

# Time-series forecasting (with Python)

Execute track | version January 2021

# Attribution and copyright notice

This lecture is based on the following material available in the commons:

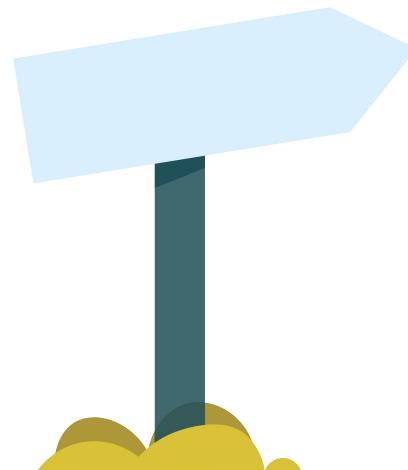
- [Forecasting: Principles and Practice \(2nd edition\)](#), by Rob J Hyndman and George Athanasopoulos (referenced as FPP2)
- The [user guide from the statsmodels library](#)
- The [user guide from the pmdarima library](#)
- The [sktime library](#) by the Alan Turing Institute
  - peer-reviewed journals ([Löning 2019](#), [Löning 2020](#))
  - repositories: [main project](#), [tutorial Pydata Amsterdam 2020](#)

© by [Daniel Kapitan](#), Time-series forecasting (with Python).

This work is licensed under a  
[Creative Commons Attribution-ShareAlike 4.0 International License](#).

# Learning objectives

- \_ Know how to distinguish different time-series learning tasks
- \_ Know how to formulate and frame a business problem to an appropriate time-series learning task
- \_ Know how to do time-series specific tasks within the CRISP-DM framework (decomposition, differencing, lagging, reduction)
- \_ Know how to use the most commonly used Python libraries for time-series forecasting



# Classical univariate forecasting

Given past observations:

$$y = (y(t_1) \dots y(t_T))$$

... learn a forecaster

$$\hat{y} = \hat{f}(h_j)$$

... that can predict:

$$\hat{y} = (\hat{y}(h_1) \dots \hat{y}(h_H))$$

... for the forecast horizon:

$$h_1 \dots h_H$$

# Time-series regression

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t$$

- forecast the time series of interest  $y$  assuming that it has a linear relationship with other time series  $x$
- note difference in type of forecast
  - ex-ante: use only information available until that time  $\rightarrow$  you need to forecast all  $x_t$ , too
  - ex-post: use information available later  $\rightarrow$  complete  $x_t$  is known

# Today we will just cover the basics

- \_ Decomposition
- \_ Exponential smoothing models (ETS)
- \_ Autoregressive Integrated Moving Average models (ARIMA)

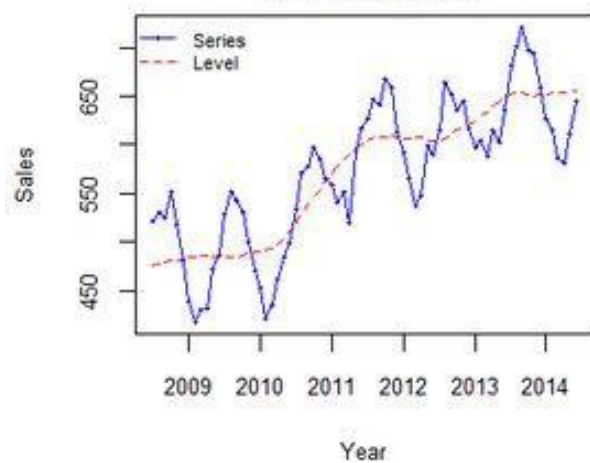
→ more complex models are often a combination of these models

# **Exponential smoothing (ETS models)**

# Decomposition: season, trend-cycle and remainder

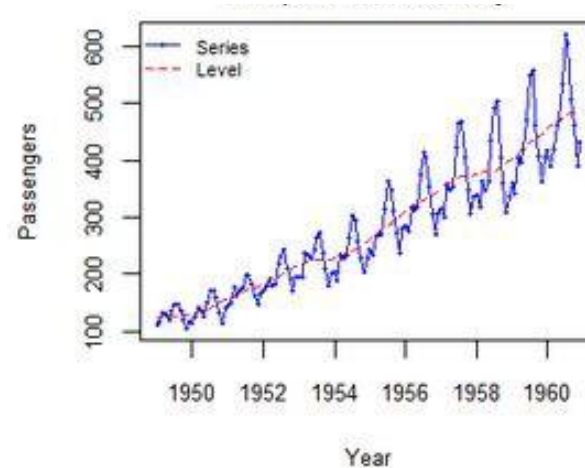
## Additive

$$y_t = S_t + T_t + R_t$$



## Multiplicative

$$y_t = S_t * T_t * R_t$$





# Decomposition methods: simple

`(statsmodels.tsa.seasonal.seasonal_decompose())`

- \_ Additive and multiplicative, widely used
- \_ Not recommended:
  - The estimate of the trend-cycle is unavailable for the first few and last few observations
  - The trend-cycle estimate tends to over-smooth rapid rises and falls in the data
  - Classical decomposition methods assume that the seasonal component repeats from year to year
  - Occasionally, the values of the time series in a small number of periods may be particularly unusual. For example, the monthly air passenger traffic may be affected by an industrial dispute, making the traffic during the dispute different from usual. The classical method is not robust to these kinds of unusual values.

