Time Series Analysis

* Forecasting method

**Simple Moving Average (SMA)**

* We start with a time series, then suppose we have some fixed length window. Then at each point in the Time series, we drag the window along and we calculate the sample mean of all the points in that window.
* Why use it →
  + we might want to calculate mean/variance of stock's returns
  + But we have volatility clustering in time series:
    - we can use all values
    - or use only recent values for calculation → better as it accommodates with changing values
* In python, use pandas for calculating this. Using pandas' rolling, we can calculate mean, var, cov, etc of the given window
* SMA usually lags behind the normal time series. This effect becomes more pronounced as window size gets larger

**Weighted Moving Average (WMA):**

* In the above, we have all the terms of the windows weighed equally. We can take a weighted average of the values of the moving window to get WMA. Usually more recent values are assigned more weights than past values

**Exponential Weighted Moving Average (EWMA)**

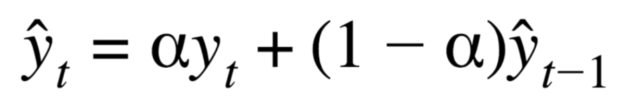
* Also called exponential smoothing, or low pass filter
* common in ml, stats, finance and signal processing
* moving average at time t is the weighted average of value at time t, and EMA of time t-1

**Value of alpha**

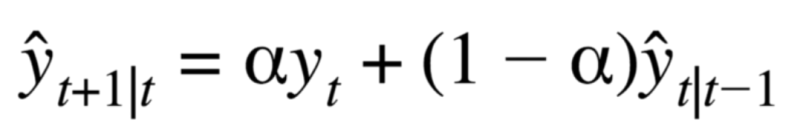
* Alpha is something like a decay factor
* usually b/w 0 and 1

**Simple Exponential Smoothing (SES) for forecast**

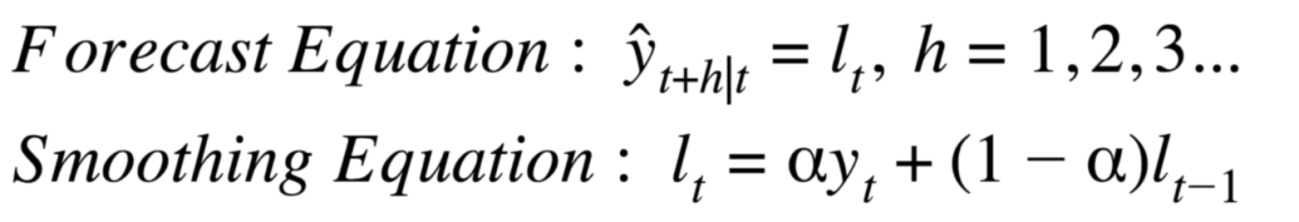
* for forecasting non-trending, non-seasonal tie series
* This is exactly same as EWMA, just we are using different notations for forecasting
* simple ses model:



* alpha is the smoothing parameter
* Forecast model: The below is a forecast model. Before this, we were not forecasting, just calculating averages

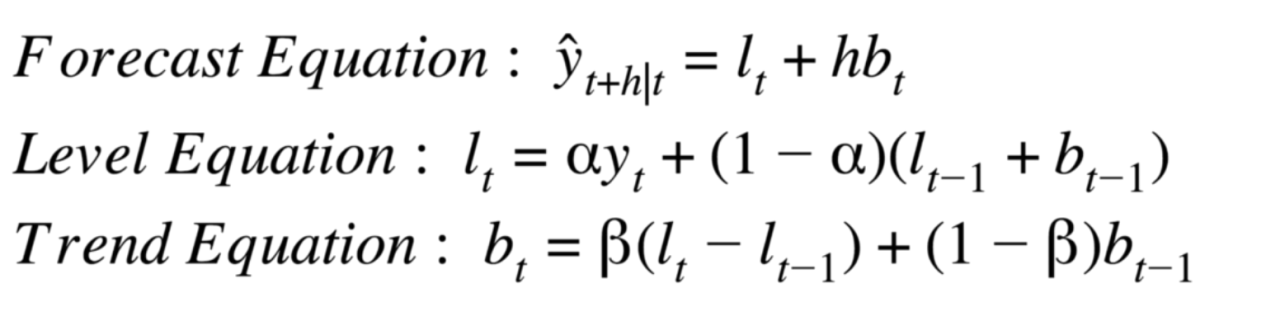


* Now we will see the how holt model is developed from ses model:

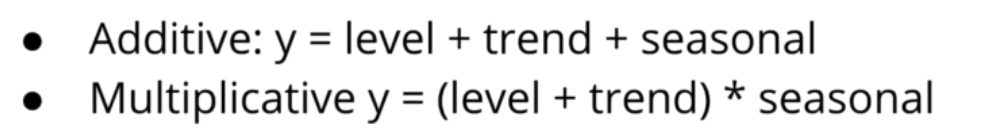


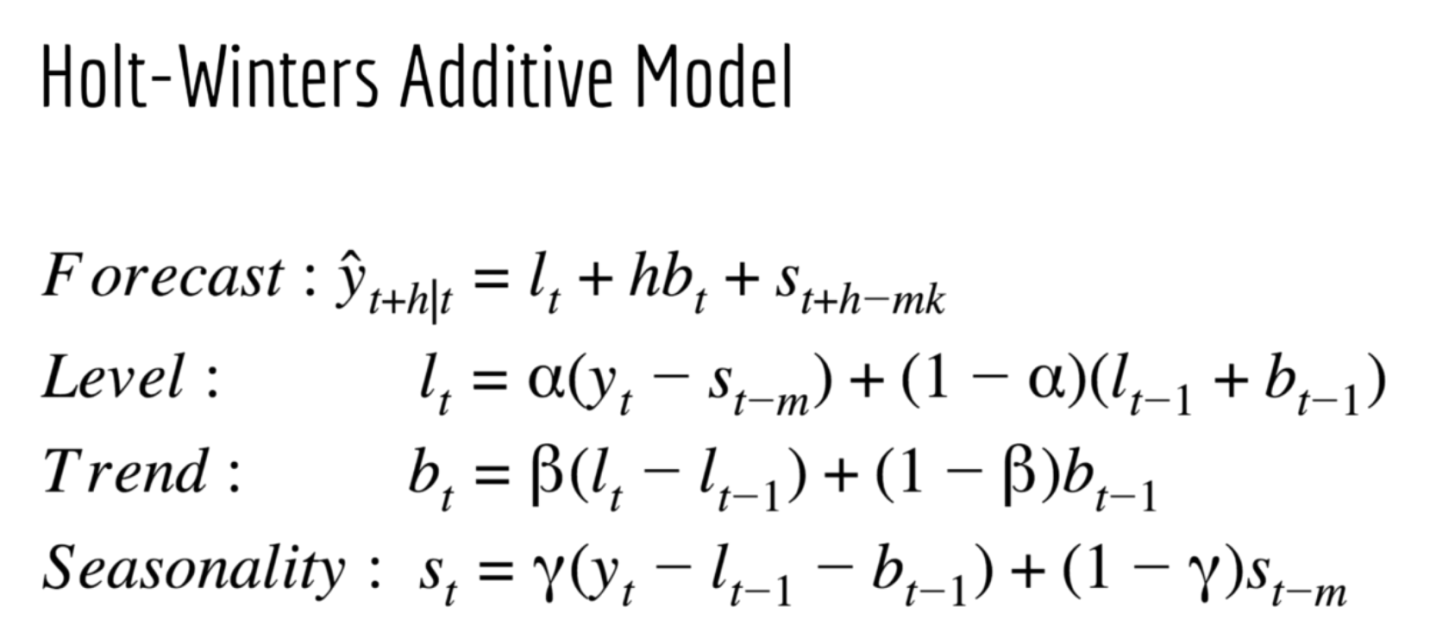
* This means that now, lt is not the forecast, but the forecast is based on lt. For this model, the forcast is simply assigned to be lt.
* lt  = level = average value of the signal around which the actual signal may fluctuate
* forecast is a constant value, no matter how many steps you want to forecast
* EWMA, SMA are just methods of estimating mean

