

# AI UX & Data Visualisation Design Principles (CA6002)

**Associate Professor Goh Wooi Boon**

College of Computing and Data Science  
Nanyang Technological University

email: [aswbgo@ntu.edu.sg](mailto:aswbgo@ntu.edu.sg)



1

1

## Chapter 2.3 – Time Series Analysis

### Contents

- What is a Time Series?
- Handling Time Series Datasets
- Components of a Time Series
- Time Series Decomposition
- Visualising Seasonality
- Visualising Trend
- Decomposing The Time Series



© A/P Goh Wooi Boon (CCDS/NTU)

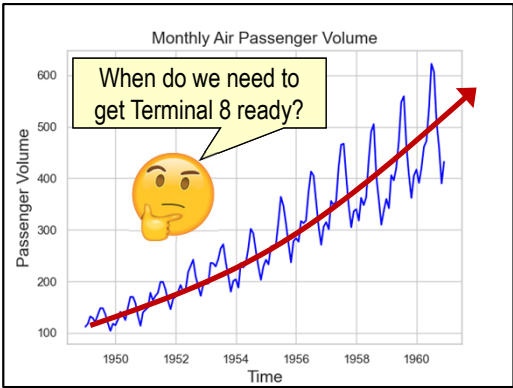
2

2

# What Is A Time Series?

## The Only Constant is Change

- A time series is a sequence of data points that occur in **successive order** over some period of **time**<sup>[1]</sup>.
- Depending on the observation frequency, the series may span over just a second (e.g. sampled every msec) or many years (e.g. sampled daily or monthly).
- Time series analysis is very important, especially to the scientific and financial community as it is a preparatory step in being able to make meaningful and accurate **forecast** of future observations or trends<sup>[1]</sup>.



3

# Handling Time Series Datasets

## Importing Time Series Data

- Most single parameter time series datasets contain two columns. The **time stamp** (e.g. date) and its associated **measured value**.
- Pandas provides useful features (e.g. **parse\_date**) to normalise the date values to its standard format when reading the time series dataset, which makes subsequent analysis more convenient<sup>[2]</sup>.

```
Data = pd.read_csv(           # read csv file function
    'Air Passenger.csv',      # time series file name
    parse_dates=['Month'])    # re-format date column
```

Month	#Passengers
1949-01	112
1949-02	118
...	...
1960-11	390
1960-12	432

*Air Passenger.csv*

- The standard pandas date format is in the form **YYYY-MM-DD**.

	Month	#Passengers
0	1949-01-01	112
1	1949-02-01	118
..	...	...
142	1960-11-01	390
143	1960-12-01	432

4

## Handling Time Series Datasets

### Modifying the Time Series Dataframe

- Seaborn plots allow the **hue** parameter to be modified by a categorical variable. As such, it is helpful to create some additional columns in the time series dataframe for some date-related info (e.g. 'Year', 'Month')<sup>[2]</sup>.

```
Data.rename(columns = {'Month':'Date'}, inplace=True) # rename date column
Data['Year'] = [d.year for d in Data.Date]           # insert the year column
Data['Month'] = [d.strftime('%b') for d in Data.Date] # insert abbrev months
```

Month	#Passengers
1949-01	112
1949-02	118
...	...
1960-11	390
1960-12	432

Air Passenger.csv

	Month	#Passengers
0	1949-01-01	112
1	1949-02-01	118
...	...	...
142	1960-11-01	390
143	1960-12-01	432

Using `parse_date` in `read_csv`

	Date	#Passengers	Year	Month
0	1949-01-01	112	1949	Jan
1	1949-02-01	118	1949	Feb
...	...	...	...	...
142	1960-11-01	390	1960	Nov
143	1960-12-01	432	1960	Dec

Create Year and Month columns



[2] Selva Prabhakaran, Time Series Analysis in Python – A Comprehensive Guide with Example  
- <https://www.machinelearningplus.com/time-series/time-series-analysis-python/>

