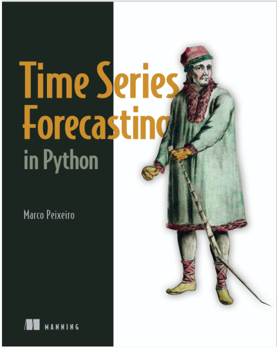
Advanced Data Analysis with Python

Cecilia Graiff

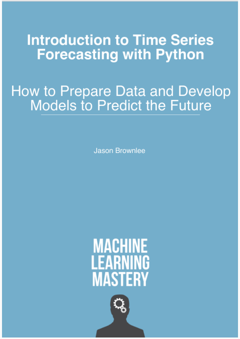
October 16, 2025

ALMAnaCH, Inria Paris - École d'Affaires Publiques, SciencesPo
cecilia.graiff@sciencespo.fr

Resources used for this class



Time Series Forecasting in Python



Introduction to Time Series Forecasting in Python

The slides greatly follow the structure of the first book. Some examples in these slides come from the execution of the code from the GitHub repository of Time Series Forecasting in Python by Marco Pexeiro, freely reusable and licensed under Apache License 2.0.

Homework Correction

An introduction to time series

What is a time series?

Definition (Time series)

A time series is a sequence of observations taken sequentially in time.

Source: **Time Series Analysis: Forecasting and Control**, page 1.

What is a time series?

Definition (Time series)

A time series is a sequence of observations taken sequentially in time.

Source: **Time Series Analysis: Forecasting and Control**, page 1.

Examples:

- Daily temperatures
- Stock market data
- Inflation rate
- Unemployment rate

What is a time series?

The data is equally spaced in time, meaning that it was recorded at every hour, minute, month, or quarter.

Source: Time Series Forecasting in Python.

Time series: Example

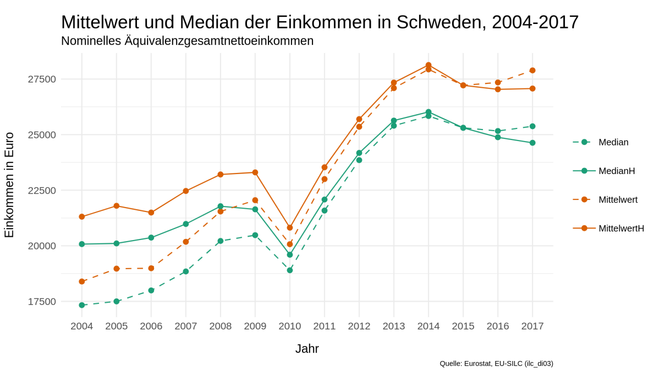


Figure 1: Mean and Median of earnings in Sweden from 2004 to 2017.

Source: Wikimedia Commons

Time series nomenclature

- $t - n$: A prior or **lag time** (e.g. $t - 1$ for the previous time).
- t: A **current time** and point of reference.
- $t + n$: A future or **forecast time** (e.g. $t + 1$ for the next time).

Time series nomenclature

- **Time series analysis**: descriptive
- **Forecasting**: predictive
- **Univariate** (single variable over time) vs **multivariate** (multiple variables over time)

Time series nomenclature

- **Time series analysis**: descriptive
- **Forecasting**: predictive
- **Univariate** (single variable over time) vs **multivariate** (multiple variables over time)

Multivariate more complex to model; in this lecture, only an introduction to time series concepts and naive models for **univariate** time series.

Time series nomenclature

- **Time series analysis**: descriptive
- **Forecasting**: predictive
- **Univariate** (single variable over time) vs **multivariate** (multiple variables over time)

> **Multivariate** more complex to model; in this lecture, only an introduction to time series concepts and naive models for **univariate** time series.

Examples:

- **Describe** the changes in the earning of Johnson & Johnson from 1960 to 1980
- Use the data about earnings of Johnson & Johnson from 1960 to 1970 to **predict** the earnings from 1970 to 1980

Components of Time Series

- **Observed**: The observed value.

