Mohammad Idrees Bhat

Tech Skills Trainer | AI/ML Consultant

Introduction to Time Series Analysis and Forecasting

Basics of time series analysis and decomposition

AGENDA

1. Introduction to Time Series
2. Time Series Decomposition

Objectives

- Understand the basics of time series analysis.
- Learn the concept of time series decomposition.
- Apply ARIMA (AutoRegressive Integrated Moving Average) and Exponential Smoothing for forecasting.

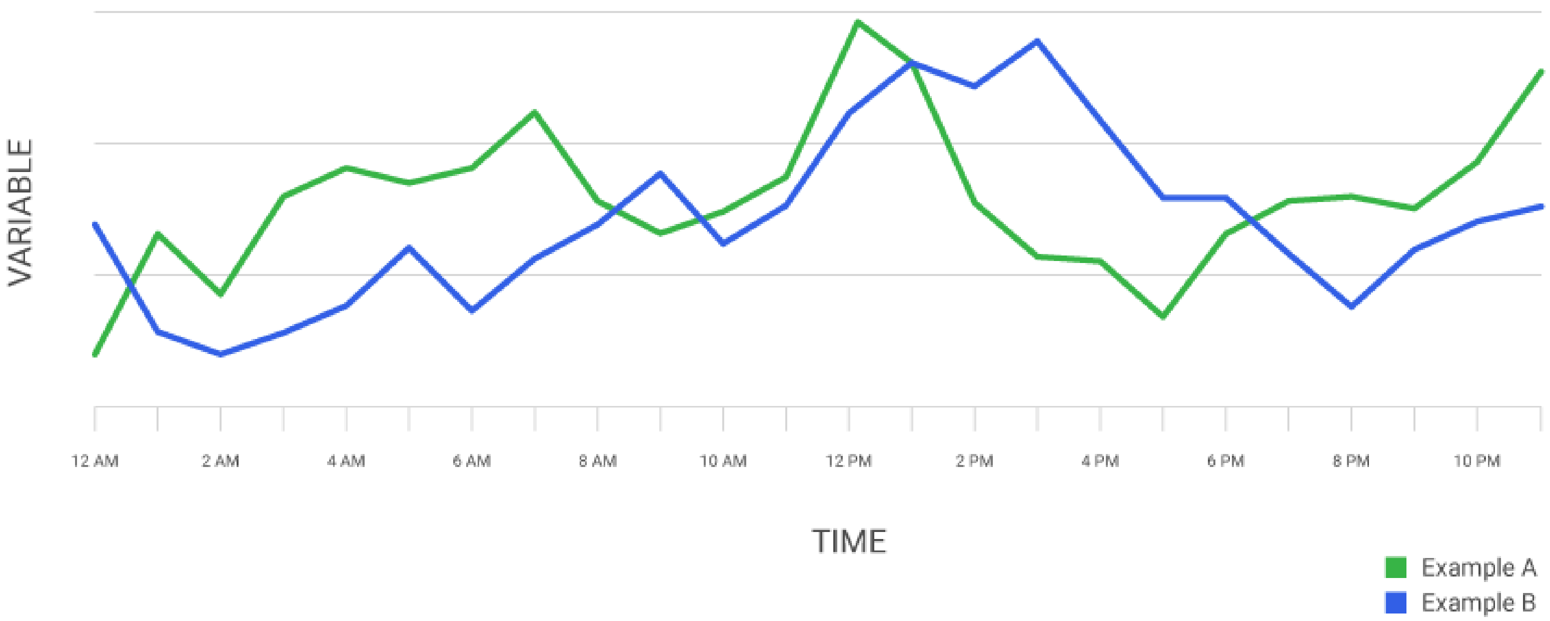
If you were a superhero, what would your superpower be?

1. Introduction to Time Series

What is Time Series Data?

Time series is a sequence of data points collected or recorded at successive points in time, usually at uniform intervals.

Unlike random data, time series data is ordered in time, making it useful for observing patterns and trends over time.



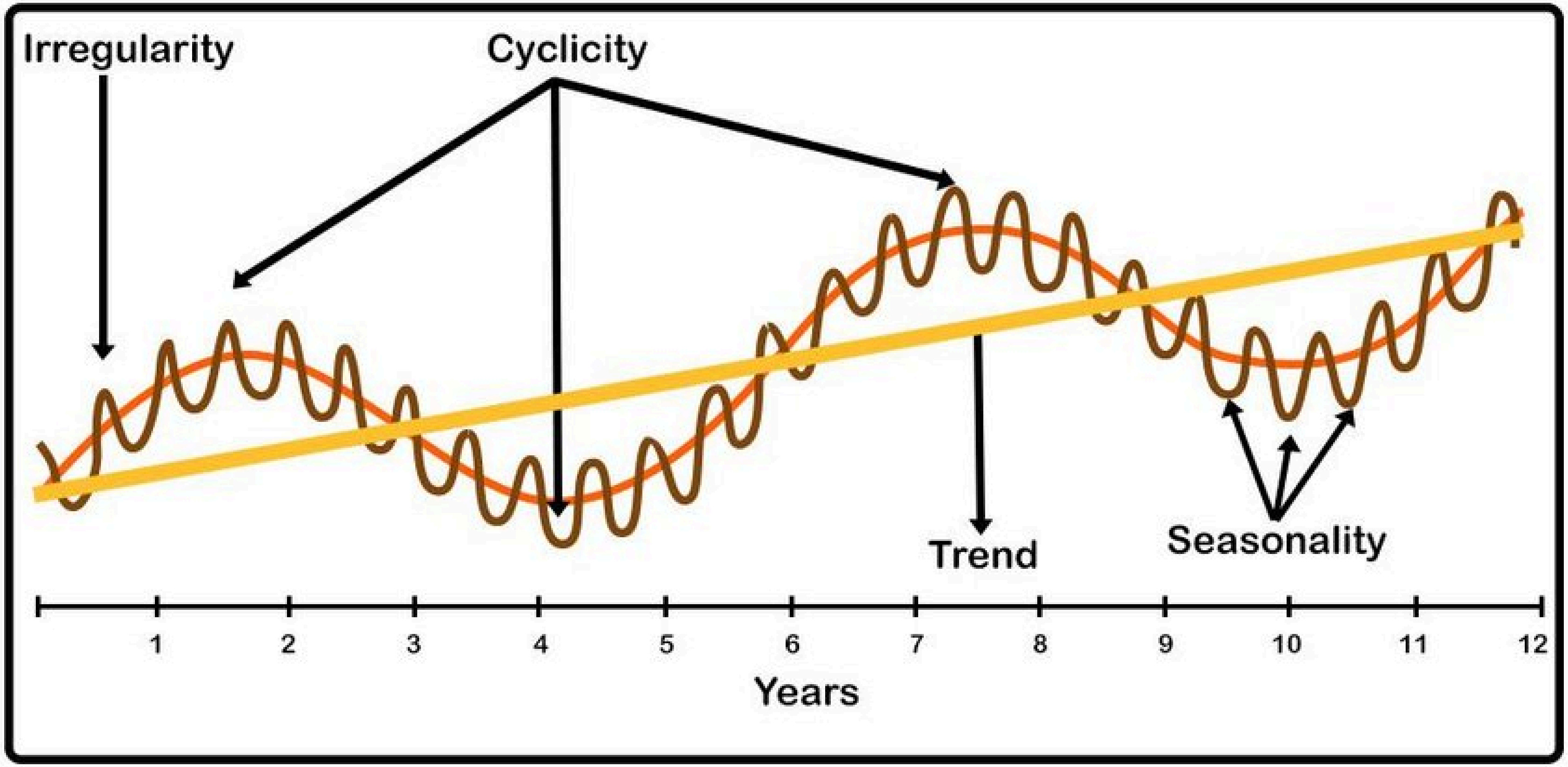
2. **Some Characteristics of Time Series:**

