Data Journalism

Data Analysis
Part 3

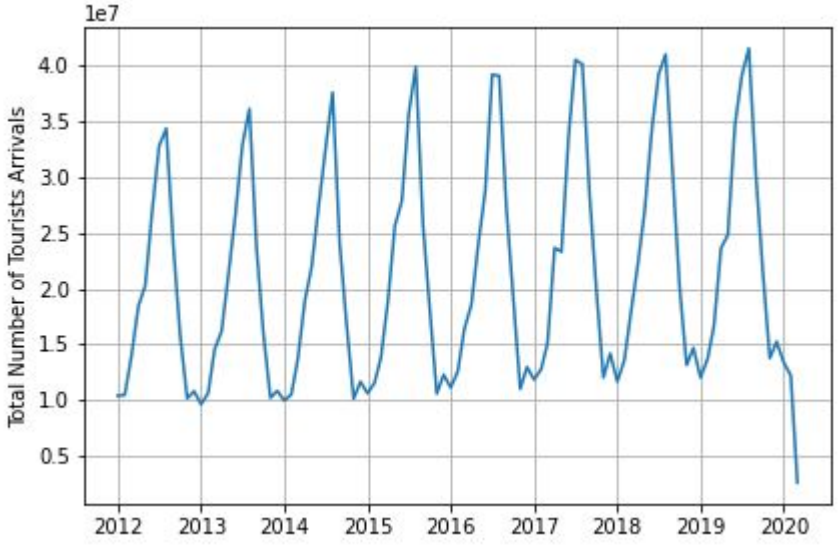
Angelica Lo Duca
angelica.loduca@iit.cnr.it

A **time series** is a series of data points indexed in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time.

From Wikipedia

Example 1

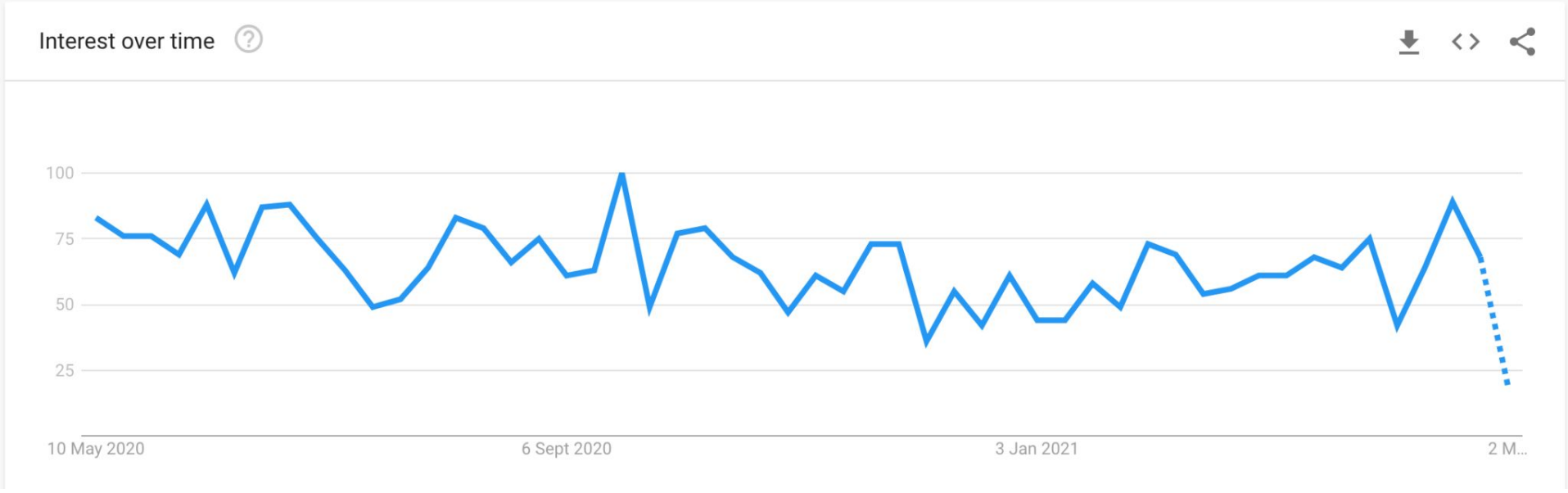
The number of tourists arrivals to Italy from 1990 to 2020



Example 2 - Google Trends

What the world is searching

word: tourism



Time Series VS Supervised/Unsupervised Learning

Supervised/Unsupervised Learning - no order in data observations

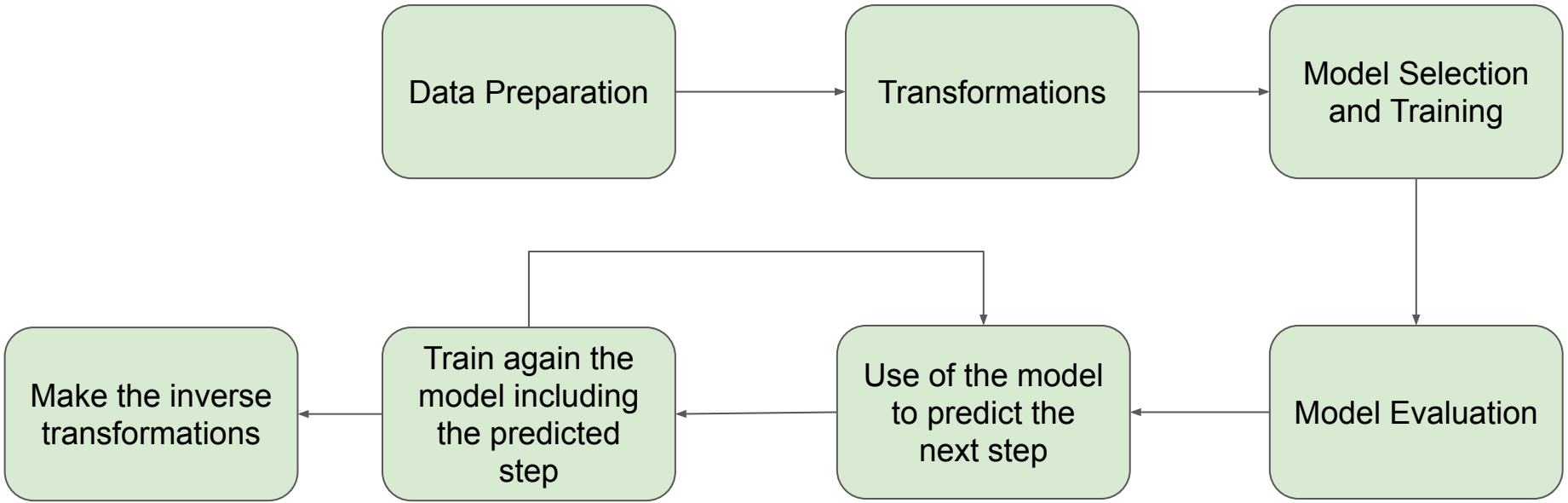
Time series - natural temporal ordering in data observations.

Time Series Analysis

Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data.

The objective of Time Series Analysis is Time series **forecasting**, i.e. the use of a model to predict future values based on previously observed values.

Time Series Analysis Workflow



Data Preparation

Create a time series from a dataset

```
import pandas as pd
df = pd.read_csv('data/tourist_arrivals.csv')
df['date'] = pd.to_datetime(df['date'])
df.set_index('date', inplace=True)
ts = df['value']
```

Transformations

Some data preparation techniques to consider include:

- Decomposition
- Differencing
- Power Transform
- *Standardize to center*
- *Normalize to rescale*

Time Series **Decomposition**

Trend T refers to the general direction in which the time series is moving. Time series can have a positive or a negative trend, but can also have no trend. A trend exists when there is a persistent increasing or decreasing direction in the data.

Seasonality S reflects seasonal variation. A seasonal pattern exists when a time series is influenced by seasonal factors.

Noise N describes random, irregular influences. It represents the residuals or remainder of the time series after the other components have been removed.

