# Smoothing-based methods

## Exponential Smoothing

* The exponential smoothing technique is a weighted moving average procedure where the exponential decline of weights happens as the data becomes older.
* More weightage is given to the recent observations and less weightage is given to the past/old observations.
* This method overcomes the shortcomings of the moving average method.
* The smoothing parameters control how fast the weights decay and these parameter values lie between 0 and 1.
* There are three types of exponential smoothing methods:
  + SES
  + DES
  + TES

### Simple Exponential Smoothing (SES)

* The key idea of this method is to keep some memory of the entire time series, but also, we want to give more value to the recent data and less value to the past value. This forms a decaying trend.
* This method is used when there is no trend or seasonality present in the data.
* Let's consider the weight we assign to the recent most value be .

is called the **smoothing parameter**.

So, our forecast at time *t* for the time t+1 is:



* The above formulation is recursive in nature and expands in the following form:



* We can observe from the formulation that the weights are exponentially decaying. Therefore, we give more weightage to the most recent values, and this weightage keeps decreasing for earlier values.

A graph with a line and dots

Description automatically generated

* The recommended starting value of is:

A black text with black text

Description automatically generated

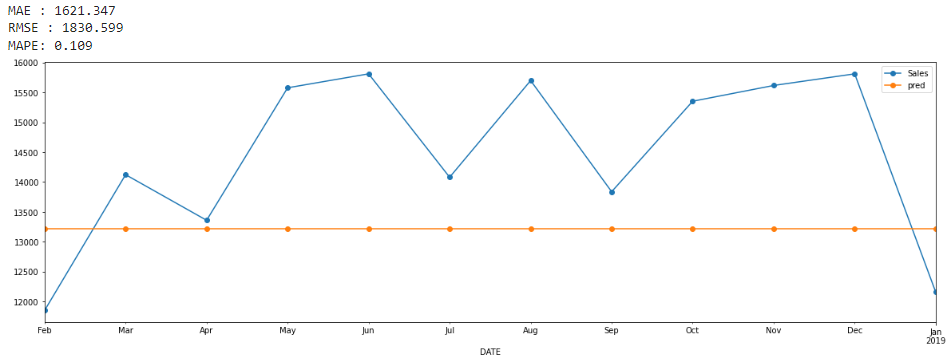
* Let’s fit the SES model on the sales data:

A graph showing a line of orange and blue

Description automatically generated

Unlike the moving averages, it does not have the offset at the beginning and end, because this method is initialized properly.

* The forecast plot of this model on the sales test data is given below:



* The advantage of the above forecast is that the **level** of the forecasted values is right.
* However, the forecast is a completely straight line This is because we don't have the previous actual value available for horizon > 1. So the current forecast is used for all the next values.
* The prediction is a straight line, but the error is 10% which is less than the error of the moving average.
* The higher the value of  **(i.e, nearer to 1)** the forecast becomes more sensitive to the latest observations.
* The lower the value of  **(i.e, nearer to 0)** the forecast will be less sensitive to the latest observations.
* Disadvantage of this model is that it is missing both trend and seasonality
* Now, we have the right levels, if we can predict the trend and seasonality right, we should get a good forecast.

### Double Experimental Smoothing (DES)

* The shortcomings of SES model, it doesn’t capture the trend and only gives one unique value.
* In this method, we incorporate the trend of the entire time series in the SES formulation in order to forecast future values and we will have to provide weights to the trend value also.
* This method is used when there is only a trend present in the data.
* The weights are assigned to the trend value also and this forms an exponentially decaying series. Hence this is called Double Exponential smoothing (**aka Holt's method**). So basically, we are doing exponential smoothing on trend too.
* There are two components present in this method:
  + Level: Captures the short-term average value.
  + Trend: Captures the trend.
* The formulation of DES is as follows:



where



**lt** is called as the **level** of time series at time t.

* Calculation of the trend value:
  + Slope of a curve for Δt=1 is given as

A black text with black lines

Description automatically generated

This slope value is actually equal to the trend of the series.

By plugging this in, we get,

A number and a line

Description automatically generated

where, **(lt -**  **lt-1**) is representing the current slope of the curve and b**t-1**  represents the previous slope.