Components of Time Series

- **Observed**: The observed value.
- **Trend**: The optional increasing or decreasing behavior of the series over time (often linear).

Components of Time Series

- **Observed**: The observed value.
- **Trend**: The optional increasing or decreasing behavior of the series over time (often linear).
- **Seasonality**: The optional repeating patterns or cycles of behavior over time.

Components of Time Series

- **Observed**: The observed value.
- **Trend**: The optional increasing or decreasing behavior of the series over time (often linear).
- **Seasonality**: The optional repeating patterns or cycles of behavior over time.
- **Noise**: The optional variability in the observations that cannot be explained by the model.

Decomposition: Example

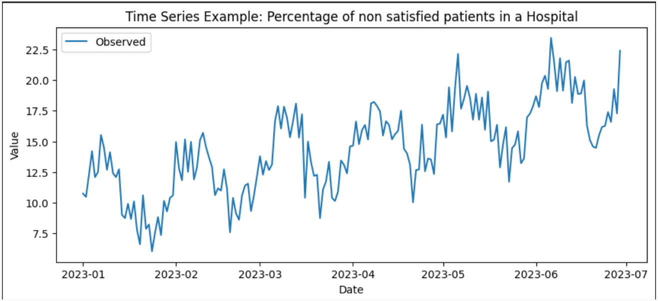


Figure 2: A time series generated with Python and random numbers, not corresponding to reality.

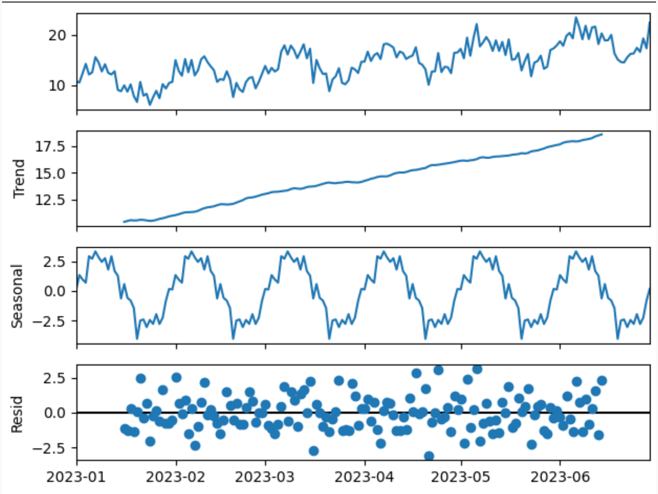


Figure 3: Decomposition of the time series from the previous slide.

Baseline models for time series

Baseline: Naive Prediction

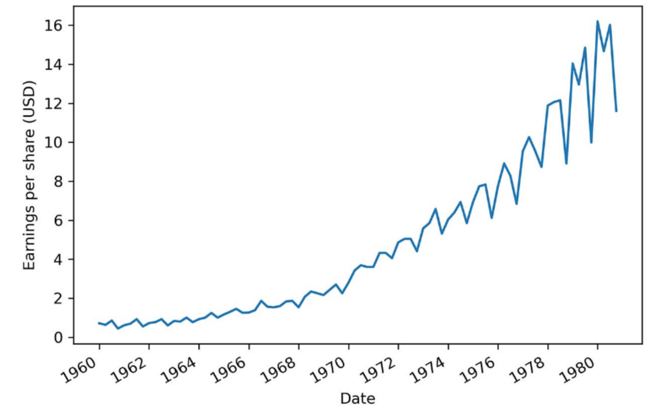


Figure 4: Quarterly earnings of Johnson & Johnson in USD from 1960 to 1980 showing a positive trend and a cyclical behavior. **Task: Use the data from 1960 to 1979 to predict the earnings of 1980.**

Source: Time Series Forecasting in Python.

Baseline model

Definition

A **baseline model** a simple solution based on basic statistics.