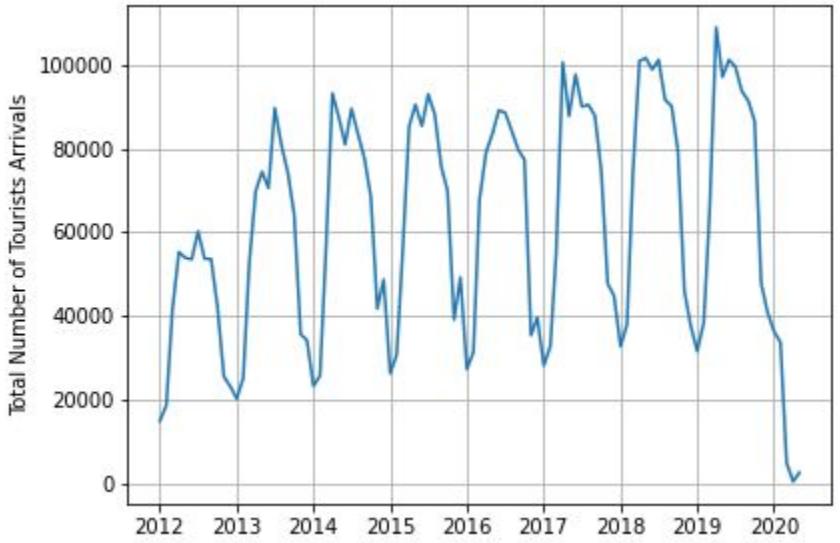
Assumptions

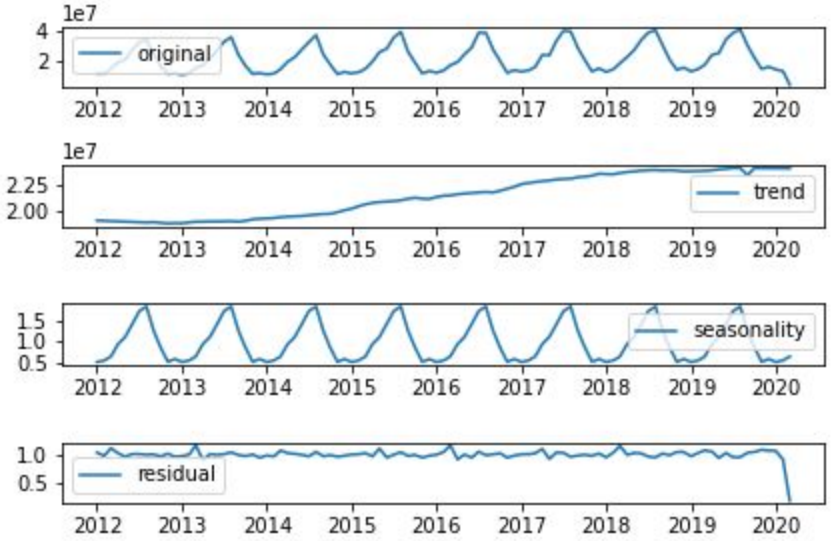
Decomposition assumes that the seasonal component is constant throughout the entire series.

While this may be an accurate assumption for short time periods, this assumption becomes untenable for longer periods.

Source: Different Types of Time Series Decomposition

Example


Total Number of Tourists Arrivals
100000
80000
60000
40000
20000
0
2012
2013
2014
2015
2016
2017
2018
2019
2020


1e7
4
2
original
2012
2013
2014
2015
2016
2017
2018
2019
2020
1e7
2.25
2.00
trend
1.5
1.0
0.5
seasonality
1.0
0.5
residual

Multiplicative VS Additive Decomposition

Multiplicative

$ts = T \times S \times N$

Additive

$ts = T + S + N$

Decomposition in Python

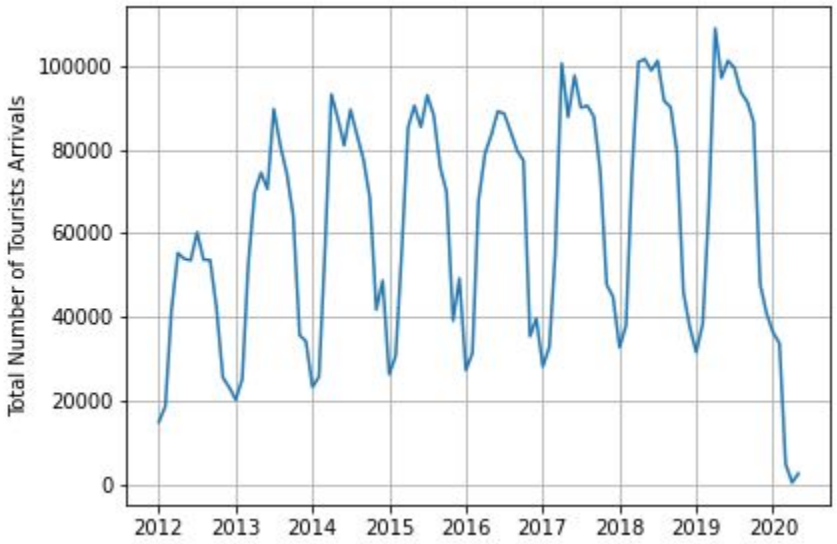
```
from statsmodels.tsa.seasonal import seasonal_decompose

decomposition = seasonal_decompose(ts, model='additive')

ts_trend = decomposition.trend

ts_seasonal = decomposition.seasonal

ts_residual = decomposition.resid
```

Time Series **Differencing**

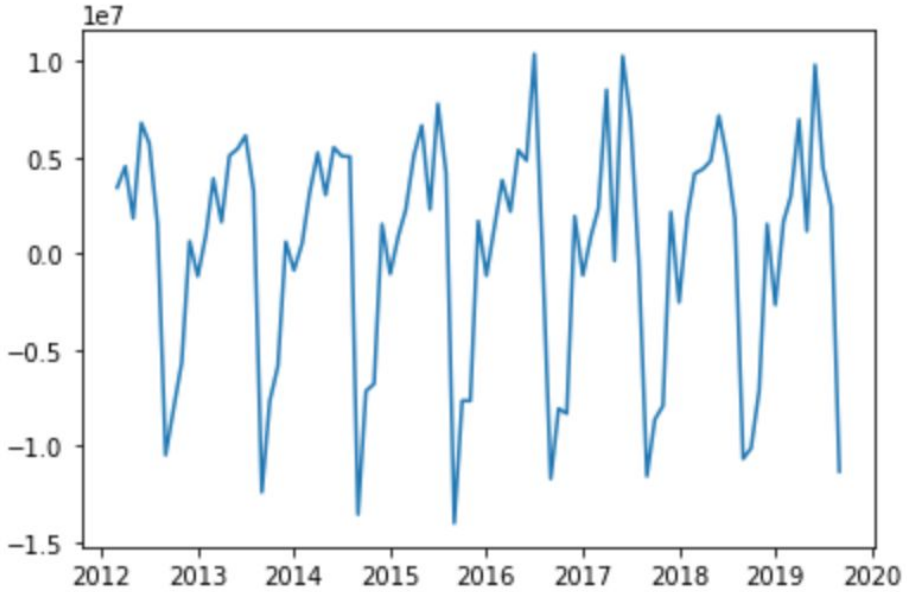
Differencing is a method of transforming a time series dataset.

It can be used to remove the series dependence on time, so-called temporal dependence. This includes structures like trends and seasonality.

