High Integrity Systems Project
Time Series Analysis

Chitra Khatri
Hellyben Shah
Aniket Nighot
Hardik Kunt

October 28, 2024

1 What is *Time Series* ?

A time series represents a collection of values obtained from sequential measurements over time. There are two key features that distinguish time series from other data types:

- **Time Attribute**: Every variable's records need to be chronologically ordered and have a time dimension. Some data types, such as market basket data, do not possess such attributes [1].
- **Sequence Attribute**: The record values are continuous over a certain period and follow specific laws.
- **Types of Time Series** : Time series can be divided into univariate time series (UTS) and multivariate time series (MTS) based on the number of variables.

The specific definitions are as follows :

A time series S is defined as:

$$S = \{v_i(t) | i = 1, 2, \ldots, m; t = 1, 2, \ldots, n\}$$

where: - t (where $t = 1, 2, \ldots, n$) represents time, - i (where $i = 1, 2, \ldots, m$) represents variables, - $v_i(t)$ denotes the record of variable i at time t.

When $m = 1$, S is defined as a univariate time series (UTS). When $m > 1$, S is defined as a multivariate time series(MTS) [2] [1].

2 Summary:*Deep Learning for Time Series Cookbook* Chapter 1 [3]

This chapter introduces time series data analysis using a solar radiation dataset collected by the U.S. Department of Agriculture. The dataset contains hourly observations from October 1, 2007, to October 1, 2013, with a total of 52,608 data points.

A) Univariate Time Series

A univariate time series records changes over time in a single variable. An example would be temperature readings taken every second from a sensor, which would yield a single value (temperature) per second.

Data Loading and Visualization

- **Data Loading:** Load the dataset as a CSV using `pandas`. For a univariate time series, it is stored as a `Series` object.
- **Data Visualization:** Time series data visualization with pandas and seaborn .
  - `Pandas:` The plot() method in pandas offers a quick and straightforward way to visualize time series data, making it easy to generate basic plots directly from a DataFrame..

– `Seaborn:` A dedicated data visualization library in Python, provides a more advanced toolkit for creating detailed and aesthetically pleasing plots, ideal for enhancing the visual appeal and depth of data analysis..

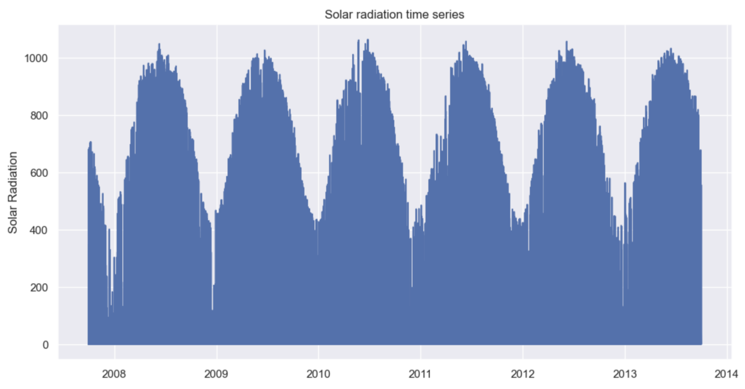

Figure 1: Time series plot using seaborn

Key Recipes for Cooking Time Series

1. Resampling a Time Series

Changing the frequency of a time series can involve:

- **Downsampling**: Aggregating, e.g., from hourly to daily.
- **Upsampling**: Interpolating, e.g., from daily to hourly.

The majority of techniques used in time series analysis operate on the premise that the data is collected at consistent time intervals, such as daily measurements. This is what we refer to as a regular time series. However, it's important to note that not all time series conform to this pattern. Some datasets are inherently irregular in nature. A prime example of this is retail sales data, where transactions occur at random times throughout the day as customers make purchases. These irregular patterns create challenges for traditional time series analysis methods.

When dealing with irregular time series, resampling can:

- Transform irregular time series into regular ones.
- Adjust the granularity of the data.

a. Changing Time Granularity

If we have a time series with hourly granularity but wish to analyze it daily, we can resample it to daily granularity using aggregation functions such as *sum* or *mean*.

b. Converting an Irregular Time Series to a Regular One

For an irregular time series, resampling can involve aggregation (e.g., using *count* or *mean*) to convert it into a regular series.

2. Dealing with Missing Values

Missing values in time series data, often due to sensor issues or errors, can be imputed by:

- Using mean or median values for non-temporal data.
- Applying temporal imputation, such as filling missing values with the last known observation, to maintain the temporal dynamics of the data.

