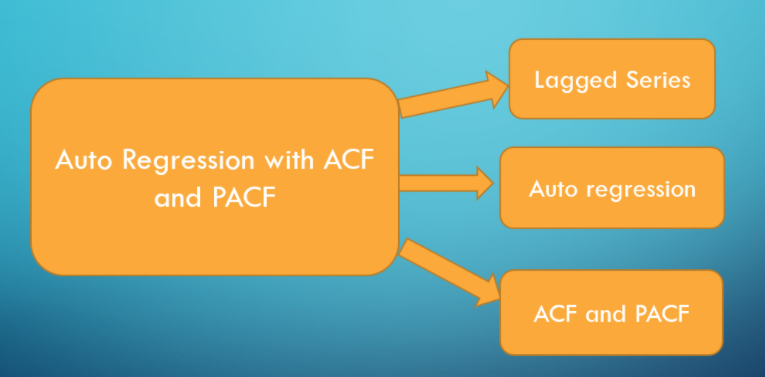
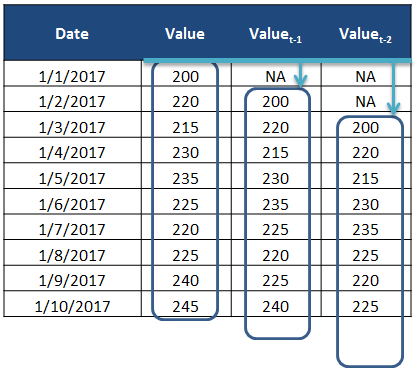
Auto Regression with ACF and PACF

**Auto regression** is a time series model that uses observations from previous time steps as input to a **regression** equation to predict the value at the next time step. It is a very simple idea that can result in accurate forecasts on a range of time series problems. A statistical model is autoregressive if it predicts future values based on past values.**Auto regression** analysis is a standard technique in signal processing where a linear predictor estimates the value of each sample of a signal by a linear.



### **What is Lag in a time series**

A “lag” is a fixed amount of passing time; One set of observations in a [time series](https://www.statisticshowto.com/timeplot/) is plotted (lagged) against a second, later set of data. The kth lag is the time period that happened “k” time points before time i. For example:  
Lag1(Y2) = Y1 and Lag4(Y9) = Y5.



**Lag Time Series Example:**

# Original time series = {21,22,21,20,19}

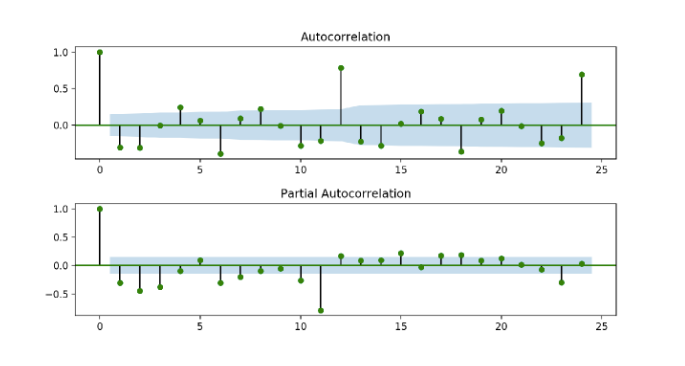
# Lagged time series by 1 lag = {22,21,20,19}

# Lag 1 correlation will be correlation between {21,22,21,20} and {22,21,20,19}

Lag time series is always come by some numbers lags. In this above Lag time series example we show Original time series (21,22,21,20,19) and lagged 1-time series (22,21,20,19). In original time series T will be 21 and T-1 will be 22. In lag 1-time series shift 1space value from the left side and take the remaining value hence our lag 1-time series start from 22. In this above example first value out of this series from original time series. lag1 correlation will be the correlation between original time series and lagged 1-time series. In our original time series, we have 5 observations and lagged 1-time series we have 4 observations. Hence the last observation of original time series has been taken out similarly lag 2 and lag 3 correlation.

