Time Series Modeling

ARIMA Model

Contents

Box-Jenkins (ARIMA) Models

- ARIMA (**A**uto **R**egressive **I**ntegrated **M**oving **A**verage) models are **Regression models that use lagged values of the dependent variable and/or random disturbance term as explanatory variables.**

- ARIMA models **rely heavily on the autocorrelation pattern** in the data.

- ARIMA models can also be developed in the presence of seasonality in the time series.

- ARIMA models thus **essentially ignore domain theory (by ignoring "traditional" explanatory variables)**

When to Use ARIMA Models

Little or nothing is known about the dependent variable being forecasted

The independent variables known to be important cannot be forecasted effectively

Objective is to obtain short term forecasts

Basic ARIMA Models

1. Autoregressive model of order p (AR(p)):

$$y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t$$

Where y_t depends on its *p* previous values

2. Moving Average model of order q (MA(q))

$$y_t = \delta + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q}$$

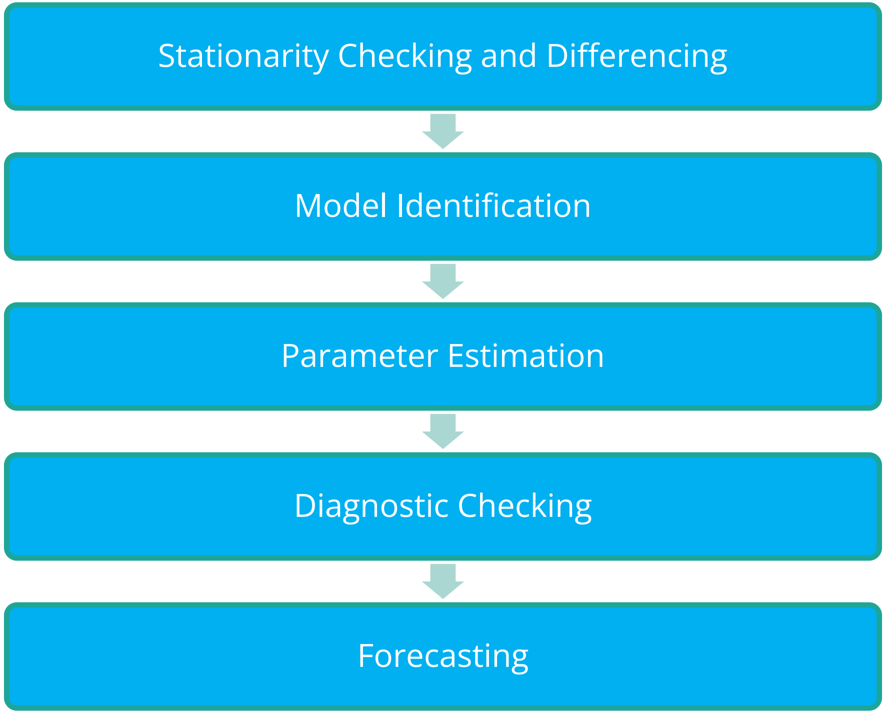
y_t depends on *q* previous random error terms

3. Autoregressive-Moving Average model of order p and q (ARMA(p,q))

$$\begin{aligned} y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} \\ + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} \end{aligned}$$

y_t depends on its *p* previous values and *q* previous random error terms

Five-Step Iterative Procedure



Step 1: Stationarity Checking

Differencing

- Differencing continues until stationarity is achieved

$$\boxed{\Delta y_t = y_t - y_{t-1}} \qquad \boxed{\Delta^2 y_t = \Delta(\Delta y_t) = \Delta(y_t - y_{t-1}) = y_t - 2y_{t-1} + y_{t-2}}$$

- The differenced series has n-1 values after taking the first-difference, n-2 values after taking the second difference, and so on

- The number of times that the original series must be differenced in order to achieve stationarity is called the order of integration, denoted by d ?

?

How many times should a series be differenced?

In practice, it is not required to go beyond second difference.