Example

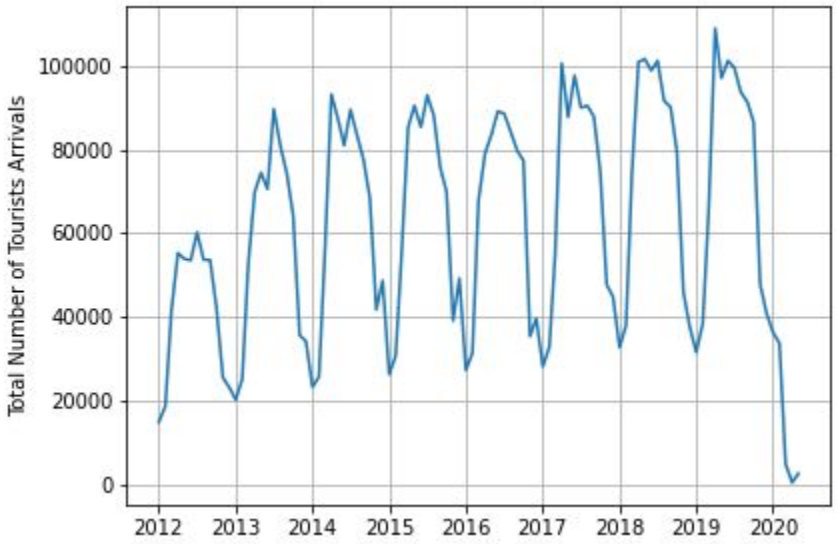


REMOVE TREND

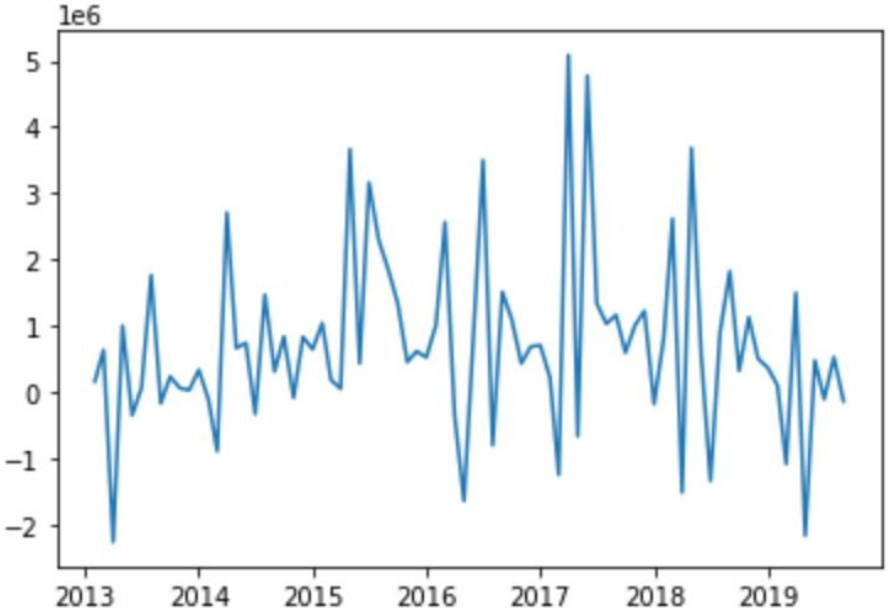



```
ts_diff = ts.diff()
```

Example (cont.)



REMOVE SEASONALITY




```
ts_diff = ts.diff(periods=12)
```

Comparison between diff and diff(periods=12)

date	value	diff	diff_12
2012-02-01	10468842	NaN	NaN
2012-03-01	13908950	3440108.0	NaN
2012-04-01	18456089	4547139.0	NaN
2012-05-01	20294254	1838165.0	NaN
2012-06-01	27101300	6807046.0	NaN
2012-07-01	32838284	5736984.0	NaN
2012-08-01	34392050	1553766.0	NaN
2012-09-01	23910073	-10481977.0	NaN
2012-10-01	15828202	-8081871.0	NaN
2012-11-01	10155960	-5672242.0	NaN
2012-12-01	10804312	648352.0	NaN
2013-01-01	9632532	-1171780.0	NaN
2013-02-01	10628786	996254.0	159944.0
2013-03-01	14540671	3911885.0	631721.0

Recursive Differencing

Apply differencing on a differenced time series

The number of times that the differencing is performed is called the **difference order**.

In Python recursive differencing is obtained through `np.diff(ts, n=2)`, where n is the difference order.

np.diff() VS ts.diff()

Month	#Passengers	diff_1	diff_2	diff_o1	diff_o2
1949-01	112	NaN	NaN	NaN	NaN
1949-02	118	6.0	NaN	6.0	NaN
1949-03	132	14.0	20.0	14.0	8.0
1949-04	129	-3.0	11.0	-3.0	-17.0
1949-05	121	-8.0	-11.0	-8.0	-5.0
...	...	...	...	...	...
1960-08	606	-16.0	71.0	-16.0	-103.0
1960-09	508	-98.0	-114.0	-98.0	-82.0
1960-10	461	-47.0	-145.0	-47.0	51.0
1960-11	390	-71.0	-118.0	-71.0	-24.0
1960-12	432	42.0	-29.0	42.0	113.0

Source: [An intuitive guide to differencing time series in Python]

Differencing is done to make a time series stationary.

Stationary Time Series

A stationary time series is a time series that has stable/constant statistical properties (mean, variance, etc.) over time.

The presence of a trend makes the mean of the time series changing over time.

The presence of a seasonality makes the variance varying over time.

Long-term Forecasting is possible on a stationary time series.

Dickey-Fuller Test to test if a TS is Stationary

Dickey-Fuller Test

```
from statsmodels.tsa.stattools import adfuller

dftest = adfuller(timeseries)
```

output:

```
pvalue, adf, critical values
```

The Time Series is stationary if

pvalue < 0.05 and adf < critical_values

Power Transform

- Square Root
- Log
- Box Cox
- …

More details at How to Use Power Transforms for Time Series Forecast Data with Python

Decomposition VS Differencing

Decomposition → forecast is done on every component of the time series and then the total forecast is calculated by composing the obtained predictions

Differencing → forecast is done on the differenced time series and then the inverse differencing operation is done to build the forecast of the original time series

Types of Time Series Analysis

Time Domain

- cross correlation
- autocorrelation
- partial autocorrelation

Frequency Domain

- spectral analysis
- wavelet analysis

Cross Correlation

Often, two time series vary together.

Cross Correlation refers to the study of the correlation between two time series.

The **correlation coefficient** is a measure of how much two series vary together.

High correlations mean that the two series strongly vary together.

A low correlation means they vary together, but there is a weak association.

Cross Correlation of Trending Time Series

Consider two time series that are both trending.

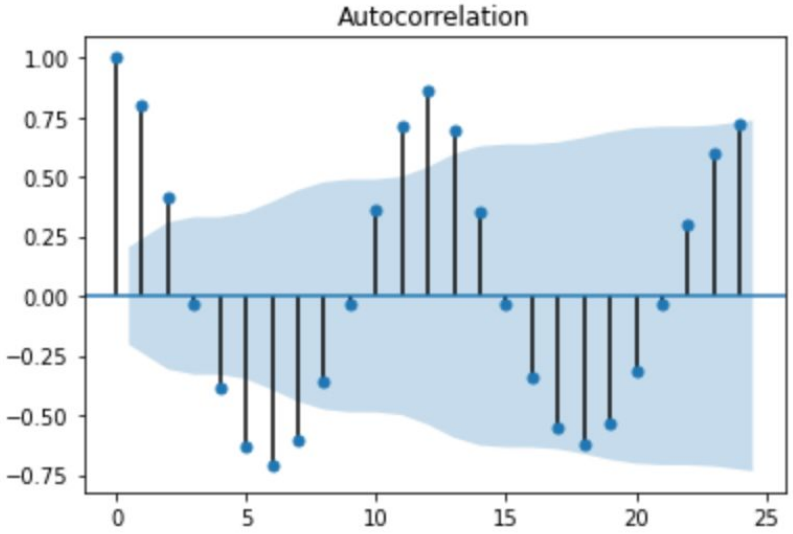
Even if the two series are totally unrelated, you could still get a very high correlation.

Do not calculate correlation between the trends of two time series!

Autocorrelation (ACF) or Serial Correlation

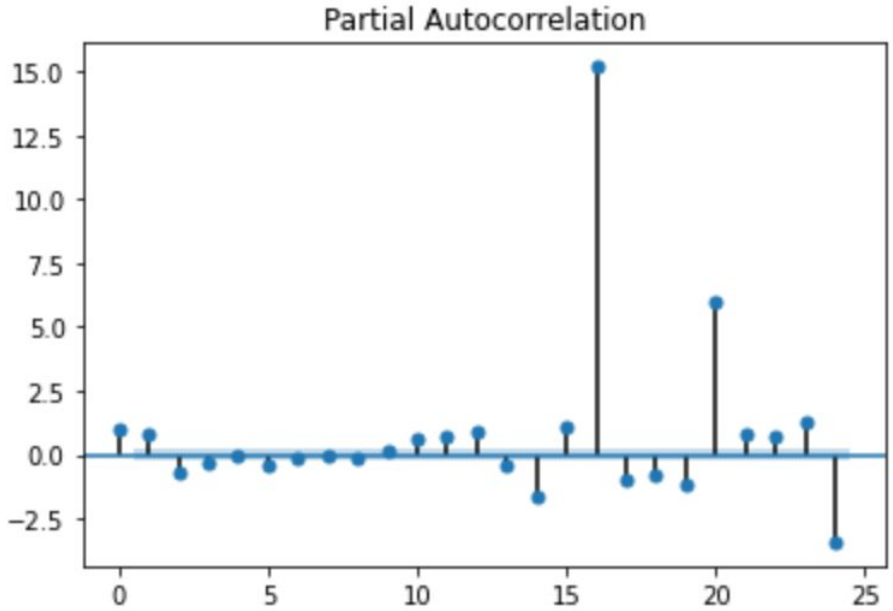
Autocorrelation is the correlation of a single time series with a lagged copy of itself.