3. Decomposing a Time Series

Process of splitting a time series into its basic components, such as trend or seasonality. Breaking a time series into its components is useful to understand the underlying structure of the data. Decomposition splits a time series into basic components:

- **Additive model**:
$$Y_i = \text{Trend}_i + \text{Seasonality}_i + \text{Remainder}_i$$
- **Multiplicative model**:
$$Y_i = \text{Trend}_i \times \text{Seasonality}_i \times \text{Remainder}_i$$

a. Classical Decomposition

The trend is estimated using a moving average, such as a 24-hour average, and seasonality is determined by averaging values for each period.

b. Seasonal and Trend Decomposition Using LOESS (STL)

A more adaptable technique for decomposing a time series is STL. Complex patterns, including irregular trends or outliers, can be handled using it. To extract each component, STL uses LOESS, or locally weighted scatterplot smoothing.

c. Multiple Seasonal and Trend Decomposition Using LOESS (MSTL)

Multiple seasonal patterns can be found in time series that are gathered at high sampling frequency, like hourly or daily. An hourly time series, for instance, may demonstrate regular daily and weekly both variations.

Decomposition is often used for exploring data but can also serve as a pre-processing step for forecasting. Studies show that removing seasonality before training neural networks can improve forecasting accuracy

Key Components of a Time Series

- **Trend**: Long-term directional movement in data.
- **Seasonality**: Recurring patterns over fixed periods, such as daily or yearly cycles.
- **Remainder**: The irregular component left after removing the trend and seasonality.

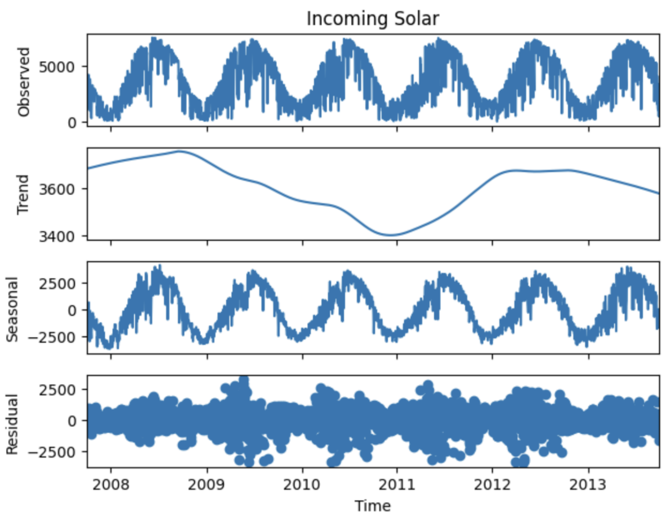

Figure 2: Time series components after Decomposition

4. Computing Autocorrelation

Correlation is a statistical measure that assesses the linear relationship between two random variables. Autocorrelation builds on this concept for time series data, indicating that the value at a specific time point tends to be similar to those at previous time points. The autocorrelation function measures the linear relationship between a time series and a version of itself that has been shifted back by a certain number of periods, known as a lagged time series.

Autocorrection score ranges from -1 to +1 similar to correlation. +1 represents perfect positive correlation and -1 represents perfect negative correlation.The analysis of autocorrelation is a useful approach to detecting seasonality.

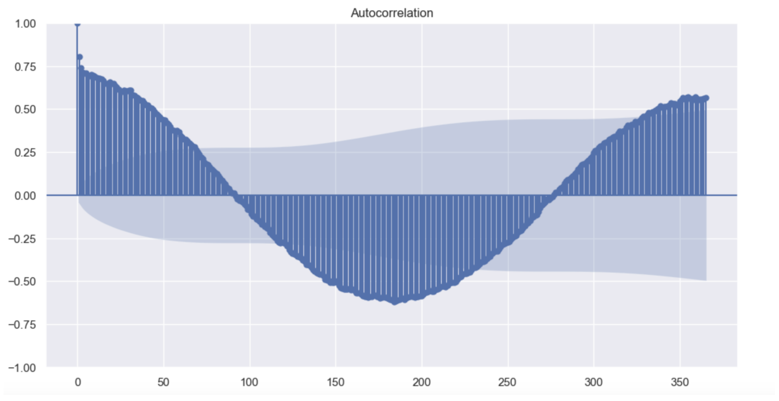

Figure 3: Autocorrelation scores up to 365 lags.

5. Stationarity

Stationarity is essential for many modeling techniques. A stationary time series has constant mean and variance. Non-stationary series can be transformed using **differencing**, where consecutive values are subtracted:

- **Step A**: Determine necessary differencing using tests like the Augmented Dickey-Fuller (ADF) test or the KPSS test.

- **Step B**: Apply differencing to achieve stationarity.

Once the series is stationary, a forecasting model can be developed based on the differenced data. The resulting forecasts can be converted back to the original scale by reversing the differencing. This approach highlights the importance of achieving stationarity for effective time series modeling and forecasting.