What is ACF and PACF in time series?

A **time series** can have components like trend, seasonality, cyclic and residual. **ACF** considers all these components while finding correlations hence it's a 'complete auto-correlation plot'. **PACF** is a partial auto-correlation function.



**ACF:**

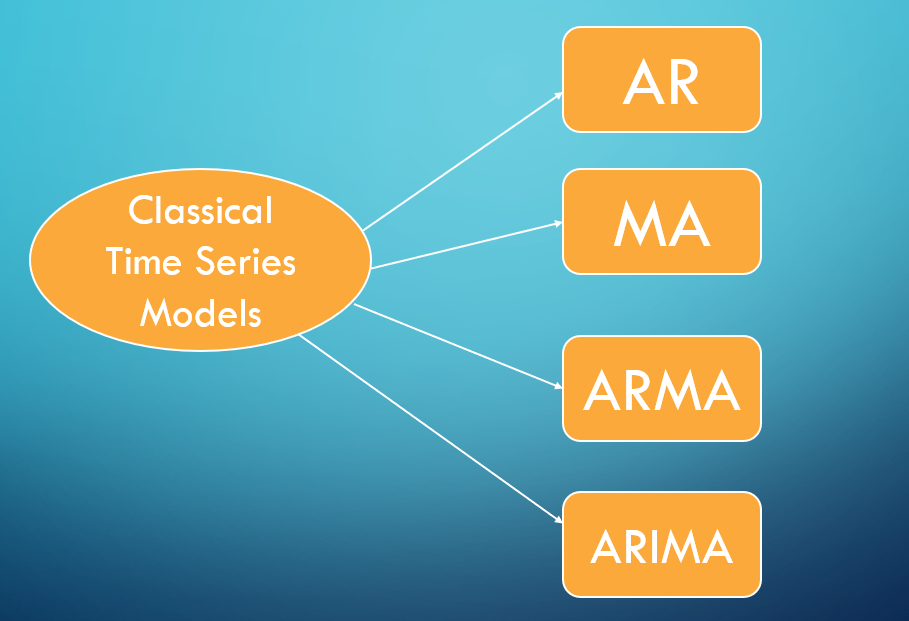
**ACF** is an (complete) auto-correlation function which gives us values of auto-correlation of any series with its lagged values. The correlation between the observation at the time spot and the observation at previous time spot. We plot these values along with the confidence band. We have an ACF plot. In simple terms, it describes how well the present value of the series is related with its past values. A time series can have components like trend, seasonality, cyclic and residual. ACF considers all these components while finding correlations hence it’s a ‘complete auto-correlation plot’.

**PACF:**

# ****PACF**** is a partial auto-correlation function an indirect function to find Auto correlation after removing the relationship explained by previous lags. The correlations between observations at two time spots given that we consider both observations are correlated to observations at other time spot. For example, todays stock price can be correlated to the day before yesterday, and yesterday can also be correlated to the day before yesterday. Then PACF of yesterday is the real correlation between today and yesterday after taking out the influence of the day before yesterday.

Basically instead of finding correlations of present with lags like ACF, it finds correlation of the residuals (which remains after removing the effects which are already explained by the earlier lag(s)) with the next lag value hence ‘partial’ and not ‘complete’ as we remove already found variations before we find the next correlation. So if there is any hidden information in the residual which can be modelled by the next lag, we might get a good correlation and we will keep that next lag as a feature while modelling. Remember while modelling we don’t want to keep too many features which are correlated as that can create multicollinearity issues. Hence we need to retain only the relevant features.

Time Series Models AR, MA, ARMA, ARIMA



Autoregressive (AR) models

In a multiple regression model, we forecast the variable of interest using a linear combination of predictors. In an auto regression model, we forecast the variable of interest using a linear combination of past values of the variable. The term auto regression indicates that it is a regression of the variable against itself.