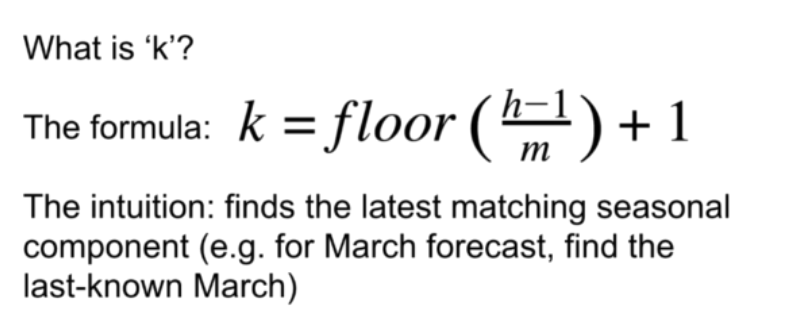
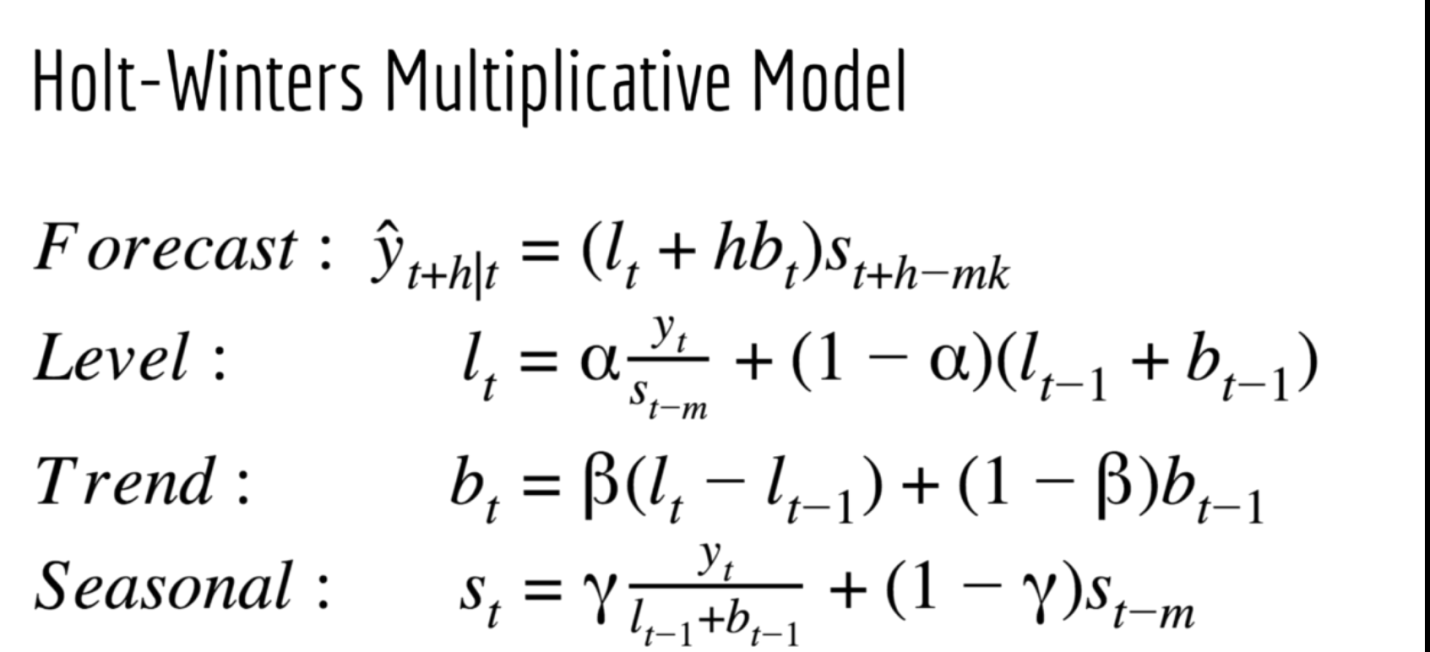
**Holt Model**

* Trending, non-seasonal series
* 
* Forecast eqn is the equation of line. Lt  represents the interncept, bt  = slope
* Level equation -> still something like exponential smoothing, but except of yesterday’s forecast, we’re using lt-1+bt-1
* Trend eqn -> still has a exponential smoothing type, but this time, the present value of bt is estimated by lt-lt-1
* Alpha, beta are estimated based on minimizing the Mean Squared Error.

**Holt-Winter's Model**

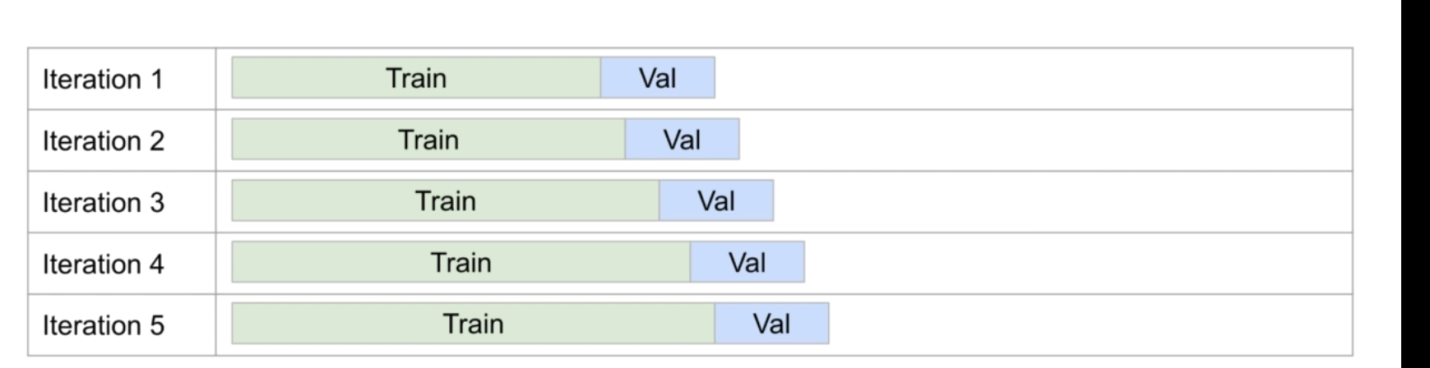
* Trending, seasonal series
* Adds a seasonal component to Holt’s model
* We can have two models:
* 
* For additive model, used when seasonal component is fixed each year, and we just need to add it to the model
* For multiplicative model, use when seasonal variations are proportionate to the rtrend values/level values

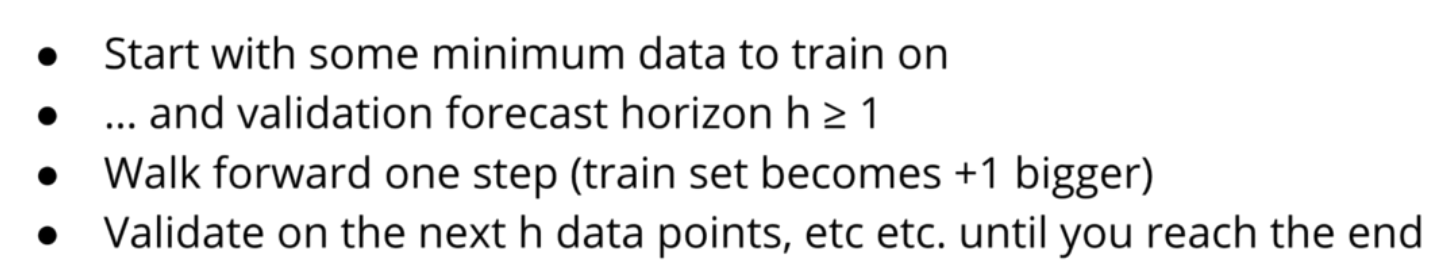
]

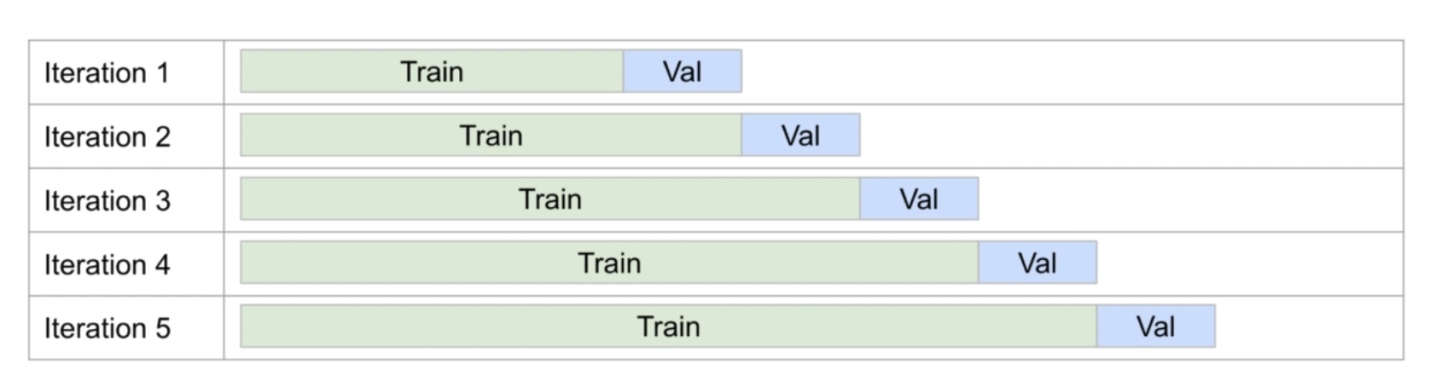
* m = frequency/year
* period of seasonal component/cycle/ length of time it takes for the signal to repeat itself
* 
* K is an number chosen so that we look back appropriate number of steps back in the data to choose correct seasonal component
* Level formula -> we subtract the seasonal component from y.
* Trend eqn is same
* Seasonality -> exp moving average. Old value comes from 1 period ago, not from one time step ago. For the actual value of seasonality, we calculate it as actual value – trend-level
* 
* In stats model, you can even get a multiplicative trend, use this when trend is non-linear

# Walk Forward Validation

* Data is prone to overfit. … why we need a cross validation set…..
* 
* This is regular machine learning CV.
* But we cannot use this for Time series, because there is time dependence of the data. You cannot split the data randomly
* So, we use wal forward validation





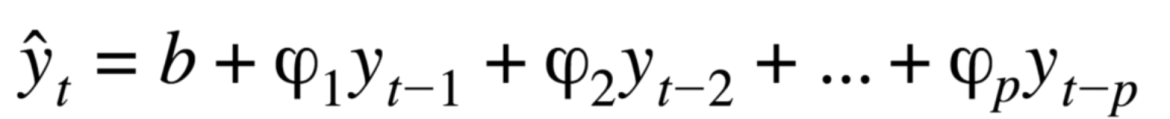
* Not necessary to use all the past data, if it is no longer useful
* Another option is to walk h teps forward in time, so that no validation set overlaps with each other
* 
* On using the sckitlearn TimeSeries split function -> might save you work under the right circumstances.
  + Limitiation is that you must use a sckitlearn model on it.
  + Cannot use model from other libraries.
  + Not flexible, must use non-overlapping blocks.
  + Cannot choose the size of first block. All blocks will be of equal size
  + Also, except of coding this, you just ‘hunt for API’