# Decomposition methods: STL

(statsmodels.tsa.seasonal.STL())

- Advantages:

- any type of seasonality, not only monthly and quarterly data.
- seasonal component is allowed to change over time with rate of change parameter
- smoothness parameter for trend-cycle
- robust to outliers

- Limitations:

- does not handle trading day or calendar variation automatically
- only for additive decompositions

# Taxonomy ETS models




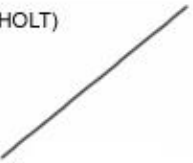

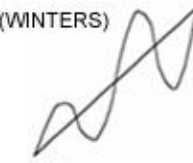






- Error = {A, M}
- Trend = {N, A,  $A_d$ }
- Seasonal = {N, A, M}

ETS(A, N, N): simple exponential smoothing with additive errors

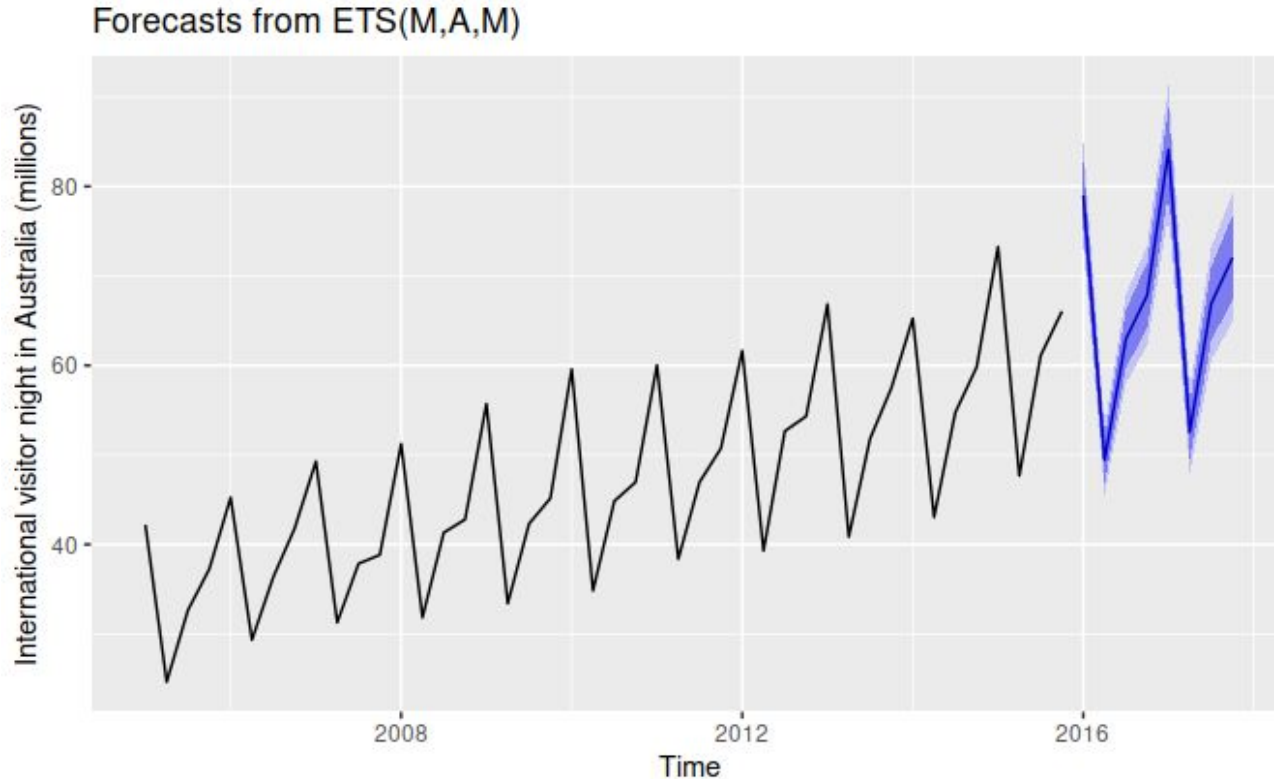
ETS(M, A, N): multiplicative errors, additive trend, multiplicative seasonal

Source: [FPP2, chapter 7](#)