Thus, an autoregressive model of order pp can be written as:

**yt=c+ϕ1yt−1+ϕ2yt−2+⋯+ϕpyt−p+εt**

where εt is white noise. This is like a multiple regression but with lagged values of yt as predictors. We refer to this as an **AR(**p**) model**, an autoregressive model of order p.

**Example:**

In this example we create our own linear data like:

Data = [10,12,14,16,18,20,20,22]

We use this data and create Autoregressive (AR) Model.

First of all, we load all **important libraries** like:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import os

import seaborn as sns

import warnings

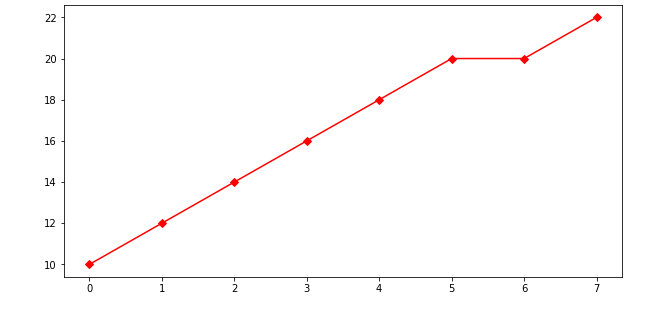
warnings.filterwarnings('ignore')

from statsmodels.tsa.ar\_model import AutoReg

Here we load numpy, pandas, matplotlib, seaborn, and AutoReg. AutoReg is very important library for creating AR model. We import AutoReg library from statsmodels.tsa.ar\_model import AutoReg.



Here we create Own dataset [10,12,14,16,18,20,20,22] and plot this data using matplotlib library.



In this visualization we have showing 7 observations. These 7 observations like upward trend. We use these 7 past observations and predict future 8-12 observations by using Auto Regression (AR) model.

**Code:**

#Creating Auto regression (AR) model

AR\_model = AutoReg(data, lags=1)

AR\_model\_fit = AR\_model.fit() #Fitting model

# making predictions future value between 8-12 Observation

y\_pred = AR\_model\_fit.predict(8,12)

print(y\_pred)

pred\_list = y\_pred.tolist()

combinedlist = data+pred\_list

combinedlist

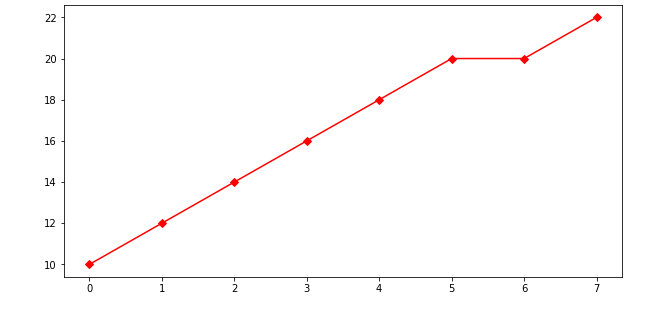
# Visualization of AR Model

plt.figure(figsize=(10,5))

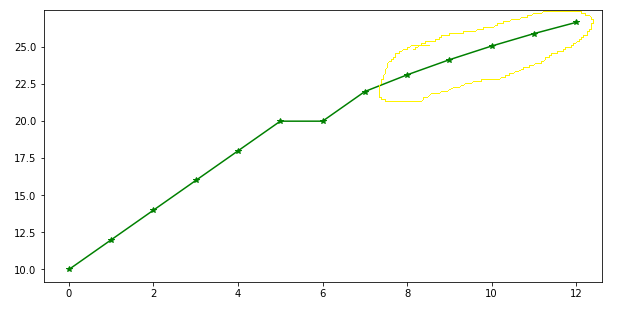
plt.plot(combinedlist,color='Green',Marker='\*')

plt.show()

**Original**



**AR Model**



As per above both visualization we predict 8,9,10,11 and 12 observations using AR model. These predicted 5 values are also showing upward trend. In this AR model we previous data like [10,12,14,16,18,20,20,22] and also taking lag value is 1. After taking these to parameters we fit the AR model and predict future [8-12] observation. That predicted observations like: [23.125 24.14453125 25.06848145 25.90581131 26.6646415 ].

