TIME SERIES 501

Lesson 3: Smoothing Time Series

Learning Objectives

You will be able to do the following:

- Explain the need for data smoothing.
- List common data-smoothing techniques.
- Explain how common data-smoothing techniques work.
- Use Python* to smooth time-series data.

$x = 12, \quad 13, \quad 15$

weights: 1,5 — 2 — 1

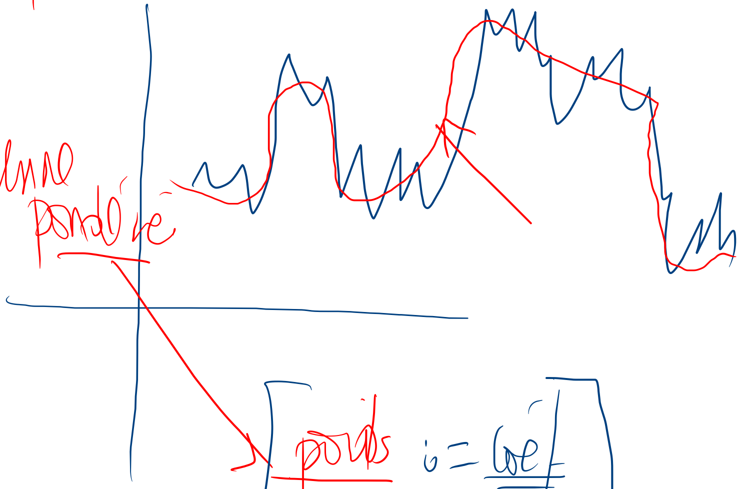
$\rightarrow \frac{12+13+15}{3}$

What Is Smoothing?

$\bar{x} = \frac{12 \times 1.5 + 13 \times 2 + 15 \times 1}{1,5 + 2 + 1}$

Smoothing is a process that allows you to extract useful patterns from data. There are many ways to smooth data. In this lesson, you'll learn about the following:

- Simple average smoothing — moyenne
- Equally weighted moving average — moyenne pondérée
- Exponentially weighted moving average



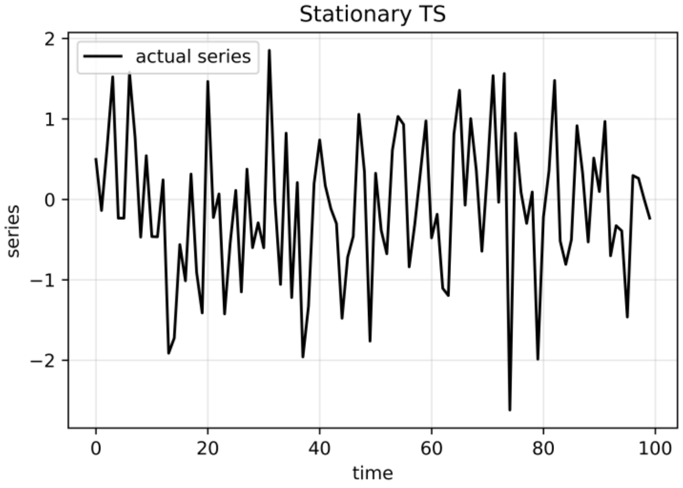
poids i = coef

t	1	2		n
X_t	x	x		

Why Is Smoothing Important?

Smoothing is one important tool that allows you to make future-looking forecasts.

- Consider the stationary data to the right.
- How would you go about predicting what's going to happen one, two, or more steps into the future?



Forecasting with Simple Average

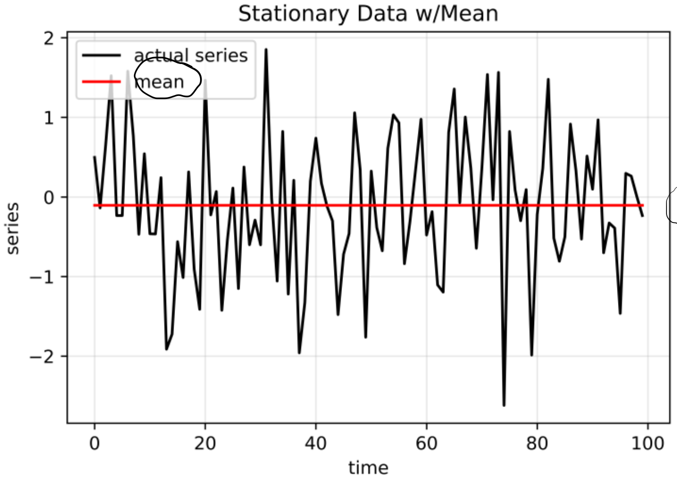
An obvious solution is to calculate the mean of the series and predict that value into the future.

- Looks quite reasonable in this case.
- However, we should be more rigorous and calculate how far off our estimate is from reality.
- Next is a quick detour about mean squared error (MSE).

Mean squared Error

La moyenne des erreurs quadratiques

métrique : → précision



Mean Squared Error $\boxed{\text{(MSE)}}$

minimise l'erreur ⇒ [bon modèl]

Mean squared error is a metric commonly employed to quantitatively measure the efficacy of an estimate.

data réelle

- The formula $\boxed{MSE} = \frac{1}{n}\sum_{i=0}^{n}(observed_i - estimate_i)^2$ → [Model | $\hat{d}$]
- Let's walk through a simple example…

	i	1	2	…	n
observées	d	d_1	d_2		d_n
Model	$\hat{d}$	$\hat{d}_1$	$\hat{d}_2$		$\hat{d}_n$
	$\sum(d_i-\hat{d}_i)^2$	$(d_1-\hat{d}_1)^2$			$(d_n-\hat{d}_n)^2$

$MSE = \frac{\#}{n}$

MSE Example

Say we have a sensor that took the following readings:

1.5	2.0	2.5	3.0	3.5	4.0

MSE Example

Also say we created a model called **Model A** with the following estimates:

1.5	2.0	2.5	3.0	3.5	4.0	measurements
1.3	2.2	2.5	3.1	3.9	4.6	estimates

y predict

MSE Example

Then MSE is calculated like so:

Squared error

$SE = (1.5 – 1.3)^2 + (2.0 – 2.2)^2 + (2.5 – 2.5)^2 + (3.0 – 3.1)^2 + (3.5 – 3.9)^2 + (4.0 – 4.6)^2$

$SE = (0.2)^2 + (-0.2)^2 + (0)^2 + (0.1)^2 + (-0.4)^2 + (-0.6)^2$

$SE = 0.61$

Mean squared error

$MSE = 0.61 / 6 = \boxed{0.102}$

MSE Comparison

The nice thing about using a metric like MSE is that we can compare different models or estimates to see which is doing the best job.

- What if you built another model called **Model B** that created an array of estimates that looked like this: [1.5, 2.1, 2.5, 3.1, 3.6, 4.6]?
- Is **Model A** or **Model B** better?

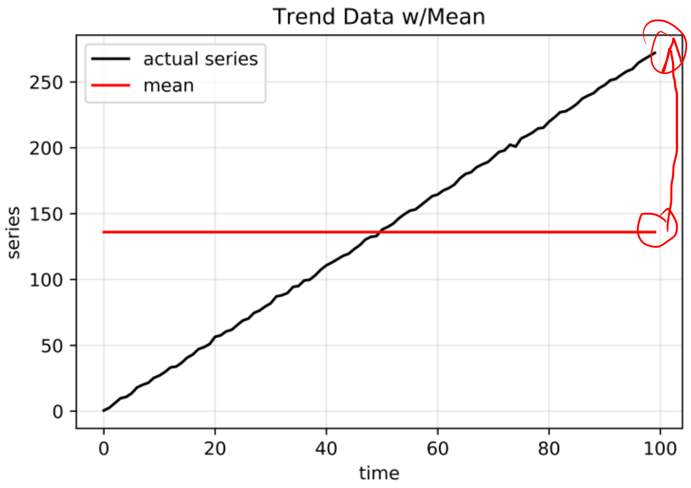
X_t

Forecasting Trend with Simple Average

tendencia

What if there's a trend?