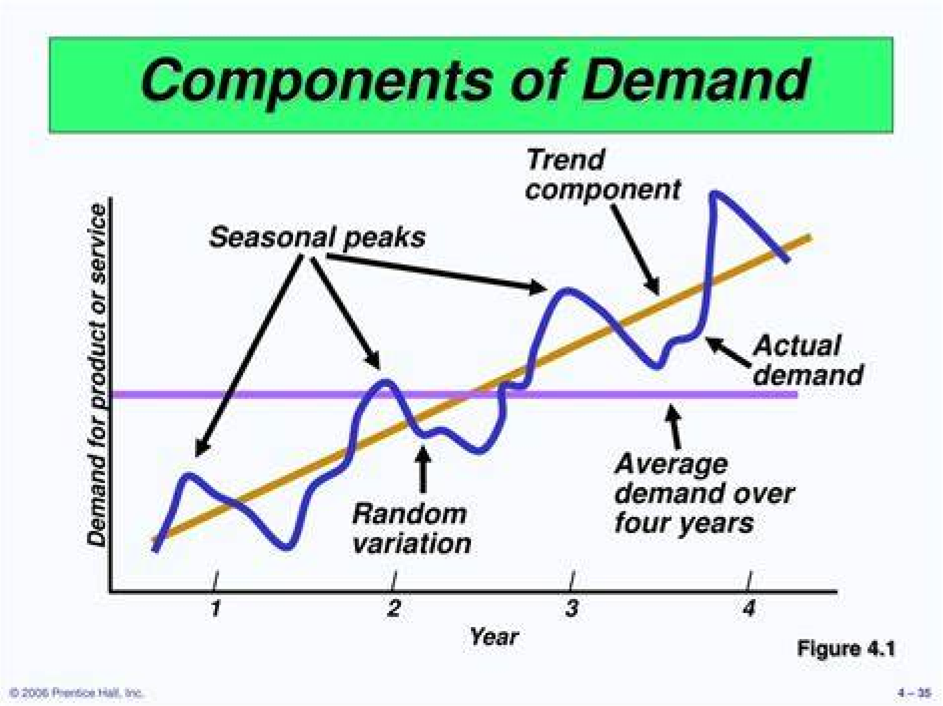
   - **Time-Indexed Data:** Each **data point** is associated with a **specific time**.
   - **Temporal Dependency:** Past data can influence future values.

Examples of Time Series Data:

- **Stock Prices:** Daily closing prices of a company's stock over a year.
  Example: `[120.5, 121.0, 122.3, 123.7, 124.9, ...]` (collected daily).
- **Weather Records:** Hourly temperature readings over a week.
  Example: `[25°C, 26°C, 27°C, 28°C, 27°C, ...]` (collected hourly).
- **Sales Data:** Monthly revenue of a retail store over several years.
  Example: `[5000, 5200, 4800, 5100, ...]` (collected monthly).
- **Electricity Consumption:** Daily electricity usage in kilowatt-hours.
  Example: `[320, 340, 310, 330, ...]` (collected daily).