5

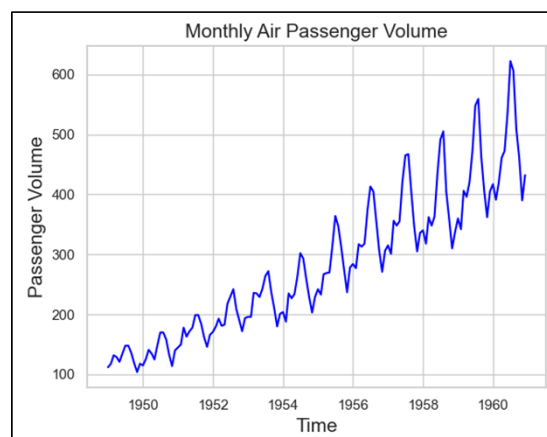
5

## Handling Time Series Datasets

### Visualising Time Series

- The modified time series dataframe can now be plotted in various ways.
- As a simple **line plot**.

```
import seaborn as sns # import Seaborn
sns.lineplot(         # lineplot function
    data=Data,         # time series dataframe
    x='Date',          # Date for x-axis
    y='#Passengers',   # series values for y-axis
    color='blue')      # use blue colour for line
```



Simple line plot of *Air Passenger.csv*



6

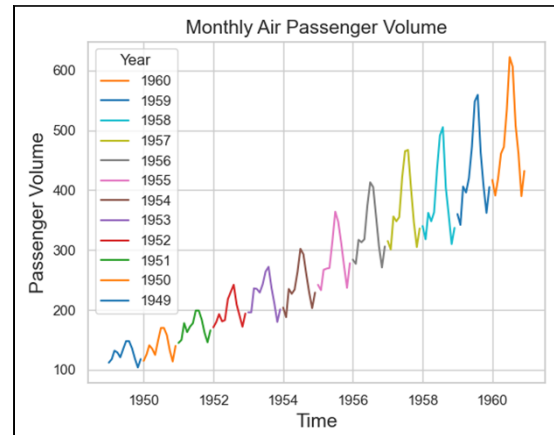
6

## Handling Time Series Datasets

### Visualising Time Series

- The modified time series dataframe can now be plotted in various ways.
- Modifying the **hue** parameter with the 'Year' information

```
sns.lineplot(          # lineplot function
    data=Data,          # time series dataframe
    x='Date',           # Date for x-axis
    y='#Passengers',    # series values for y-axis
    hue='Year',         # colour based on years
    palette='tab10')    # specify palette
```



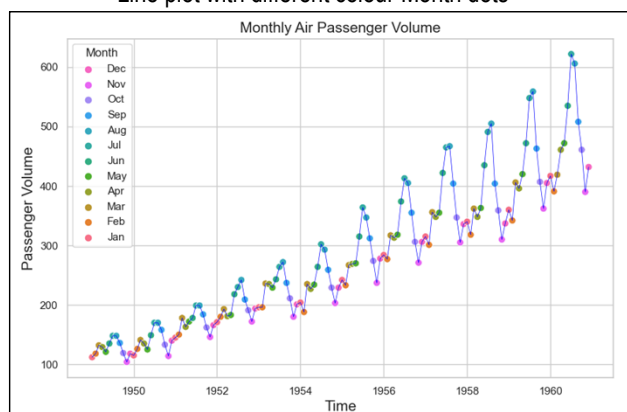
Line plot with different colour for each year (*Air Passengers.csv*)

## Handling Time Series Datasets

### Visualising Time Series

- The modified time series dataframe can now be plotted in various ways.
- Combine a scatter plot that has 'Month' information in the **hue** parameter.
- With an overlaid blue line plot to connect the scattered colour dots.

Line plot with different colour Month dots



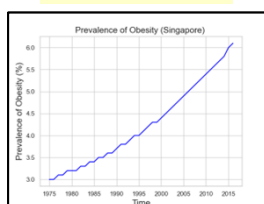
```
sns.lineplot(data=Data,x='Date',y='#Passengers', ,color='blue') # draw line plot
sns.scatterplot(data=Data,x='Date',y='#Passengers', hue='Month') # add Month colour dots
```

## Components Of A Time Series

### The Parts Making Up the Whole

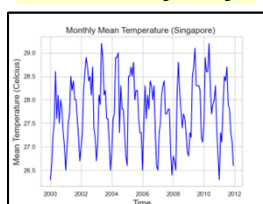
- A time series can be thought of as being made up of 4 components<sup>[3]</sup>.
  - Seasonal component ( $S_t$ )
  - Trend component ( $T_t$ )
  - Cyclical component ( $C_t$ )
  - Noise or Residual component ( $R_t$ )
- In some time series, only a subset of these components may be visually obvious.

