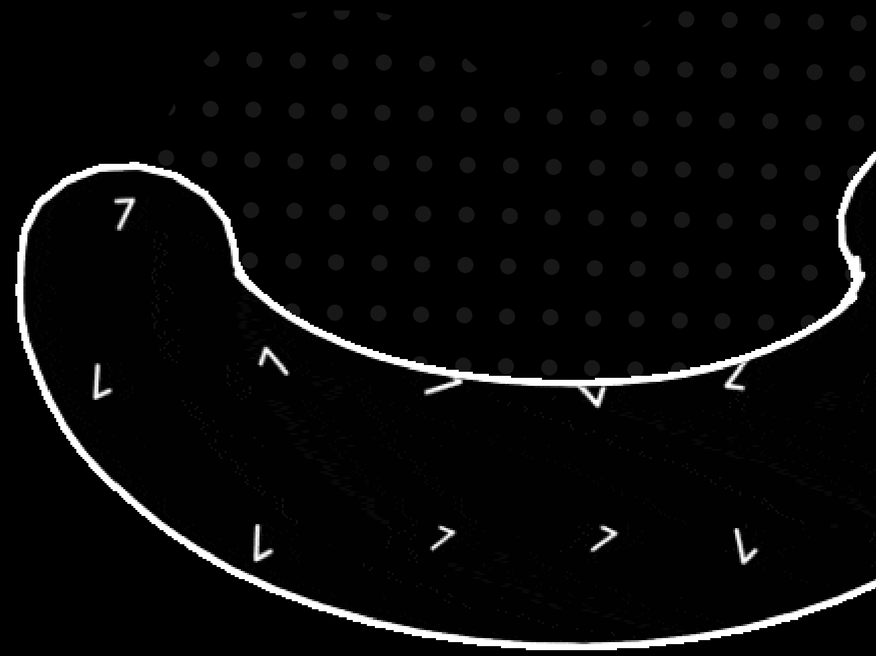


A review of gap-filling techniques in time series data

SARA IRIS GARCIA

PYCON HK 2020

About me

Data scientist based in Guatemala, and an active Python community member. I'm the co-lead of PyLadies Guatemala City, and the Women in Data Guatemala city chapter co-lead .

Content

- Linear interpolation
- Spline interpolation
- Simple, Cumulative and Exponential moving average
- Kalman smoothing

What is time series data?

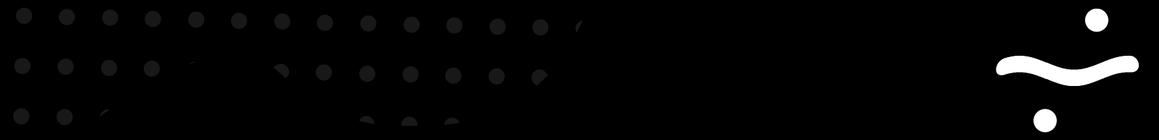
Series of observations ordered by successive, equally spaced time intervals. **Time series prediction is about to forecast future values based on historical records**.

For several reasons, observations are often missing or recorded at irregular time steps, which is very common in sociology data, geology, genomics, physical, financial, and sales data.

Why data goes missing?