- Typically, a **naive prediction** will not be accurate enough to reflect real world data
- Used for explorative purposes
- Used as evaluation for more complex models
- In some cases (and with many caveats), it can be used for forecasting. **If you plan to do so, beware of the possible limitations.**

The process explained in this section comes from the execution of the code from the GitHub repository of Time Series Forecasting in Python by Marco Pexeiro, freely reusable and licensed under Apache License 2.0.

Naive prediction methods

- Historical mean
- Last mean
- Last known value
- Last season

Historical Mean

Definition

Forecasting the **historical mean** involves using the arithmetical mean of past values as prediction.

Historical Mean

Definition

Forecasting the **historical mean** involves using the arithmetical mean of past values as prediction.

- Calculate mean of training set

Historical Mean

Definition

Forecasting the **historical mean** involves using the arithmetical mean of past values as prediction.

- Calculate mean of training set
- Use it as prediction for the test set

Evaluating Historical Mean Baseline

- Evaluation metric: **Mean Absolute Percentage Error (MAPE)**

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right| \times 100$$

  where

  - A_i is the actual value at time i
  - F_i is the forecasted value at time i

- Percentage of how much the forecast values deviate from the observed or actual values on average.
- Higher MAPE = worse performance

Example

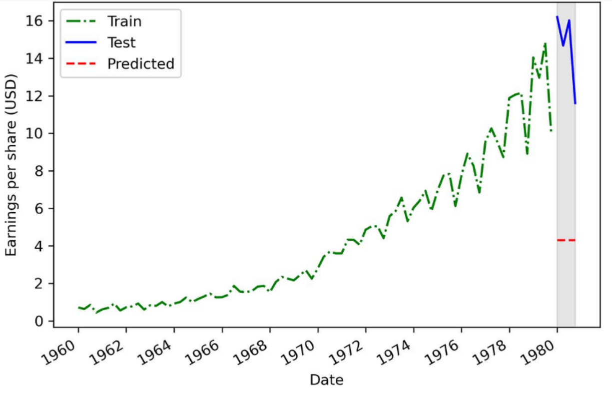


Figure 5: Historical mean as baseline is not working well in this case: this model had a MAPE of 70%.

Graph taken from the GitHub repository of Time Series Forecasting in Python by Marco Pexeiro, freely reusable and licensed under Apache License 2.0.

Baseline model: last mean

What to do if the historical mean does not yield accurate predictions?

Definition

Forecasting the **last mean** involves using the arithmetical mean of the last values as prediction.

Baseline model: last mean

What to do if the historical mean does not yield accurate predictions?

Definition

Forecasting the **last mean** involves using the arithmetical mean of the last values as prediction.

- Calculate mean of training set on the last year

Baseline model: last mean

What to do if the historical mean does not yield accurate predictions?

Definition

Forecasting the **last mean** involves using the arithmetical mean of the last values as prediction.

- Calculate mean of training set on the last year
- Use it as prediction for the test set

Example

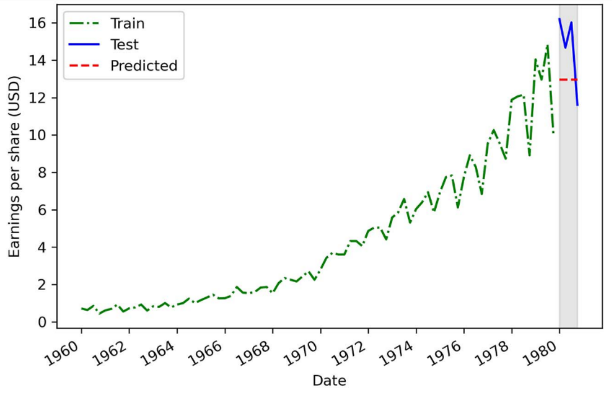


Figure 6: Using the mean on 1979 works better: here, MAPE is 15.6%.

Graph taken from the GitHub repository of Time Series Forecasting in Python by Marco Pexeiro, freely reusable and licensed under Apache License 2.0.