Case Study

Background

- Annual Sales for a specific company from year 1961 to 2017

Objective

- To develop time series model and forecast sales for next 3 years

Available Information

- Number of cases: 57
- Variables: Year, sales(in 10's GBP)

Data Snapshot

turnover_annual data

Variables

Observations on Discrete Time Scale

Year	sales
1961	224786
1962	230034
1963	236562
1964	250960
1965	261615
1966	268316
1967	283589
1968	280160
1969	301422
1970	308018
1975	392503

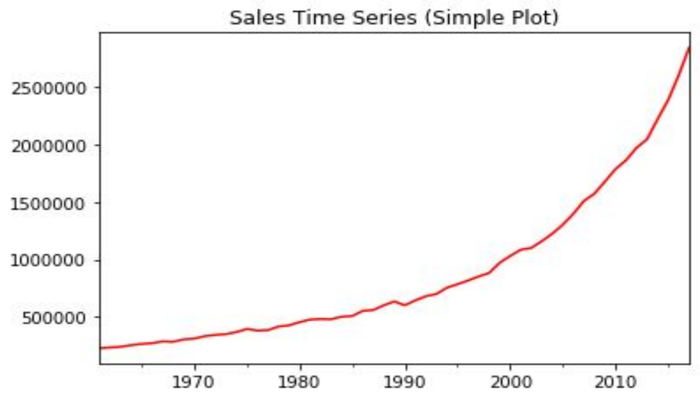
Columns	Description	Type	Measurement	Possible values
Year	Financial Year	Numeric	-	-
sales	sales(in 10's GBP)	Numeric	In British Pound	Positive values

Creating and Plotting Time Series in Python

```
#Importing turnover_annual data
import pandas as pd
salesdata=pd.read_csv('turnover_annual.csv')
#Creating and Plotting a Time Series Object
rng = pd.date_range('01-01-1961','31-12-2017',freq='Y')
s = salesdata.sales.values
salesseries = pd.Series(s, rng)

salesseries.plot(color='red', title ="Sales Time Series
(Simple Plot)")
```

- **date_range()** creates pandas date object.
- **freq ='Y' indicates yearly data**
- **pd.Series()** creates time series object
- Plot function gives line chart



Interpretation :

- The time-series clearly shows a positive trend.

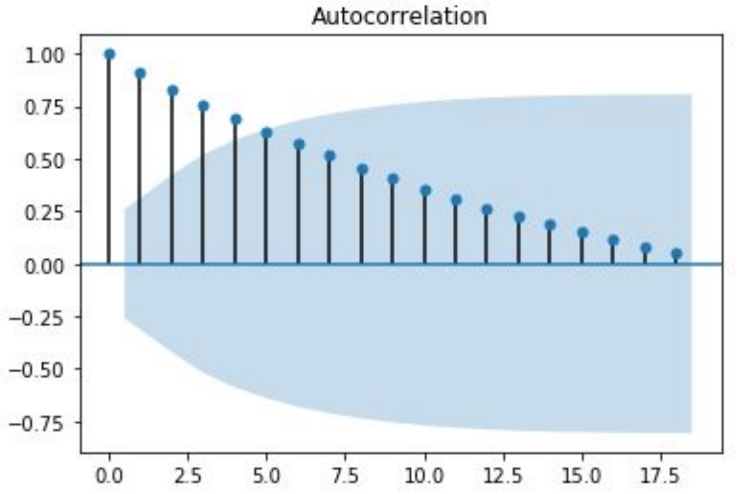
Checking Stationarity – Correlogram

ACF Plot

```
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plot_acf(salesseries)
```

- **plot_acf()** returns an ACF (Auto Correlation Function) plot.

Output



Interpretation :

- We can observe that there is a very slow decay which is a sign of Non-stationarity.

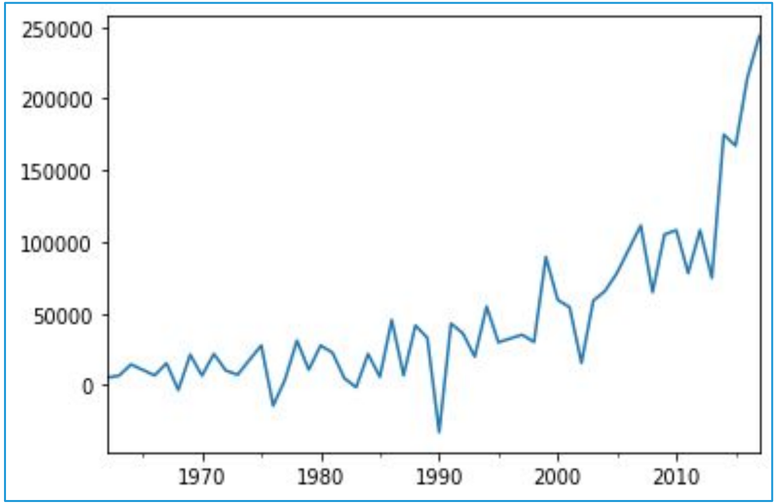
Plot of 1st Order Differenced Time Series

```
# Creating and Plotting a Difference Series
```

```
from statsmodels.tsa.statespace.tools import diff
salesdiff = diff(salesseries)
salesdiff.plot()
```