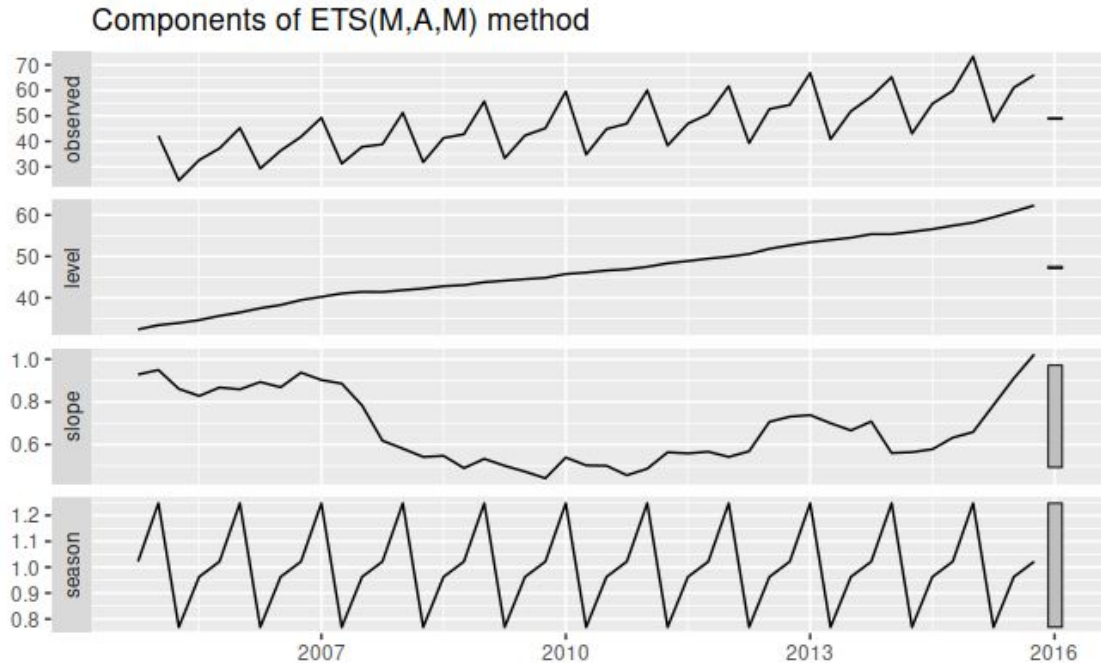
Figure: [Gardner \(1985\), Exponential Smoothing: the State of the Art](#)

	Nonseasonal	Additive Seasonal	Multiplicative Seasonal
Constant Level (SIMPLE)	 NN	 NA	 NM
Linear Trend (HOLT)	 LN	 LA	 LM
Damped Trend (0.95)	 DN	 DA	 DM
Exponential Trend (1.05)	 EN	 EA	 EM

# Example: forecasting visitors to Australia



# Example: forecasting visitors to Australia



# Using ETS models

## Advantages

- Information criteria can be used for model selection
  - Different information criteria, which are all a function of number of observations  $T$  and number of predictors  $k$
- Prediction intervals come for ‘free’
- Easy to explain to non-technical audience

## Disadvantages

- Forecasts will lag. In other words, the forecast will be behind, as the trend increases or decreases over time.
- Will fail to account for the dynamic changes at work in the real world, and the forecast will constantly require updating to respond new information.

# Facebook's Prophet library

- Conceptually similar to additive ETS, but with some key differences
  - Change points at which trends are allowed to change (kind of spline method)
  - Seasonality are estimated using partial Fourier sums
- Convenient library
  - Sensible defaults, e.g. with automatic detection of change points
  - Interpretable, e.g. [decomposition in yearly, weekly, holiday components](#)
- Limitations
  - Only univariate, although you can use this as the basis for more complex models
  - Only additive

# **Autoregressive Integrated Moving Average (ARIMA)**



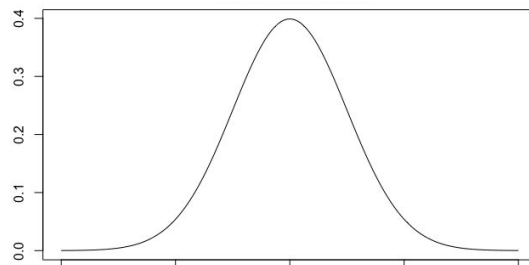
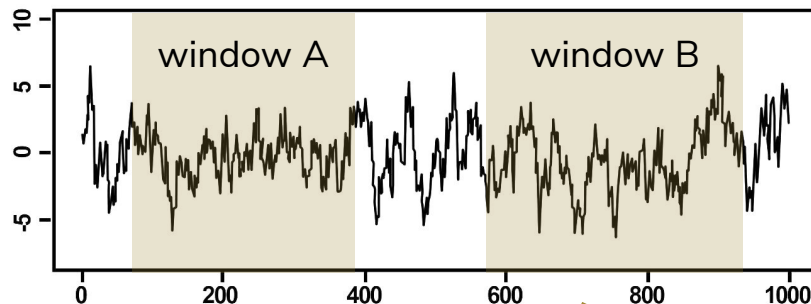
# Stationarity

