Topic 7
Time Series

ST1511 (AIML) AI & MACHINE LEARNING

Learning Outcomes

- Understand Time Series
  - Understand time series concepts
    - Understand time series use case
    - Perform stationarity test on time series data
  - Perform seasonality breakdown in time series
    - Identify trend, seasonal and random components
    - Distinguish additive vs. multiplicative models
- Apply Time Series Forecasting Models
  - Apply various forecast models on time series
    - Exponential Smoothing Model
    - Autoregressive Integrated Moving Average Model
  - Evaluate model algorithms and interpret model outputs

Understanding time series concepts

Imagine This!

What do these applications have in common:

- SP Power is predicting the electricity consumption of a household for the next three months
- LTA is estimating traffic on roads at certain periods

Are we able to predict future?

Whether we wish to predict the trend in financial markets or electricity consumption, **time** is an important factor that must now be considered in our models.

Why do we want to predict the future

The ability to observe the development of an area of interest (over time) and make predictions upon historical observations creates a competitive advantage within the analytics based on 21st century global business environment.

- You'll gain valuable insights
- You'll learn from the past mistakes
- It will decrease your cost

Time Series Prediction Use Case

- Weather Forecasting: application of modelling to predict conditions of the atmosphere for a given location and time, based on past weather.
- Electricity consumption: Energy providers, i.e. SP group, are using models to predict household energy consumption based on historical energy consumption.
- Sales Forecast
- Financial Forecast
- Many more…

What is time series?

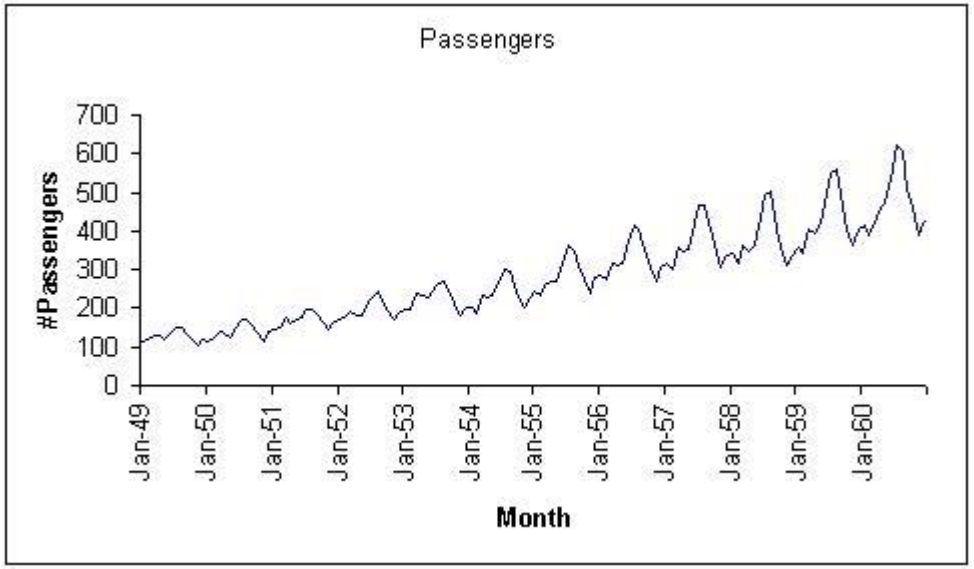
- Time series is numerical data ordered over time.
- The time interval can be annually, quarterly, daily, hourly, etc.
- The sequence of the observation is important.
- Example:

Year:	2005	2006	2007	2008	2009
Sales:	75.3	74.2	78.5	79.7	80.2

Time-Series Plot

A time-series plot is a two-dimensional plot of time series data

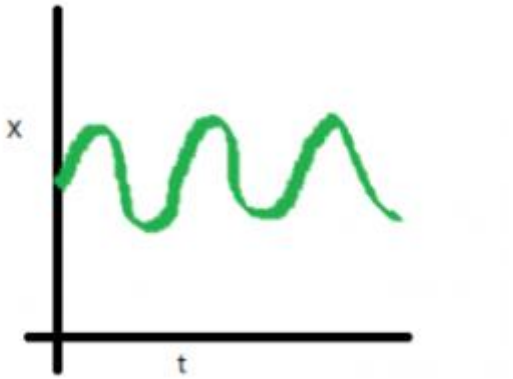
- The vertical axis measures the variable of passenger numbers.
- The horizontal axis corresponds to the time periods.



Stationarity in time series analysis

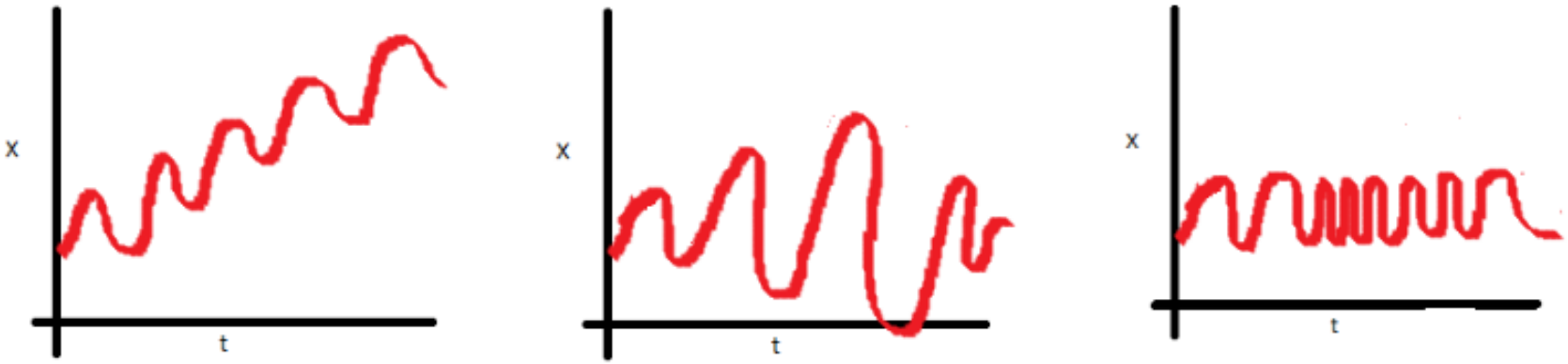
Introduction to Stationarity

- Stationarity is one of the most important concepts you will come across when working with time series data.
- A stationary series is one in which the properties – **mean**, **variance** and **covariance**, do not vary with time.



A stationary time series

Non-stationary time series



- In the first plot, we can clearly see that the mean varies (increases) with time which results in an upward trend. Thus, this is a non-stationary series.
- In the second plot, we certainly do not see a trend in the series, but the variance of the series is a function of time. As mentioned previously, a stationary series must have a constant variance.
- Look at the third plot, the spread becomes closer as the time increases, which implies that the covariance is a function of time

Methods to Check Stationarity

Step 1: Load the air passenger data

Step 2: Convert Month column to datetime format

Step 3: Set Month column as index

Step 4: Drop Month column

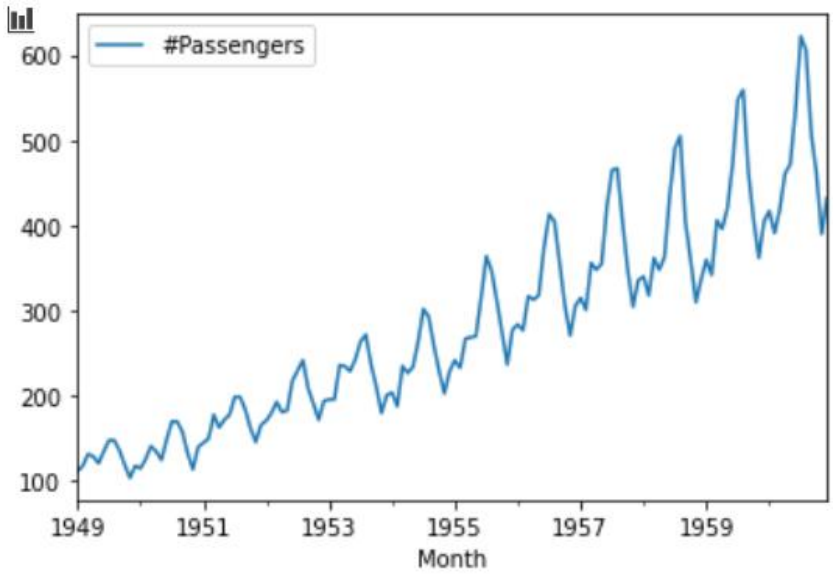
Month	#Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

Methods to Check Stationarity

Visual Test

- Consider the concept we mentioned, we are able to identify in which mean and variance are changing with time, simply by looking at this plot.
- Although it's very clear that we have a trend (varying mean) in this time series, this visual approach might not always give accurate results. It is better to confirm the observations using some statistical tests.

```
#reading the dataset
df = pd.read_csv('data/AirPassengers.csv')
df.index = pd.to_datetime(df.Month , format = '%Y-%m')
df.drop('Month',axis = 1, inplace = True)
df.head()
```



Methods to Check Stationarity

ADF (Augmented Dickey Fuller) Test

- The Dickey Fuller test is one of the most popular statistical tests. It can be used to determine if the series is stationary or not.
  - **Null Hypothesis**: The series is non-stationary
  - **Alternate Hypothesis**: The series is stationary.

