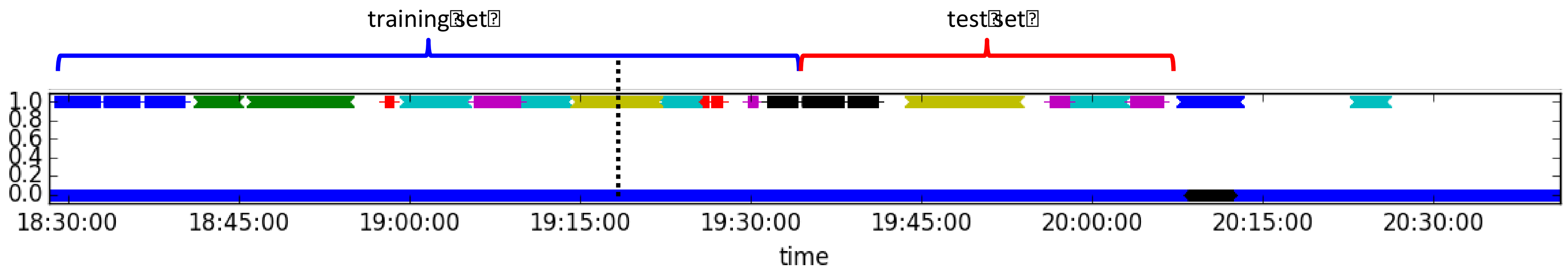
Case study

- CrowdSignals: Heart rate prediction?



Case study

- Recall from last time: how did we tune the parameters?
  - Cross validation
  - Does that make sense now?
  - Nope it does not…..



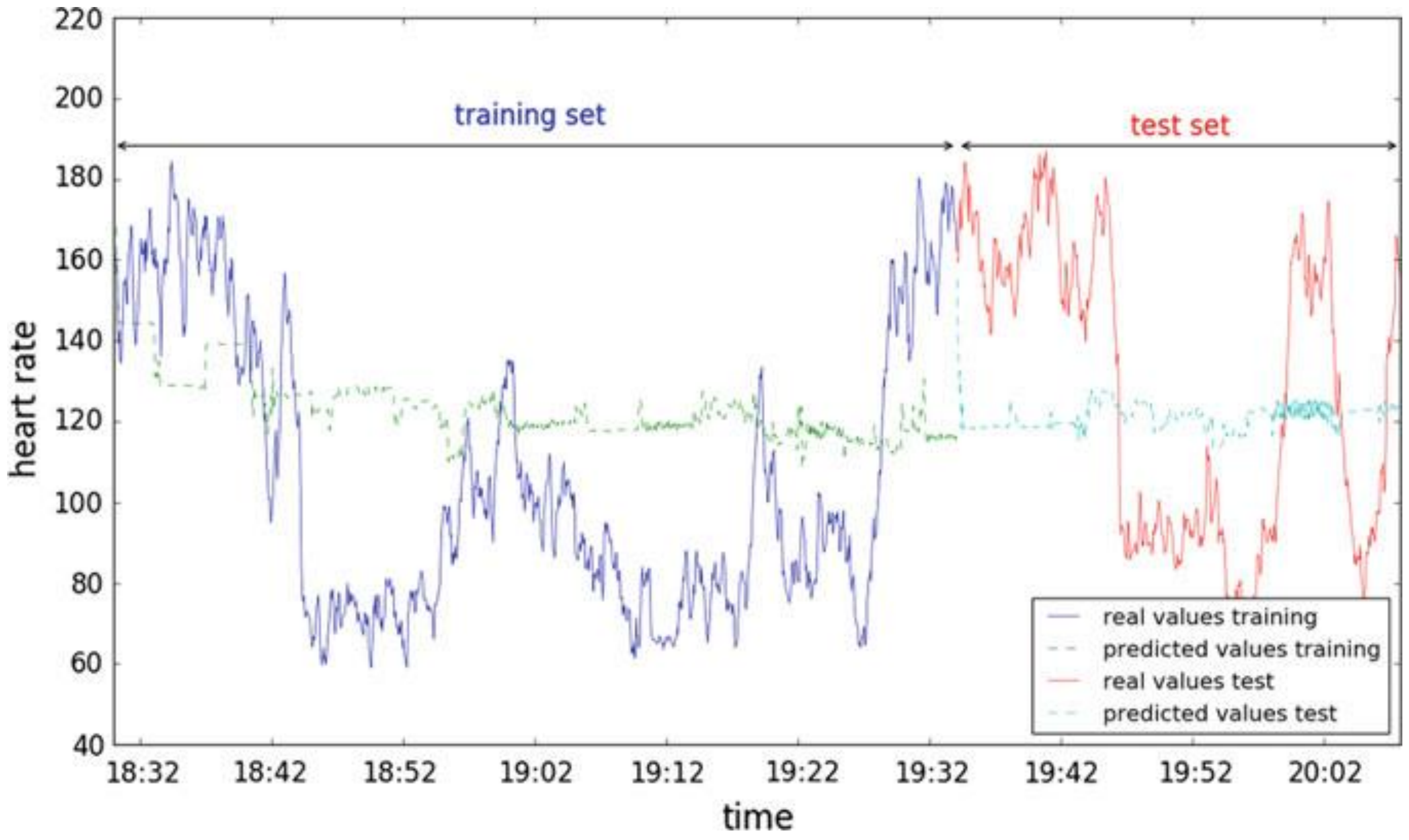
ARIMAX vs. RNN

- What techniques do we use?