AR, MA, ARMA, ARIMA

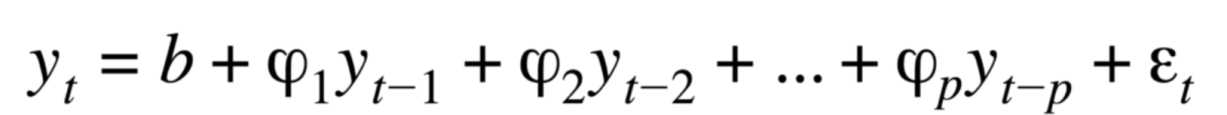
* EWMA Models are very specific, they model trend, and seasonality. ARIMA models do not impose such structure

# Auto-Regressive Models: AR(p)

* Regressive models in which the predictors are past p datapoints of the time series
* Estimation Eqn

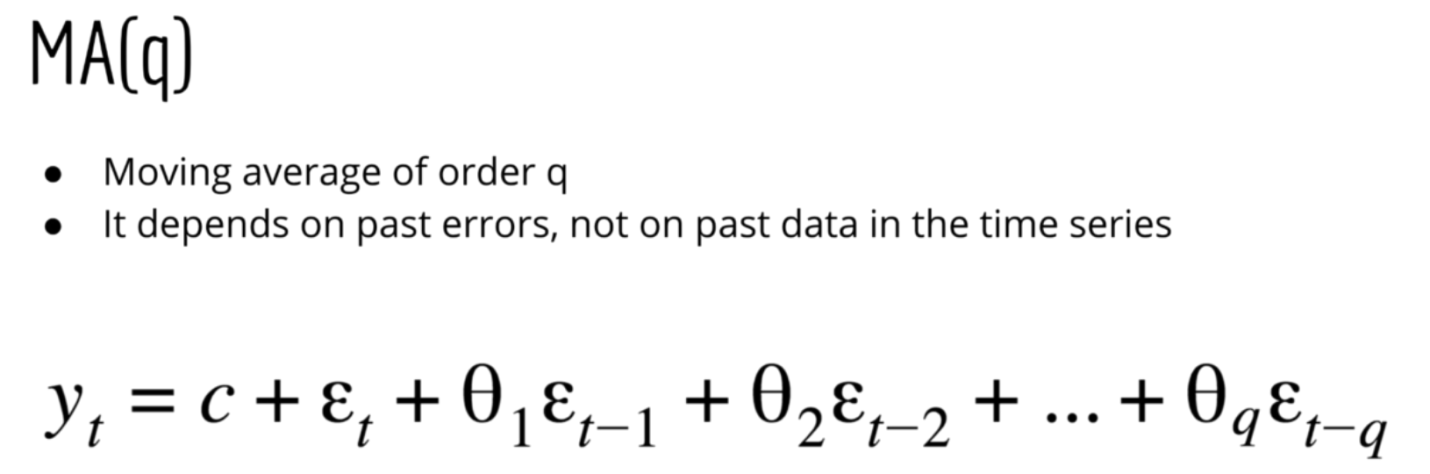


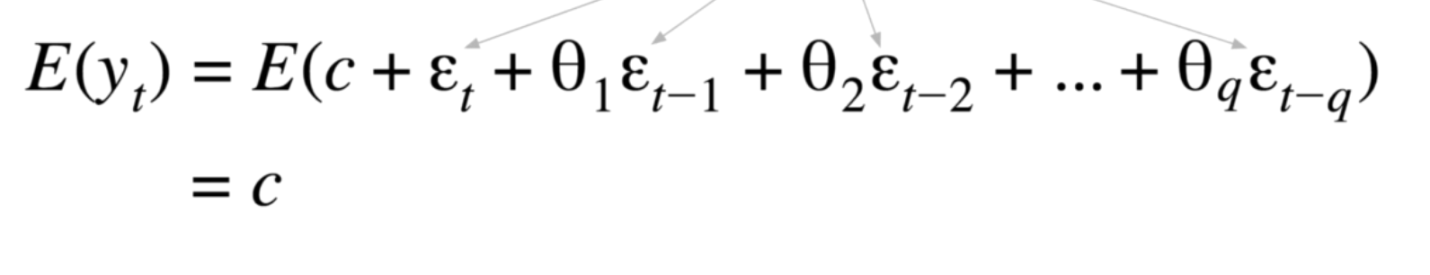
* Model Eqn:



* Both eqns are also represented as an equation without b.
* Linear models are not that powerful. They can only model lines and planes

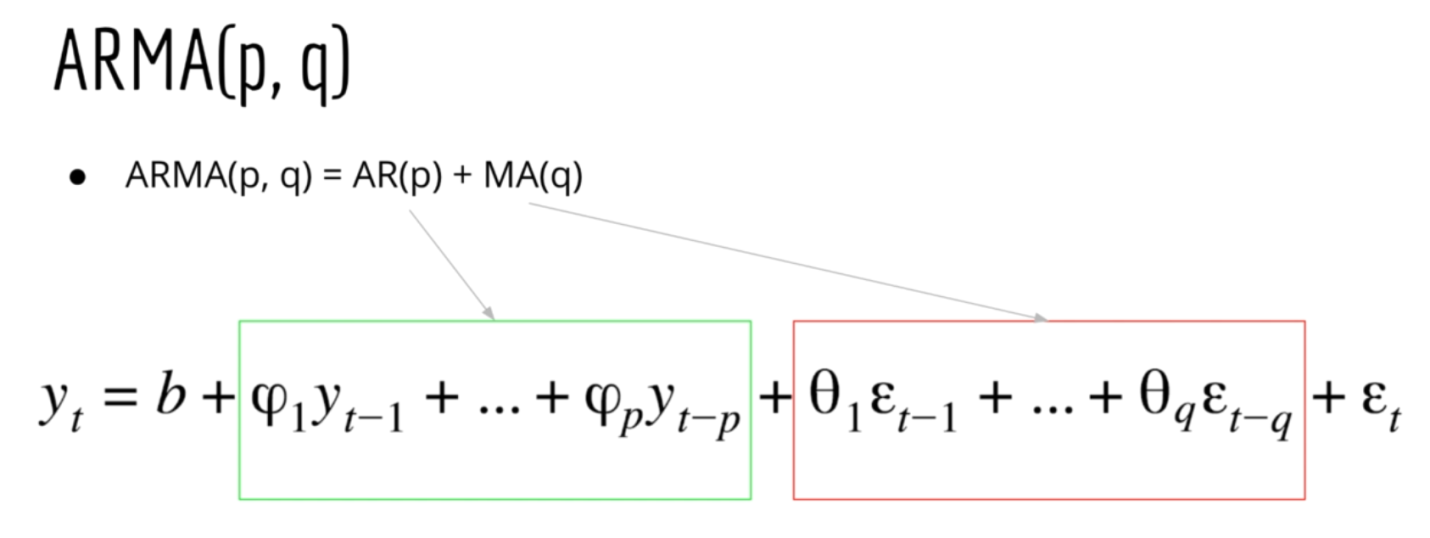
# Moving Average Model



* In MA, there is no input data, only past errors
* 
* Expected value ofyt is 0, ans we can think of errors as fluctuations that go up or down

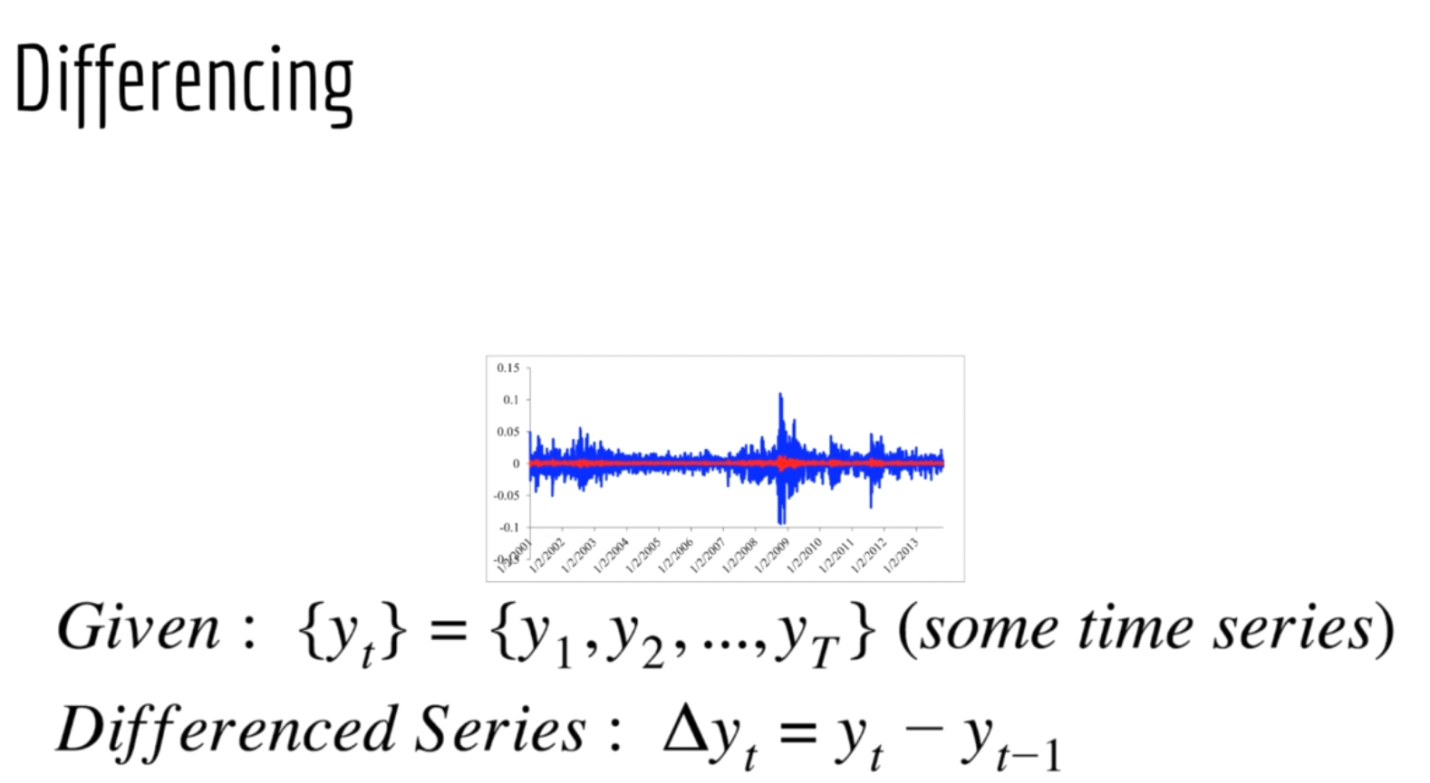
# ARMA (p,q)