A time-series  $\{y_t\}$  is **stationary**

if for all  $s$

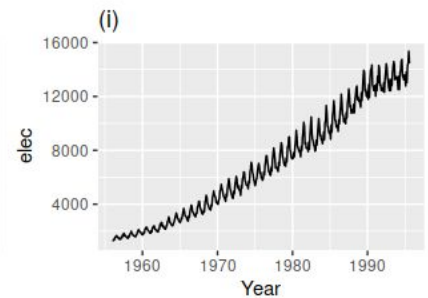
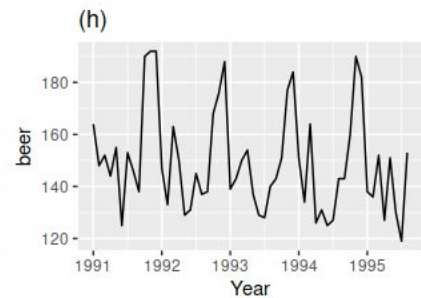
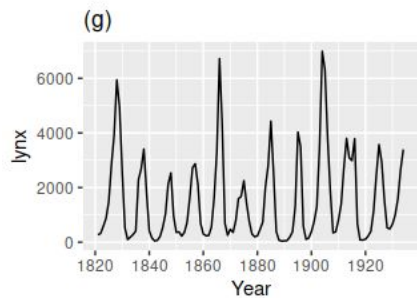
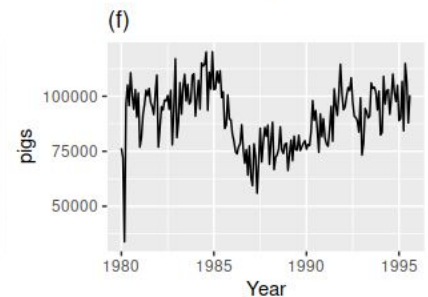
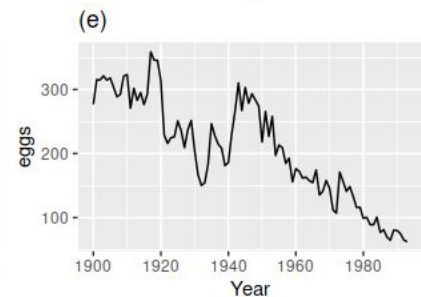
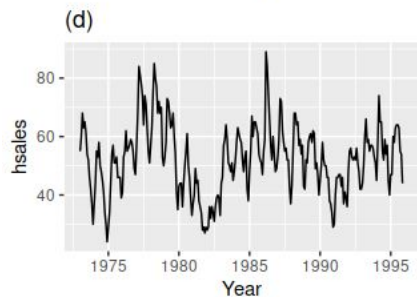
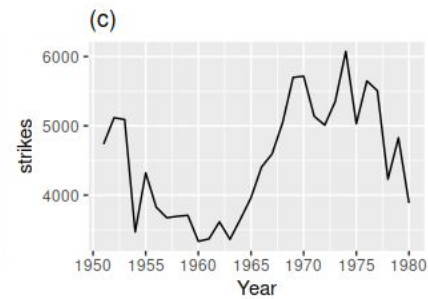
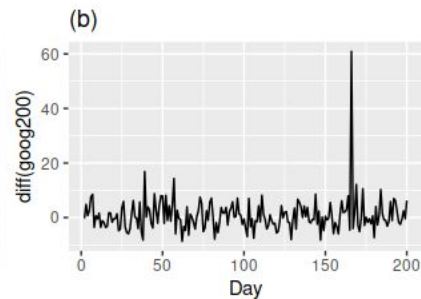
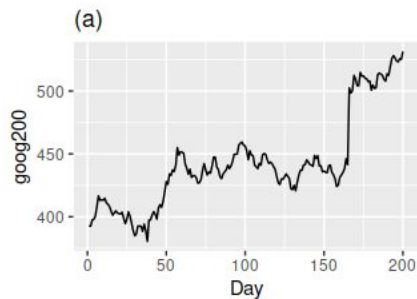
the distribution  $\{y_t, \dots, y_{t+s}\}$   
does not depend on  $t$