* The first smoothing parameter corresponds to the level series.
* The second smoothing parameter **β** corresponds to the trend series.
* β  is a parameter that needs to be tuned while training the model.
* Let’s fit the DES model on sales data:

A graph with blue and orange lines

Description automatically generated

* It is a better fit than Simple exponential smoothing.
* The performance of the DES model on the test set:

A graph with blue and orange lines

Description automatically generated

* The error (8.3%) is less as compared to the SES model (10%).
* Disadvantage of this model is that it does not consider the seasonality of time series.

### Triple Exponential smoothing (aka Holt-Winters Method)

* Triple Exponential Smoothing is an extension of Double Exponential Smoothing that explicitly adds support for **seasonality** to the univariate time series. The seasonality value of the entire time series is also incorporated in this model.
* There are two components present in this method:
  + Level: Captures the short-term average value.
  + Trend: Captures the trend.
  + Seasonal: Captures the seasonality.
* Upon incorporating the seasonality, our equation becomes,



where, **m** -> frequency of the seasonality

Therefore, s**t+h-m** is representing the smoothed seasonality.

Also,

A math equations on a white background

Description automatically generated

* The first smoothing parameter corresponds to the level series.
* The second smoothing parameter **β** corresponds to the trend series.
* The third smoothing parameter corresponds to the seasonality series.
* Let’s fit the TES model on sales train data:

A graph showing a number of data

Description automatically generated

It can be clearly observed that this model captures more information than Double Exponential smoothing.

* Let’s look at the forecasts of the TES  model on the sales test data:

A graph with blue and orange lines

Description automatically generated

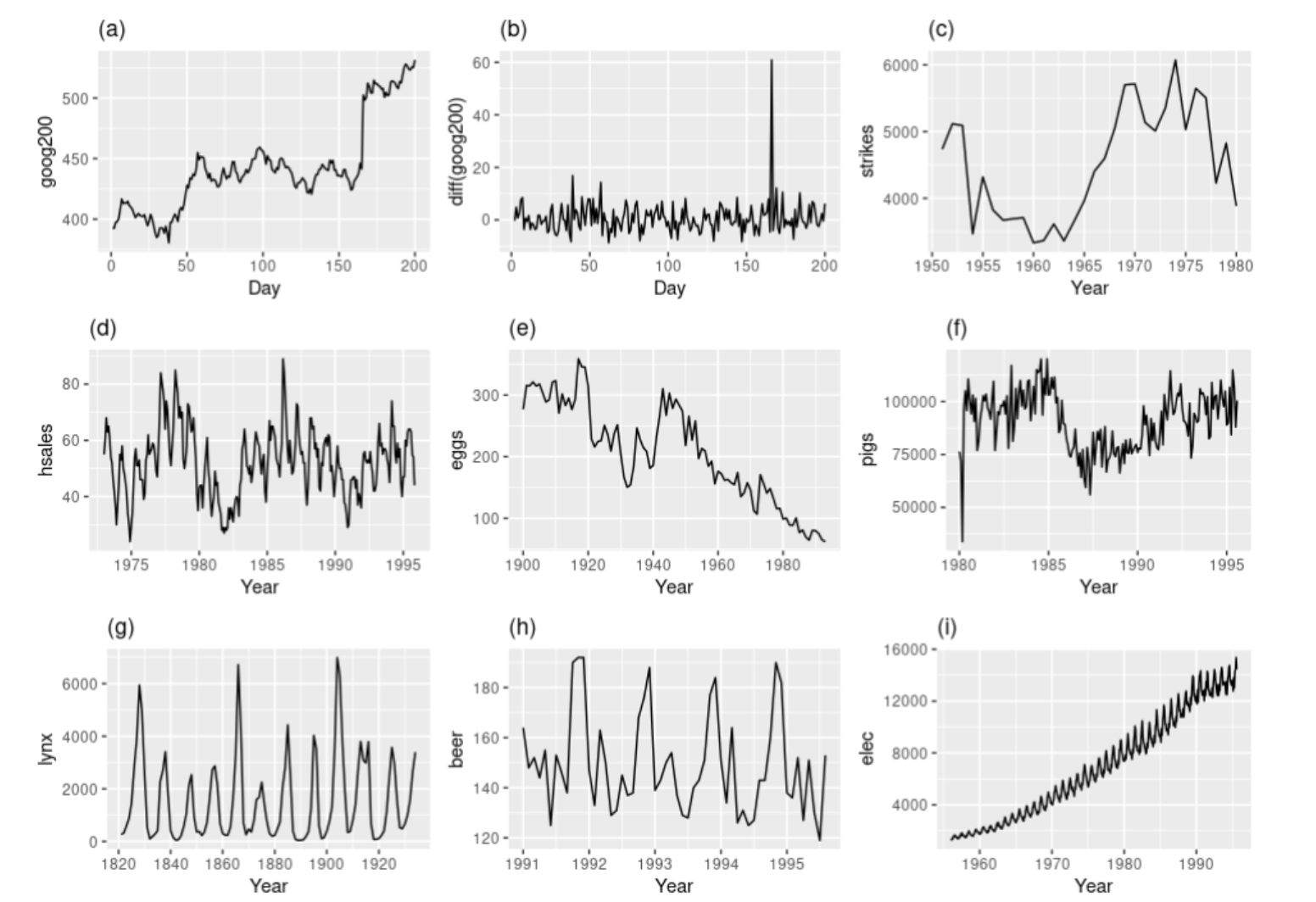
We can clearly see that this model has a better performance as compared to SES and DES.