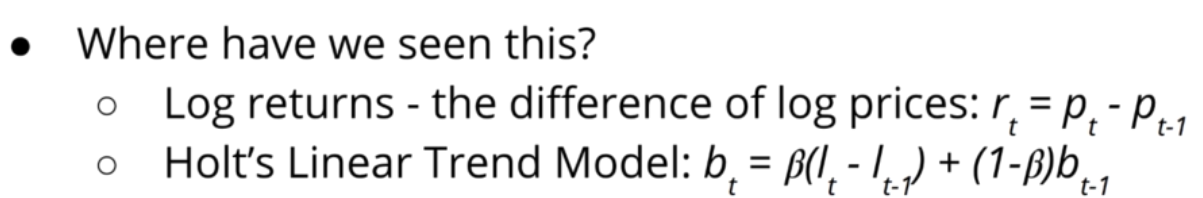
* Just a model having AR(p) and MA(q) both

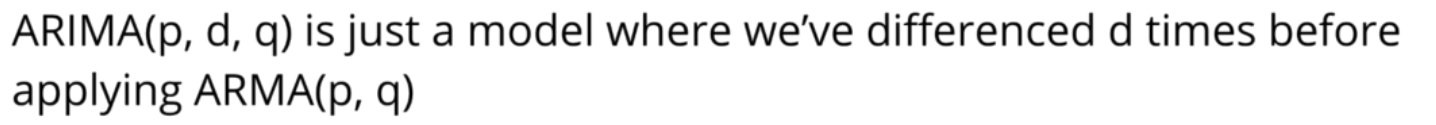


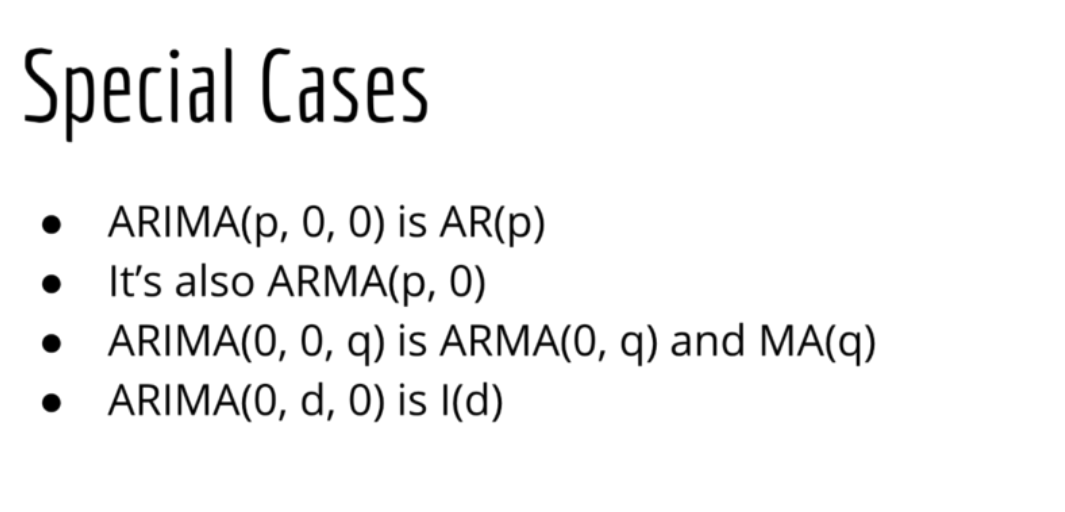
# ARIMA (p,q,q):

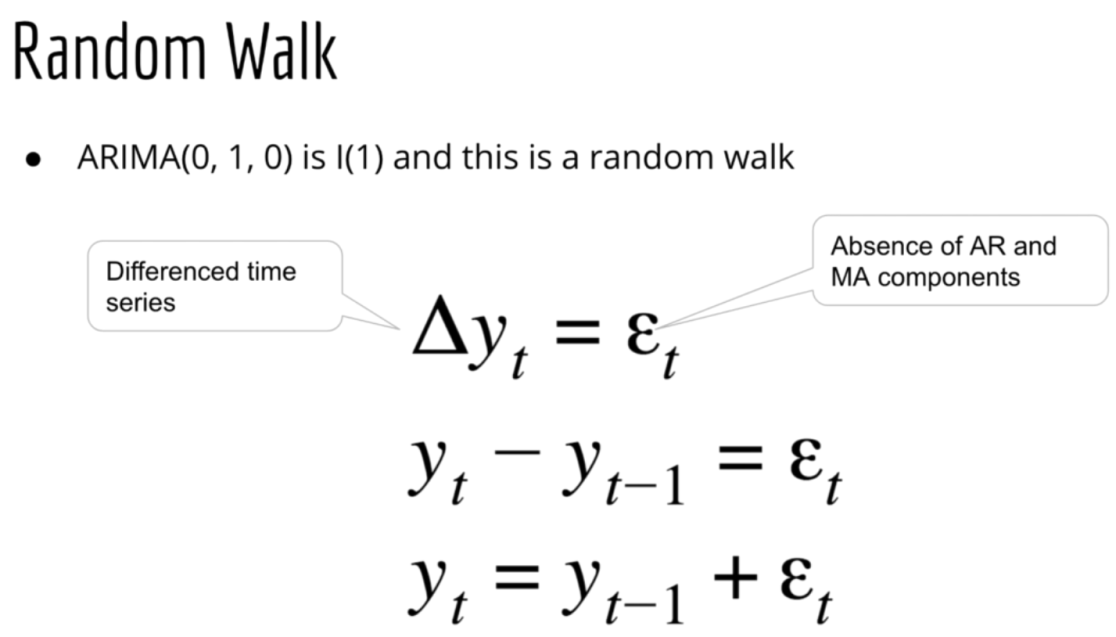
* AutoRegressive Integrated Moving Average

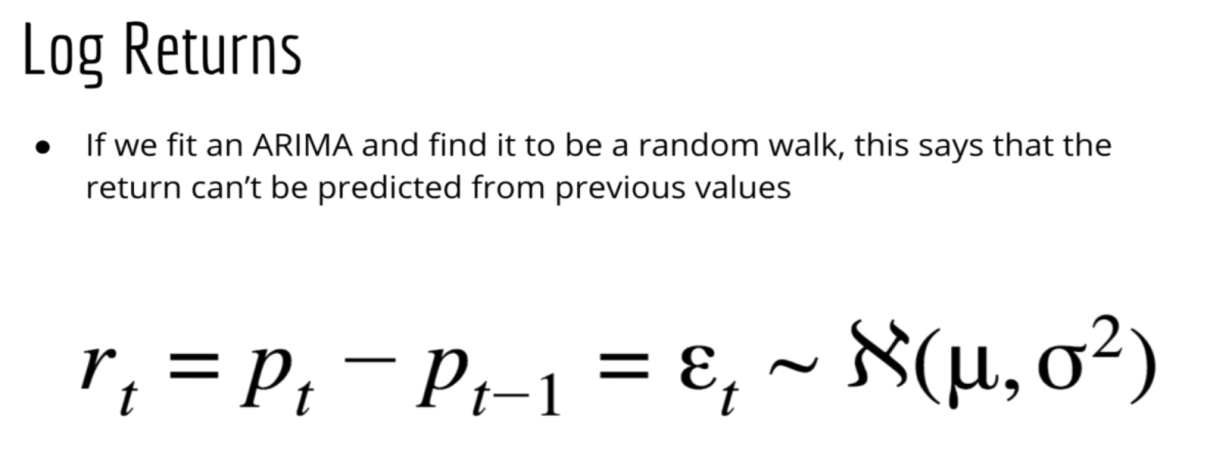




* One way to see differencing is detrending method
* Why we need it?
  + AR,MA,ARMA models need the time series to be close to stationary
* Differencing is a method to make non-stationary time series more stationary
* Usually we do not difference more than twice. d represents this
* 
* 
* The difficult question here is what p,q,d to use
* All AR, MA models are special cases of ARIMA model:

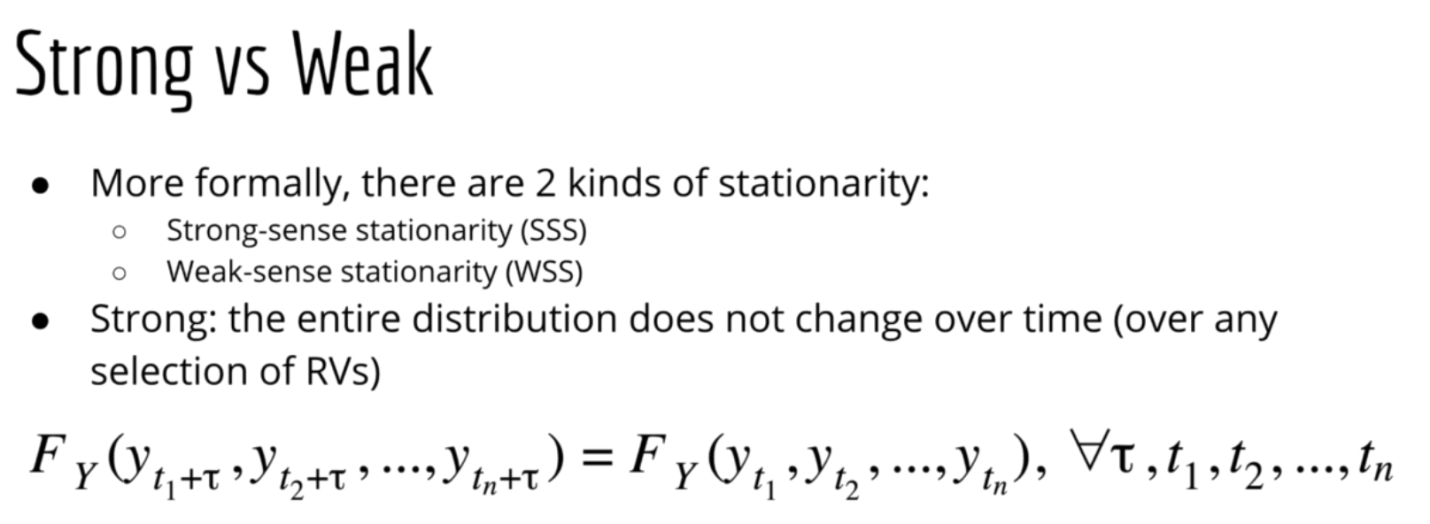




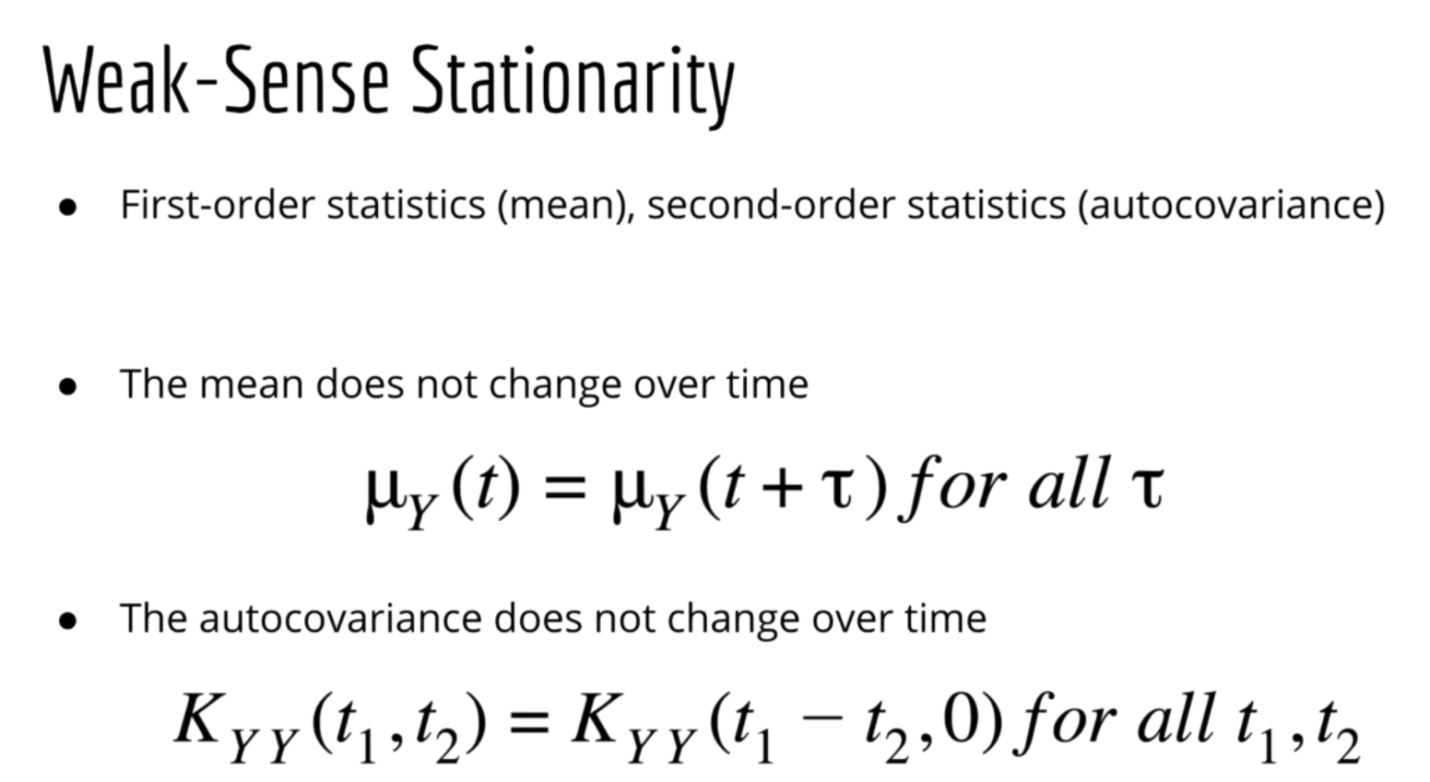


# Stationarity

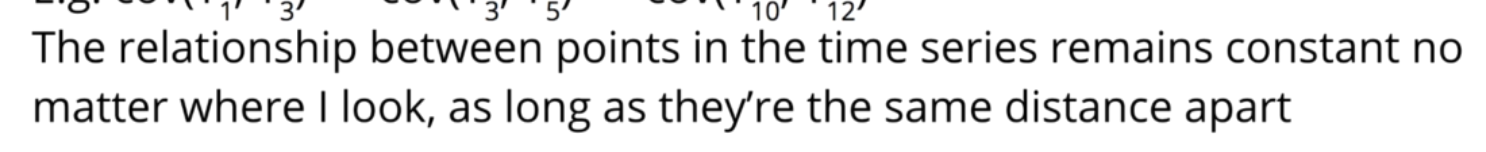
* We define time series as **stationary** if a shift in time doesn’t cause a change in the shape of the distribution.
* The basic of distribution we are talking about is **mean, variance and covariance. For** stationarity absolute values of characteristic equation must be less than 1. It should remain within the unit circle.



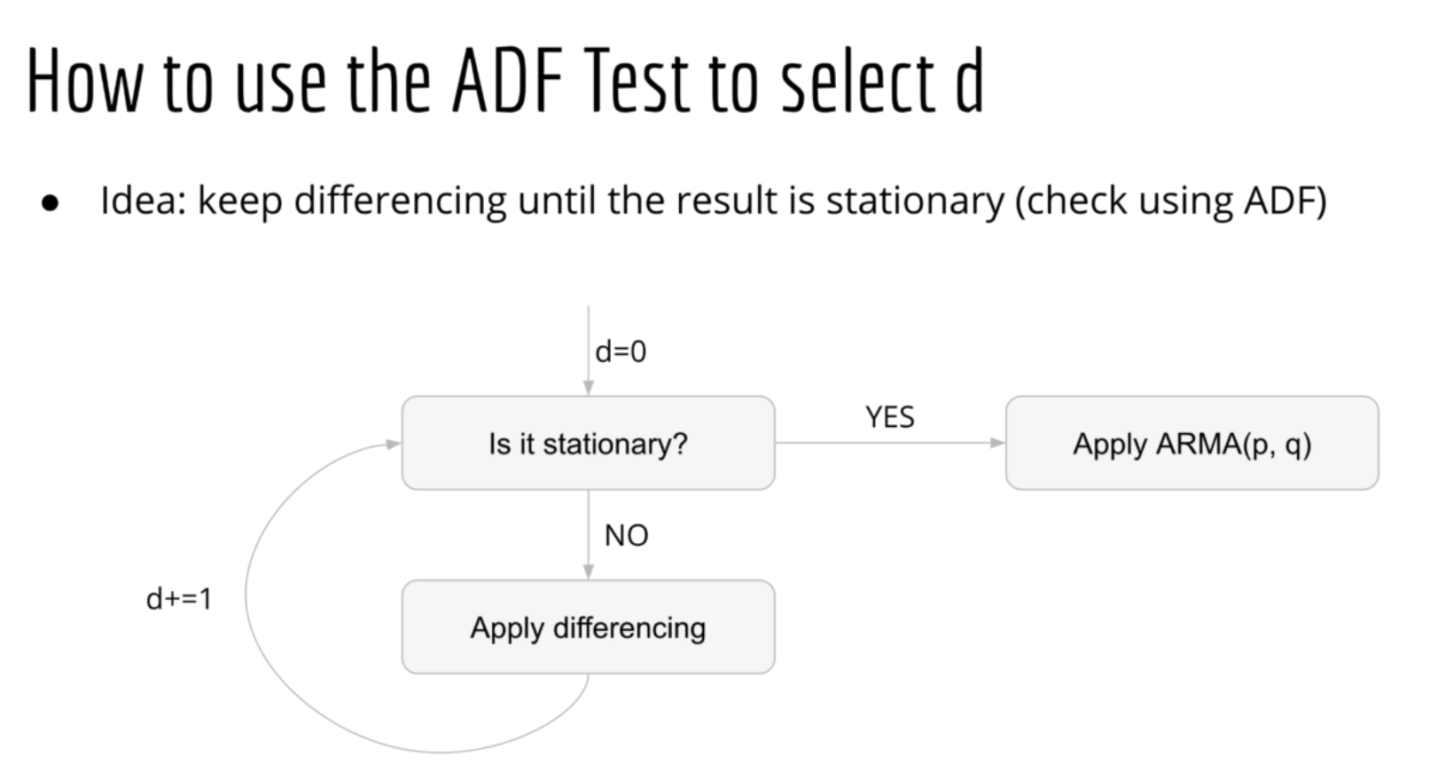
* If you have a time window, no matter where you shift it in the time series, we will have the same distribution of the series
* Strong stationarity condition is used very less in practical applications



