✨ Member-only story

Interpreting ACF and PACF Plots for Time Series Forecasting

How to determine the order of AR and MA models

Leonie Monigatti · Follow
Published in Towards Data Science
11 min read · Aug 3, 2022

Listen Share ••• More

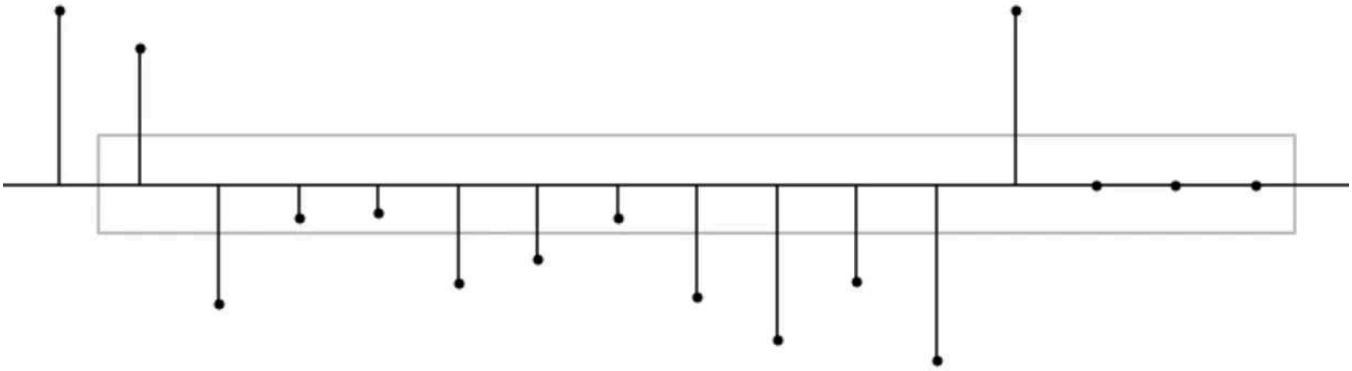


Image by the author via Kaggle

Autocorrelation analysis is an important step in the Exploratory Data Analysis of time series forecasting. **The autocorrelation analysis helps detect patterns and check for randomness.** It's especially important when you intend to use an autoregressive–moving-average (ARMA) model for forecasting because it helps to determine its parameters. The analysis involves looking at the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots.

> This article helps you build an intuition for interpreting ACF and PACF plots.

This article helps you build an intuition for interpreting these ACF and PACF plots. We'll briefly go over the fundamentals of the ACF and PACF. However, as the focus lies in the **interpretation** of the plots, a detailed discussion of the underlying mathematics is beyond the scope of this article. We'll refer to other resources instead.

This article is a revisited version of my Kaggle Notebook, which was originally published in December 2021. You can download or fork the code there.

Fundamentals

The ACF and PACF plots are used to figure out the order of AR, MA, and ARMA models. In this section, we'll only briefly touch on the relevant terms. For detailed explanations, we'll refer to other resources.

Auto-Regressive and Moving Average Models

Auto-Regressive Model

The Auto-Regressive (AR) model assumes that the current value (y_t) is **dependent on previous values** (y_(t-1), y_(t-2), ...). Because of this assumption, we can build a **linear** regression model.

$$\hat{y}_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p}$$

To figure out the **order of an AR model**, you need to **look at the PACF**.

Moving Average Model

The Moving Average (MA) model assumes that the current value (y_t) is **dependent on the error terms** including the current error (ϵ_t, ϵ_(t-1),...). Because error terms are random, there's **no linear** relationship between the current value and the error terms.

$$\hat{y}_t = \epsilon_t + \beta_1 \epsilon_{t-1} + \cdots + \beta_q \epsilon_{t-q}$$

To figure out the **order of an MA model**, you need to **look at the ACF**.

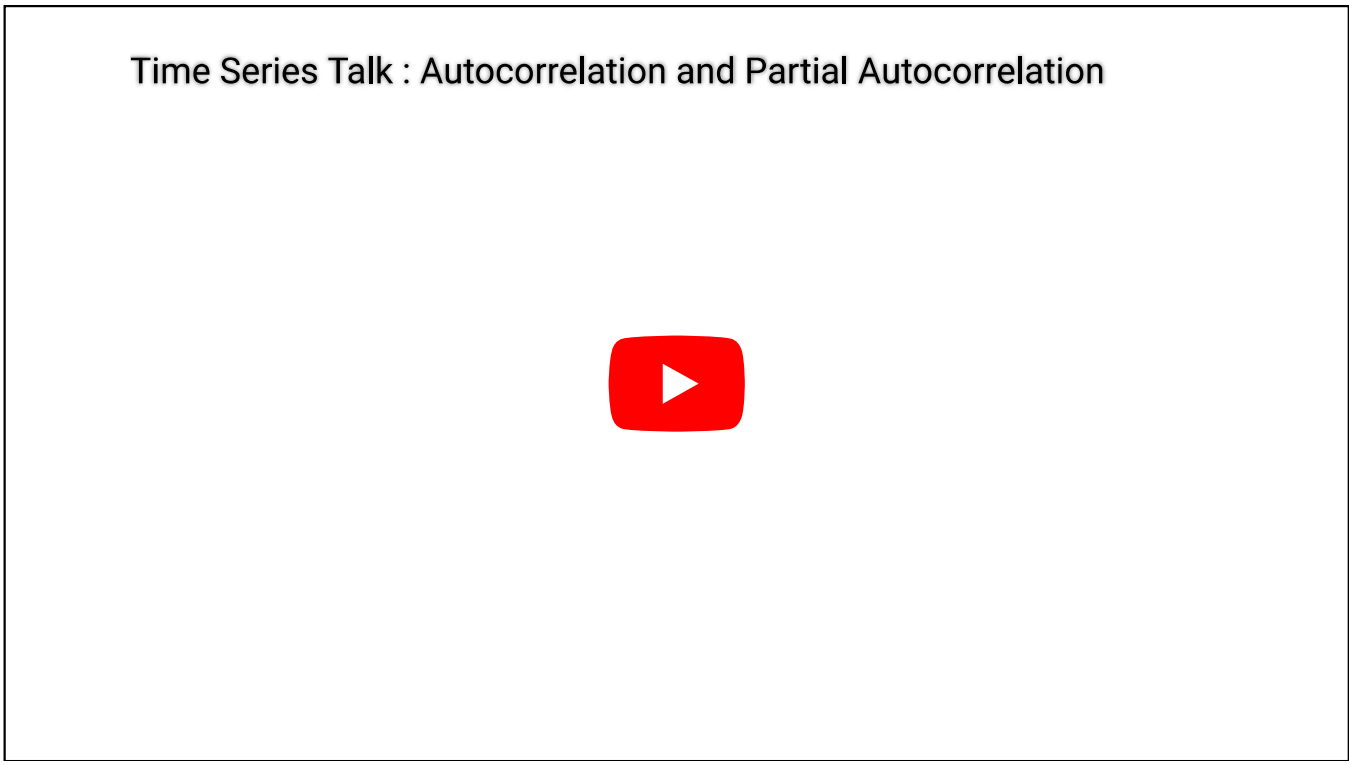
Precondition: Stationarity

ACF and PACF assume stationarity of the underlying time series.

Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)

The ACF and PACF are used to figure out the order of AR, MA, and ARMA models.

If you need some introduction to or a refresher on the ACF and PACF, I recommend the following video:



Autocorrelation Function (ACF)

Autocorrelation is the correlation between a time series with a lagged version of itself. The ACF starts at a lag of 0, which is the correlation of the time series with itself and therefore results in a correlation of 1.

We'll use the `plot_acf` function from the `statsmodels.graphics.tsaplots` library [5]. For this article, we'll only look at 15 lags since we are using minimal examples.

```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(time_series_values, lags = 15)
```

The ACF plot can provide answers to the following questions:

- Is the observed time series **white noise/random**?
- Is an observation related to an adjacent observation, an observation twice-removed, and so on?
- Can the observed time series be modeled with an **MA model**? If yes, what is the order?

Partial Autocorrelation Function (PACF)

> *The partial autocorrelation at lag k is the autocorrelation between X_t_t and X_(t-k) that is not accounted for by lags 1 through k−1. [4]*

We'll use the `plot_pacf` function from the `statsmodels.graphics.tsaplots` library with the parameter `method = "ols"` (regression of time series on lags of it and on constant)[5].

```
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(time_series_values, lags = 15, method = "ols")
```

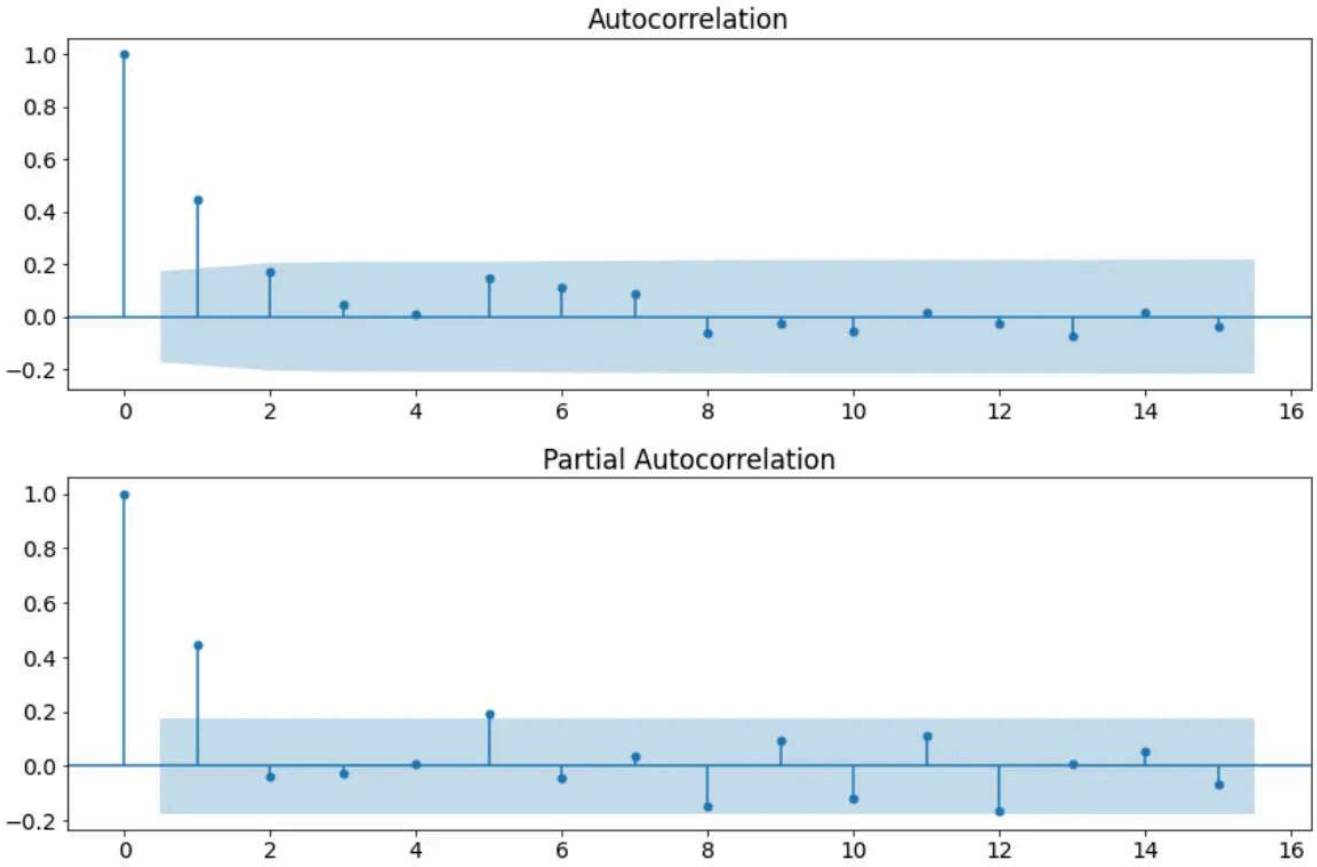
Sidenote: The default parameter for `method` is `yw` (Yule-Walker with sample-size adjustment in the denominator for acovf). However, this default value is causing some implausible autocorrelations higher than 1 on the sample data. Therefore, we change the `method` parameter to one that is not causing this issue. `ywmle` would also work fine as suggested in this StackExchange post [3].

