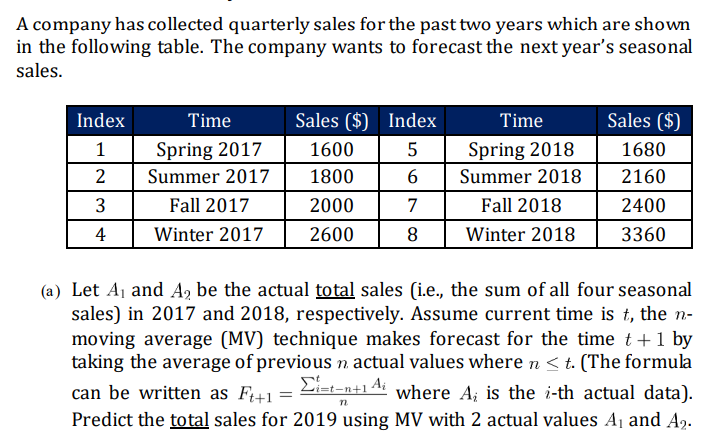
**MKTG5883 Exercise 5: Time Series Analysis**

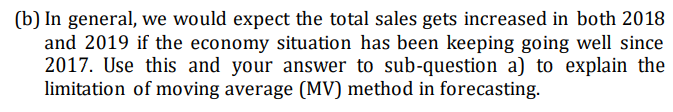
1. **Exercise 1 – Seasonality**

****

**Given Data:**

* Total Sales for 2017:
  + Spring: 1600
  + Summer: 1800
  + Fall: 2000
  + Winter: 2600
  + Total for 2017 = 1600 + 1800 + 2000 + 2600 = 8000
* Total Sales for 2018:
  + Spring: 1680
  + Summer: 2160
  + Fall: 2400
  + Winter: 3360
  + Total for 2018 = 1680 + 2160 + 2400 + 3360 = 9600
* **Moving Average Calculation:**

Using the 2-year moving average:

****

The moving average (MV) method is a popular forecasting technique, but it has several important limitations, especially in the context of seasonal sales.

1. **Sluggish Reaction:**
   * Data: Total sales in 2017 were 8000 and in 2018 were 9600.
   * **Forecast for 2019 using MV:**
   * Although sales increased from 8000 to 9600, the MV method only predicts 8800 for 2019. This shows that MV does not capture the growth in sales promptly, leading to inaccurate forecasts.
2. **Ignores Seasonality:**
   * The company's sales have clear seasonal variations (spring, summer, fall, winter).
   * For example, winter sales were 2600 in 2017 and 3360 in 2018, indicating a seasonal increase. However, the MV method does not account for these fluctuations. It simply averages total sales without considering seasonal factors, resulting in inaccurate predictions for each season.

**Explain:**

\* **Lagging Effect:** Since the MV method takes the average of past values, it reacts slowly to changes in the data

\* **Smoothing Out Important Variations:** it removes important information about trends and sudden growth that might be crucial for accurate forecasting.

\* The MV method gives equal importance to all past data points used in the average => treats 2017 and 2018 as equally relevant



Spring:

Summer:

Autumn:

Winter:

A text on a white background

AI-generated content may be incorrect.

**Step i: Seasonal Indexes**

| **Season** | **2017 Sales** | **2017 Index (÷ 2000)** | **2018 Sales** | **2018 Index (÷ 2400)** |
| --- | --- | --- | --- | --- |
| Spring | 1600 | 1600 / 2000 = 0.80 | 1680 | 1680 / 2400 = 0.70 |
| Summer | 1800 | 0.90 | 2160 | 0.90 |
| Fall | 2000 | 1.00 | 2400 | 1.00 |
| Winter | 2600 | 1.30 | 3360 | 1.40 |

**Step ii: Average Indexes**

| **Season** | **Avg Index** |
| --- | --- |
| Spring | (0.80 + 0.70) / 2 = 0.75 |
| Summer | 0.90 |
| Fall | 1.00 |
| Winter | (1.30 + 1.40) / 2 = 1.35 |

**Step iii: Forecast for 2019**

* Average sales for 2019 = From (a): **8800 / 4 = 2200**

**Step iv: Multiply by Indexes**

| **Season** | **Forecasted Sales 2019** |
| --- | --- |
| **Spring** | **2200 × 0.75 = 1650** |
| **Summer** | **2200 × 0.90 = 1980** |
| **Fall** | **2200 × 1.00 = 2200** |
| **Winter** | **2200 × 1.35 = 2970** |

* Check Total: 1650 + 1980 + 2200 + 2970 = 8800

A graph with a line and a red arrow

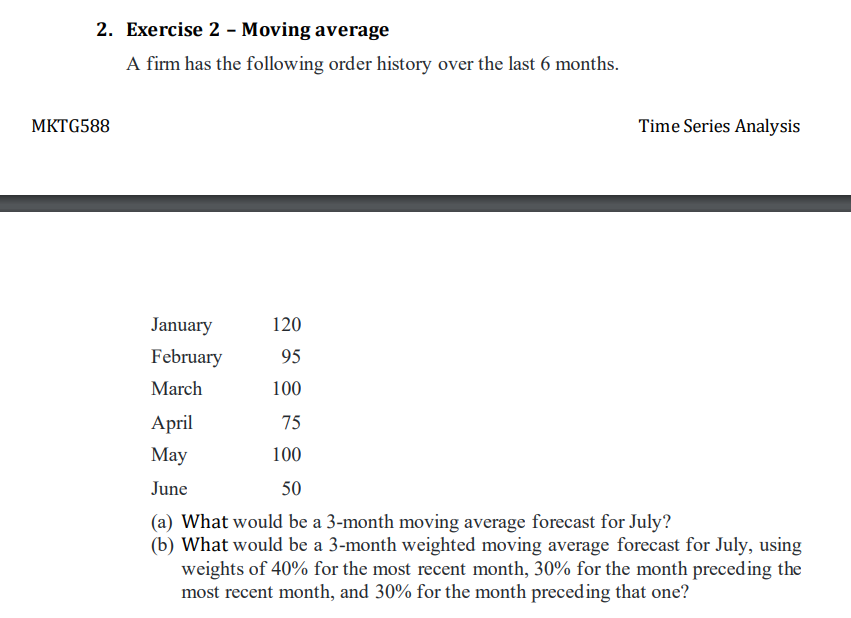
AI-generated content may be incorrect.

A graph of different colored bars

AI-generated content may be incorrect.

A graph with different colored bars

AI-generated content may be incorrect.



**(a) 3-month Moving Average Forecast for July**

Use Apr, May, and Jun:

= (75 + 100 + 50) / 3 = **75**

**(b) 3-month Weighted Moving Average Forecast for July**

Weights:

* June (50) × 0.40 = 20
* May (100) × 0.30 = 30
* April (75) × 0.30 = 22.5

Forecast = 20 + 30 + 22.5 = **72.5**

A graph with blue and red lines

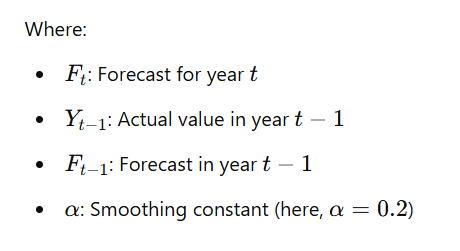
AI-generated content may be incorrect.

A black and white text on a white background

AI-generated content may be incorrect.

The question asks for a forecast *in* 2010 based on simple exponential smoothing (SES). This typically means forecasting the value *for* 2010 using data available up to the end of 2009.

SES formula: F(t+1) = α \* A(t) + (1 - α) \* F(t)



* **Issue:** The smoothing constant α is not given. Simple Exponential Smoothing requires α. Without it, we cannot calculate a specific forecast. We also need an initial forecast (F1 or F2) to start the process.
* **Assumption:** Let's assume a common value for demonstration, α = 0.3. Let's also assume the initial forecast for 2002 was the actual value for 2001 (a common naive initialization): F(2002) = A(2001) = 82.

Given that the data is from 2001 to 2010, we will take F, as the price in 2001. Let's assume

**Initial Forecast:**

F1=Y0=82(Actual value of 2001)

1. **Year 2002** (Y₁ = 80):

F1=0.2⋅82+0.8⋅82=82.00

1. **Year 2003** (Y₂ = 76):

F2=0.2⋅80+0.8⋅82=81.60

1. **Year 2004** (Y₃ = 73):

F3=0.2⋅76+0.8⋅81.60=80.48

1. **Year 2005** (Y₄ = 72):

F4=0.2⋅73+0.8⋅80.48=79.18

1. **Year 2006** (Y₅ = 73):

F5=0.2⋅72+0.8⋅79.18=77.74

1. **Year 2007** (Y₆ = 72):

F6=0.2⋅73+0.8⋅77.74=76.59

1. **Year 2008** (Y₇ = 73):

F7=0.2⋅72+0.8⋅76.59=75.67

**Year 2009** (Y₈ = 77):

F8=0.2⋅73+0.8⋅75.67=75.13

1. **Year 2010** (Y₉ = 74):

F9=0.2⋅77+0.8⋅75.13=75.50

**Mean Absolute Deviation (MAD):**

To evaluate the accuracy of the forecasting model, we use the **Mean Absolute Deviation (MAD)**:

A math equation with black text

AI-generated content may be incorrect.

**The answer is ~ 3.7**

**Model Selection Viewpoint:**

We observed that each forecasted value is computed using the actual and forecasted values from the previous year. Therefore, **Simple Exponential Smoothing** is a useful technique for time series data that **does not exhibit clear trends or seasonal patterns**.

In this case, by examining the distribution of rubber prices from **2001 to 2010**, we can conclude that there is **no strong trend or seasonality** in the data. The values fluctuate around a certain level but do not consistently increase or decrease.

As a result, **Simple Exponential Smoothing is an appropriate and reliable model** for forecasting this dataset.

A graph with a line and a line

AI-generated content may be incorrect.

import pandas as pd

import matplotlib.pyplot as plt

# Given data

years = list(range(2001, 2011))

actual = [82, 80, 76, 73, 72, 73, 72, 73, 77, 74]

# Smoothing constant

alpha = 0.2

# Initialize forecast list

forecast = [actual[0]]  # F1 = Y0 = 82

# Calculate forecasts

for t in range(1, len(actual)):

    ft = alpha \* actual[t - 1] + (1 - alpha) \* forecast[-1]

    forecast.append(round(ft, 2))

# Create a DataFrame

df = pd.DataFrame({

    'Year': years,

    'Actual': actual,

    'Forecast': forecast

})

# Plotting

plt.figure(figsize=(10, 5))

plt.plot(df['Year'], df['Actual'], marker='o', label='Actual')

plt.plot(df['Year'], df['Forecast'], marker='o', linestyle='--', label=f'Forecast (α={alpha})')

plt.title('Simple Exponential Smoothing (α=0.2)')

plt.xlabel('Year')

plt.ylabel('Price')

plt.grid(True)

plt.legend()

plt.tight\_layout()

plt.show()