



# Business Club **Time Series**

---

Business Club Analytics Team

December 2017

---

## Introduction

1. Motivation - Typical Case Study
2. Important Terminology
  - a. Trends
  - b. Seasonality
  - c. Stationary Process
  - d. White Noise
  - e. Covariance and Autocovariance
  - f. Autocorrelation and ACF
  - g. PACF
  - h. General Linear Process ( $\Psi$  and  $\Pi$  weights)
3. Major Models used their ACF and PACF plots
  - a. AR Models
  - b. MA Models
  - c. ARMA Models
  - d. ARIMA Models
4. Implementation
  - a. Model Identification
  - b. Implementation on our Case Study

## Motivation

Time series models are useful while working with serially correlated data. Most business houses work on time series data to analyze sales number (for the next year), website traffic, competition position and much more.

Some other important application spheres include:

1. Economics: monthly data for unemployment, hospital admissions, etc.
2. Finance - daily exchange rate, a share price, etc.
3. Environmental - daily rainfall, air quality readings.
4. Medicine - e.g., ECG brain wave activity every  $2^{-8}$  secs

We will consider in detail the the AirPassengers dataset while discussing time series in detail. You can find the dataset here:

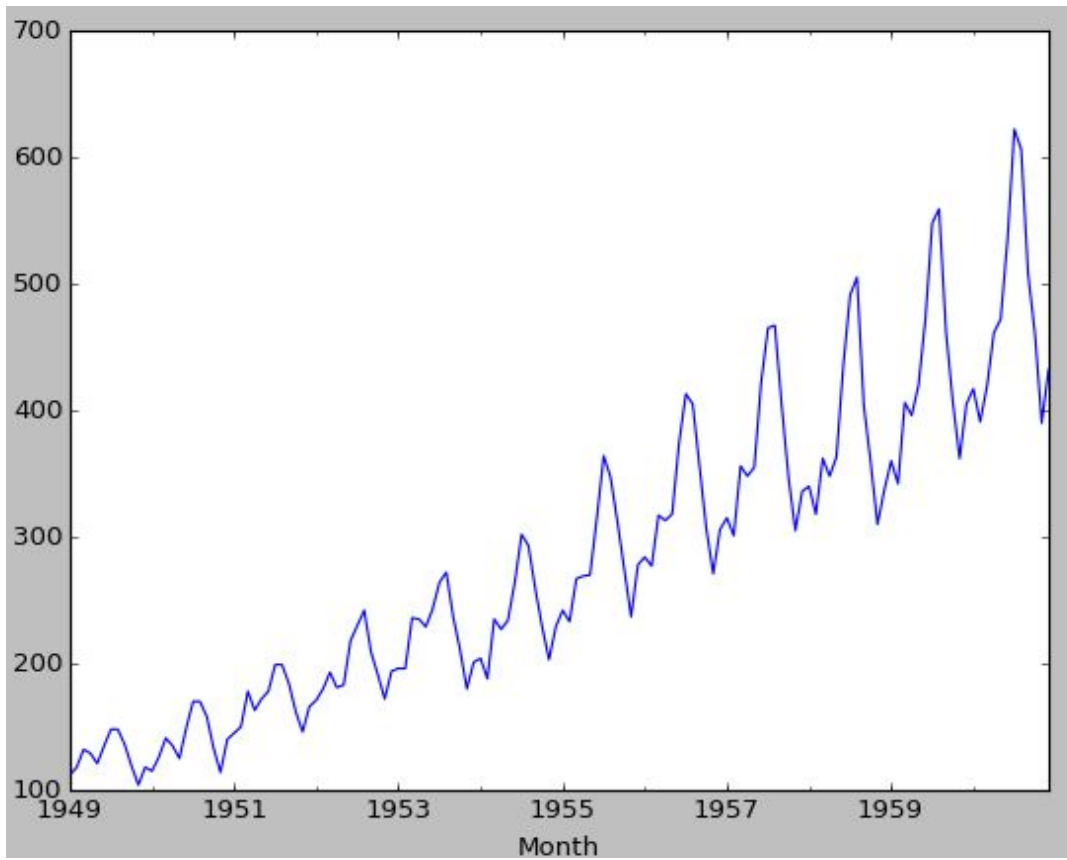
(<https://www.kaggle.com/rakannimer/air-passengers/data>)

It is a simple dataset with just two columns and we will be working on Python all the while. As evident from the figure below, the data gives information about the number of passengers travelling each month.

This is how the data head looks like:

```
Month
1949-01-01    112
1949-02-01    118
1949-03-01    132
1949-04-01    129
1949-05-01    121
Name: #Passengers, dtype: int64
```

The trends look like:



We see a definite rise in the overall passenger traffic as well as a seasonal nature

## Few Important Terms

- **Trends:** Long term movement of mean
- **Seasonality:** Cyclic fluctuations in calendar.

For example, there is seasonality in the monthly data, where high values always tend to occur in some particular months and low values always tend to occur in other particular months. In case,  $S = 12$  (months per year) is the span of the periodic seasonal behavior. For quarterly data,  $S = 4$  time periods per year.

- **Stochastic Process:**

A **stochastic** or **random process** is a mathematical object usually defined as a collection of random variables.

- **Stationary Process**

- **Strictly Stationary:** Sequence  $\{X_t, t\}$  is strictly stationary, if:

$$(X_{t_1}, \dots, X_{t_k}) \stackrel{D}{=} (X_{t_1+h}, \dots, X_{t_k+h})$$

for all sets of points  $t$  and integer  $h$

- **Weakly Stationary:** A process is weakly stationary or second order stationary if:

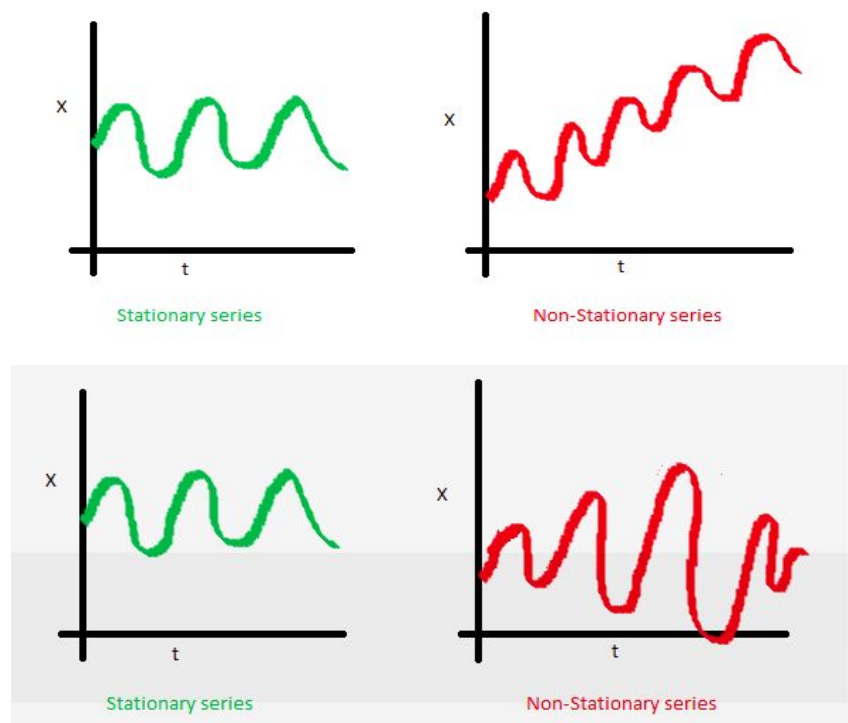
$$(a) \mathbb{E}(X_t) = \mu, \text{ and}$$

$$(b) \text{cov}(X_t, X_{t+k}) = \gamma_k,$$

where  $\mu$  is constant and  $\gamma_k$  is independent of  $t$ .

Where,  $E$  is the expectation function

Basically, the mean and variance should not be a function of time (property of constant variance is also called homoscedasticity), i.e. In the first figure, the mean of the series is a function of time while in the second series the variance of the series changes over time i.e. the width between peaks and troughs.