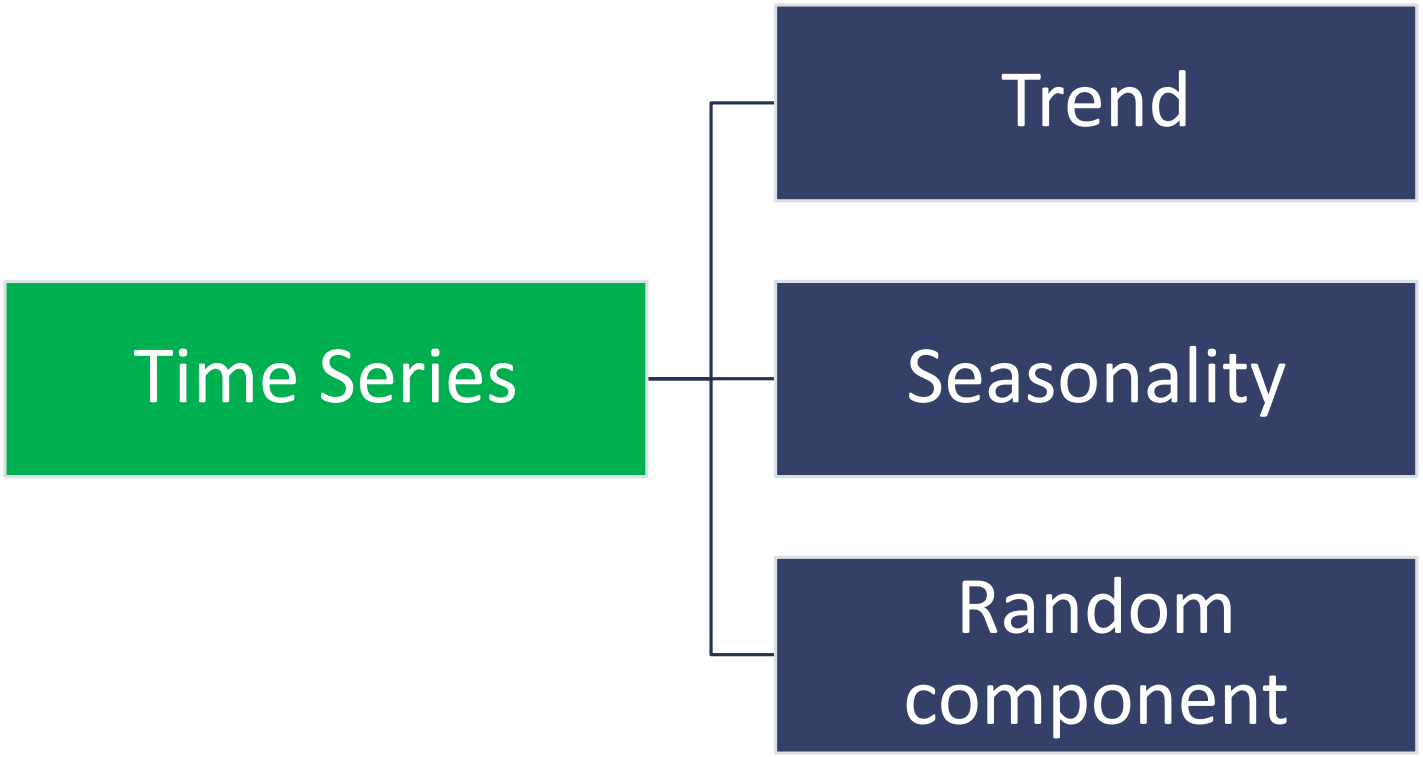
```
from statsmodels.tsa.stattools import adfuller
adftest = adfuller(df)
print('number of observation:', adftest[3], '\np-value:', adftest[1])
```

```
number of observation: 130
p-value: 0.9918802434376411
```

- In this test, the p-value is much larger than 0.05, therefore we cannot reject the null hypothesis. The time series is non-stationary

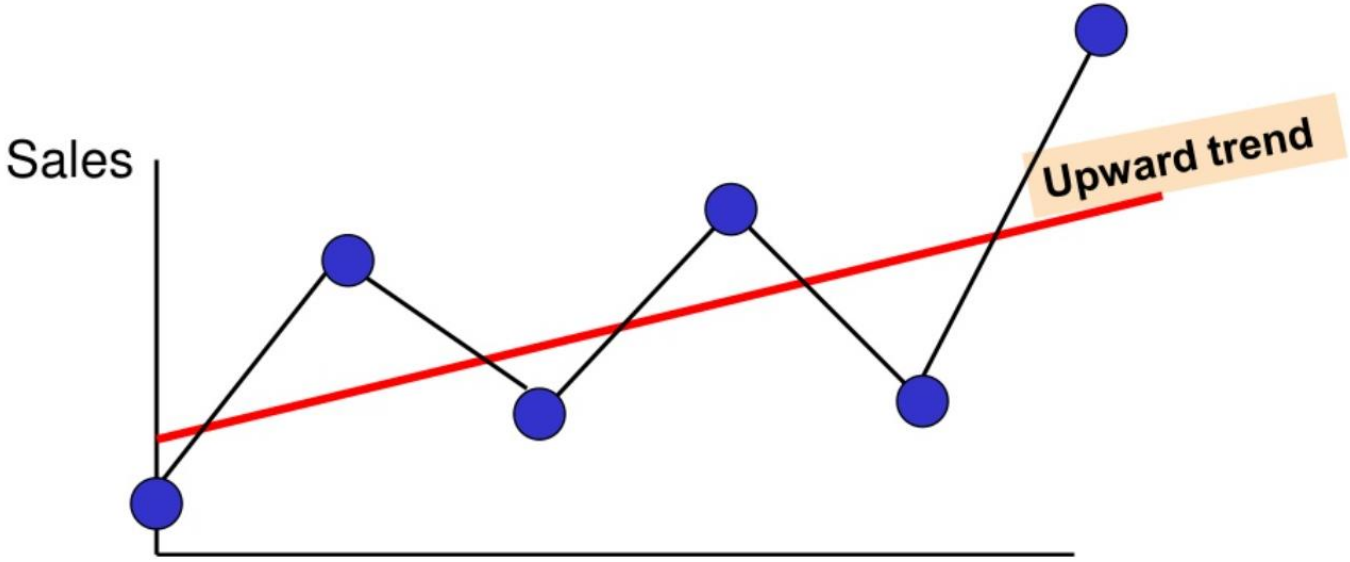
Identify the trend, seasonality and random components in a time series