The MAPE error is only 4% now.

* We can take a mixture of additive and multiplicative models. There is no rule for which model (multiplicative/additive) to use when. We need to try and see which performs better.

# **Stationarity**

* For a time-series model to be stationary, the parameters (like mean, variance, amplitude, and frequency) of the models should **not be dependent on time**.
* Thus, time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times.
* **For example** Heartbeats (mean=0; standard deviation=1) are stationary — it does not matter when you observe it, it should look much the same at any point in time.
* In general, a stationary time series will have no predictable patterns in the long term.
* If a model's parameter varies over time, then there is a complex relationship that needs to be modeled, which all models are not able to account for.
* Many models assume the series to be stationary to be able to give useful results.
* So, either we want to have a stationary time series, or convert to it. Categorizing a time series by just looking at it can be a little subjective.



* Plots a, c, e, and f, are not stationary they either have a trend or mean changing with time.
* Plots d and h are not stationary as the plots have seasonality
* Plot i is not stationary, it has a trend, seasonality and variance is also not stable.
* Plot b is stationary, there is 1 outlier, and can't say anything about mean seems to be just noise.
* Plot g is stationary, predicting this is dicey, so we assume it to be stationary, and try building a model, and seeing if it performs. Looks like a cyclic time series. But these are not at regular intervals. So, even though there is some seasonality, it can't be predicted.
* There are a few methods to check whether the time series is stationary or not.
  + - Can be checked visually using the time series plot.
    - Statistical test (i.e, Dickey-Fuller Test)

## **Dickey-Fuller Test**

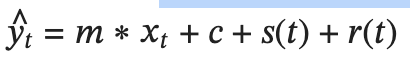
* There is a **statistical** method called the **Dickey-Fuller test**, which is designed for testing for stationarity.
* It fits an auto-regressive model and checks if it worked or not. If it did, then that means it was a stationary time series.
* There is a complicated mechanism to it. We don't need to know how it works. Just need to be aware of this test, as it can be handy.
* For this test,
  + - H0(Null- Hypothesis): The time series is non-stationary.
    - H1(Alternative- Hypothesis): The time series is stationary.
* **How to implement Dickey-Fuller Test?**
  + We can find this as a built-in function under the **statmodels** library as sm.tsa.stattools.adfuller().
* **How do we interpret the result of the Dickey-Fuller test?**
  + This test returns the **p-value**. In order for a time series to be stationary, the **p-value** should be less than 0.05

## **Converting a Non-stationary Time-Series to a Stationary Time-Series**

* As per basic intuition, if we remove trend and seasonality, from our time series, it should become stationary.
* If it's still not stationary, then maybe there's still some seasonality or trend component left in the series.
* Ideally, the trend gets removed in one step. However, seasonality can take multiple steps to be removed.
* These processes are called **Detrending** and **Deseasonalising** respectively.

