Data Analytics for IoT

ΕΠΛ 428: IOT PROGRAMMING

Dr. Panayiotis Kolios
Assistant Professor, Dept. Computer Science,
KIOS CoE for Intelligent Systems and Networks
Office: FST 01, 116
Telephone: +357 22893450 / 22892695
Web: https://www.kios.ucy.ac.cy/pkolios/

Πανεπιστήμιο Κύπρου

- Real **value** is in IoT data
- However, as more IoT devices added:
    - The data becomes overwhelming (big data)
    - Consume precious network bandwidth
    - Server resources use to store, and process data
- New IoT methods implemented for this need:
    - Big data technologies: Hadoop, NoSQL, MapReduce
    - Edge streaming analytics: real-time, on-device
    - Network analytics: support efficient functionality

Level 5 (full control) autonomous vehicle >50 sensors:

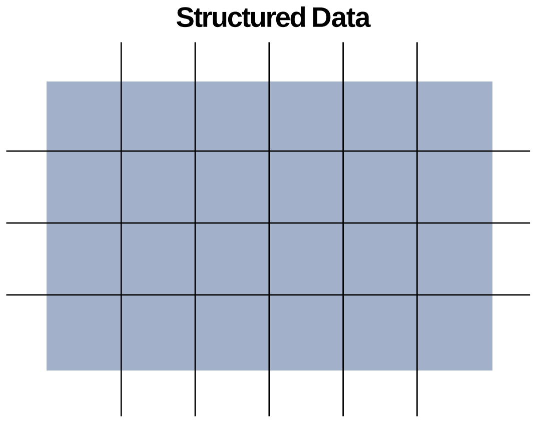
- ultrasonic, surround camera, and long- and short-range radar, long range and stereo cameras, LiDAR, dead reckoning

Sensor	Raw data rate
Radar	**0.1-15 Mbps/sensor**
LIDAR	**20-100 Mbps/sensor**
Camera	**500-3500 Mbps/sensor**
Ultrasonic	**<0.01 Mbps/sensor**

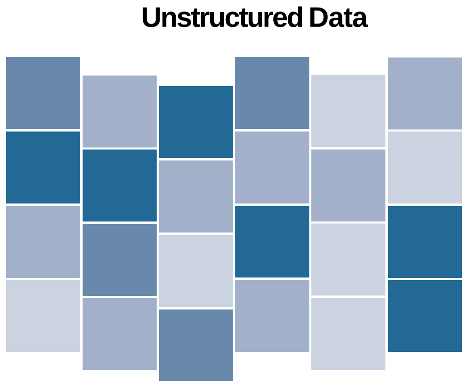
- Approx. 40 Gbps or 19 TB per hour or 5,894 TB per year (50min driving per day)
- 1 billion cars globally -> **589,400 Yottabytes**

https://blogs.sw.siemens.com/polarion/the-data-deluge-what-do-we-do-with-the-data-generated-by-avs/

- Analyzing this amount of data in the most efficient manner possible falls under the umbrella of data analytics
- Data analytics provide knowledge and actionable insights



Organized Formatting (e.g., Spreadsheets, Databases)



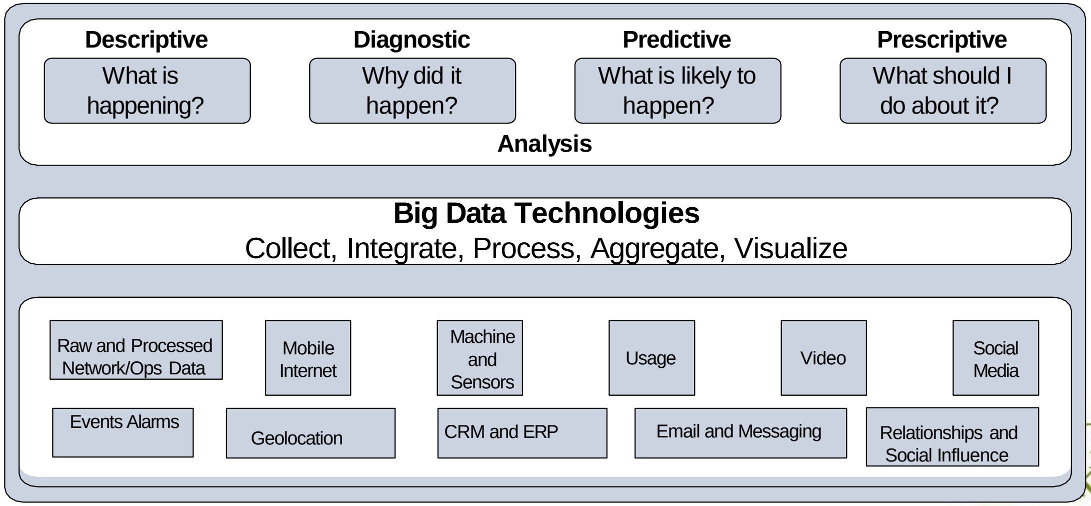
Does not Conform to a Model
(e.g., Text, Images, Video, Speech)