#### Trend only



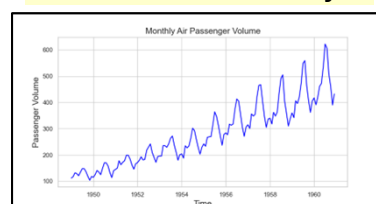
Obesity prevalence

#### Seasonality only



Monthly Temperatures

#### Trend and Seasonality



Air Passenger Volume

[3] How To Isolate Trend, Seasonality And Noise From A Time Series - <https://timeseriesreasoning.com/contents/time-series-decomposition/>

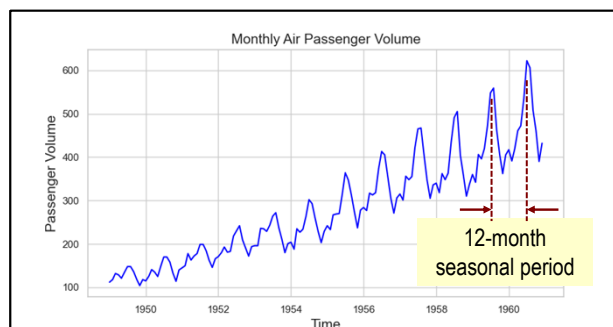
9

9

## Components Of A Time Series

### The Seasonal Component

- The seasonal component ( $S_t$ ) describes the **periodic** fluctuations observed in the time series<sup>[3]</sup>.
- The repeated seasonal variations are mostly in terms of day, week, month, quarter (e.g. financial data) or year.
- Many of these time markers are related to the temporal patterns in **nature** (e.g. the earth's rotation about its axis and around the sun), while others (e.g. quarters) are just widely accepted international **human norms**.



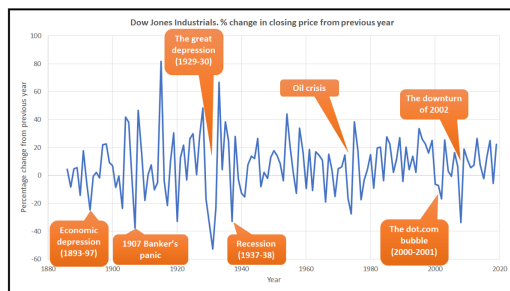
Monthly Air Passenger Volume

10

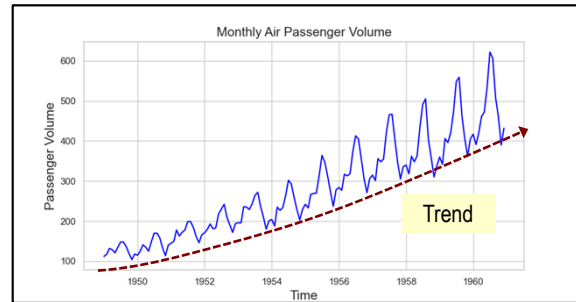
## Components Of A Time Series

### The Trend and Cyclical Components

- The trend component ( $T_t$ ) describes the **pattern** in the time series data that **spans across** seasonal periods<sup>[3]</sup>.
- The cyclical component describes medium term, non-periodic deviations from the trend. In most cases, it is combined into a single trend-cycle component ( $T_t$ )<sup>[4]</sup>.



Cycles of boom and bust in the stock market <sup>[3]</sup>



Monthly Air Passenger Volume

[4] Rob Hyndman, Cyclic and seasonal time series-  
<https://robhyndman.com/hyndsight/cyclists/>

