# ****Complete Guide on Time Series Analysis in Python****[¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#Complete-Guide-on-Time-Series-Analysis-in-Python)

# ****1. Introduction to Time-Series Analysis**** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#1.-Introduction-to-Time-Series-Analysis-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* A **time-series** data is a series of data points or observations recorded at different or regular time intervals. In general, a time series is a sequence of data points taken at equally spaced time intervals. The frequency of recorded data points may be hourly, daily, weekly, monthly, quarterly or annually.
* **Time-Series Forecasting** is the process of using a statistical model to predict future values of a time-series based on past results.
* A time series analysis encompasses statistical methods for analyzing time series data. These methods enable us to extract meaningful statistics, patterns and other characteristics of the data. Time series are visualized with the help of line charts. So, time series analysis involves understanding inherent aspects of the time series data so that we can create meaningful and accurate forecasts.
* Applications of time series are used in statistics, finance or business applications. A very common example of time series data is the daily closing value of the stock index like NASDAQ or Dow Jones. Other common applications of time series are sales and demand forecasting, weather forecasting, econometrics, signal processing, pattern recognition and earthquake prediction.

### **Components of a Time-Series**

* **Trend** - The trend shows a general direction of the time series data over a long period of time. A trend can be increasing(upward), decreasing(downward), or horizontal(stationary).
* **Seasonality** - The seasonality component exhibits a trend that repeats with respect to timing, direction, and magnitude. Some examples include an increase in water consumption in summer due to hot weather conditions.
* **Cyclical Component** - These are the trends with no set repetition over a particular period of time. A cycle refers to the period of ups and downs, booms and slums of a time series, mostly observed in business cycles. These cycles do not exhibit a seasonal variation but generally occur over a time period of 3 to 12 years depending on the nature of the time series.
* **Irregular Variation** - These are the fluctuations in the time series data which become evident when trend and cyclical variations are removed. These variations are unpredictable, erratic, and may or may not be random.
* **ETS Decomposition** - ETS Decomposition is used to separate different components of a time series. The term ETS stands for Error, Trend and Seasonality.
* In this notebook, I conduct time series analysis of video game sales over time.

# ****2. Types of data**** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#2.-Types-of-data-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

As stated above, the time series analysis is the statistical analysis of the time series data. A time series data means that data is recorded at different time periods or intervals. The time series data may be of three types:-

1 **Time series data** - The observations of the values of a variable recorded at different points in time is called time series data.

2 **Cross sectional data** - It is the data of one or more variables recorded at the same point in time.

3 **Pooled data**- It is the combination of time series data and cross sectional data.

# ****3. Time Series terminology**** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#3.-Time-Series-terminology-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

There are various terms and concepts in time series that we should know. These are as follows:-

1 **Dependence**- It refers to the association of two observations of the same variable at prior time periods.

2 **Stationarity**- It shows the mean value of the series that remains constant over the time period. If past effects accumulate and the values increase towards infinity then stationarity is not met.

3 **Differencing**- Differencing is used to make the series stationary and to control the auto-correlations. There may be some cases in time series analyses where we do not require differencing and over-differenced series can produce wrong estimates.

4 **Specification** - It may involve the testing of the linear or non-linear relationships of dependent variables by using time series models such as ARIMA models.

5 **Exponential Smoothing** - Exponential smoothing in time series analysis predicts the one next period value based on the past and current value. It involves averaging of data such that the non-systematic components of each individual case or observation cancel out each other. The exponential smoothing method is used to predict the short term prediction.

6 **Curve fitting** - Curve fitting regression in time series analysis is used when data is in a non-linear relationship.

7 **ARIMA** - ARIMA stands for Auto Regressive Integrated Moving Average.

# ****4. Patterns in a Time Series**** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#6.-Patterns-in-a-Time-Series-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* Any time series visualization may consist of the following components: **Base Level + Trend + Seasonality + Error**.

### **Trend**

* A **trend** is observed when there is an increasing or decreasing slope observed in the time series.