The PACF plot can provide answers to the following question:

- Can the observed time series be modeled with an **AR model**? If yes, what is the order?

Order of AR, MA, and ARMA Models

Below you can see an example of an ACF and PACF plot. These plots are called "lollipop plots" [2].



Example of an ACF and a PACF plot. (Image by the author via Kaggle)

Both the ACF and PACF start with a **lag of 0**, which is the correlation of the time series with itself and therefore results in a **correlation of 1**.

The difference between ACF and PACF is the inclusion or exclusion of indirect correlations in the calculation.

Additionally, you can see a **blue area** in the ACF and PACF plots. This blue area depicts the 95% confidence interval and is an indicator of the **significance threshold**. That means, anything within the blue area is statistically close to zero and anything outside the blue area is statistically non-zero.

To determine the order of the model, you check:

> *"How [many] lollipops are above or below the confidence interval before the next lollipop enters the blue area?" — [2]*

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off (Geometric decay)	Significant at lag q / Cuts off after lag q	Tails off (Geometric decay)
PACF	Significant at each lag p / Cuts off after lag p	Tails off (Geometric decay)	Tails off (Geometric decay)

Image by the author via Kaggle inspired by [1]

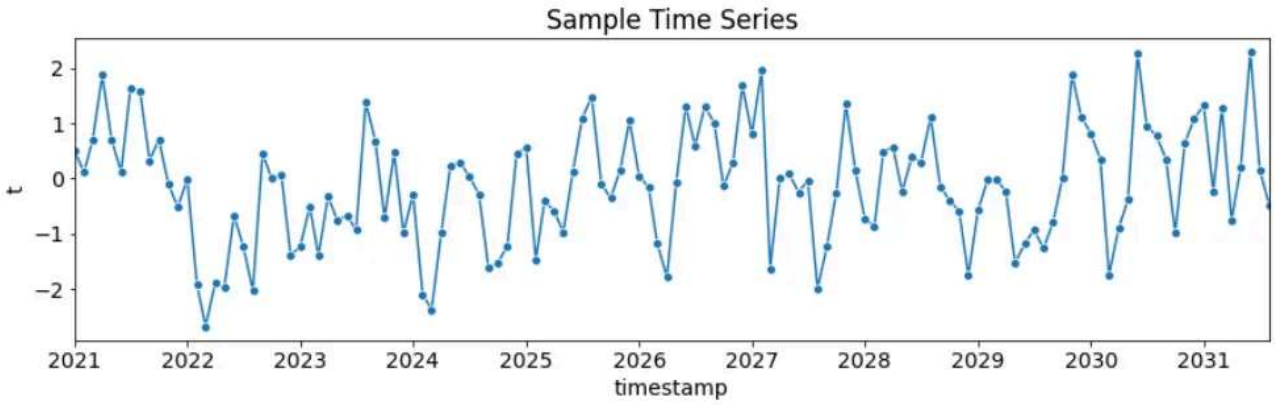
Examples

In this section, we'll look at a few time series examples and look at:

- What the ACF and PACF plots look like
- How to determine whether to model the time series with an AR or MA model
- How to determine the order of the AR or MA model
- How to find the parameters of the AR or MA model

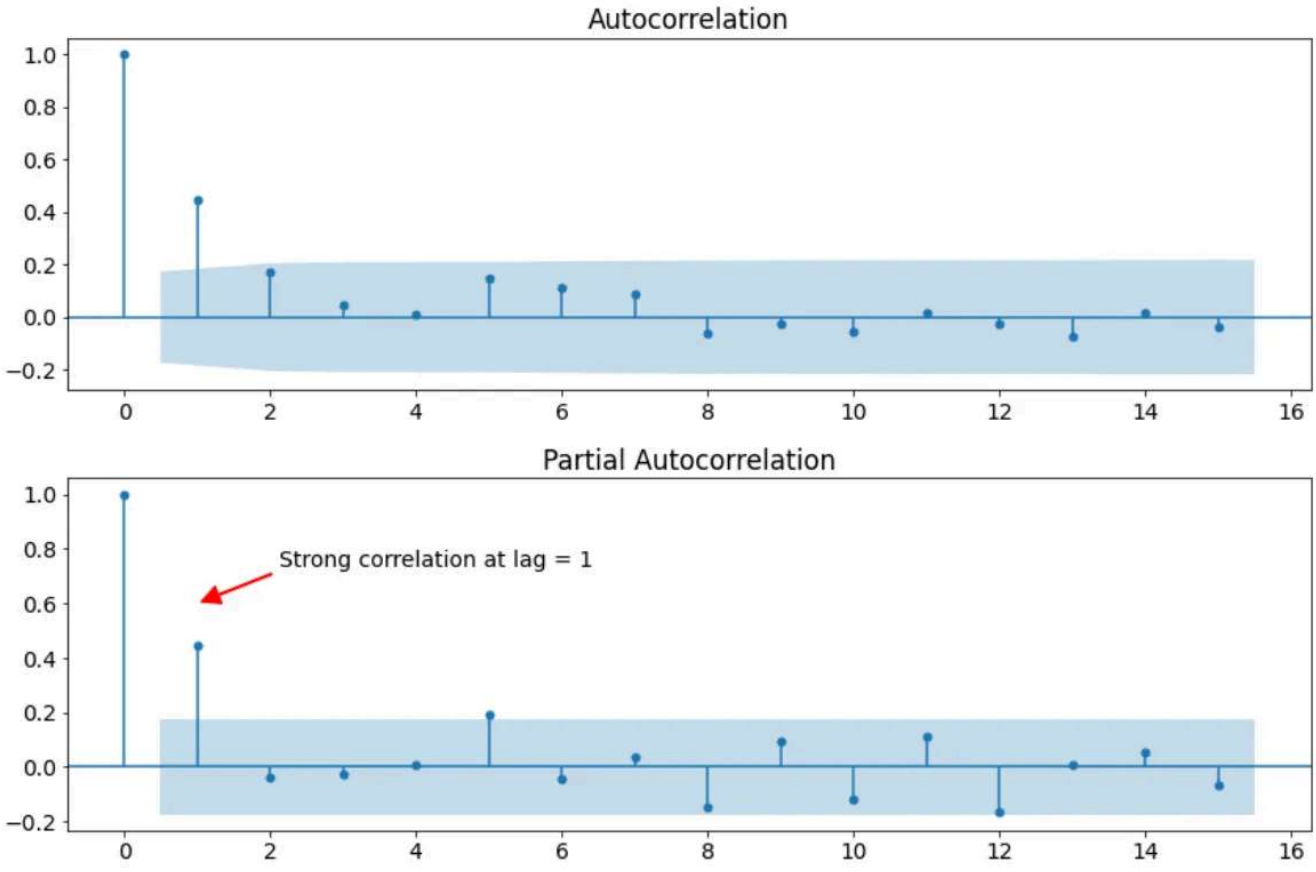
AR(1) Process

The following time series is an AR(1) process with 128 timesteps and `alpha_1 = 0.5`. It meets the precondition of stationarity.



Fictional Sample Time Series: AR(1) Process with alpha_1 = 0.5 (Image by the author via Kaggle)

The following figure shows the resulting ACF and PACF plots:



ACF and a PACF plot of the AR(1) process. (Image by the author via Kaggle)

We can make the following observations:

- There are several autocorrelations that are significantly non-zero. Therefore, the time series is non-random.
- High degree of autocorrelation between adjacent (lag = 1) in PACF plot
- Geometric decay in ACF plot

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off (Geometric decay)	Significant at lag q / Cuts off after lag q	Tails off (Geometric decay)
PACF	Significant at each lag p / Cuts off after lag p	Tails off (Geometric decay)	Tails off (Geometric decay)

Based on the above table, we can use an **AR(1) model** to model this process.

With AR(p=1), the formula

$$\hat{y}_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p}$$

can be rewritten to the following:

$$\hat{y}_t = \alpha_1 y_{t-1}$$

To find the parameter `alpha_1` we fit the AR model as follows:

```
from statsmodels.tsa.ar_model import AutoReg
ar_model = AutoReg(X_train, lags = 1).fit()
ar_model.summary()
```

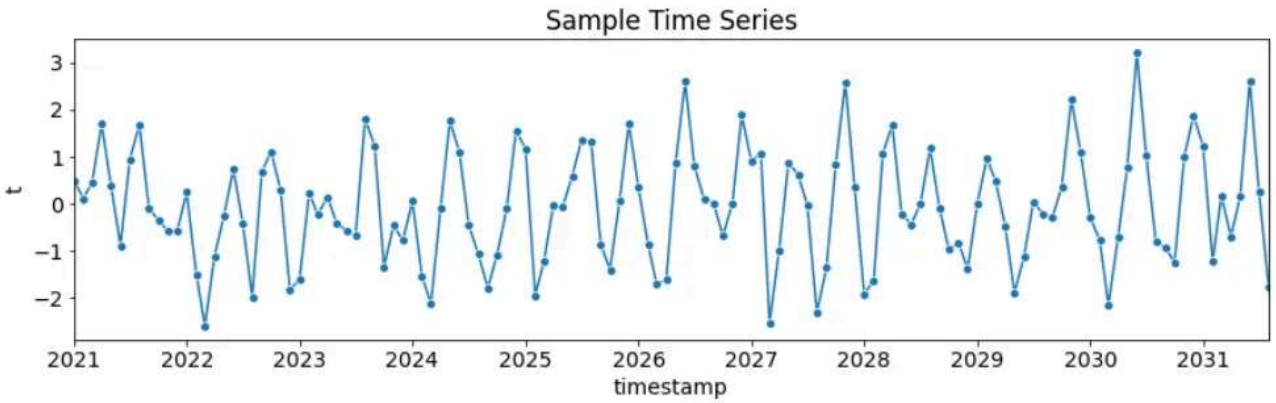
	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-0.1322	0.092	-1.434	0.152	-0.313	0.048
t.L1	0.4710	0.088	5.353	0.000	0.299	0.643

Parameter fitted by the AR model. (Image by the author via Kaggle)

As you can see, the AR(1) model fits an `alpha_1 = 0.4710`, which is quite close to the `alpha_1 = 0.5` that we have set.