- no trends
- no seasonality
- no changing variance



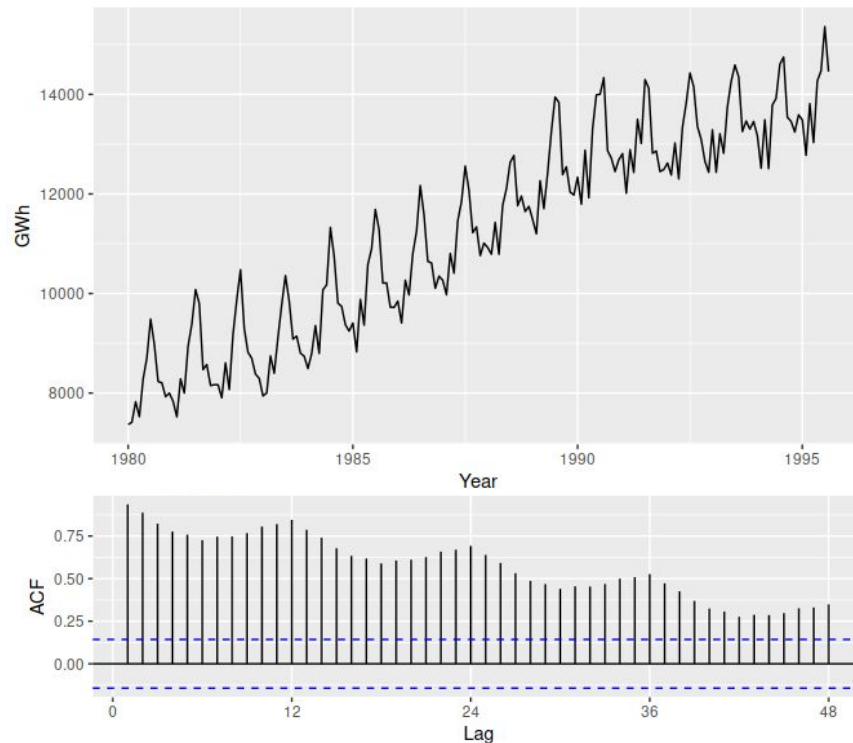
same distribution

# Which time-series are stationary?

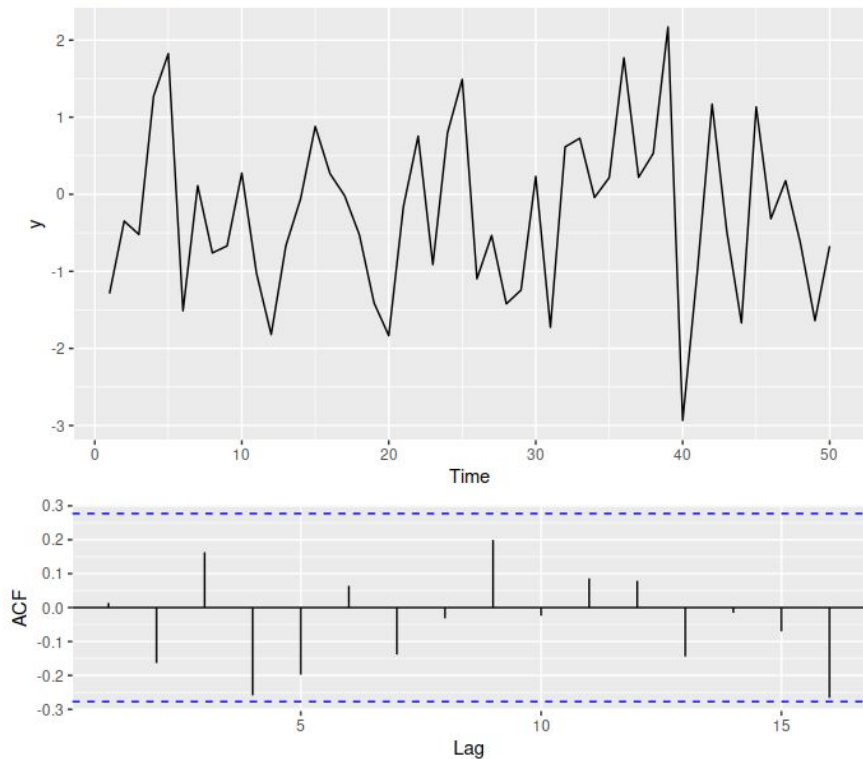


# Autocorrelation function (ACF)

- Correlation measures linear relationship between two variables
- Autocorrelation measures the linear relationship between lagged values of a time series
- Example: Australian electricity demand



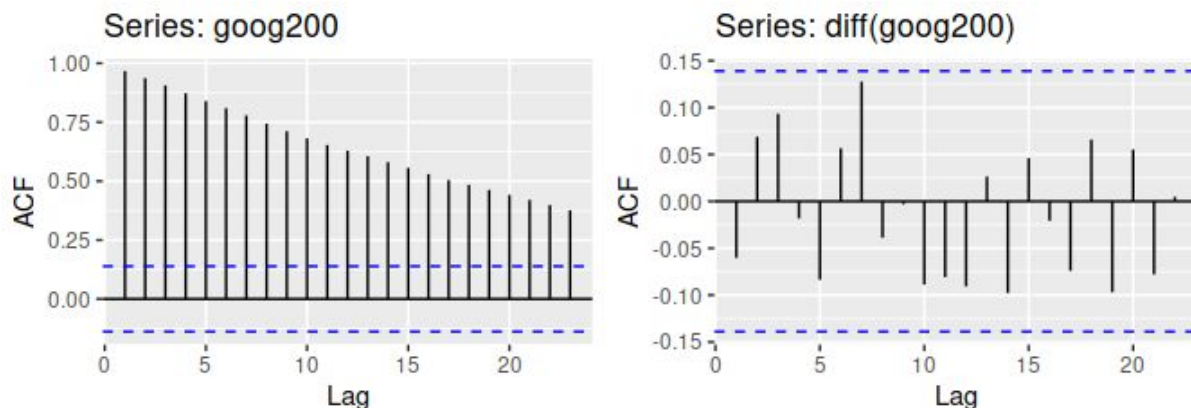
# Autocorrelation: white noise



source: [EPP2, paragraph 2.9](#)

- Autocorrelation is close to zero for white noise
- Close to zero =  $\pm 2/\sqrt{T}$  with  $T$  the length of the time series
- Common to plot these bounds on a graph of the ACF (the blue dashed lines above)
- $T = 50 \rightarrow \pm 2/\sqrt{50} = \pm 0.28$

# Differencing to make time-series stationary



- \_ Example: Google daily stockprice
- \_ Interpretation: daily change in the Google stock price is essentially a random amount which is uncorrelated with that of previous days
- \_ Sometimes second-order difference is required
- \_ Unit root test to assess stationarity with p-testing

# Non-seasonal ARIMA models

## ARIMA (p, d, q) models

---

AR(p)    order of autoregressive part

---

I(d)    degree of first differencing

---

MA(q)    order of moving average part

! moving average models  
≠ exponential smoothing

MA is used to make forecasts  
*exponential smoothing* estimates  
trend-cycle of past values