- **White Noise:** A sequence of independent random variables with zero mean and variance  $\sigma^2$  is called white noise. It is a weakly stationary series with  $\gamma_0 = \sigma^2$  and  $\gamma_k = 0$  ( $k \neq 0$ )
- **Co-Variance:** A statistical measure of variation or association of one variable X with the other, Y

$$\text{cov}(X, Y) = \mathbf{E} [(X - \mathbf{E}[X])(Y - \mathbf{E}[Y])]$$

Where  $\mathbf{E}[X]$  is the expectation of  $X$ , also known as mean of random variable  $X$   
 $\text{cov}(X, Y)$  is also denoted by  $\sigma(X, Y)$

The above equation can be simplified using linearity property of Expectation function

$$\begin{aligned}\text{cov}(X, Y) &= \mathbf{E}[(X - \mathbf{E}[X])(Y - \mathbf{E}[Y])] \\ &= \mathbf{E}[XY - X\mathbf{E}[Y] - \mathbf{E}[X]Y + \mathbf{E}[X]\mathbf{E}[Y]] \\ &= \mathbf{E}[XY] - \mathbf{E}[X]\mathbf{E}[Y] - \mathbf{E}[X]\mathbf{E}[Y] + \mathbf{E}[X]\mathbf{E}[Y] \\ &= \mathbf{E}[XY] - \mathbf{E}[X]\mathbf{E}[Y].\end{aligned}$$

Also,

$$\text{cov}(X, X) = \text{var}(X) \equiv \sigma^2(X) \equiv \sigma_X^2.$$

- **Autocovariance:** Autocovariance is the covariance between a stochastic process at different times.

$$\gamma(k) = \text{cov}(X_{n+k}, X_n)$$

- **Autocorrelation:** The autocorrelation function is defined as ratio of  $\gamma(k)$  and  $\gamma(0)$

$$\rho(k) = \frac{\gamma(k)}{\gamma(0)}$$

- Autocorrelation and Autocovariance Function:

The autocovariance function of a weakly stationary process is a capture of variation of autocovariance with  $k$ . It can be expressed as  $f(k) = \gamma(k)$ . For example, the autocovariance function of a stochastic process

$$x_t = u_t + \theta u_{t-1}$$

Where  $u_t$  is the white noise ( $WN(0, \sigma^2)$ ).

The autocovariance function for the above process is given by:

$$\begin{aligned} \gamma_x(k) &= \text{Cov}(x_t, x_{t-k}) \\ &= \begin{cases} (1 + \theta^2) \sigma_u^2 & \text{for } k = 0 \\ \theta \sigma_u^2 & \text{for } k = 1 \\ 0 & \text{for } k = 2, 3, \dots \end{cases} \end{aligned}$$

- Note the autocovariance cuts off after lag 1
- Autocorrelation function can be similarly modelled,  $\rho(k)$  is used instead of  $\gamma(k)$

- Autocovariance Matrix

$$\Gamma_n = \begin{bmatrix} \gamma_0 & \gamma_1 & \gamma_2 & \dots & \gamma_{n-1} \\ \gamma_1 & \gamma_0 & \gamma_1 & \dots & \gamma_{n-2} \\ \gamma_2 & \gamma_1 & \gamma_0 & \dots & \gamma_{n-3} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \gamma_{n-1} & \gamma_{n-2} & \gamma_{n-3} & \dots & \gamma_0 \end{bmatrix}$$

- Autocorrelation Matrix

$$\begin{bmatrix} 1 & \rho_1 & \rho_2 & \dots & \rho_{n-1} \\ \rho_1 & 1 & \rho_1 & \dots & \rho_{n-2} \\ \rho_2 & \rho_1 & 1 & \dots & \rho_{n-3} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \rho_{n-1} & \rho_{n-2} & \rho_{n-3} & \dots & 1 \end{bmatrix}$$



---

- **Partial Autocorrelation Function (PACF):**

**Partial correlation** is different from variance in terms that it is a measure of association between two random variables, with the effect of one set of contributing variables removed.

For example, we have data on the consumption, income, and wealth of various individuals and we wish to see if there is a relationship between consumption and income, failing to control for wealth. When we compute a correlation coefficient between consumption and income, the result would be misleading. Since income might be numerically related to wealth, which in turn might be numerically related to consumption; a measured correlation between consumption and income might actually be contaminated by these other correlations. The use of a partial correlation helps avoid this problem.

- Partial Autocorrelation is the partial correlation of a time series with its own lagged values, controlling for the values of the time series at all shorter lags.

The Partial Autocorrelation function(PACF) plays an important role in identifying the lag of an autoregressive process. The use of this function was introduced as part of the [Box–Jenkins](#) approach to time series modelling.

Going into the mathematical insights of PACF is out of the scope for this text. Only the plots of PACF are useful in understanding the order of AR(p) and ARMA(p,q) process.

- General Linear Process:

### Ψ weights

It is a representation of a stochastic process as the output from a linear filter, whose input is white noise  $a_t$ :

$$\begin{aligned}\tilde{z}_t &= a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \dots \\ &= a_t + \sum_{j=1}^{\infty} \psi_j a_{t-j}\end{aligned}$$

where,

$$\tilde{z}_t = z_t - \mu$$

It allows us to represent the process as a weighted sum of present and past values of the white noise process  $a_t$ . The following properties of the white noise should be noted:

$$E[a_t] = 0 \quad \text{var}[a_t] = \sigma_a^2$$

### Π weights

An alternative way of modelling a linear process would be by representing the current deviation as a weighted sum of the previous deviations  $z_{t-1}, z_{t-2}, z_{t-3}, \dots$

$$\begin{aligned}\tilde{z}_t &= \pi_1 \tilde{z}_{t-1} + \pi_2 \tilde{z}_{t-2} + \dots + a_t \\ &= \sum_{j=1}^{\infty} \pi_j \tilde{z}_{t-j} + a_t\end{aligned}$$

Relation between Ψ and Π weights is given by

$$\pi(B) = \psi^{-1}(B)$$

Where B is the backshift operator

$$Bz_t = z_{t-1} \quad \text{and hence} \quad B^j z_t = z_{t-j}$$

## AR Models- AR(p) Process

An autoregressive process of order p is represented as

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \dots + \phi_p \tilde{z}_{t-p} + a_t$$

or

$$(1 - \phi_1 B - \dots - \phi_p B^p) \tilde{z}_t = \phi(B) \tilde{z}_t = a_t$$

Where B is the backshift operator and  $\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \dots$  are adjustable parameters.


Mathematically, we compare an AR(p) process with a general linear process to check stationarity, by imposing the condition:

$$\sum_{j=0}^{\infty} \psi_j < \infty \quad \text{(summation zero to infinity)}$$

For an AR(1) process the maths works like:

$$(1 - \phi_1 B) \tilde{z}_t = a_t$$

Which can written as:


$$\tilde{z}_t = (1 - \phi_1 B)^{-1} a_t = \sum_{j=0}^{\infty} \phi_1^j a_{t-j}$$

On comparing with general linear process,

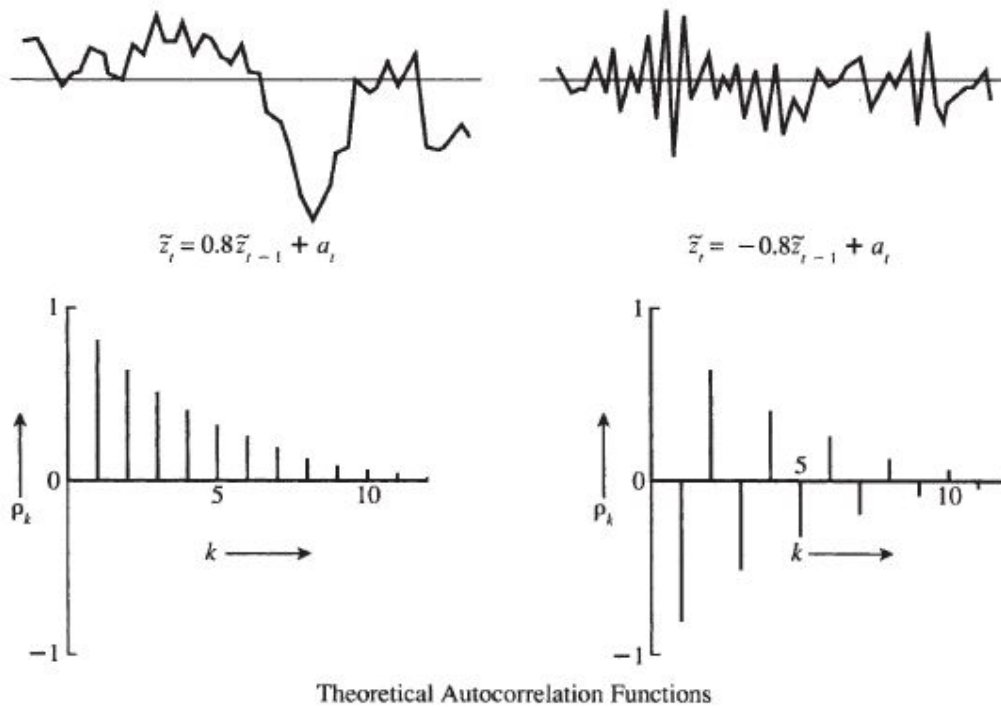
$$\psi(B) = (1 - \phi_1 B)^{-1} = \sum_{j=0}^{\infty} \phi_1^j B^j$$