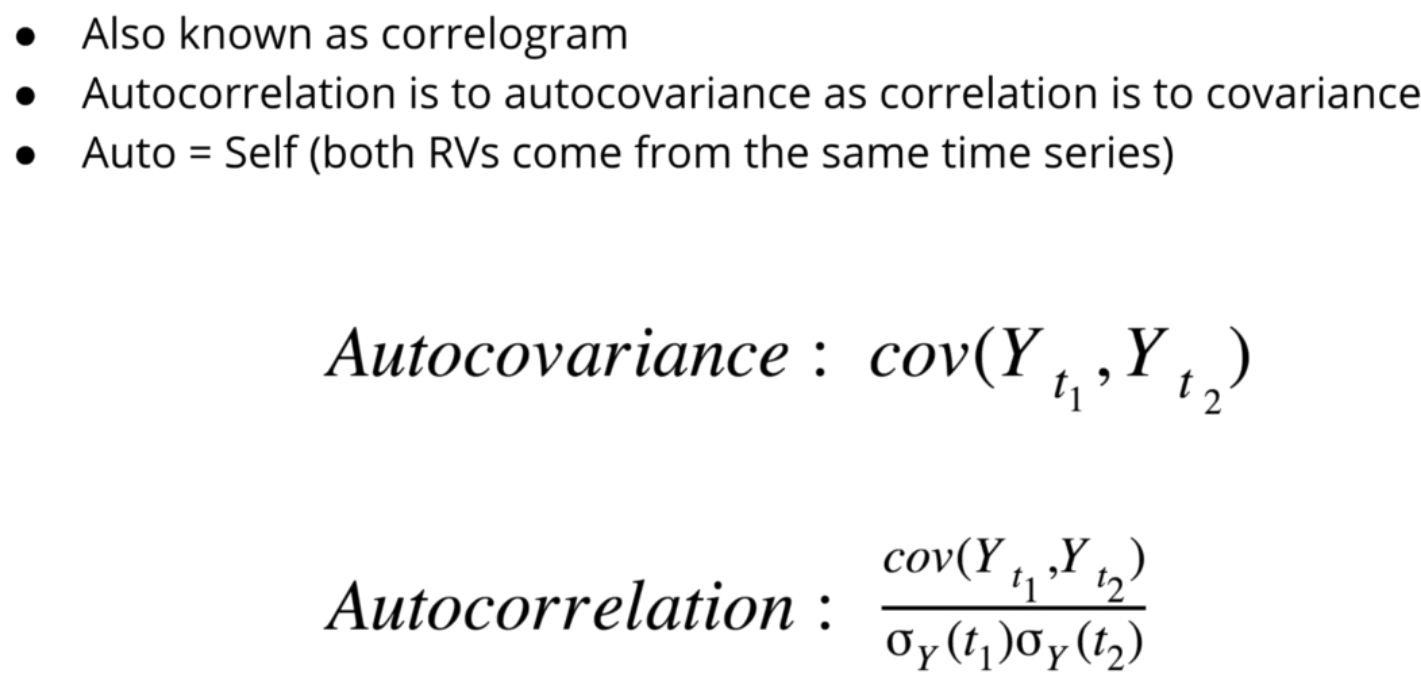
* Autocovarianve is same meaning:



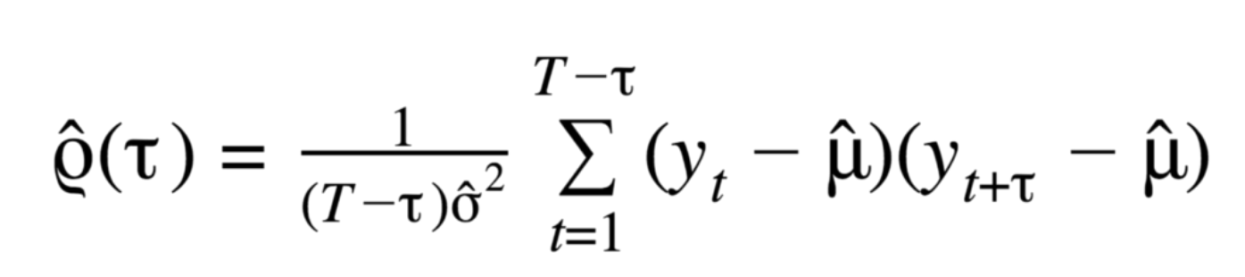
* Also, Variance is constant over time
* Testing for stationarity :
  + ADF test -> Augmented Dickey Fuller Test
    - Null: Time Series is non-stationary
    - Alternative: Tiem Sereis is stationary
  + KPSS test
  + Zivot and Andrews test
  + Variance Ratio Test

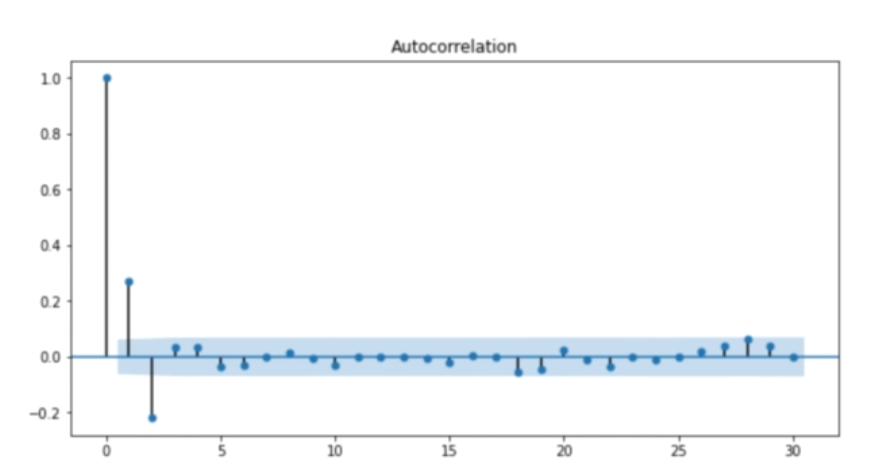


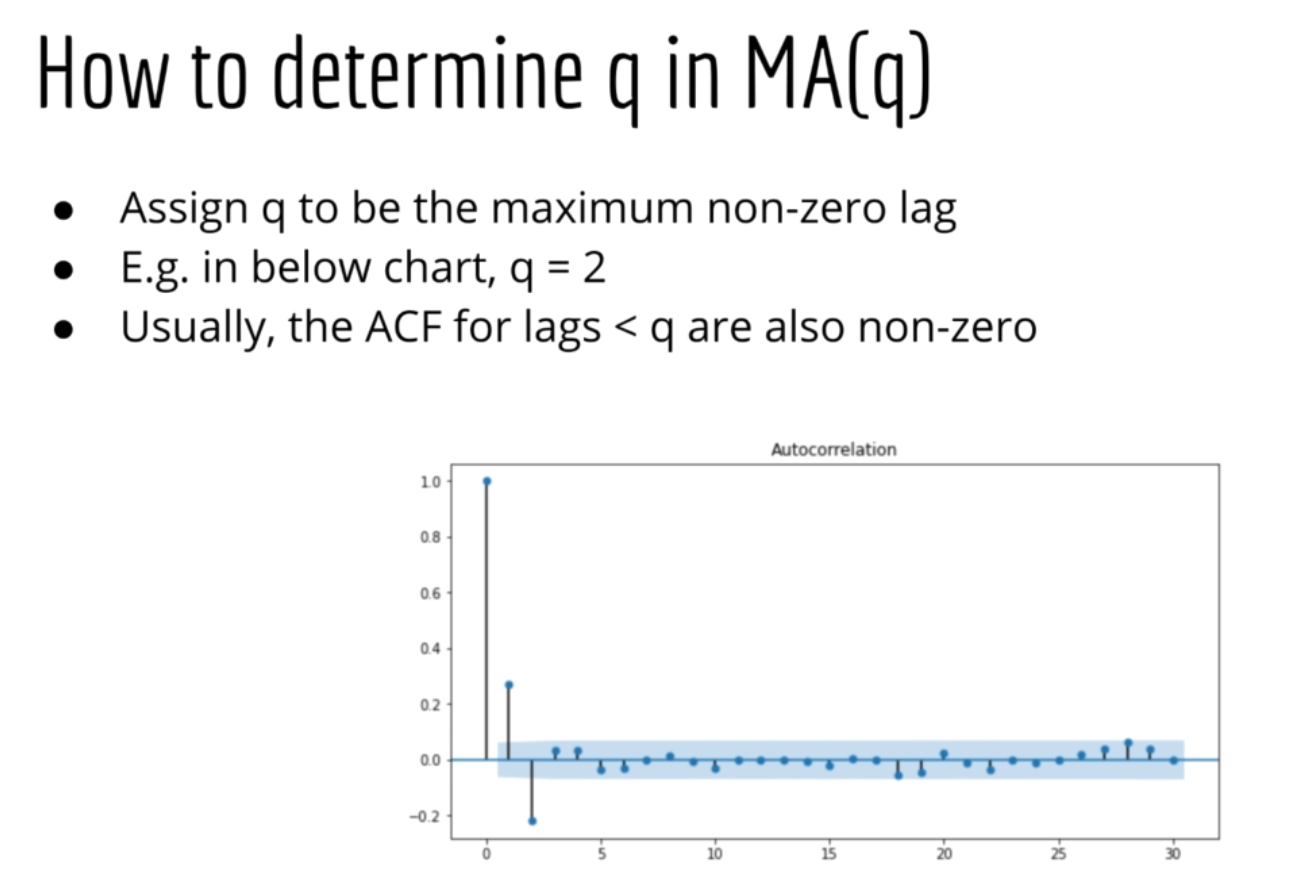
# AutoCorrelation Function



* Autocorrelation is scaled version of covariance
* Now, since we have only on observed variable for each time, we cannot use the traditional formula to calculate the autocorrelation.
* To calculate autocorrelation, we assume that the time series is stationary. And for any time difference, we use the subsequent lagged time series to calculate the autocorrelation