Moving Average (MA) Model

In time series analysis, the moving-average model, also known as moving-average process, is a common approach for modelling univariate time series. The moving-average model specifies that the output variable depends linearly on the current and various past values of a stochastic term.

Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

**yt=c+εt+θ1εt−1+θ2εt−2+⋯+θqεt−q**

where εt is white noise. We refer to this as an **MA(**q**) model**, a moving average model of order qq. Of course, we do not observe the values of εt, so it is not really a regression in the usual sense.

Notice that each value of yt can be thought of as a weighted moving average of the past few forecast errors. However, moving average models should not be confused with the moving average smoothing. A moving average model is used for forecasting future values, while moving average smoothing is used for estimating the trend-cycle of past values.

**Example:**

In this example we create our own linear data like:

Data = [22,23,23,25,26.7,27,28,28,30]

We use this data and create Moving Average (MA) Model.

First of all, we load all **important libraries** like:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import os

import seaborn as sns

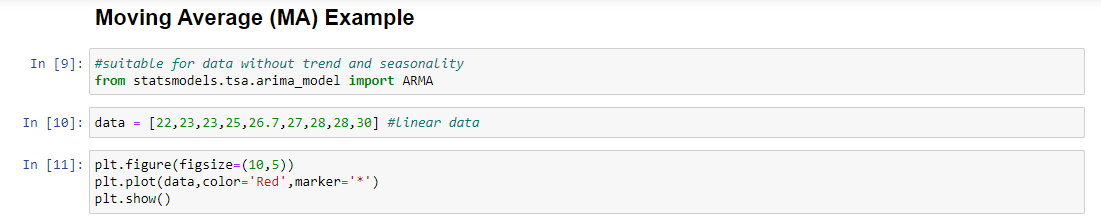
import warnings

warnings.filterwarnings('ignore')

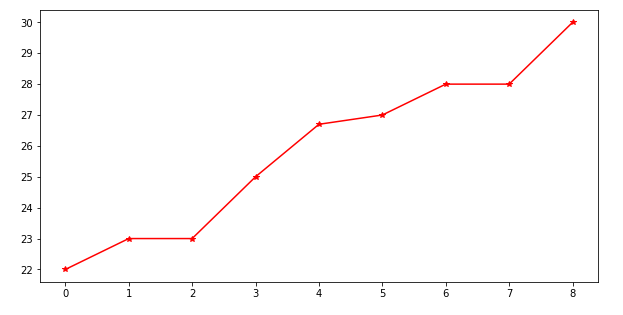
#suitable for data without trend and seasonality

from statsmodels.tsa.arima\_model import ARMA

Here we load numpy, pandas, matplotlib, seaborn, and ARMA. ARMA is very important library for creating MA Model. We import ARMA library from statsmodels.tsa.arima\_model.



Here we create Own dataset [22,23,23,25,26.7,27,28,28,30] and plot this data using matplotlib library.



In this visualization we have showing 8 observations. These 8 observations like upward trend. We use these 8 past observations and predict future 9-12 observations by using Moving Average (MA) model.

**Code:**

#suitable for data without trend and seasonality

from statsmodels.tsa.arima\_model import ARMA

data = [22,23,23,25,26.7,27,28,28,30] #linear data

plt.figure(figsize=(10,5))

plt.plot(data,color='Red',marker='\*')

plt.show()

# fit model

MA\_model = ARMA(data, order=(0, 1)) #model with AR=0 and MA=1

MA\_model\_fit = MA\_model.fit(disp=False)