For  $\psi(B)$  to converge , we have

$$\sum_{j=0}^{\infty} |\phi_1|^j < \infty$$

- The autocorrelation function of an AR(p) Process dies down eventually
- The PACF function for an AR(p) Process cuts off

## Autocorrelation Plot:



## Partial Autocorrelation Plot:

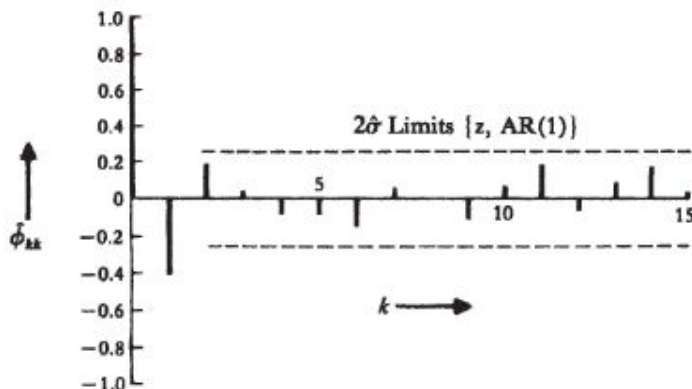
The partial autocorrelation function is represented by  $\phi_{kk}$   $k=1,2,3,\dots$

For an AR process solving for  $k=1,2,3,\dots$  We obtain

$$\phi_{11} = \rho_1$$
$$\phi_{22} = \frac{\begin{vmatrix} 1 & \rho_1 \\ \rho_1 & \rho_2 \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{vmatrix}} = \frac{\rho_2 - \rho_1^2}{1 - \rho_1^2}$$

$$\phi_{33} = \frac{\begin{vmatrix} 1 & \rho_1 & \rho_1 \\ \rho_1 & 1 & \rho_2 \\ \rho_2 & \rho_1 & \rho_3 \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{vmatrix}}$$

- For an AR(p) process  $\phi_{kk}=0$  for  $k > p$  and non zero for  $k \leq p$
- Basically the PACF cuts off after lag p



The above figure is a plot of an estimated PACF with two standard error limits assuming the model is AR(1). Since  $E[\phi_{22}]$  is also significant, there is a possibility of the process being AR(2). We can make further investigations to clear our doubts.

## MA(q) Models : Moving Average Models

An MA(q) process looks like:

$$\begin{aligned} \tilde{z}_t &= a_t - \theta_1 a_{t-1} - \dots - \theta_q a_{t-q} \\ &= (1 - \theta_1 B - \dots - \theta_q B^q) a_t \\ &= \theta(B) a_t \end{aligned}$$

Since  $\theta_B$  is a finite set, on comparing an MA process to a general linear process we always have the summation of  $\psi(B)$  converging. Hence an MA process is always stationary. We have to work to establish invertibility of an MA process.

- The autocorrelation function of an MA(q) process cuts off after q

$$\rho_k = \begin{cases} \frac{-\theta_k + \theta_1\theta_{k+1} + \dots + \theta_{q-k}\theta_q}{1 + \theta_1^2 + \dots + \theta_q^2} & k = 1, 2, \dots, q \\ 0 & k > q \end{cases}$$

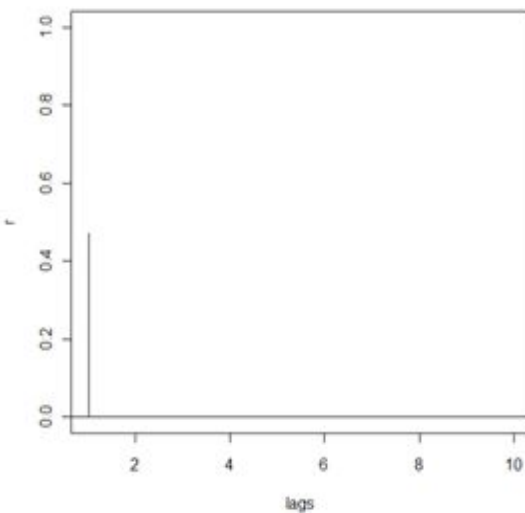
- While the PACF dies down gradually

### Autocorrelation Function:

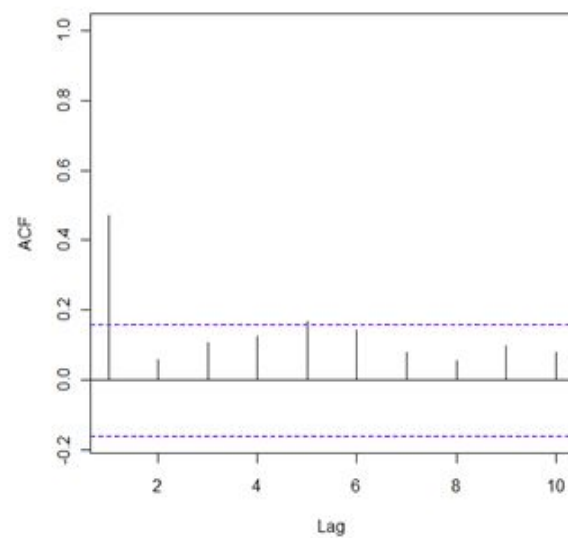
Consider an MA(1) process with  $\theta_1=0.7$

$$x_t = 10 + w_t + .7w_{t-1}$$

The theoretical and practical ACF plot might look dissimilar but the trend follows an overall similar pattern as PACF for an AR process.

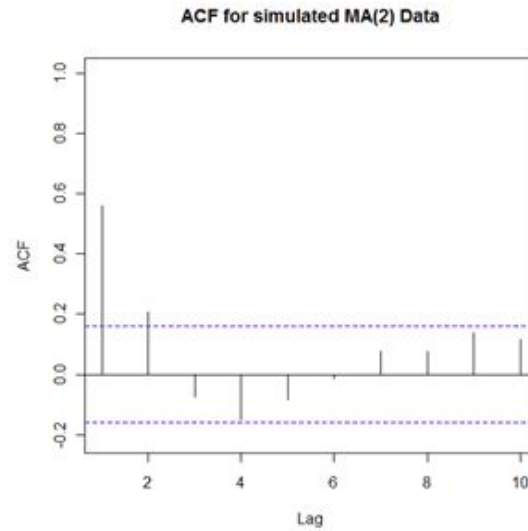
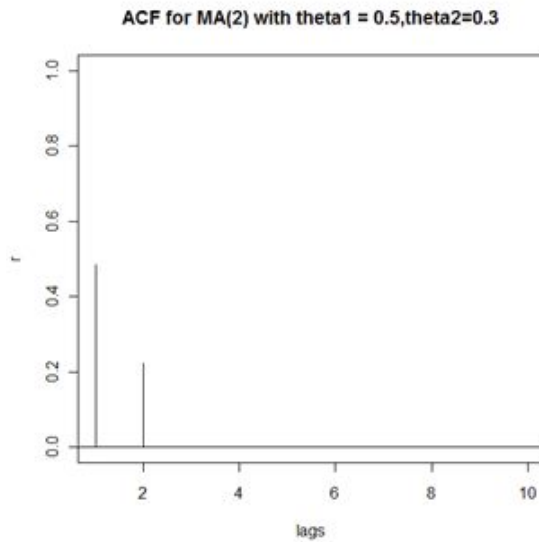


Theoretical



Practical

For an MA(2) process



### Partial Autocorrelation Function:

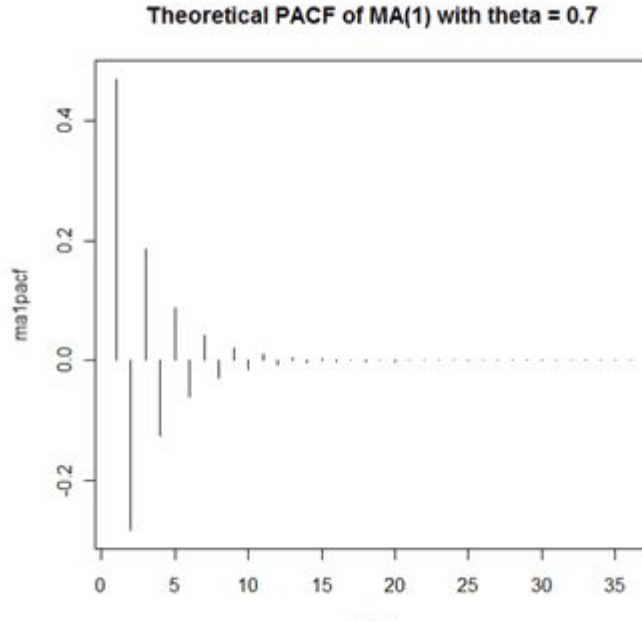
- PACF for an MA(q) process gradually tapers towards zero in some manner

Consider an MA(1) process

$$x_t = 10 + w_t + 0.7w_{t-1}$$

The PACF plot looks like





## ARMA(p,q) Process: AutoRegressive Moving Average

It is just the linear combination of the above two processes studied.