## special cases of ARIMA models

---

white noise    ARIMA(0, 0, 0)

---

random walk    ARIMA(0, 1, 0)  
with no constant

---

random walk  
with drift    ARIMA(0, 1, 0)  
with constant

---

autoregression    ARIMA(p, 0, 0)

---

moving average    ARIMA(0, 0, q)

---

# auto.arima (Hyndman-Khandakar algorithm)

## steps auto.arima

1. The number of differences  $0 \leq d \leq 2$  is determined using repeated KPSS tests.

2. The values of  $p$  and  $q$  are then chosen by minimising the AICc after differencing the data  $d$  times (stepwise search traversing model space)

Python implementation: [pmdarima](#)

a. Four initial models:

- ARIMA(0,  $d$ , 0)
- ARIMA(2,  $d$ , 2)
- ARIMA(1,  $d$ , 0)
- ARIMA(0,  $d$ , 1)

A constant is included unless  $d = 2$

If  $d \leq 1$ , an additional model is also fitted:

- ARIMA(0,  $d$ , 0) without a constant.

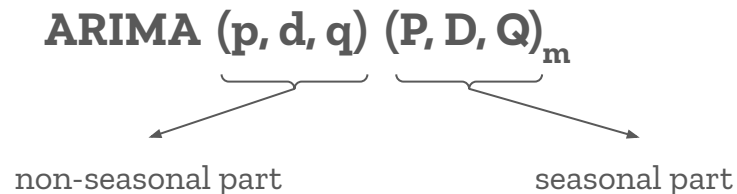
b. The best model (with the smallest AICc value) fitted in step (a) is set to be the “current model.”

c. Variations on the current model are considered:

- vary  $p$  and/or  $q$  from the current model by  $\pm 1$
- include/exclude  $c$  from the current model

d. Repeat c. until no lower AICc can be found

# Seasonal ARIMA models



The modelling procedure is almost the same as for non-seasonal data, except that we need to select seasonal AR and MA terms as well as the non-seasonal components of the model.



# Using ARIMA models (vs. ETS)

## Advantages

- Universal learner, you don't need to know underlying data generating process (cf. polynomial fitting)
- Can produce very accurate forecasts

## Disadvantages

- Harder to interpret c.q. communicate
- Only applicable to stationary time-series with no sudden 'jumps'

# **Beyond univariate time-series forecasting**

# More advanced time-series methods

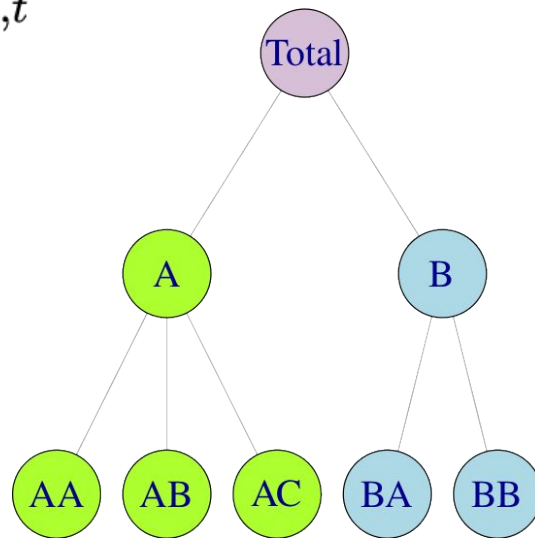
- [FPP2 chapter 9](#): Dynamic regression models
  - Combines ARIMA with time-series regression
- [FPP2 chapter 10](#): Hierarchical and grouped forecasting
- [FPP2 chapter 11](#):
  - Vector Autoregressions
  - Bagging and bootstrapping ETS
  - Using neural networks (deep learning) to tackle these tasks (see [sktime-dl](#), work in progress)

# Hierarchical forecasting

$$y_t = y_{AA,t} + y_{AB,t} + y_{AC,t} + y_{BA,t} + y_{BB,t}$$

$$y_{A,t} = y_{AA,t} + y_{AB,t} + y_{AC,t}$$

$$y_{B,t} = y_{BA,t} + y_{BB,t}$$



# Different approaches

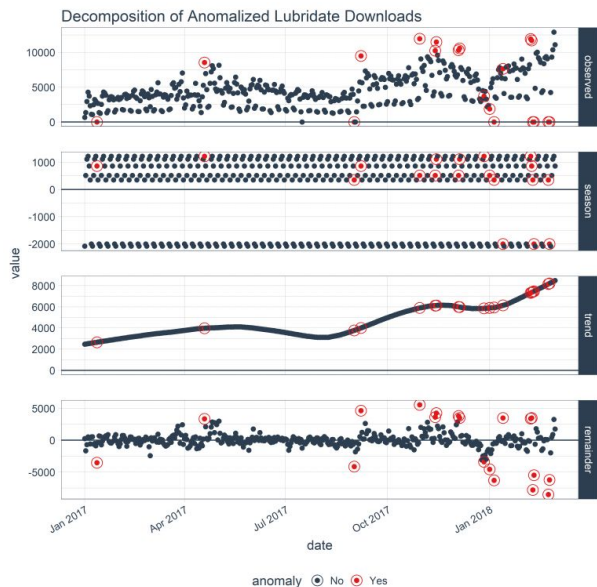
Approach	Advantages	Disadvantages
bottom-up	no loss of information	noisy and challenging to forecast
top-down	simplicity, disaggregation based on proportions	loss of information, bias
middle-out	<i>ibidem</i>	<i>ibidem</i>