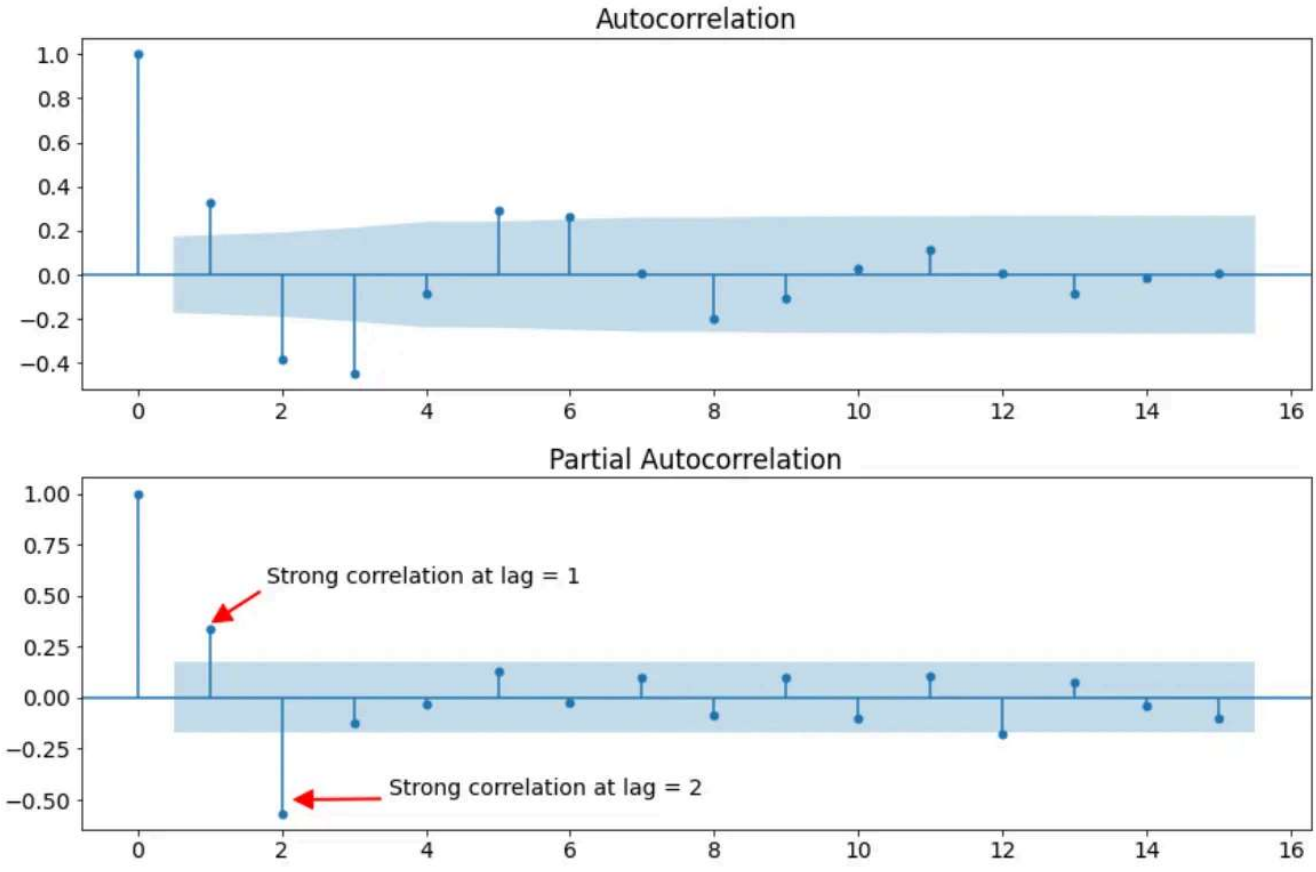
AR(2) Process

The following time series is an AR(2) process with 128 timesteps, `alpha_1 = 0.5`, and `alpha_2 = -0.5`. It meets the precondition of stationarity.



Fictional Sample Time Series: AR(2) Process with alpha_1 = 0.5 and alpha_2 = -0.5 (Image by the author via Kaggle)

The following figure shows the resulting ACF and PACF plots:



ACF and a PACF plot of the AR(2) process. (Image by the author via Kaggle)

We can make the following observations:

- There are several autocorrelations that are significantly non-zero. Therefore, the time series is non-random.
- High degree of autocorrelation between adjacent (lag = 1) and near-adjacent (lag = 2) observations in PACF plot
- Geometric decay in ACF plot

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off (Geometric decay)	Significant at lag q / Cuts off after lag q	Tails off (Geometric decay)
PACF	Significant at each lag p / Cuts off after lag p	Tails off (Geometric decay)	Tails off (Geometric decay)

Image by the author via Kaggle inspired by [1]

Based on the above table, we can use an **AR(2) model** to model this process.

With AR(p=2), the formula

$$\hat{y}_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p}$$

can be rewritten to the following:

$$\hat{y}_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2}$$

To find the parameters `alpha_1` and `alpha_2` we fit the AR model as follows:

```
from statsmodels.tsa.ar_model import AutoReg
ar_model = AutoReg(X_train, lags = 2).fit()
ar_model.summary()
```

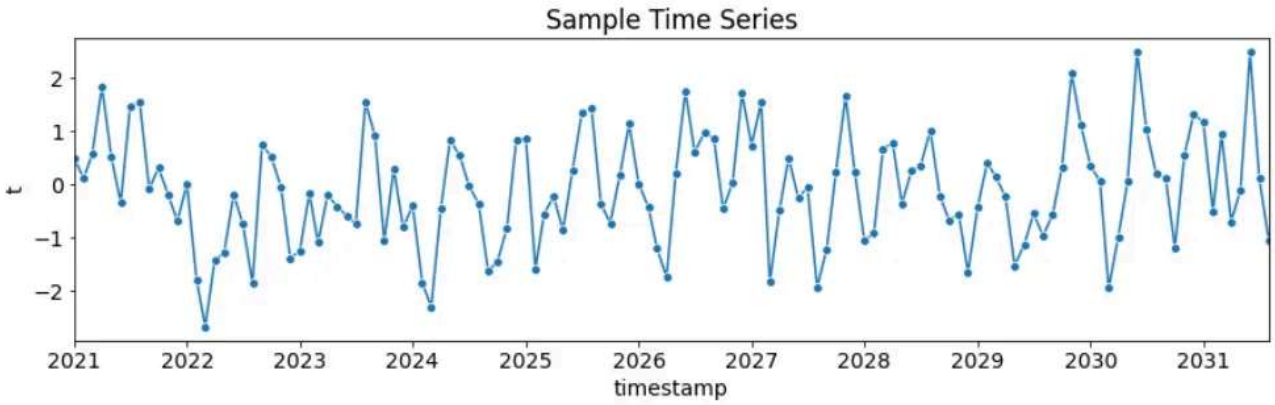
	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-0.1321	0.091	-1.446	0.148	-0.311	0.047
t.L1	0.5191	0.082	6.363	0.000	0.359	0.679
t.L2	-0.5855	0.082	-7.103	0.000	-0.747	-0.424

Parameters fitted by the AR model. (Image by the author via Kaggle)

As you can see, the AR(2) model fits an `alpha_1 = 0.5191` and `alpha_2 = -0.5855`, which is quite close to the `alpha_1 = 0.5` and `alpha_2 = -0.5` that we have set.

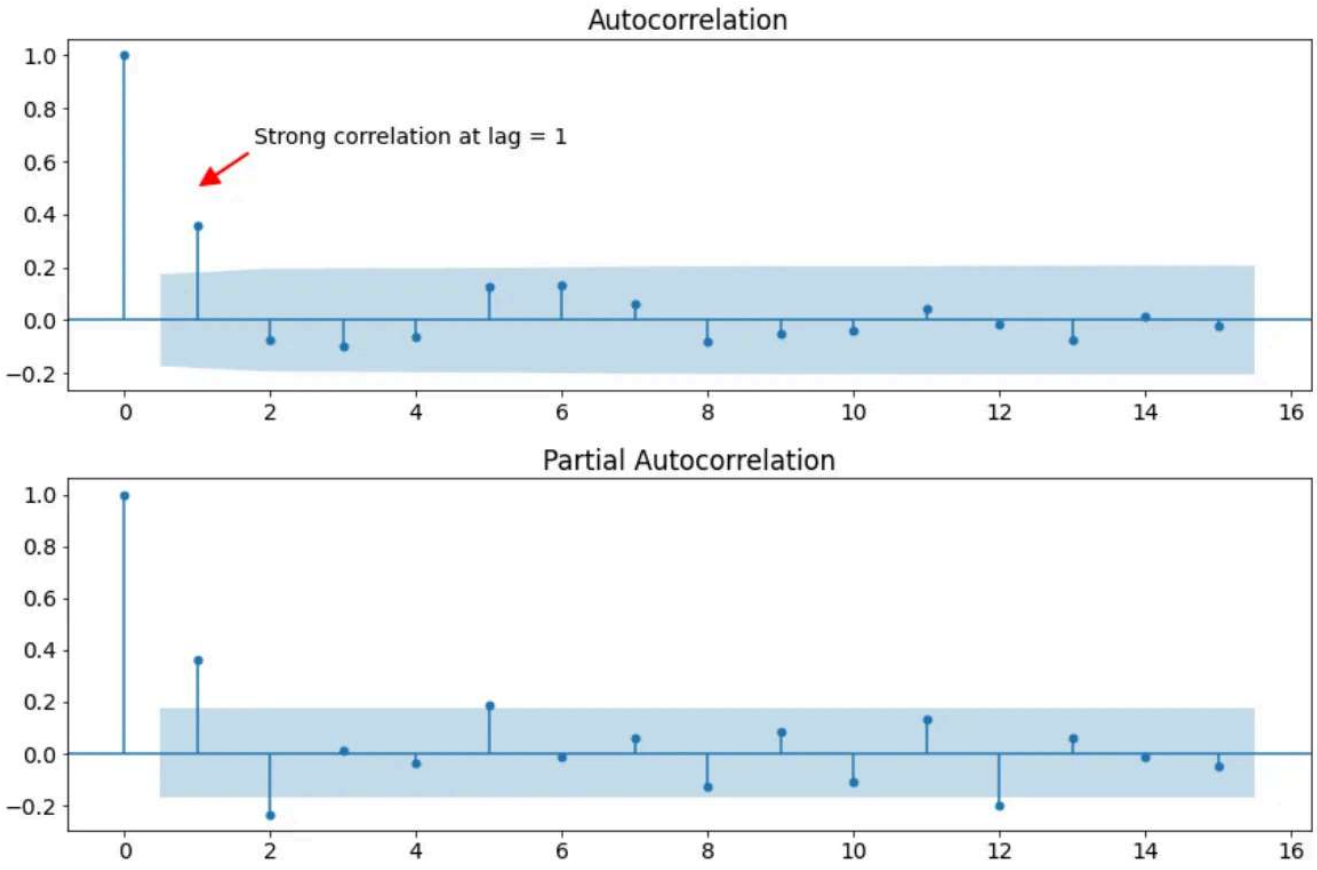
MA(1) Process

The following time series is an MA(1) process with 128 timesteps and `beta_1 = 0.5`. It meets the precondition of stationarity.