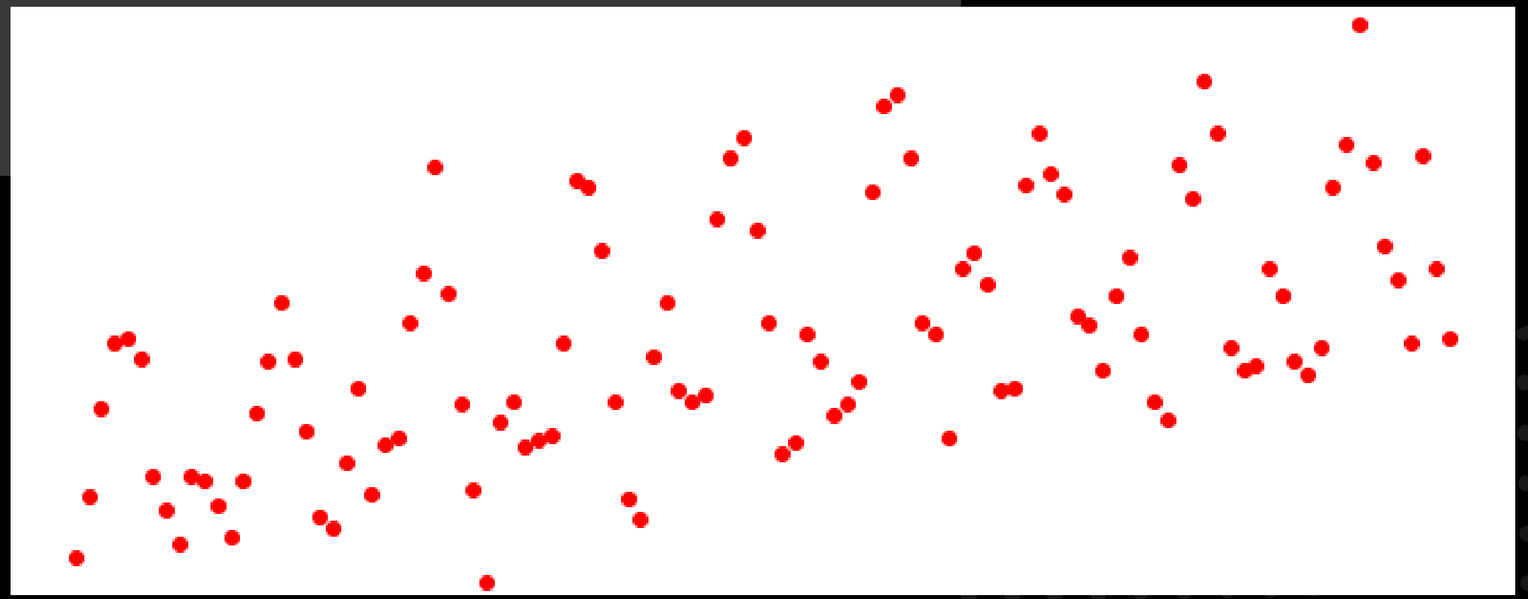
- **Missing at Random**: there is not sufficient evidence that we are systematically missing data.
- **Missing Completely at Random**: we find NO significant differences between the primary variable of interest and all the rest of the variables.
- **Missing not at Random**: the value of the variable that is missing is related to the reason it's missing.

Our dataset



```
1 # download car sales dataset
2 df = pd.read_csv('https://raw.githubusercontent.com/jbrownlee/Datasets/master/monthly-car-sales.csv')
3 df.head()
```

	Month	Sales
0	1960-01	6550
1	1960-02	8728
2	1960-03	12026
3	1960-04	14395
4	1960-05	14587

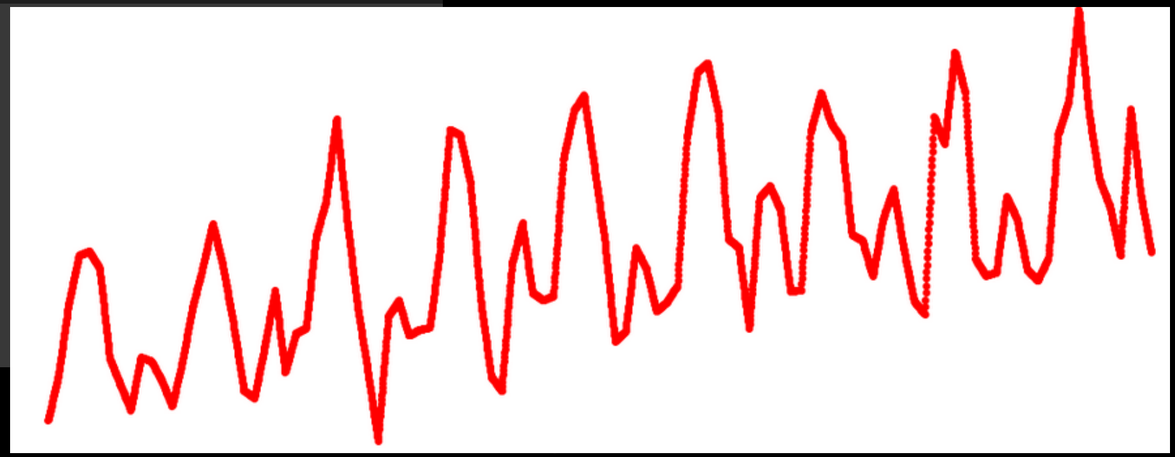


Linear Interpolation

Linear interpolation calculates values that lie on a line between two known data points. Good choice for fairly linear data, like a series with a strong trend.