- diff() gives 1st order differences
- plot function gives line chart for differenced series

```
# Output
```



Interpretation :

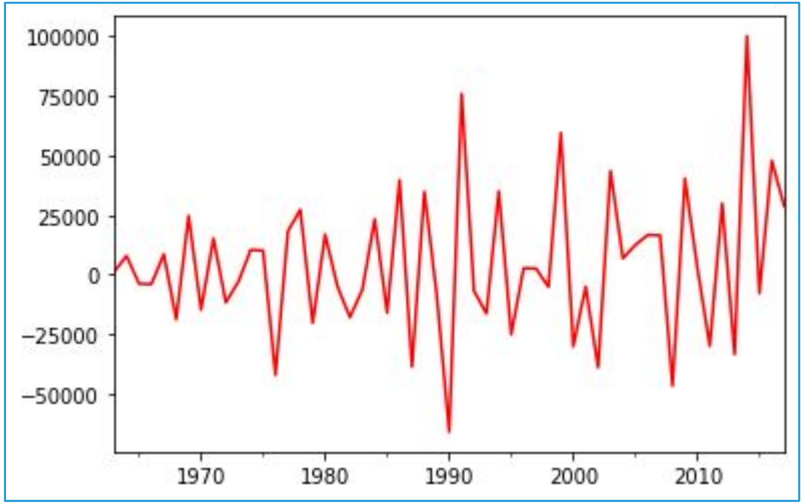
- Even after first order differencing, the series looks non-stationary.

Plot of 2nd Order Differenced Time Series

#Creating and Plotting 2nd Difference Series

```
salesdiff2 = diff(salesdiff)
salesdiff2.plot(color='red')
```

Output

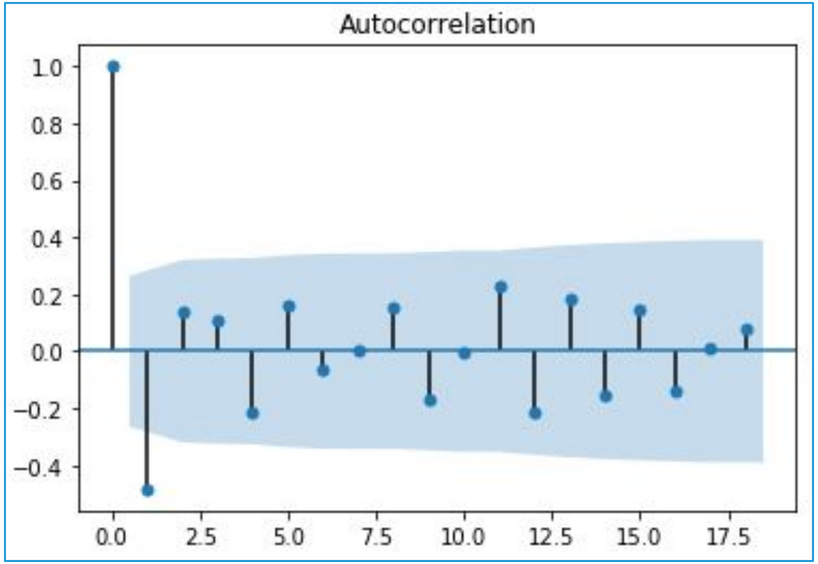


Interpretation :

- After 2nd order differencing, the series looks **stationary**.

Correlogram for 2nd Order Differenced Time Series

```
# ACF Plot
plot_acf(salesdiff2)
# Output
```



Interpretation :

- Stationarity is achieved with 2nd order difference.

Dickey Fuller Test

Install “arch”

```
pip install arch
```

Import “ADF” from library “arch”

```
from arch.unitroot import ADF

adf = ADF(salesseries,lags=0,trend='nc')
adf.summary()
```

- **ADF()** performs a Dickey Fuller unit root test on time series data.
- **lags=** allows to mention the number of lags to use in the ADF regression. We have used zero.
- **trend=‘nc’** specifies no trend and constant in regression

Output

```
   Augmented Dickey-Fuller Results
=====================================
Test Statistic                 19.275
P-value                         1.000
Lags                                0
-------------------------------------

Trend: No Trend
Critical Values: -2.61 (1%), -1.95 (5%), -1.61 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

Interpretation :

- Time series is non-stationary as value of test statistic is greater than 5% critical value.

Dickey Fuller Test

```
# Checking stationarity for series with difference of order 2
adf = ADF(salesdiff2,lags=0,trend='nc')
adf.summary()
```

```
# Output
```

```
   Augmented Dickey-Fuller Results
=====================================
Test Statistic                -11.908
P-value                         0.000
Lags                                0
-------------------------------------

Trend: No Trend
Critical Values: -2.61 (1%), -1.95 (5%), -1.61 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

Interpretation :

- Time series is stationary as value of test statistic is less than 5% critical value.