Fictional Sample Time Series: MA(1) Process with beta_1 = 0.5 (Image by the author via Kaggle)

The following figure shows the resulting ACF and PACF plots:



ACF and a PACF plot of the MA(1) process. (Image by the author via Kaggle)

We can make the following observations:

- There are several autocorrelations that are significantly non-zero. Therefore, the time series is non-random.
- High degree of autocorrelation between adjacent (lag = 1) in ACF plot

- Geometric decay in PACF plot

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off (Geometric decay)	Significant at lag q / Cuts off after lag q	Tails off (Geometric decay)
PACF	Significant at each lag p / Cuts off after lag p	Tails off (Geometric decay)	Tails off (Geometric decay)

Image by the author via Kaggle inspired by [1]

Based on the above table, we can use an **MA(1) model** to model this process.

$$y_t = \epsilon_t + \rho_1 \epsilon_{t-1} + \dots + \rho_q \epsilon_{t-q}$$

can be rewritten to the following:

$$\hat{y}_t = \epsilon_t + \beta_1 \epsilon_{t-1}$$

To find the parameter `beta_1` we fit the MA model as follows:

```
from statsmodels.tsa.arima_model import ARMA
ma_model = ARMA(X_train, order = (0, 1)).fit()
ma_model.summary()
```

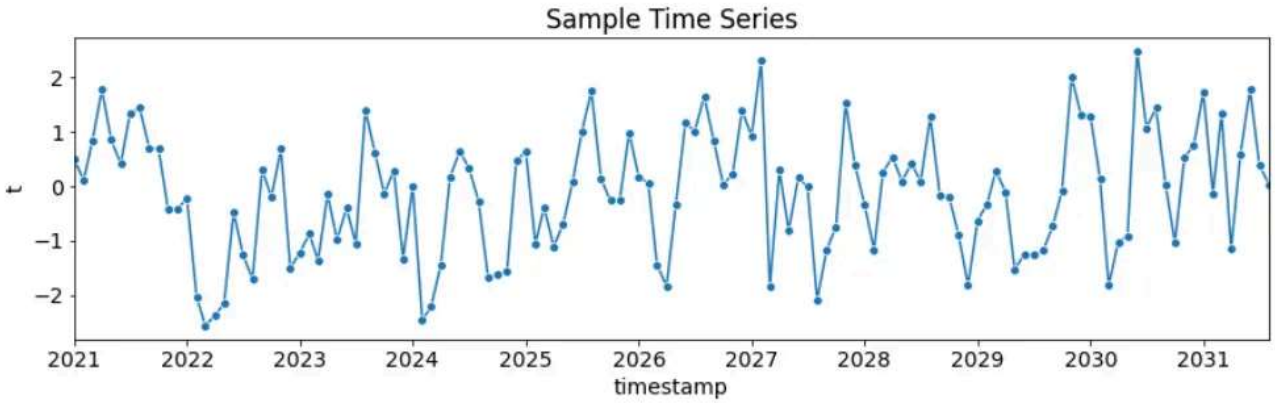
	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.1765	0.135	-1.303	0.192	-0.442	0.089
ma.L1.t	0.5172	0.100	5.173	0.000	0.321	0.713

Parameter fitted by the (AR)MA model. (Image by the author via Kaggle)

As you can see, the MA(1) model fits a `beta_1 = 0.5172`, which is quite close to the `beta_1 = 0.5` that we have set.

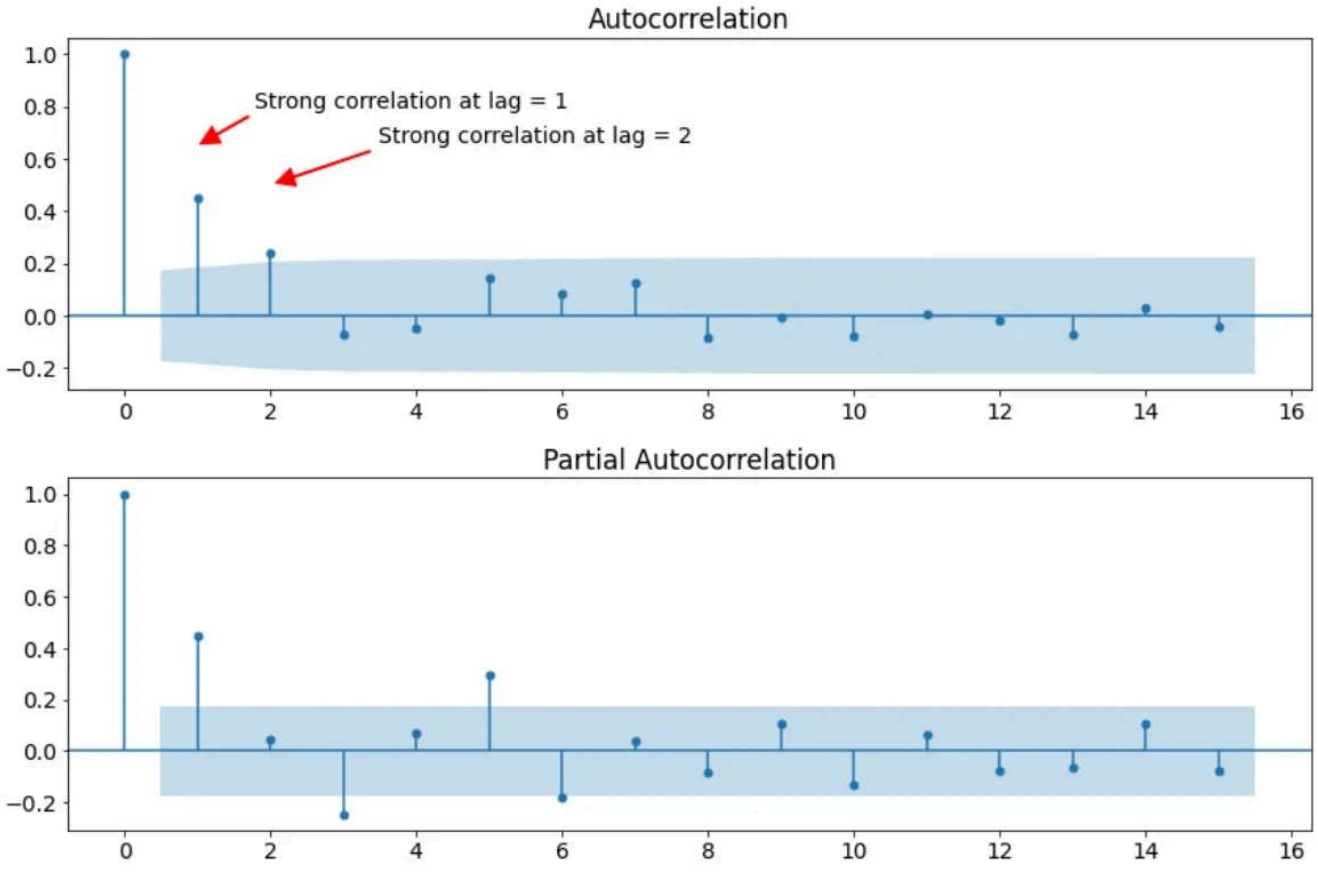
MA(2) Process

The following time series is an MA(2) process with 128 timesteps and `beta_1 = 0.5` and `beta_2 = 0.5`. It meets the precondition of stationarity.



Fictional Sample Time Series: MA(2) Process with beta_1 = 0.5 und beta_2 = 0.5 (Image by the author via Kaggle)

The following figure shows the resulting ACF and PACF plots:



ACF and a PACF plot of the MA(2) process. (Image by the author via Kaggle)

We can make the following observations:

- There are several autocorrelations that are significantly non-zero. Therefore, the time series is non-random.
- High degree of autocorrelation between adjacent (lag = 1) and near-adjacent (lag = 2) observations in ACF plot
- Geometric decay in PACF plot

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off (Geometric decay)	Significant at lag q / Cuts off after lag q	Tails off (Geometric decay)
PACF	Significant at each lag p / Cuts off after lag p	Tails off (Geometric decay)	Tails off (Geometric decay)

Image by the author via Kaggle inspired by [1]

Based on the above table, we can use an **MA(2) model** to model this process.

With MA(q=2), the formula

$$\hat{y}_t = \epsilon_t + \beta_1 \epsilon_{t-1} + \cdots + \beta_q \epsilon_{t-q}$$

can be rewritten to the following:

$$\hat{y}_t = \epsilon_t + \beta_1 \epsilon_{t-1} + \beta_2 \epsilon_{t-2}$$

To find the parameters `beta_1` and `beta_2` we fit the MA model as follows:

```
from statsmodels.tsa.arima_model import ARMA
ma_model = ARMA(X_train, order = (0, 2)).fit()
ma_model.summary()
```

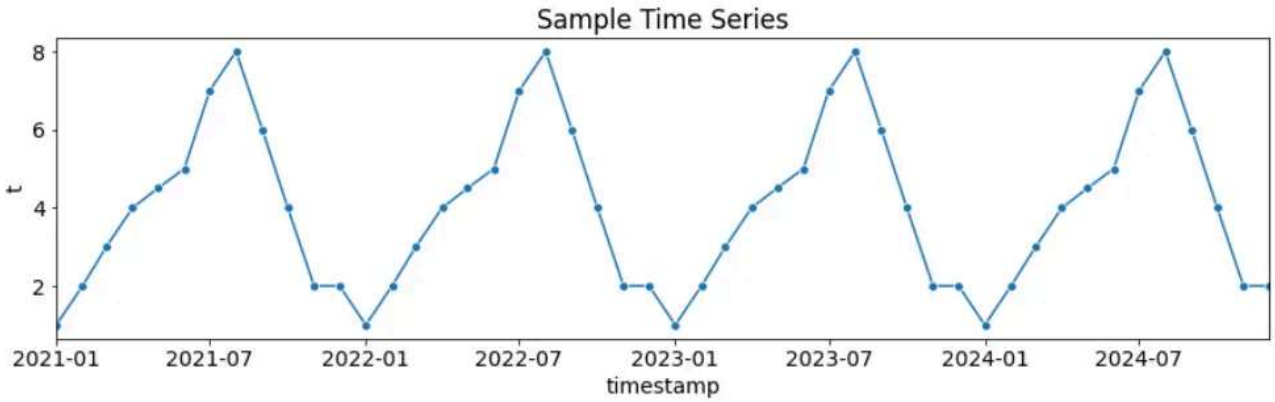
	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.2292	0.186	-1.230	0.219	-0.594	0.136
ma.L1.t	0.5226	0.084	6.235	0.000	0.358	0.687
ma.L2.t	0.5843	0.110	5.318	0.000	0.369	0.800

Parameters fitted by the (AR)MA model. (Image by the author via Kaggle)

As you can see, the MA(2) model fits a `beta_1 = 0.5226` and `beta_2 = 0.5843`, which is quite close to the `beta_1 = 0.5` and `beta_2 = 0.5` that we have set.

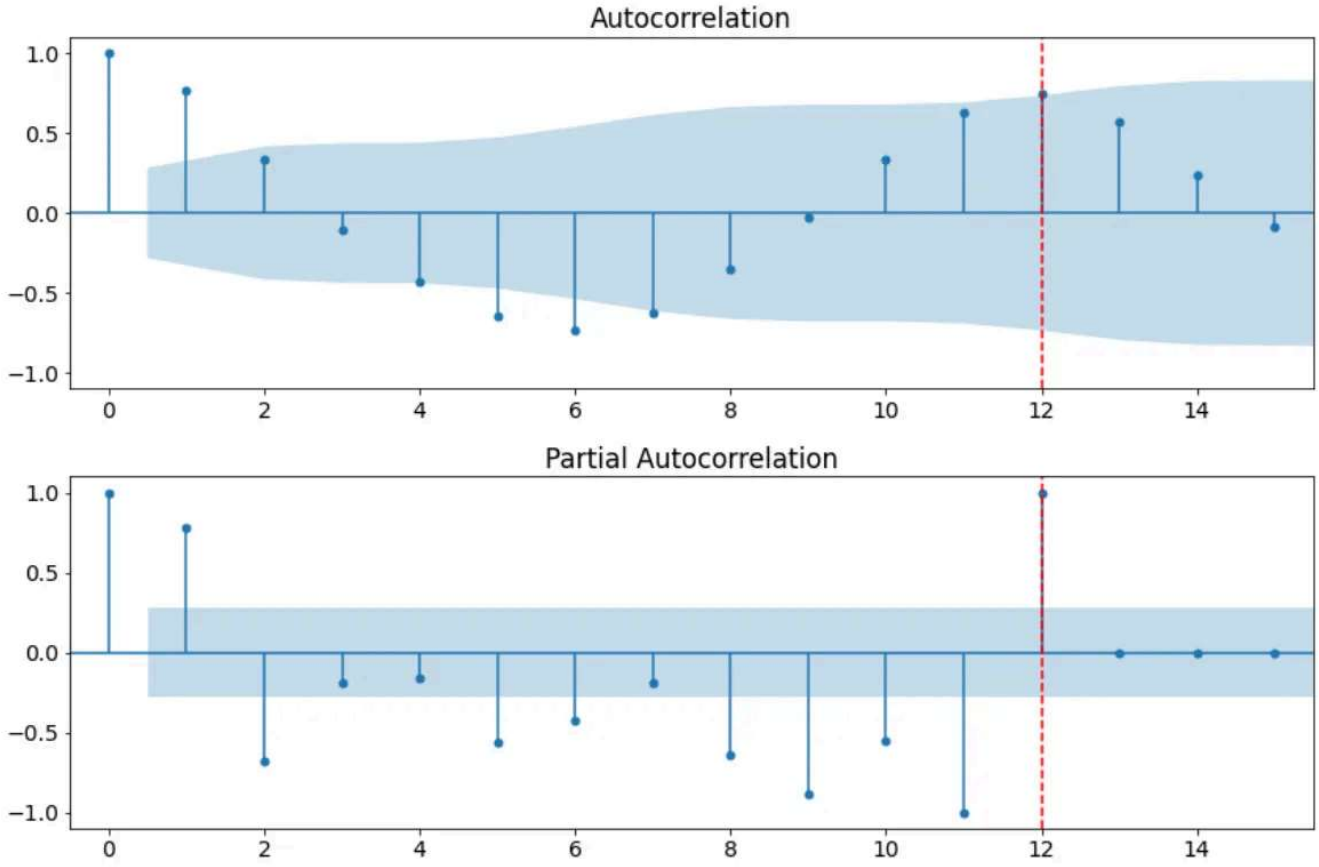
Periodical

The following time series is periodical with T=12. It consists of 48 timesteps.



Fictional Sample Time Series: Periodical with T=12 (Image by the author via Kaggle)

The following figure shows the resulting ACF and PACF plots:



ACF and a PACF plot of the periodical process. (Image by the author via Kaggle)

We can make the following observations:

- There are several autocorrelations that are significantly non-zero. Therefore, the time series is non-random.
- High degree of autocorrelation between adjacent (lag = 1) and near-adjacent observations in PACF plot
- From both the ACF and PACF plot, we can see a strong correlation with the adjacent observation (lag = 1) and also at a lag of 12, which is the value of T.

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off (Geometric decay)	Significant at lag q / Cuts off after lag q	Tails off (Geometric decay)
PACF	Significant at each lag p / Cuts off after lag p	Tails off (Geometric decay)	Tails off (Geometric decay)

Image by the author via Kaggle inspired by [1]

With AR(p=12), the formula

$$\hat{y}_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p}$$

can be rewritten to the following:

$$\hat{y}_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \alpha_3 y_{t-3} + \alpha_4 y_{t-4} + \alpha_5 y_{t-5} + \alpha_6 y_{t-6} + \alpha_7 y_{t-7} + \alpha_8 y_{t-8} + \alpha_9 y_{t-9} + \alpha_{10} y_{t-10} + \alpha_{11} y_{t-11} + \alpha_{12} y_{t-12}$$

To find the parameters `alpha_1` through `alpha_12` we fit the AR model as follows:

```
from statsmodels.tsa.ar_model import AutoReg
ar_model = AutoReg(X_train, lags = 12).fit()
ar_model.summary()
```

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0205	9.48e-18	2.16e+15	0.000	0.021	0.021
t.L1	-0.0004	5.9e-15	-7.17e+10	0.000	-0.000	-0.000
t.L2	-0.0004	6.05e-15	-7e+10	0.000	-0.000	-0.000
t.L3	-0.0004	6.03e-15	-7.01e+10	0.000	-0.000	-0.000
t.L4	-0.0004	5.97e-15	-7.09e+10	0.000	-0.000	-0.000
t.L5	-0.0004	5.56e-15	-7.61e+10	0.000	-0.000	-0.000
t.L6	-0.0004	5.64e-15	-7.5e+10	0.000	-0.000	-0.000
t.L7	-0.0004	5.89e-15	-7.18e+10	0.000	-0.000	-0.000
t.L8	-0.0004	5.67e-15	-7.46e+10	0.000	-0.000	-0.000
t.L9	-0.0004	5.8e-15	-7.3e+10	0.000	-0.000	-0.000
t.L10	-0.0004	6e-15	-7.05e+10	0.000	-0.000	-0.000
t.L11	-0.0004	5.75e-15	-7.35e+10	0.000	-0.000	-0.000
t.L12	0.9996	5.61e-15	1.78e+14	0.000	1.000	1.000

Parameters fitted by the AR model. (Image by the author via Kaggle)

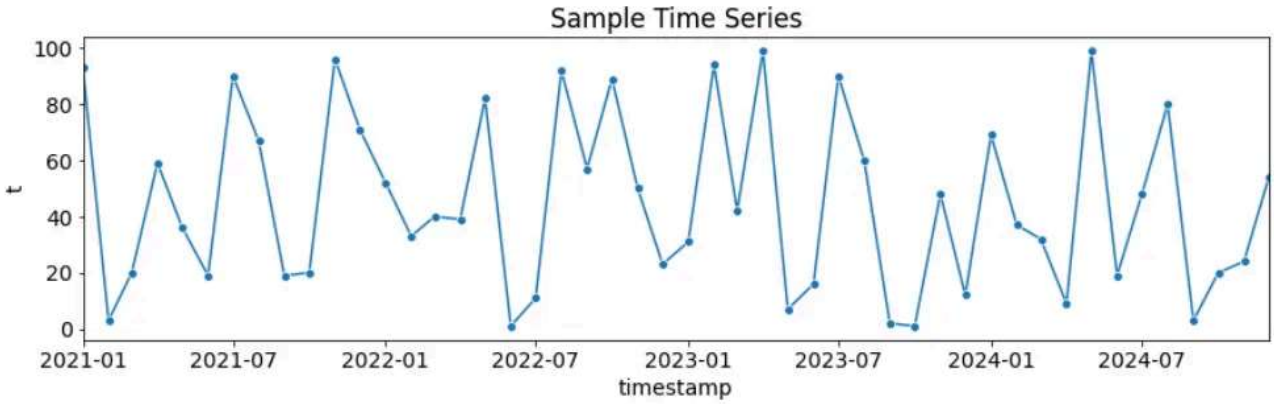
As you can see, the MA(2) model fits the parameters `alpha_1..11 = -0.0004` and `alpha_12 = 0.9996`, which is quite close to the `alpha_1..11 = 0` and `alpha_12 = 1` that we have set.

With these parameters, the formula can be rewritten as shown below:

$$\hat{y}_t = y_{t-12}$$

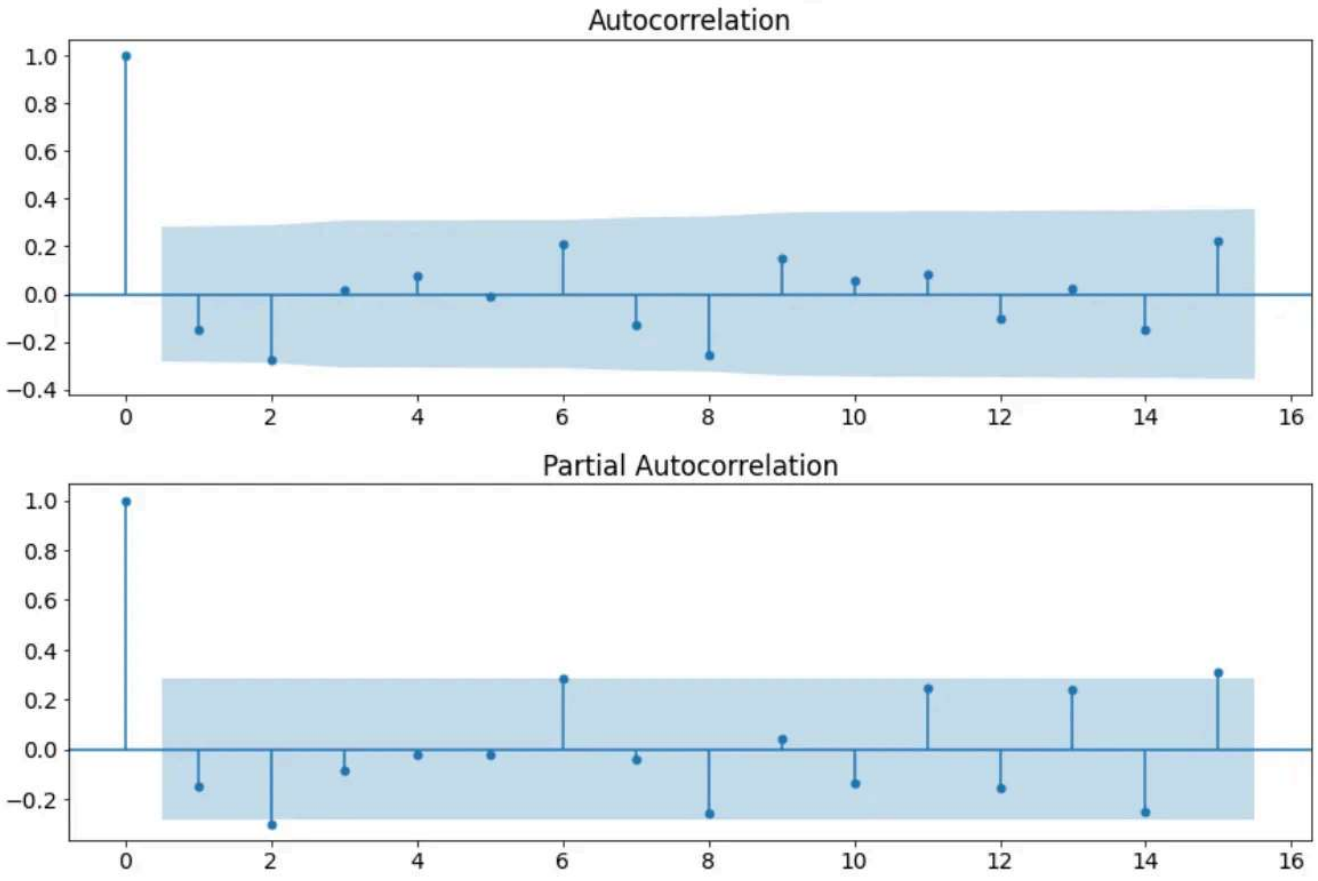
White Noise

The following time series is random. It consists of 48 timesteps.



Fictional Sample Time Series: White Noise (Image by the author via Kaggle)

The following figure shows the resulting ACF and PACF plots:



ACF and a PACF plot of the white noise. (Image by the author via Kaggle)

We can make the following observation:

- There's only one autocorrelation that is significantly non-zero at a lag of 0. Therefore, the time series is random.

Modeling white noise is difficult because we can't retrieve any parameters from the ACF and PACF plots.

Conclusion

In this article, we looked at various examples of AR and MA processes, periodical time series, and white noise to help you build an intuition for interpreting ACF and PACF plots.

This article discussed:

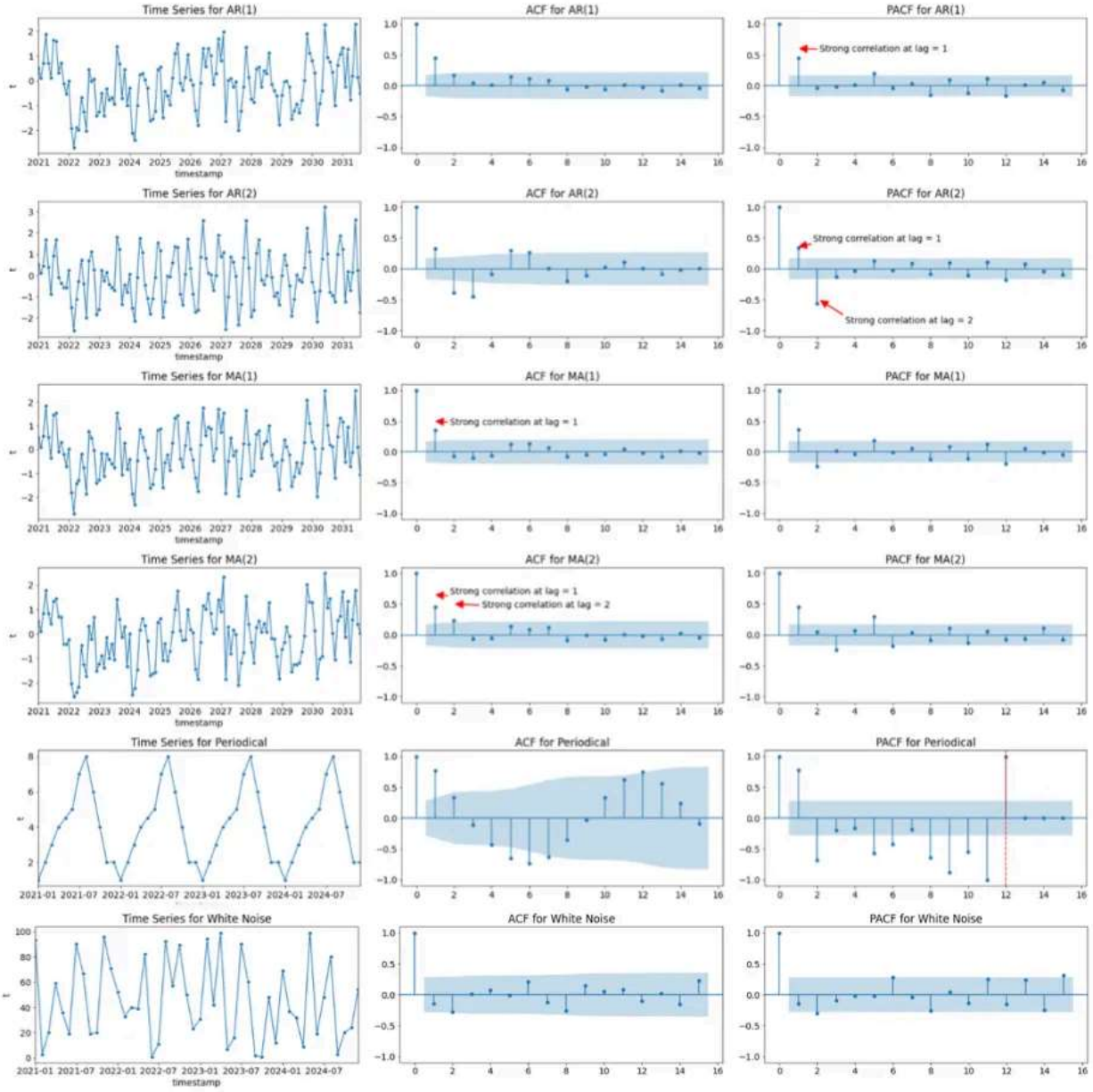
- How to detect randomness in a time series
- How to determine whether to model a time series with an AR or MA model
- How to determine the order of the AR or MA model
- How to find the parameters of the AR or MA model

The following figure is a visual summary of this article as a cheat sheet:

Interpreting PACF and ACF

Precondition: Time series must be stationary

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off (Geometric decay)	Significant at lag q / Cuts off after lag q	Tails off (Geometric decay)
PACF	Significant at each lag p / Cuts off after lag p	Tails off (Geometric decay)	Tails off (Geometric decay)



Interpreting ACF and PACF Cheat Sheet (Image by the author)

Enjoyed This Story?

Subscribe for free to get notified when I publish a new story.

Find me on LinkedIn, Twitter, and Kaggle!

References

[1] S. Ali, "Reading the ACF and PACF Plots — The Missing Manual / Cheatsheet". linkedin.com. https://www.linkedin.com/pulse/reading-acf-pacf-plots-missing-manual-cheatsheet-saqib-ali/ (accessed July 27, 2022)

[2] "Arauto", "How to choose the parameters for the model". arauto.readthedocs.io. https://arauto.readthedocs.io/en/latest/how_to_choose_terms.html (accessed July 29, 2022)

[3] "Cross Validated", "What do very high PACF values (>10) mean?". *stackexchange.com.* https://stats.stackexchange.com/questions/380196/what-do-very-high-pacf-values-10-mean (accessed July 27, 2022)

[4] NIST, "6.4.4.6.3. Partial Autocorrelation Plot". nist.gov. https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc4463.htm (accessed July 27, 2022)

[5] "statsmodels 0.14.0 (+497)", "statsmodels.tsa.stattools.acf". statsmodels.org. https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.acf.html (accessed July 27, 2022)

Data Science | Time Series Analysis | Time Series Forecasting | Python | Exploratory Data Analysis

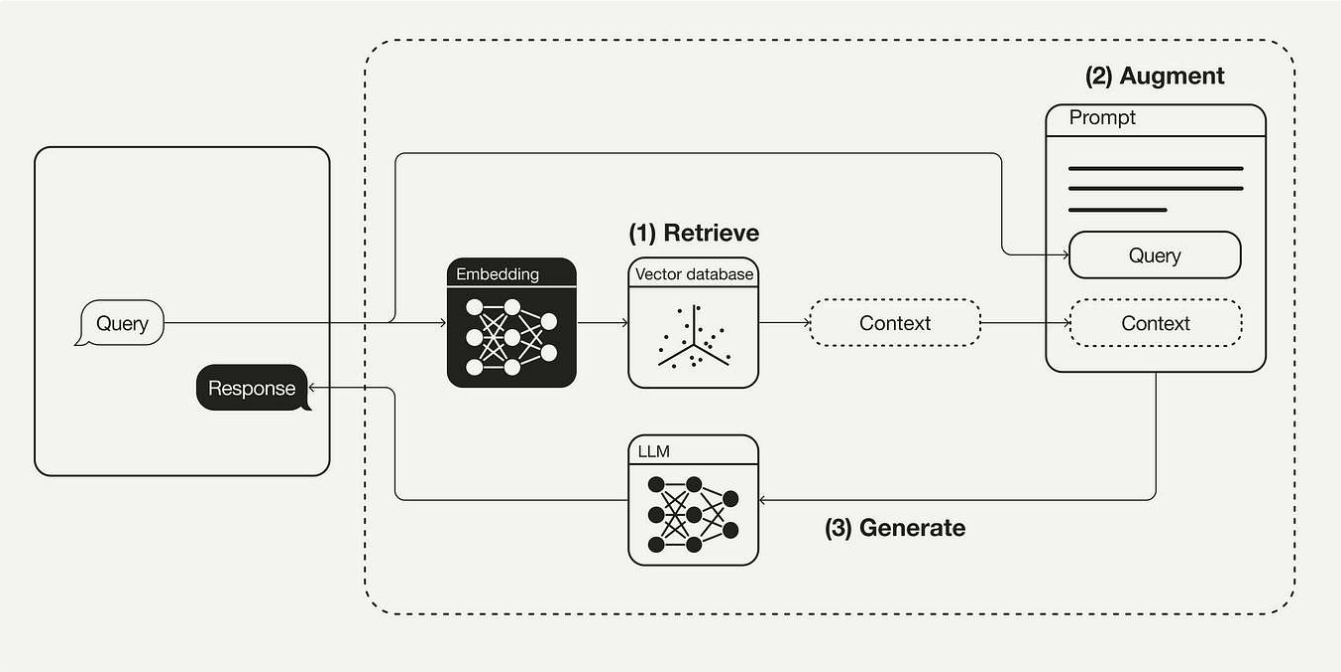
Follow

Written by Leonie Monigatti

32K Followers · Writer for Towards Data Science

Developer Advocate @ Weaviate. Follow for practical data science guides - whether you're a data scientist or not. linkedin.com/in/804250ab

More from Leonie Monigatti and Towards Data Science



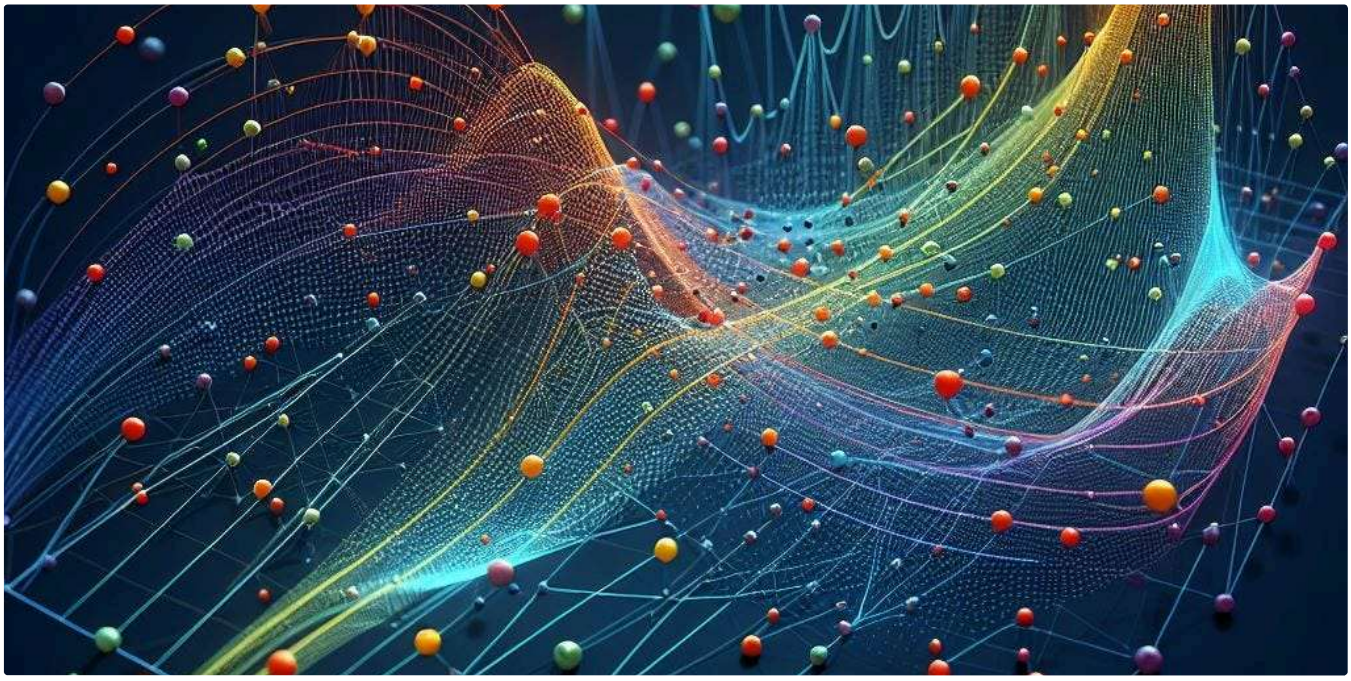
Leonie Monigatti in Towards Data Science

Retrieval-Augmented Generation (RAG): From Theory to LangChain Implementation

From the theory of the original academic paper to its Python implementation with OpenAI, Weaviate, and LangChain

7 min read · Nov 14, 2023

1.5K 13



Tim Sumner in Towards Data Science

A New Coefficient of Correlation

What if you were told there exists a new way to measure the relationship between two variables just like correlation except possibly...

10 min read · Mar 31, 2024

3K 36

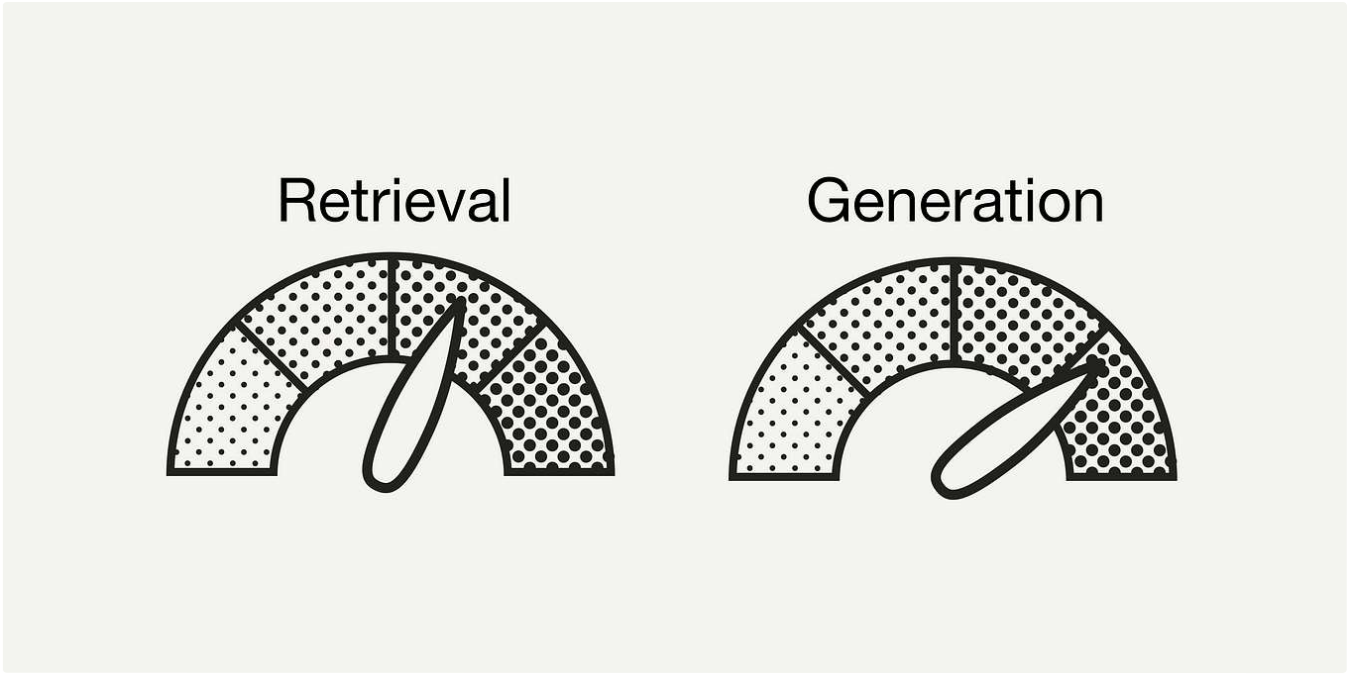
Rami Krispin in Towards Data Science

Setting A Dockerized Python Environment — The Elegant Way

This post provides a step-by-step guide for setting up a Python dockerized development environment with VScode and the Dev Containers...

9 min read · Apr 3, 2024

792 1



Leonie Monigatti in Towards Data Science

Evaluating RAG Applications with RAGAs

A framework with metrics and LLM-generated data to evaluate the performance of your Retrieval-Augmented Generation pipeline

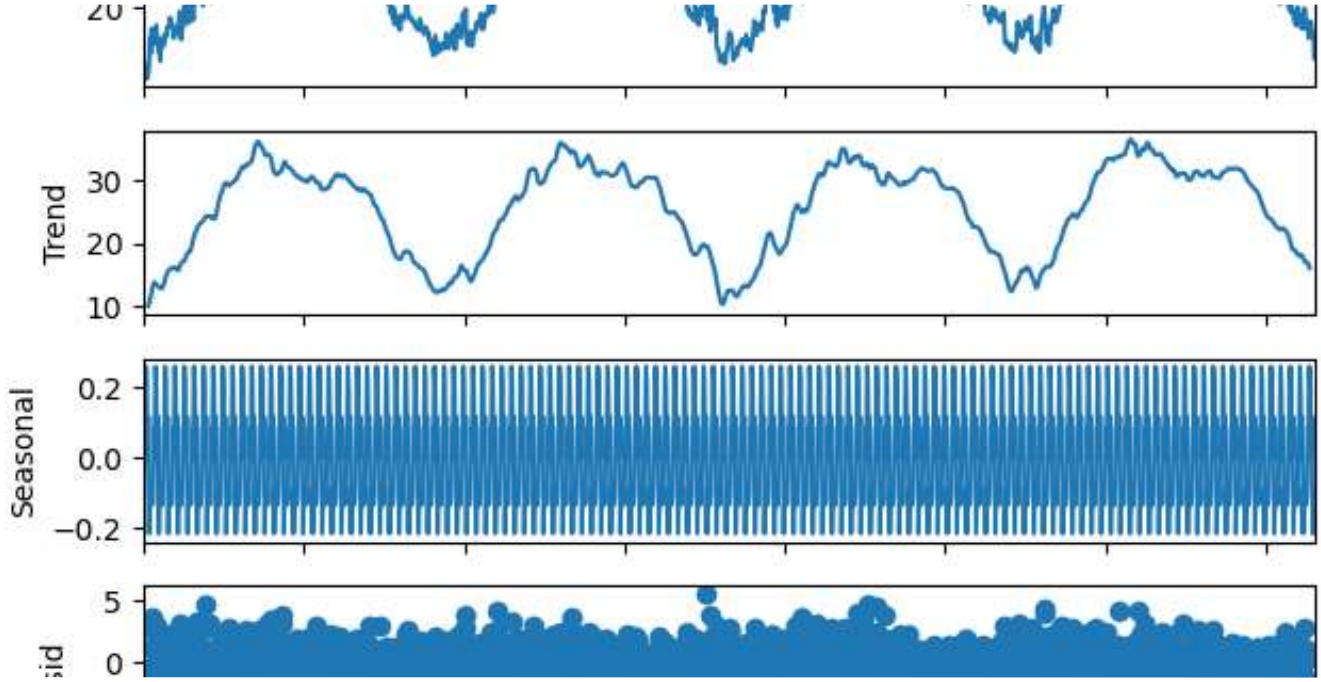
8 min read · Dec 13, 2023

1.2K 4

See all from Leonie Monigatti

See all from Towards Data Science

Recommended from Medium



Tirtha Mutha

Time Series forecasting using SARIMA in Python

A time series is a series of data points ordered in time. In a time series, time is often the independent variable, and the goal is usually...

5 min read · Nov 10, 2023

33

S. Do. in LatinXinAI

Time series forecasting — ARIMA and SARIMA

Time series forecasting is one of the most useful (and complex) fields of Machine Learning. In this article, second part of the...

