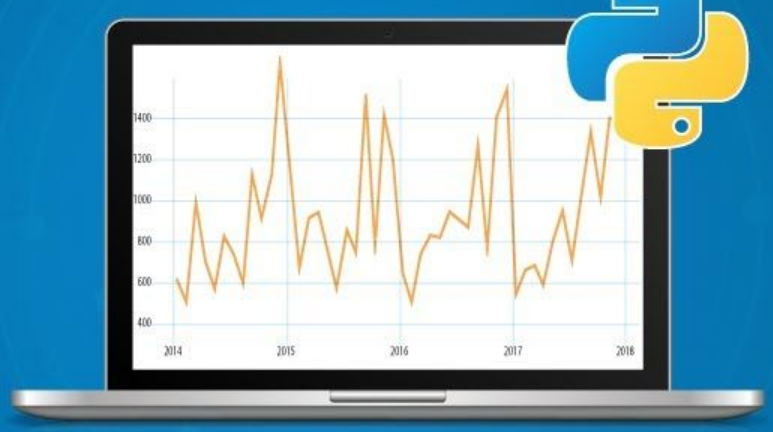
TIME SERIES ANALYSIS IN PYTHON



Topics Covered in Today's Training

01 Why Time Series Analysis?

02 What is Time Series?

03 Components of Time Series

04 When not to use Time Series?

05 What is Stationarity?

06 ARIMA model

07 Demo: Forecast future

Why Time Series Analysis?

In this analysis, you just have one variable – **TIME**

You can analyse this time series data in order to extract meaningful statistics and other characteristics

What Is Time Series?

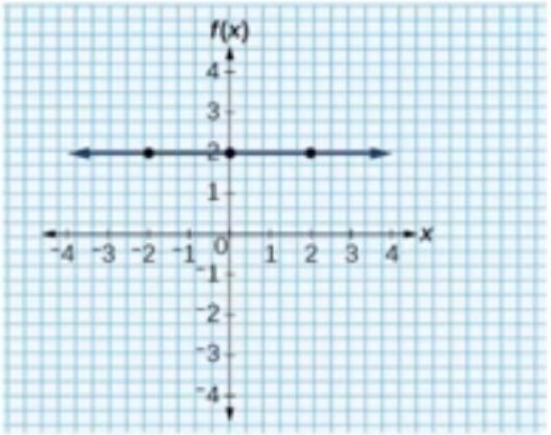
- A time series is a set of observation taken at specified times usually at equal intervals
- It is used to predict the future values based on the previous observed values

- Business Forecasting
- Understand past behavior
- Plan future
- Evaluate current accomplishment

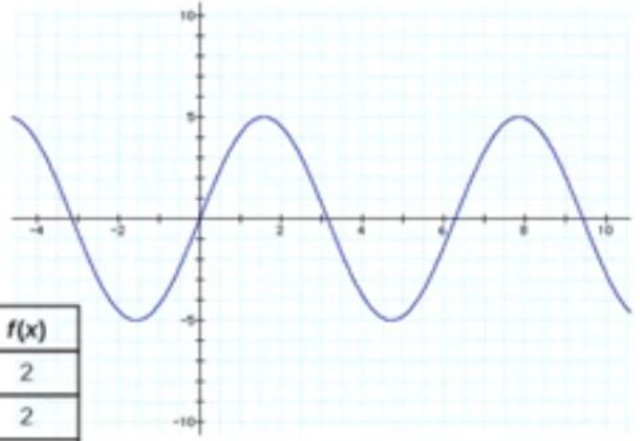
Components Of Time Series

When Not To Use Time Series Analysis?

1. Values are constant



2. Values in the form of functions



x	f(x)
−2	2
0	2
2	2

What Is Stationarity?

TS has a particular behaviour over time, there is a very high probability that it will follow the same in the future.

How to remove Stationarity?