Step 2: Model Identification

Model Identification

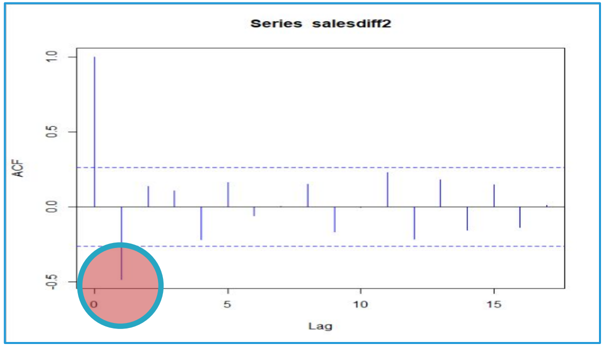
- When the data are confirmed stationary, proceed to tentative identification of models through **visual inspection of correlogram and partial correlogram**

Model	AC	PAC
	Dies down	Cuts off after lag p
	Cuts off after lag q	Dies down
	Dies down	Dies down

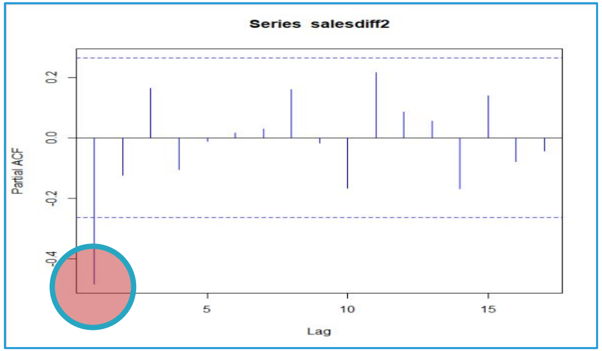
Model Identification

- ARIMA model is expressed as arima (p,d,q) where
  - p = no. of autoregressive terms
  - d = order of differencing
  - q = no. of moving average terms

ACF Plot



PACF Plot



- ACF and PACF correlograms will help in determining the MA and AR values respectively.

<u>Indicative Model :
arima(2,2,2)</u>

Step 3: Parameter Estimation

Parameter Estimation in Python

```
# Simple Estimation
from statsmodels.tsa.arima_model import ARIMA
model = ARIMA(salesseries, order=(2, 2, 2)).fit(trend='nc')
```

- **ARIMA()** fits a model to a univariate time series.
- **order=** argument gives the model (p,d,q) order.

```
model.params
model.aic
```

params() and **aic()** return the model coefficients and AIC value.

```
# Output
ar.L1.D2.None   -1.235413
ar.L2.D2.None   -0.670320
ma.L1.D2.None    0.785072
ma.L2.D2.None    0.330062
dtype: float64
```

```
# Output
1285.9836066562698
```

Interpretation :

- Smaller the AIC value, better is the model. We need to try out various combinations of AR and MA terms to arrive at final model.

Automatic Estimation of Model Parameters

```
# Automatic Model Identification and Parameter Estimation
```

```
import pmdarima as pm
model = pm.auto_arima(salesseries,max_p=2, max_q=2, d=2,
                      seasonal=False, trace=True)
```

- **auto_arima()** generates the best order arima model. The function conducts a search over possible model within the order constraints provided.
- **trace=** True returns the list of all models considered.
- **max_p and max_q** gives maximum values of p and q respectively.
- **seasonal**= allows you to specify whether to fit a seasonal ARIMA or not.