Algorithm	Variant description	Parameters varied
Recurrent Neural Network (RNN)	Recurrent neural network with one layer of hidden neurons with a sigmoid activation function and sigmoid output nodes	number of hidden neurons: $\{50, 100\}$ maximum iterations over the entire dataset: $\{250, 500\}$
Time series	ARIMAX algorithm using Bayesian inference	p: $\{0, 1, 3, 5\}$ q: $\{0, 1, 3, 5\}$ d: $\{0, 1\}$

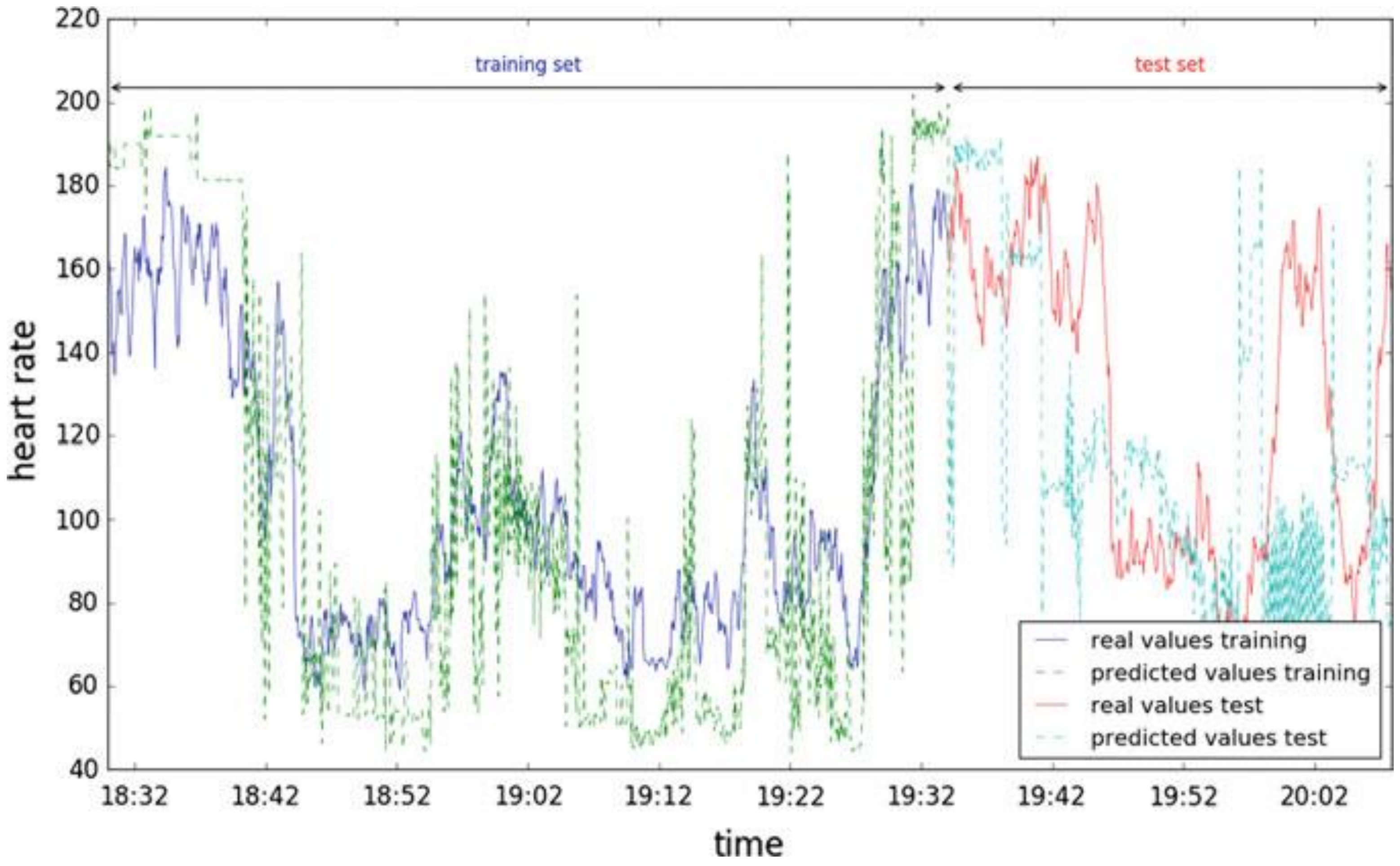
Results

- Actual versus predicted values for ARIMAX



Results

- Actual versus predicted values for RNN $\rightarrow$ better



Summary

- Time series data handling - must consider stationarity (possible to check it with autocorrelation)
- Filtering & smoothing
  - Moving averaging, exponential averaging
  - De-trending with differencing (previous steps or exponential averages)
- ARIMA (auto-regression + moving average of errors)
  - S-ARIMA (seasonal effects) + ARIMAX (explanatory variables: features)
- Recurrent neural networks consider temporal aspects
  - Diverse layer configurations are feasible (e.g., # hidden units, layers, dropouts, statefulness)