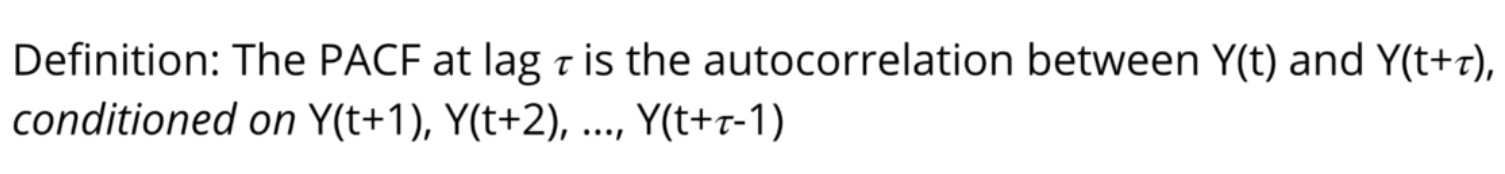


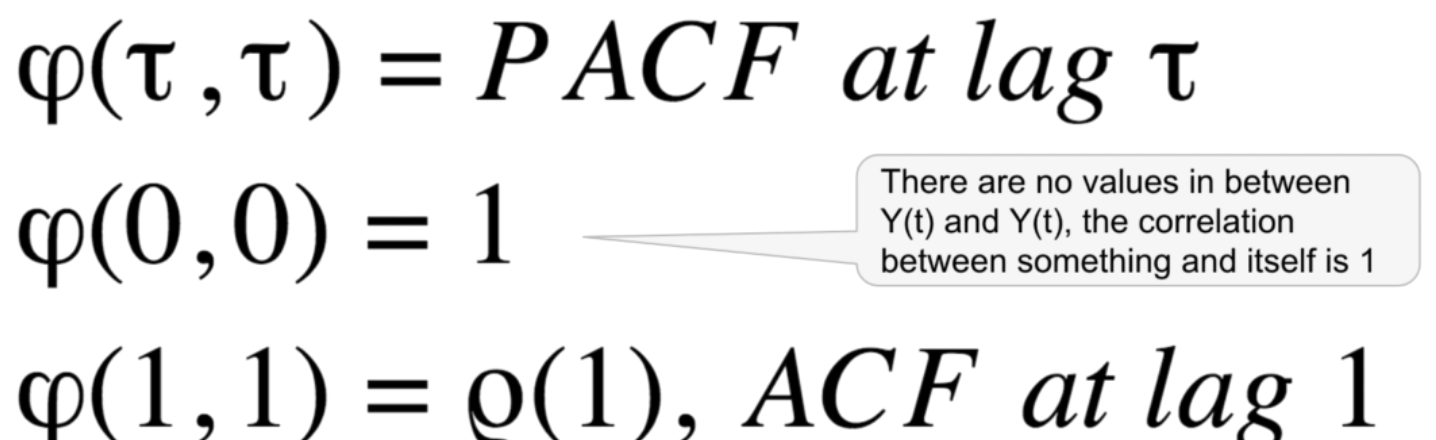
* ACF of lag 0 = variance/variance = 1
* Plot the ACF Function
* 
* Almost all ACF plot functions also give the confidence interval (the blue shaded area). If any point that is outside this confidence interval, we reject that they are equal to 0. Usually, it is 95% confidence interval

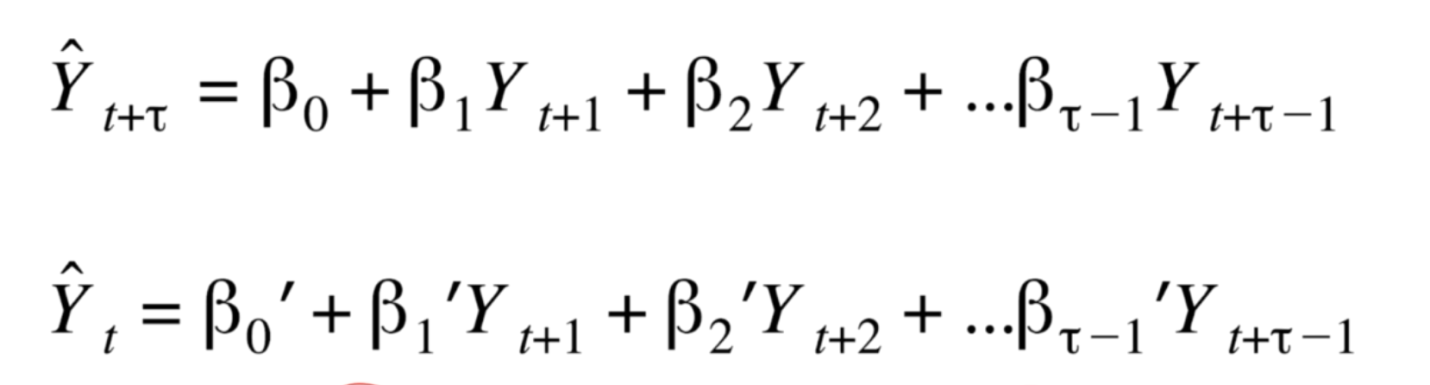


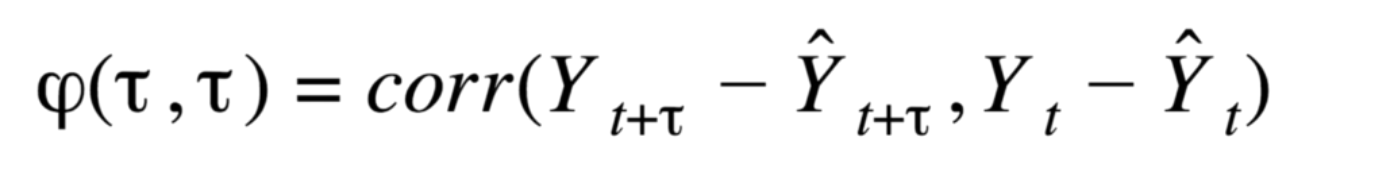
* This process is theoretically justified, but may not give the best result

# Partial Auto-Correlation Function

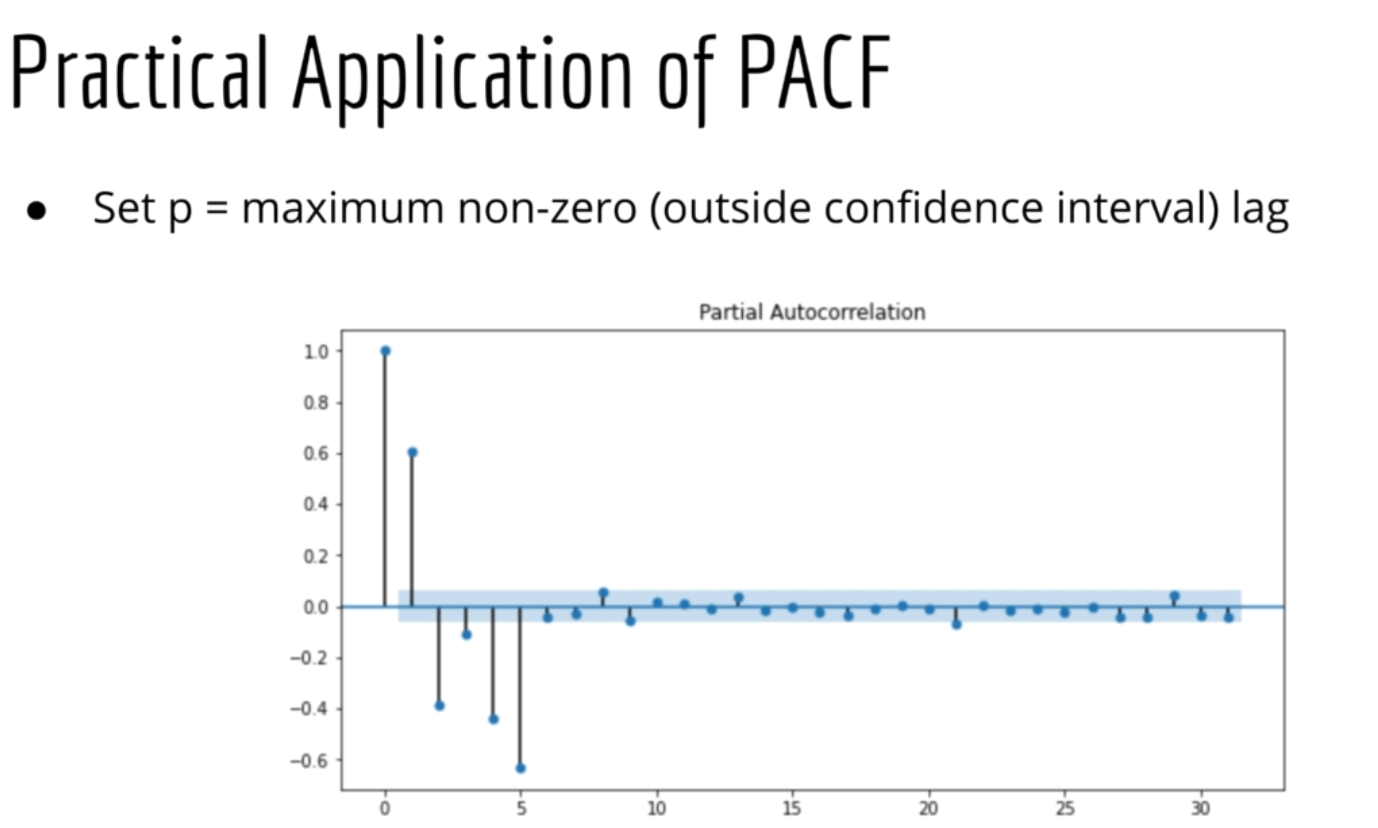
* 
* It is conditional autocorrelation







* This can be thought of as correlation between any noise after subtracting off the parts that can be predicted by the regressions
* It’s the correlation that can’t be explained by the in-between variables
* It is used to choose p for AR(p) model. Works in the same way as ACF for MA(q)