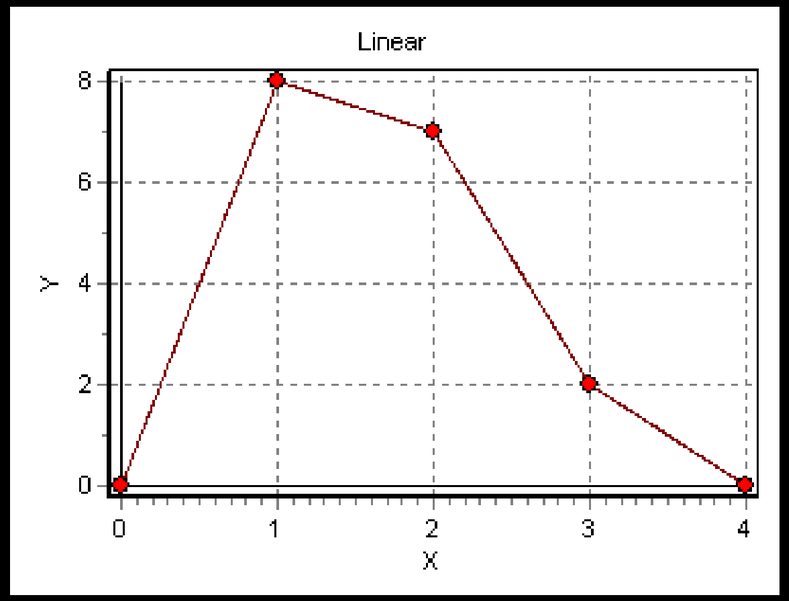
```
1 # Resample data and fill the values using linear interpolation
2 resampled = df.Sales.resample('1D')
3 linear = resampled.interpolate(method='linear')
4 linear.head()
```

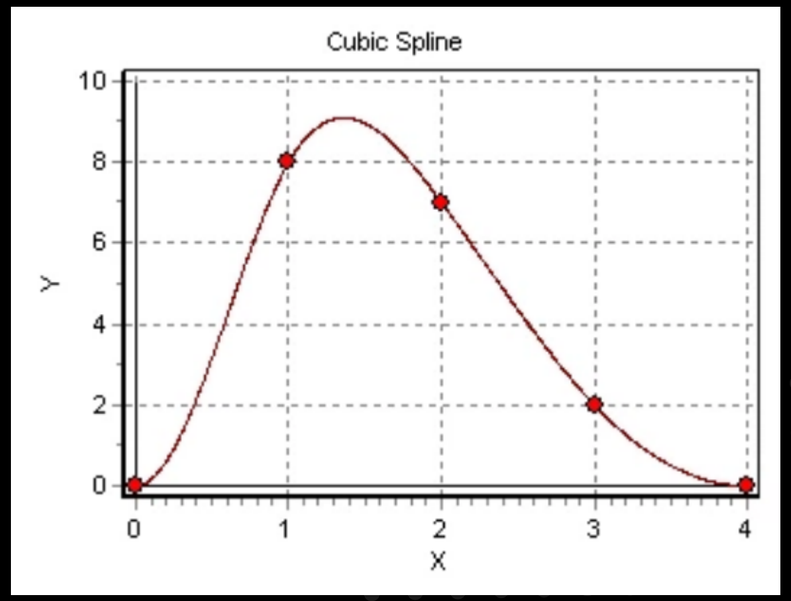
```
Month
1960-01-01    6550.000000
1960-01-02    6620.258065
1960-01-03    6690.516129
1960-01-04    6760.774194
1960-01-05    6831.032258
Freq: D, Name: Sales, dtype: float64
```



Spline Interpolation

Spline interpolation is a form of interpolation where the interpolant is a special type of piecewise polynomial called a spline. It often does not have as much error as polynomial interpolation, and is more appropriate for series without a strong trend, because it calculates a non-linear approximation using multiple data points.