## Python implementation:

- [scikit-hts](#)
- [scikit-hts-examples](#)

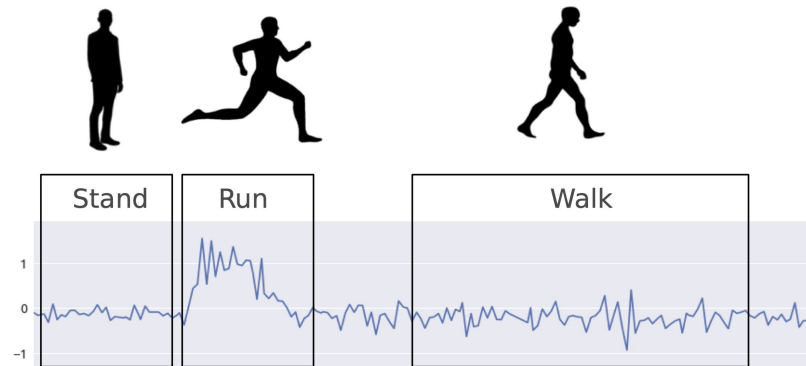
# Time-series annotation and classification

**Annotation:** detecting change points or anomalies



Source: ['Anomalize' is a R Package that Makes Anomaly Detection in Time Series Extremely Simple and Scalable](#)

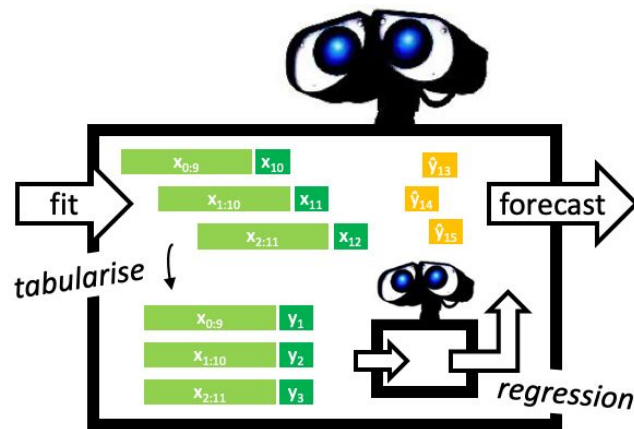
**Classification:** activity detection



Source: [Javier Ordóñez \(2017\)](#)

# Why use sktime?

- modular and compatible with scikit-learn, so that we can easily apply any scikit-learn regressor to solve our forecasting problem
- tuneable, allowing us to tune hyper-parameters like the window length or strategy to generate forecasts
- adaptive, in the sense that it adapts the scikit-learn's estimator interface to that of a forecaster, making sure that we can tune and properly evaluate our model