# make prediction

y\_pred = MA\_model\_fit.predict(9,12)

print(y\_pred)

pred\_list = y\_pred.tolist()

combinedlist = data+pred\_list

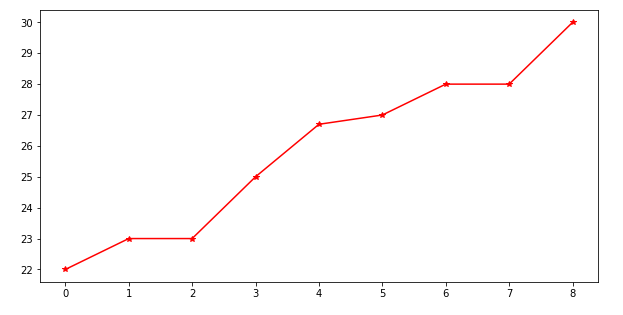
combinedlist

plt.figure(figsize=(10,5))

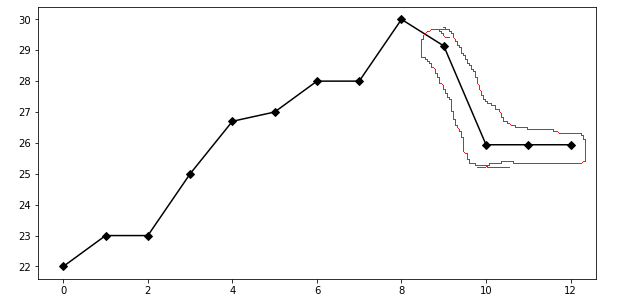
plt.plot(combinedlist,color='Black',marker='D')

plt.show()

**Original**



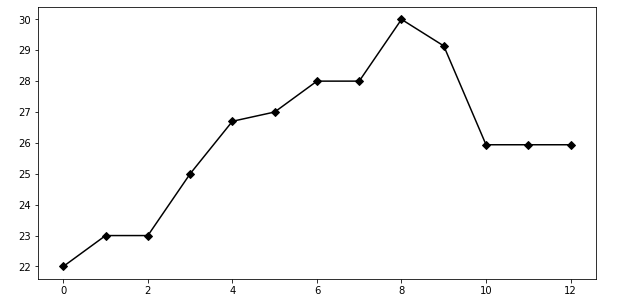
## Moving Average (MA)



Above both visualization we predict 9,10,11 and 12 observations using MA model. Now we see this char is very different from AR chart. Auto Regression will internally fit regression equation based on what was the past trend. But here Moving Average Model take the errors of the residuals and then it will forecast the prediction value. In real world scenario AR and MA independently do not solve the Business problems. In this MA model we take data like [22,23,23,25,26.7,27,28,28,30] and also taking order value is (0,1). Here 0-means AR model and 1-means MA model so that’s way we choose (0,1). After taking these to parameters we fit the MA model and predict future [9-12] observation. That predicted observations

like: [29.13966432 25.93999988 25.93999988 25.93999988]

after that we create new list and combine this two list like actual data and predicted data in one single list and then plot the visualization.



## Autoregressive Moving Average (ARMA)

In the statistical analysis of time series, autoregressive–moving-average (ARMA) models provide a parsimonious description of a (weakly) stationary stochastic process in terms of two polynomials, one for the auto regression (AR) and the second for the moving average (MA).

ARMA (p, q) models have a rich history in the time series literature, but they are not nearly as common in ecology as plain AR(p) models. As we discussed in lecture, both the ACF and PACF are important tools when trying to identify the appropriate order of p and q. Here we will see how to simulate time series from AR(p), MA(q), and ARMA (p, q) processes, as well as fit time series models to data based on insights gathered from the ACF and PACF.

We can write an ARMA (p, q) as a mixture of AR(p) and MA(q) models, such that

**xt=ϕ1xt−1+ϕ2xt−2+⋯+ϕpxt−p+wt+θwt−1+θ2wt−2+⋯+θqxt−q**

and the wt are white noise.

**Example:**

Here we have import TimeSeries.csv data set. This dataset available on our GitHub.