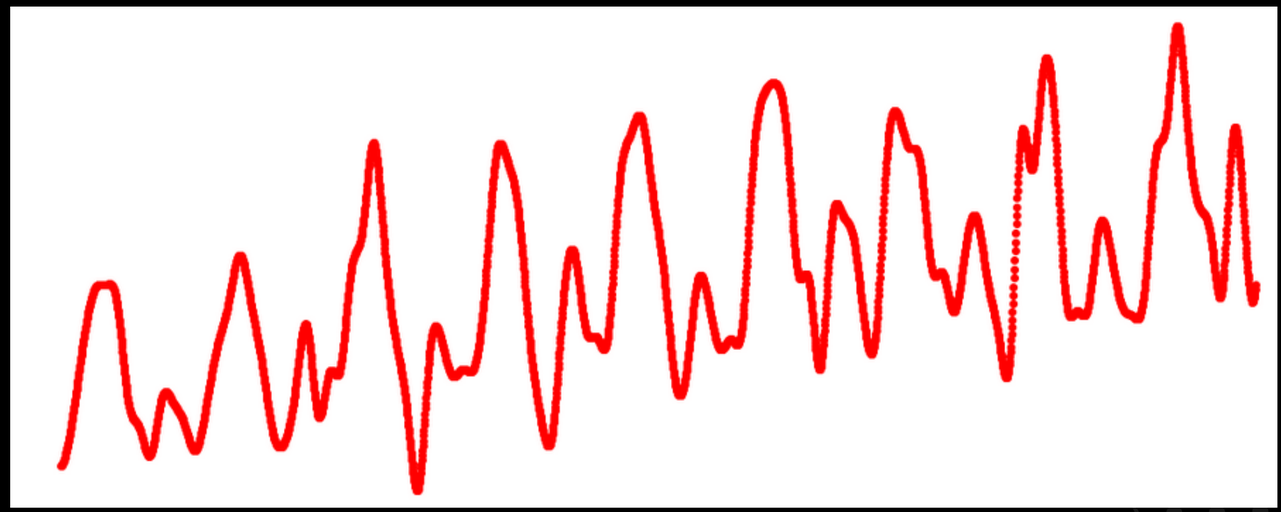




Spline Interpolation

```
1 spline = resampled.interpolate(method='spline', order=3)
2 spline.head()
```

```
Month
1960-01-01    6550.000000
1960-01-02    6571.041915
1960-01-03    6596.262592
1960-01-04    6625.571852
1960-01-05    6658.876538
Freq: D, Name: Sales, dtype: float64
```