Informalmente, è la similarità tra un’osservazione e la successiva, come funzione del ritardo (lag) considerato.



Partial Autocorrelation (PACF)

Partial autocorrelation can be imagined as the correlation between the series and its lag, after excluding the contributions from the intermediate lags.



TS Modelling in the Time Domain

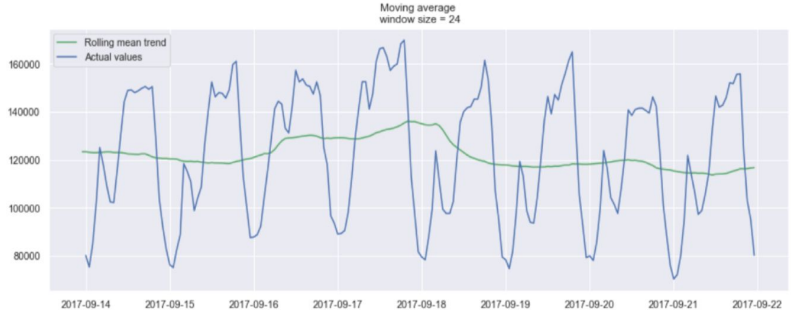
Time Series Modelling can be done only if the Time Series is stationary

Main Models

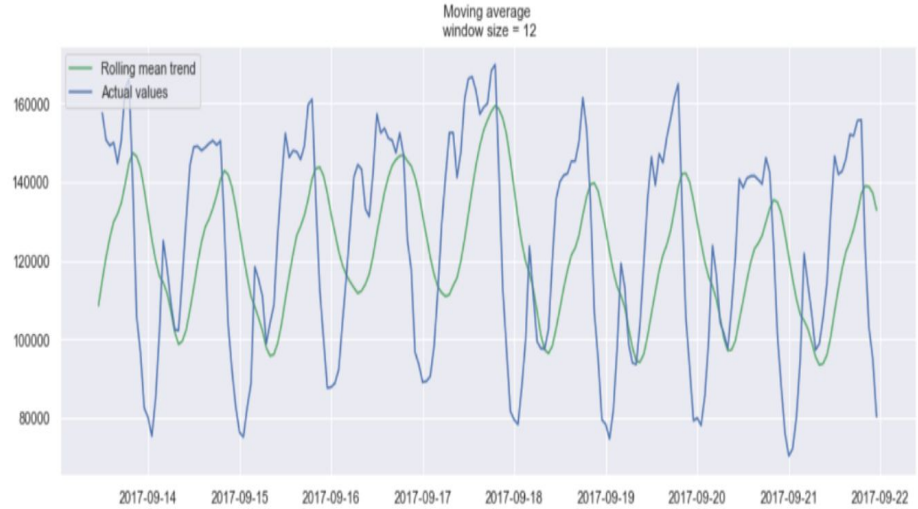
- Moving Average
- Exponential Smoothing
- Auto Regressive
- ARMA
- ARIMA
- SARIMA

Moving Average

The next observation is the mean of all past observations.



Example of a moving average on a 24h window



Example of a moving average on a 12h window

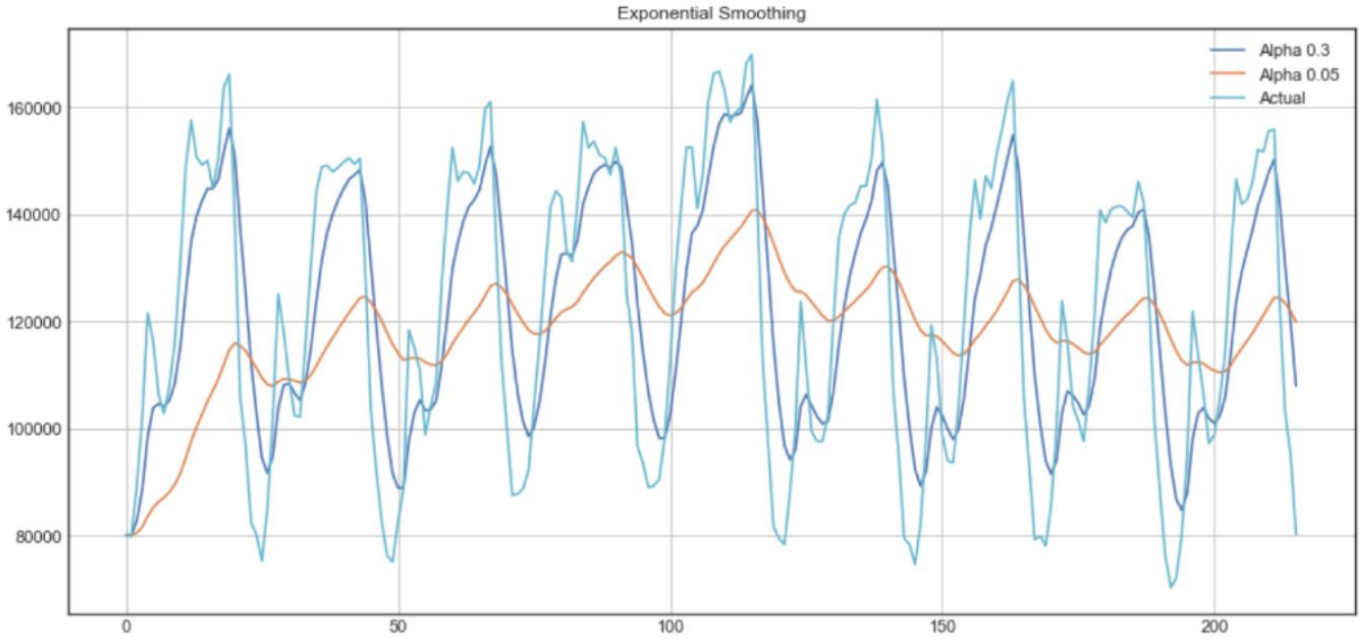
Source: [The Complete Guide to Time Series Analysis and Forecasting]

Exponential Smoothing

Exponential smoothing uses a similar logic to moving average, but this time, a different decreasing weight is assigned to each observations.

alpha is a smoothing factor that takes values between 0 and 1. It determines how fast the weight decreases for previous observations.

Example



Example of exponential smoothing

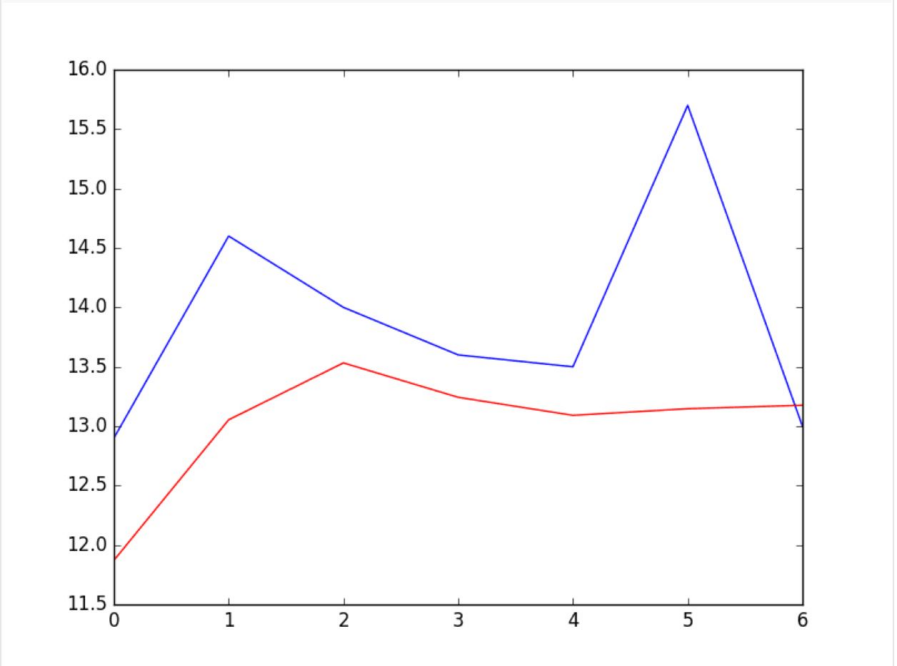
Autoregressive

The next observation depends linearly on the previous observations.