1. Constant mean
2. Constant Variance
3. Autocovariance that does not depend on time

Tests to Check Stationarity

1 Rolling Statistics

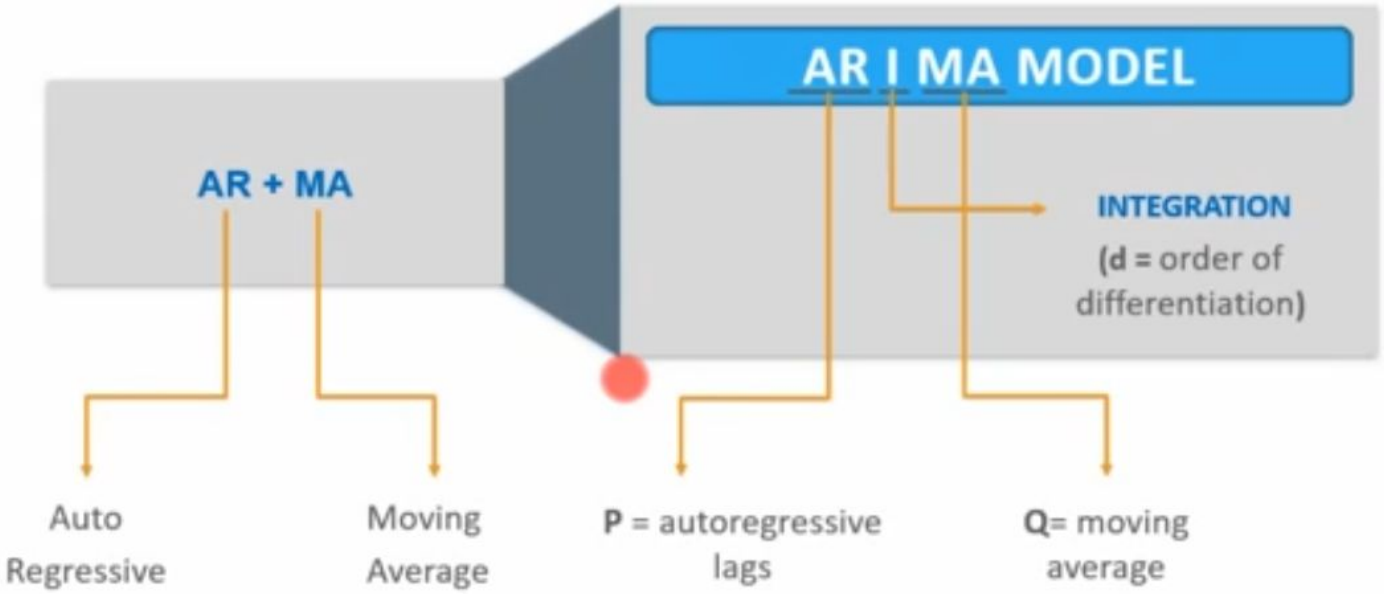
Plot the moving average or moving variance and see if it varies with time.

More of a visual technique.

2 ADCF Test

Null hypothesis is that the TS is non-stationary. The test results comprise of a *Test Statistic* and some Critical values.

ARIMA MODEL


AR I MA MODEL
AR + MA
INTEGRATION
(d = order of differentiation)
Auto Regressive
Moving Average
P = autoregressive lags
Q= moving average

5. Forecasting a Time Series

We saw different techniques and all of them worked reasonably well for making the TS stationary. Lets make model on the TS after differencing as it is a very popular technique. Also, its relatively easier to add noise and seasonality back into predicted residuals in this case. Having performed the trend and seasonality estimation techniques, there can be two situations:

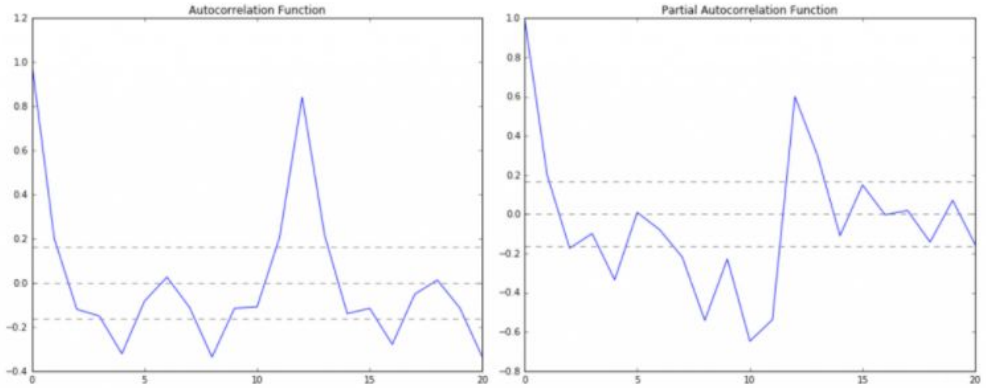
1. A **strictly stationary series** with no dependence among the values. This is the easy case wherein we can model the residuals as white noise. But this is very rare.
2. A series with significant **dependence among values**. In this case we need to use some statistical models like ARIMA to forecast the data.