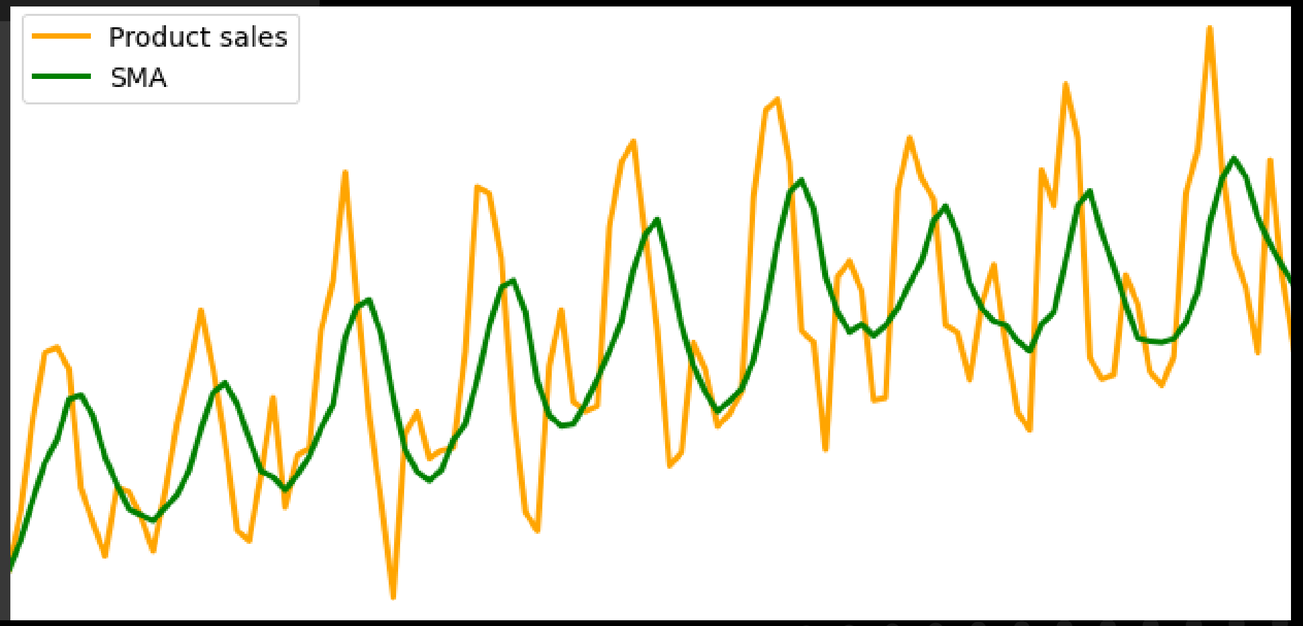


Simple Moving Average

Simple Moving Average uses a sliding window to take the average over a set number of time periods. It is an equally weighted mean of the previous n data.

```
1 df['SMA'] = df.rolling(window=5, min_periods=1).mean()
2 df.head()
```

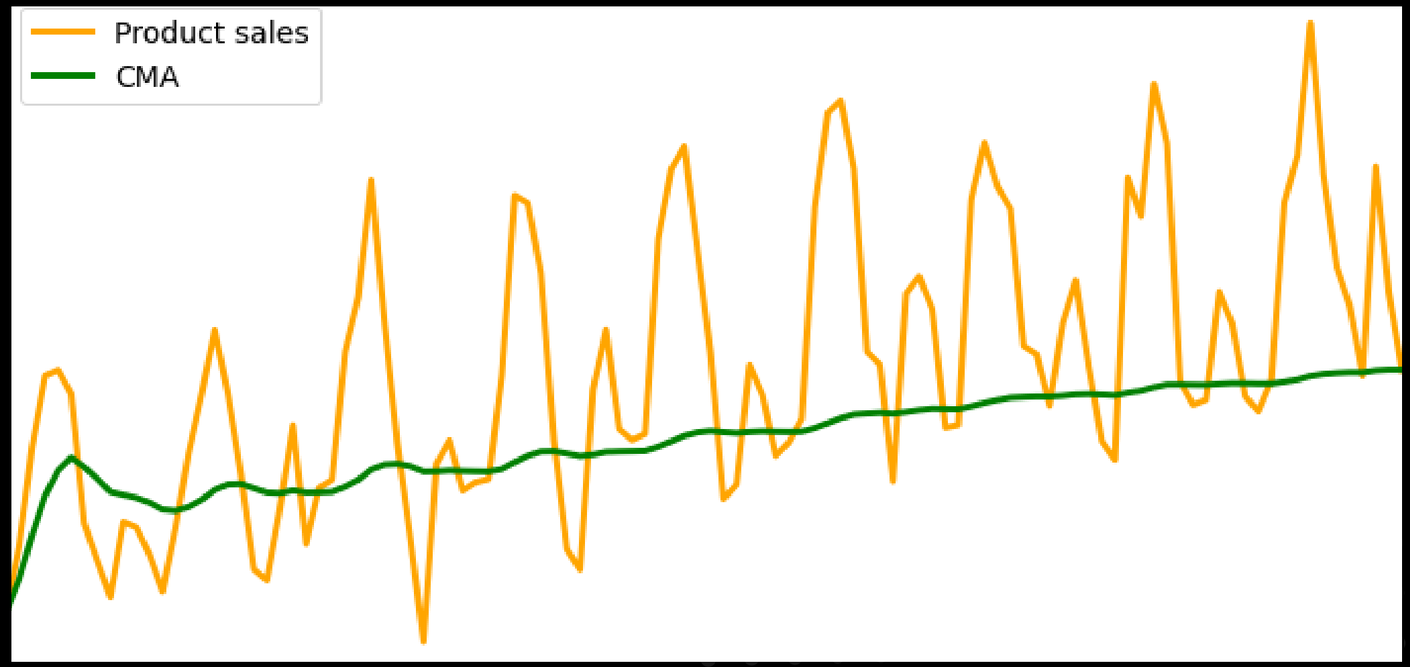
Month	Sales	SMA
1960-01-01	6550	6550.000000
1960-02-01	8728	7639.000000
1960-03-01	12026	9101.333333
1960-04-01	14395	10424.750000
1960-05-01	14587	11257.200000