Prediction = red

Actual = blue

Source [Autoregression Models for Time Series Forecasting With Python]



ARMA, ARIMA, SARIMA

ARMA = Autoregressive (AR) + Moving Average (MA)

ARIMA = AR + Integrated Order (I) + MA

ARIMA can be used when the time series is not seasonal

SARIMA = Seasonal ARIMA

ARIMA

A non seasonal ARIMA model is classified as an "**ARIMA(p,d,q)**" model, where:

p is the order of the AR term,

d is the number of non seasonal differences needed for stationarity, and

q is the order of the MA term.

How to calculate p,d, q

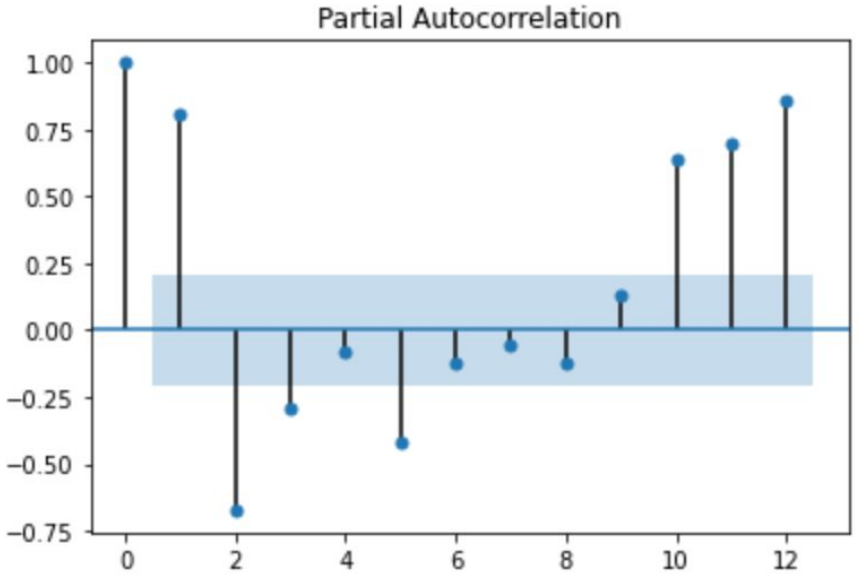
- q = look at an autocorrelation graph of the stationary/stationarized ts
- p = look at a partial autocorrelation graph of the stationary/stationarized ts
- Akaike's Information Criterion (AIC) on a set of models and investigate the models with the lowest AIC values
- Schwartz Bayesian Information Criterion (BIC) and investigate the models with the lowest BIC values

More details at: ARIMA Model – Complete Guide to Time Series Forecasting in Python

Example for p

Look at PACF

try p = 1, 2, 3 and take the best output



Example in Python

```
from statsmodels.tsa.arima_model import ARIMA

p = 1

q = 0

d = 1

model = ARIMA(ts, order=(p,d,q))

model_fit = model.fit()

predictions = model_fit.forecast(steps=1)
```

How to Determine p,d and q automatically

Python library pmdarima

```
pip install pmdarima
```

use function auto_arima

Example

```
stepwise_model = auto_arima(data,
                            start_p=1, d=1, start_q=1,
                            max_p=3, max_d=1, max_q=3,
                            m=12,
                            seasonal=True, stationary=False,
                            information_criterion='aic',
                            stepwise=True,
                            n_jobs=-1,
                            maxiter=10)
stepwise_model.fit(data)
stepwise_model.predict(n_periods=1)
```

Model Evaluation

MAPE (Mean Absolute Percentage Error): errore percentuale medio assoluto.

ME (Mean Error): errore medio.

MPE (Mean Percentage Error): errore percentuale medio.

NMRSE (Normalized Root Mean Square Error): misura la differenza tra i valori previsti e i valori osservati.

More details at: MODEL EVALUATION

Summary

Plotting data to see if any trend or seasonality is present or not.

Check for stationarity by using methods such as the Augmented Dickey-Fuller (ADF) Test

If stationarity is not found use transformations. If stationarity is not achieved then find the order of differencing by using ACF plots and execute differencing to make the data stationary

Fit the model

Evaluate the model

Use the model for new predictions

Example
Thesis by Sofia Montagna

Analisi dell’impatto del COVID-19 sull’andamento dei decessi in Italia e in Europa

Tesi

- Il numero di decessi attribuiti al COVID-19 non è sufficiente a giustificare l'eccesso di mortalità registrato.
- I decessi per COVID-19 sono stati **sottostimati**.

Cause

- Strutture sanitarie sovraccariche e mancanza di personale sanitario.
- Mancanza di tamponi molecolari.
- Mancanza di diagnosi.

Obiettivo della tesi

- Creare un **modello predittivo** in grado di prevedere il numero di decessi attesi per l'anno 2020 in condizioni di normalità sulla base dei decessi storici mensili (2015-2019).
- Stimare i **decessi in eccesso** (**SDE**) come la differenza tra i decessi osservati e i decessi attesi secondo il modello.
- Stimare i **decessi COVID-19 sommersi** (**SDS**) come la differenza tra la stima dei decessi in eccesso e il numero ufficiale di decessi per COVID-19.

Raccolta dei dati

Decessi totali registrati in Italia	ISTAT	gennaio 2015 - settembre 2020
Decessi COVID-19 nazionali	GitHub - Protezione Civile	marzo 2020 - settembre 2020
Decessi COVID-19 regionali	GitHub - Protezione Civile	marzo 2020 - settembre 2020
Decessi totali registrati in Europa	EUROSTAT	gennaio 2015 - settembre 2020
Decessi COVID-19 registrati in Europa	GitHub - Johns Hopkins University Center	marzo 2020 - settembre 2020