### **Seasonality**

* A **seasonality** is observed when there is a distinct repeated pattern observed between regular intervals due to seasonal factors. It could be because of the month of the year, the day of the month, weekdays or even time of the day.

However, It is not mandatory that all time series must have a trend and/or seasonality. A time series may not have a distinct trend but have a seasonality and vice-versa.

# ****7. Additive and Multiplicative Time Series****

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* We may have different combinations of trends and seasonality. Depending on the nature of the trends and seasonality, a time series can be modeled as an additive or multiplicative time series. Each observation in the series can be expressed as either a sum or a product of the components.

### **Additive time series:**

Value = Base Level + Trend + Seasonality + Error

### **Multiplicative Time Series:**

Value = Base Level x Trend x Seasonality x Error

**8. Decomposition of a Time Series** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#8.-Decomposition-of-a-Time-Series-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* Decomposition of a time series can be performed by considering the series as an additive or multiplicative combination of the base level, trend, seasonal index and the residual term.
* The seasonal\_decompose in statsmodels implements this conveniently.

In [7]:

**9. Stationary and Non-Stationary Time Series** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#9.-Stationary-and-Non-Stationary-Time-Series-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* Now, we wil discuss **Stationary and Non-Stationary Time Series**. **Stationarity** is a property of a time series. A stationary series is one where the values of the series is not a function of time. So, the values are independent of time.
* Hence the statistical properties of the series like mean, variance and autocorrelation are constant over time. Autocorrelation of the series is nothing but the correlation of the series with its previous values.
* A stationary time series is independent of seasonal effects as well.
* Now, we will plot some examples of stationary and non-stationary time series for clarity.

**10. How to make a time series stationary?** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#10.-How-to-make-a-time-series-stationary?-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* We can apply some sort of transformation to make the time-series stationary. These transformation may include:

1. Differencing the Series (once or more)
2. Take the log of the series
3. Take the nth root of the series
4. Combination of the above

* The most commonly used and convenient method to stationarize the series is by differencing the series at least once until it becomes approximately stationary.

## **10.1 Introduction to Differencing**[**¶**](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#10.1-Introduction-to-Differencing-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* If Y\_t is the value at time t, then the first difference of Y = Yt – Yt-1. In simpler terms, differencing the series is nothing but subtracting the next value by the current value.
* If the first difference doesn’t make a series stationary, we can go for the second differencing and so on.
* For example, consider the following series: [1, 5, 2, 12, 20]
* First differencing gives: [5-1, 2-5, 12-2, 20-12] = [4, -3, 10, 8]
* Second differencing gives: [-3-4, -10-3, 8-10] = [-7, -13, -2]

## **10.2 Reasons to convert a non-stationary series into stationary one before forecasting**[**¶**](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#10.2-Reasons-to-convert-a-non-stationary-series-into-stationary-one-before-forecasting-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

There are reasons why we want to convert a non-stationary series into a stationary one. These are given below:

* Forecasting a stationary series is relatively easy and the forecasts are more reliable.
* An important reason is, autoregressive forecasting models are essentially linear regression models that utilize the lag(s) of the series itself as predictors.
* We know that linear regression works best if the predictors (X variables) are not correlated against each other. So, stationarizing the series solves this problem since it removes any persistent autocorrelation, thereby making the predictors(lags of the series) in the forecasting models nearly independent.

**11. How to test for stationarity?** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#11.-How-to-test-for-stationarity?-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* The stationarity of a series can be checked by looking at the plot of the series.
* Another method is to split the series into 2 or more contiguous parts and computing the summary statistics like the mean, variance and the autocorrelation. If the stats are quite different, then the series is not likely to be stationary.
* There are several quantitative methods we can use to determine if a given series is stationary or not. This can be done using statistical tests called [Unit Root Tests](https://en.wikipedia.org/wiki/Unit_root). This test checks if a time series is non-stationary and possess a unit root.
* There are multiple implementations of Unit Root tests like:

**1. Augmented Dickey Fuller test (ADF Test)**

**2. Kwiatkowski-Phillips-Schmidt-Shin – KPSS test (trend stationary)**

**3. Philips Perron test (PP Test)**

## **11.1 Augmented Dickey Fuller test (ADF Test)**[**¶**](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#11.1-Augmented-Dickey-Fuller-test-(ADF-Test)-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* **Augmented Dickey Fuller test or (ADF Test)** is the most commonly used test to detect stationarity. Here, we assume that the null hypothesis is the time series possesses a unit root and is non-stationary. Then, we collect evidence to support or reject the null hypothesis. So, if we find that the p-value in ADF test is less than the significance level (0.05), we reject the null hypothesis.
* Feel free to check the following links to learn more about the ADF Test.

**12. Difference between white noise and a stationary series** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#12.-Difference-between-white-noise-and-a-stationary-series-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* Like a stationary series, the white noise is also not a function of time. So, its mean and variance does not change over time. But the difference is that, the white noise is completely random with a mean of 0. In white noise there is no pattern.
* Mathematically, a sequence of completely random numbers with mean zero is a white noise.

In [8]:

**13. Detrend a Time Series** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#13.-Detrend-a-Time-Series-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* Detrending a time series means to remove the trend component from the time series. There are multiple approaches of doing this as listed below:

1. Subtract the line of best fit from the time series. The line of best fit may be obtained from a linear regression model with the time steps as the predictor. For more complex trends, we may want to use quadratic terms (x^2) in the model.
2. We subtract the trend component obtained from time series decomposition.
3. Subtract the mean.
4. Apply a filter like Baxter-King filter(statsmodels.tsa.filters.bkfilter) or the Hodrick-Prescott Filter (statsmodels.tsa.filters.hpfilter) to remove the moving average trend lines or the cyclical components.

Now, we will implement the first two methods to detrend a time series.

In [9]:

**14. Deseasonalize a Time Series** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#14.-Deseasonalize-a-Time-Series-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

There are multiple approaches to deseasonalize a time series. These approaches are listed below:

* 1. Take a moving average with length as the seasonal window. This will smoothen in series in the process.
  2. Seasonal difference the series (subtract the value of previous season from the current value).
  3. Divide the series by the seasonal index obtained from STL decomposition.

If dividing by the seasonal index does not work well, we will take a log of the series and then do the deseasonalizing. We will later restore to the original scale by taking an exponential.

In [11]:

**15. How to test for seasonality of a time series?** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#15.-How-to-test-for-seasonality-of-a-time-series?-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

The common way to test for seasonality of a time series is to plot the series and check for repeatable patterns in fixed time intervals. So, the types of seasonality is determined by the clock or the calendar.

1. Hour of day
2. Day of month
3. Weekly
4. Monthly
5. Yearly

However, if we want a more definitive inspection of the seasonality, use the **Autocorrelation Function (ACF) plot**. There is a strong seasonal pattern, the ACF plot usually reveals definitive repeated spikes at the multiples of the seasonal window.

In [12]:

**16. Autocorrelation and Partial Autocorrelation Functions** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#16.-Autocorrelation-and-Partial-Autocorrelation-Functions-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* **Autocorrelation** is simply the correlation of a series with its own lags. If a series is significantly autocorrelated, that means, the previous values of the series (lags) may be helpful in predicting the current value.
* **Partial Autocorrelation** also conveys similar information but it conveys the pure correlation of a series and its lag, excluding the correlation contributions from the intermediate lags.