These results tell us that the values for 1980 depends on values that are in the past, but **not too far back in the past.**

Last known value

Hypothesis: Predictions based on the **last known value** will be even more accurate than the ones based on **last year's mean**.

Example

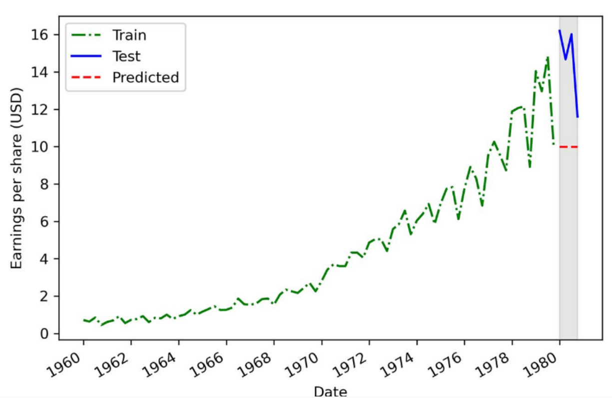


Figure 7: This model has a MAPE of 30.45%, meaning that our hypothesis was incorrect: predicting based on the last value yields **less accurate results** than predicting based on last year's mean.

Graph taken from the GitHub repository of Time Series Forecasting in Python by Marco Pexeiro, freely reusable and licensed under Apache License 2.0.

Why is this result worse?

Cyclical behaviour of the data: high during the first three quarters and then falls at the last quarter.

Idea: let's take into account **seasonality**

Seasonal forecasting

Definition

The naive seasonal forecast takes the last observed cycle and repeats it into the future.

Example

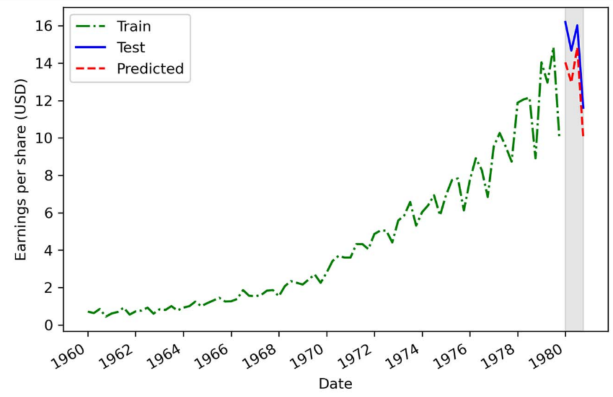


Figure 8: Seasonal forecasting yields the best results on this dataset, with a MAPE of 11.56%.

Graph taken from the GitHub repository of Time Series Forecasting in Python by Marco Pexeiro, freely reusable and licensed under Apache License 2.0.

Random Walks, Stationarity, Autocorrelation

Random Walk

A **Random Walk** in which there is an equal chance of going up or down by a random number.

$$X_t = C + X_{t-1} + \epsilon_t$$

where

- C is a constant
- X_t = value at time t, X_{t-1} value at present timestep $t-1$
- ϵ_t = random error term (**white noise**)

Random Walk: Example

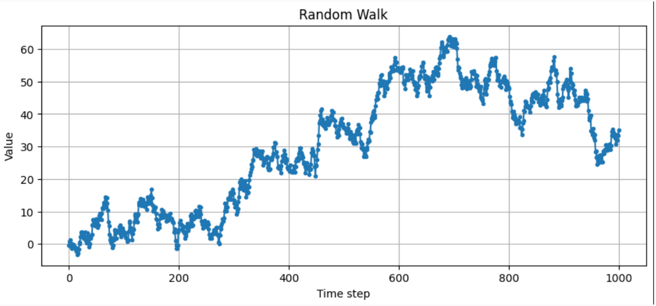


Figure 9: Example random walk generated automatically with Python.

