Time series

Time series analysis

- Time series analysis is a statistical technique used to analyze and interpret data points collected over a specific period at regular intervals. It focuses on studying the patterns, trends, and dependencies that exist within the time-ordered data
- Time series data is commonly encountered in various fields, including finance, economics, weather forecasting, stock market analysis, and more.

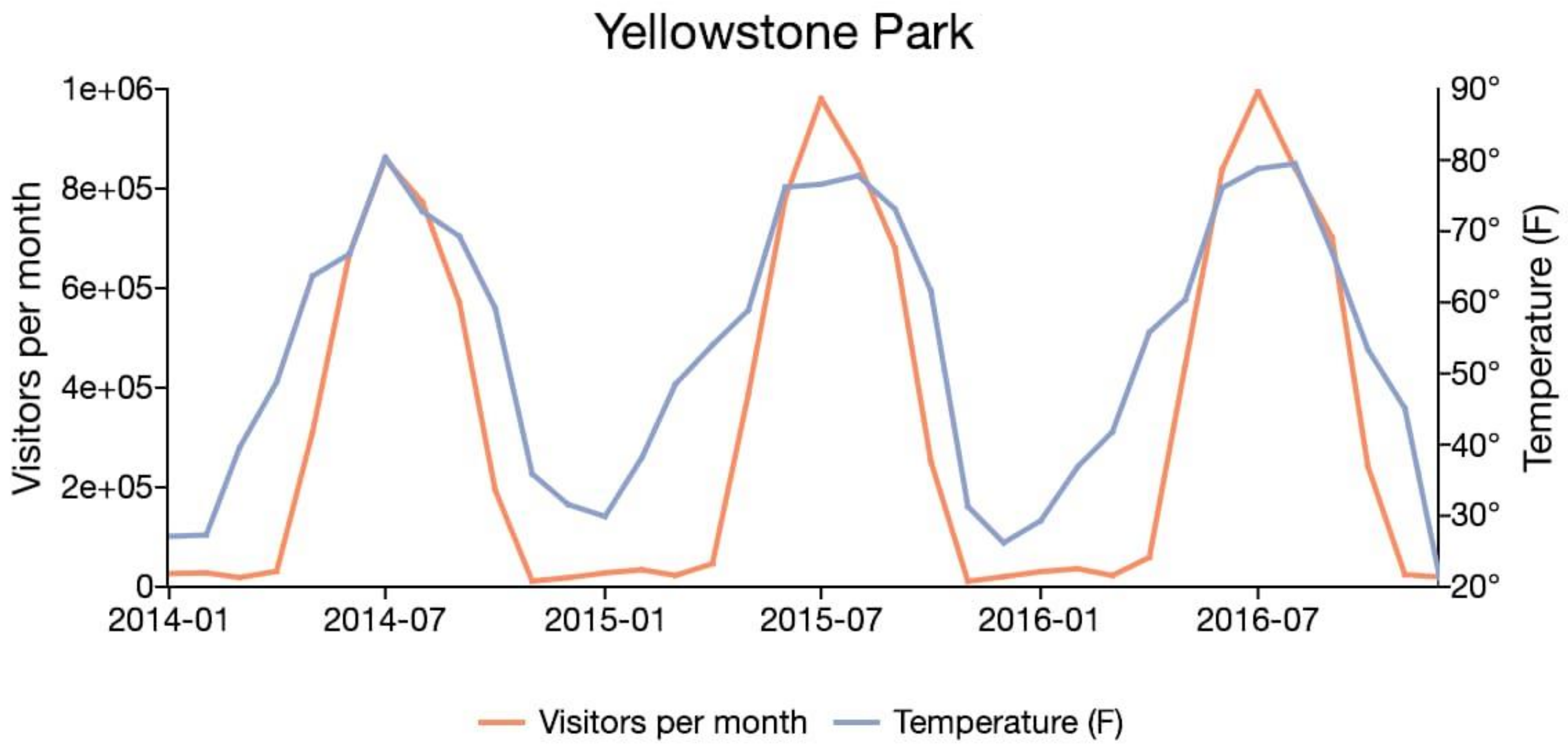
Time series – key concepts

- Time Series Data: Time series data consists of observations or measurements taken at successive time points. Each data point is associated with a specific timestamp or time interval. Examples include daily stock prices, monthly sales figures, hourly temperature readings, etc.
- Trend: The trend refers to the long-term behavior or movement of the data over time. It represents the overall direction of the data, whether it is increasing, decreasing, or remaining relatively stable.
- Seasonality: Seasonality refers to repetitive patterns or fluctuations that occur within a fixed time period. These patterns can be daily, weekly, monthly, or seasonal in nature. Seasonality can impact various time series, such as sales data, where certain periods exhibit regular fluctuations due to holidays or seasonal factors.
- Cyclical Patterns: Cyclical patterns are longer-term fluctuations in the data that are not of fixed frequency like seasonality. They may occur over periods longer than a year and are often influenced by economic or business cycles.

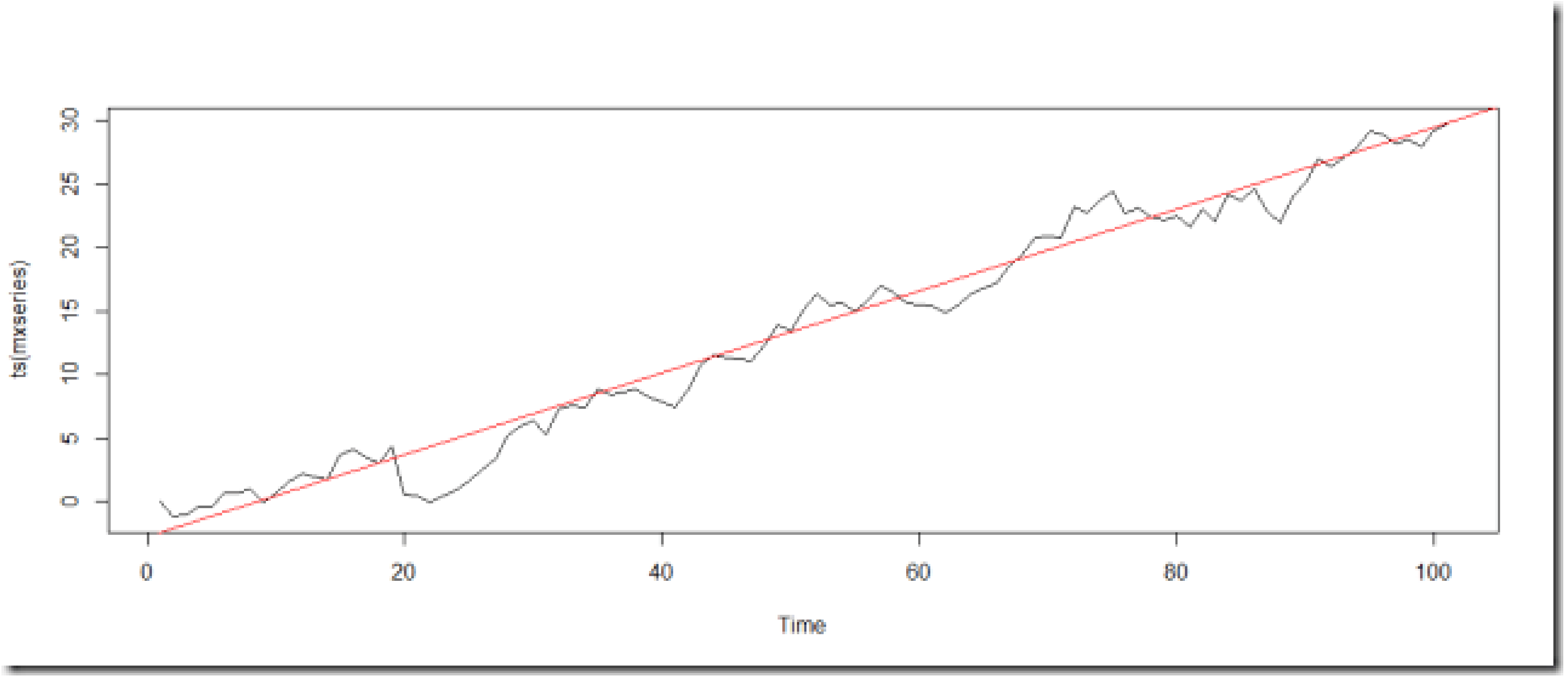
Time series – key concepts

- Autocorrelation: Autocorrelation measures the relationship between a time series and a lagged version of itself. It helps identify dependencies or correlations between past and present values. Positive autocorrelation indicates a relationship where high values tend to be followed by high values, and vice versa.
- Stationarity: Stationarity refers to a time series that exhibits consistent statistical properties over time. A stationary time series has a constant mean, variance, and autocorrelation structure. Stationarity is a fundamental assumption in many time series models.
- Forecasting: Time series analysis allows for forecasting future values based on historical patterns and trends. Various statistical models, such as ARIMA (Autoregressive Integrated Moving Average), exponential smoothing, and state space models, are commonly used for time series forecasting.

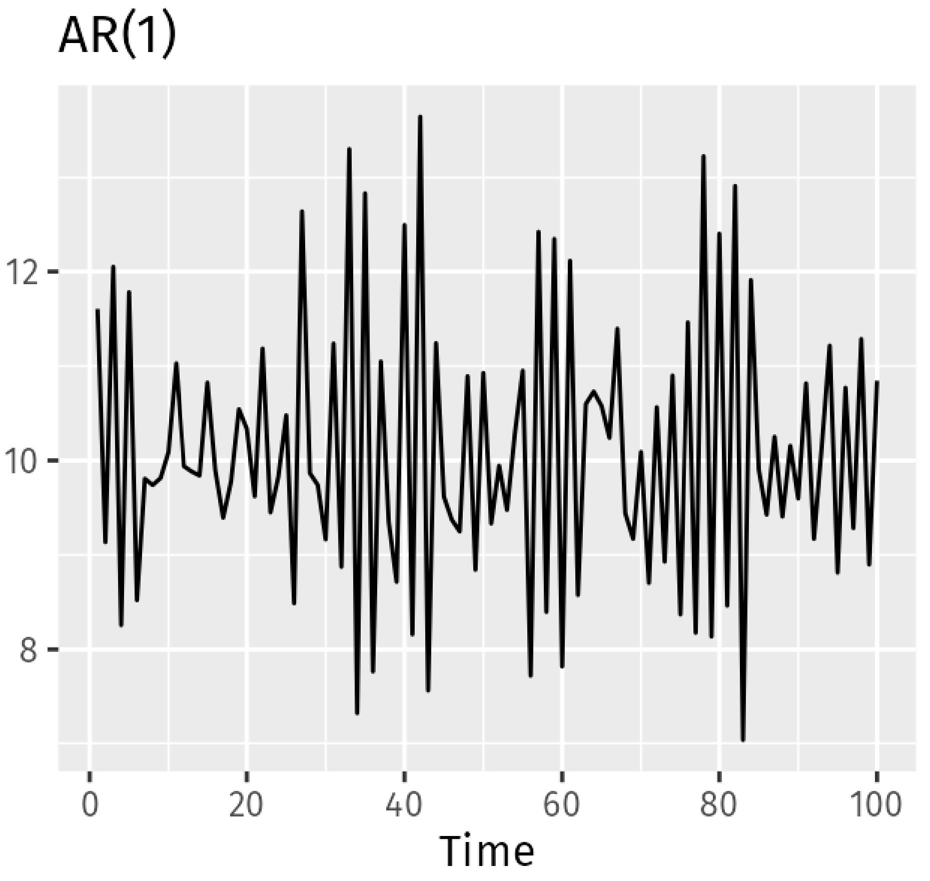
Time series - seasonality

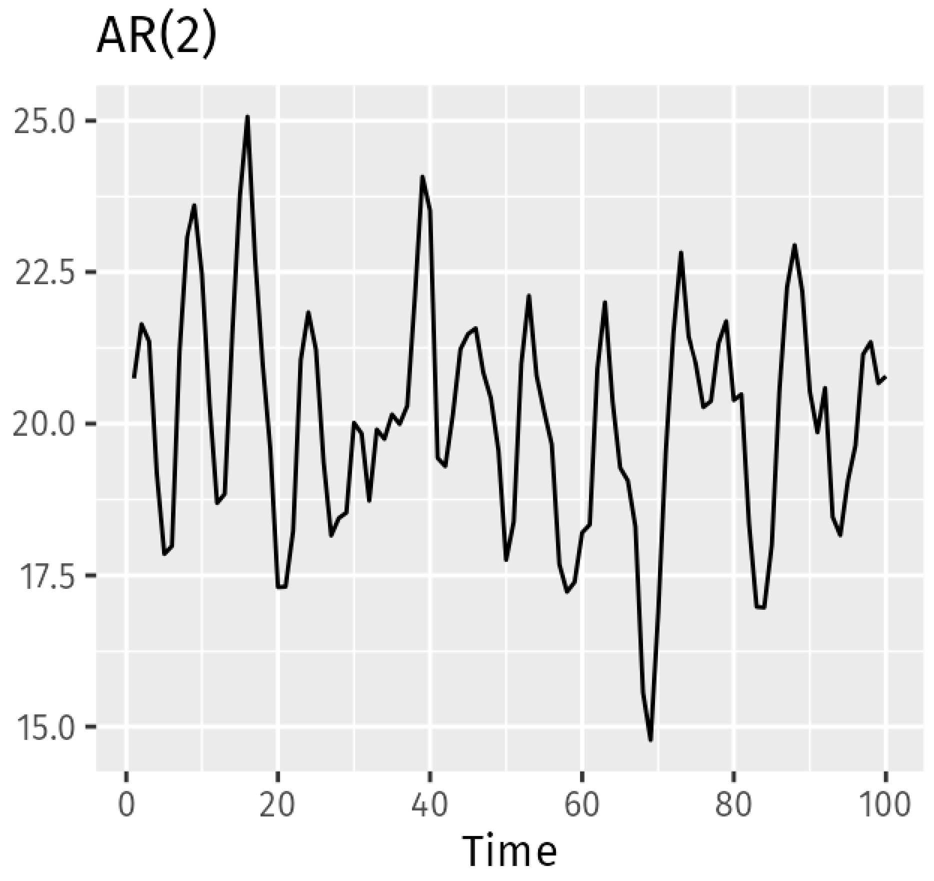


Time series - trend



Autoregressive model





Autoregressive model

- The notation $AR(p)$ indicates an autoregressive model of order p. The $AR(p)$ model is defined as

$$X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t \,,$$

where $\phi_1, \phi_2, \dots, \phi_n$ are the parameters of the model, and ϵ_t is a white noise.

The coefficients $\phi_1, \phi_2, \dots, \phi_n$ determine the strength and direction of the relationship between the current value and its lagged values. The error term ϵ_t captures the unexplained variation in the variable at time t.

Autoregressive model

- An autoregressive model is a type of statistical model that predicts future values of a variable based on its past values
- It assumes that the value at any given time depends linearly on its previous values and possibly on other relevant factors
- In an autoregressive model, the dependent variable is regressed on its own lagged values

Autoregressive model

- The term "autoregressive" refers to the fact that the model uses the variable's own past values as predictors
- The order of an autoregressive model, denoted by "p," represents the number of lagged values included in the model
- For example, an AR(1) model uses only the immediate lagged value, while an AR(2) model uses the two most recent lagged values.

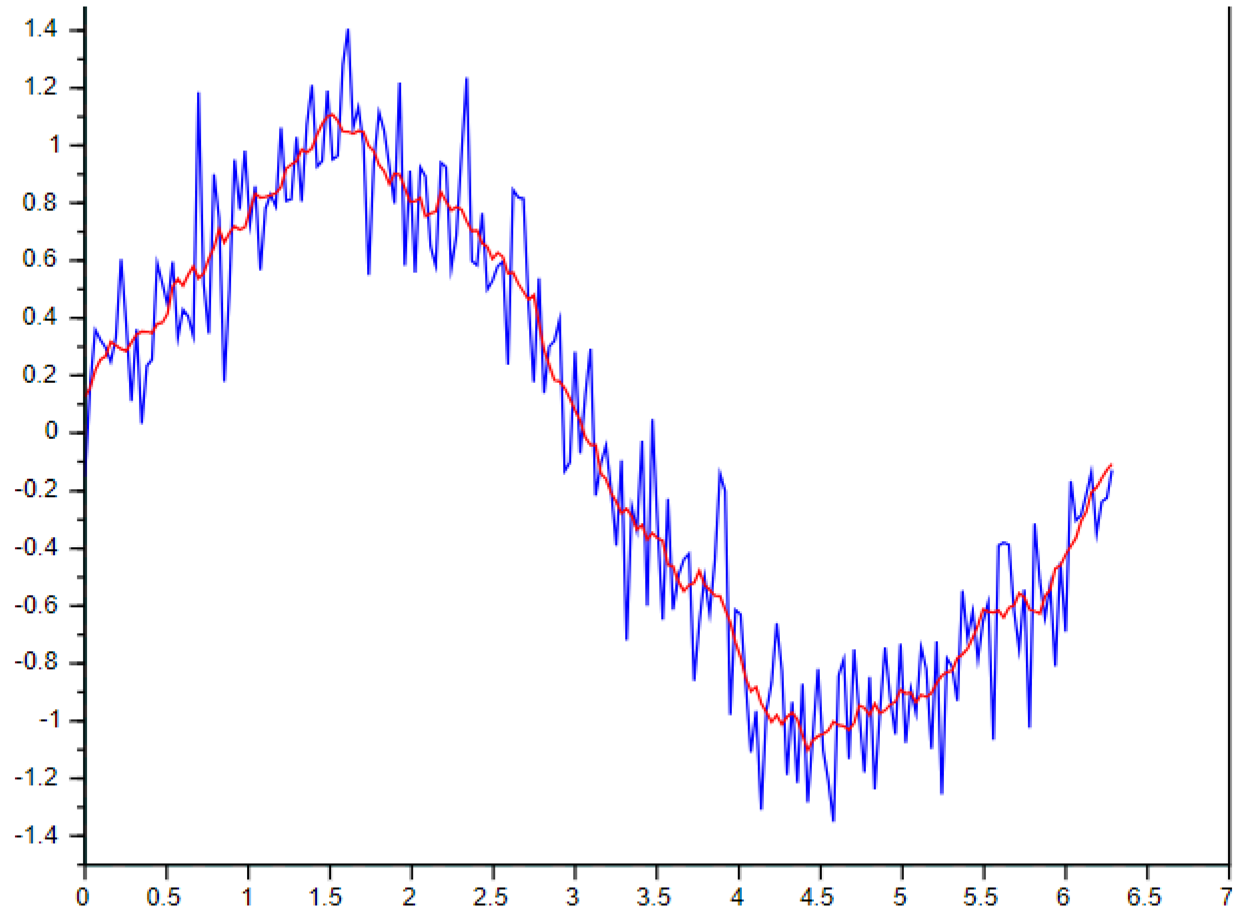
Autoregressive model

- Autoregressive models are commonly used in time series analysis to model and forecast data that exhibit temporal dependencies
- They have been applied in various fields, including economics, finance, weather forecasting, and signal processing
- The selection of the appropriate order p for an autoregressive model is often determined through statistical methods, such as the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC)

Autoregressive model

- It's worth noting that autoregressive models assume stationarity, which means that the statistical properties of the time series remain constant over time
- If stationarity is not present, pre-processing techniques like differencing may be applied to transform the data before fitting an autoregressive model

Moving average model



Moving average model

- The notation $MA(q)$ refers to the moving average model of order q:

$$X_t = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t,$$

where μ is the mean of the series, the $\theta_1, \theta_2, \dots, \theta_q$are the parameters of the model and and ϵ_t is a white noise. The value of q is called the order of the MA model

Moving average model

- A moving average model, often referred to as an MA model, is a type of statistical model used to analyze and forecast time series data
- Unlike autoregressive models that focus on the relationship between past values of the variable, MA models emphasize the relationship between the current value and the error terms from previous time periods.

Moving average model

- In an MA model, the value at a given time is expressed as a linear combination of the error terms from the current and previous time periods
- The order of an MA model, denoted by "q," represents the number of lagged error terms included in the model.

Moving average model

- Similar to autoregressive models, the appropriate order q for an MA model is typically determined through statistical methods such as the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC).

Moving average model

- MA models are often used in conjunction with autoregressive models to create more comprehensive models known as autoregressive moving average (ARMA) models or autoregressive integrated moving average (ARIMA) models
- These models combine the autoregressive and moving average components to capture both the temporal dependencies and the error structure in the data.

Moving average model

- MA models are widely used in time series analysis and forecasting, particularly in finance, economics, and signal processing
- They provide valuable insights into the short-term dynamics and trends present in time series data.

White noise

- Time series, usually denotated as $\epsilon_t, t \in \mathbb{N}$
- Formally, a time series $\{\epsilon\}_t$ is considered white noise if it satisfies the following properties:
  - Constant mean: The expected value of each observation is zero (E[ϵ_t] = 0).
  - Constant variance: The variance of each observation is constant (Var[ϵ_t] = σ^2).
  - No autocorrelation: The correlation between any two observations at different time points is zero (Corr[ϵ_t, ϵ_s] = 0 for t ≠ s).

White noise

- White noise, in the context of time series analysis, refers to a type of random signal or stochastic process that has a constant mean and variance and no autocorrelation at any lag
- In other words, it is a sequence of uncorrelated random variables with zero mean and constant variance

White noise

- In a Time series, white noise represents the random component that cannot be explained by any predictable pattern or trend
- It is characterized by randomness and unpredictability. Each observation in a time series affected by white noise is independent of all other observations.

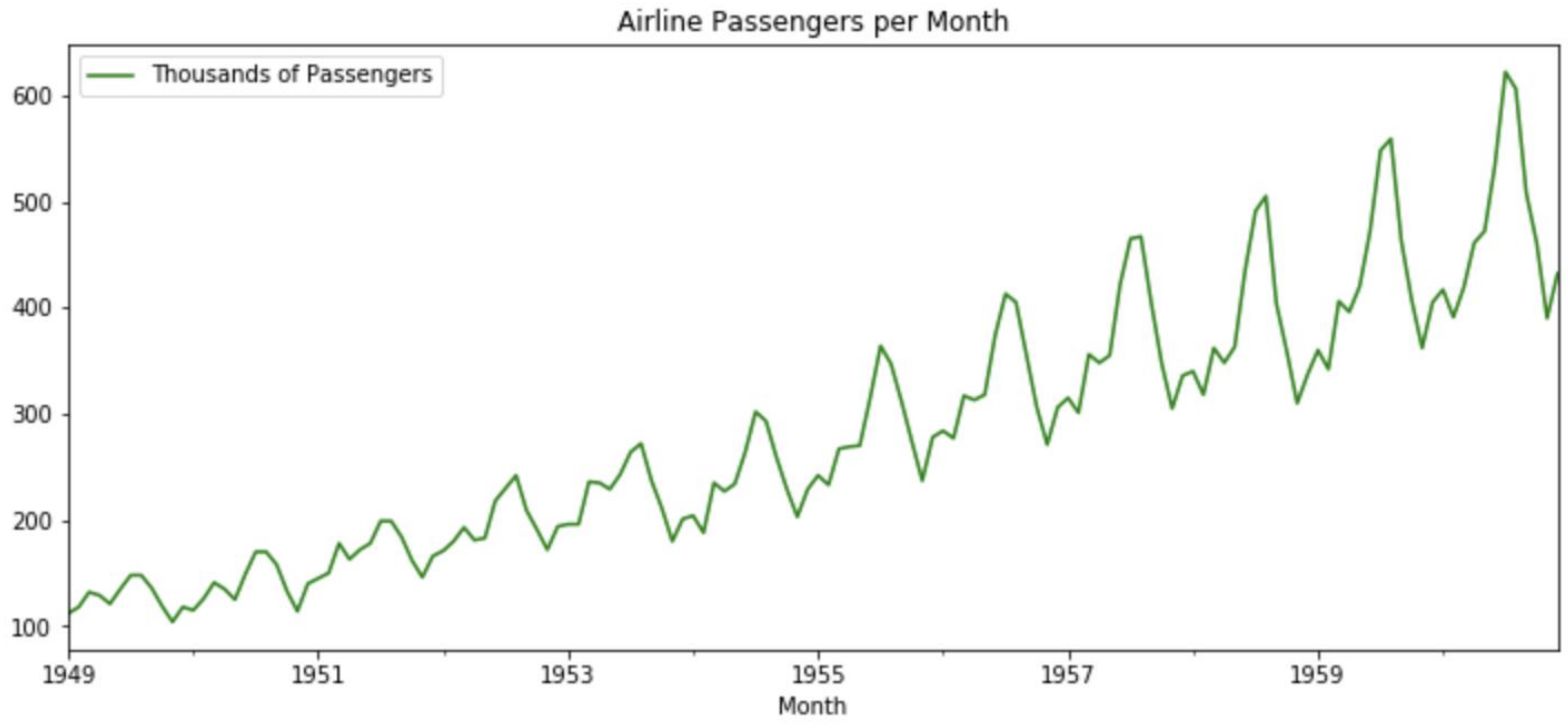
White noise correlation – illusation in Python

```
import numpy as np

# Generate white noise
np.random.seed(42)
white_noise = np.random.normal(0, 1, size=1000)

# Calculate correlation coefficient
correlation = np.corrcoef(white_noise[:-1], white_noise[1:])[0, 1]
print("Correlation coefficient:", correlation)
```

ARIMA model

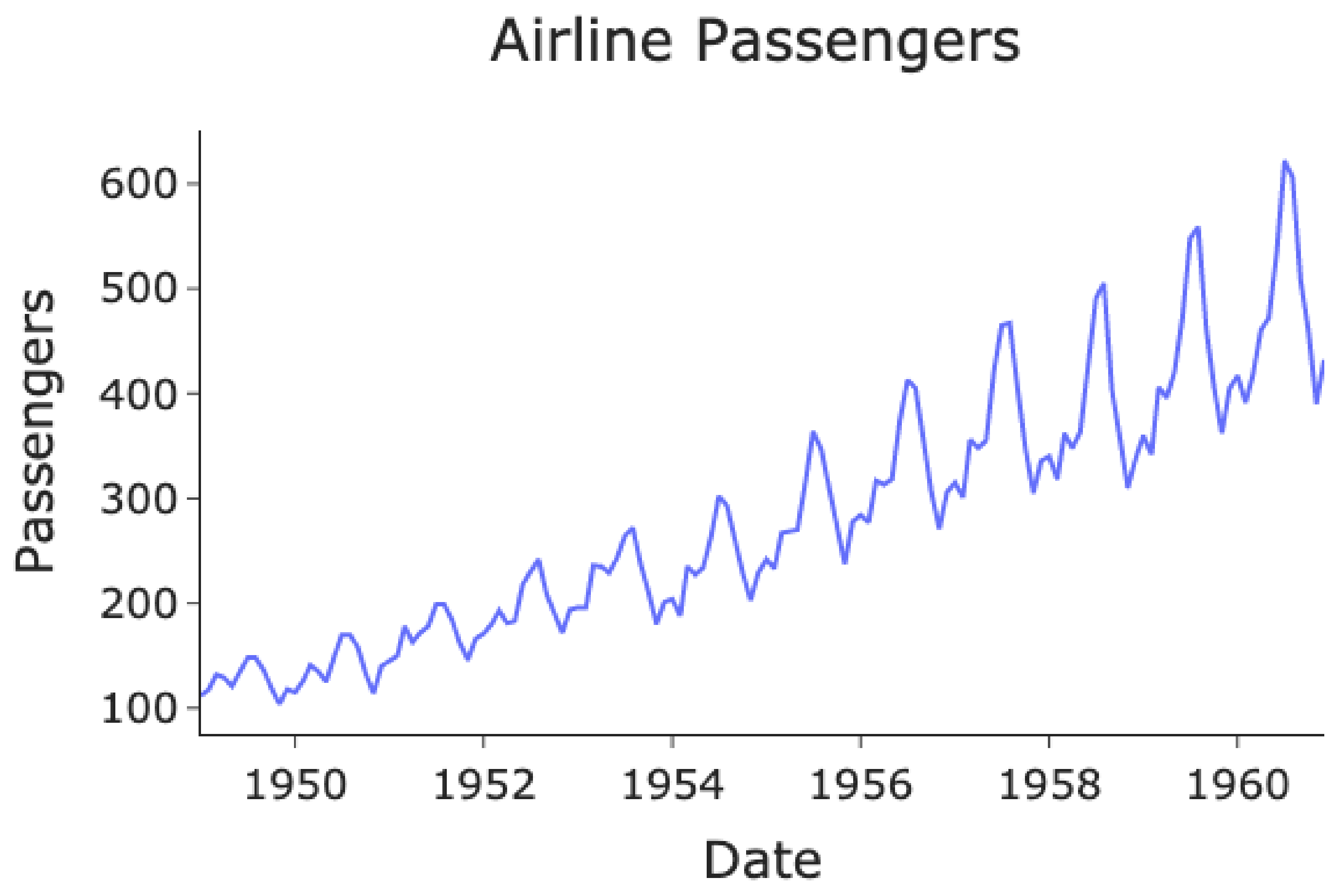

Airline Passengers per Month
Thousands of Passengers
600
500
400
300
200
100
1949
1951
1953
1955
1957
1959
Month

ARMA(p, d) model

- A mixed autoregressive moving average process of order (p, q), an ARMA(p, q) process, is a
- stationary process $\{X\}_t$ which satisfies the relation

$$X_t = \mu + \sum_{i=1}^{p} \phi_i (X_{t-i} - \mu) + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t,$$

Seasonality



https://towardsdatascience.com/seasonality-of-time-series-5b45b4809acd

Removing seasonality

- We can remove seasonality in the data using differencing, which calculates the difference between the current value and its value in the previous season. The reason this is done is to make the time series stationary rendering its statistical properties constant through time. Seasonality causes the mean of the time series to be different when we are in a particular season. Hence, its statistical properties are not constant.

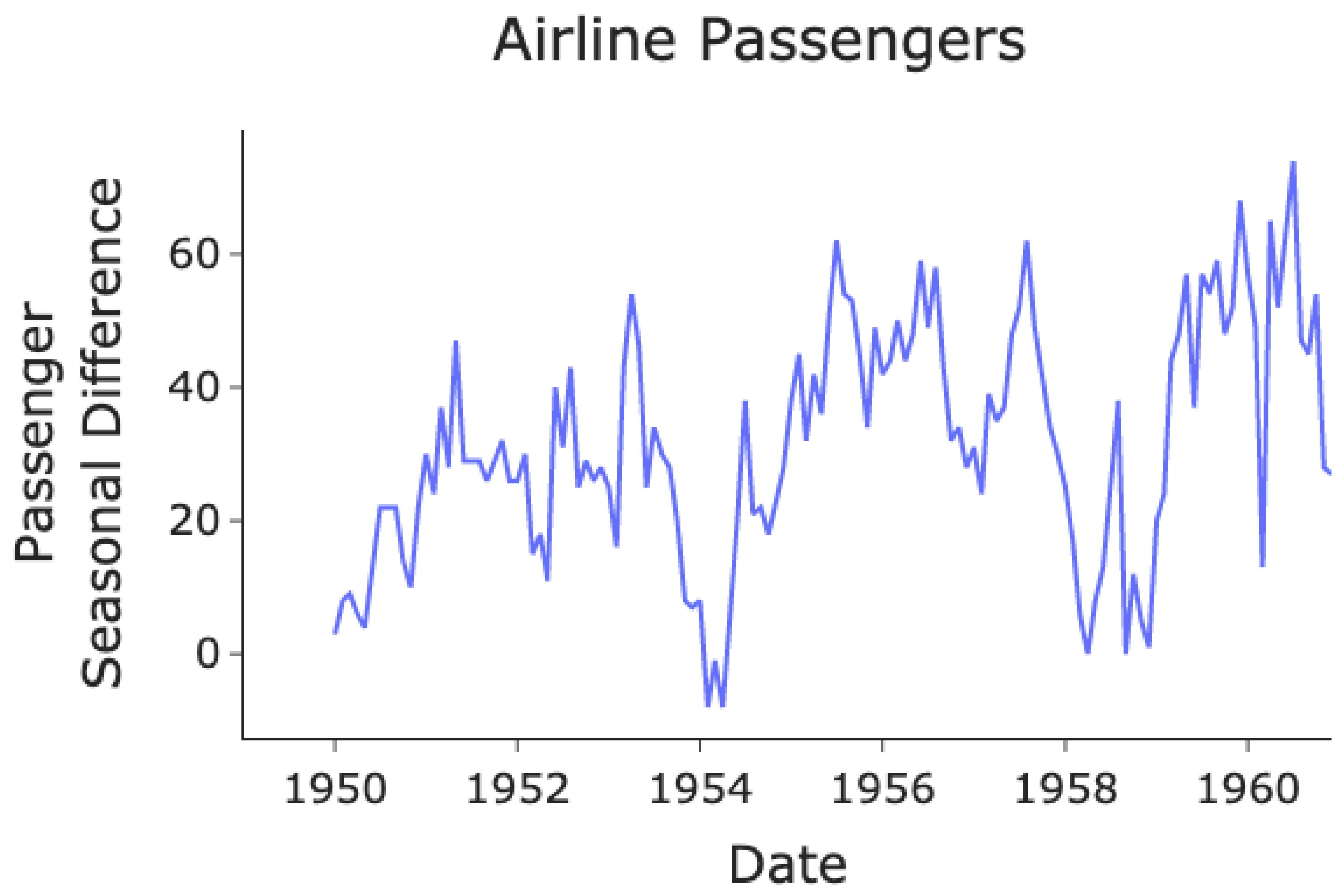
Removing seasonality

- Seasonal differencing is mathematically described as:

$$d(t) = y(t) - y(t - m)$$

Where ***d(t)*** is the differenced data point at time ***t***, ***y(t)*** is the value of the series at ***t***, ***y(t-m)*** is the value of the data point at the previous season and ***m*** is the length of one season. In our case ***m=12*** as we have yearly seasonality.

Removing seasonality



Cycle

- The yearly seasonality has disappeared now, however we now observe some cycle. This is another common feature time series which is similar to seasonality but are typically on a longer timescale as observed here.
- We can test that the resultant series is stationary using the Augmented Dickey-Fuller (ADF) test. The null hypothesis of this test is that the series is non-stationary.

Cycle test

```
# Import adf test
from statsmodels.tsa.stattools import adfuller


def adf_test(series):
    """Using an ADF test to determine if a series is stationary"""
    test_results = adfuller(series)
    print('ADF Statistic: ', test_results[0])
    print('P-Value: ', test_results[1])
    print('Critical Values:')
    for thres, adf_stat in test_results[4].items():
        print('\t%s: %.2f' % (thres, adf_stat))


adf_test(data["Passenger_Season_Diff"][12:])
```

ACF - autocorrelation function

- The mean of a time series $X_1, X_2, \dots, X_n$ is

$$\sum_{i=1}^{n} \frac{X_i}{n} = \bar{X},$$

The **autocovariance function** at lag k, for k ≥ 0, of the time series is defined by

$$s_k = \sum_{i=k+1}^{n} \frac{(X_i - \bar{X})(X_{i-k} - \bar{X})}{n}$$

ACF - autocorrelation function

- The **autocorrelation function (ACF**) at lag k, for k ≥ 0, of the time series is defined by

$$r_k = \frac{s_k}{s_0} = \frac{s_k}{VAR(X_t)}$$

Time series - Python

Python offers several libraries for time series analysis, including:

- pandas: pandas provides powerful data manipulation and analysis capabilities, making it suitable for handling time series data.
- NumPy: NumPy offers a range of mathematical functions and tools that can be applied to time series data.
- statsmodels: statsmodels is a library focused on statistical modeling, including time series analysis and forecasting.
- scikit-learn: scikit-learn provides machine learning algorithms that can be applied to time series data for tasks such as regression and classification.

Time series - Python

- Additionally, there are specialized libraries like Prophet, developed by Facebook, which simplifies time series forecasting with intuitive APIs.
- Time series analysis is a vast and complex field, with many advanced techniques and models beyond the scope of this introduction. However, the mentioned concepts provide a starting point for understanding and exploring time series data.

Examples (pandas)

```
import pandas as pd
# Create a DataFrame with time series data
data = {
   'date': ['2021-01-01', '2021-01-02', '2021-01-03'],
   'value': [10, 20, 15]
}
df = pd.DataFrame(data)
# Convert the 'date' column to datetime type
df['date'] = pd.to_datetime(df['date'])
# Set 'date' as the index of the DataFrame
df.set_index('date', inplace=True)
# Resample the data to a monthly frequency
monthly_data = df.resample('M').mean()
print(monthly_data)
```

In this example, we create a DataFrame with two columns: 'date' and 'value'. We convert the 'date' column to a datetime type using `pd.to_datetime()` and set it as the index of the DataFrame using `set_index()`. Finally, we resample the data to a monthly frequency using `resample()` and calculate the mean value for each month.

Examples (NumPy)

```
import numpy as np

# Create a NumPy array with date range
dates = np.arange('2021-01', '2022-01',
dtype='datetime64[M]')

# Generate random values for the time series
values = np.random.randn(len(dates))

print(dates)
print(values)
```

In this example, we use NumPy's `arange()` function to create a range of dates from January 2021 to December 2021 at monthly intervals. We specify the `dtype` parameter as `datetime64[M]` to ensure that the dates are represented as monthly intervals. We also generate random values for the time series using NumPy's `random.randn()` function.

Examples (matplotlib)

```
import matplotlib.pyplot as plt
import pandas as pd
# Create a DataFrame with time series data
data = {
    'date': pd.date_range(start='2021-01-01', periods=365),
    'value': np.random.randn(365).cumsum()
}
df = pd.DataFrame(data)
# Plot the time series
plt.plot(df['date'], df['value'])
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Time Series Plot')

plt.show()
```

In this example, we create a DataFrame with a 'date' column generated using `pd.date_range()` and a 'value' column with random cumulative sum values