Automatic Estimation of Model Parameters

```
# Output
```

```
Fit ARIMA: order=(2, 2, 2) seasonal_order=(0, 0, 0, 0); AIC=1294.586, BIC=1306.630, Fit
time=0.056 seconds
Fit ARIMA: order=(0, 2, 0) seasonal_order=(0, 0, 0, 0); AIC=1295.387, BIC=1299.402, Fit
time=0.006 seconds
Fit ARIMA: order=(1, 2, 0) seasonal_order=(0, 0, 0, 0); AIC=1291.568, BIC=1297.590, Fit
time=0.019 seconds
Fit ARIMA: order=(0, 2, 1) seasonal_order=(0, 0, 0, 0); AIC=1292.152, BIC=1298.174, Fit
time=0.018 seconds
Fit ARIMA: order=(0, 2, 0) seasonal_order=(0, 0, 0, 0); AIC=1294.532, BIC=1296.540, Fit
time=0.006 seconds
Fit ARIMA: order=(2, 2, 0) seasonal_order=(0, 0, 0, 0); AIC=1291.358, BIC=1299.387, Fit
time=0.022 seconds
Fit ARIMA: order=(2, 2, 1) seasonal_order=(0, 0, 0, 0); AIC=1292.717, BIC=1302.754, Fit
time=0.044 seconds
Fit ARIMA: order=(1, 2, 1) seasonal_order=(0, 0, 0, 0); AIC=1293.092, BIC=1301.122, Fit
time=0.029 seconds
Total fit time: 0.203 seconds
```

```
ARIMA(order=(2, 2, 0))
```

Lowest AIC

Interpretation :

- Model with the lowest AIC value is selected as the best model.

ARIMA Model Using BEST Order

```
# Run arima() for cross checking parameters based on model suggested
# by auto.arima
```

```python
from statsmodels.tsa.arima_model import ARIMA
model = ARIMA(salesseries, order=(2, 2, 0)).fit(trend='nc')

model.params
model.aic
```

```
# Output
```

```
ar.L1.D2.None    -0.506320
ar.L2.D2.None    -0.101797
dtype: float64
```

```
# Output
```

```
1285.114088844849
```

Step 4: Diagnostic Checking

Residual Analysis

If an ARMA(p,q) model is an adequate representation of the data generating process then the residuals should be 'White Noise'

- White Noise time series has **zero mean, constant variance and zero covariance with lagged time series.**

- Residual plot is commonly used method for checking if the residuals are white noise process.

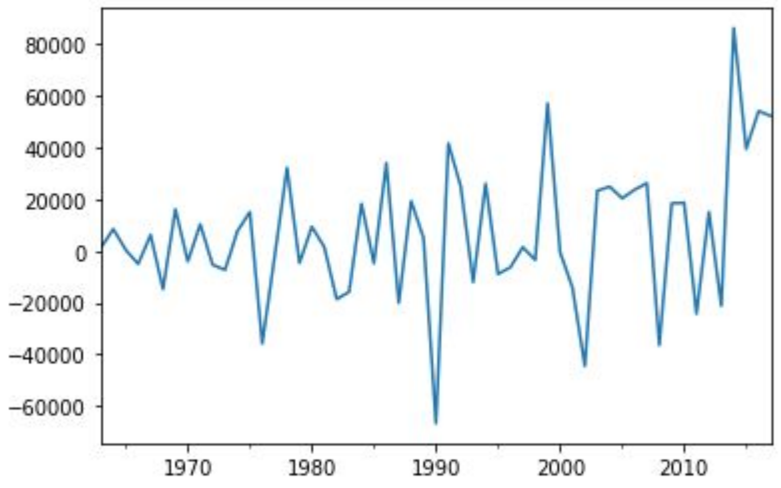
Residual Plot In Python

```
resi = model.resid

resi.plot()
```

resid() calculates residual values.

```
# Output
```



Interpretation :

- Errors follow white noise process.

Step 5: Forecasting

Forecasting

```
# Forecast for next 3years
model.forecast(steps=3)
```

forecast() function here gives predicted values for 3 years

```
# Output
(array([3072357.58495709, 3303466.18670477, 3533054.4598124 ]),
 array([27092.85937931, 48699.96239955, 75757.91735836]),
 array([[3019256.55633543, 3125458.61357874],
        [3208016.0143532 , 3398916.35905634],
        [3384571.67024626, 3681537.24937854]]))
```

Interpretation :

- **forecast()** returns three arrays:
- array of three forecasts
- array of these standard error of the forecasts
- array of the confidence interval for the forecast

Quick Recap

Stationarity Checking	• Plot correlogram using plot_`acf()` and ADF() for Dickey-Fuller Test
Model Identification	• Tentative identification of models through visual inspection of correlogram and partial correlogram
Parameter Estimation	• `auto_arima()` is recommended for obtaining best ARIMA model • It uses AIC as the model selection criteria
Diagnostic Checking	• `Residual plot` for checking whether errors follow white noise process
Forecasting	• Use `forecast()` to generate forecasts