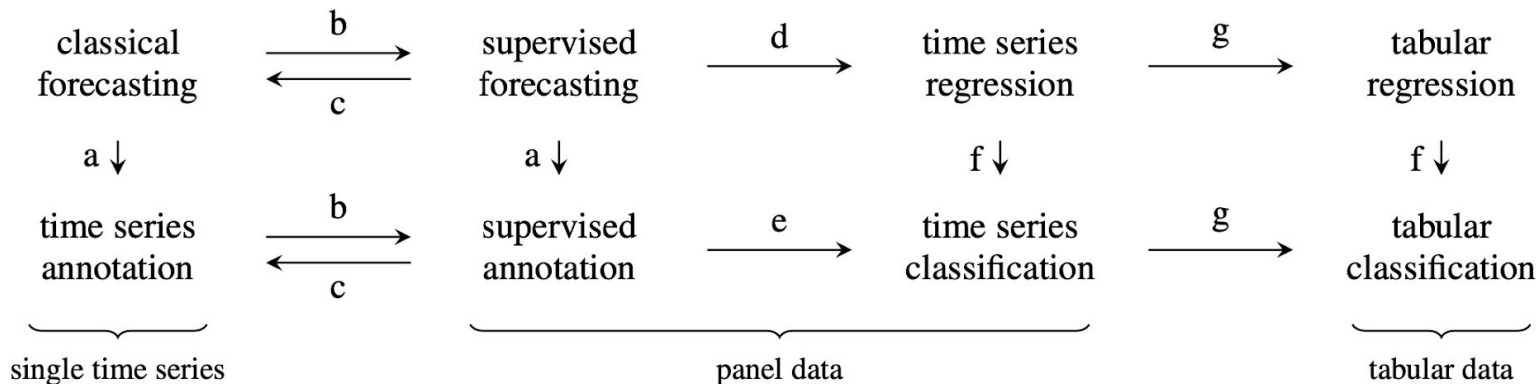


# Why use sktime?

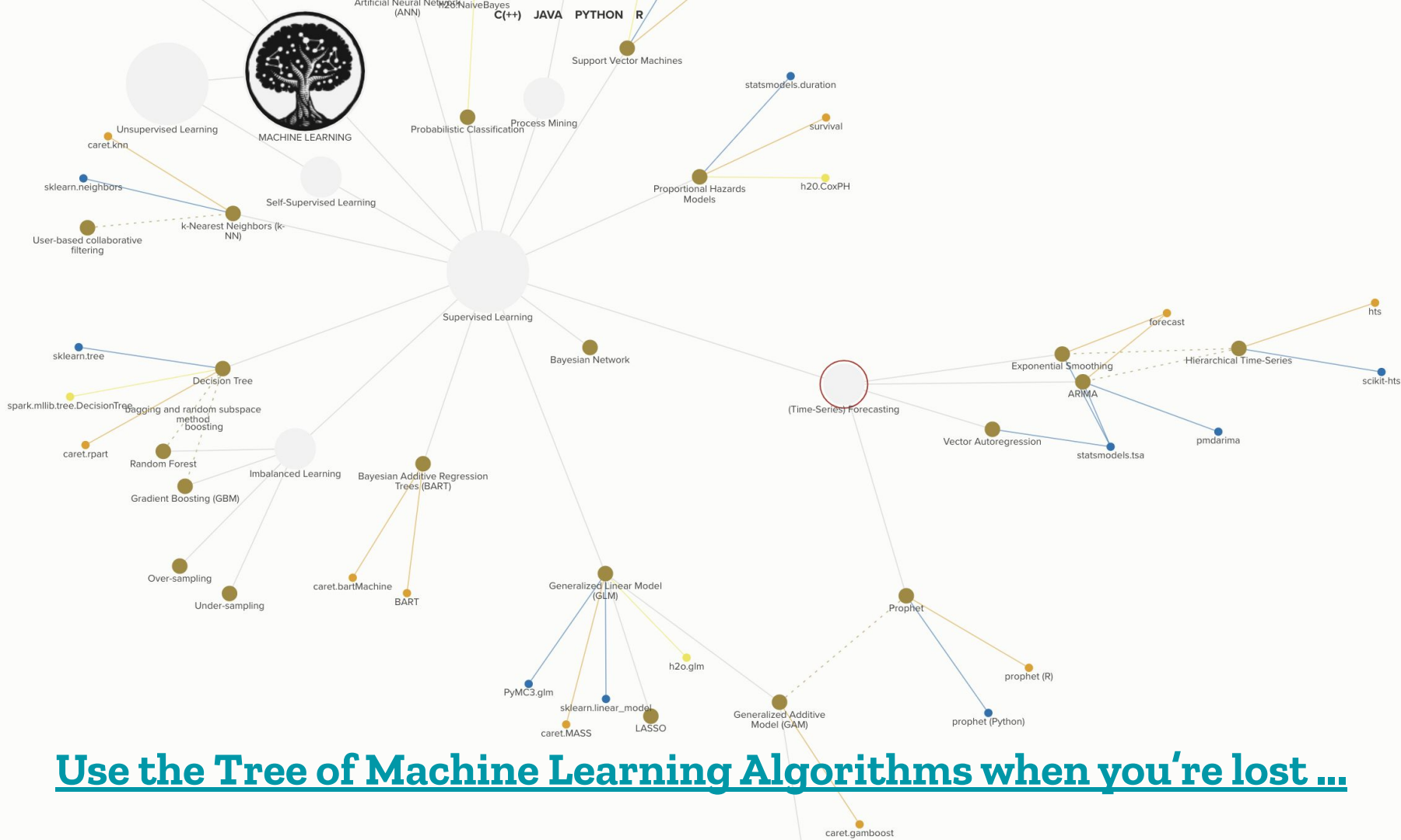
Pitfall	Solution
How to do model validation?	<code>temporal_train_test_split()</code>
How to apply regression algorithms?	functions and methods to simplify process of reduction
How to generate forecasts?	functions and methods for recursive multi-step ahead forecasting



# Through reduction, given time-series learning task can be formulated in different ways



*Notes:* (a) annotate time series with future values, (b) rolling window method to convert single series into panel data with multiple output time periods [12], (c) ignore training set (e.g. fit forecaster on test set only) or use training set for model selection, (d) iterate over output periods, optionally time binning/aggregation of output periods [12], (e) rolling window method to convert single series into panel data with single output period [23], (f) discretise output into one or more bins, (g) feature extraction [26, 19] or time binning/aggregation of input time points.



**Use the Tree of Machine Learning Algorithms when you're lost ...**

# Want to get your hands dirty?

- M-competitions on forecasting
  - [Original M4 competition \(2018\)](#) covered [61 methods](#). The data is also [available on Kaggle](#)
  - M5 (2020): hierarchical forecasting of Walmart sales, with separate competitions for [accuracy](#) and [uncertainty](#)
- UCI Human Activity Recognition with Smartphones
  - Time-series classification task
  - Available on [Kaggle](#)

# Recap: Learning objectives

- \_ Know how to distinguish different time-series learning tasks
- \_ Know how to formulate and frame a business problem to an appropriate time-series learning task
- \_ Know how to do time-series specific tasks within the CRISP-DM framework (decomposition, differencing, lagging, reduction)
- \_ Know how to use the most commonly used Python libraries for time-series forecasting