Random Walk: Example

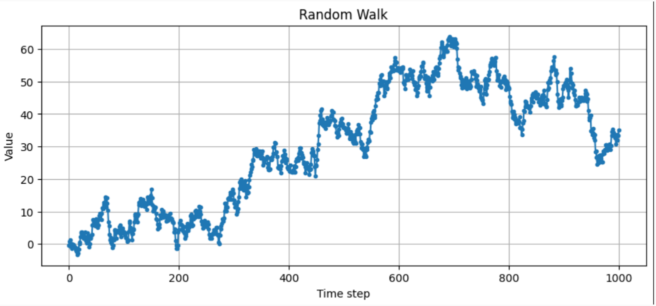


Figure 9: Example random walk generated automatically with Python.

- No apparent logic
- Abrupt changes

In a random walk:

- The process is highly time-dependant (=**stationary**)
- Each variable depends highly on the previous one (=**autocorrelation**)

Intuitively, it is impossible to statistically predict something that is **completely random**.

Stationarity

A time series is stationary if its statistical properties (mean, variance, autocorrelation) do not change over time. This means that those properties are **independent from time**.

Stationarity

A time series is stationary if its statistical properties (mean, variance, autocorrelation) do not change over time. This means that those properties are **independent from time**.

- Many models require stationarity (all the models we will see in this class)
- Therefore, a **transformation** is necessary

Transformations

A transformation is a mathematical operation applied to a time series in order to make it stationary.

- **Differencing**: calculating change from one timestep to another. Useful for stabilizing the mean.
- **Applying a log function** to the series: useful for stabilizing variance.

Differencing

Time t	**Original** y_t	**Differenced** $\Delta y_t = y_t - y_{t-1}$
$t-2$	2.0	-
$t-1$	2.5	2.5 - 2.0 $=$ 0.5
t	3.0	3.0 - 2.5 $=$ 0.5
$t+1$	3.5	3.5 - 3.0 $=$ 0.5
$t+2$	3.7	3.7 - 3.5 $=$ 0.2

Explanation: Each differenced value shows the change from the previous time point. After differencing, the series is **stationary**.

Important: You lose the first data point, as it cannot be differentiated.

Differencing: Example

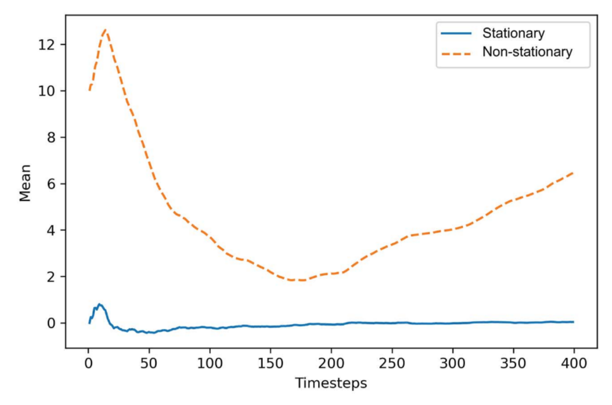


Figure 10: The mean of the stationary process becomes constant after the first few timesteps, while the mean of a non-stationary process changes with time.

Augmented Dickey-Fuller (ADF) Test

- **Goal:** Test for non-stationarity
- **Hypotheses:**
  - H_0: Series is non-stationary
  - H_1: Series is stationary
- **Procedure:** Regress Δy_t on lagged y_{t-1} and lagged differences $\Delta y_{t-1}, ..., \Delta y_{t-p}$
- **Interpretation:**
  - ADF statistic $<$ critical value $\Rightarrow$ reject H_0 (stationary)
  - ADF statistic $\geq$ critical value $\Rightarrow$ fail to reject H_0 (non-stationary)

We will not delve into the way **critical values** are computed; Python's package `statsmodels` automatically reports the values alongside the result of ADF.

Autocorrelation

Reminder

We discussed **correlation** in the previous lectures as a measure of relation between two variables.

- **Autocorrelation** measures the linear relationship between lagged values of a time series.
  - "How the correlation between any two values change as the lag increases?"
  - **lag** = number of timesteps separating two values (t, $t-1$, ...)