- Structured data follows a model that defines how the data is represented/ organized
    - relational database management system (RDBMS)
    - Structured Query Language (SQL) is used to interact with RDBMS
- 80% is however unstructured data
    - NoSQL do not enforce a strict schema, and they support a complex, evolving data model; **more scalable**

Πανεπιστήμιο Κύπρου

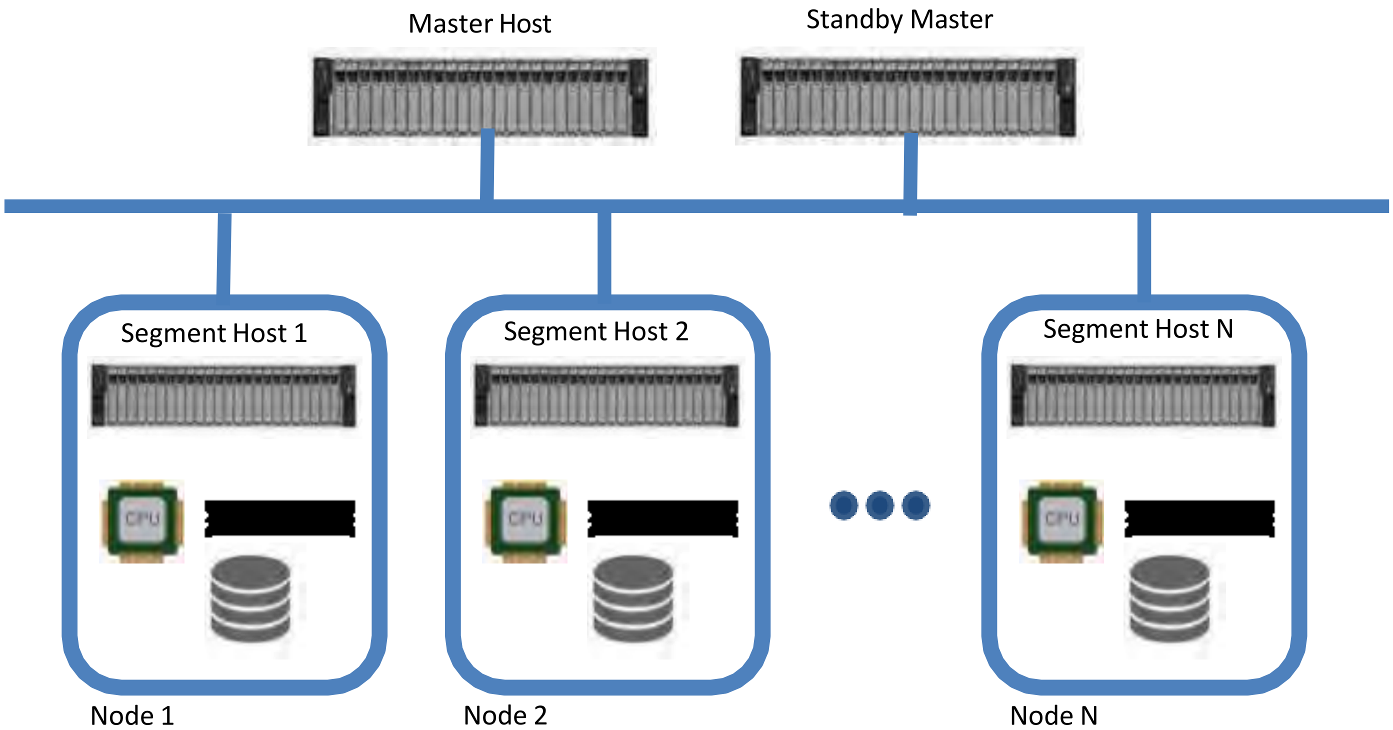
DATA IN MOTION vs DATA AT REST

- IoT data can be transit ("data in motion") or stored ("data at rest")
- Devices generate data that is exchanged (in motion) and acted upon (processed at the edge, fog)
- At the data center, processed also in real-time. Tools with this sort of capability, such as Spark, Storm, and Flink; tools part of Hadoop ecosystem
- Data at rest found in IoT brokers or storage at data center. Myriad tools, for structured data in relational databases, also in Hadoop



- RDBMS used for storing structured data in data warehouses
    - used for longer-term archiving and data queries (take minutes or hours to respond)
- Massively parallel processing (MPP) databases are designed to be much faster, to be efficient, and to support reduced query times
- MPP take advantage of multiple nodes (computers) designed in a scale-out architecture with data processing distributed across multiple systems
- MPPs designed to allow for fast query processing and often have built-in analytic functions
- MPP architecture typically contains a single master node that is responsible for the coordination of all the data storage and processing across the cluster

Πανεπιστήμιο Κύπρου

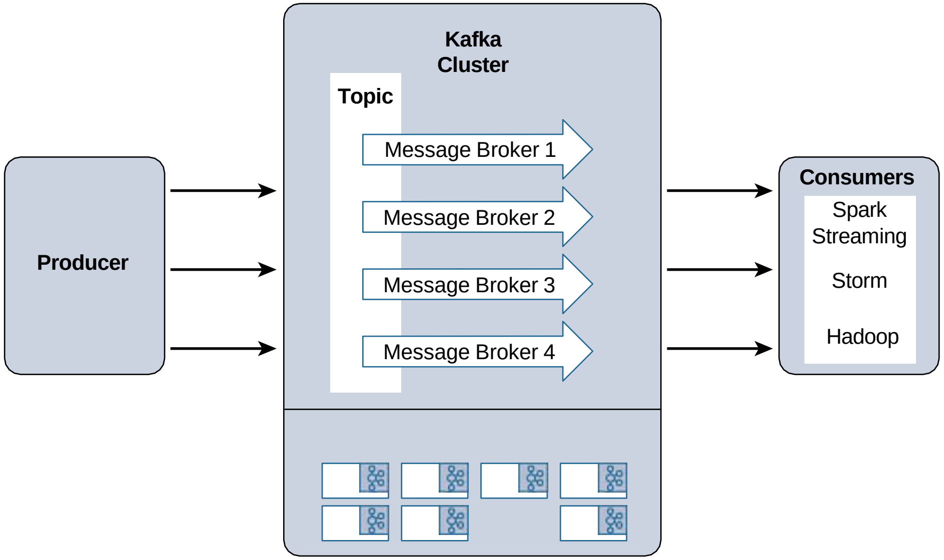


- Nodes have local processing, memory, and storage
- Data storage is optimized across the nodes in a structured SQL-like format that allows data analysts to work with the data using common SQL tools and applications

- NoSQL an umbrella term encompassing different types of databases:
- **Document stores**: Stores semi-structured data, such as XML or JSON
- **Key-value stores**: Store associative arrays where a key is paired with an associated value; easy to build and easy to scale.
- **Wide-column stores**: Similar to a key-value store, but the formatting can vary from row to row, even in the same table
- **Graph stores**: Organized based on the relationships between elements. Graph stores are commonly used for social media or natural language processing, where the connections between data are very relevant
- NoSQL was developed to support the high-velocity, real-time analytics that typically do not require much repeated use