Let me give you a brief introduction to **ARIMA**. I won't go into the technical details but you should understand these concepts in detail if you wish to apply them more effectively. ARIMA stands for **Auto-Regressive Integrated Moving Averages**. The ARIMA forecasting for a stationary time series is nothing but a linear (like a linear regression) equation. The predictors depend on the parameters (p,d,q) of the ARIMA model:

1. **Number of AR (Auto-Regressive) terms (p):** AR terms are just lags of dependent variable. For instance if p is 5, the predictors for x(t) will be x(t-1)….x(t-5).
2. **Number of MA (Moving Average) terms (q):** MA terms are lagged forecast errors in prediction equation. For instance if q is 5, the predictors for x(t) will be e(t-1)….e(t-5) where e(i) is the difference between the moving average at i^{th} instant and actual value.
3. **Number of Differences (d):** These are the number of nonseasonal differences, i.e. in this case we took the first order difference. So either we can pass that variable and put d=0 or pass the original variable and put d=1. Both will generate same results.

An importance concern here is how to determine the value of 'p' and 'q'. We use two plots to determine these numbers. Lets discuss them first.

```
plt.title('Partial Autocorrelation Function')
plt.tight_layout()
```



In this plot, the two dotted lines on either sides of 0 are the confidence interevals. These can be used to determine the 'p' and 'q' values as:

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.

g individual as well as combined effects. I will also print the

This tutorial is divided into four parts; they are:

1. What's Wrong with ARIMA
2. What Is SARIMA?
3. How to Configure SARIMA
4. How to use SARIMA in Python

What's Wrong with ARIMA

Autoregressive Integrated Moving Average, or ARIMA, is a forecasting method for univariate time series data.

As its name suggests, it supports both an autoregressive and moving average elements. The integrated element refers to differencing allowing the method to support time series data with a trend.

A problem with ARIMA is that it does not support seasonal data. That is a time series with a repeating cycle.

ARIMA expects data that is either not seasonal or has the seasonal component removed, e.g. seasonally adjusted via methods such as seasonal differencing.

For more on ARIMA, see the post:

- How to Create an ARIMA Model for Time Series Forecasting with Python

An alternative is to use SARIMA.

What is SARIMA?

Seasonal Autoregressive Integrated Moving Average, SARIMA or Seasonal ARIMA, is an

What is SARIMA?

Seasonal Autoregressive Integrated Moving Average, SARIMA or Seasonal ARIMA, is an extension of ARIMA that explicitly supports univariate time series data with a seasonal component.

It adds three new hyperparameters to specify the autoregression (AR), differencing (I) and moving average (MA) for the seasonal component of the series, as well as an additional parameter for the period of the seasonality.

> *A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA […] The seasonal part of the model consists of terms that are very similar to the non-seasonal components of the model, but they involve backshifts of the seasonal period.*

— Page 242, Forecasting: principles and practice, 2013.

How to Configure SARIMA

Configuring a SARIMA requires selecting hyperparameters for both the trend and seasonal elements of the series.

Trend Elements

There are three trend elements that require configuration.

They are the same as the ARIMA model; specifically:

- **p**: Trend autoregression order.
- **d**: Trend difference order.

How to Configure SARIMA

Configuring a SARIMA requires selecting hyperparameters for both the trend and seasonal elements of the series.

Trend Elements

There are three trend elements that require configuration.

They are the same as the ARIMA model; specifically:

- **p**: Trend autoregression order.
- **d**: Trend difference order.
- **q**: Trend moving average order.

Seasonal Elements

There are four seasonal elements that are not part of ARIMA that must be configured; they are:

- **P**: Seasonal autoregressive order.
- **D**: Seasonal difference order.
- **Q**: Seasonal moving average order.
- **m**: The number of time steps for a single seasonal period.