Time Series Components



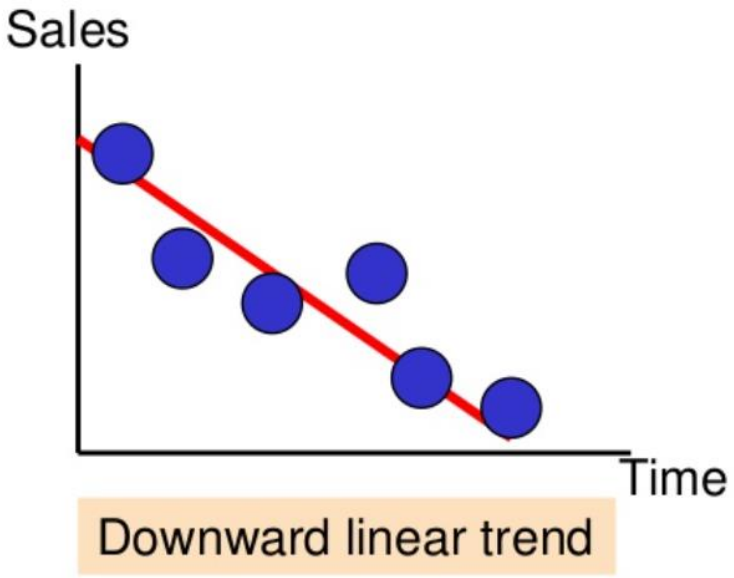
Trend Component

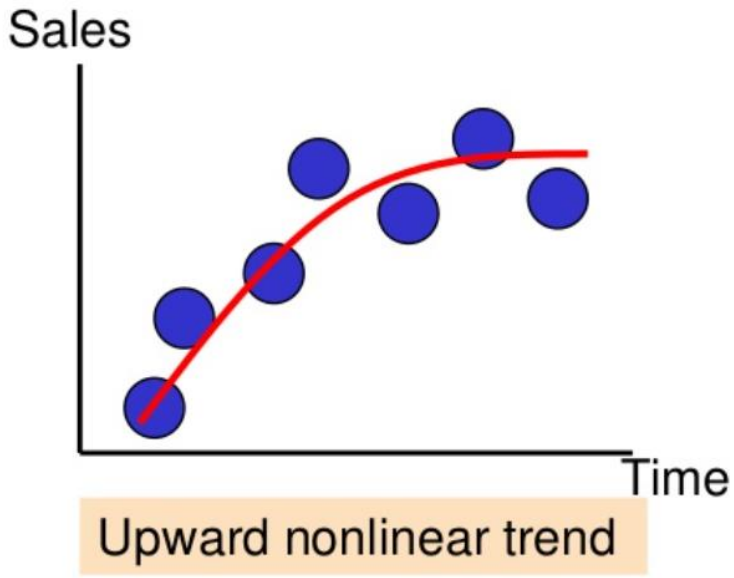
- **Long-run** increase or decrease over time (overall upward or downward movement)
- Data taken over a long period of time



Trend Component

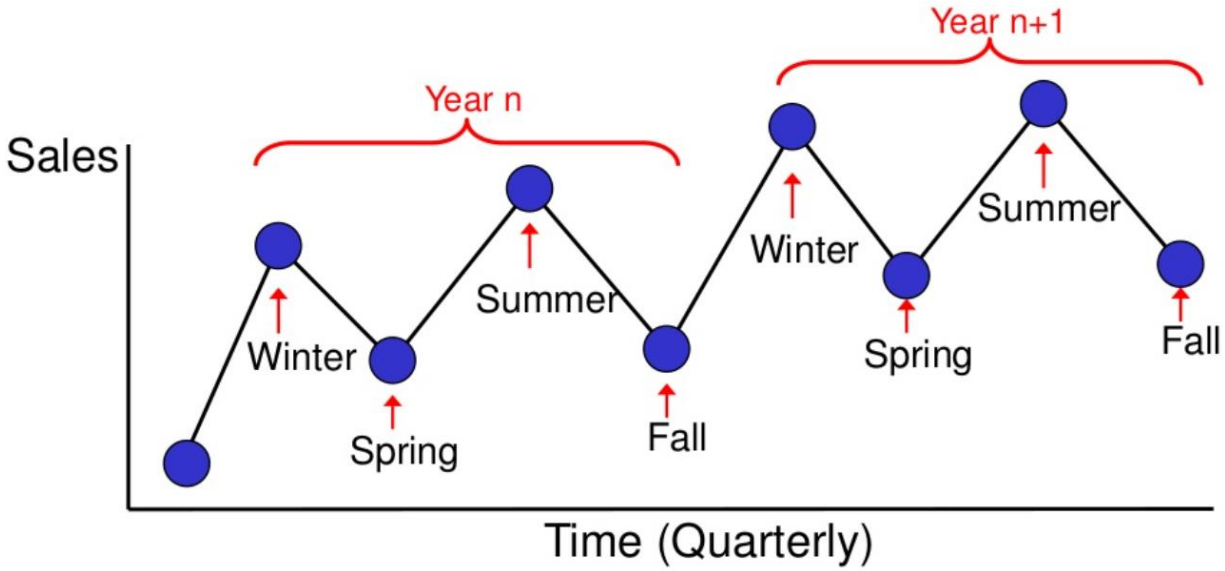
- Trend can be upward or downward
- Trend can be linear or non-linear





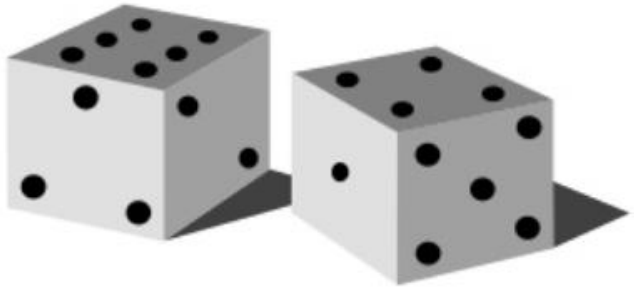
Seasonal Component

- **Short-term** regular wave-like patterns
- Observed within 1 year
- Often monthly or quarterly



Random Component

- Unpredictable, random, “residual” fluctuations
- Due to random variations of
  - Nature
  - Accidents or unusual events
- “Noise” in the time series



Time Series Decomposition

Step 1: Load the air passenger data

Step 2: Convert Month column to datetime format

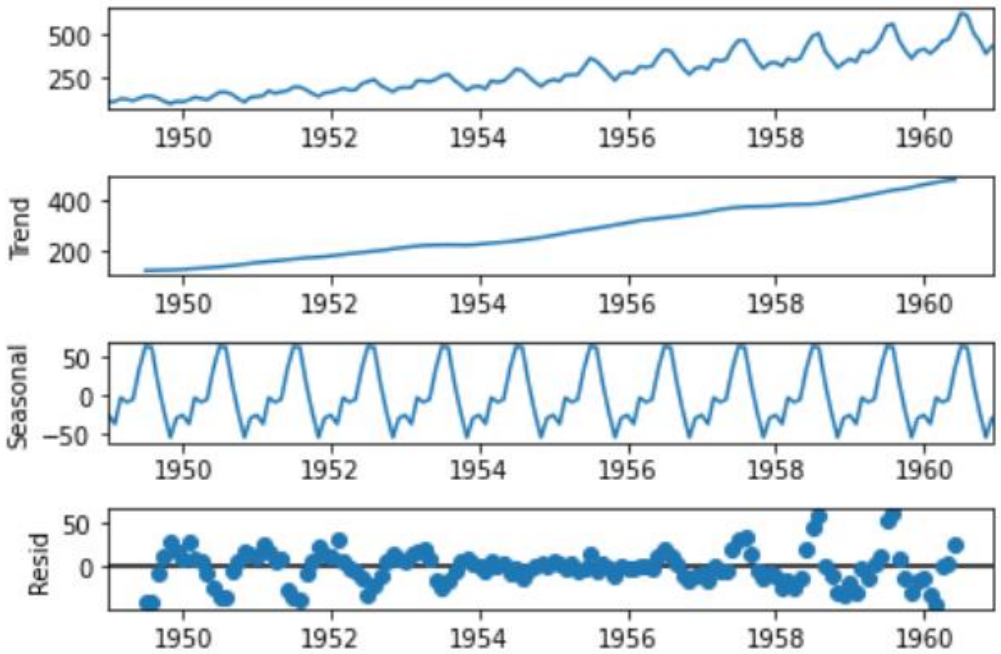
Step 3: Set Month column as index

Step 4: Drop Month column

Month	#Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

Time Series Decomposition

```
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(df, model='additive')
result.plot()
plt.show()
```



Observed data

Distinguish additive vs. multiplicative models

Time Series Component Analysis

- Used primarily for forecasting
- Observed value in time series is the sum or product of the components
- Additive Model:

  Data = Trend + Seasonal + Random

- Multiplicative model:

  Data = Trend * Seasonal * Random

Additive or Multiplicative?