Dataset Link:

<https://github.com/SMIIT-Projects/Time-Series-AR-MA-ARMA-Models>

In this dataset having a 204 rows and 2 columns. First column name is ‘Date’ and second one is ‘Value’. As per the Time Series first of all we need to convert ‘Date’ column in to datetime format and once it will convert then set index as a ‘Date’ column. After set index as date then we have only 1 column in our dataset that name is value. Now our shape of data is 204 rows and 1 column. In this dataset our index starts from 01-07-1991 to 01-06-2008.

**Code:**

import pandas as pd

df = pd.read\_csv('TimeSeries.csv', parse\_dates=['Date'], index\_col='Date')

df.shape

df.head()

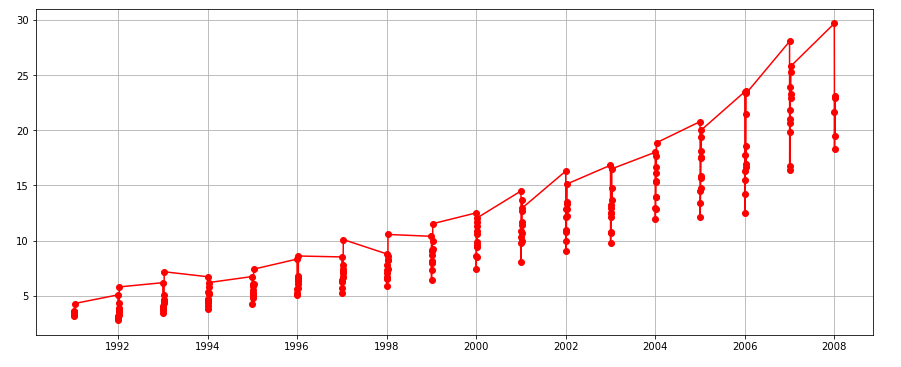
import matplotlib.pyplot as plt

plt.grid()

plt.rcParams.update({'figure.figsize': (15,6)})

plt.plot(df['Value'],color='Red',marker='o')

plt.show()



**Fit ARMA Model**

import warnings

warnings.filterwarnings('ignore')

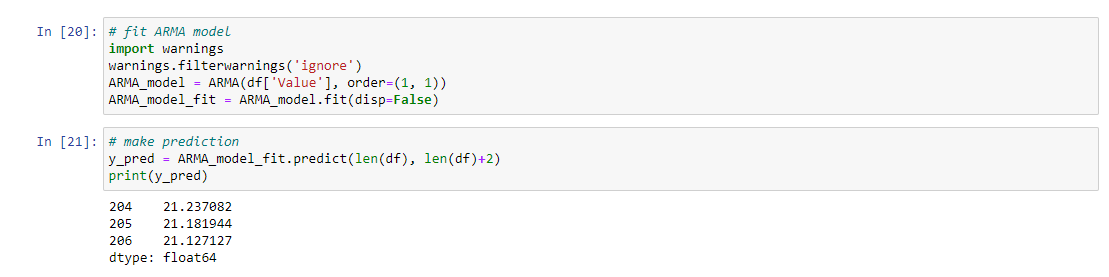
ARMA\_model = ARMA(df['Value'], order=(1, 1))

ARMA\_model\_fit = ARMA\_model.fit(disp=False)

# make prediction

y\_pred = ARMA\_model\_fit.predict(len(df), len(df)+2)

print(y\_pred)



As per the visualization we predict below value using ARMA model.

**204 21.237082**

**205 21.181944**

**206 21.127127**

In this ARMA model we take order value is (1,1). Here first 1-means AR model and second 1-means MA model so that’s way we choose (1,1). After taking these to parameters we fit the ARMA model and predict future value.

These Blog and Python Coding is available on our GitHub.

**GitHub Link:**

<https://github.com/SMIIT-Projects/Time-Series-AR-MA-ARMA-Models>

## **Tools and Technologies:**

The Code is written in Python 3.8.5.

## **Used libraries:**

numpy==1.20.1

pandas==1.2.3

matplotlib==3.3.4

seaborn==0.11.1

statsmodels==0.12.2

Thank You!