In [13]:

from statsmodels.tsa.stattools import acf, pacf

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

*# Draw Plot*

fig, axes = plt.subplots(1,2,figsize=(16,3), dpi= 100)

plot\_acf(df['Number of Passengers'].tolist(), lags=50, ax=axes[0])

plot\_pacf(df['Number of Passengers'].tolist(), lags=50, ax=axes[1])

**17. Computation of Partial Autocorrelation Function** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#17.-Computation-of-Partial-Autocorrelation-Function-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

* The partial autocorrelation function of lag (k) of a series is the coefficient of that lag in the autoregression equation of Y. The autoregressive equation of Y is nothing but the linear regression of Y with its own lags as predictors.
* For example, if **Y\_t** is the current series and **Y\_t-1** is the lag 1 of Y, then the partial autocorrelation of **lag 3 (Y\_t-3)** is the coefficient α3α3 of Y\_t-3 in the following equation:

**20. Smoothening a Time Series** [¶](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#20.-Smoothening-a-Time-Series-)

[Table of Contents](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#0.1)

Smoothening of a time series may be useful in the following circumstances:

* Reducing the effect of noise in a signal get a fair approximation of the noise-filtered series.
* The smoothed version of series can be used as a feature to explain the original series itself.
* Visualize the underlying trend better.

We can smoothen a time series using the following methods:

* Take a moving average
* Do a LOESS smoothing (Localized Regression)
* Do a LOWESS smoothing (Locally Weighted Regression)

## **Moving Average**[**¶**](https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook#Moving-Average)

* **Moving average** is the average of a rolling window of defined width. We must choose the window-width wisely, because, large window-size will over-smooth the series. For example, a window-size equal to the seasonal duration (ex: 12 for a month-wise series), will effectively nullify the seasonal effect.

## **Localized Regression**

* LOESS, short for ‘Localized Regression’ fits multiple regressions in the local neighborhood of each point. It is implemented in the statsmodels package, where you can control the degree of smoothing using frac argument which specifies the percentage of data points nearby that should be considered to fit a regression model.

**Read more on**

https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/data

**ARIMA Model for Time Series Forecasting**

# ****1. Introduction to Time Series Forecasting**** [¶](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#1.-Introduction-to-Time-Series-Forecasting-)

[Table of Contents](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#0.1)

* A **Time Series** is defined as a series of data points recorded at different time intervals. The time order can be daily, monthly, or even yearly.
* Time Series forecasting is the process of using a statistical model to predict future values of a time series based on past results.
* We have discussed various aspects of **Time Series Forecasting** in the previous notebook - [Complete Guide to Time Series Analysis in Python](https://www.kaggle.com/prashant111/complete-guide-on-time-series-analysis-in-python).
* Forecasting is the step where we want to predict the future values the series is going to take. Forecasting a time series is often of tremendous commercial value.

#### **Forecasting a time series can be broadly divided into two types.**

* If we use only the previous values of the time series to predict its future values, it is called **Univariate Time Series Forecasting.**
* If we use predictors other than the series (like exogenous variables) to forecast it is called **Multi Variate Time Series Forecasting.**
* This notebook focuses on a particular type of forecasting method called **ARIMA modeling.**

**2. Introduction to ARIMA Models** [¶](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#2.-Introduction-to-ARIMA-Models-)

[Table of Contents](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#0.1)

* **ARIMA** stands for **Autoregressive Integrated Moving Average Model**. It belongs to a class of models that explains a given time series based on its own past values -i.e.- its own lags and the lagged forecast errors. The equation can be used to forecast future values. Any ‘non-seasonal’ time series that exhibits patterns and is not a random white noise can be modeled with ARIMA models.
* So, **ARIMA**, short for **AutoRegressive Integrated Moving Average**, is a forecasting algorithm based on the idea that the information in the past values of the time series can alone be used to predict the future values.
* **ARIMA Models** are specified by three order parameters: (p, d, q),

where,

* + p is the order of the AR term
  + q is the order of the MA term
  + d is the number of differencing required to make the time series stationary
* **AR(p) Autoregression** – a regression model that utilizes the dependent relationship between a current observation and observations over a previous period. An auto regressive (AR(p)) component refers to the use of past values in the regression equation for the time series.
* **I(d) Integration** – uses differencing of observations (subtracting an observation from observation at the previous time step) in order to make the time series stationary. Differencing involves the subtraction of the current values of a series with its previous values d number of times.
* **MA(q) Moving Average** – a model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations. A moving average component depicts the error of the model as a combination of previous error terms. The order q represents the number of terms to be included in the model.