General Equation:

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \dots + \phi_p \tilde{z}_{t-p} + a_t - \theta_1 a_{t-1} - \dots - \theta_q a_{t-q}$$

Or

$$(1 - \phi_1 B - \phi_2 B^2 - \dots - \phi_p B^p) \tilde{z}_t = (1 - \theta_1 B - \theta_2 B^2 - \dots - \theta_q B^q) a_t$$

Or

$$\phi(B) \tilde{z}_t = \theta(B) a_t$$

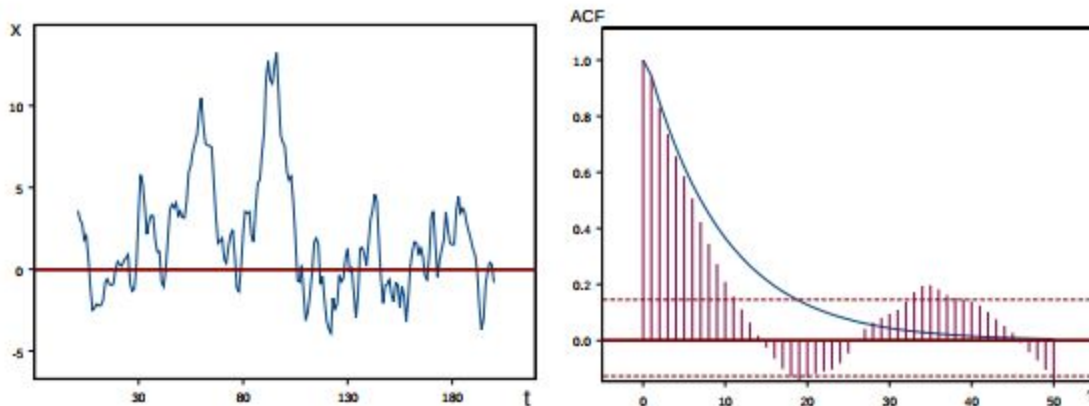


Figure 6.1: ARMA(1,1) simulated process  $x_t - 0.9x_{t-1} = z_t + 0.5z_{t-1}$ , sample ACF and the theoretical ACF of this process.

## ARIMA(p,d,q) Models- Autoregressive Integrated Moving Average Models

It is just a generalization of the ARMA models. ARIMA models are applied in some cases where data shows evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.

### Non- seasonal ARIMA Models:

If we combine differencing with autoregression and a moving average model, we obtain a non-seasonal ARIMA model. ARIMA is an acronym for AutoRegressive Integrated Moving Average model ("integration" in this context is the reverse of differencing). The full model can be written as

$$y'_t = c + \phi_1 y'_{t-1} + \dots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \dots + \theta_q e_{t-q} + e_t,$$

where  $y'_t$  is the differenced series (it may have been differenced more than once). The "predictors" on the right hand side include both lagged values of  $y'_t$  and lagged errors. We call this an **ARIMA(p,d,q) model**, where

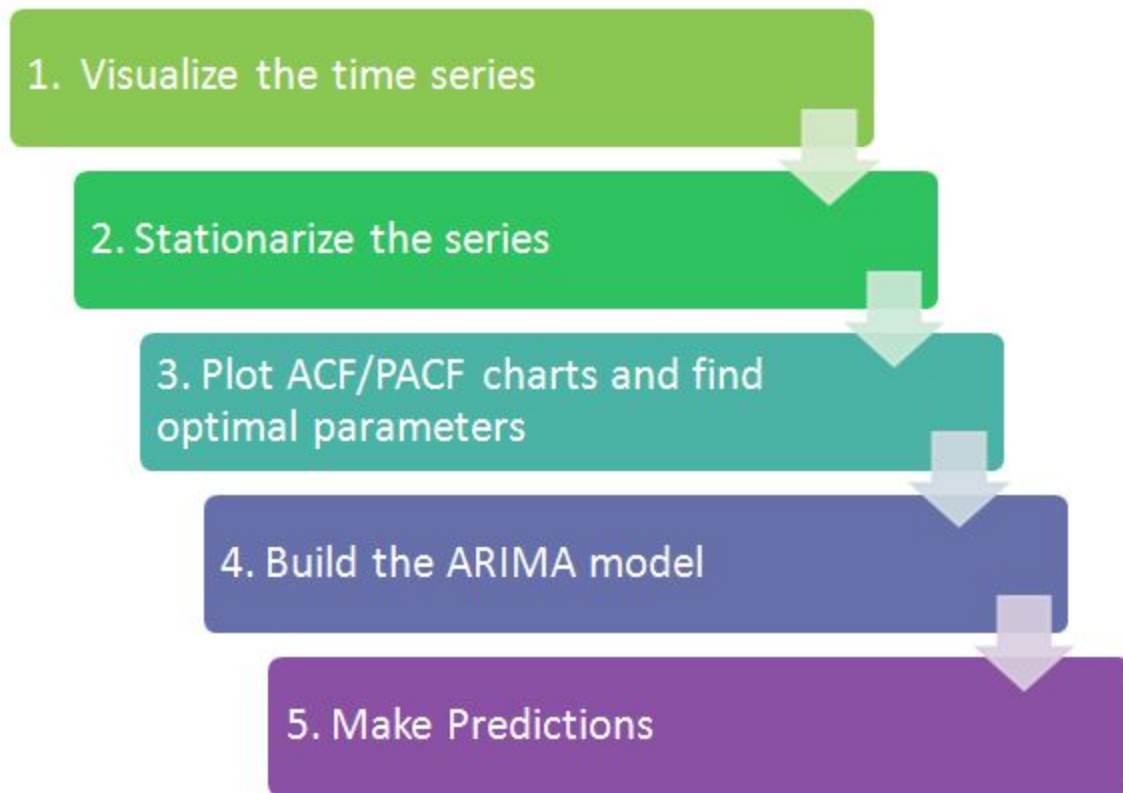
$$\begin{array}{ccccc}
 (1 - \phi_1 B - \dots - \phi_p B^p) & (1 - B)^d y_t & = & c + (1 + \theta_1 B + \dots + \theta_q B^q) e_t \\
 \uparrow & \uparrow & & \uparrow \\
 \text{AR}(p) & d \text{ differences} & & \text{MA}(q)
 \end{array}$$

## Implementation and Model Identification:

Remember: The trends of ACF and PACF would play a key role in choosing a model

Model	ACF	PACF
AR(p)	Tails off gradually	Cuts off after lag p
MA(q)	Cuts off after lag q	Tails off gradually
ARMA(p,q)	Tails off gradually	Tails off gradually

Implementation on our Case Study:



### 1) Visualizing and deriving Inferences

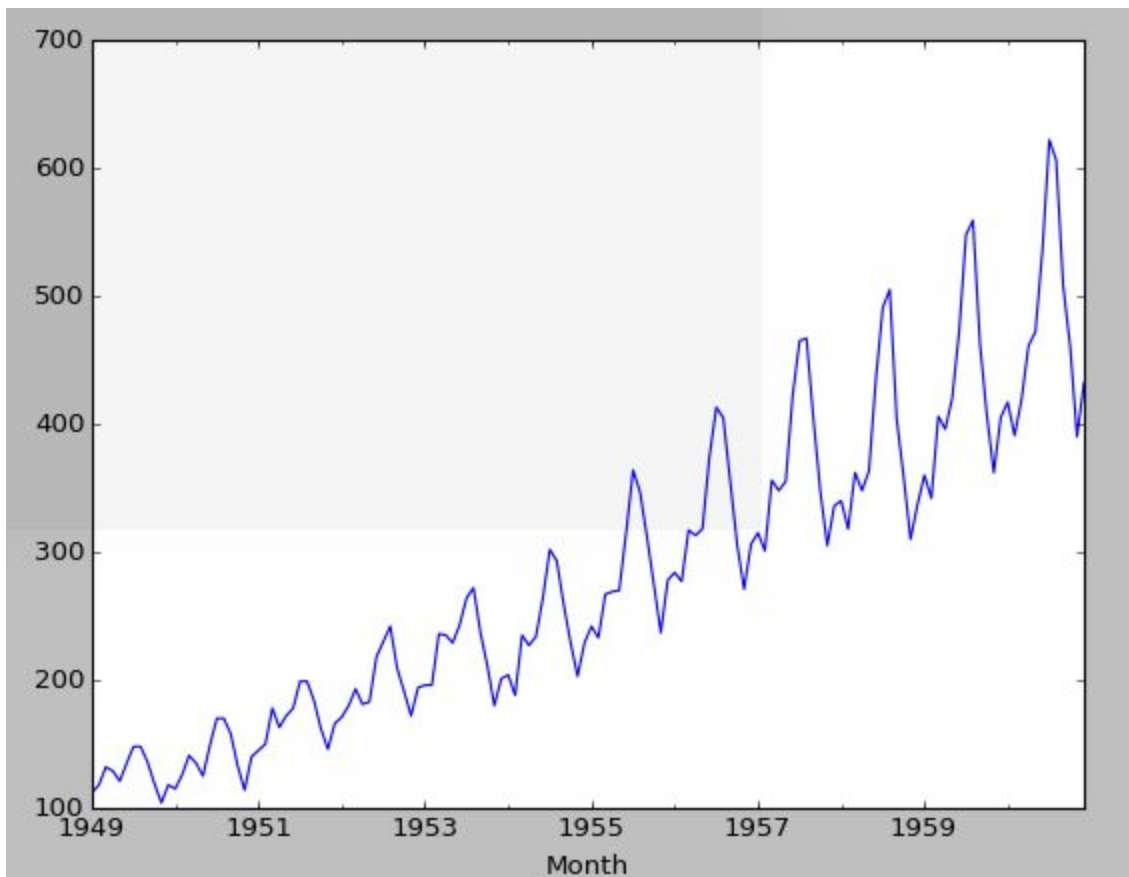
The above is a data visualization from the AirPassengers dataset, capturing their travel frequencies against time. Such a plot clearly demonstrates seasonality and non-stationarity (since the variance or the width of the peaks keep on increasing with time).

```
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('AirPassengers.csv', header=0)
print(series.head())
```

```
Month
1949-01-01    112
1949-02-01    118
1949-03-01    132
1949-04-01    129
1949-05-01    121
Name: #Passengers, dtype: int64
```

```
import numpy as np
import pandas as pd
```

```
series.plot()
pyplot.show()
```



The data seems to have a trend and seasonality. We will try to break it down into four components. To make our lives easier, Python has a library called statsmodels. It would break the data down into an additive model or a multiplicative model.

### Additive Model:

It works on the principle where the models looks like:

$$y(t)=\text{Level}+\text{Trend}+\text{Seasonality}+\text{Noise}$$

- An additive model is linear where changes over time are consistently made by the same amount
- A linear trend is a straight line
- A linear seasonality has the same frequency (width of cycles) and amplitude (height of cycles)

### Multiplicative Model:

$$y(t)=\text{Level}*\text{Trend}*\text{Seasonality}*\text{Noise}$$

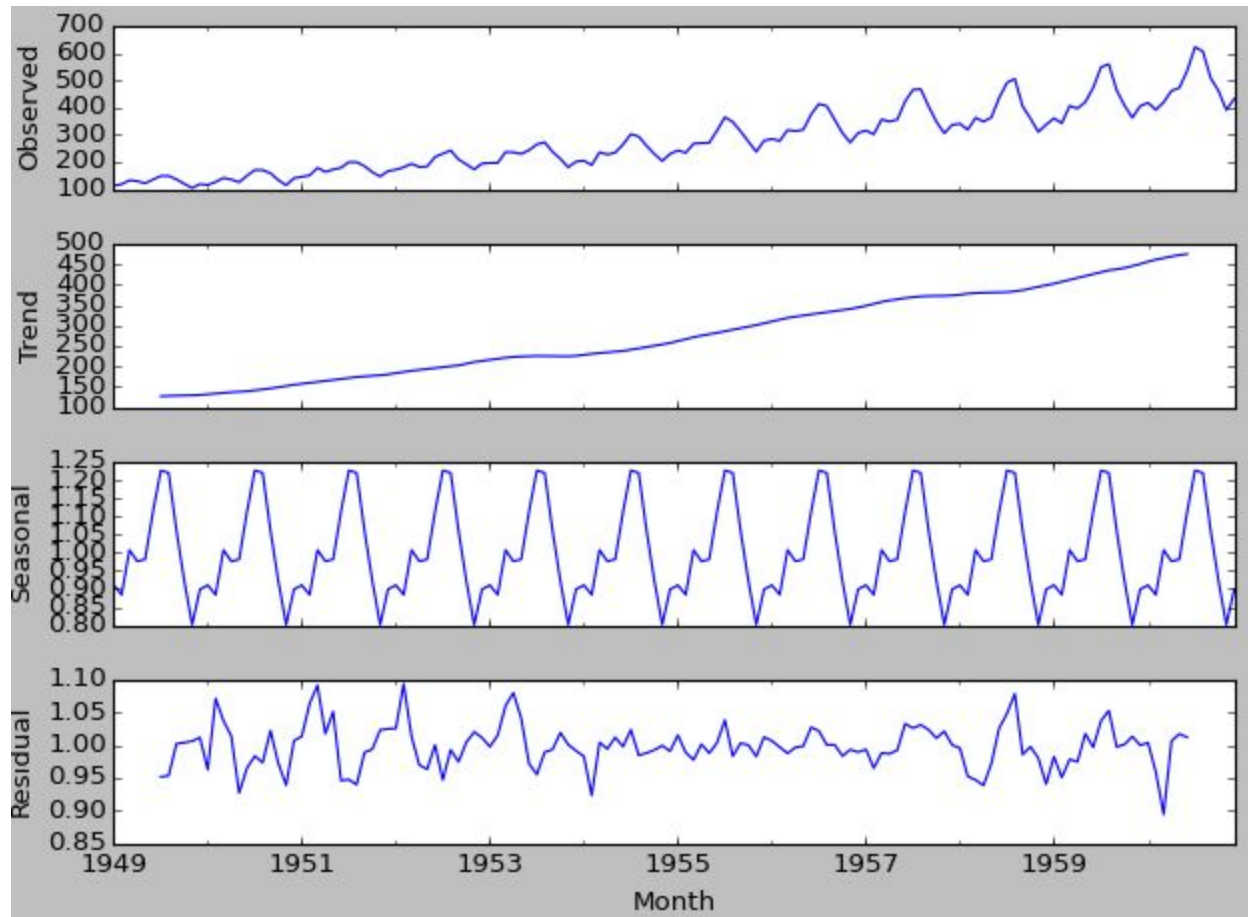
- A multiplicative model is nonlinear, such as quadratic or exponential. Changes increase or decrease over time.
- A nonlinear trend is a curved line.
- A non-linear seasonality has an increasing or decreasing frequency and/or amplitude over time.

From the above plot we can see that it is clearly a Multiplicative model, as it is non linear, changing nearly as a power of 2.

Real-world problems are messy and noisy. There may be additive and multiplicative components. There may be an increasing trend followed by a decreasing trend. There may be non-repeating cycles mixed in with the repeating seasonality components.

Nevertheless, these abstract models provide a simple framework that you can use to analyze your data and explore ways to think about and forecast your problem.

```
result = seasonal_decompose(series, model='multiplicative')
result.plot()
pyplot.show()
```



1. A clear inference is the seasonality is at least 12 months.
2. The year on year trend show the number of passengers are increasing without fail.
3. July and August have a significantly higher seasonal passenger traffic compared to other months.

## 2. Stationarize the series

When modeling, there are assumptions that the time series we are dealing with is stationary. In reality, this assumption can be easily violated in time series by the addition of a trend, seasonality, and other time-dependent structures.

Once we know the patterns, trends, cycles and seasonality, we can check if the series is stationary or not. Dickey – Fuller is a popular test to check the same. You can read up on the maths behind Dickey Fuller test; for now we will cover implementation and inferences

### Augmented Dickey Fuller Test:

Statistical tests make strong assumptions about your data. They can only be used to inform the degree to which a null hypothesis can be accepted or rejected. The result must be interpreted for a given problem to be meaningful.

Nevertheless, they can provide a quick check and confirmatory evidence that your time series is stationary or non-stationary.

The Augmented Dickey-Fuller test is a type of statistical test called a unit root test.

The intuition behind a unit root test is that it determines how strongly a time series is defined by a trend.

There are a number of unit root tests and the Augmented Dickey-Fuller is one of the more widely used. It uses an autoregressive model and optimizes an information criterion across multiple different lag values.

The null hypothesis of the test is that the time series can be represented by a unit root, that it is not stationary (has some time-dependent structure). The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary.