Components of a Time Series

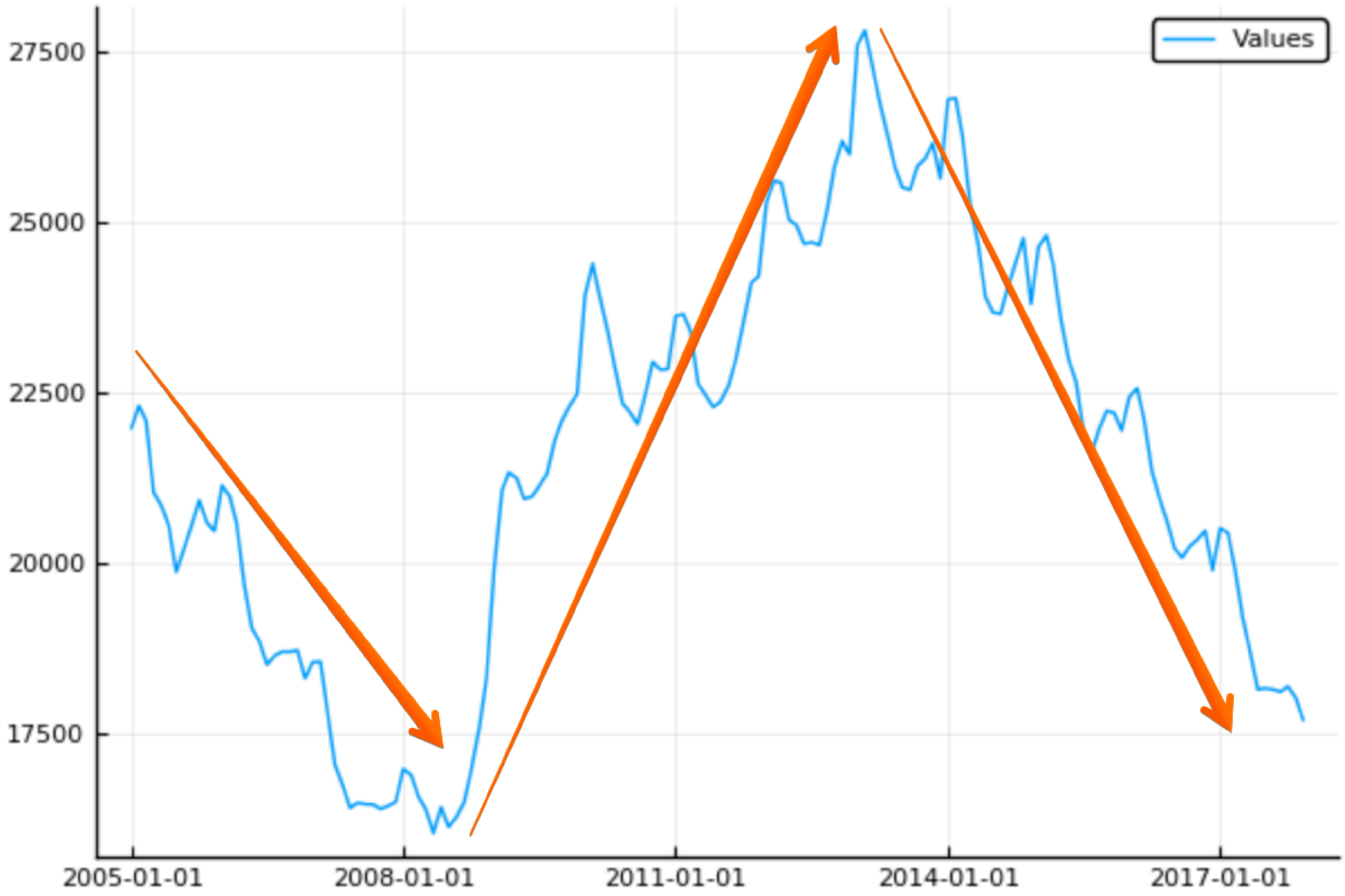

Irregularity
Cyclicity
Trend
Seasonality
1
2
3
4
5
6
7
8
9
10
11
12
Years



Components of Demand

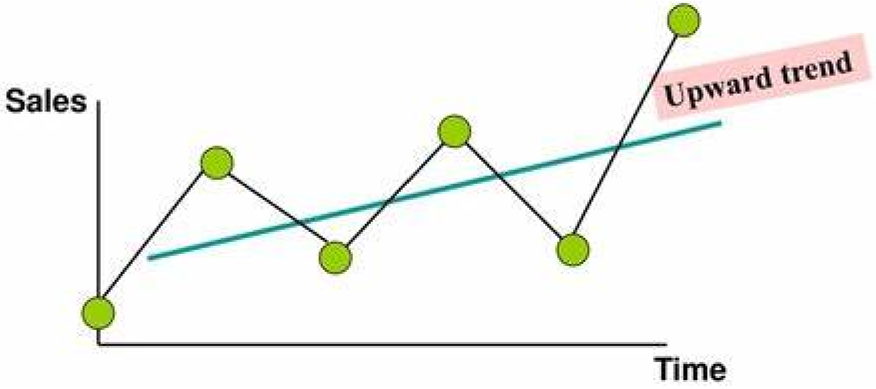
Figure 4.1

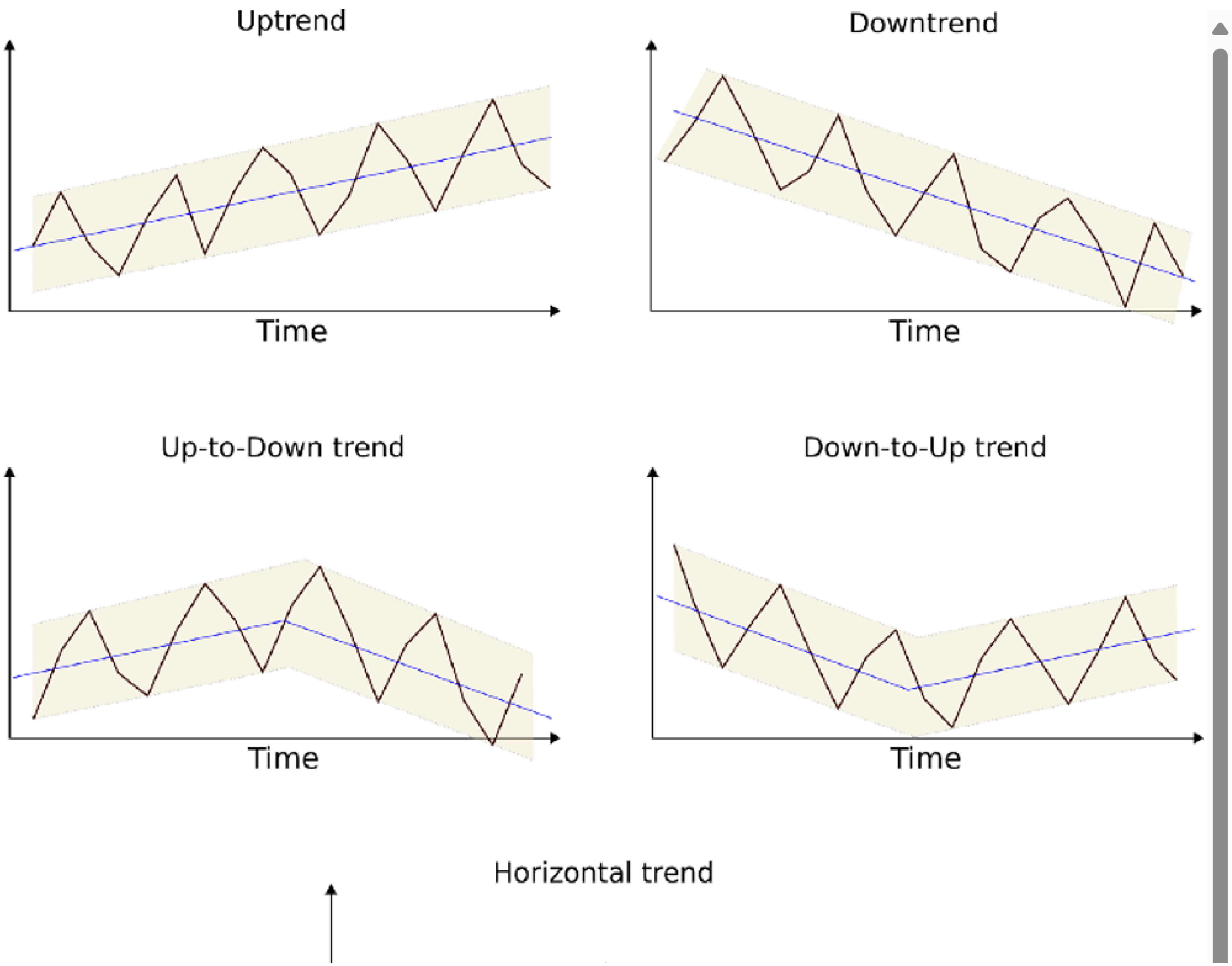
1. **Trend:**
    - Represents the long-term progression of data, indicating whether values are generally increasing, decreasing, or remaining constant over time.
    - **Characteristics:**
        - Can be linear or nonlinear.
        - Does not account for short-term fluctuations.


Values
27500
25000
22500
20000
17500
2005-01-01
2008-01-01
2011-01-01
2014-01-01
2017-01-01

Trend Component

- Overall upward or downward movement
- Data taken over a period of years





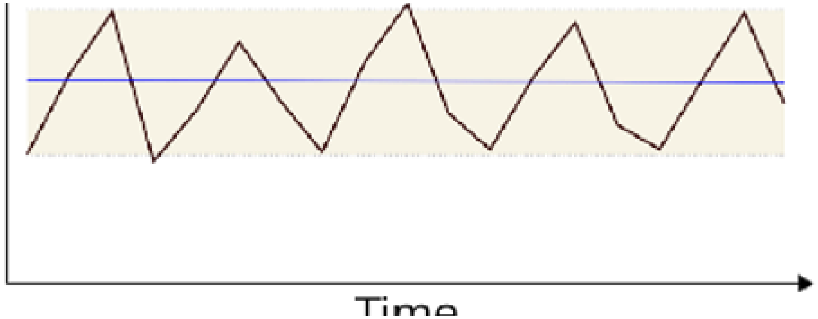
Uptrend

Downtrend

Up-to-Down trend

Down-to-Up trend

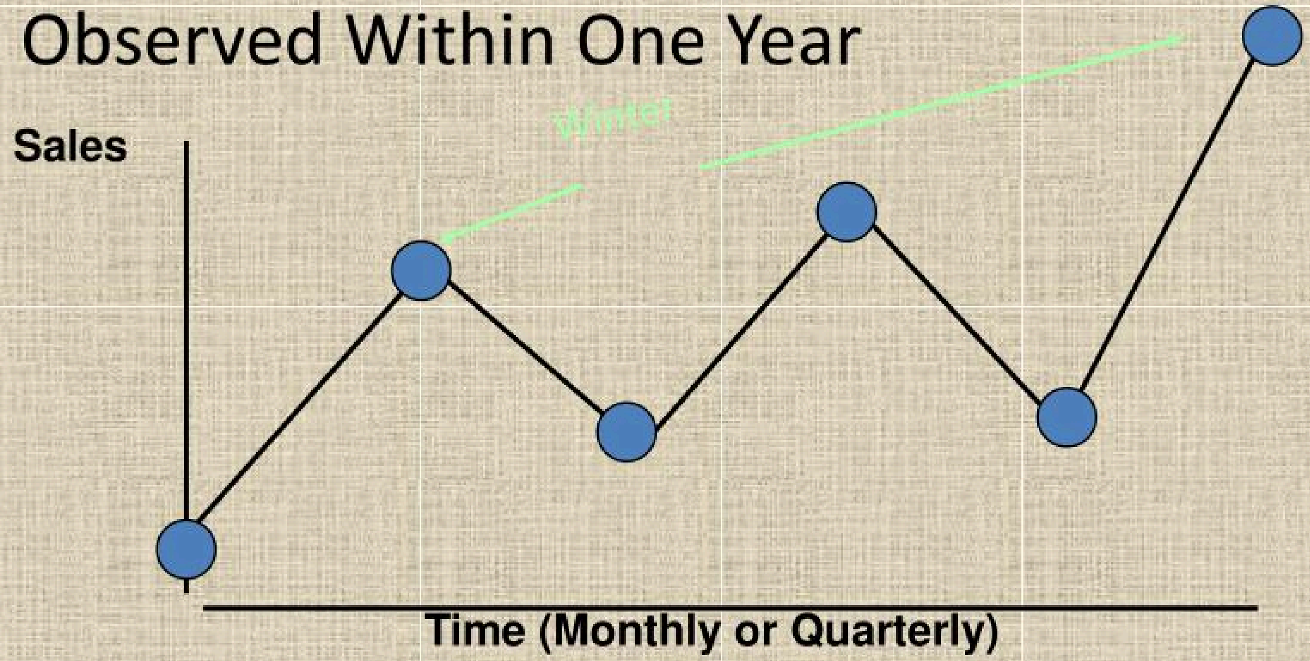
Horizontal trend

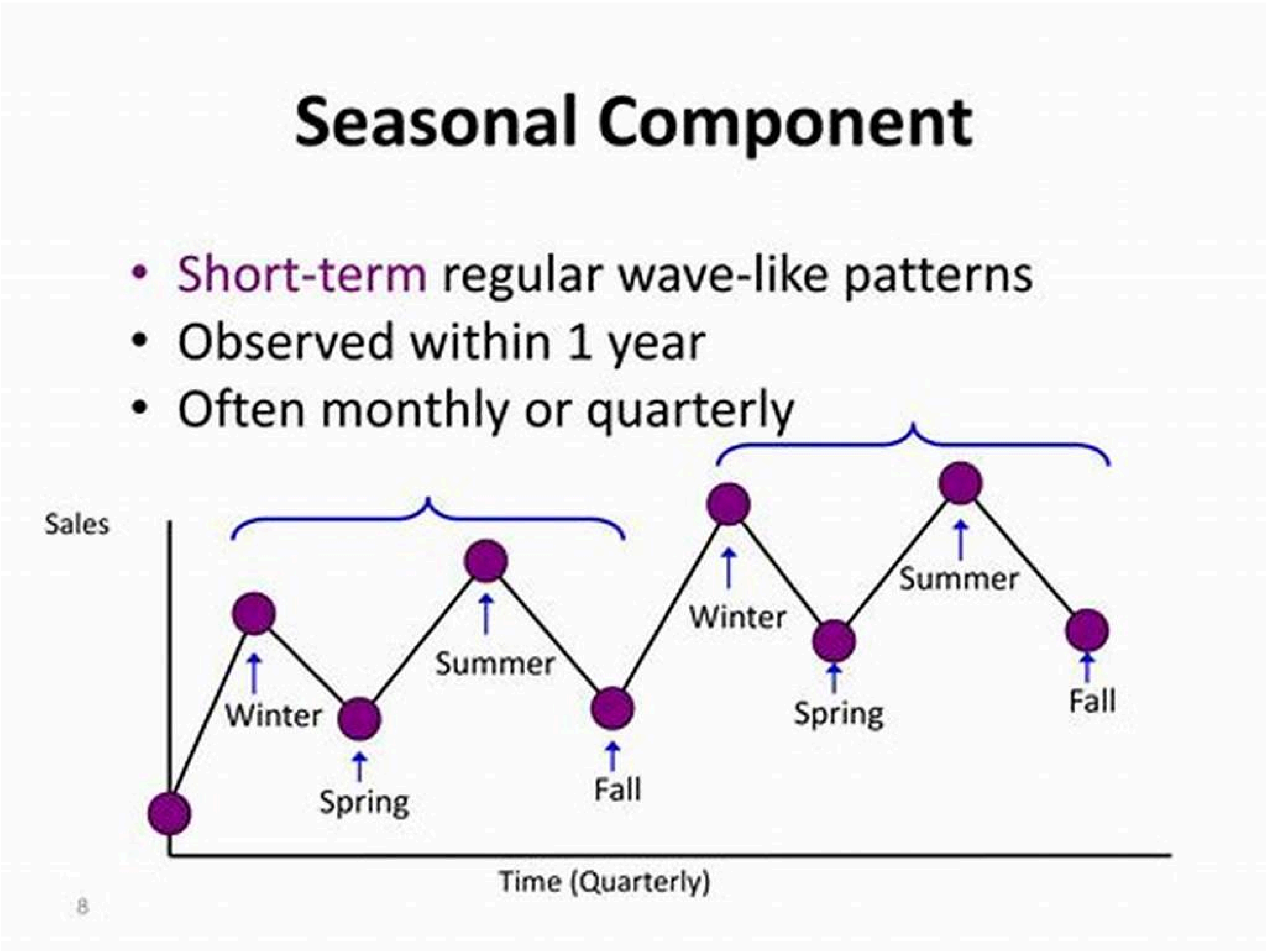


2. **Seasonality:**
    - Refers to regular, predictable patterns that repeat over specific intervals, such as daily, monthly, or annually.
    - **Characteristics:**
        - Caused by factors like holidays, weather, or business cycles.
        - Always periodic and of fixed duration.
    - **Examples:**
        - Increased retail sales during the holiday season.
        - Higher electricity bills in summer due to air conditioning.

Seasonal Component

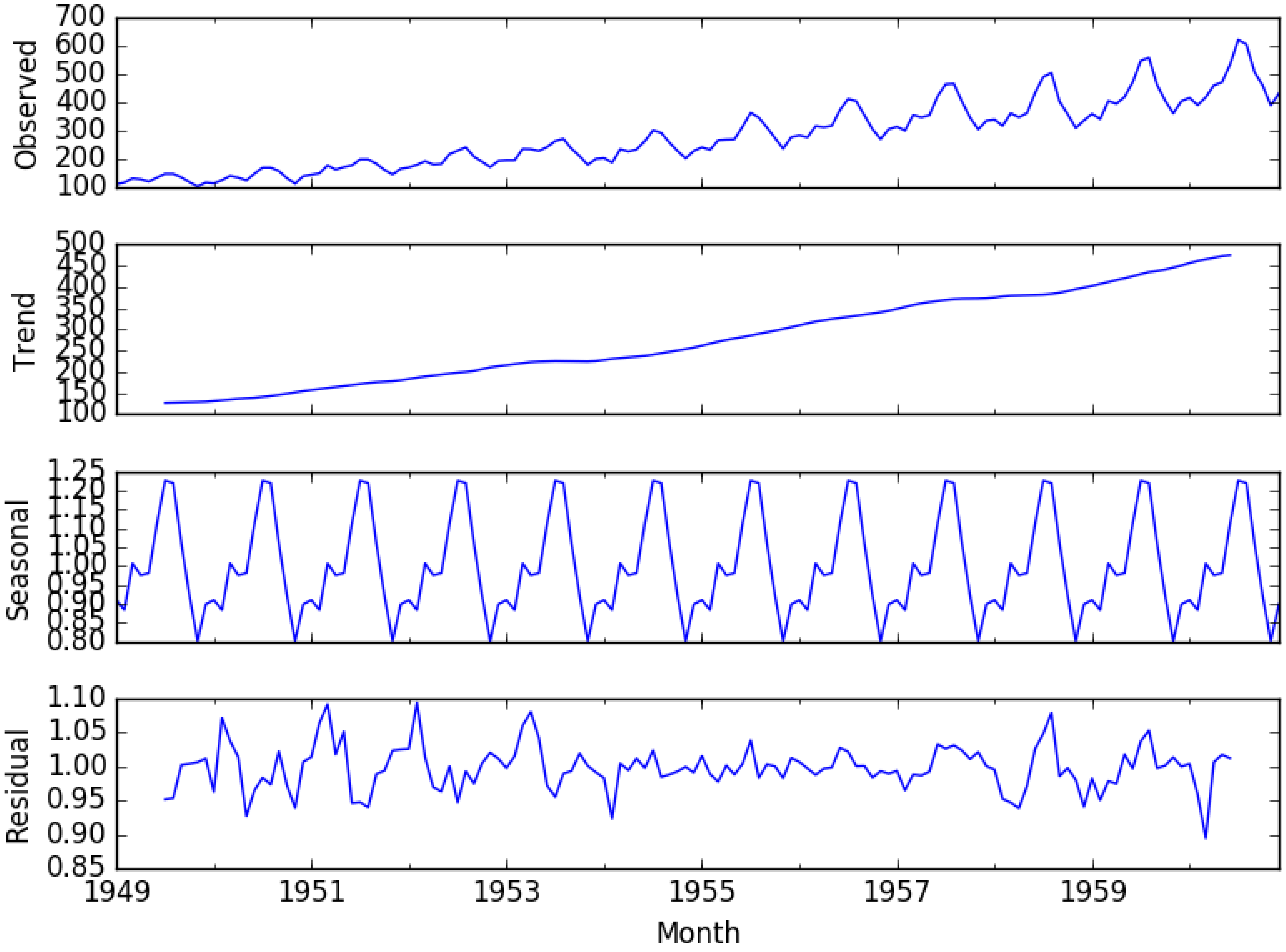
- Upward or Downward Swings
- Regular Patterns
- Observed Within One Year





Seasonal Component

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



3. **Noise (Random Variations):**

- Represents unpredictable, irregular fluctuations in the data that cannot be attributed to trend or seasonality.

- **Characteristics:**
  - Caused by random, unforeseen factors.
  - Often considered "error" in a time series model.
- **Examples:**
  - Sudden changes in stock prices due to unexpected events.
  - Random spikes in traffic data caused by accidents.

4. **Cyclic Patterns:**
   - Refers to fluctuations that occur over longer, irregular periods and are not tied to a fixed calendar cycle.
   - **Characteristics:**
     - Often influenced by economic, business, or environmental cycles.
     - Unlike seasonality, cyclic patterns do not have a fixed duration.
   - **Examples:**
     - Business cycles with phases of boom and recession.
     - Long-term changes in demand for a product due to market shifts.

5. **Residuals (Error):**
   - Represents the part of the time series that remains after removing the trend, seasonality, and cyclic components.
   - Residuals are simply the difference between the actual data points and the values predicted by the model.
   - **Characteristics:**
     - Often modeled as random noise with a mean of zero.
     - Provides insight into anomalies or unexplained variations in the data.
   - **Examples:**
     - Deviation in stock prices due to unexpected news.
     - Errors in weather forecasts after accounting for trends and patterns.
   - **Practical Use:**
     - Residual analysis helps evaluate the accuracy of forecasting models.
     - Identifying patterns in residuals may indicate an improperly specified model.

2. Time Series Decomposition

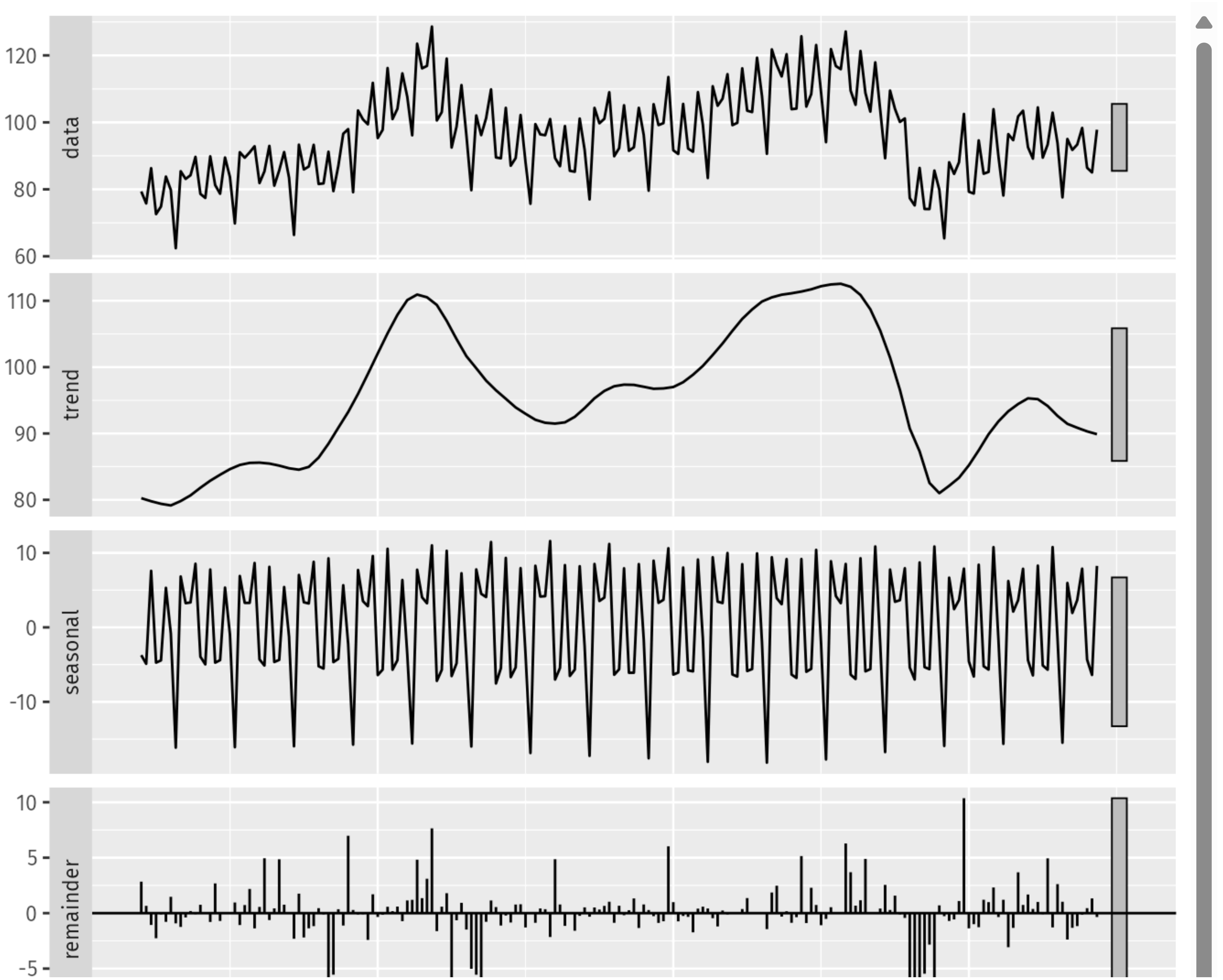
Additive vs. Multiplicative Models:

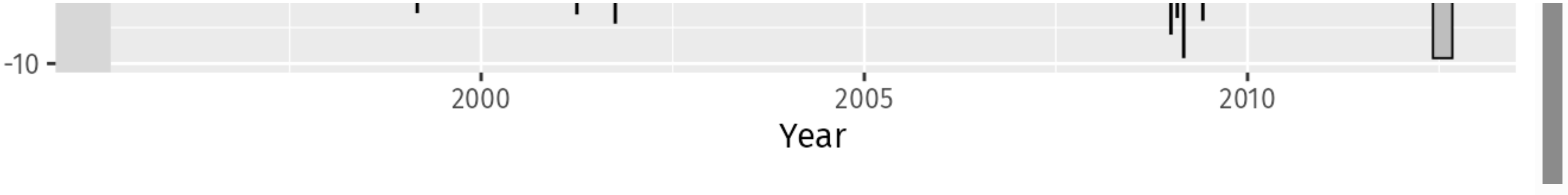
1. **Additive:**

- When the time series components add up:

  y(t) = T(t) + S(t) + e(t)

  - Data = Trend Component + Seasonal Component + Remainder/Residual Component
- Suitable for time series where seasonal fluctuations and trend remain relatively constant over time.


data
120
100
80
60
trend
110
100
90
80
seasonal
10
0
-10
remainder
10
5
0
-5



2. **Multiplicative:**

- When components multiply:

y(t) = T(t) * S(t) * e(t)

- Suitable for time series where seasonal fluctuations vary in proportion to the trend.

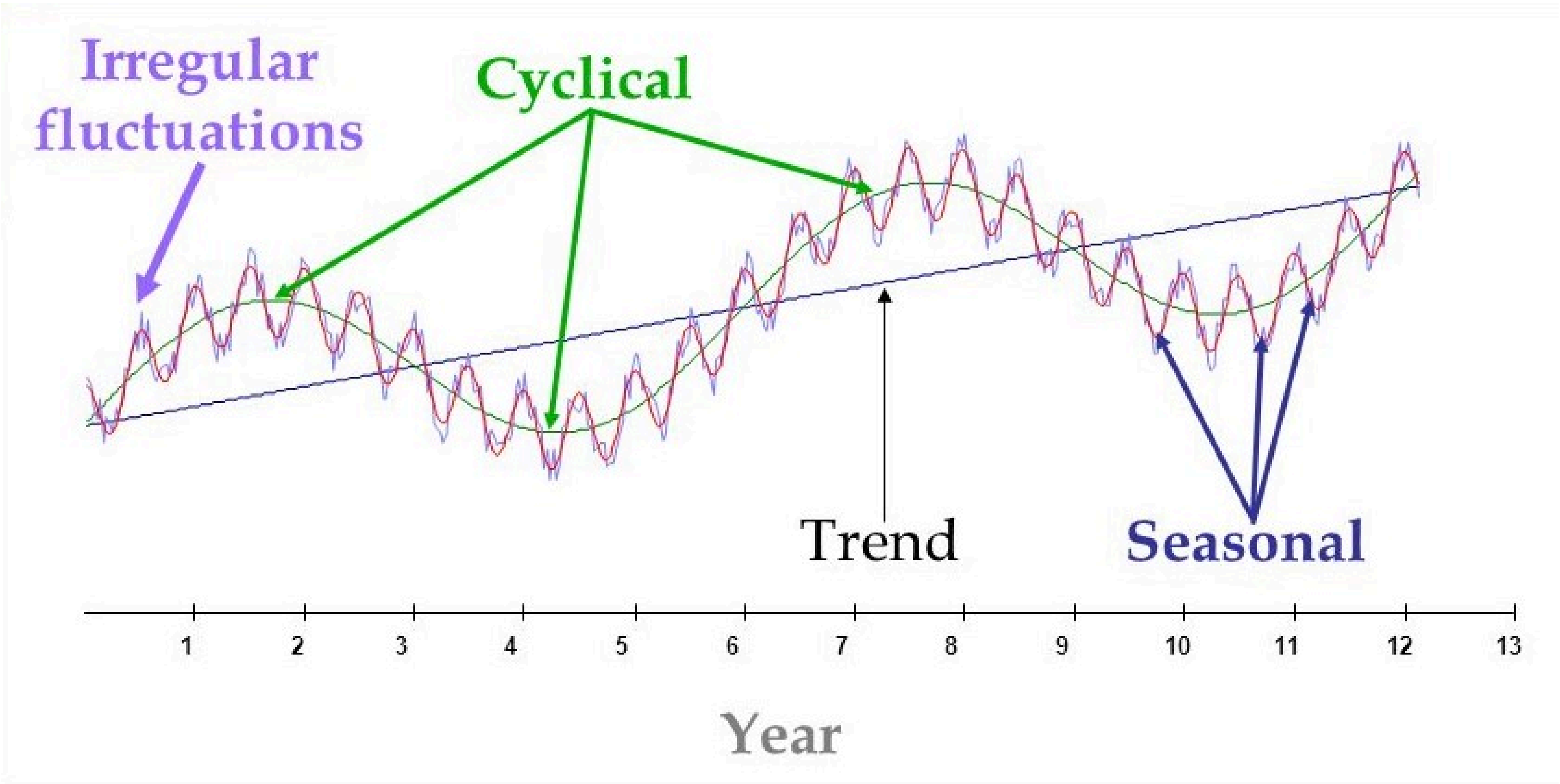
Techniques for Decomposition:

1. **Classical Decomposition:** Separates components using predefined mathematical formulas.

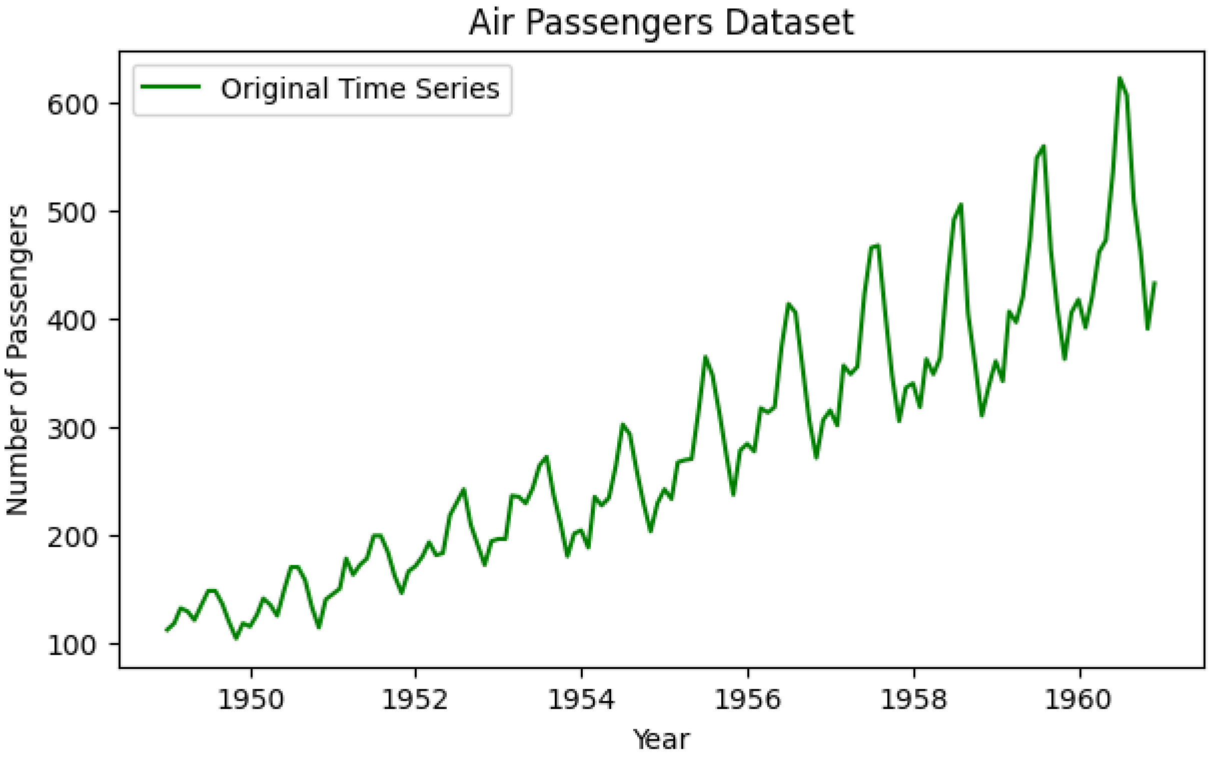
2. **STL Decomposition (Seasonal-Trend using LOESS):**

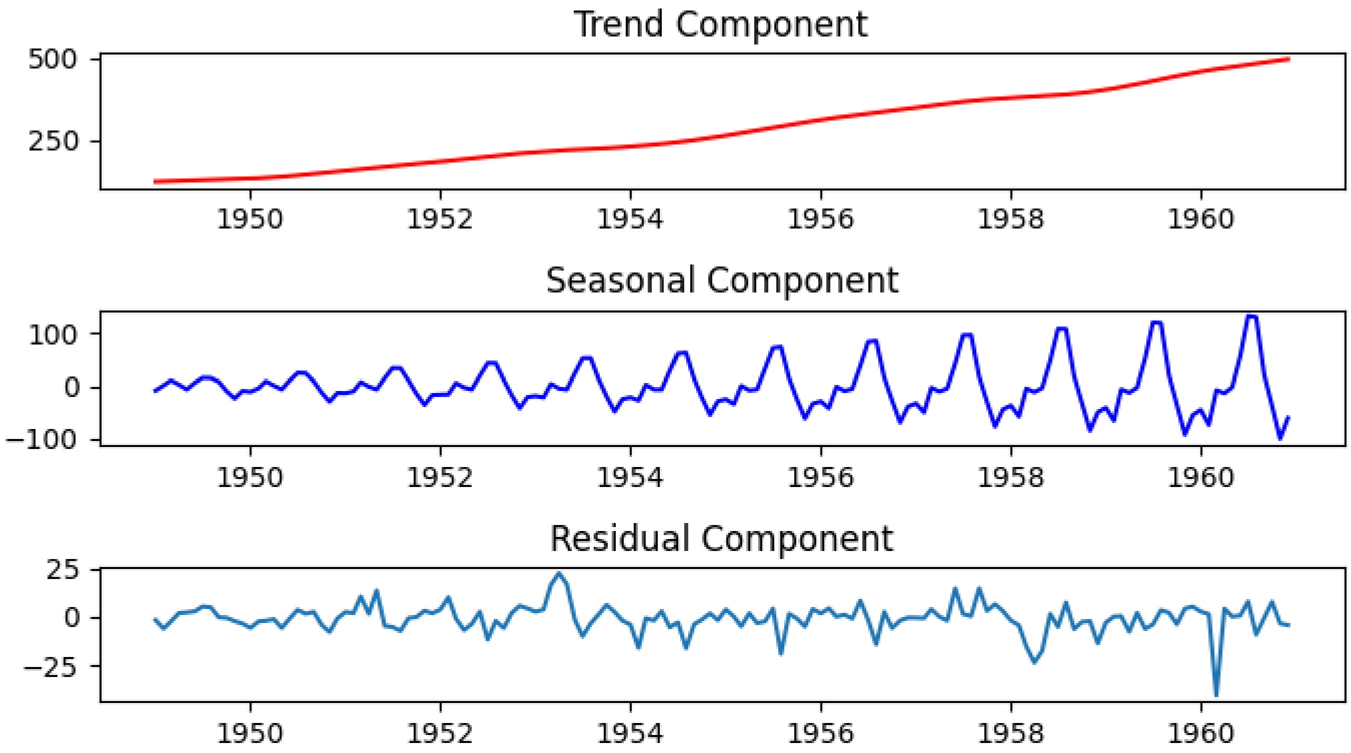
- A flexible method that can handle complex seasonality and non-linear trends.
- Provides robustness against outliers.
- Handles seasonality, trend, and noise effectively.
- Visual output helps in understanding component separation.

STL Decomposition Process:

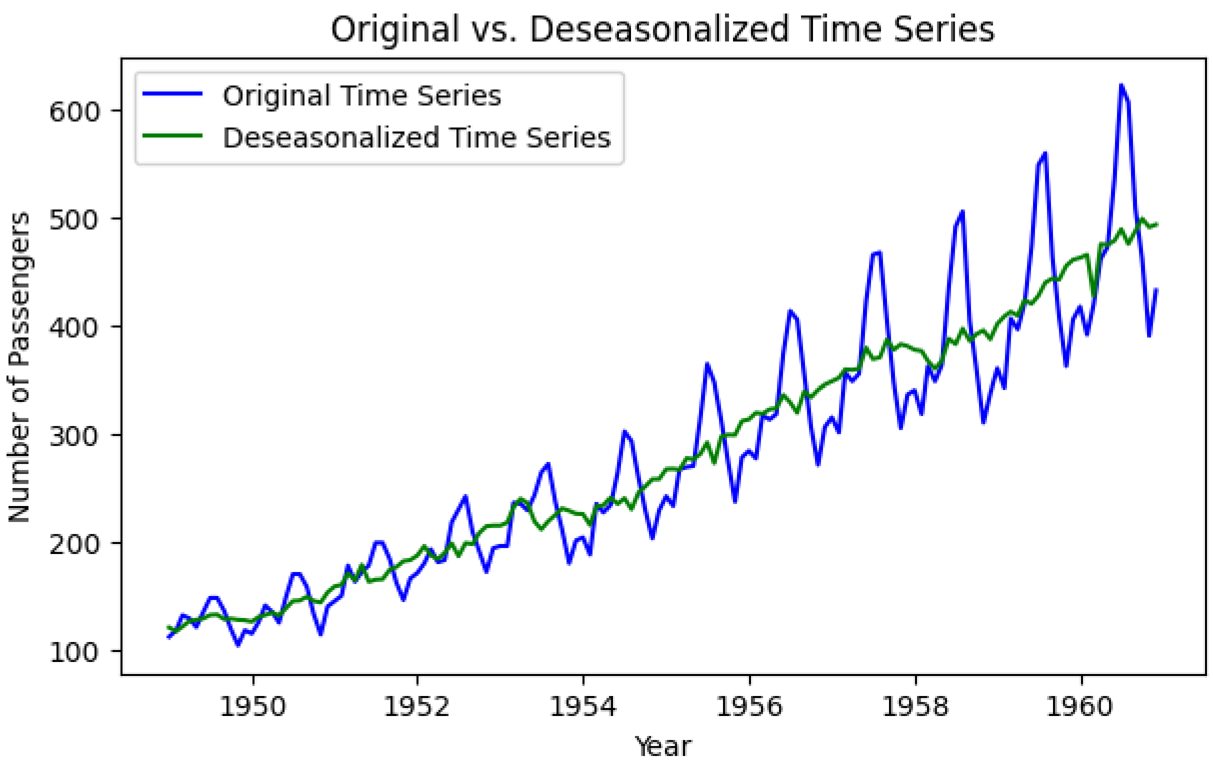


- **Step 1: Identify Trend Component**
  - Use techniques like moving averages or LOESS (Locally Estimated Scatterplot Smoothing) to smooth out short-term variations.
  - Seasonal-Trend decomposition using Loess (STL) is a specific decomposition method that employs the Loess technique to separate a time series into its trend, seasonal, and residual components.
  - Learn more here...


Air Passengers Dataset
Original Time Series
Number of Passengers
100
200
300
400
500
600
1950
1952
1954
1956
1958
1960
Year



- **Step 2: Extract Seasonality Component**
  - Identify repeating patterns by examining fixed intervals (e.g., monthly or yearly).



- **Step 3: Calculate Residuals (Noise)**
  - Subtract (in the additive model) or divide (in the multiplicative model) the trend and seasonality from the original series.

Exploring Time Series Data

1. **Exploring Time Series Data:**
   Exploring time series data involves understanding its structure and identifying patterns such as trends, seasonality, and randomness. This is typically done through visualization and statistical tools.

2. **Data Visualization for Time Series:**

   Visualization is a crucial first step in time series exploration as it provides an intuitive understanding of the data.

   - **Line Plots:** Show trends over time.
   - **Seasonal Plots:** Highlight recurring patterns.

   Line Plots:

   - **Purpose:** Display raw time series data over time.
   - **Insights:** Identify trends, seasonality, and anomalies.
   - **Example:**

     ```
     import pandas as pd
     import matplotlib.pyplot as plt

     data = pd.read_csv('time_series.csv')
     plt.plot(data['date'], data['value'])
     plt.title('Time Series Line Plot')
     plt.xlabel('Date')
     plt.ylabel('Value')
     plt.show()
     ```

   Seasonal Plots:

   - **Purpose:** Highlight repeating seasonal patterns by visualizing data for each period (e.g., months, years).
   - **Insights:** Understand periodic behavior.

   Box Plots:

   - **Purpose:** Summarize the distribution of data across periods (e.g., months or seasons).
   - **Insights:** Compare variability and detect outliers.
   - **Example:**

     ```
     import seaborn as sns
     sns.boxplot(x='month', y='value', data=data)
     plt.title('Seasonal Box Plot')
     plt.show()
     ```

Autocorrelation in Time Series

Learn More...

Autocorrelation measures the relationship between a time series and a lagged version of itself.

It is like asking, "Does the past predict the future?" It measures how similar a time series is to itself at different points in time.

Analogy example: Imagine This:

- You're looking at sales of ice cream over months.
- You notice sales are high every summer (June, July, August).
- If sales this June are high, you might expect next June to also have high sales.

This similarity between current and past values is autocorrelation!

What is Autocorrelation?

- The degree to which current values of the series are related to its past values.
- **Positive Autocorrelation:** If a high value is followed by another higvalue (or low by low).
  - Example: Stock prices rising for several days in a row.
- **Negative Autocorrelation:**

If a high value is followed by a low value (or vice versa).

- Example: Alternating days of high and low rainfall.
- **No Autocorrelation:** If values seem completely random and unrelated to past values
- **Lag:** The time difference between the original and shifted series.

ACF (Autocorrelation Function): Measures correlation at different lags.

PACF (Partial Autocorrelation Function):
Measures the correlation of the series with a specific lag after removing the effects of shorter lags.

These can be visualised with Use ACF/PACF plots

~~~
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

     plot_acf(data['value'], lags=20)
     plt.title('Autocorrelation Function')
     plt.show()

     plot_pacf(data['value'], lags=20)
     plt.title('Partial Autocorrelation Function')
     plt.show()
     ```
---
~~~

Insights from Autocorrelation:

- **High Autocorrelation:** Indicates strong dependencies between observations over time.
- **Low Autocorrelation:** Suggests weak or no dependencies.
- Useful for identifying suitable models for forecasting (e.g., ARIMA).

Durbin-Watson Test:

A statistical test to detect the presence of autocorrelation in the residuals of a regression model.

```
    ~ 2: No autocorrelation.
    < 2: Positive autocorrelation.
    > 2: Negative autocorrelation.
```

```python
from statsmodels.stats.stattools import durbin_watson

dw_stat = durbin_watson(model.resid)
print(f'Durbin-Watson statistic: {dw_stat}')
```

END

THANK YOU!

Live Exercise Solutions

```
In [ ]:
```

Mohammad Idrees Bhat

Tech Skills Trainer | AI/ML Consultant