## **Types of ARIMA Model**[**¶**](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#Types-of-ARIMA-Model)

* **ARIMA** : Non-seasonal Autoregressive Integrated Moving Averages
* **SARIMA** : Seasonal ARIMA
* **SARIMAX** : Seasonal ARIMA with exogenous variables

If a time series, has seasonal patterns, then we need to add seasonal terms and it becomes SARIMA, short for **Seasonal ARIMA**.

# ****3. The meaning of p, d and q in ARIMA model**** [¶](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#3.-The-meaning-of-p,-d-and-q-in-ARIMA-model-)

[Table of Contents](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#0.1)

## **3.1 The meaning of p**

* p is the order of the **Auto Regressive (AR)** term. It refers to the number of lags of Y to be used as predictors.

## **3.2 The meaning of d**

* The term **Auto Regressive**’ in ARIMA means it is a linear regression model that uses its own lags as predictors. Linear regression models, as we know, work best when the predictors are not correlated and are independent of each other. So we need to make the time series stationary.
* The most common approach to make the series stationary is to difference it. That is, subtract the previous value from the current value. Sometimes, depending on the complexity of the series, more than one differencing may be needed.
* The value of d, therefore, is the minimum number of differencing needed to make the series stationary. If the time series is already stationary, then d = 0.

## **3.3 The meaning of q**

* **q** is the order of the **Moving Average (MA)** term. It refers to the number of lagged forecast errors that should go into the ARIMA Model.

# ****4. AR and MA models**** [¶](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#4.-AR-and-MA-models-)

[Table of Contents](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#0.1)

## **4.1 AR model**

* An **Auto Regressive (AR) model** is one where Yt depends only on its own lags.
* That is, Yt is a function of the lags of Yt. It is depicted by the following equation -

image source : <https://www.machinelearningplus.com/wp-content/uploads/2019/02/Equation-1-min.png?ezimgfmt=ng:webp/ngcb1>

where,

* Yt−1Yt−1 is the lag1 of the series,
* β1β1 is the coefficient of lag1 that the model estimates, and
* αα is the intercept term, also estimated by the model.

## **4.2 MA model**[**¶**](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#4.2-MA-model)

* Likewise a **Moving Average (MA) model** is one where Yt depends only on the lagged forecast errors. It is depicted by the following equation -

image source : <https://www.machinelearningplus.com/wp-content/uploads/2019/02/Equation-2-min.png?ezimgfmt=ng:webp/ngcb1>

where the error terms are the errors of the autoregressive models of the respective lags.

The errors Et and E(t-1) are the errors from the following equations :

image source : <https://www.machinelearningplus.com/wp-content/uploads/2019/02/Equation-3-min.png?ezimgfmt=ng:webp/ngcb1>

Thus, we have discussed AR and MA Models respectively.

## **4.3 ARIMA model**[**¶**](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#4.3-ARIMA-model)

* An ARIMA model is one where the time series was differenced at least once to make it stationary and we combine the AR and the MA terms. So the equation of an ARIMA model becomes :

image source : <https://www.machinelearningplus.com/wp-content/uploads/2019/02/Equation-4-min-865x77.png?ezimgfmt=ng:webp/ngcb1>

### **ARIMA model in words:**

Predicted Yt = Constant + Linear combination Lags of Y (upto p lags) + Linear Combination of Lagged forecast errors (upto q lags)

**5. How to find the order of differencing (d) in ARIMA model** [¶](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#5.-How-to-find-the-order-of-differencing-(d)-in-ARIMA-model--)

[Table of Contents](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#0.1)