Autocorrelation Function (ACF)

Definition: The ACF measures the linear relationship between lagged values of a time series.

$$\rho_k = \frac{\text{Cov}(y_t, y_{t-k})}{\sigma(y_t)\sigma(y_{t-k})}$$

Key Points:

- Autocorrelation at time k
- σ is the standard deviation
- Useful to identify moving average (MA) patterns.
- Plot ACF to see how correlations change over time.

Partial Autocorrelation Function (PACF)

Definition: PACF measures correlation between y_t and y_{t-k} after removing the effects of intermediate lags $1, 2, \ldots, k-1$.

Key Points:

- Helps identify autoregressive (AR) order.
- PACF cuts off after lag p in an AR(p) process.
- Plot PACF to detect direct relationships at specific lags.

Random Walk Pipeline

✓ Plot dataset

Readapted from Time Series Forecasting in Python by Marco Pexeiro.

Random Walk Pipeline

- ✓ Plot dataset
- ✓ Check stationarity

Readapted from Time Series Forecasting in Python by Marco Pexeiro.

Random Walk Pipeline

- ✓ Plot dataset
- ✓ Check stationarity
  - ✓ How: with ADF test
  - ✓ Eventually: differentiate

Readapted from Time Series Forecasting in Python by Marco Pexeiro.

Random Walk Pipeline

- ✓ Plot dataset
- ✓ Check stationarity
  - ✓ How: with ADF test
  - ✓ Eventually: differentiate
- ✓ Check autocorrelation

Readapted from Time Series Forecasting in Python by Marco Pexeiro.

Random Walk Pipeline

- ✓ Plot dataset
- ✓ Check stationarity
  - ✓ How: with ADF test
  - ✓ Eventually: differentiate
- ✓ Check autocorrelation
  - ✓ How: by plotting ACF

Readapted from Time Series Forecasting in Python by Marco Pexeiro.

Random Walk Pipeline

- ✓ Plot dataset
- ✓ Check stationarity
  - ✓ How: with ADF test
  - ✓ Eventually: differentiate
- ✓ Check autocorrelation
  - ✓ How: by plotting ACF
- ✓ If autocorrelation: not a random walk, else: random walk

Readapted from Time Series Forecasting in Python by Marco Pexeiro.

Forecasting a Random Walk

Because a **random walk** does not exhibits trends or patterns, but random changes, it is impossible to forecast it with a statistical model. Hence, we use **naive methods**.

Therefore, the best solution is to **predict the next timestep of a time series**. **(Beware that this depends from the use case)**

Forecasting the next timestep of a random walk

Suppose we have data for the new hiring of a company every 5 years from 2005 to 2025:

Time t	**Observed** T_t	**Forecast** T_{t+1}
2005	67	-
2010	78	67
2015	92	78
2020	198	92
2025	201	198

Explanation: Each forecast T_{t+1} uses the current observed value T_t.

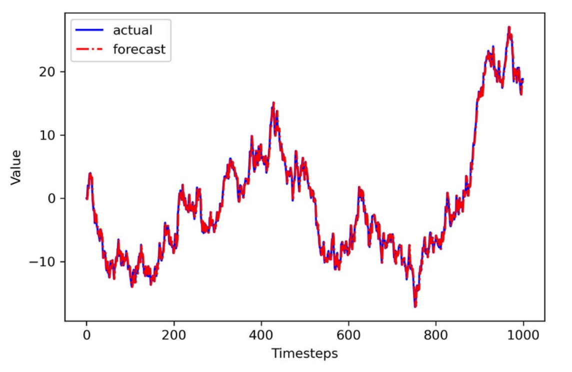


Figure 11: Forecasted vs actual values.

Things are not as amazing as they might seem...