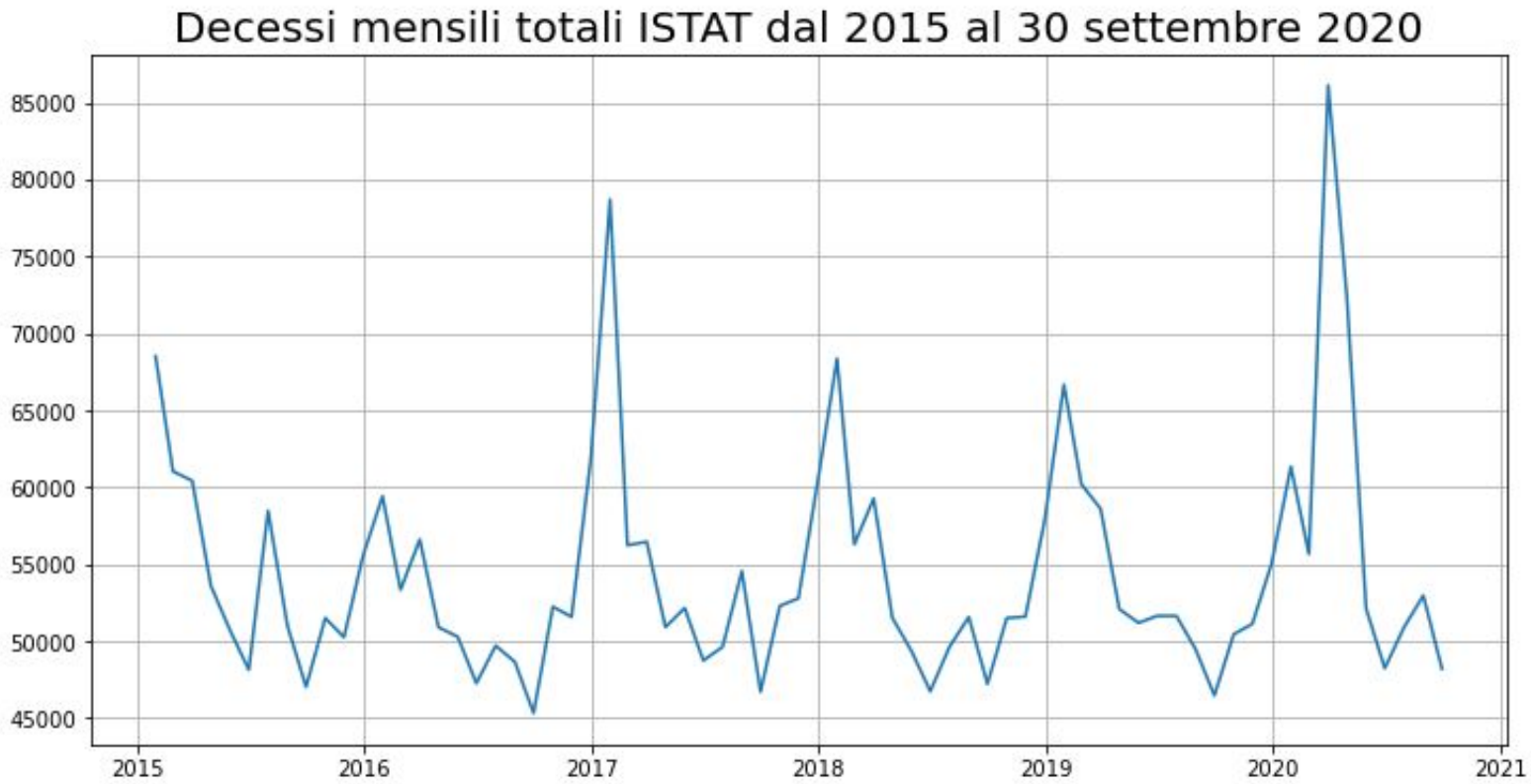
Analisi dei dati

- I dati raccolti sono stati interpretati come **serie storiche** con l'obiettivo di creare un modello predittivo.

***Serie storica**: insieme di valori ordinati rispetto al tempo che esprimono la dinamica di un certo fenomeno.*

- Il procedimento può essere generalizzato a tutte le serie storiche analizzate.

Studio di una serie storica



Fase 1: analisi prima della pandemia

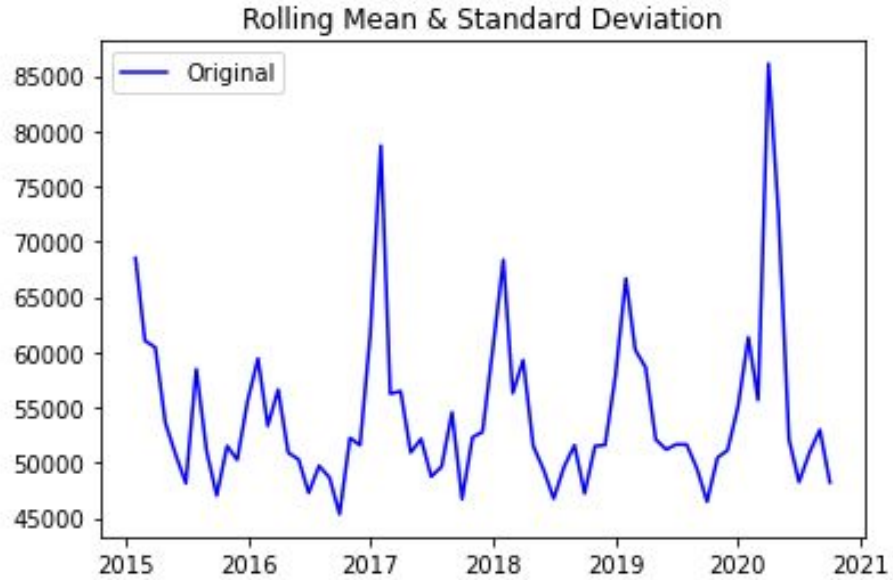
Analisi della serie storica prima di un **evento di rottura** (inizio della pandemia).

Studio delle caratteristiche:

- Presenza di **stagionalità**.
- Verifica che la serie storica sia stazionaria.

Dickey-Fuller Test

- Esito negativo: la serie necessita di essere modificata (**differenziazione**).




```
Results of Dickey-Fuller Test:
Test Statistic                  -1.305184
p-value                          0.626797
#Lags Used                      11.000000
Number of Observations Used     57.000000
Critical Value (1%)             -3.550670
Critical Value (5%)             -2.913766
Critical Value (10%)            -2.594624
dtype: float64
X is not stationary
```

Scelta del modello predittivo

Modello **SARIMA** (*Seasonal Autoregressive Integrated Moving Average*)

- Restituisce i valori futuri della serie storica sulla base dei suoi valori passati.
- Tiene conto della **stagionalità**.
- Riceve in ingresso tre variabili: **SARIMA(p,d,q)**

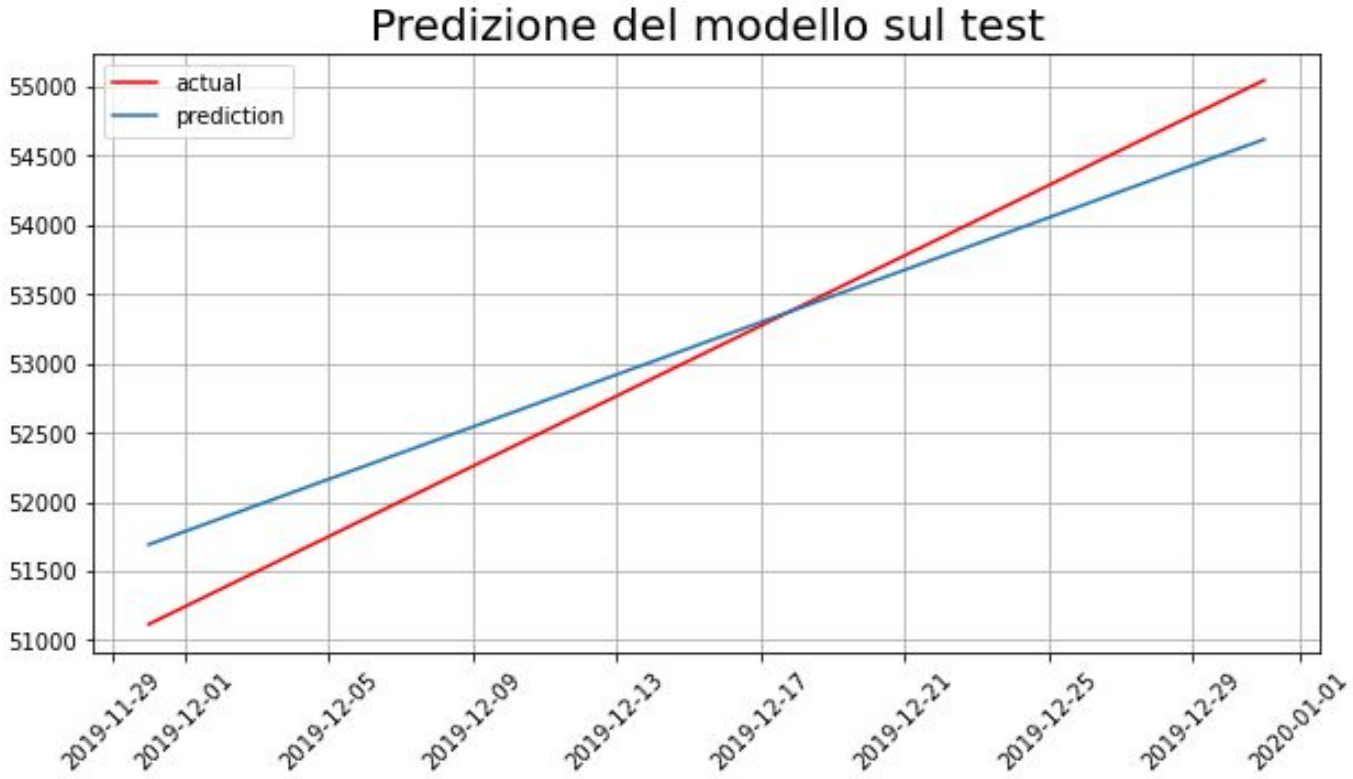
Modello SARIMA(12,1,1)

Suddivisione della serie storica:

- **Train (gennaio 2015 - ottobre 2019)**: parte di serie storica in cui è stato allenato il modello SARIMA.
- **Test (novembre - dicembre 2019)**: parte di serie storica in cui è stato valutato il modello.