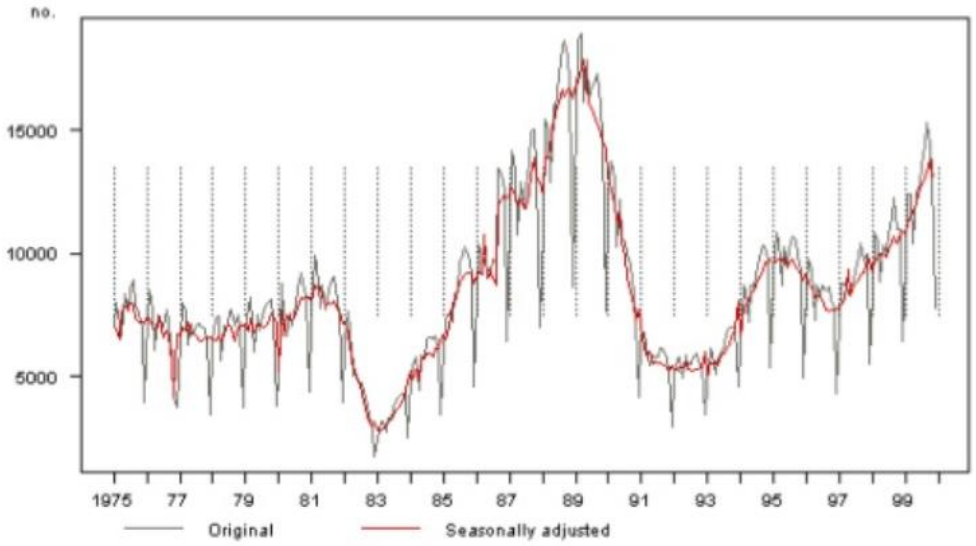
- The time series techniques we will discuss are approached from an additive standpoint.
- Fortunately, multiplicative models are equally easy to fit to data as additive models! The trick to fit a multiplicative model is to take logarithms on both sides of the model.

$$\text{Log(Data)} = \text{Log(Trend)} + \text{Log(Seasonal)} + \text{Log(Random)}$$

- After taking logarithms (either natural log or to base 10), the three components of the time series again act additively.

Multiplicative Example

- In many time series involving big quantities (e.g. money, wheat production, …), the absolute difference in values is of less interest rather than the percentage changes. This is a scenario where multiplicative approach will be used.
- In multiplicative time series, the magnitude of both the seasonal and random components increase as the level of trend increases.



Time Series Forecasting Model

SIMPLE MOVING AVERAGE

Simple Moving Average

- Calculate moving averages to get an overall impression of the pattern of movement over time.
- This smooths out the irregular / random component.

Moving Average: averages of a designated number of consecutive time series values

Simple Moving Average

- **Example**: Three-year moving average
  - First average:

$$X_3^* = \frac{X_1 + X_2 + X_3}{3}$$

  - Second average:

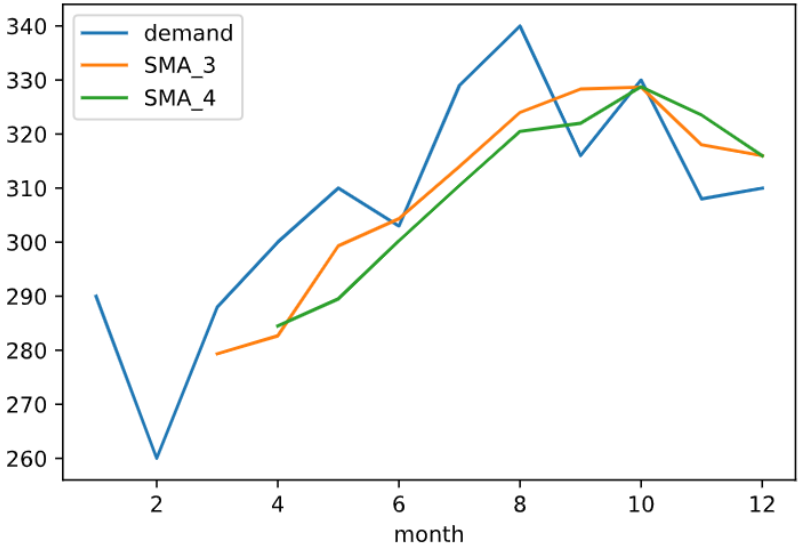
$$X_4^* = \frac{X_2 + X_3 + X_4}{3}$$

Simple Moving Average

```
product = {'month' : [1,2,3,4,5,6,7,8,9,10,11,12],
        'demand':[290,260,288,300,310,303,329,340,316,330,308,310]}
df = pd.DataFrame(product)
df.set_index('month', inplace=True)

df['SMA_3'] = df['demand'].rolling(window=3).mean()
df['SMA_4'] = df['demand'].rolling(window=4).mean()
```

month	demand
1	290
2	260
3	288
4	300
5	310
6	303
7	329

month	demand	SMA_3	SMA_4
1	290	NaN	NaN
2	260	NaN	NaN
3	288	279.333333	NaN
4	300	282.666667	284.50
5	310	299.333333	289.50
6	303	304.333333	300.25
7	329	314.000000	310.50



Time Series Forecasting Model

EXPONENTIAL SMOOTHING

Exponential Smoothing

- A weighted moving average
  - Weights decline exponentially
  - Most recent observation weighted the most

- Used for smoothing and short term forecasting (often a few periods into the future).

Exponential Smoothing

- The weight (smoothing coefficient) is **α**
  - Subjectively chosen
  - Range from 0 to 1
  - Smaller α gives more smoothing

- The weight is:
  - Close to 0 for smoothing out unwanted irregular components
  - Close to 1 for forecasting

Exponential Smoothing

- Exponential Smoothing Forecast Model

$$\hat{X}_1 = X_1$$

$$\hat{X}_{t+1} = \alpha X_t + \alpha(1-\alpha)X_{t-1} + \alpha(1-\alpha)^2 X_{t-2} + \cdots,$$

$\hat{X}_{t+1} = forecast\ value\ for\ period\ t+1$
$X_t = observed\ value\ in\ period\ t$
$\alpha =$ weight (smoothing coefficient) between 0 to 1

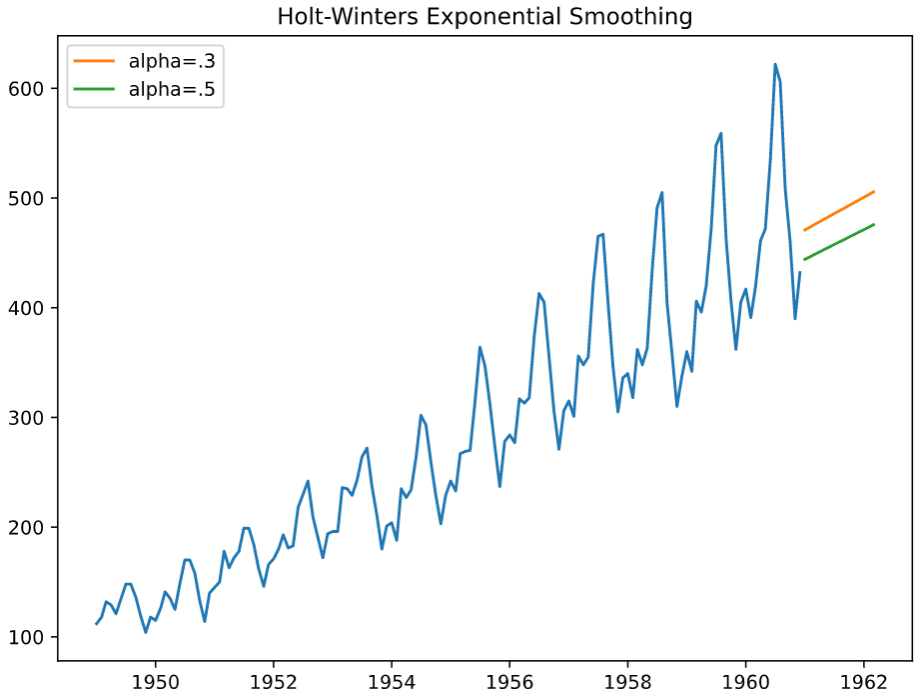
Holt-Winters Exponential Smoothing in Python

- Again, we will use the air passenger data as example.

```
from statsmodels.tsa.holtwinters import
Holt
model1 = Holt(df).fit(smoothing_level=0.3)
model2 = Holt(df).fit(smoothing_level=0.5)
pred1 = model1.forecast(steps=15)
pred2 = model2.forecast(steps=15)
```

Month	#Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

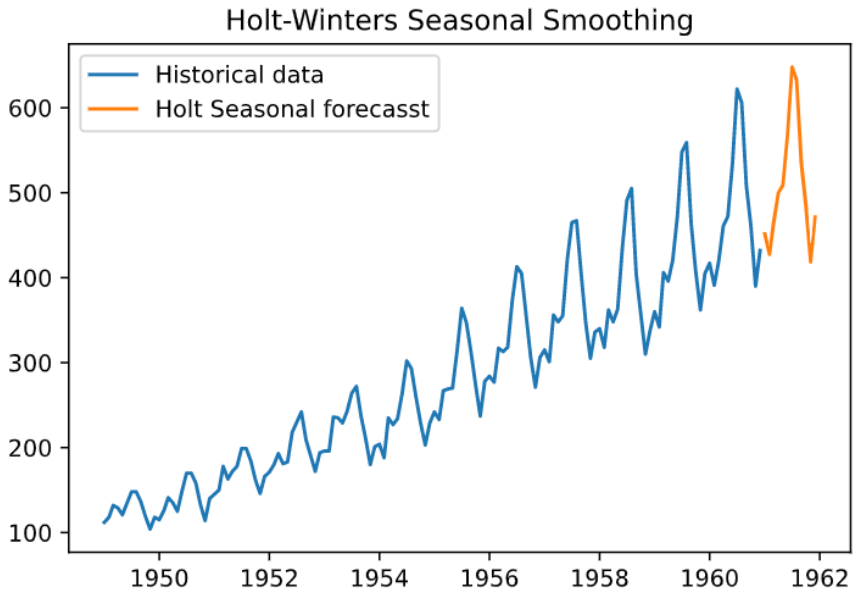
Holt-Winters Exponential Smoothing in Python



The problem with Holt-Winters linear trend method is that the trend is constant in the future, increasing or decreasing indefinitely. For long-term forecast, this can be problematic.

Holt-Winter's Seasonal Smoothing model

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

model = ExponentialSmoothing(df, trend="add",
                             seasonal="add",
                             seasonal_periods=12).fit()
pred1 = model.forecast(12)
```



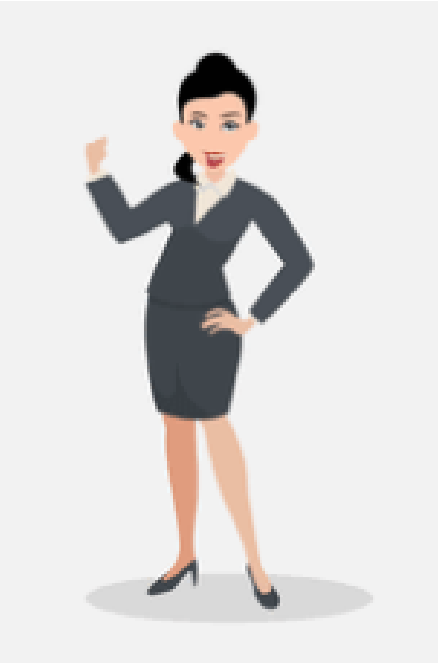
- With the seasonal component involved, we can observe that the future forecast is more realistic.

Time Series Forecasting Model

ARIMA

Introduction to ARIMA models

- Exponential smoothing methods are useful for making forecast, and make no assumptions about the correlation between successive value of time series.
- In some cases, we can make better predictive model by taking correlation in the data into account.
- **A**uto**r**egressive **I**ntegrated **M**oving **A**verage (ARIMA) models include an explicit statistical model for the irregular component (randomness) of a time series.



AR & MA Models

- Autoregressive AR process:
  - AR(p) – Current values depend on its own p-previous values
  - p is the order of AR process.
- Moving average MA process:
  - MA(q) The current deviation from mean depends on q-previous deviations.
  - q is the order of MA process.

- ARIMA is a popular because algorithm due to its generality; It can handle many time series, with or without seasonal elements.

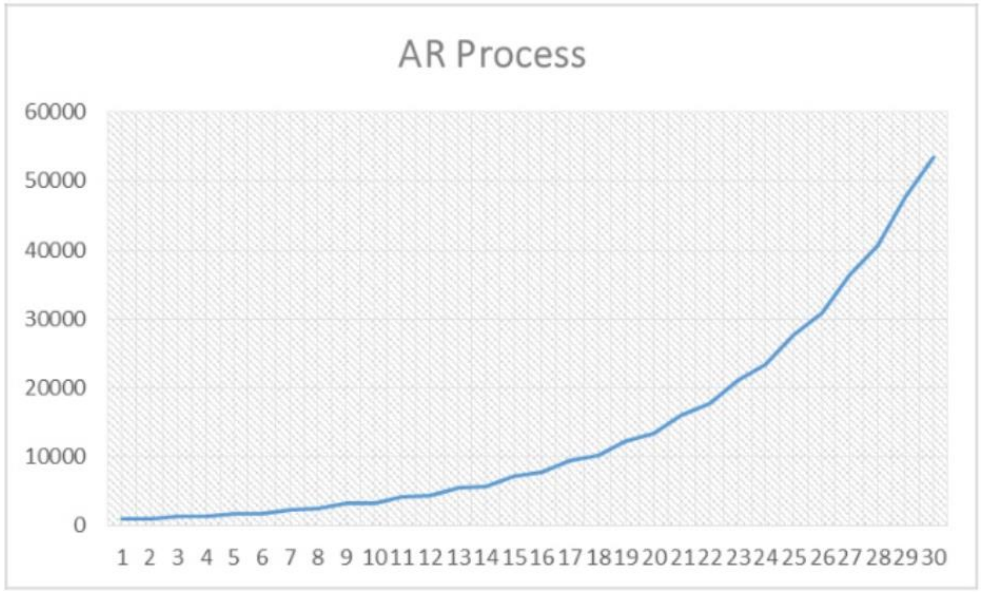
Autoregressive Models

- Used for forecasting
- Takes advantage of autocorrelation
  - 1st order – correlation between consecutive values
  - 2nd order – correlation between values 2 periods apart
- **pth order** autoregressive model:

$$X_t = \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + \ \ldots \ + \varphi_p X_{t-p}$$

- X_{t-1}, X_{t-2}, …, X_{t-p} are the time series inputs
- $\varphi_1, \varphi_2, \ldots, \varphi_p$ are model parameters

Autoregressive Process

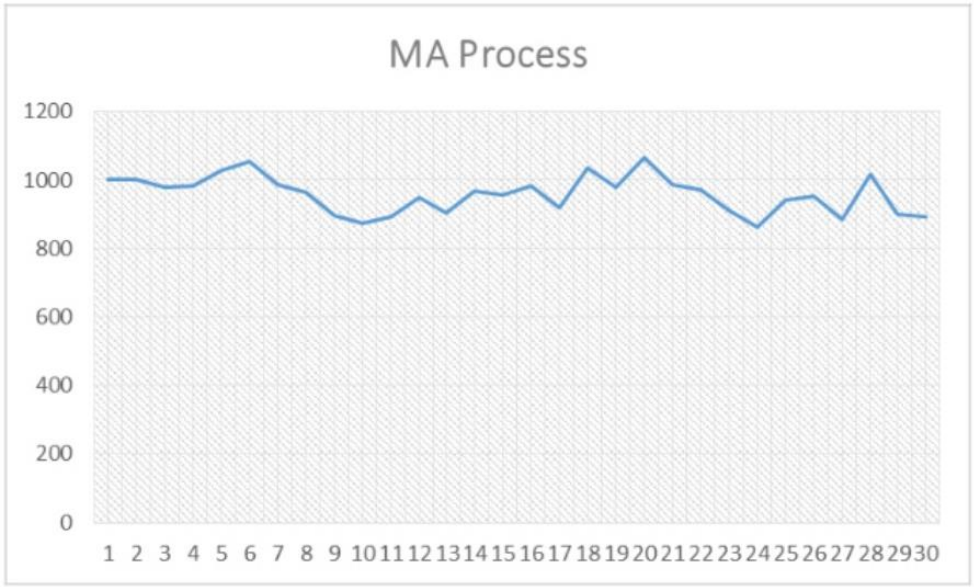


AR(1) $y_t = a1 * y_{t-1}$

AR(2) $y_t = a1 * y_{t-1} + a2 * y_{t-2}$

AR(3) $y_t = a1 * y_{t-1} + a2 * y_{t-2} + a3 * y_{t-3}$

Moving Average Process



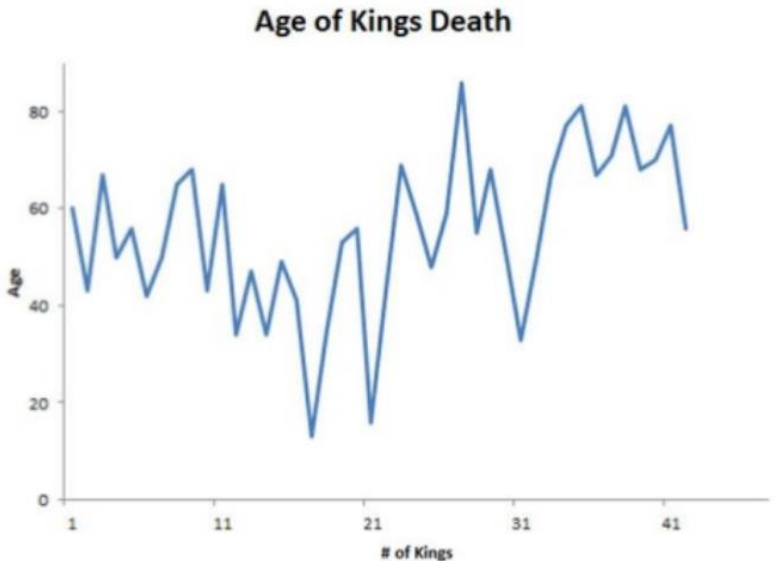
MA(1) $\varepsilon_t = b1 * \varepsilon_{t-1}$

MA(2) $\varepsilon_t = b1 * \varepsilon_{t-1} + b2 * \varepsilon_{t-2}$

MA(3) $\varepsilon_t = b1 * \varepsilon_{t-1} + b2 * \varepsilon_{t-2} + b3 * \varepsilon_{t-3}$

Differencing Process

- ARIMA models are for **stationary time series**. If we start with a non-stationary time series, we will first need to "difference" the time series.
- We have to difference the time series **d** times to obtain a stationary series, then we will have an ARIMA(p, d, q) model, where **d** is the order of differencing used.



- Original Dataset

- Difference = 1

ARIMA Model

Y_t → **AR filter** → **Integration filter** → **MA filter** → ε_t

(long term) (stochastic trend) (short term) (white noise error)

ARIMA (2,0,1) $y_t = a1 * y_{t-1} + a2 * y_{t-2} + b1 * \varepsilon_{t-1}$

ARIMA (3,0,1) $y_t = a1 * y_{t-1} + a2 * y_{t-2} + a3 * y_{t-3} + b1 * \varepsilon_{t-1}$

ARIMA (1,1,0) $\Delta y_t = a1 * \Delta y_{t-1} + \varepsilon_t, where\ \Delta y_t = y_t - y_{t-1}$

ARIMA (2,1,0) $\Delta y_t = a1 * \Delta y_{t-1} + a2 * \Delta y_{t-2} + \varepsilon_t, where\ \Delta y_t = y_t - y_{t-1}$

Identification of orders p and q

- ARIMA(p,d,q)
- Identification starts with d, in most of cases, d = 1. By setting this, the time series will become stationary.
- Then, we need to learn about ACF & PACF test to identify p and q.

- Once we are working with a stationary time series, we can examine the ACF and PACF to help identify the proper number of previous *y* (AR) terms and ε (MA) terms.

Autocorrelation Function (ACF)

- Autocorrelation is a correlation coefficient. However, instead of correlation between two different variables, the correlation is between the same variable at time X_t and X_{t+h}.
- Correlation with lag-1, lag-2, lag-3, etc.
- The ACF represents the degree of persistence over respective lags of variables.

$$ACF = Corr(X_{t+h}, X_t)$$

$$ACF(0) = 1$$

Partial Autocorrelation Function (PACF)

- In general, the partial correlation between two variables is the amount of correlation between them which is not explained by their mutual relationship with a specified set of other variables.
- For example, if we are regressing a variable Y with other variables X_1, X_2, and X_3, the partial correlation between Y and X_3 is the amount of correlation that is not explained by their common correlation with X_1 and X_2.
- **Partial correlation** measures the degree of association between two random variables, with the effect of a set of controlling random variable removed.

Interpret ACF & PACF: https://online.stat.psu.edu/stat510/lesson/3/3.1

Measuring ACF and PACF in Python

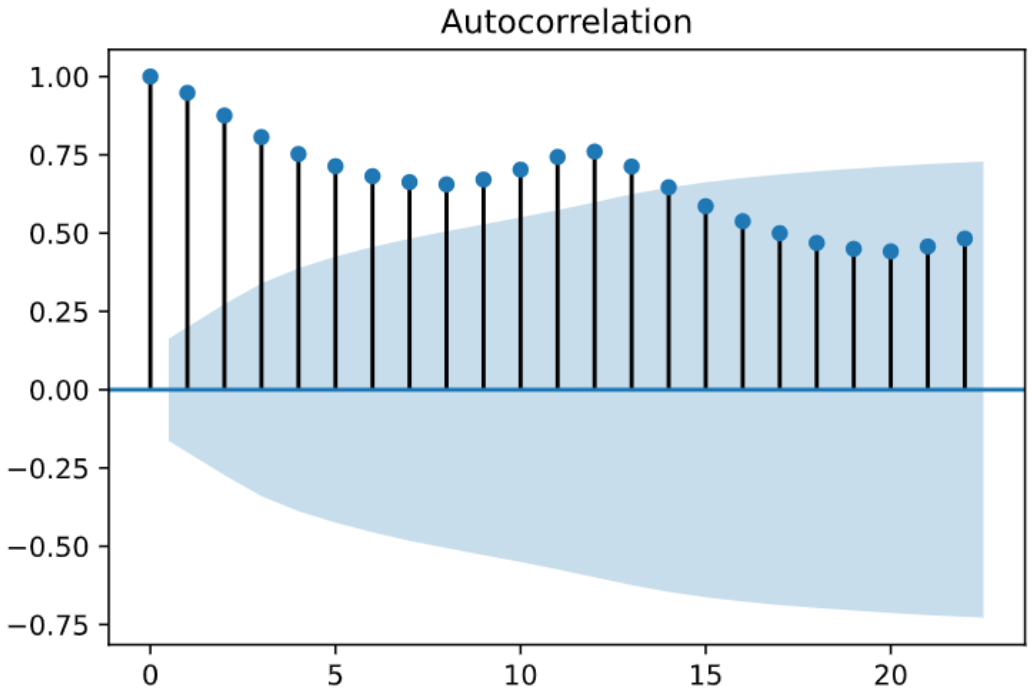
- Again, we will use the air passenger data as example.

```
df = pd.read_csv('data/AirPassengers.csv')
df.index = pd.to_datetime(df.Month , format = '%Y-%m')
df.drop('Month',axis = 1, inplace = True)
```

Month	#Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

Measuring ACF and PACF in Python

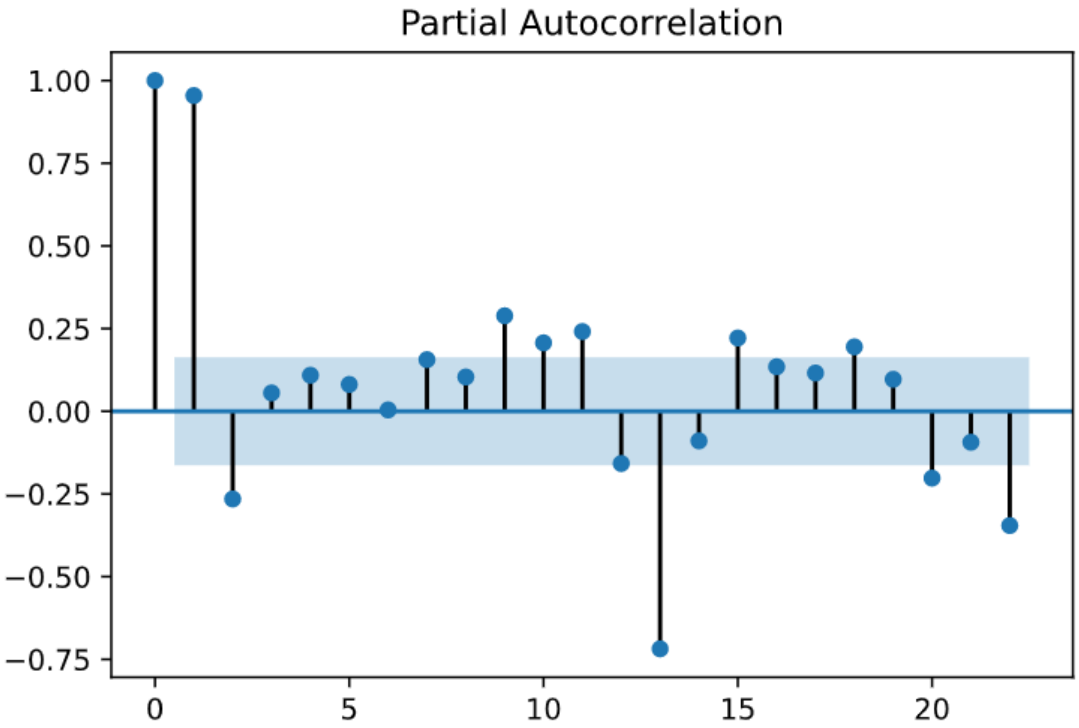
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(df)
plt.show()
```



The ACF function will be used to determine the **order of MA(q)** process. MA(1) has one significant spike at lag1, MA(2) has two significant spike at lag 1 and 2, etc.

Measuring ACF and PACF in Python

```
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(df)
plt.show()
```



The PACF function will be used to determine the **order of AR(p)** process. AR(1) has one significant spike at lag1, AR(2) has two significant spike at lag 1 and 2, etc.

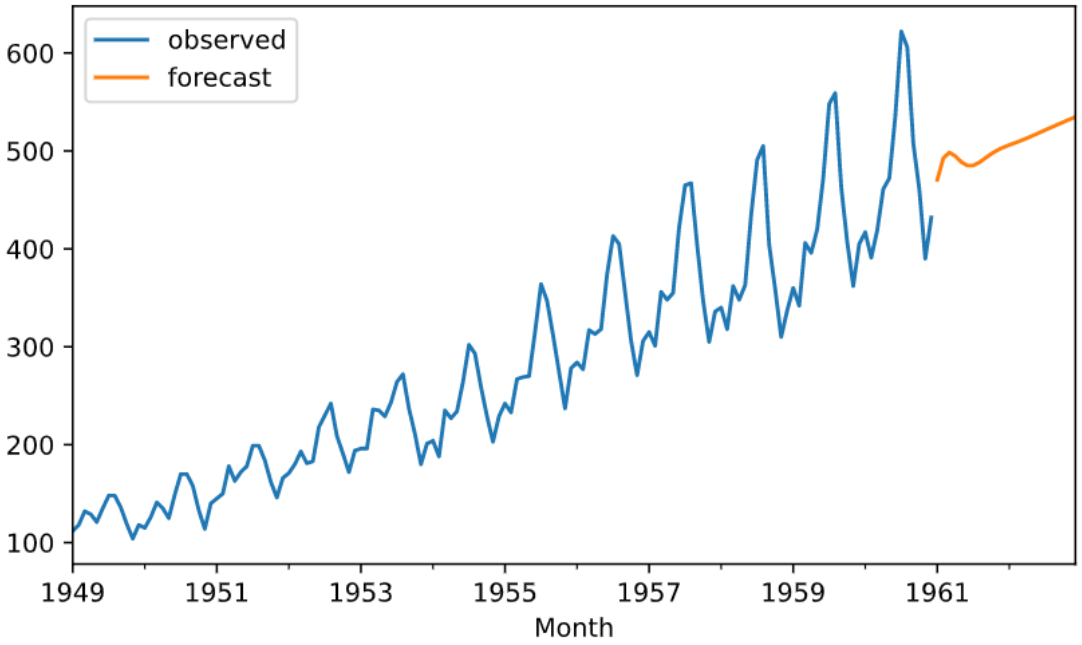
Build ARIMA Model

```
from statsmodels.tsa.arima_model import ARIMA

arima_model = ARIMA(df, order=(2,1,1)).fit()
arima_model.summary()
```

ARIMA Model Results

Dep. Variable:	D.#Passengers	No. Observations:	143
Model:	ARIMA(2, 1, 1)	Log Likelihood	-675.848
Method:	css-mle	S.D. of innovations	26.881
Date:	Wed, 17 Mar 2021	AIC	1361.696
Time:	08:54:31	BIC	1376.510
Sample:	02-01-1949	HQIC	1367.716
	- 12-01-1960		

Build ARIMA Model



- As seen from the model prediction, the model did capture the increasing trend of the passengers.
- However, the prediction is not able to follow the past seasonality. Therefore, a more advanced ARIMA model is required!

Seasonal ARIMA Model

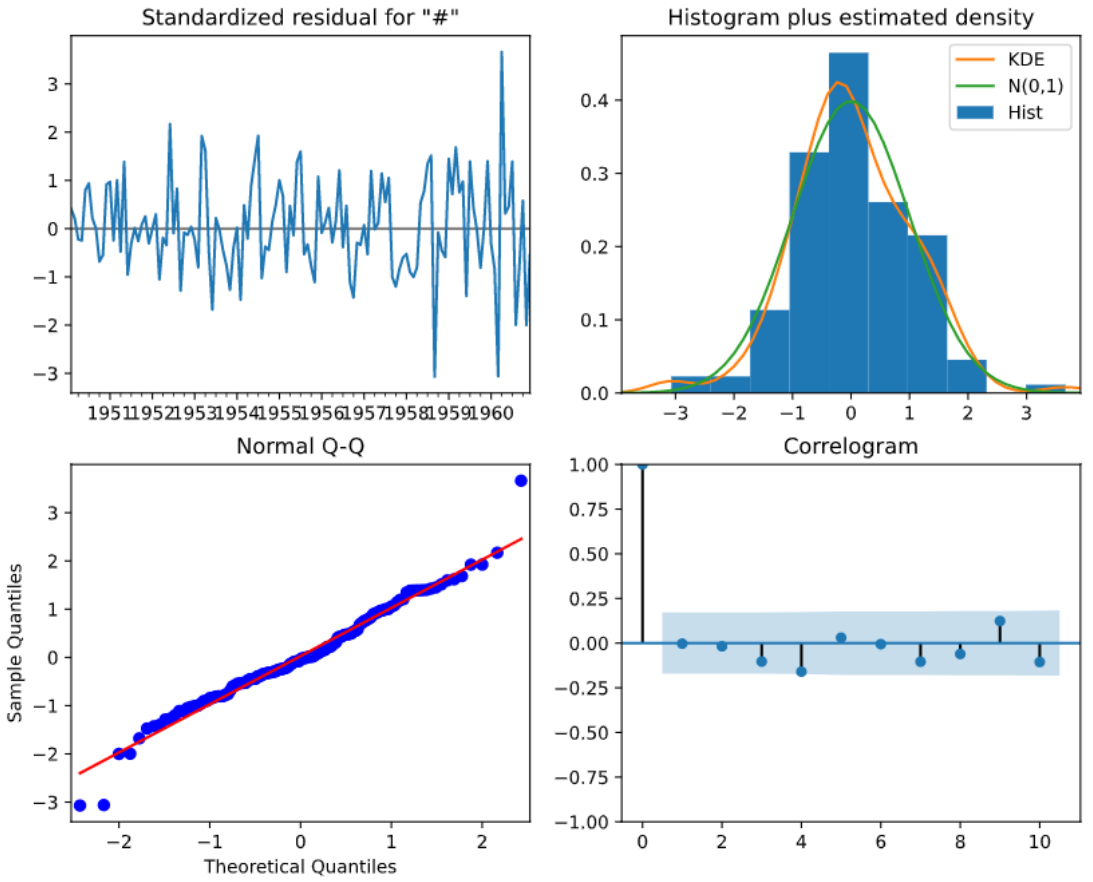
```
from statsmodels.tsa.statespace.sarimax import SARIMAX

arima_model = SARIMAX(df, order=(2,1,0),
                      seasonal_order=(1,1,0,12)).fit()
arima_model.summary()
```

SARIMAX Results

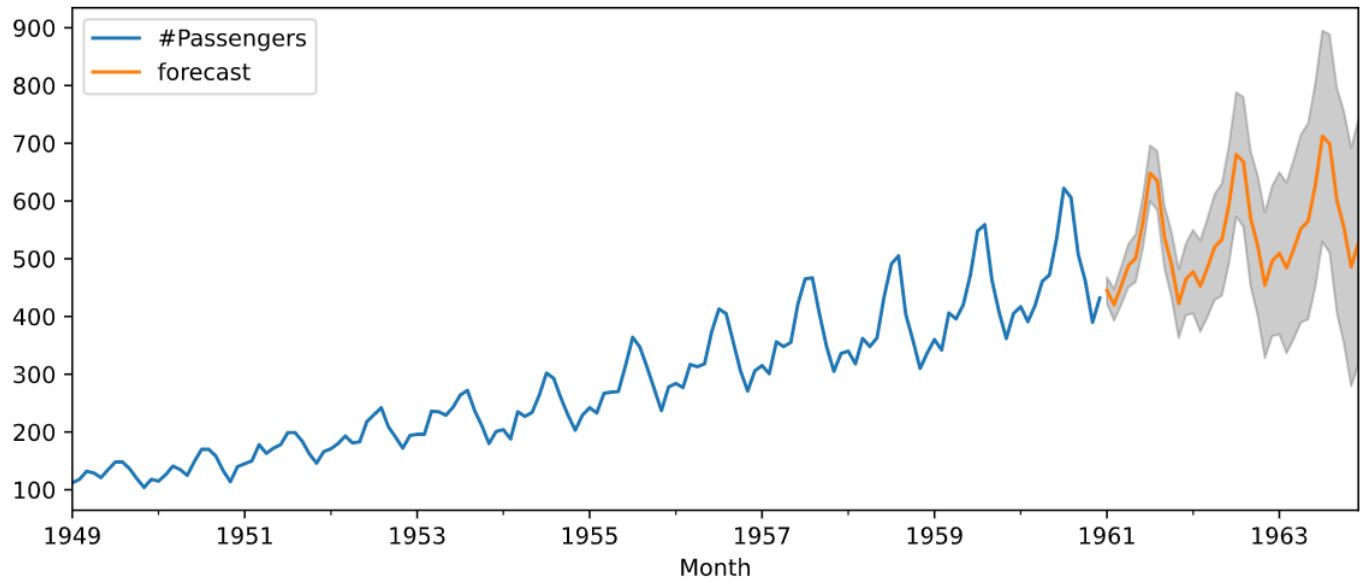
Dep. Variable:	#Passengers	**No. Observations:**	144
Model:	SARIMAX(2, 1, 0)x(1, 1, 0, 12)	**Log Likelihood**	-507.186
Date:	Tue, 16 Mar 2021	**AIC**	1022.373
Time:	17:52:12	**BIC**	1033.874
Sample:	01-01-1949	**HQIC**	1027.046

Seasonal ARIMA Model

- Let's perform diagnostics plots for the ARIMA model. The four graphs are standardized residual over time, histogram of standardized residual, normal Q-Q plot, and correlogram.
- From these plots, we can observe that the residuals conform to normal distribution, also the correlogram suggests that ther is no autocorrelation in the residuals, so they are effectively white noise.
- In conclusion, the ARIMA model is a relatively good fit.



Seasonal ARIMA Model



- The last step is to use the ARIMA model to predict the future air passenger numbers for the next 3 years.
- As we can observed from the prediction, as the confident interval is getting larger, the model is less confident to predict the more distant future,

Auto-ARIMA Model

```
from pmdarima.arima import auto_arima

arima_model = auto_arima(df, seasonal=True, m=12)

arima_model.summary()
```

SARIMAX Results

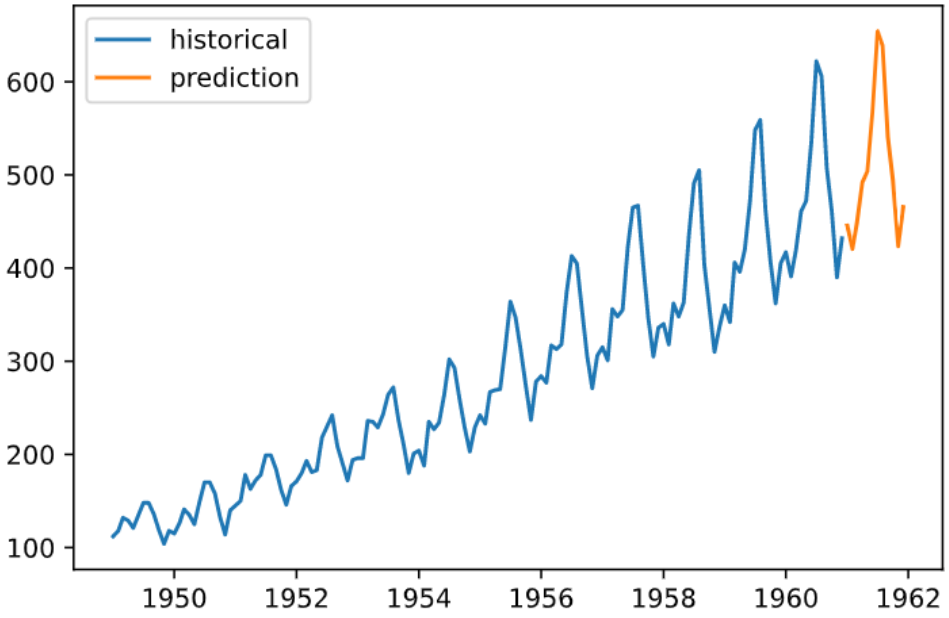
Dep. Variable:	y	No. Observations:	144
Model:	SARIMAX(2, 1, 1)x(0, 1, [], 12)	Log Likelihood	-504.923
Date:	Sat, 13 Mar 2021	AIC	1017.847
Time:	10:54:59	BIC	1029.348
Sample:	0	HQIC	1022.520
	- 144		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5960	0.085	6.987	0.000	0.429	0.763
ar.L2	0.2143	0.091	2.343	0.019	0.035	0.393
ma.L1	-0.9819	0.038	-25.595	0.000	-1.057	-0.907
sigma2	129.3137	14.556	8.884	0.000	100.784	157.844

Auto-ARIMA Model

Forecasting the future

```
prediction = pd.DataFrame(arima_model.predict(n_periods=12),
                          index=pd.date_range('1961-01', periods=12, freq='MS'),
                          columns=['prediction'])
```



Validation: How good is my model?

- Does our model really give an adequate description of the data?
- Two criteria to check the goodness of fit:
  - Akaike information criterion (AIC)
  - Bayesian information criterion (BIC)
- These two measures are useful in comparing two models.
- The smaller the AIC & BIC, the better the model.

Fully Automatic time-series Prediction Model: Facebook Prophet

Why Facebook Prophet model

- It provides us with the ability to make time series predictions with good accuracy using simple intuitive parameters
- Robust to missing data and shifts in the trend, and typically handles outliers .

If you want to taste on the Facebook Prophet, below is the documentation link:

https://facebook.github.io/prophet/

Summary

What we have learnt

- Understand time series concepts.
- Breakdown time series into trend, seasonality and random components.
- Perform statistical test on time series data.
- Application of time series models for prediction purpose.

Additional Resources

- Forecasting: Principles and Practices: https://otexts.com/fpp2/
- MIT open course: https://ocw.mit.edu/courses/economics/14-384-time-series-analysis-fall-2013/
- Oregon State University: http://stat565.cwick.co.nz/
- PennState University: https://online.stat.psu.edu/stat510/
- Coursera: https://www.coursera.org/learn/practical-time-series-analysis