Cumulative Moving Average

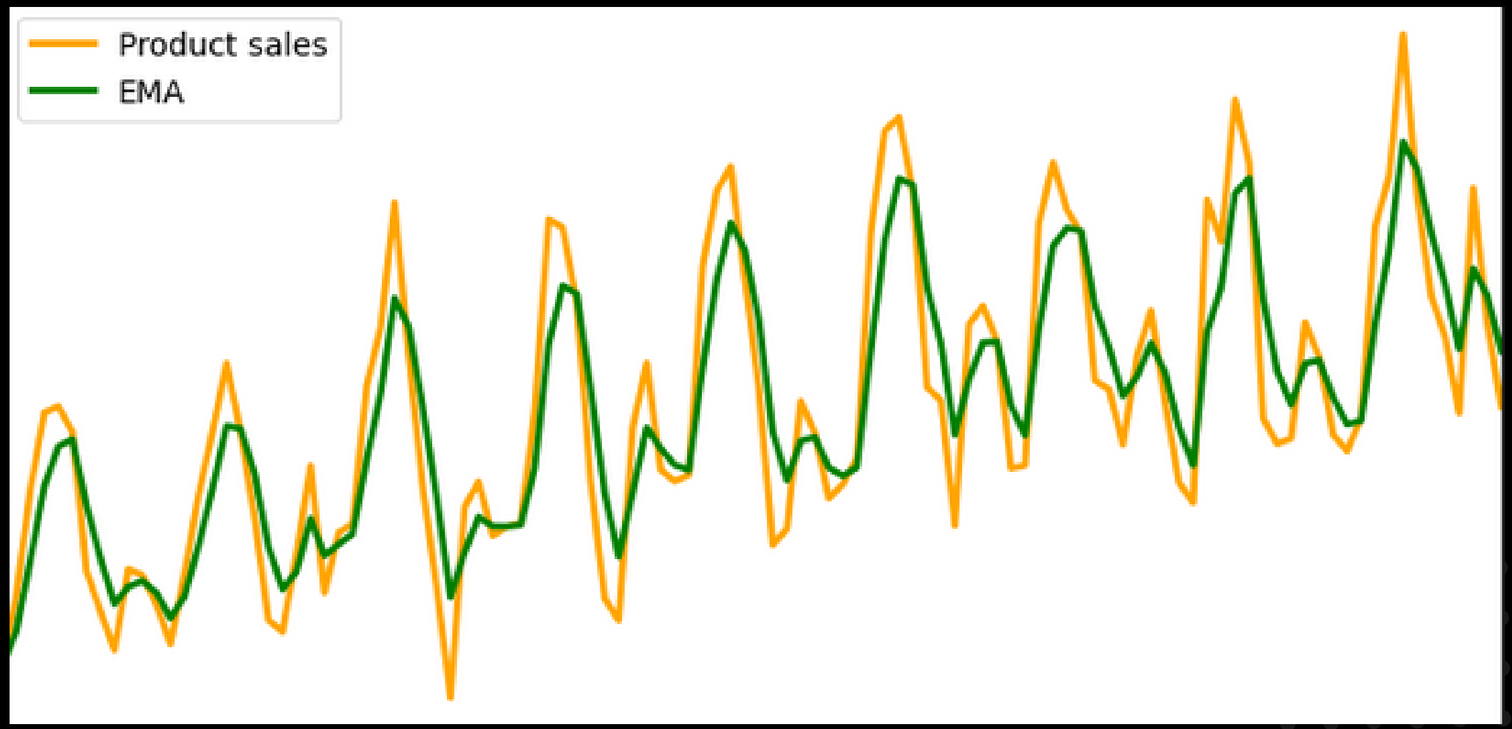
```
1 df['CMA'] = df['Sales'].expanding().mean()
2 df.head(10)
```