- Obviously, this technique is not going to work well.
- What's a better approach?



[moyenne Mobile]

MOVING AVERAGE

Moving Average

There is another technique called moving average that has greater sensitivity towards local changes in the data.

- Moving average comes in two flavors:
  - Equally weighted ; poids identique = 1 [même coef = 1]
  - Exponentially weighted moyenne pondérée par des coefs (exponentielle)

Equally Weighted Moving Average – Example

lag

$\boxed{k=6} \rightarrow \bar{x}$

Say we have the same sensor readings as before:

1.5	2.0	2.5	3.0	3.5	4.0

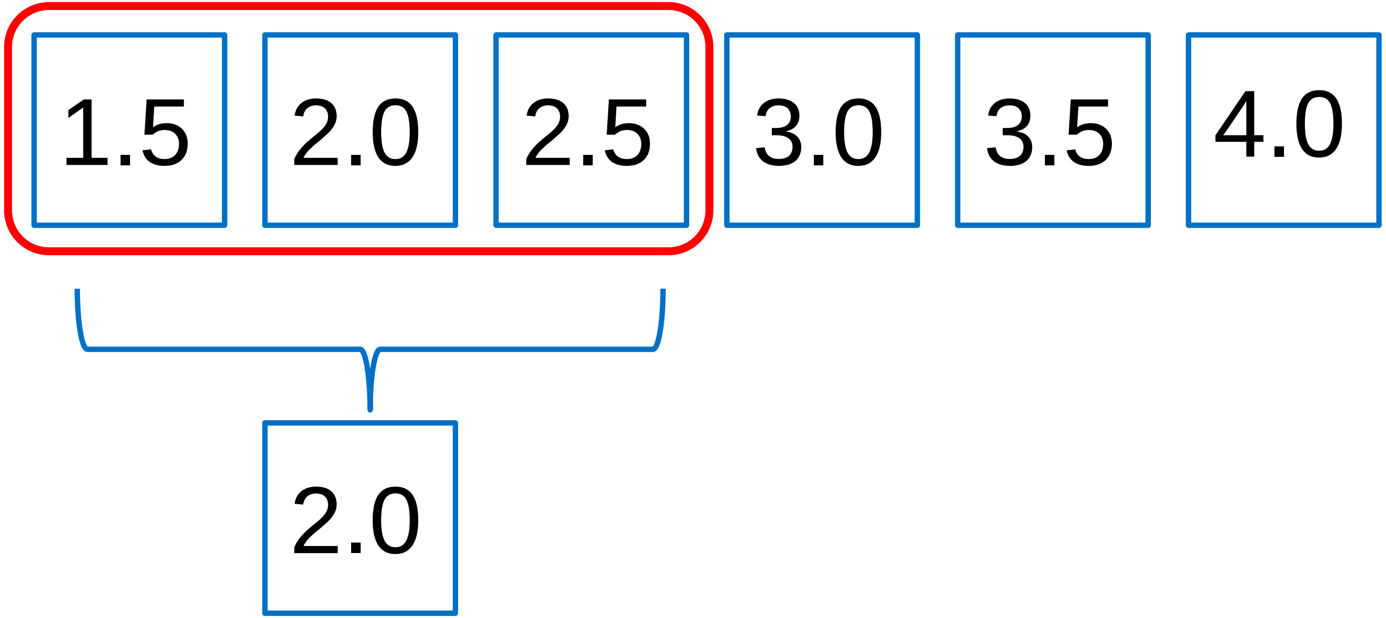
Equally Weighted Moving Average – Example

The first step in calculating moving average is to select a window size (we’ll use 3).

1.5	2.0	2.5	3.0	3.5	4.0

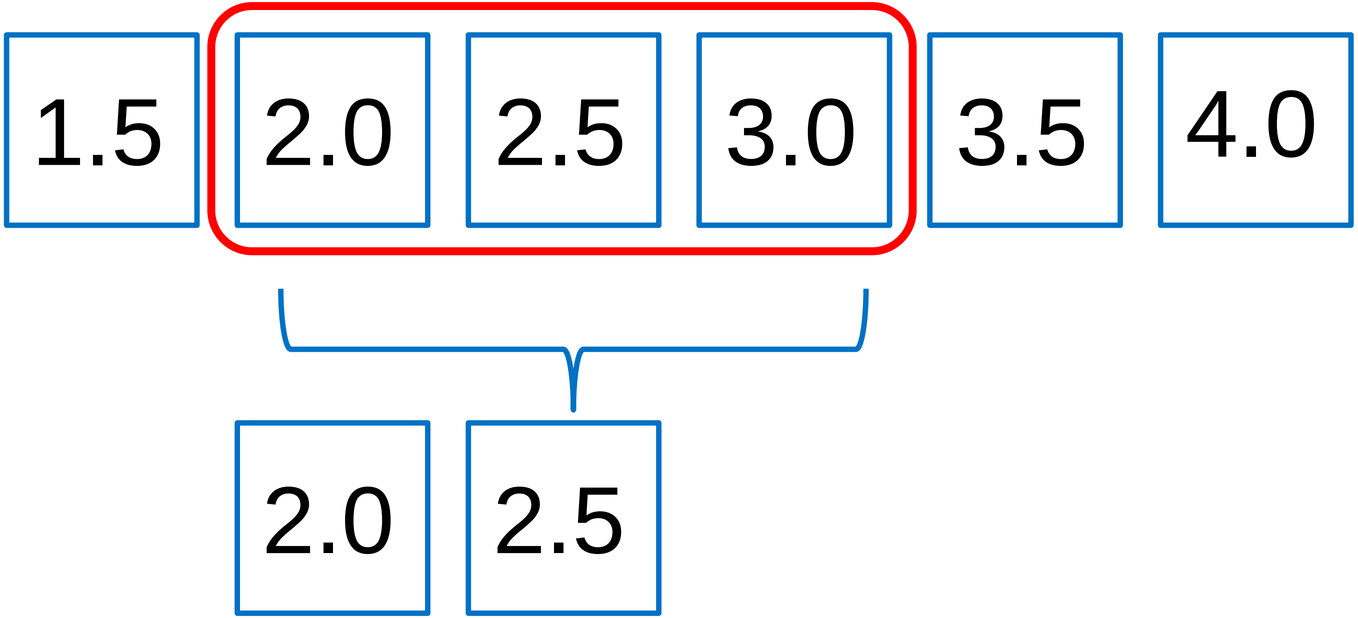
Equally Weighted Moving Average – Example

We slide the window over the first 3 values and calculate the mean within.



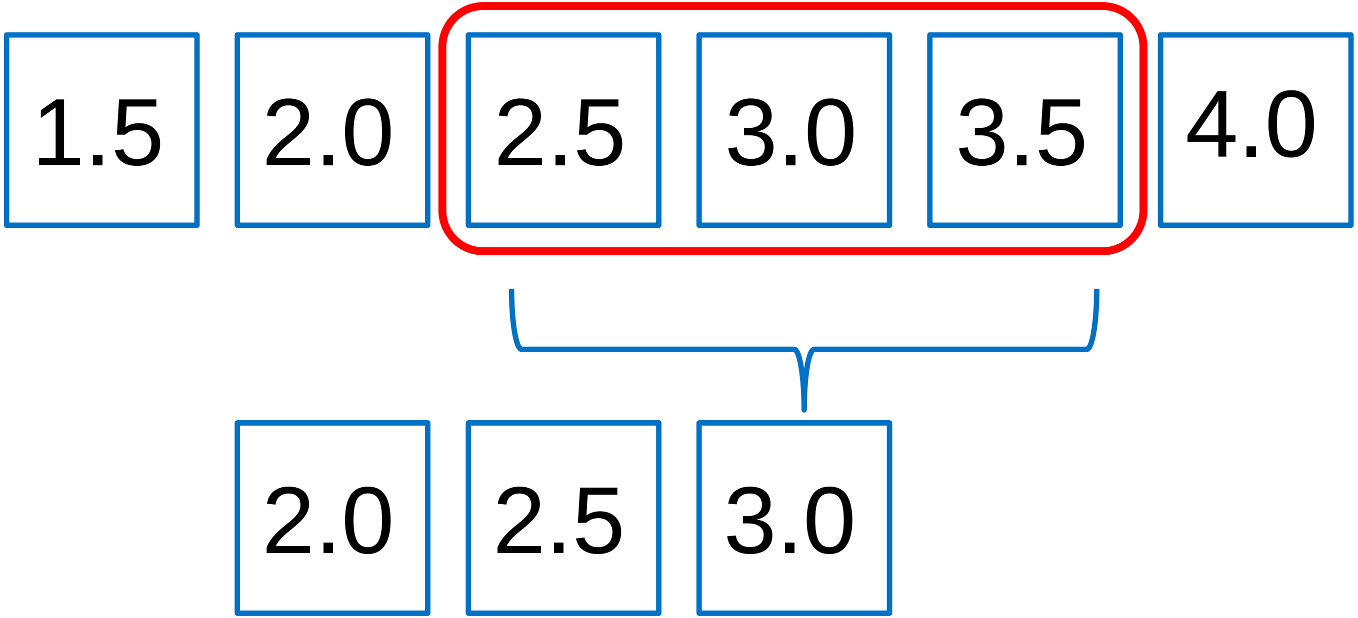
Equally Weighted Moving Average – Example

We slide the window over one place and calculate the new mean.



Equally Weighted Moving Average – Example

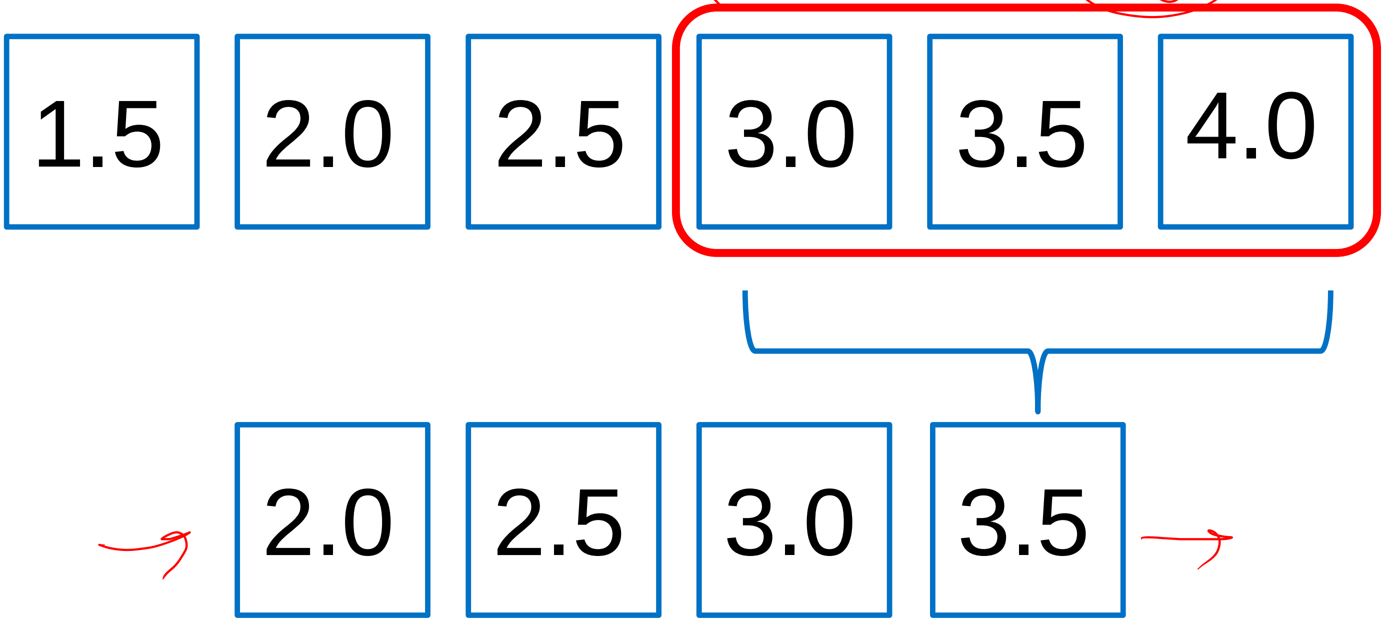
We continue this process until we reach the end.



Equally Weighted Moving Average – Example

lag

We continue this process until we reach the end.

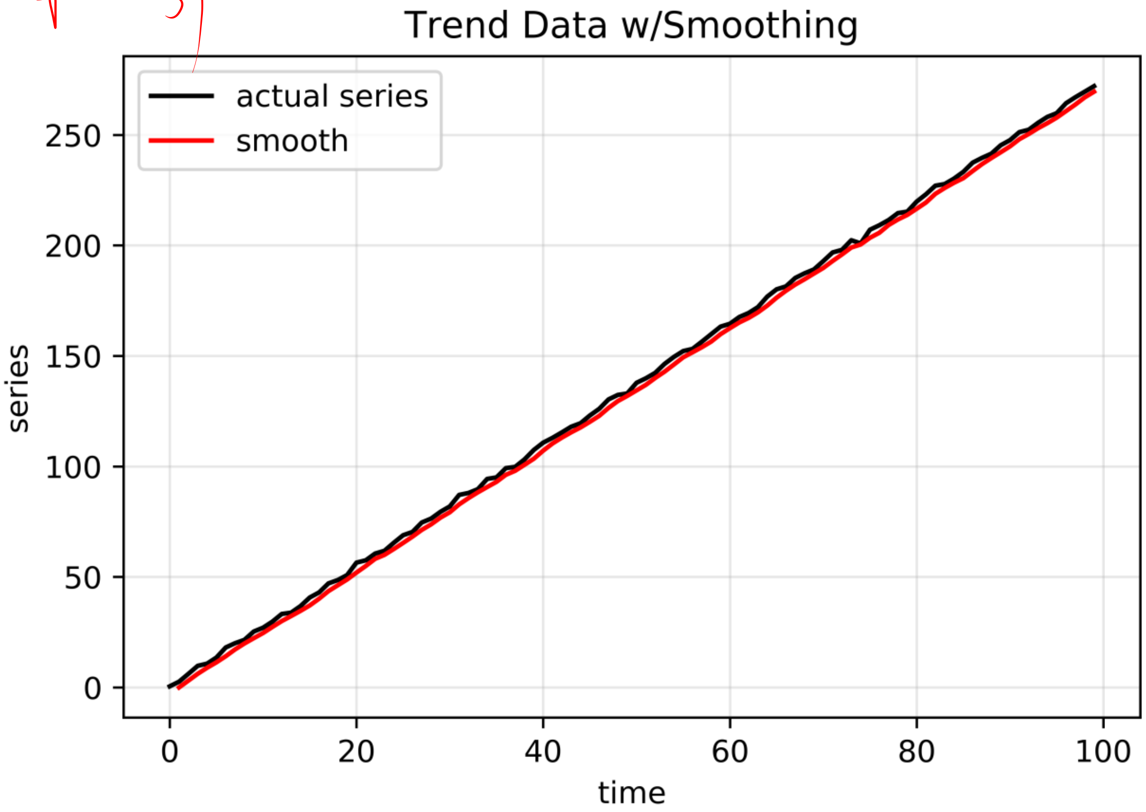


Equally Weighted Moving Average (C3)

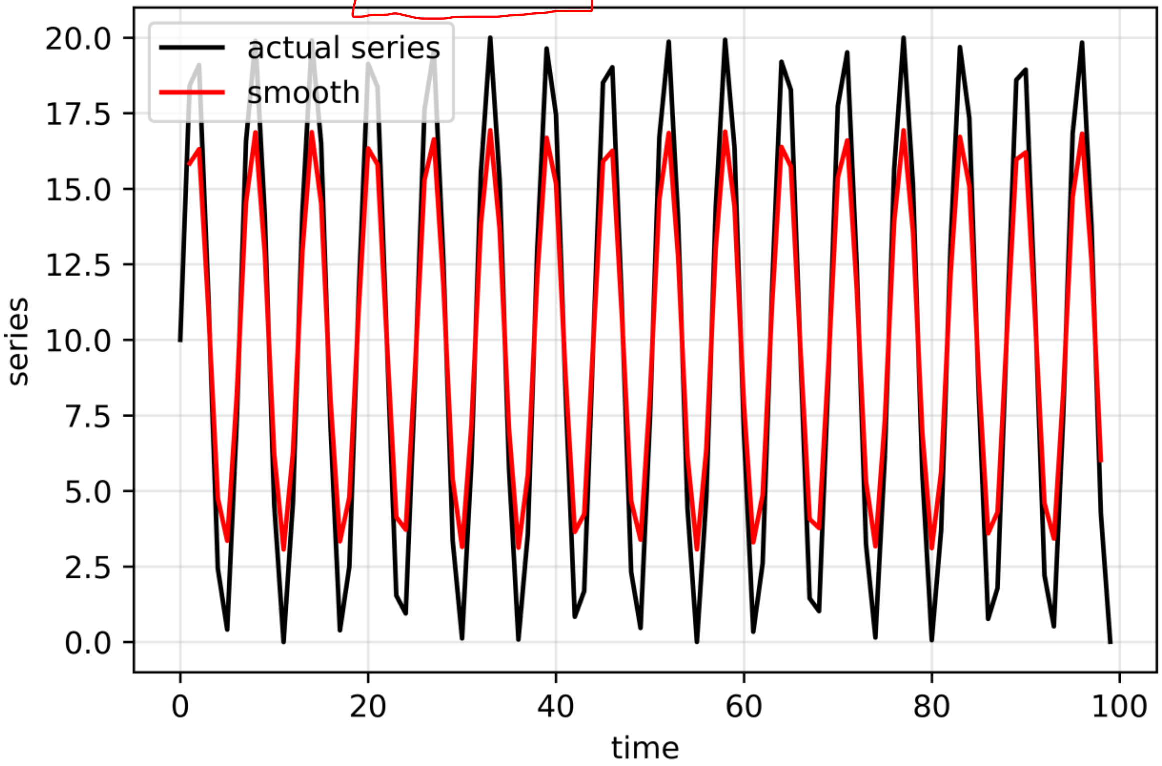
Let's apply this equally weighted moving average technique to three datasets:

- One with trend [tendance]
- One with seasonality saisonnier
- One with trend and seasonality les deux

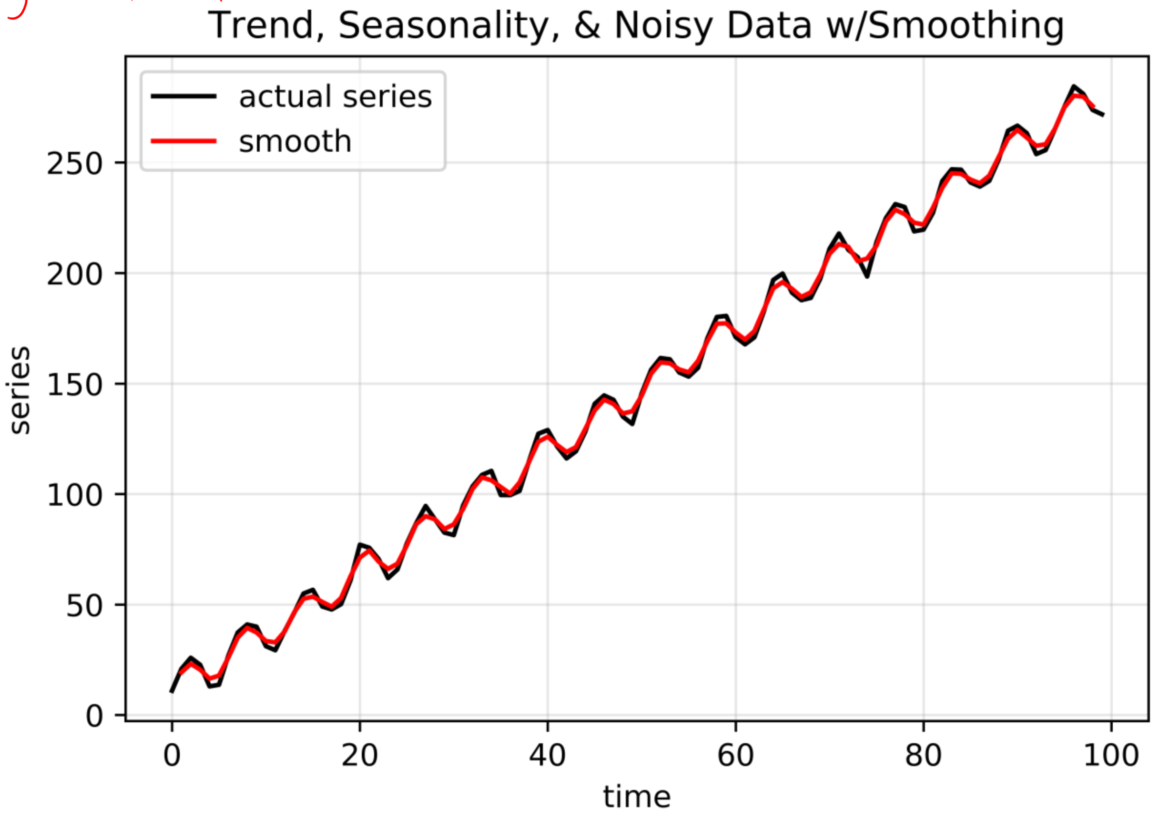
(Với T_p c_3)


Trend Data w/Smoothing
actual series
smooth
series
250
200
150
100
50
0
time
0
20
40
60
80
100

Seasonality Data w/Smoothing



[C3] → à voir

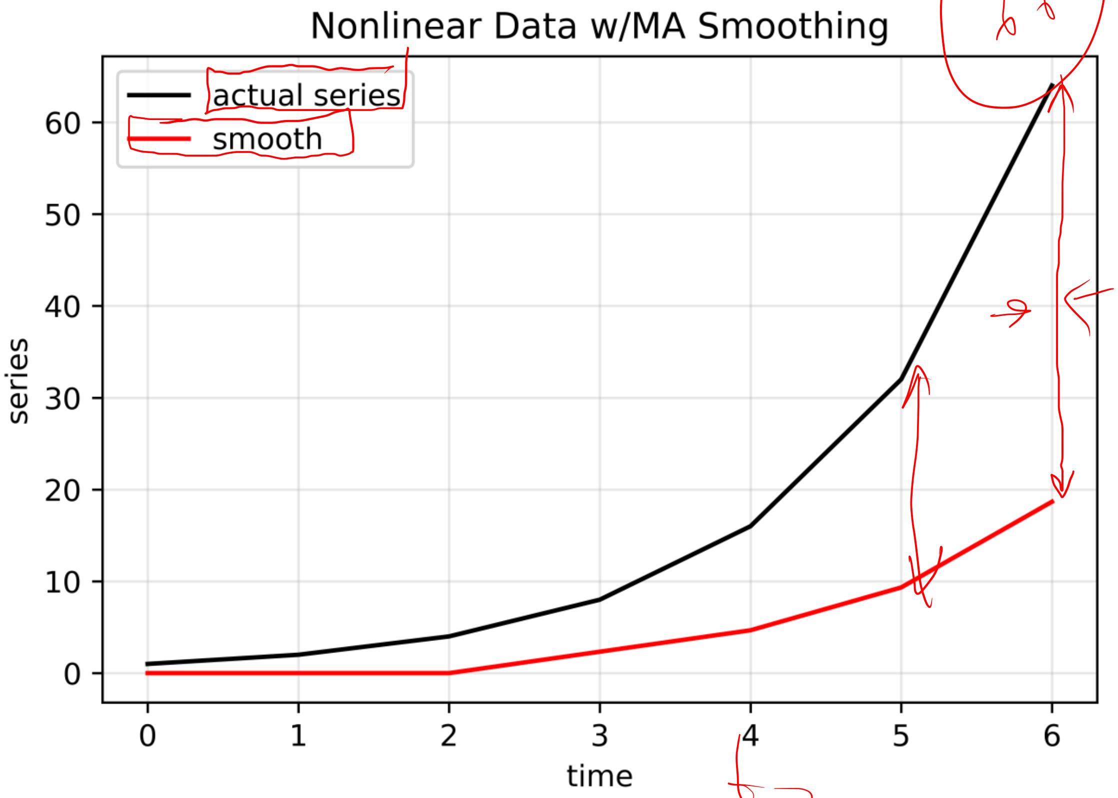

Trend, Seasonality, & Noisy Data w/Smoothing
actual series
smooth
series
0
50
100
150
200
250
time
0
20
40
60
80
100

Equally Weighted Moving Average – Recap

We saw in all three cases that this simple moving average technique extracted the key patterns within the data.

- A few questions should come to mind:
  - How well does this method do from a forecasting perspective?
  - Is equal weighting the best weighting scheme?

prediction


Nonlinear Data w/MA Smoothing
actual series
smooth
series
time
0
10
20
30
40
50
60
0
1
2
3
4
5
6

Equally Weighted Moving Average – Issues

This technique clearly lags the trend. That becomes a bigger problem as the trend becomes more aggressive.

- Now is the time to explore another weighting scheme to see if we can do better.
- Next up is exponentially weighted moving average (sometimes known as single exponential smoothing).

pros (use)

exponential

Exponentially Weighted Moving Average – Example

moyenne mobile

Say we have the same sensor readings as before:

1.5	2.0	2.5	3.0	3.5	4.0

Exponentially Weighted Moving Average – Example

The first step is the same - select a window size (we'll use 3 again).

1.5	2.0	2.5	3.0	3.5	4.0

Exponentially Weighted Moving Average – Example

We slide the window over the first 3 values and calculate the mean but differently.

1.5	2.0	2.5	3.0	3.5	4.0

Instead of applying equal weights to all 3 observations, let's apply exponential weights.

Exponential Weights

There are many ways to create exponential weights. To keep things simple, we'll leverage this simple formula:

$$w + w^2 + w^3 = 1$$

$$w = w_{t-1} \sim 0.543$$

$$w^2 = w_{t-2} \sim 0.294$$

$$w^3 = w_{t-3} \sim 0.160$$

Exponentially Weighted Moving Average – Example

We slide the window over the first 3 values and calculate the mean but differently.

1.5	2.0	2.5	3.0	3.5	4.0

$$(\boxed{w_{t-3}} \times 1.5) + (\boxed{w_{t-2}} \times 2.0) + (\boxed{w_{t-1}} \times 2.5) = \boxed{\mathbf{2.2}} \approx \boxed{w_t}$$

0.16 0,294 0,543

Exponentially Weighted Moving Average – Example

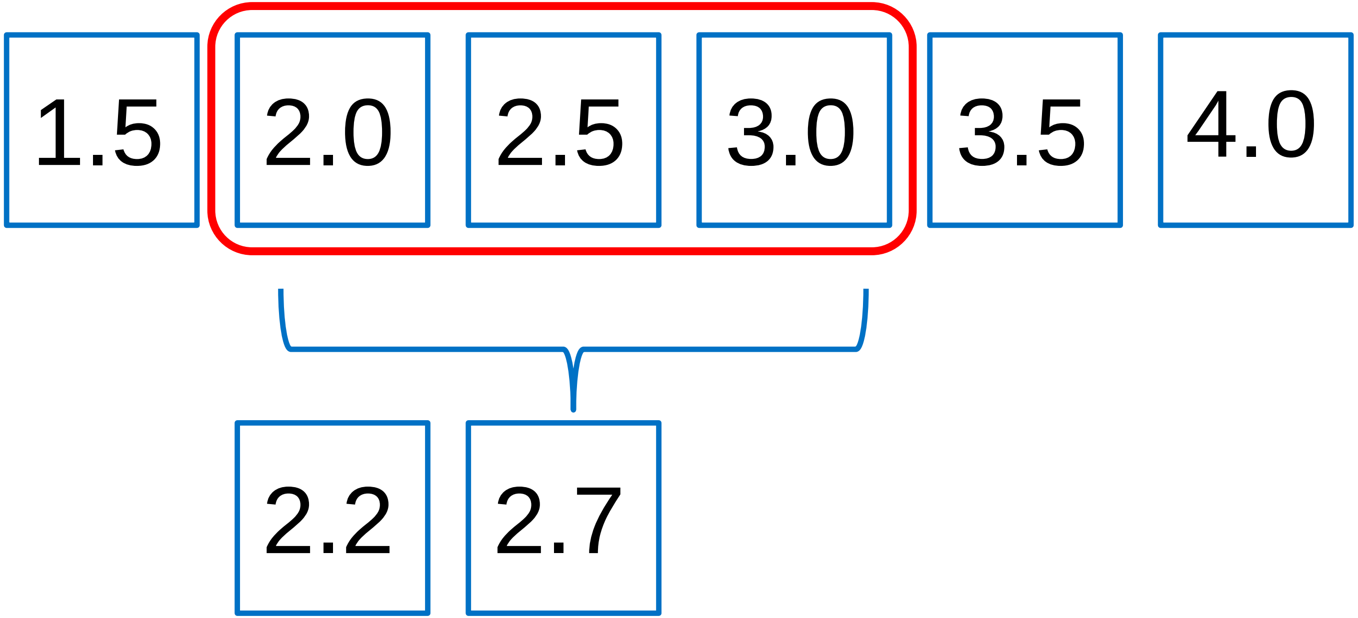
We slide the window over the first 3 values and calculate the mean but differently.

1.5	2.0	2.5	3.0	3.5	4.0

$\underbrace{1.5 \quad 2.0 \quad 2.5}_{2.2}$

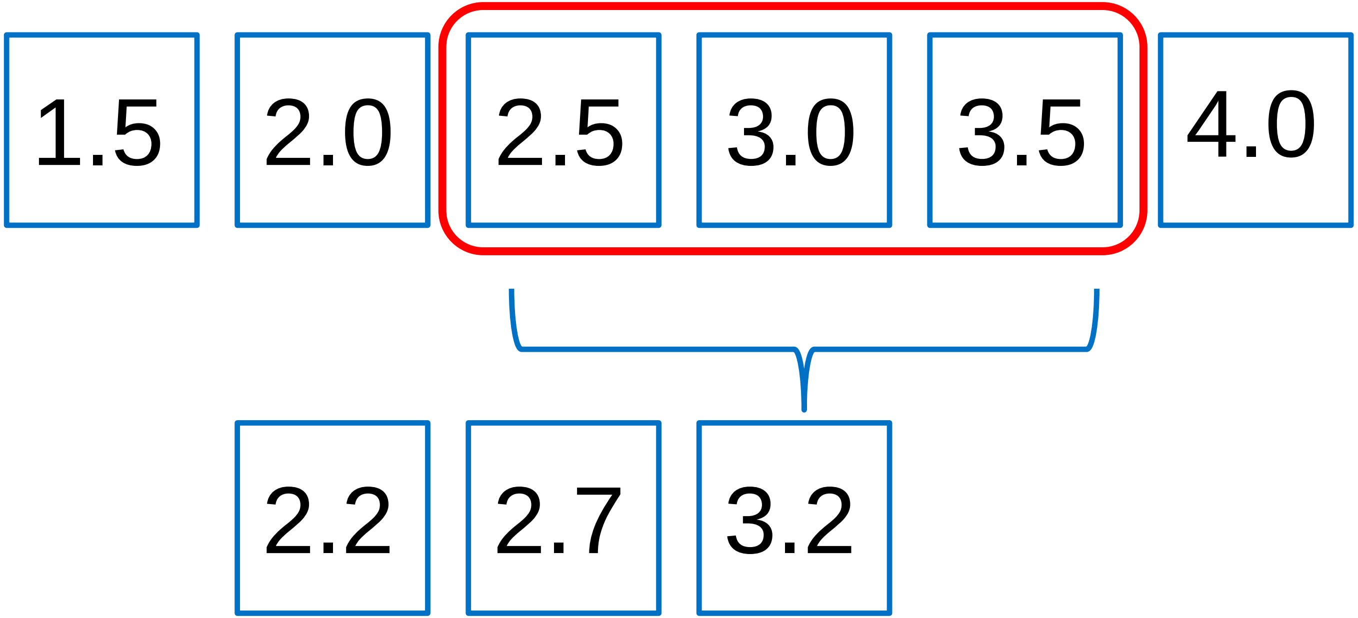
Exponentially Weighted Moving Average – Example

We slide the window over one place and calculate the new mean.



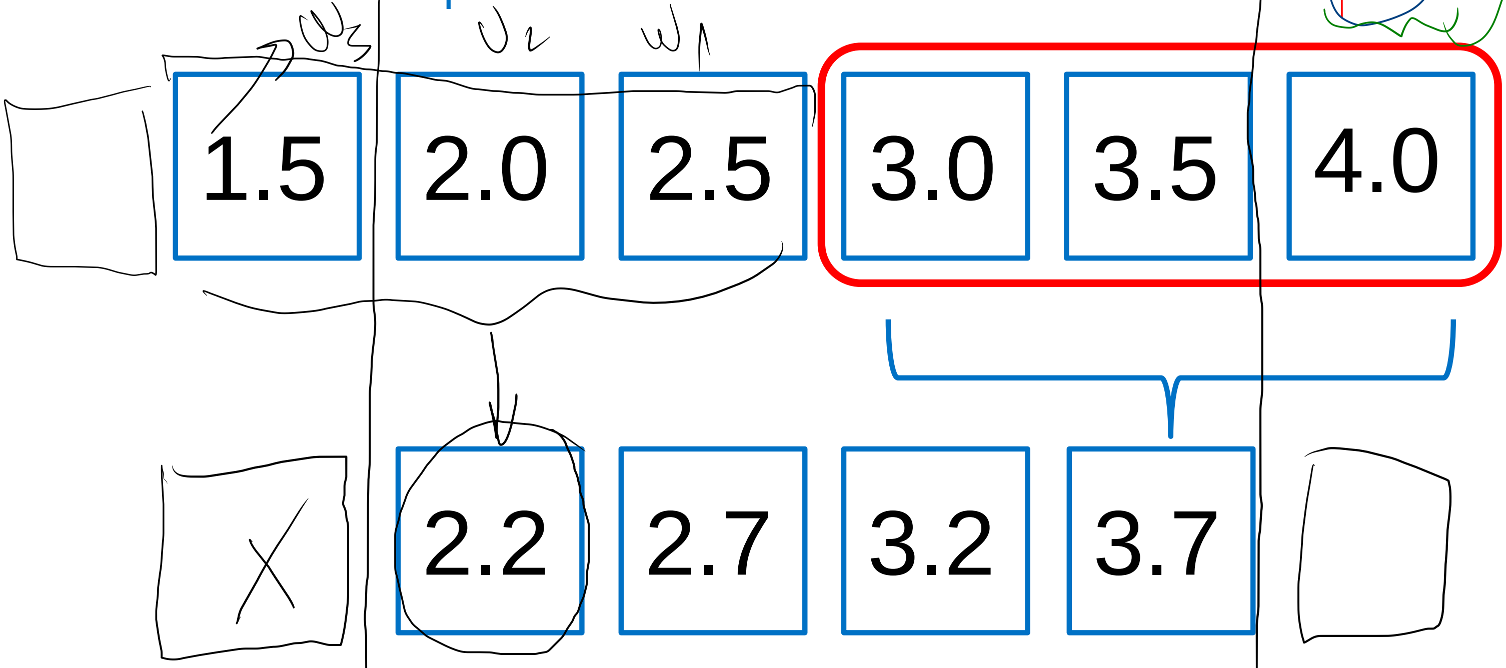
Exponentially Weighted Moving Average – Example

We continue this process until we reach the end.



Exponentially Weighted Moving Average – Example

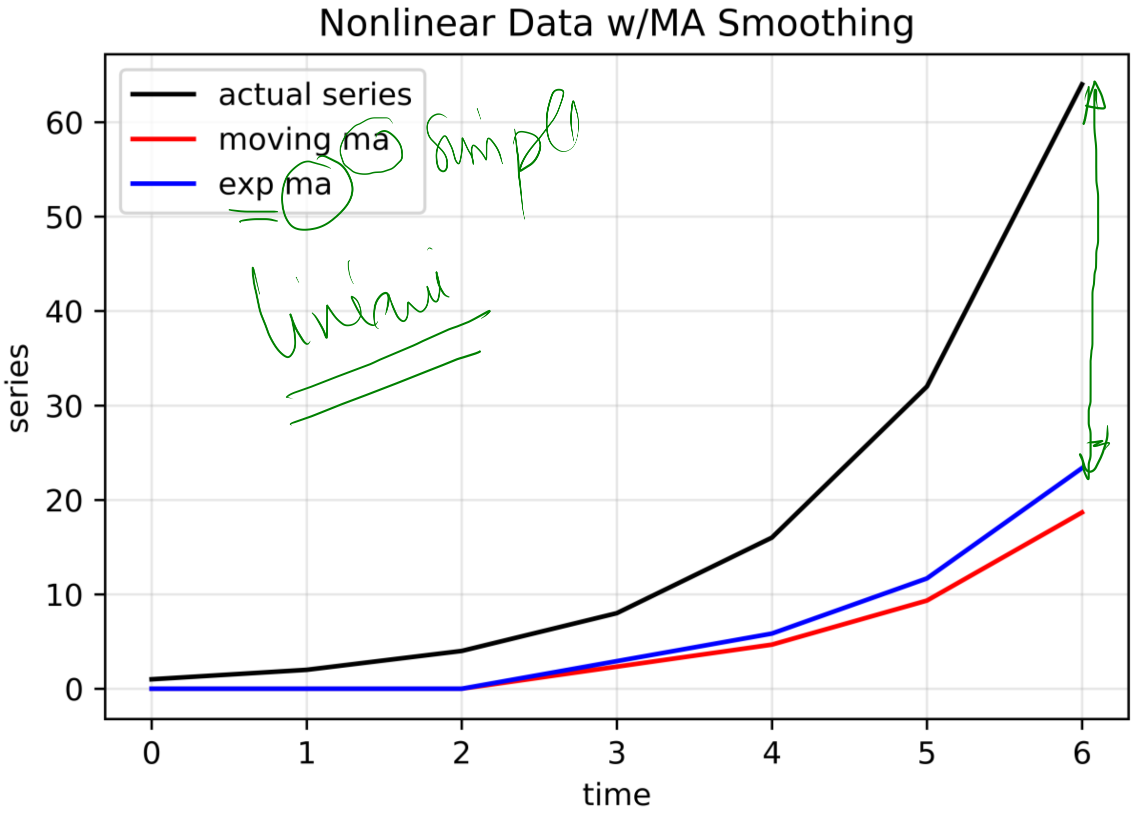
We continue this process until we reach the end.



Exponentially Weighted Moving Average – Recap

Exponentially weighted moving average works by smoothing the series as a whole.

- Now that you know how it works, a few questions should come to mind:
  - Do you think this method will do a better job forecasting than equally weighted moving average?
  - Is exponentially weighted smoothing sufficient for forecasting in general?


Nonlinear Data w/MA Smoothing
actual series
moving ma
exp ma
simple
linear
series
time
0
10
20
30
40
50
60
0
1
2
3
4
5
6

Exponentially Weighted Moving Average – Issues

Comparing exponentially weighted moving average to equally weighted moving average:

- Exponential is more sensitive to local changes.
- However, it still lags significantly.
- Therefore, we need to explore more complex forecasting mechanisms that leverage smoothing.

ADVANCED SMOOTHING

Single Exponential Formulation

$[\alpha, (1-\alpha), (1-\alpha)^2, \ldots\ldots, (\ )]$

What we have been examining so far is exponential weighted average smoothing. This is also known as single exponential smoothing, and has this formula: $\approx S_t$

$$
\begin{aligned}
\hat{S_t} = \alpha * X_t &+ (1-\alpha) * \hat{S_{t-1}} \\
&+ (1-\alpha)^2 * \hat{S_{t-2}} \\
&+ \ldots
\end{aligned}
$$

$$t \geq 3$$

meilleur approximation

where:

$\hat{S_t}$ = Smoothed value at time t

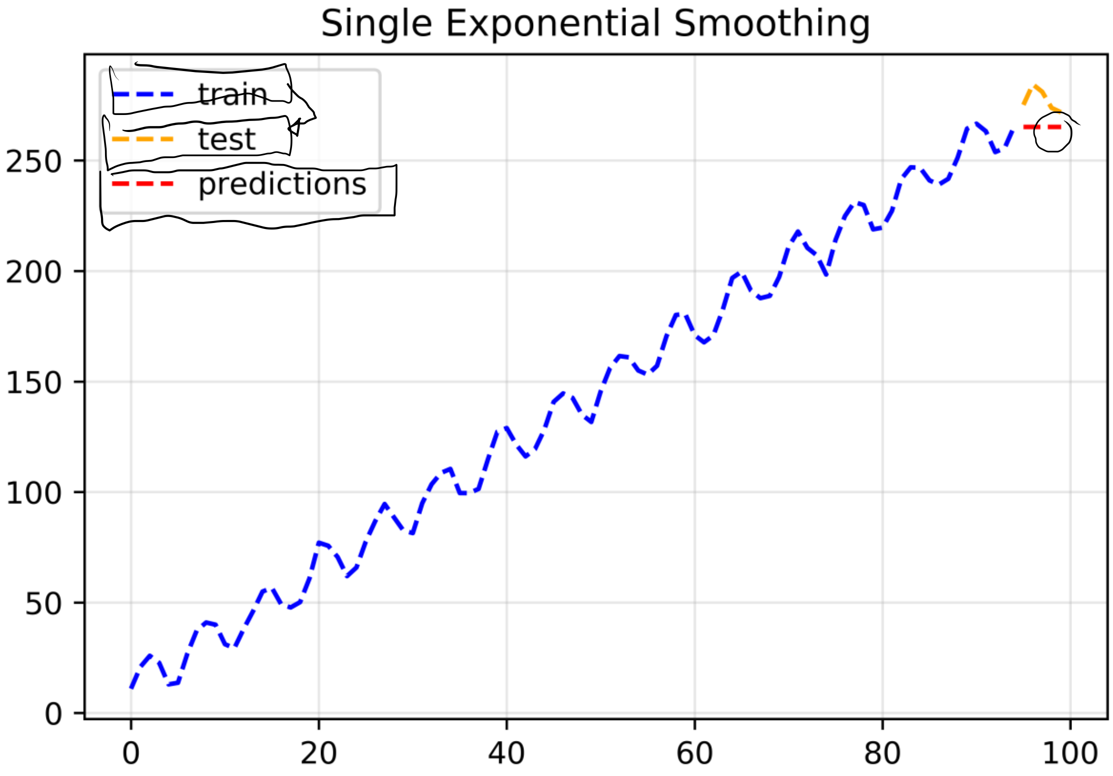
X_t = Actual value at time t

α = Parameter optimized to fit past data

From Single Exponential to Advanced Smoothing

Let's revisit the data containing trend and seasonality.

- Specifically, let's chop off the last 5 observations and treat them as a test set.
- We'll begin by applying single exponential smoothing to the training set and forecasting forward 5 observations.
- We will then compare the forecast with actual observations using the MSE metric discussed earlier.


Single Exponential Smoothing
train
test
predictions
250
200
150
100
50
0
0
20
40
60
80
100

MSE = 830

The Need for Advanced Smoothing Techniques

Single exponential smoothing produces the same value pushed out over the forecast horizon.

- Clearly, it is picking up neither trend nor seasonality.
- Therefore, we turn to double exponential smoothing.

Double Exponential Smoothing - Recap

Double exponential smoothing has the ability to pick up trend. It does this by adding a second component into its formulation that smooths out trend.

$$\boxed{S_t} = \alpha * X_t + (1 - \alpha) * (S_{t-1} + b_{t-1})$$

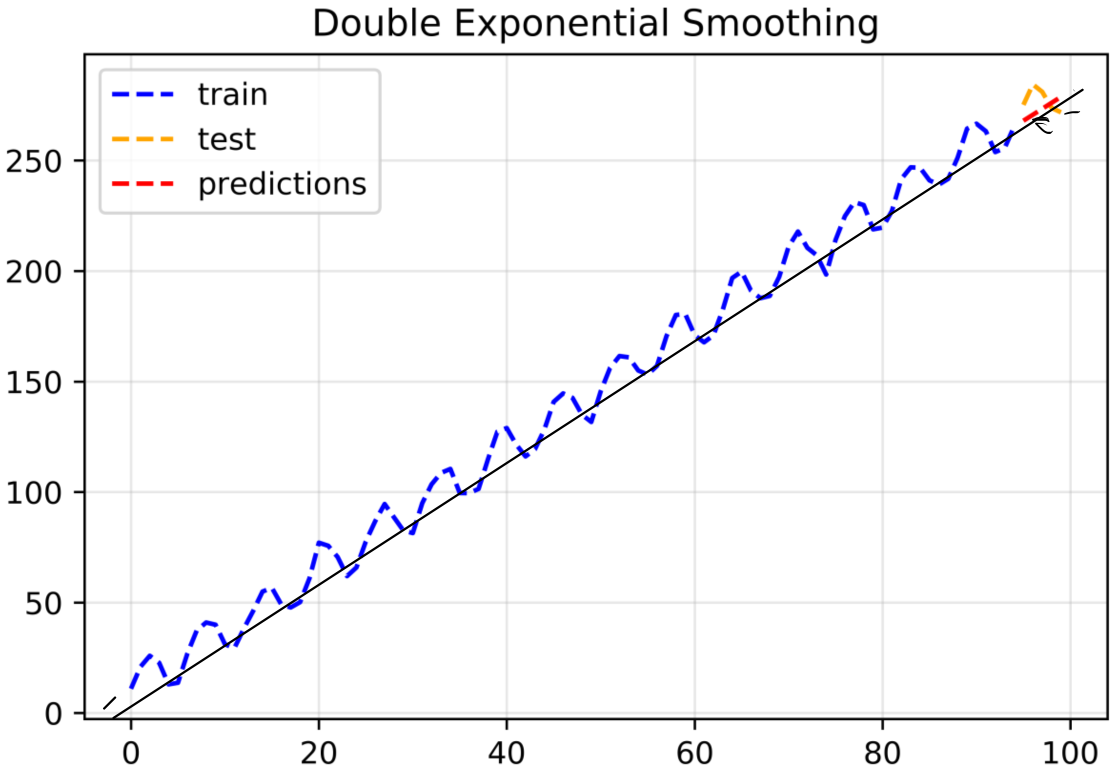
} Smooths the <u>value</u> of the series

$$\boxed{b_t} = \boxed{\beta} * (S_t - S_{t-1}) + (1 - \beta) * b_{t-1}$$

} Smooths the <u>trend</u> of the series

$$\hat{X}_{t+1} = S_t + b_t$$

} Future prediction of series = sum of value and trend


Double Exponential Smoothing
train
test
predictions
0
50
100
150
200
250
0
20
40
60
80
100

MSE = 354

Double Exponential Smoothing - Recap

Double exponential smoothing has the ability to pick up trend.

- This is a step in the right direction.
- However, did you notice that it fails to pickup seasonality?
- For that we need triple exponential smoothing.

Triple Exponential Smoothing

Triple exponential smoothing has the ability to pickup trend and seasonality. It does this by adding a third component to its formulation that smooths out seasonality.

$$S_t = \alpha * (X_t - c_{t-L}) + (1 - \alpha) * (S_{t-1} + b_{t-1}) \} \text{ Smooths the value of the series}$$

$$b_t = \beta * (S_t - S_{t-1}) + (1 - \beta) * b_{t-1} \} \text{ Smooths the trend of the series}$$

$$c_t = \gamma * (X_t - S_{t-1} - b_{t-1}) + (1 - \gamma) * c_{t-L} \} \text{ Smooths the seasonality of the series}$$

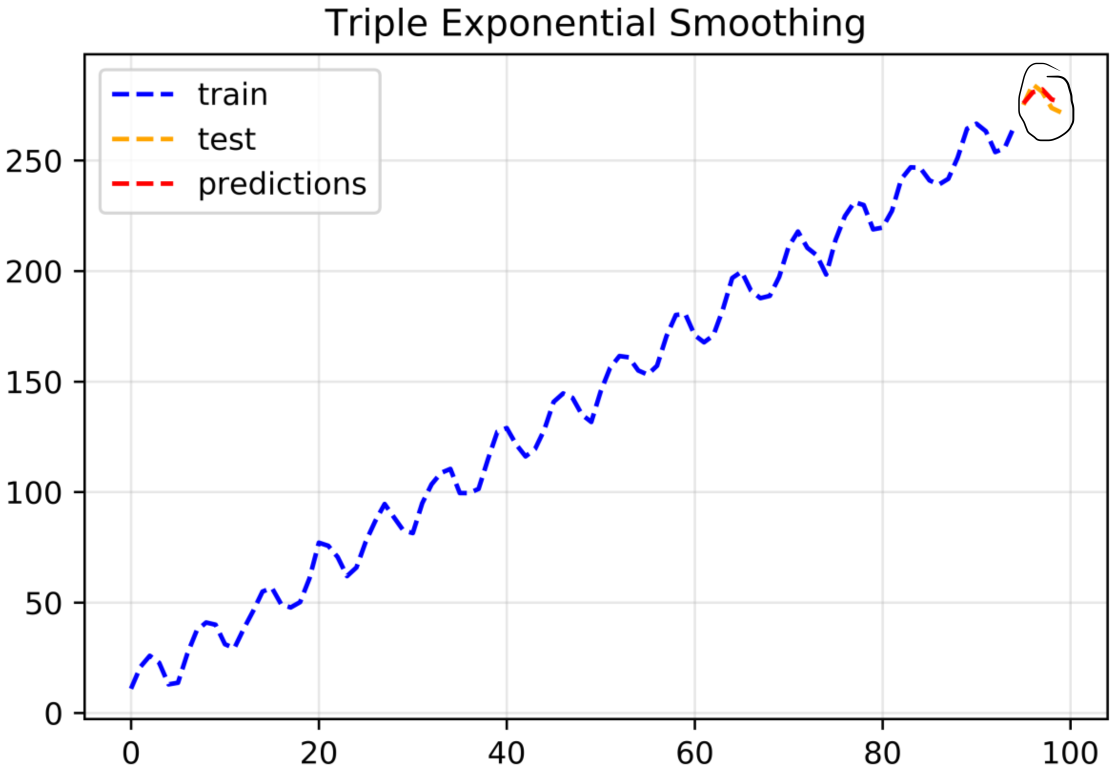
Triple Exponential Smoothing (cont.)

Triple exponential smoothing has the ability to pickup trend and seasonality. It does this by adding a third component to its formulation that smooths out seasonality.

$$(\hat{X})_{t+m} = (S_t + m * b_t) * c_{t-L+(m-1)} \boxed{\mod L}$$

$\boxed{L}$ = Length of time of the seasonality

period L


Triple Exponential Smoothing
train
test
predictions
0
50
100
150
200
250
0
20
40
60
80
100

MSE = 50

Single vs. Double vs. Triple Exponential Smoothing

A comparison of MSE shows just how significant an impact using the best modeling strategy has on a forecast.

	MSE
Single Exponential	830
Double Exponential	354
Triple Exponential	50

Exponential Smoothing - Recap

Here's how to know whether to use single, double, or triple exponential smoothing:

- Does your data lack a trend?
  - Use single exponential smoothing.
- Does your data have trend but no seasonality?
  - Use double exponential smoothing.
- Does your data have trend and seasonality?
  - Use triple exponential smoothing.

APPLICATIONS IN PYTHON

Use Python to Smooth Time-Series Data

Next up is a look at applying these concepts in Python.

- See notebook entitled *Introduction_to_Smoothing_student.ipynb*

Learning Objectives Recap

In this session you learned the following:

- Why smoothing can be useful in time series
- Common data-smoothing techniques
- How common data-smoothing techniques work
- How to use Python to smooth time-series data

Legal Notices and Disclaimers

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

Sample source code is released under the Intel Sample Source Code License Agreement.

intel®