Together, the notation for an SARIMA model is specified as:

```
SARIMA(p,d,q)(P,D,Q)m
```

Where the specifically chosen hyperparameters for a model are specified; for example:

```
SARIMA(3,1,0)(1,1,0)12
```

ARIMAX (1, 0, 2)

SARIMAX (p, d, q) (P, D, Q, s)

Home | AirPassengers_Forecast

Not secure | bdlabs.edureka.co:50005/user/edureka_234431/notebooks/AirPassengers_Forecast.ipynb

Apps | Gmail | Edureka - Workplace | Quora | Edureka Blog — Wi | JupyterHub | GotoWebinars | Peopleworks | Complete guide to

jupyter AirPassengers_Forecast Last Checkpoint: Last Wednesday at 11:21 AM (unsaved changes)

Control Panel | Logout

File Edit View Insert Cell Kernel Widgets Help

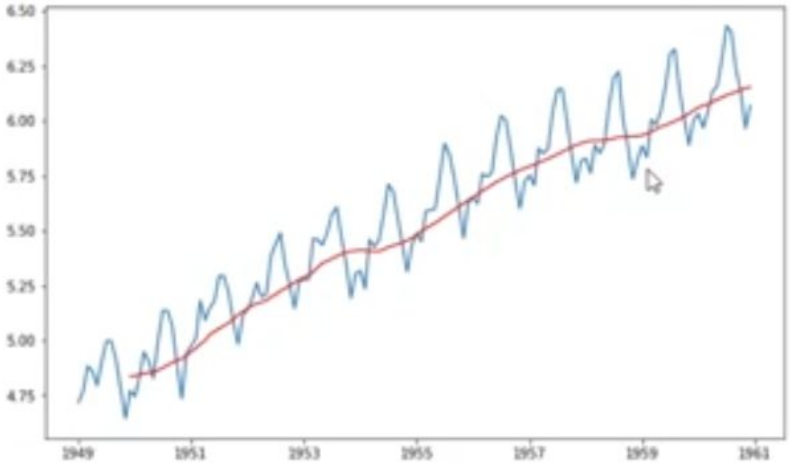
Trusted | Python 3

Code

In [11]:

```
movingAverage = indexedDataset_logScale.rolling(window=12).mean()
movingSTD = indexedDataset_logScale.rolling(window=12).std()
plt.plot(indexedDataset_logScale)
plt.plot(movingAverage, color='red')
```

Out[11]: [<matplotlib.lines.Line2D at 0x7f8a9a9b6198>]



Home | AirPassengers_Forecast

Not secure | bdlabs.edureka.co:50005/user/edureka_234431/notebooks/AirPassengers_Forecast.ipynb

Apps | Gmail | Edureka - Workplace | Quora | Edureka Blog — Wi | JupyterHub | GotoWebinars | Peopleworks | Complete guide to

jupyter AirPassengers_Forecast Last Checkpoint: Last Wednesday at 11:21 AM (autosaved)

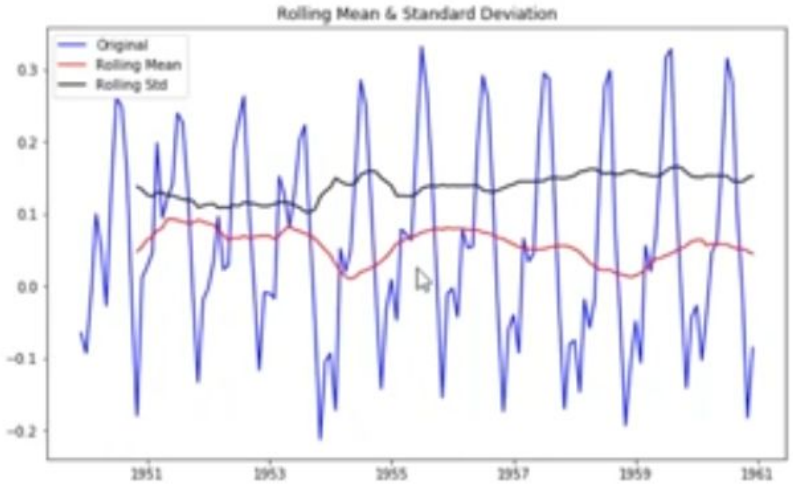
Control Panel | Logout

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3

Code

```
        dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)
```

In [14]:
```
test_stationarity(datasetLogScaleMinusMovingAverage)
```




```
Results of Dickey-Fuller Test:
Test Statistic                  -3.162908
```

References

1. General introduction
2. Some more Definitions