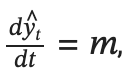
## **De-Trending**

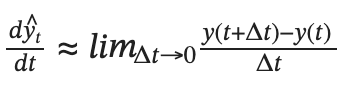
* The trend-seasonality decomposition that a time series can be written as:



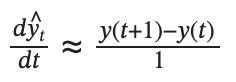
Where; represents the trend component.

* In order to remove this, we can **differentiate** theabove equation with respect to time t. This way, we'll get a stationary time series. This gives us:

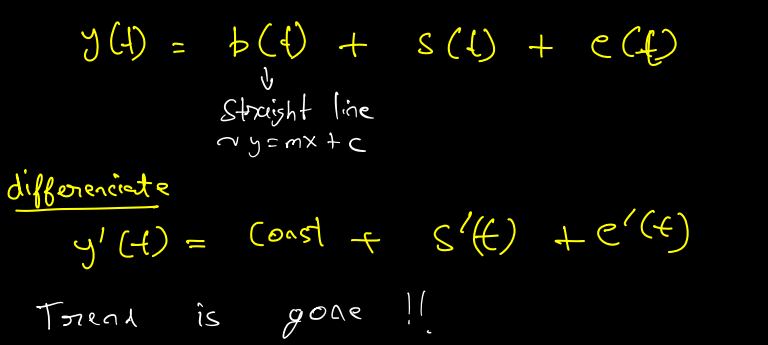




* The minimum value of that we have is 1, since we're talking in terms of time, and our minimum step is 1 month. So the equation becomes:

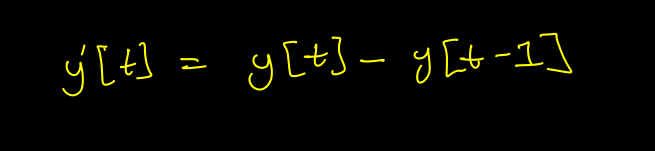


* This process is called **differencing**.
* **NOTE:**
  + Differencing gives a good approximation of differentiating
  + Differentiating gives us a De-trended time series.