Train del modello

Test del modello



Metriche di valutazione del modello

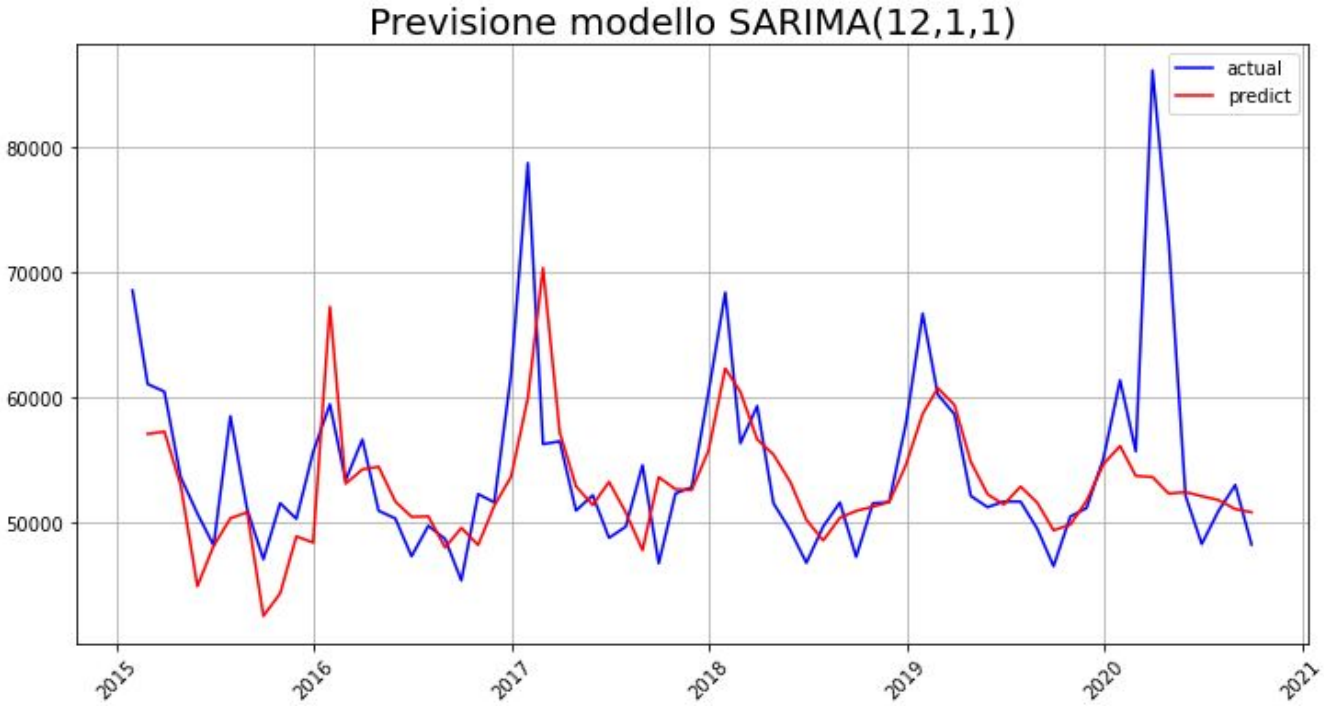
Misurano gli errori emessi dal modello:

- **MAPE**: errore percentuale medio assoluto.
- **ME**: errore medio.
- **MPE**: errore percentuale medio.
- **NMRSE**: misura la differenza tra i valori previsti e i valori osservati.

Nazione	Modello	Metriche di accuratezza			
		MAPE	ME	MPE	NRMSE
Italia	SARIMA(12,1,1)	0,001	74	0,001	0,13

Fase 2: analisi dall'inizio della pandemia

Estendere la predizione fino al mese di settembre 2020.



Calcolo della metrica SDE

SDE (stima decessi in eccesso): decessi registrati in più per il periodo da marzo a settembre 2020, rispetto alla previsione.

SDE (medio e massimo) = decessi osservati - decessi attesi (val. medio e minimo)

Calcolo della metrica SDS

SDS (stima dei decessi sommersi): indica il possibile numero di decessi per COVID-19 non registrati dalle fonti ufficiali.

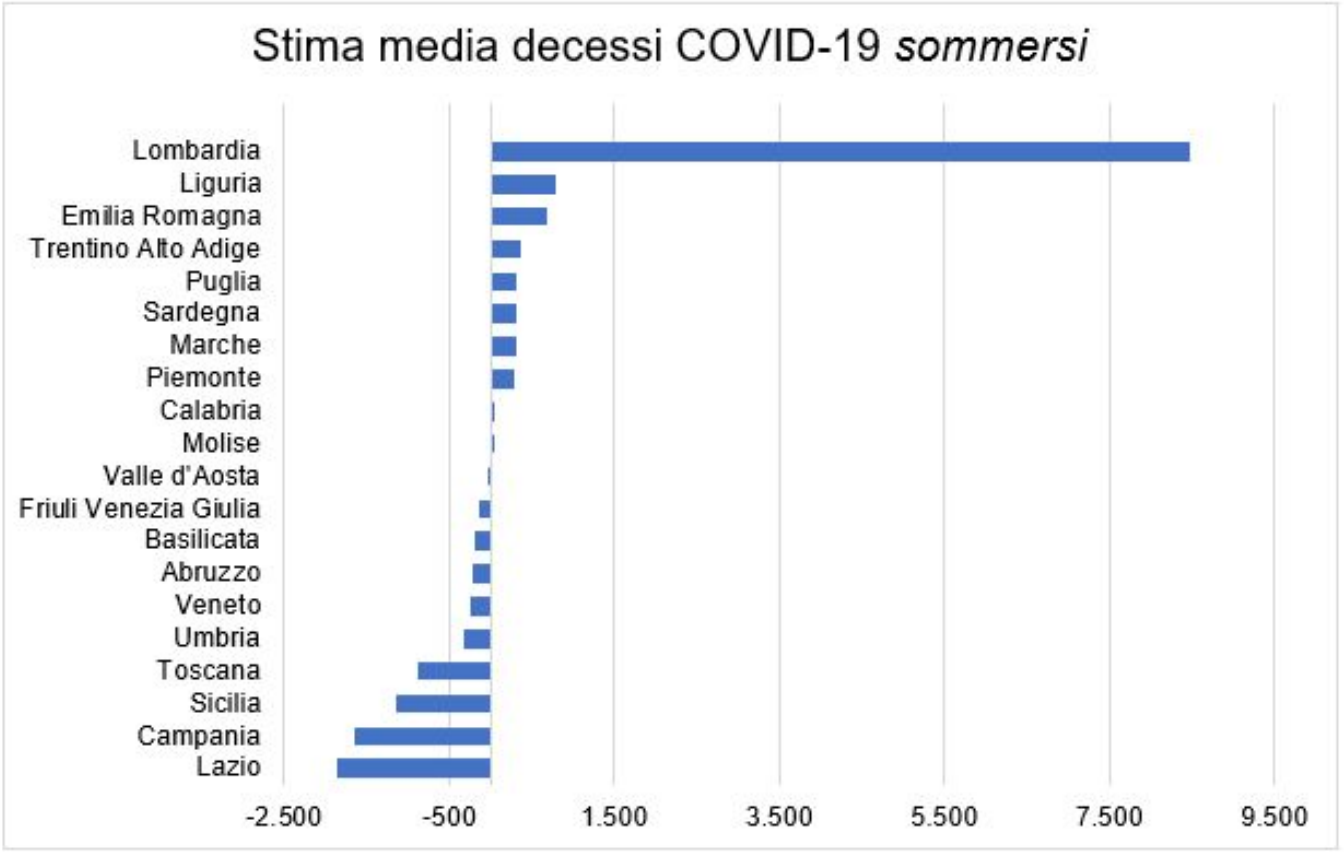
SDS (medio e massimo) = SDE (medio e massimo) - decessi COVID-19 registrati da marzo a settembre 2020