6. Dealing with Heteroskedasticity

In time series analysis, heteroskedasticity—where the variance changes over time—can make data challenging to model accurately, especially for machine learning methods like neural networks that perform best with stable data. Detecting heteroskedasticity typically involves statistical tests such as the White and Breusch-Pagan tests, which assess if a dataset's variance remains consistent. If these tests a low p-value indicates that variance is indeed fluctuating, which can hinder model performance by introducing unpredictable extremes.

A common approach to stabilizing this variance is to use a logarithmic transformation, which reduces the impact of larger values and helps keep the data's spread more consistent. In Python, a LogTransformation class can apply this transformation to the data and then reverse it later if needed. Applying these transformations can make data easier to model, especially for neural networks, as it brings the data closer to a normal distribution.

B) Multivariate Time Series

A multivariate time series includes multiple interrelated variables that evolve over time. Modeling these relationships can be complex, especially with many variables. For example, in weather forecasting, solar radiation is linked to variables like temperature and humidity. A comprehensive multivariate model can capture these dependencies, enhancing simulation accuracy and forecasting.

- **Data Loading**: The main difference in loading univariate and multivariate time series is that a multivariate time series is stored in Python as a DataFrame object rather than a Series one. Multivariate time series are better structured as pandas DataFrame objects.

- **Data Visulization**: The plot reveals that each variable exhibits a unique distribution pattern, with distinct mean values and levels of variability.

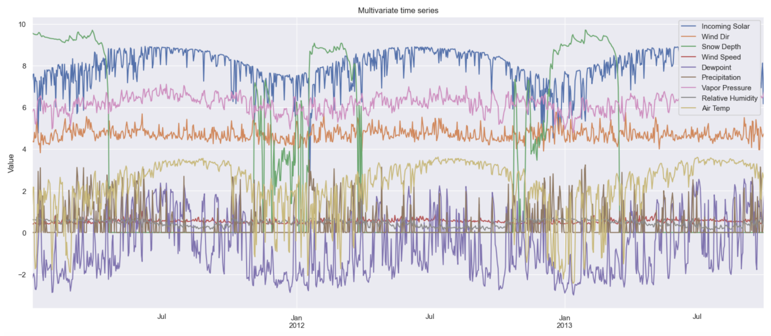

Figure 4: Multivariate time series plot

- **Resampling a multivariate time series**: Depending on the variable, we may need to use different summary statistics while resampling a multivariate time series. For instance, to determine the amount of power we could generate, we may add up the solar radiation measured every hour. Yet, taking the average, instead of the sum, is more sensible when summarizing wind speed because this variable is not cumulative.

- **Analyzing correlation among pairs of variables**: Calculating each pair's correlation is a popular method for examining the dynamics of several variables. For instance, you might only want to keep one variable when two variables have a significant correlation.This can be visualized by heatmap as shown in the picture for solar radiation multivariate data

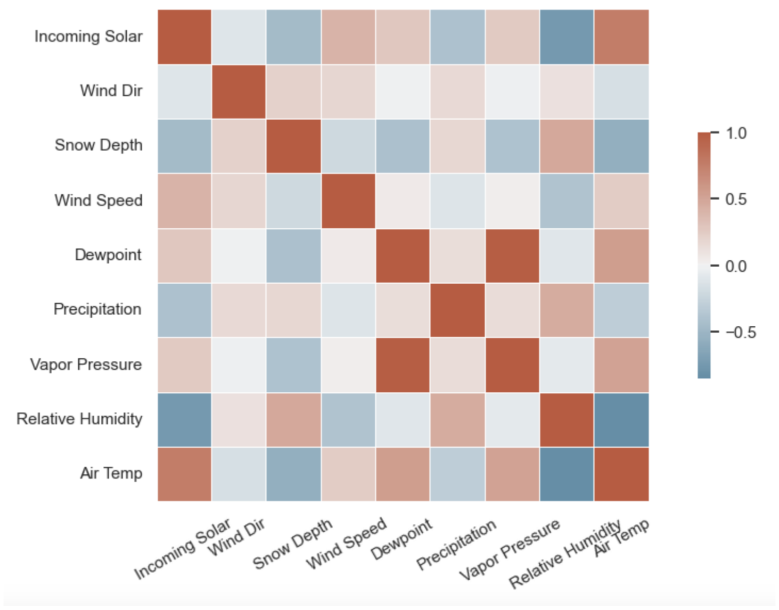

Figure 5: Correlation matrix for a mutivariate time series

References

[1] H.-S. Wu, "A survey of research on anomaly detection for time series," in *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2016, pp. 426–431.

[2] P. Esling and C. Agon, "Time-series data mining," vol. 45, no. 1. ACM New York, NY, USA, 2012, pp. 1–34.

[3] V. Cerqueira and L. Roque, *Deep Learning for Time Series Cookbook: Use PyTorch and Python recipes for forecasting, classification, and anomaly detection.* Packt Publishing Ltd, 2024.