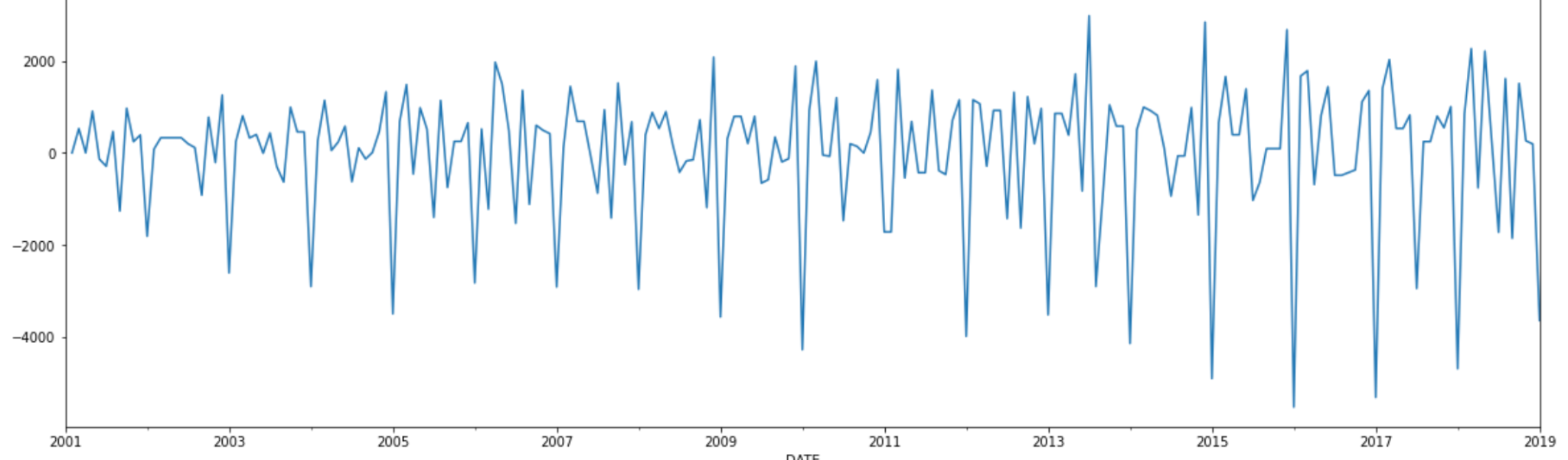


* If the time series has a non-linear trend, then we'll have to differentiate it multiple times, in order to finally achieve a stationary series.
* **NOTE:**
  + If the trend is an exponential function, then we'll not be able to convert it into stationary.
  + This is a very rare case.
  + Some exponential functions can be approximated by polynomials. In that case, we differentiate this polynomial to obtain a stationary time series.
* For applying differencing, a new series is constructed where the value at the current time step is calculated as the difference between the original observation and the observation at the previous time step.

𝑣𝑎𝑙𝑢𝑒(𝑡)=𝑜𝑏𝑠𝑒𝑟𝑣𝑎𝑡𝑖𝑜𝑛(𝑡)−𝑜𝑏𝑠𝑒𝑟𝑣𝑎𝑡𝑖𝑜𝑛(𝑡−1)



* For this, We use the **diff()** method of **pandas**.
* This method calculates the difference between a Dataframe element compared with another element in the Dataframe (default is an element in the previous row, as the default value is 1).

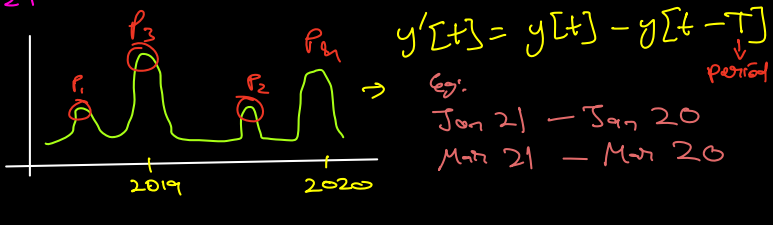


## **De-Seasonalising**

* We can use differencing, but instead of subtracting from the last point, we need to take a difference from the last mth point, where **m** is the period of the seasonality of the series.



* This is called **m-differencing**.
* If there is a seasonal component at the level of one week, then we can remove it on an observation today by subtracting the value from last week.



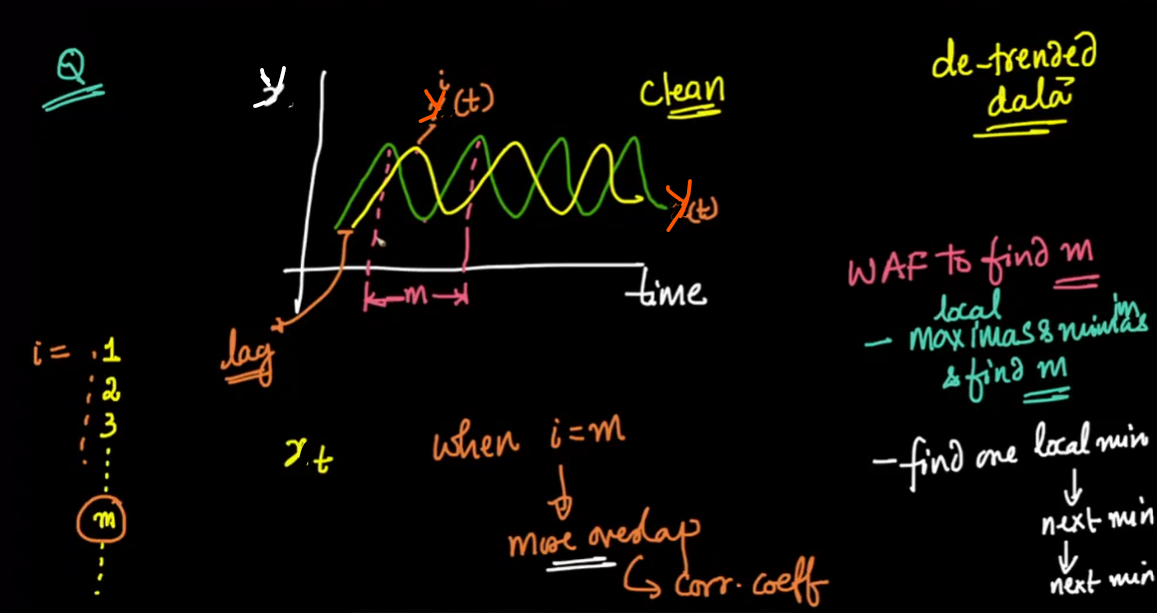
* To find the value of **m** here is tricky.
* Based on the seasonality in the data the m value for the differencing is chosen. For example,
  + - If the data have quarterly seasonality then m will be 4.
    - If the data have monthly seasonality then m will be 12.
* We use the same **diff()** method of **pandas** and first, we remove trend and then we remove seasonality from the series.

# **Autocorrelation**

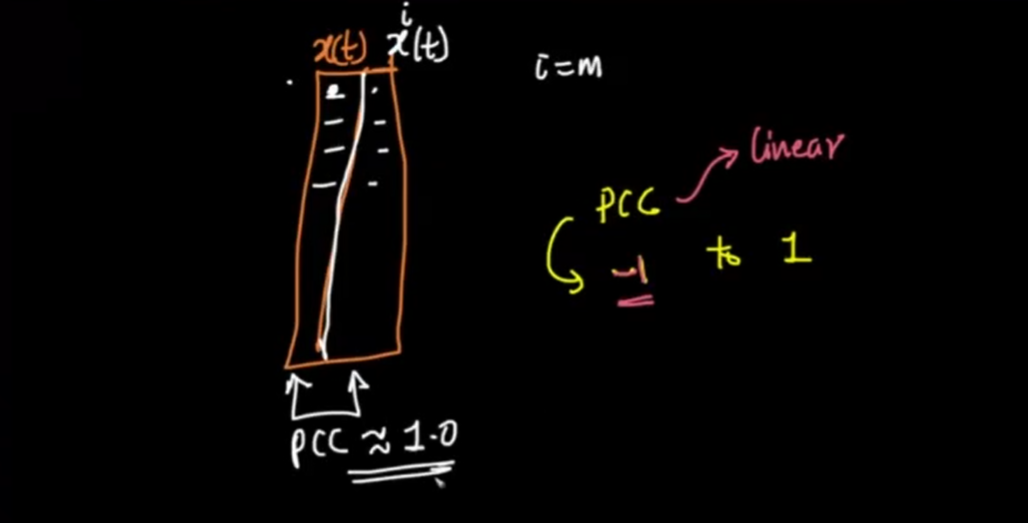
* One approach to finding the value **m** is that we find local maxima and minima in the time series, and try to analyze the intervals at which they are observed.
  + This approach makes sense.
* Another approach would be; given a time series 𝑦(𝑡),
  + What if we consider another time series where we introduce a **lag** of 1, i.e. **shift the series** by 1 unit of time: 𝑦`(𝑡)
  + We can then find the **correlation coefficient** between these two-time series: 𝑦(𝑡) and 𝑦`(𝑡).
  + Similarly, we find the correlation between the original time series 𝑦(𝑡) and a time series lagged by i units: 𝑦i(𝑡), where;

𝑖=1,2,3,... (i represents lag)

* + In doing so, we would find a value of 𝑖, where the lagged time series **roughly overlaps** over the original series.