Month	Sales	SMA	CMA
1960-01-01	6550	6550.000000	6550.000000
1960-02-01	8728	7639.000000	7639.000000
1960-03-01	12026	9101.333333	9101.333333
1960-04-01	14395	10424.750000	10424.750000
1960-05-01	14587	11257.200000	11257.200000
1960-06-01	13791	12705.400000	11679.500000
1960-07-01	9498	12859.400000	11367.857143
1960-08-01	8251	12104.400000	10978.250000
1960-09-01	7049	10635.200000	10541.666667
1960-10-01	9545	9626.800000	10442.000000

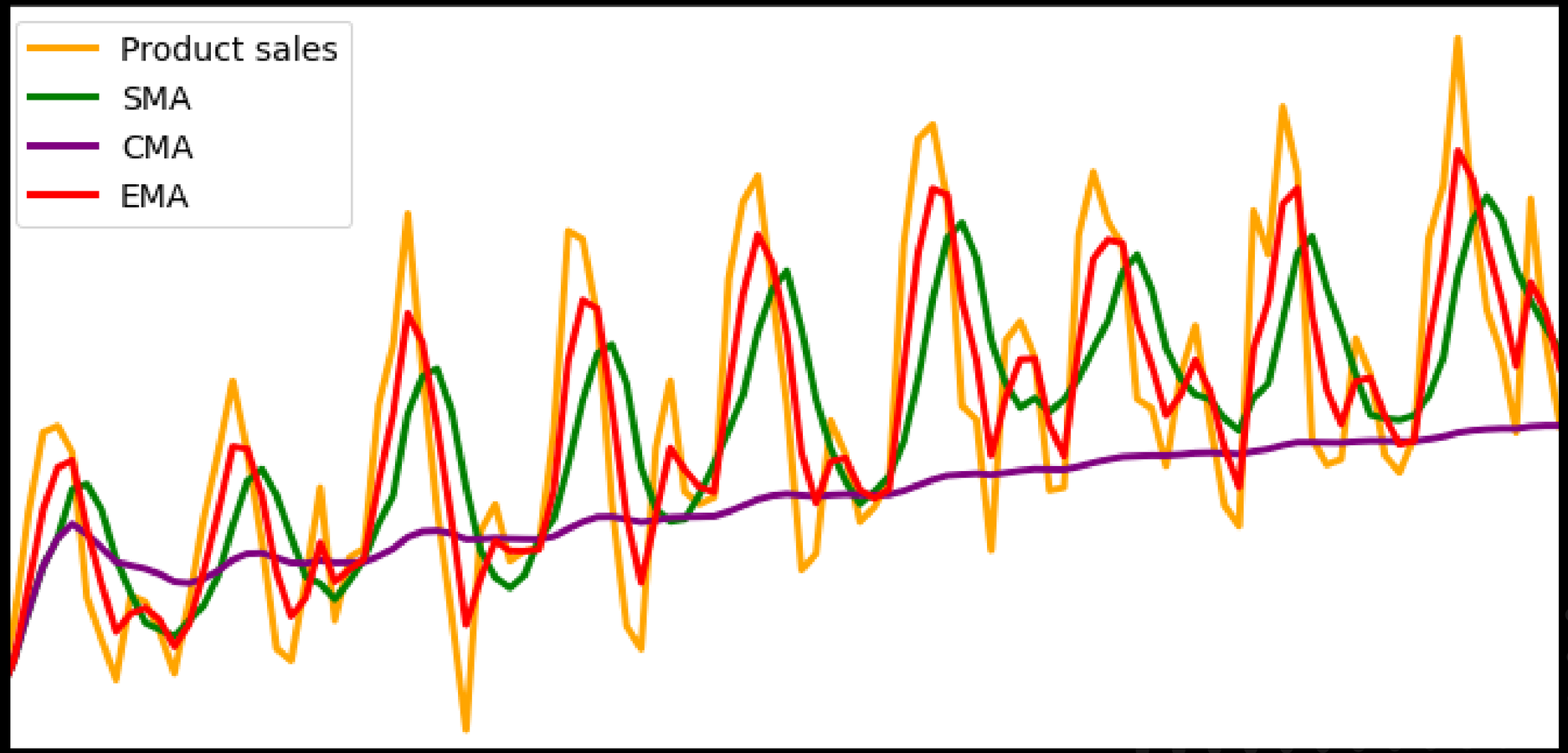


Exponential Moving Average

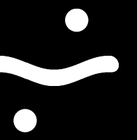
```
1 df['EMA'] = df['Sales'].ewm(alpha=0.5, adjust=False).mean()
```