- **Null Hypothesis (H0):** If accepted, it suggests that the time series has a unit root, meaning it is non-stationary. It has some time dependent structure.



- **Alternate Hypothesis (H1):** The null hypothesis is rejected; it suggests the time series does not have a unit root, meaning it is stationary. It does not have a time-dependent structure.

We interpret this result using the p-value from the test. A p-value below a threshold (such as 5% or 1%) suggests we reject the null hypothesis (stationary), otherwise a p-value above the threshold suggests we accept the null hypothesis (non-stationary).

- **p-value > 0.05:** Accept the null hypothesis (H0), the data has a unit root and is non-stationary.
- **p-value <= 0.05:** Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

Below is an example of calculating the Augmented Dickey-Fuller test, its implementation on our dataset:

```
In [6]: from statsmodels.tsa.stattools import adfuller
X = series.values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: 0.815369
p-value: 0.991880
Critical Values:
    5%: -2.884
    1%: -3.482
   10%: -2.579
```

Since the **p-value > 0.05** we accept the null hypothesis, hence the series is non-stationary

This doesn't ends here! Now we know that the series is non-stationary, we have to use some techniques to make it stationary

There are three commonly used techniques to make a time series stationary:

1. **Detrending** : Here, we simply remove the trend component from the time series. For instance, the equation of my time series is:

---

$$x(t) = (\text{mean} + \text{trend} * t) + \text{error}$$

We'll simply remove the part in the parentheses and build a model for the rest.

2. **Differencing** : This is the commonly used technique to remove non-stationarity. Here we try to model the differences of the terms and not the actual term. For instance,

$$x(t) - x(t-1) = \text{ARMA}(p, q)$$

This differencing is called as the Integration part in AR(I)MA. Now, we have three parameters

**p : AR**

**d : I**

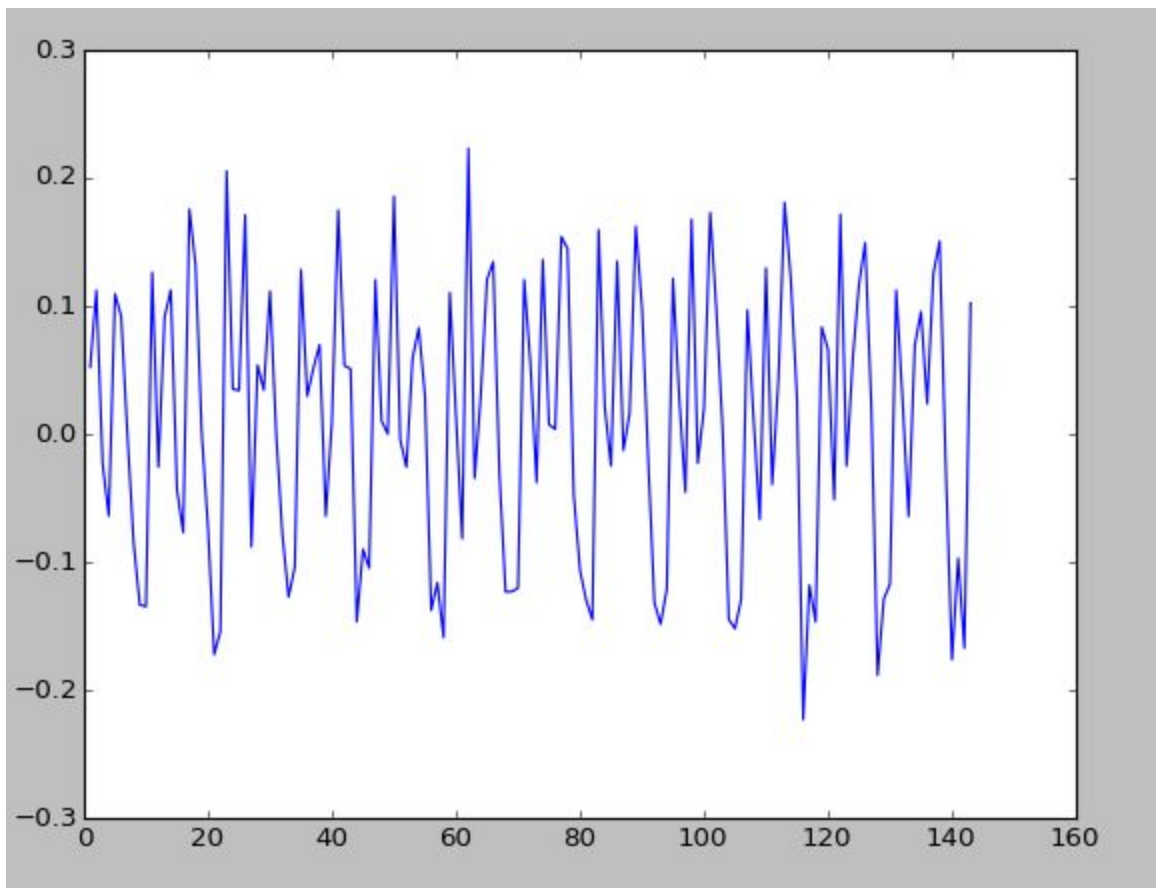
**q : MA**

3. **Seasonality** : Seasonality can easily be incorporated in the ARIMA model directly. More on this has been discussed in the applications part below.

Before implementing any of above processes on our data we should address the issue of unequal variances(**dealing with trends**). We deal with this using log operation on the series. Trends can be dealt by other mathematical operations like sq. root, cube root, log etc.

Then we resort to differencing to deal with seasonality.

```
ts_log_diff = ts_log - ts_log.shift()
plt.plot(ts_log_diff)
plt.show()
```



```

: ts_log_diff.dropna(inplace=True)
X=ts_log_diff
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

```

```

ADF Statistic: -2.717131
p-value: 0.071121
Critical Values:
    5%: -2.884
    1%: -3.483
   10%: -2.579

```

As after our operations, we have brought down the p-value to a lesser level, though not 0.05, we can reject our null hypothesis with certain confidence interval(would need to consult the p-value table for exact confidence interval), hence the series is now more stationary.

---

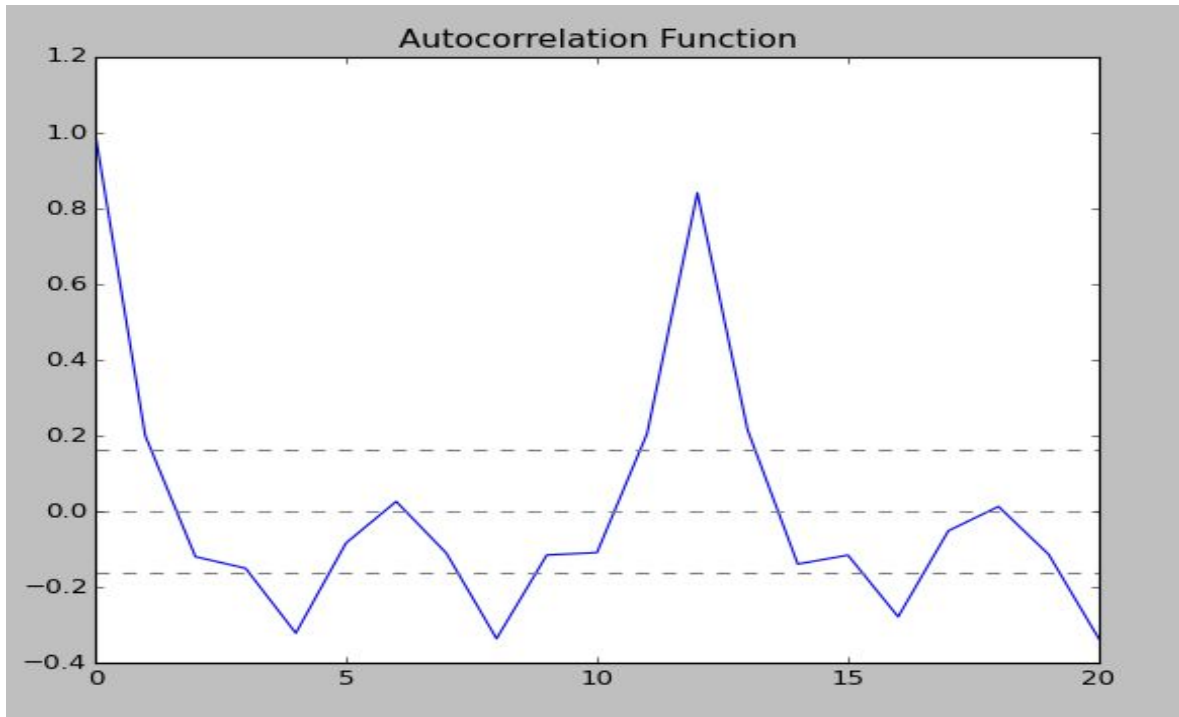
## Finding Optimal Parameters:

The parameters  $p, d, q$  can be found using ACF and PACF plots. In addition to this approach if both ACF and PACF decrease gradually, it indicates that we need to make the time series stationary and we introduce a value to “ $d$ ”.

```
from statsmodels.tsa.stattools import acf, pacf
ts_log_diff.dropna(inplace=True)
lag_acf = acf(ts_log_diff, nlags=20)
lag_pacf = pacf(ts_log_diff, nlags=20)
```

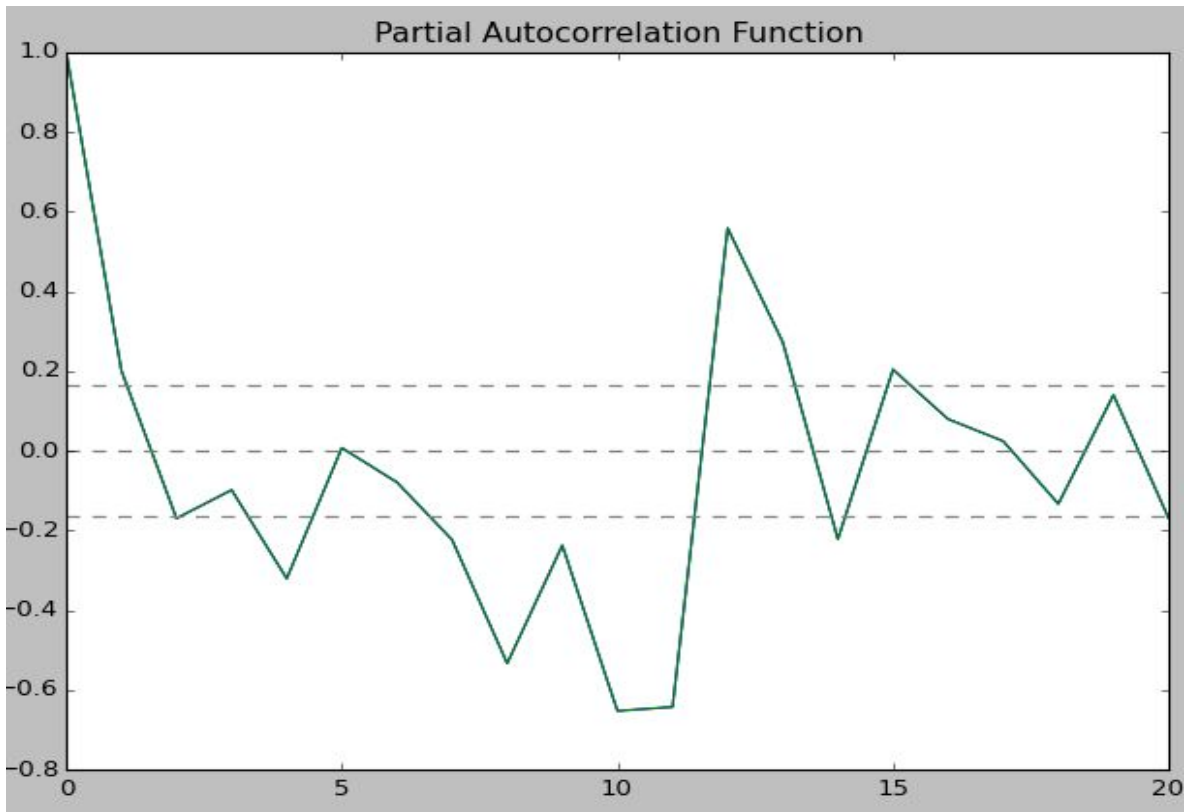
Plotting ACF

```
plt.plot(lag_acf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(ts_log_diff)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(ts_log_diff)),linestyle='--',color='gray')
plt.title('Autocorrelation Function')
plt.show()
```



Plotting PACF

```
plt.plot(lag_pacf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(ts_log_diff)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(ts_log_diff)),linestyle='--',color='gray')
plt.title('Partial Autocorrelation Function')
plt.show()
```



The ACF and PACF plots clearly don't follow the standard plots for AR and MA models. Hence the model best fitting here would most probably be an ARIMA model.

The next question is to determine the order of AR and MA involved.

1. **p** – The lag value where the **PACF** chart crosses the upper confidence interval for the first time. If you notice closely, in this case  $p=2$ .
2. **q** – The lag value where the **ACF** chart crosses the upper confidence interval for the first time. If you notice closely, in this case  $q=2$ .

## Building Model

We would consider the RSS (Residual Sum of Squares) of all the 3 possibilities, i.e.:

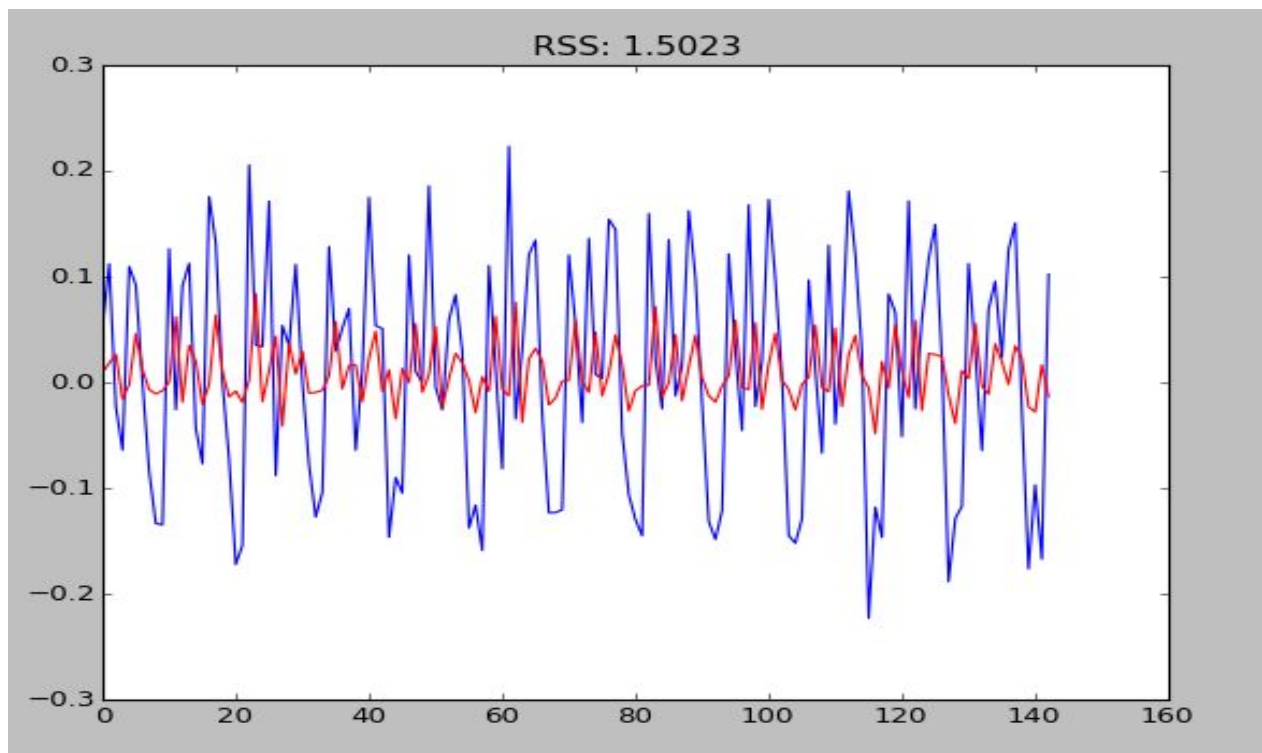
1. ARIMA(2,1,0)

2. ARIMA(0,1,2)
3. ARIMA(2,1,2)

### ARIMA(2,1,0)

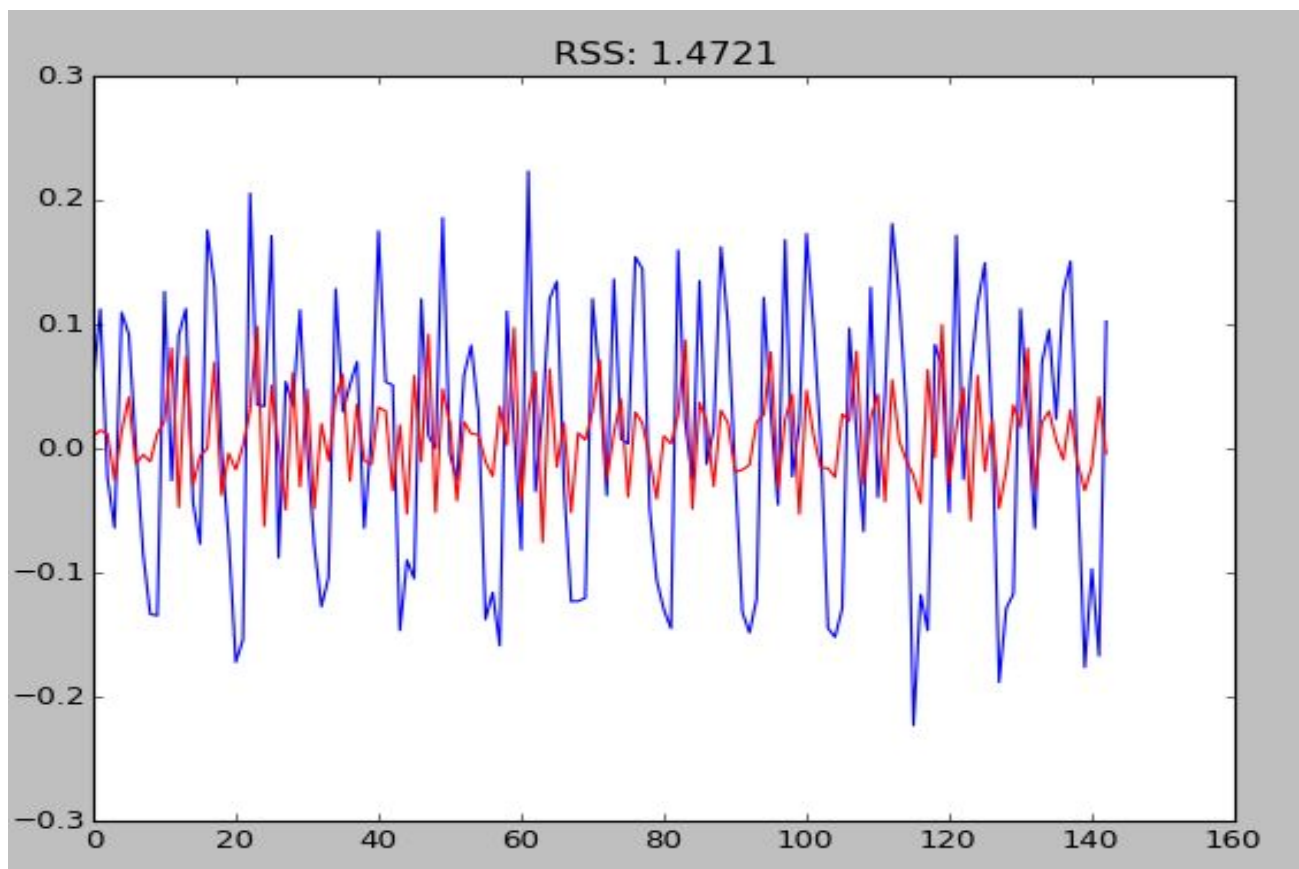
```
: from statsmodels.tsa.arima_model import ARIMA
model = ARIMA(ts_log, order=(2, 1, 0))
results_AR = model.fit(dispatch=-1)
plt.plot(ts_log_diff)
plt.plot(results_AR.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_AR.fittedvalues-ts_log_diff)**2))
plt.show()
```

\*disp controls the convergence information printed, if disp<0 no such info is printed. It is positive/true by default



### ARIMA(0,1,2):

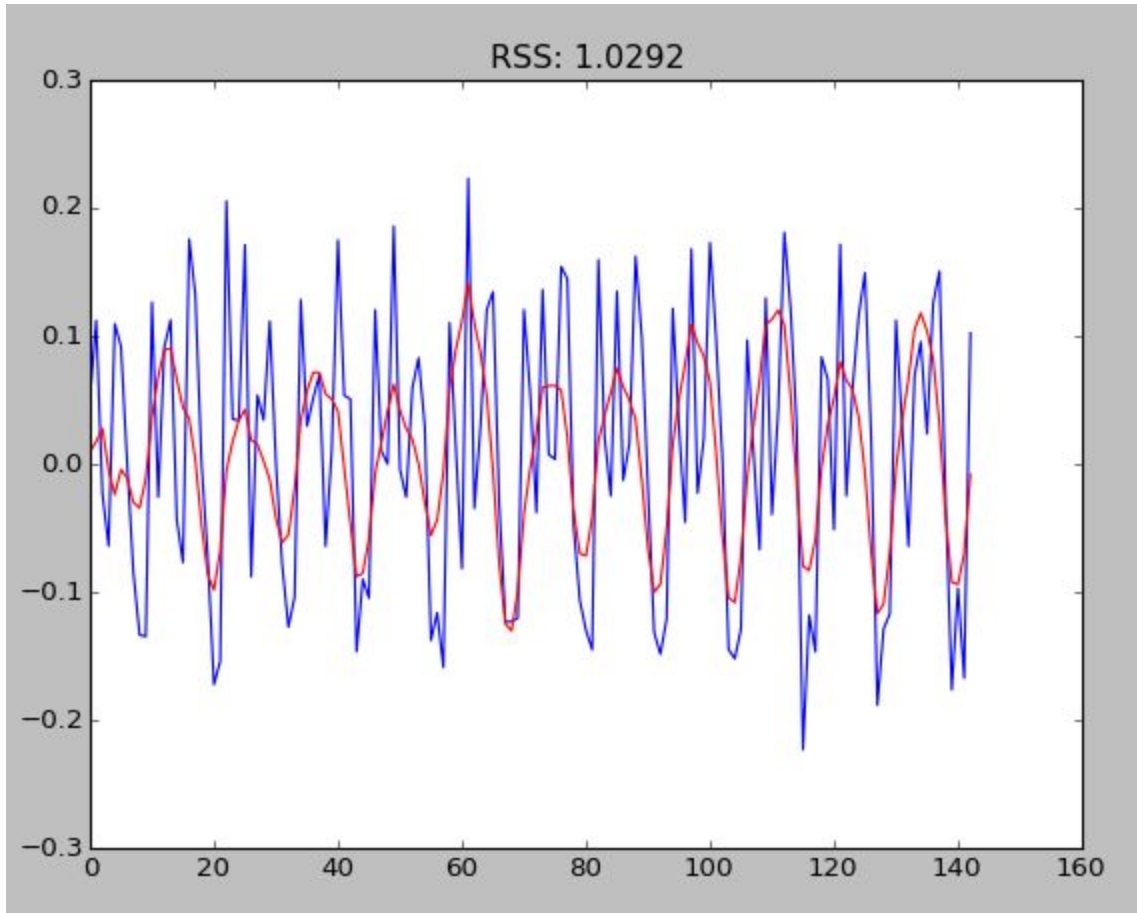
```
model = ARIMA(ts_log, order=(0, 1, 2))
results_MA = model.fit(disp=-1)
plt.plot(ts_log_diff)
plt.plot(results_MA.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_MA.fittedvalues-ts_log_diff)**2))
plt.show()
```



ARIMA(2,1,2):



```
model = ARIMA(ts_log, order=(2, 1, 2))
results_ARIMA = model.fit(dispatch=-1)
plt.plot(ts_log_diff)
plt.plot(results_ARIMA.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_ARIMA.fittedvalues-ts_log_diff)**2))
plt.show()
```



**Evidently among the three models, ARIMA(2,1,2) gives the minimum RSS and hence is most suitable**

## Making Predictions:

We need to make sure that we undo all the operations we did on the series to remove trends and seasonality. The RSS on our modified series and the original might vary, but we can't help that!

```
predictions_ARIMA_diff = pd.Series(results_ARIMA.fittedvalues, copy=True)
predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()
#adding cumulative summation, since the 1st entry was left in difference series

predictions_ARIMA_log = pd.Series(ts_log.iloc[0], index=ts_log.index)
predictions_ARIMA_log = predictions_ARIMA_log.add(predictions_ARIMA_diff_cumsum, fill_value=0)

predictions_ARIMA_log.head()
```

Month	
1949-01-01	4.718499
1949-02-01	4.728079
1949-03-01	4.745570
1949-04-01	4.773241
1949-05-01	4.768720

dtype: float64

**Finally we can check the RSS on our original data. Go ahead see how well does your data fit!**

```
predictions_ARIMA = np.exp(predictions_ARIMA_log)
plt.plot(series)
plt.plot(predictions_ARIMA)
plt.title('RMSE: %.4f' % np.sqrt(sum((predictions_ARIMA-series)**2)/len(series)))
plt.show()
```