7 min read · Dec 11, 2023

18 1

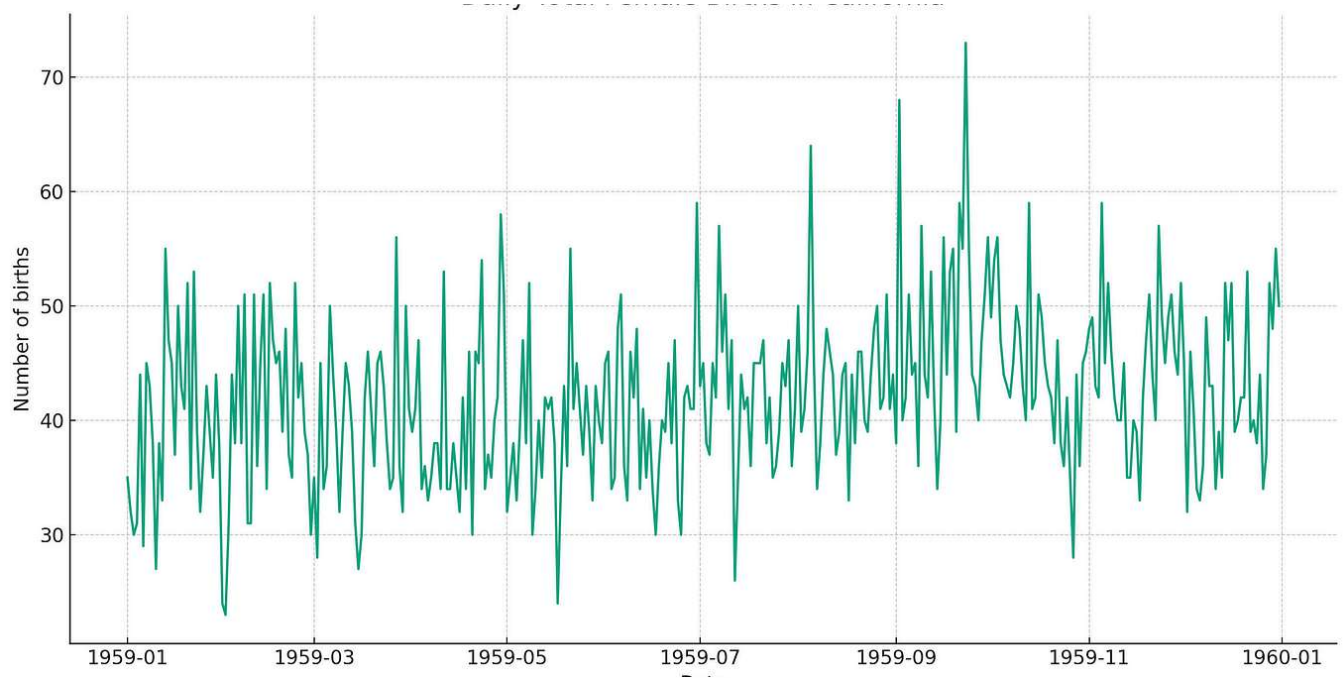
Lists

Predictive Modeling w/ Python
20 stories · 1119 saves

Practical Guides to Machine Learning
10 stories · 1343 saves

Coding & Development
11 stories · 574 saves

ChatGPT prompts
47 stories · 1461 saves



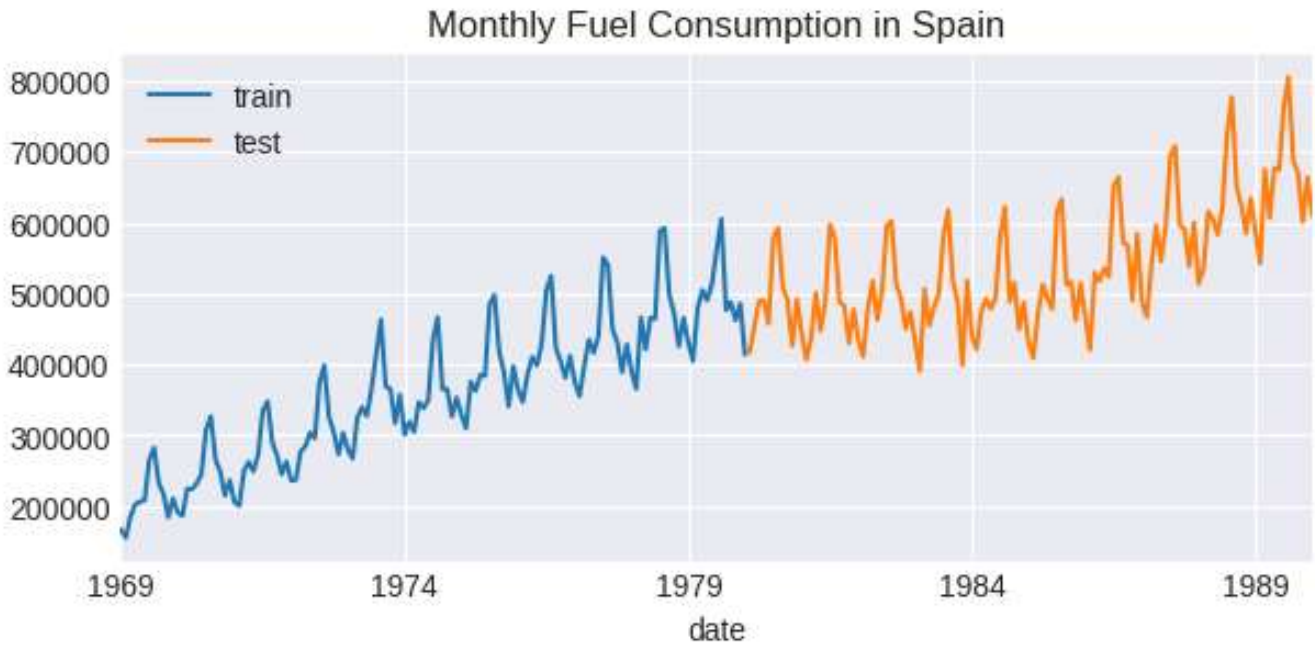
Divyesh Bhatt in The ML Classroom

Time Series Analysis and Forecasting with ARIMA in Python

Time series forecasting is a crucial area of machine learning that predicts future points in a series based on past data. It is especially...

4 min read · Nov 3, 2023

10



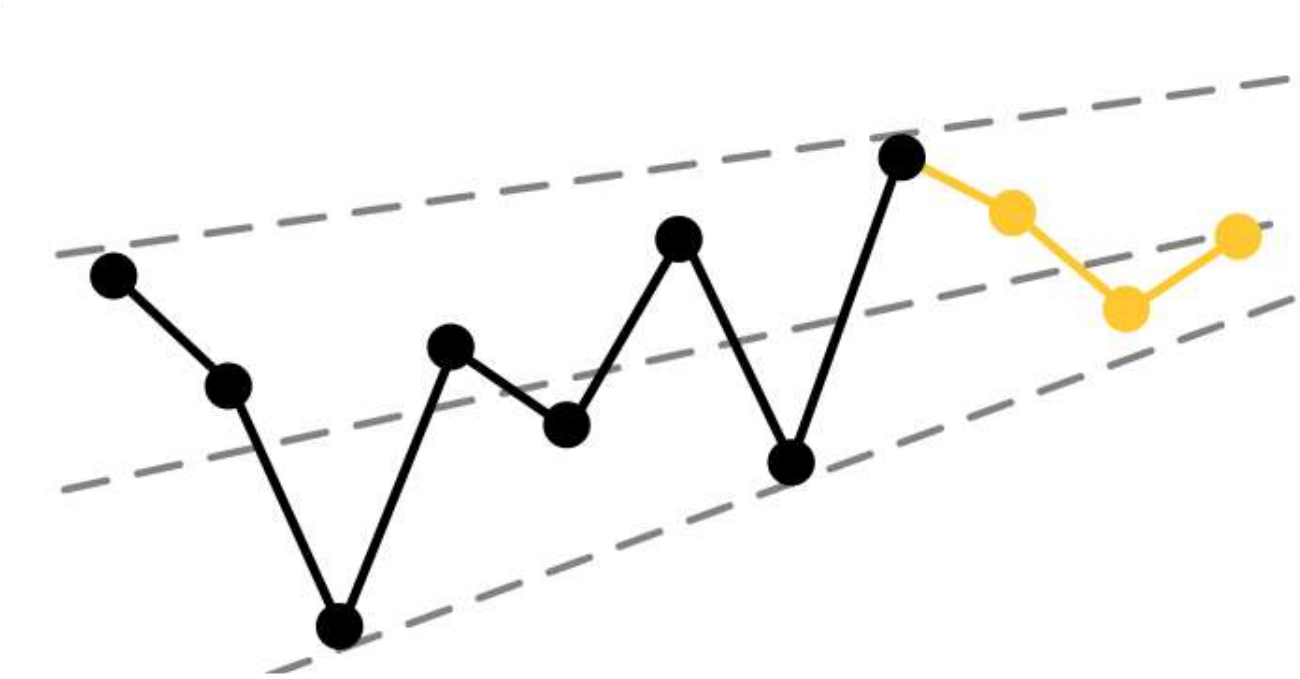
azhar in azhar labs

The Python Forecasting Toolkit: ARIMA and SARIMAX for Time Series Mastery

The world of time series forecasting using ARIMA (AutoRegressive Integrated Moving Average) and SARIMAX (Seasonal AutoRegressive Integrated...

✦ · 19 min read · Jan 6, 2024

6 2



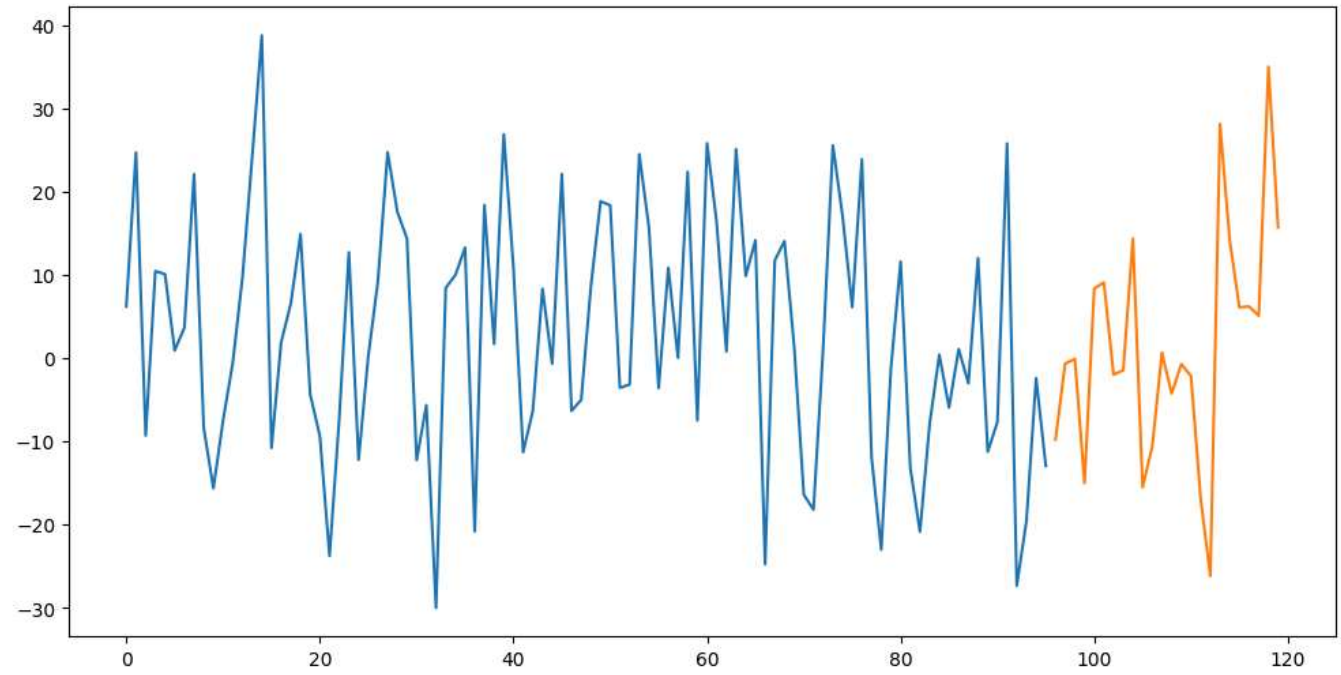
Leonie Monigatti

A Gentle Introduction to Time Series Analysis & Forecasting

Fundamental concepts around time series analysis and time series forecasting, including everything from classical approaches to modern...

✦ · 10 min read · Oct 23, 2022

349 4



Sampurnchouksey

Time Series Analysis (Forecasting with Python) Part-3

Forecasting

5 min read · Feb 25, 2024

40

See more recommendations