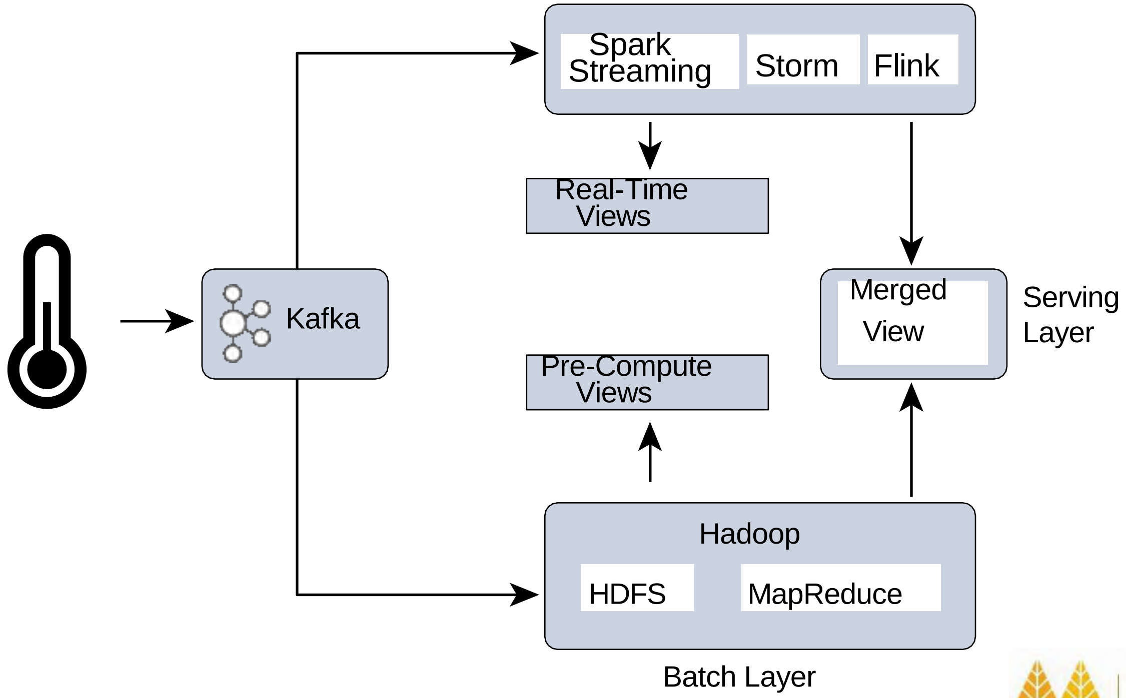
- NoSQL **key-value stores** are capable of handling indexing and persistence simultaneously at a high rate
- Makes them a great choice for time-series data
- Hadoop originally intended to index millions of websites and quickly return search results; had two key elements:
    - **Hadoop Distributed File System** (HDFS): A system for storing data across multiple nodes
        - **NameNodes**: Coordinate storage of data and provide data adds, moves, deletes, reads. Also instruct DataNodes for data replication
        - **DataNodes**: Servers that store data
    - **MapReduce**: A distributed processing engine that splits a large task into smaller ones that can be run in parallel

- Hadoop includes > 100 software packages addressing every element in the data lifecycle:
  - from collection, to storage, to processing, to analysis and visualization
- **Apache Kafka** is a distributed publisher-subscriber messaging system designed to accept data from origin and deliver the data to stream-processing engines; **broker**
  - distributed nature run in a clustered configuration
  - can handle many producers and consumers simultaneously



Πανεπιστήμιο Κύπρου

- Apache Spark
  - An in-memory distributed data analytics platform
  - At each stage of a MapReduce operation processing is moved into memory to lower latency batch processing
- Lambda architecture
  - Querying both streaming and data at rest (batch processing)



Πανεπιστήμιο Κύπρου

- In IoT, vast quantities of data are generated on the fly and often need to be analyzed and responded to in real-time
- High volume live stream IoT data that needs to be analysed to detect patterns or anomalies
    - Volume of data generated at the edge immense— bandwidth requirements to the cloud are very high
    - Time sensitivity precludes waiting for deep analysis in the cloud

Defining Streams and Windows

2016-01-08 04:05:06	Sensor_5	23.45
2016-01-08 04:06:45	Sensor_3	27.22
2016-01-08 04:06:54	Sensor_3	26.89
2016-01-08 04:07:07	Sensor_2	25.01
2016-01-08 04:07:33	Sensor_5	23.00
2016-01-08 04:08:10	Sensor_5	23.02
2016-01-08 04:09:01	Sensor_2	25.02

2016-01-08 04:07:00	Sensor_5	23.45
2016-01-08 04:07:00	Sensor_3	27.06

2016-01-08 04:08:00	Sensor_5	23.00
2016-01-08 04:08:00	Sensor_3	27.06
2016-01-08 04:08:00	Sensor_2	25.01