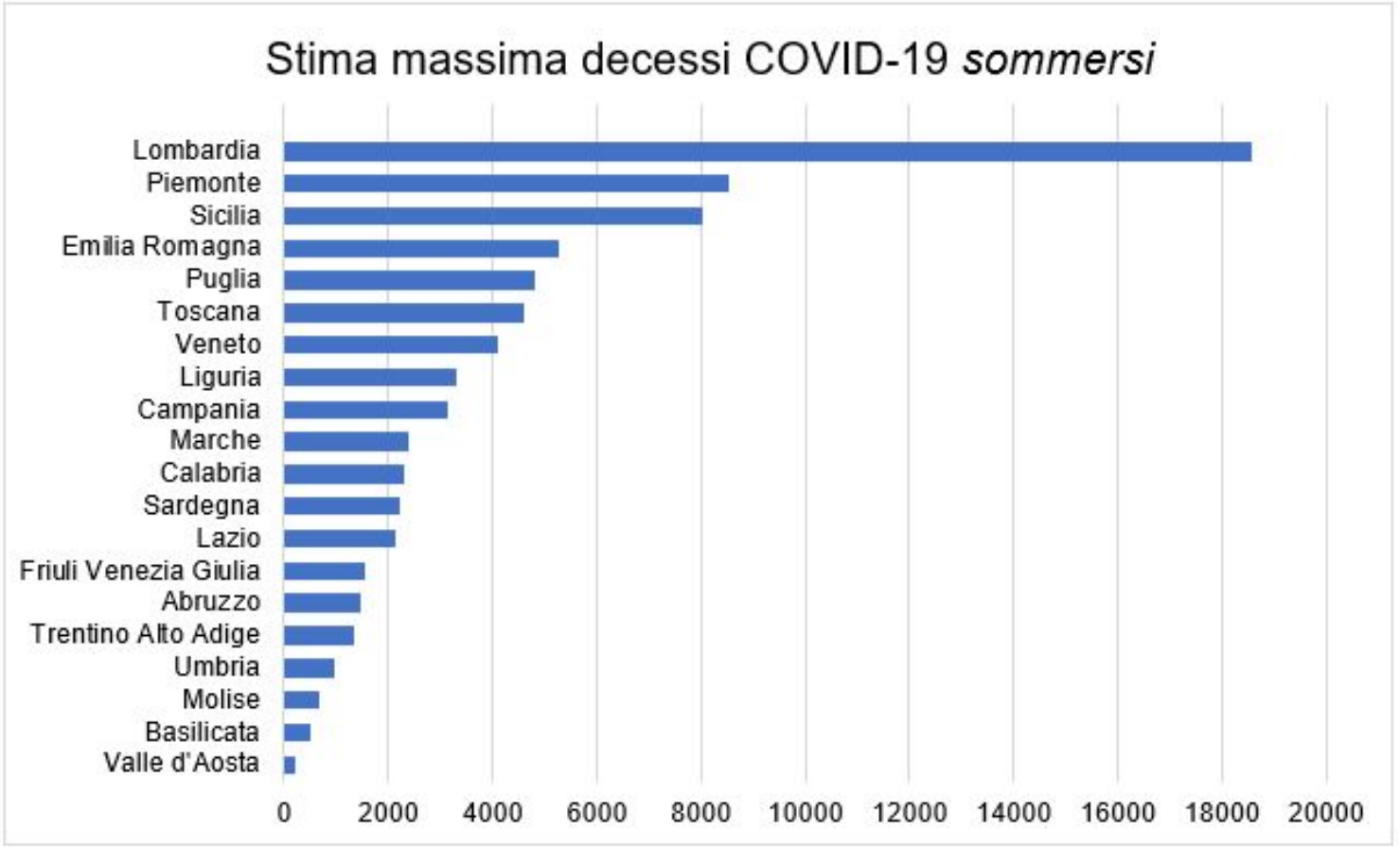
Risultati per l'Italia

Nazione	Decessi totali registrati	Decessi totali attesi dal modello		SDE: stima decessi in eccesso		Decessi accertati Covid-19	SDS: stima numero decessi COVID-19 sommersi	
		Val. medio	Val. min	Val. medio	Val. max		Val. medio	Val. max
Italia	410.899	363.891	289.426	47.008	121.473	35.865	11.143	85.608

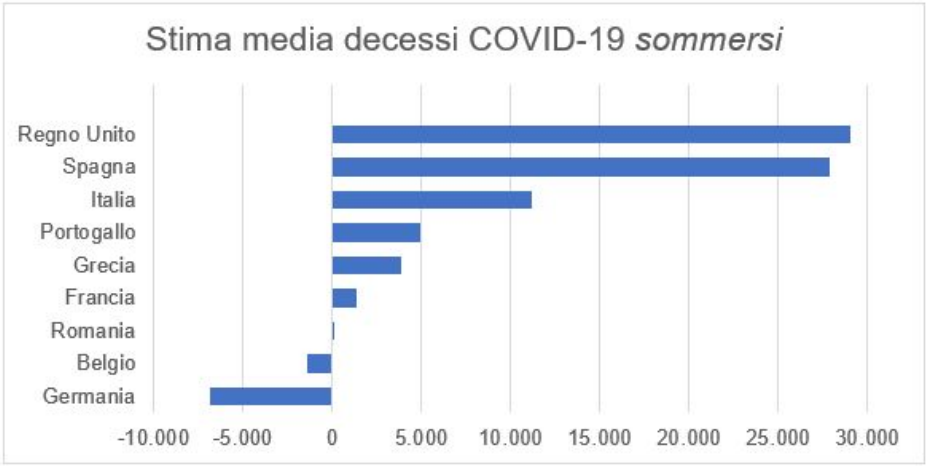
- Rispetto a quanto atteso in condizioni di normalità sono stati stimati in media 47.008 decessi in eccesso fino a un massimo di 121.473.
- I **decessi sommersi** ammontano a un valore medio di 11.143 fino a un massimo di 85.608 decessi.

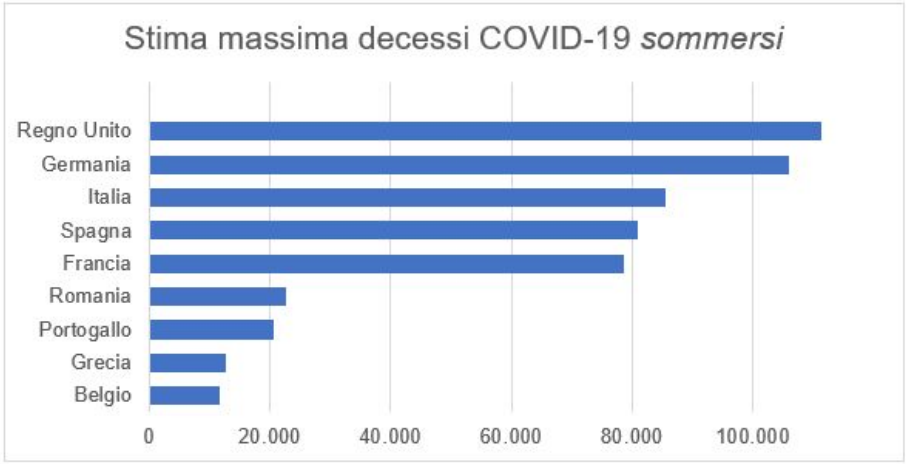
Risultati regionali




Stima massima decessi COVID-19 *sommersi*
Lombardia
Piemonte
Sicilia
Emilia Romagna
Puglia
Toscana
Veneto
Liguria
Campania
Marche
Calabria
Sardegna
Lazio
Friuli Venezia Giulia
Abruzzo
Trentino Alto Adige
Umbria
Molise
Basilicata
Valle d'Aosta
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000

Risultati analisi europa





I risultati sono riportati in **valore assoluto** non tenendo conto della popolazione delle nazioni analizzate.

Interesting readings

3 facts about time series forecasting that surprise experienced machine learning practitioners.

Time Series Analysis and Its Applications