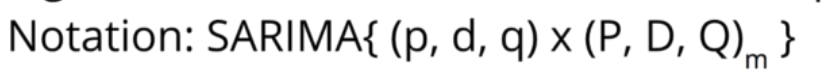


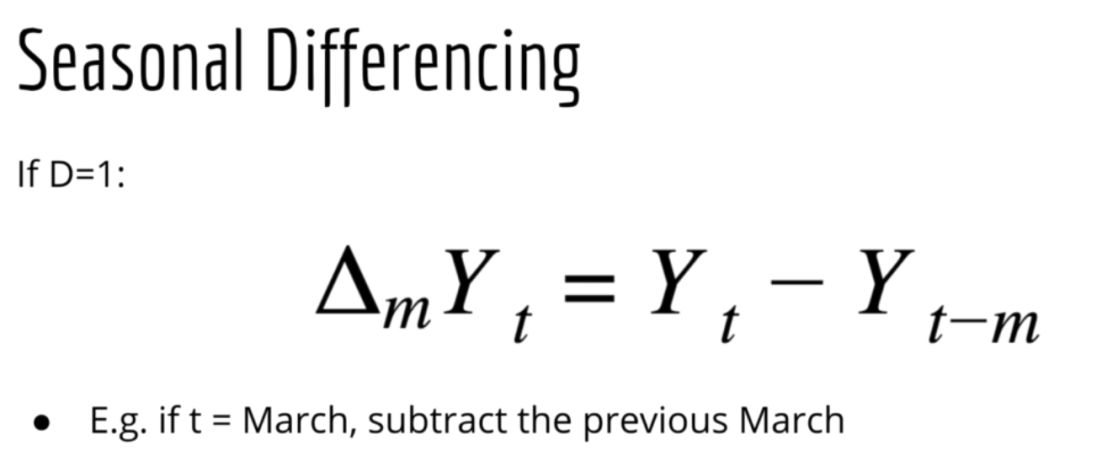
These methods of choosing p,d,q are statistically sound, but do not give the best results. Why do all this work manually, just let computer do the choosing also ->

# AutoArima

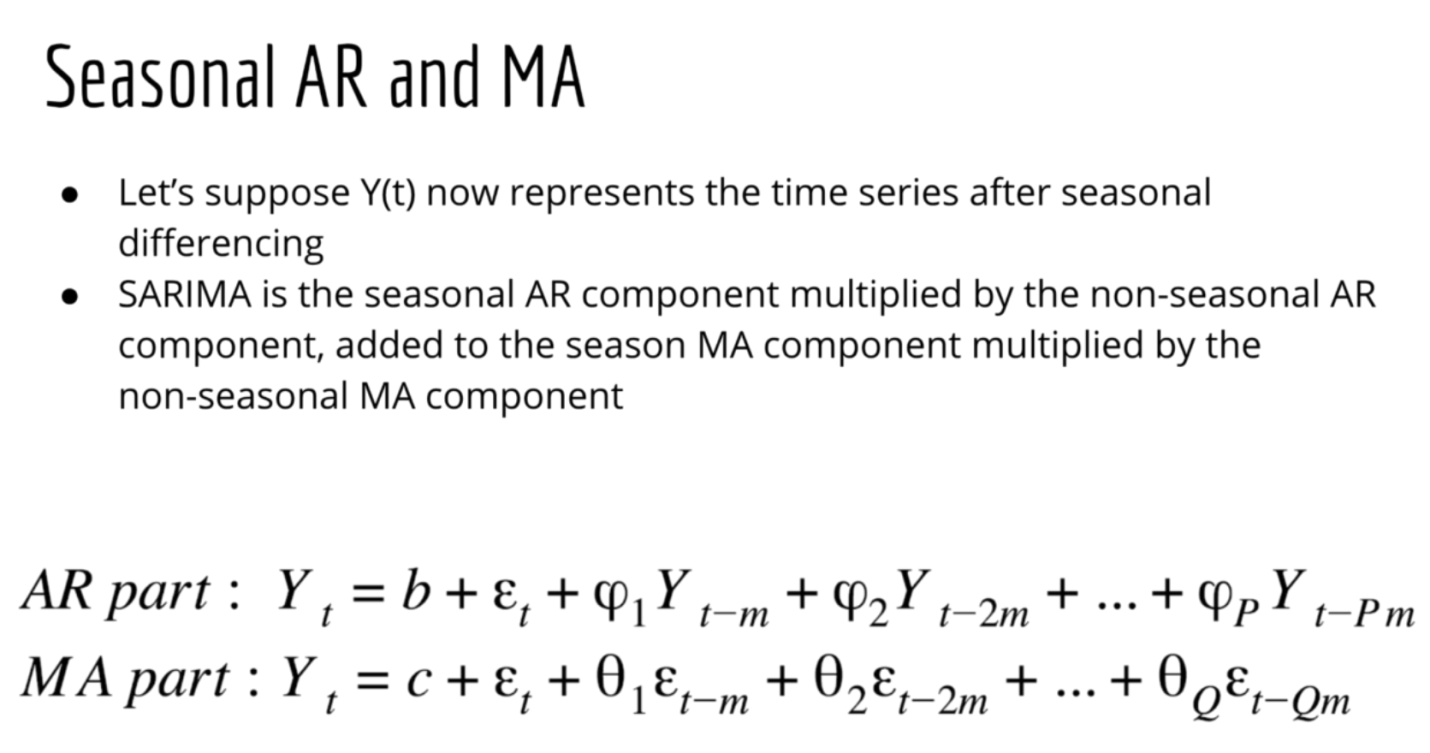
* Python library: pmdarima. Is built on statsmodels
* R package -> forecast. Function-> auto.arima()
* Asdf

# Seasonal ARIMA (SARIMA)

* Stock prices are not affected by seasonality, but other financial data might be.
* Parameters -> p,d,q (same as before), P,D,Q (new)
* , m is the seasonal period
* The cross in the notation is because in the model is multiplicative, and we multiply the non-seasonal part with seasonal part



* The series should be stationary
* AR models can represent seasonality. AR(2) model can represent sin wave. But using SARIMA gives better models (though more complex also)



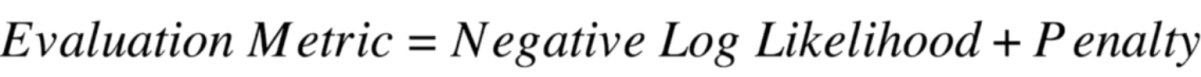
# SARIMAX

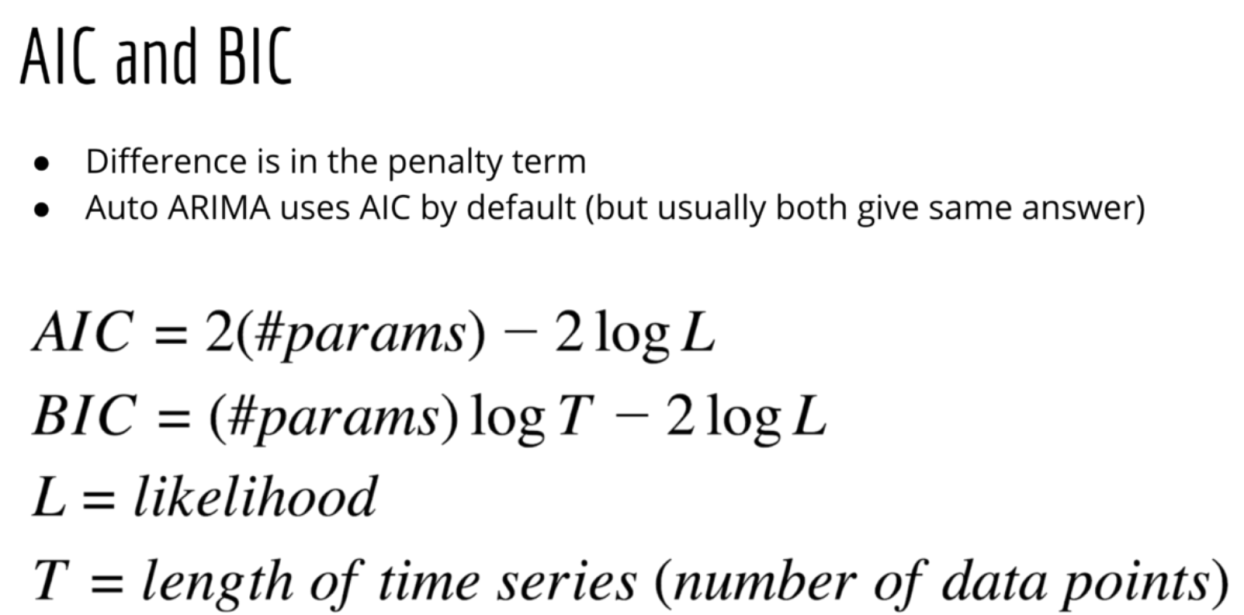
* X represents the exogenous data, e.g. sentiment analysis of company’s tweets, financial news, etc.
* You can pass these into the autoarima model.

# Choosing Hyperparameters:

* One option is randomly trying different parameters
* Grid search -> trying all possible combinations on the grid to find the best parameter. Too computationally expensive
* Autoarima can use gridsearch, but by default stepwise algorithm



* The model is fit by MLE estimators
* 
* Penality so that we do not overfit the model
* The difference between AIC, BIC is that this difference term is computed differently



* Another method to choose the best model is simply based on accuracy of models on validation. But AIC, BIC are used to get models simplest models, without overfitting. AIC, BIC are used more ion datamodelling, while accuracy is a metrics when we built a model for forecasting perspective