11

11

## Time Series Decomposition

### Additive or Multiplicative?

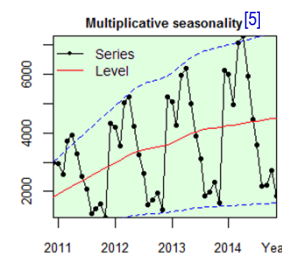
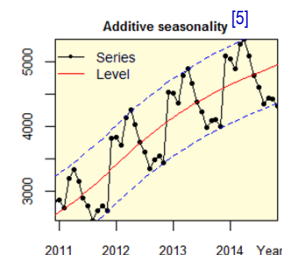
- Depending on the nature of the trend and seasonality, the time series can be modelled as **Additive**:

$$x_t = S_t + T_t + R_t \quad \text{Eqn. (2)}$$

- or as **Multiplicative**:

$$x_t = S_t T_t R_t \quad \text{Eqn. (3)}$$

- Additive** decomposition should be used when the magnitude of the seasonality has **no dependencies** on the magnitude of the values of the raw time series<sup>[4]</sup>.
- Multiplicative** decomposition should be used when the magnitude of the seasonality varies with the changing magnitude of the time series samples<sup>[4]</sup>.



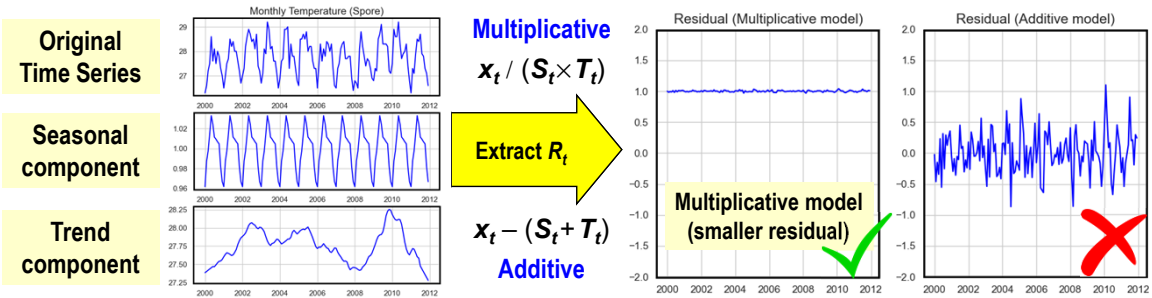
12

12

# Components Of A Time Series

## The Residual Component

- The residual component ( $R_t$ ) is what **remains** after you separate out seasonality and trend from the time series. It arises from the effects of **unknown** factors<sup>[3]</sup>.
- The **magnitude** of the residual often gives us a good indication of whether an additive or multiplicative models is the best fit for the time series being analysed<sup>[2]</sup>.



[2] Selva Prabhakaran, Time Series Analysis in Python – A Comprehensive Guide with Example - <https://www.machinelearningplus.com/time-series/time-series-analysis-python/>

13

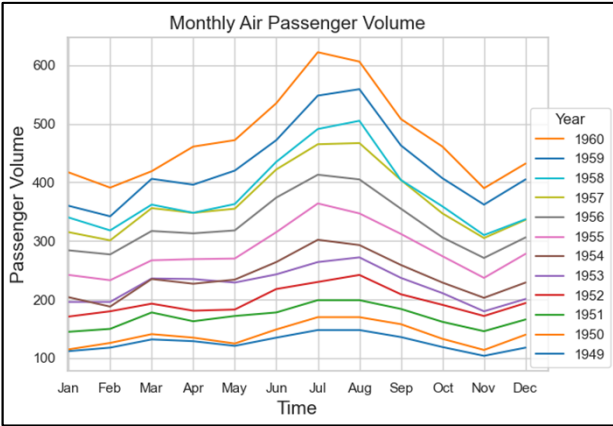
13

# Visualising Seasonality

## Comparing Each Season

- Repetitive pattern of the time series for each season (e.g. every month) can be visually compared using a series of seasonal line plots of different colours<sup>[2]</sup>.
- This allows the analysis of the repeated patterns every season and how these seasonal patterns vary after each season.

```
sns.lineplot(
    data=Data,
    x='Month',
    y='#Passengers',
    hue = 'Year',
    palette = 'tab10')
# Seaborn line plot
# dataframe
# Month column
# time series value column
# Year column
# colour palette
```



[2] S. Prabhakaran, Time Series Analysis in Python – A Comprehensive Guide with Example - <https://www.machinelearningplus.com/time-series/time-series-analysis-python/>