* As stated earlier, the purpose of differencing is to make the time series stationary. But we should be careful to not over-difference the series. An over differenced series may still be stationary, which in turn will affect the model parameters.
* So we should determine the right order of differencing. The right order of differencing is the minimum differencing required to get a near-stationary series which roams around a defined mean and the ACF plot reaches to zero fairly quick.
* If the autocorrelations are positive for many number of lags (10 or more), then the series needs further differencing. On the other hand, if the lag 1 autocorrelation itself is too negative, then the series is probably over-differenced.
* If we can’t really decide between two orders of differencing, then we go with the order that gives the least standard deviation in the differenced series.
* Now, we will explain these concepts with the help of an example as follows:-
* First, I will check if the series is stationary using the **Augmented Dickey Fuller test (ADF Test)**, from the statsmodels package. The reason being is that we need differencing only if the series is non-stationary. Else, no differencing is needed, that is, d=0.
* The null hypothesis (Ho) of the ADF test is that the time series is non-stationary. So, if the p-value of the test is less than the significance level (0.05) then we reject the null hypothesis and infer that the time series is indeed stationary.
* So, in our case, if P Value > 0.05 we go ahead with finding the order of differencing.

**6. How to find the order of the AR term (p)** [¶](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#6.-How-to-find-the-order-of-the-AR-term-(p)-)

[Table of Contents](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#0.1)

* The next step is to identify if the model needs any AR terms. We will find out the required number of AR terms by inspecting the **Partial Autocorrelation (PACF) plot**.
* **Partial autocorrelation** can be imagined as the correlation between the series and its lag, after excluding the contributions from the intermediate lags. So, PACF sort of conveys the pure correlation between a lag and the series. This way, we will know if that lag is needed in the AR term or not.
* Partial autocorrelation of lag (k) of a series is the coefficient of that lag in the autoregression equation of Y.

Yt=α0+α1Yt−1+α2Yt−2+α3Yt−3Yt=α0+α1Yt−1+α2Yt−2+α3Yt−3

* That is, suppose, if Y\_t is the current series and Y\_t-1 is the lag 1 of Y, then the partial autocorrelation of lag 3 (Y\_t-3) is the coefficient α3α3 of Y\_t-3 in the above equation.
* Now, we should find the number of AR terms. Any autocorrelation in a stationarized series can be rectified by adding enough AR terms. So, we initially take the order of AR term to be equal to as many lags that crosses the significance limit in the PACF plot.

**7. How to find the order of the MA term (q)** [¶](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#7.-How-to-find-the-order-of-the-MA-term-(q)-)

[Table of Contents](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#0.1)

* Just like how we looked at the PACF plot for the number of AR terms, we will look at the ACF plot for the number of MA terms. An MA term is technically, the error of the lagged forecast.
* The ACF tells how many MA terms are required to remove any autocorrelation in the stationarized series.
* Let’s see the autocorrelation plot of the differenced series.

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

plt.rcParams.update({'figure.figsize':(9,3), 'figure.dpi':120})

fig, axes = plt.subplots(1, 2, sharex=True)

axes[0].plot(df.value.diff()); axes[0].set\_title('1st Differencing')

axes[1].set(ylim=(0,1.2))

plot\_acf(df.value.diff().dropna(), ax=axes[1])

plt.show()

# ****8. How to handle if a time series is slightly under or over differenced**** [¶](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#8.-How-to-handle-if-a-time-series-is-slightly-under-or-over-differenced-)

# ****9. How to build the ARIMA Model**** [¶](https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#9.-How-to-build-the-ARIMA-Model-)

Now, we have determined the values of p, d and q. We have everything needed to fit the ARIMA model. We will use the ARIMA() implementation in the statsmodels package.

In [7]:

from statsmodels.tsa.arima\_model import ARIMA

*# 1,1,2 ARIMA Model*

model = ARIMA(df.value, order=(1,1,2))

model\_fit = model.fit(disp=0)

print(model\_fit.summary())

* The model summary provides lot of information. The table in the middle is the coefficients table where the values under ‘coef’ are the weights of the respective terms.
* The coefficient of the MA2 term is close to zero and the P-Value in ‘P>|z|’ column is highly insignificant. It should ideally be less than 0.05 for the respective X to be significant.
* So, we will rebuild the model without the MA2 term.

Read more at..

<https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting>