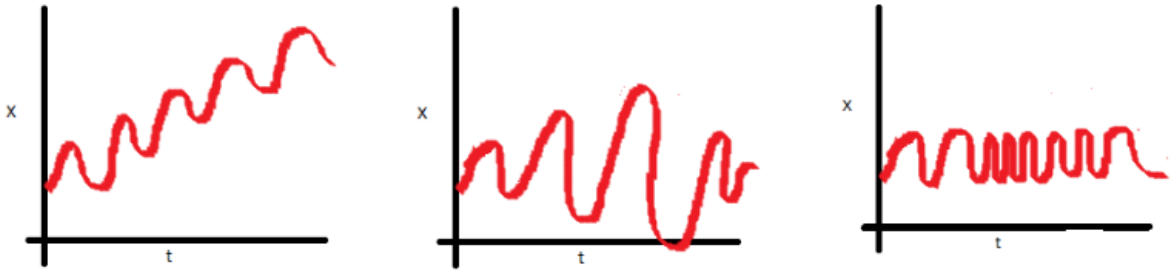
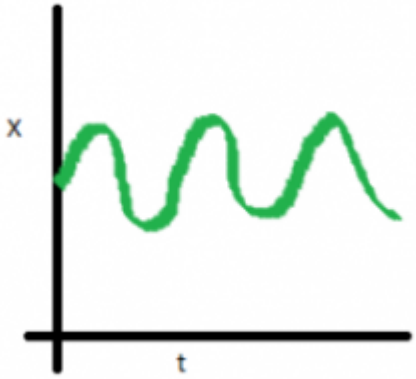
Handling a Non-Stationary Time Series

- ‘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.



- 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. For a series to be classified as stationary, it should not exhibit a trend.
- on to 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.
- If you look at the third plot, the spread becomes closer as the time increases, which implies that the covariance is a function of time.
- The three examples shown above represent non-stationary time series. Now look at stationary time-series

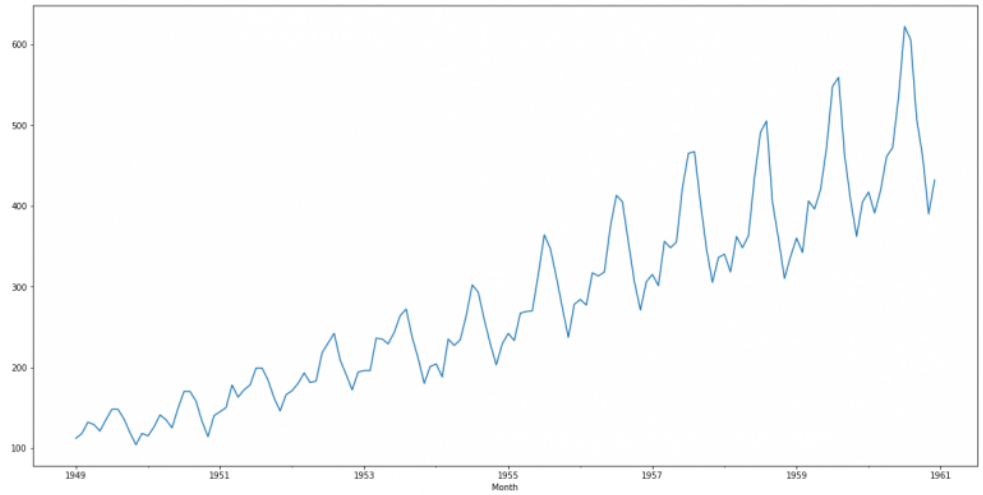


- In this case, the mean, variance and covariance are constant with time. This is what a stationary time series looks like.
- So to summarize, a stationary time series is the one for which the properties (namely mean, variance and covariance) do not depend on time. In the next section we will cover various methods to check if the given series is stationary or not.

Methods to Check Stationarity

1. Visual test :-

- Code train['feature_name'].plot() By run this it produce the graph like



- Although its very clear that we have a trend (varying mean) in the above series, this visual approach might not always give accurate results. It is better to confirm the observations using some statistical tests.

2. Statistical test (ADF (Augmented Dickey Fuller) Test):-

- The Dickey Fuller test is one of the most popular statistical tests. It can be used to determine the presence of unit root in the series, and hence help us understand if the series is stationary or not. The null and alternate hypothesis of this test are:

```
            - Null Hypothesis: The series has a unit root (value of a
 =1)(not stationary)

            - Alternate Hypothesis: The series has no unit root.(stati
onary)
```

- If we fail to reject the null hypothesis, we can say that the series is non-stationary. This means that the series can be linear or difference stationary (we will understand more about difference stationary in the next section).

```
#Ho: It is non stationary
#H1: It is stationary

## Augmented Dickey Fuller) Test
from statsmodels.tsa.stattools import adfuller
def adfuller_test(sales):
    result=adfuller(sales)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of
Observations Used']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )
    if result[1] <= 0.05:
        print("strong evidence against the null hypothesis(Ho), reject the null
hypothesis. Data has no unit root and is stationary")
    else:
        print("weak evidence against null hypothesis, time series has a unit
root, indicating it is non-stationary ")
```

Making a Time Series Stationary

1. Differencing :-

- In this method, we compute the difference of consecutive terms in the series. Differencing is typically performed to get rid of the varying mean. Mathematically, differencing can be written as:

$$yt' = yt - y(t-1)$$

- where yt is the value at a time t
- Applying differencing on our series and plotting the results:

```
train['col_diff'] = train['col_name'] - train['col_name'].shift(1)
train['col_diff'].dropna().plot()
```

2. Seasonal Differencing :-

- In seasonal differencing, instead of calculating the difference between consecutive values, we calculate the difference between an observation and a previous observation from the same season.
- For example, an observation taken on a Monday will be subtracted from an observation taken on the previous Monday. Mathematically it can be written as:

$$yt' = yt - y(t-n)$$

```
n=7
train['Col_seasonal_diff'] = train['Col_name'] - train['col_name'].shift(n)
```

3. Transformation

- Transformations are used to stabilize the non-constant variance of a series. Common transformation methods include power transform, square root, and log transform.

```
train['col_log'] = np.log(train['col_name'])
train['col_log_diff'] = train['col_log'] - train['col_log'].shift(1)
train['col_log_diff'].dropna().plot()
```