Seasonal Line Plots of Air Passenger.csv

14

14

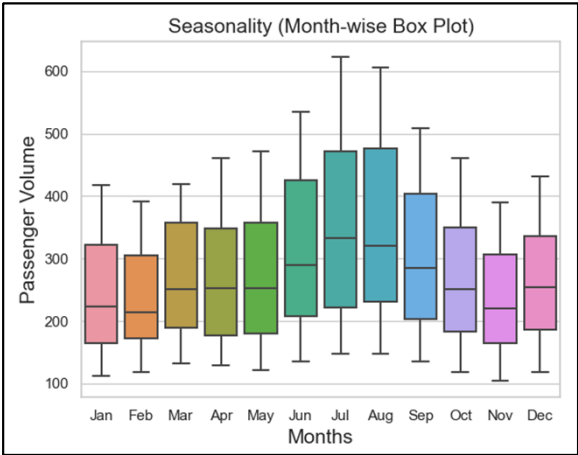
## Visualising Seasonality

### Looking at Seasonal Variations

- Variations within the season can be visualised using a **series of box plots**. E.g. if the seasonal repetition is yearly, then a **month-wise** box plot can be used to visualise the monthly distribution<sup>[2]</sup>.
- The central median lines show the **peak** and **trough** points in the season.
- The **height** of the box shows the extend of **variation** (i.e. vol. growth) in that month.

```
sns.boxplot(          # Seaborn box plot
    data=Data,         # dataframe
    x='Month',         # Month column
    y='#Passengers')   # time series value column
```

2] S. Prabhakaran, Time Series Analysis in Python – A Comprehensive Guide with Example - <https://www.machinelearningplus.com/time-series/time-series-analysis-python/>



Monthly distribution of Air Passenger.csv

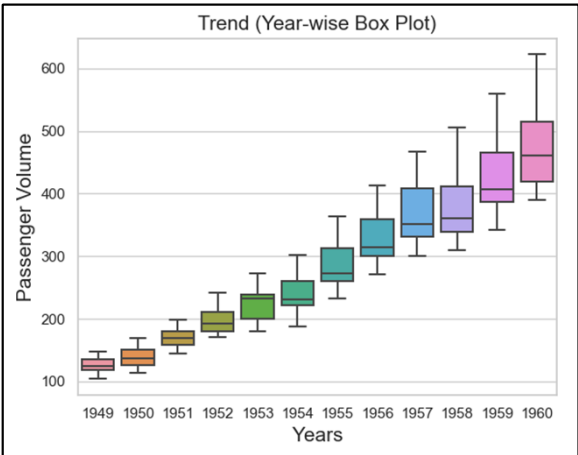
## Visualising Trend

### Looking at Trend Variations

- As the time series progresses, the trend can be observed by grouping data for each season into a separate box plot. A series of **year-wise** box plots can be used for a time series with yearly seasons<sup>[2]</sup>.
- Tracing the central **median lines** will show the changing **trend** over the years.
- The **position median** lines with the box show variations in each season.

```
sns.boxplot(          # Seaborn box plot
    data=Data,         # dataframe
    x='Year',          # Year column
    y='#Passengers')   # time series value column
```

2] S. Prabhakaran, Time Series Analysis in Python – A Comprehensive Guide with Example - <https://www.machinelearningplus.com/time-series/time-series-analysis-python/>

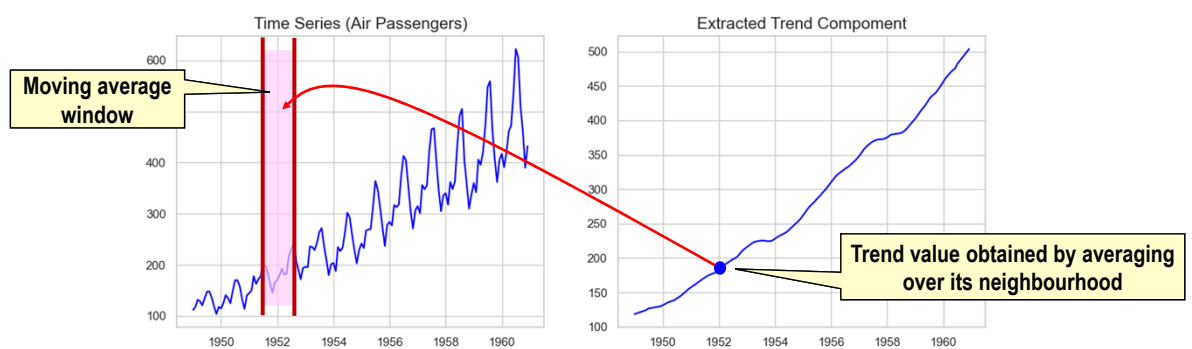


Yearly distribution of Air Passenger.csv

# Decomposing The Time Series

## Isolating Trends and Seasonality

- A time series ( $x_t$ ) can be decomposed by first isolating the trend ( $T_t$ ) using local averaging of the data samples over a suitable window (e.g using moving average)<sup>[3]</sup>.