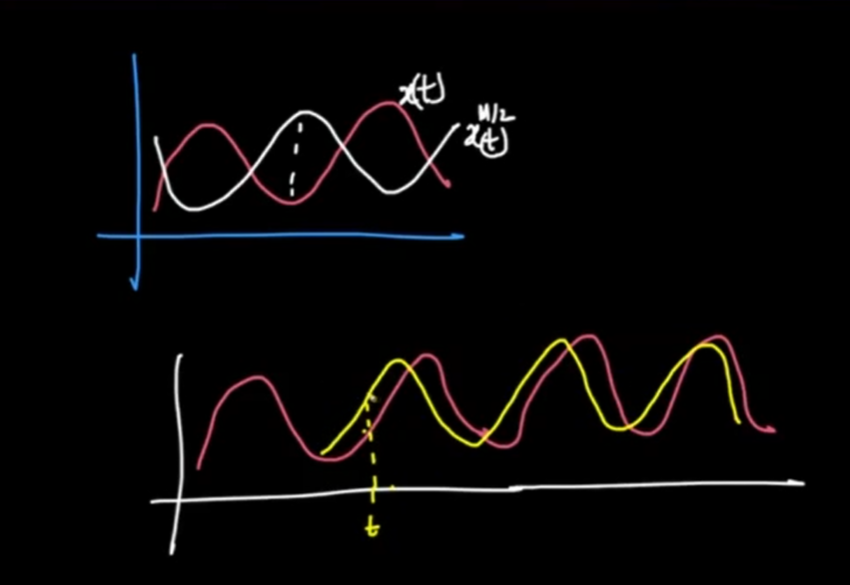
* Then, we create a table containing: 𝑦(𝑡) and 𝑦i(𝑡) time series values
* When ***i=m,*** (the lagged time series roughly overlaps over the original series), the Pearson correlation coefficient would be very close to 1.
* This value of 𝑖 would indicate the optimal value of 𝑚.



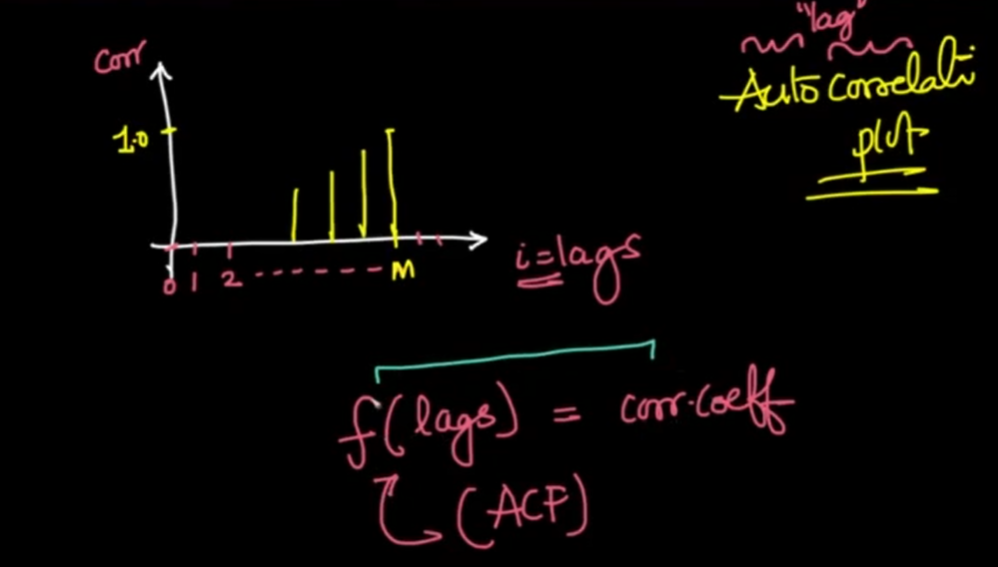
* **What are the advantages of using the concept of correlation for finding the optimal value of m?**
  + Easily interpretable
  + Value ranges from -1 to +1
  + Captures linear relationship

**Q. What will the plot between the correlation coefficient and lag (i) look like?**

* At 𝑖=𝑚, we would get a correlation value very close to 1.
* At 𝑖=𝑚/2, as the value of lagged time series increases, the value of the original series decreases, giving us a **strong correlation.**
* For a value of 𝑖 that is even close to 𝑚, though the correlation value would not be as strong as at 𝑚, it would be relatively strong.



* So, the final plot between lag value (i) and correlation coefficient would look something like this:-



* This plot is called an **Autocorrelation Plot** because we are computing the correlation of the series with itself with various values of lag.
* This can be written as a function also. That is called the **Autocorrelation Function (ACF)**.
* The ACF plot shows both the direct and the indirect impact of the correlation values. It takes the average correlation between the time series original values and the lag values for different lags.
* If the time series is random the lag values will not have much autocorrelation with the original values.