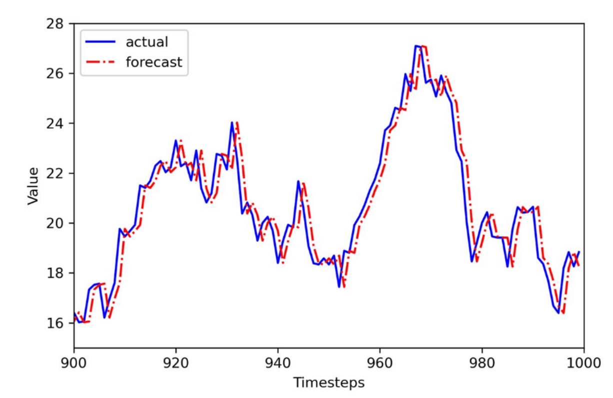


Figure 12: Close-up on the last 100 timesteps.

White Noise

- We saw an error term in the previous functions: it is **white noise**.
- Several statistical tests build on white noise
- One of the most frequent ones for the models we do in this course: **Gaussian white noise**, a sequence of uncorrelated random variables with mean zero and constant variance:

$$\epsilon_t \sim \text{i.i.d. } N(0, \sigma^2)$$

Statistical modeling of time series: A big picture

Moving Average (MA) model

Definition (Moving average model)

In a **moving average (MA) model**, the current value depends linearly on **the current error term and the past error terms**.

- Denoted as **MA(q)**, where q is the order $(MA(1), MA(2), ...)$
- Expressed as follows:

$$y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

$$y_t = \mu + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q}$$

Where:

- y_t is the value of time series at time t
- μ is a constant or the mean of the time series
- $\varepsilon_t, \varepsilon_{t-1}, \varepsilon_{t-2}, \varepsilon_{t-q}$ are the white noise terms associated with the time series at time $t, t-1, t-2, ..., t-q$
- $\theta_1, \theta_2, ..., \theta_q$ are the moving average constants.

Autoregressive model

Definition (Autoregressive model)

An autoregressive model in time series analysis is a **linear regression** of a variable against its past values.

- Denoted as $AR(p)$, where p is the order.

$$y_t = C + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

$$y_t = C + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

Where:

- y_t is the value at time t.
- C is a constant.
- $\phi_1, \phi_2, ..., \phi_p$ are the model parameters.
- $y_{t-1}, y_{t-2}, ..., y_{t-p}$ are the lagged values
- ε_t is the white noise at time t.

AR model pipeline

- For each **lag**, we seek its **autocorrelation**
  - **High autocorrelation** = strong dependence, **low autocorrelation** = weak dependence
- Useful: **plot autocorrelation function**
- Goal: understand temporal structure and **decide best lag value**

MA vs AR models

Autoregressive (AR) Model

Current value depends on past values.

Use when:

- Series shows persistence
- PACF cuts off after lag p
- Example: daily sales, temperature

Moving Average (MA) Model

Current value depends on past errors.

Use when:

- Series mostly random, but affected by recent shocks
- ACF cuts off after lag q
- Example: demand spikes due to promotions

Explore your data, visualize patterns, and determine if **past values seem to matter** (AR) or if **recent surprises seem to matter** (MA).

What if the series shows both persistance **and** influence of recent errors?

ARMA model

Definition (Autoregressive Moving Average Model)

The autoregressive moving average (ARMA) model combines autoregression (AR) and moving average (MA) and is applied to time series analysis and forecasting.

$$y_t = C + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

Where:

- y_t is the value at time t.
- C is a constant.
- $\phi_1, \phi_2, ..., \phi_p$ are the model parameters.
- $y_{t-1}, y_{t-2}, ..., y_{t-p}$ are the lagged values
- ε_{t-j}, ε_t are the white noise at time t and $t-j$.

Key take-aways

- Principles of time series (a very fascinating but complex field!): components and basic theory, intuitive understanding
- How to fit a naive model
- How and when to transform a time series
- How and when to apply one of the studied models

For the final project

- If you want to analyze **temporal data**, this is the way to go!
- **Choose accordingly**: Time series is a big field, and some models are complicated to implement. Choose a dataset that allows you to perform a naive or simpler analysis, and **be aware of eventual limitations**
- If the topic interests you, you are welcome to deepen it and use more complex models, but **explain it in the project description and check in with me about it**.

Reminder: project description deadline on November 3!