# Decomposing The Time Series

## Isolating Trends and Seasonality

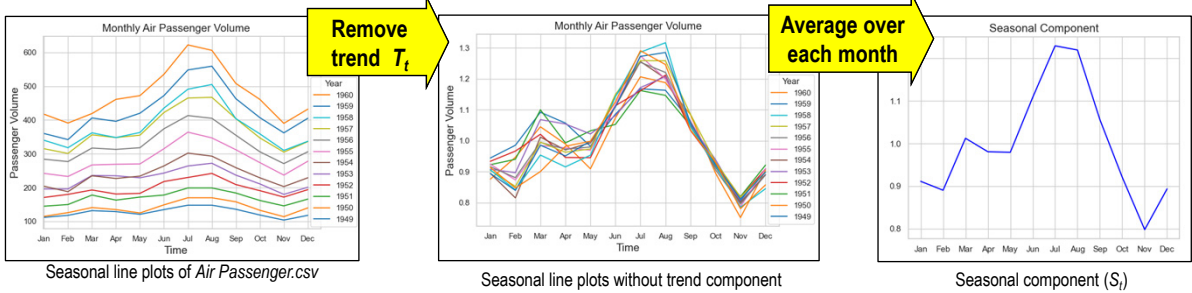
- A time series ( $x_t$ ) can be decomposed by first isolating the trend ( $T_t$ ) using local averaging of the data samples over a suitable window (e.g using moving average)<sup>[3]</sup>.
- With the isolated trend and deciding if the model is **additive** or **multiplicative**, the (seasonality and residual) can then be extracted from the time series<sup>[3]</sup>.

$S_t \times R_t = x_t / T_t$ Eqn. (4) - Multiplicative

$S_t + R_t = x_t - T_t$ Eqn. (5) - Additive

Decomposing The Time Series

Isolating Trends and Seasonality



- The ‘pure’ seasonal component ( $S_t$ ) can be estimated by computing the **average** value of the corresponding seasonal component for every available repeating season (e.g. for all months in Jan for every year, etc)<sup>[3]</sup>.

Decomposing The Time Series

Isolating Trends and Seasonality

- A time series ( $x_t$ ) can be decomposed by first isolating the trend ( $T_t$ ) using local averaging of the data samples over a suitable window (e.g using moving average)<sup>[3]</sup>.
- With the isolated trend and deciding if the model is **additive** or **multiplicative**, the (seasonality and residual) can then be extracted from the time series<sup>[3]</sup>.

$S_t \times R_t = x_t / T_t$

Eqn. (4) - Multiplicative

$S_t + R_t = x_t - T_t$

Eqn. (5) - Additive
- The ‘pure’ seasonal component ( $S_t$ ) can be estimated by computing the **average** value of the corresponding seasonal component for every available repeating season (e.g. for all months in Jan for every year, etc)<sup>[3]</sup>.
- Residual component ( $R_t$ ) is what remains after extracting seasonal component ( $S_t$ ).