Comparison



Kalman Smoothing



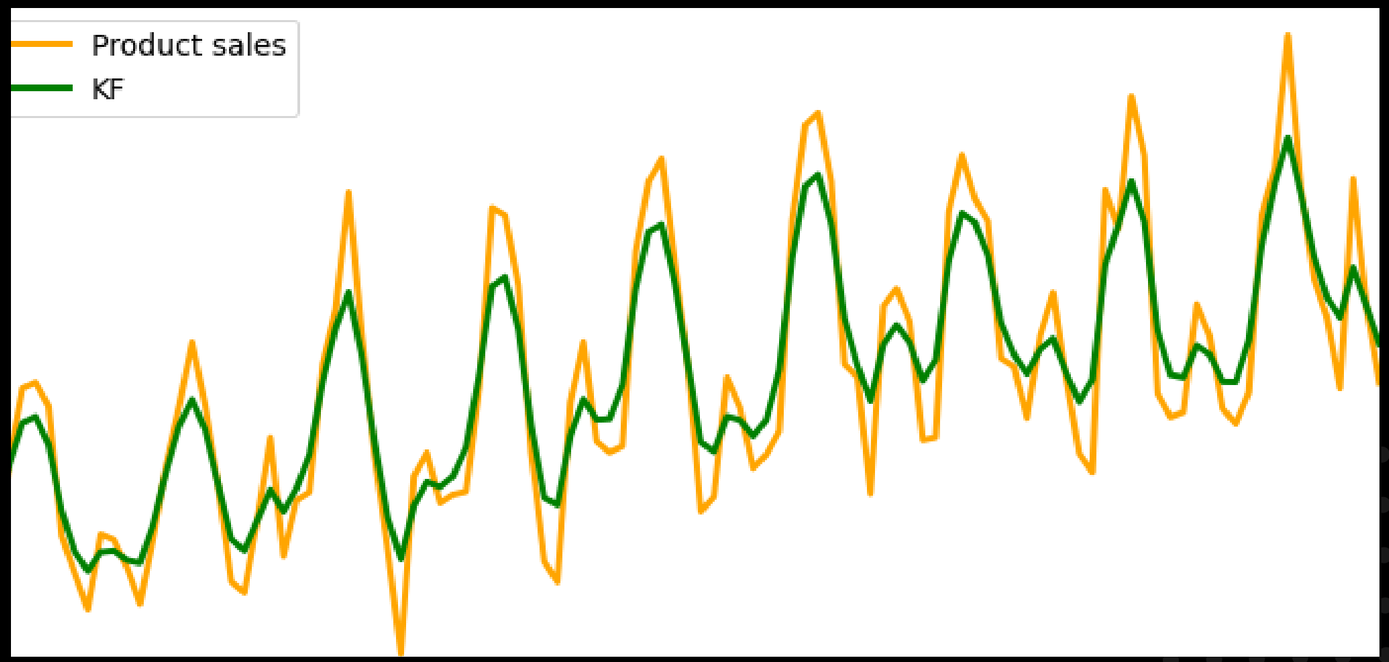
It is a recursive algorithm for estimating the state of a system in such a way that the expected minimum-mean squared error is minimized. The algorithm operates in a prediction-correction cycle.

First, the state vector and covariance of the next time step are predicted using the state-transition function. Then an observation is made and the updated state and covarianceis computed as a linear combination of the information from the measurement vector and the prediction.

```
1 pip install pykalman
2 from pykalman import KalmanFilter
```

Kalman Smoothing

```
1 kf = KalmanFilter(em_vars='Sales')
2 kf = kf.em(df['Sales'])
3 kf.initial_state_mean = df['Sales'].mean()
4 (smoothed_state_means, smoothed_state_covariances) = kf.smooth(df['Sales'])
5 df['KF'] = smoothed_state_means
```




Questions?
sarairis.garcia@gmail.com
@montjoile
github.com/montjoile/pycon
hk