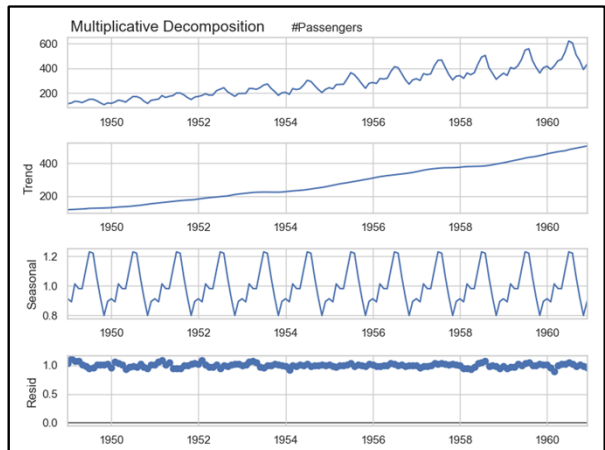
# Decomposing The Time Series

## Getting Help from Statsmodels


- Python **statsmodels** library provides a convenient function to decompose the time series into its various components.
- The **seasonal\_decompose()** function can be used for with your selected model (i.e. additive or multiplicative)<sup>[6]</sup>.

```
from statsmodels.tsa.seasonal import seasonal_decompose
Data = pd.read_csv("../Air Passengers.csv",
    parse_dates=['Month'],      # change date format
    index_col='Month')         # make date column index
Series = Data['#Passengers']   # get time series array
D = seasonal_decompose(
    Series,                    # time series array
    model='multiplicative')    # use multiplicative model
D.plot()
```

[6] Statsmodel – seasonal\_decompose documentation) - [https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal\\_decompose.html](https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html)



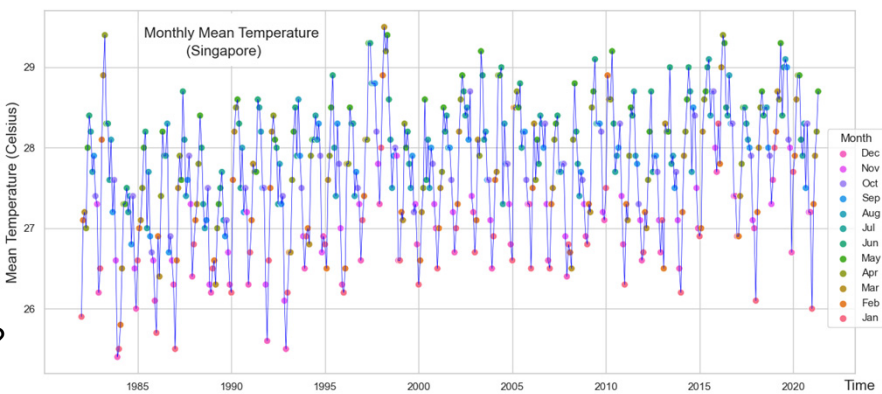
Result plots from seasonal\_decompose() on Air Passenger.csv 21




## Think and Apply

### Climate Change and Singapore

- Which are the hottest and coolest months in Singapore?
- Can you spot anomalies (e.g. hottest April or March)?
- Multiplicative or additive time series?
- What does the long-term trend say about future temperature changes in Singapore?





Monthly temperature (Singapore) - <https://data.gov.sg/>

22

## Summary

### Time Series Analysis

- Time series analysis is important for **exploring time-varying information** such as weather data, financial data, population-growth related phenomena, etc.
- **Line plots**, suitably augmented with coloured markers (e.g. dots) can be a very useful first stage of time series analysis to observe trends and repeating patterns.
- Time series data are usually made up of other **time-varying patterns** such as the trend, seasonal and cyclical components in an additive or multiplicative manner.
- **Seasonality analysis** (e.g. using box plots) allows us to understand the characteristics of the **repeating patterns** in the times series.
- **Trend analysis** (e.g. after isolating it using seasonal\_decompose) allows us to study more **long-term changes** in the time series.



23

23

## References for Time Series Analysis

- [1] Adam Hayes, What Is a Time Series? - <https://www.investopedia.com/terms/t/timeseries.asp>
- [2] Selva Prabhakaran, Time Series Analysis in Python – A Comprehensive Guide with Example – <https://www.machinelearningplus.com/time-series/time-series-analysis-python/>
- [3] How To Isolate Trend, Seasonality And Noise From A Time Series - <https://timeseriesreasoning.com/contents/time-series-decomposition/>
- [4] Rob Hyndman, Cyclic and seasonal time series - <https://robjhyndman.com/hyndsight/cyclicts/>
- [5] Images from Nikos, Additive and multiplicative seasonality – can you identify them correctly? (2014) - <https://kourentzes.com/forecasting/2014/11/09/additive-and-multiplicative-seasonality/>
- [6] Statsmodel – seasonal\_decompose documentation) - [https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal\\_decompose.html](https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html)



Note: All online articles were accessible on 13 Nov 2025

24

24