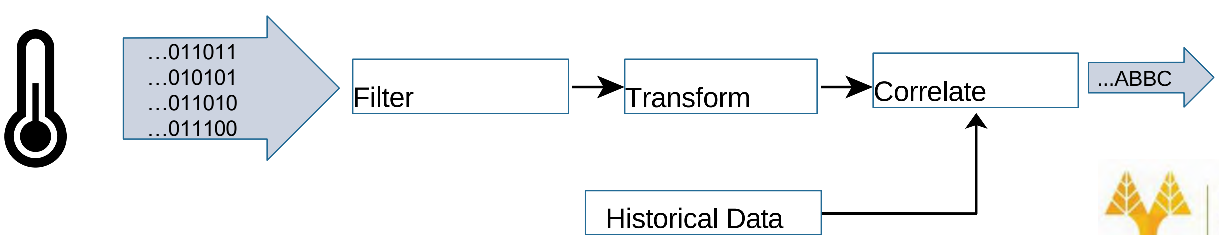
2016-01-08 04:09:00	Sensor_5	23.01
2016-01-08 04:09:00	Sensor_2	25.01

```
CREATE STREAM Temp (
        ts TIMESTAMP CQTIME USER,
        device TEXT,
        temp NUMERIC(5,2)
        );
```

```
SELECT cq_close(*), device, avg (temp)
  FROM Temp <VISIBLE '2 min' ADVANCE '1 min'>
  GROUP BY device;
```

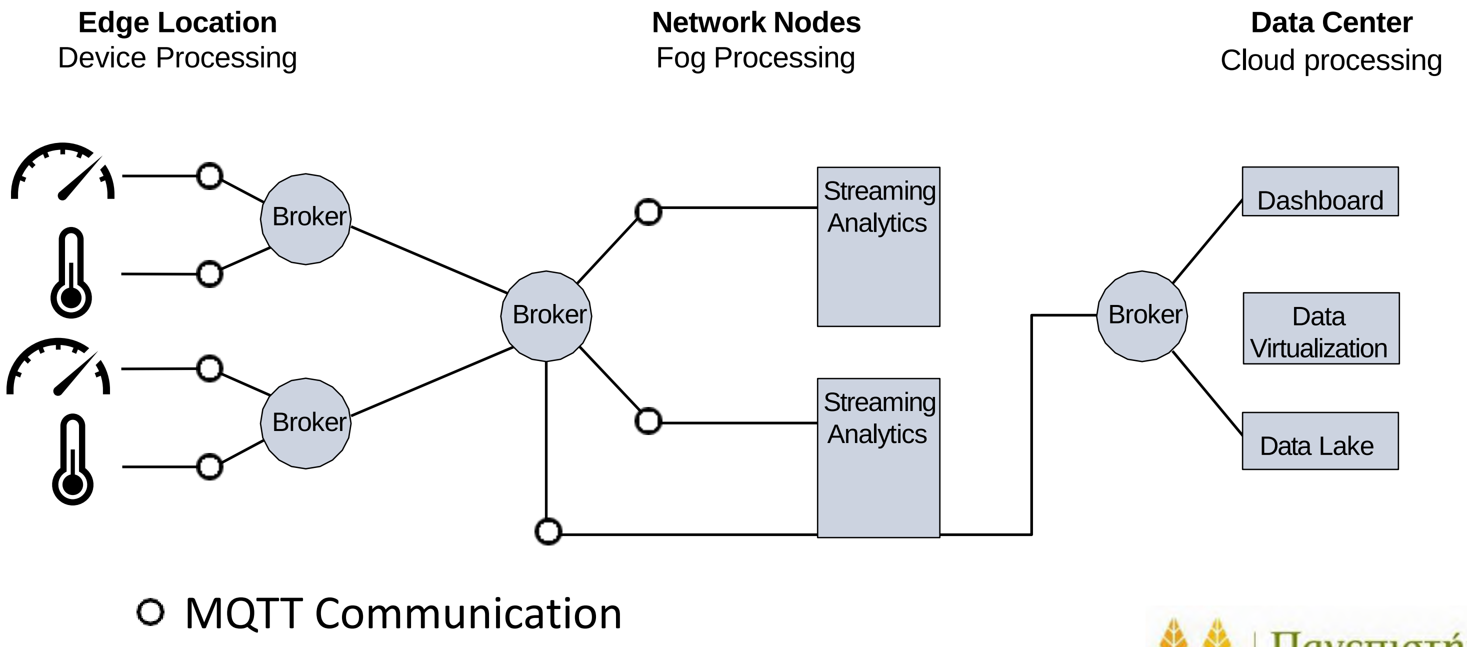
DATA CORRELATION

- Streaming data analytics most useful when multiple data streams are **combined** from different **types of sensors** or **different time periods**
- For example, in a hospital, several vital signs are measured for patients, including body temperature, blood pressure, heart rate, and respiratory rate
  - When this data is combined and analyzed, it provides an invaluable picture of the health of the patient at any given time.
- Alternatively, historical data may include the patient’s past medical history, such as blood test results
  - Combining historical data gives the live streaming data a powerful context and promotes more insights into the current condition of the patient

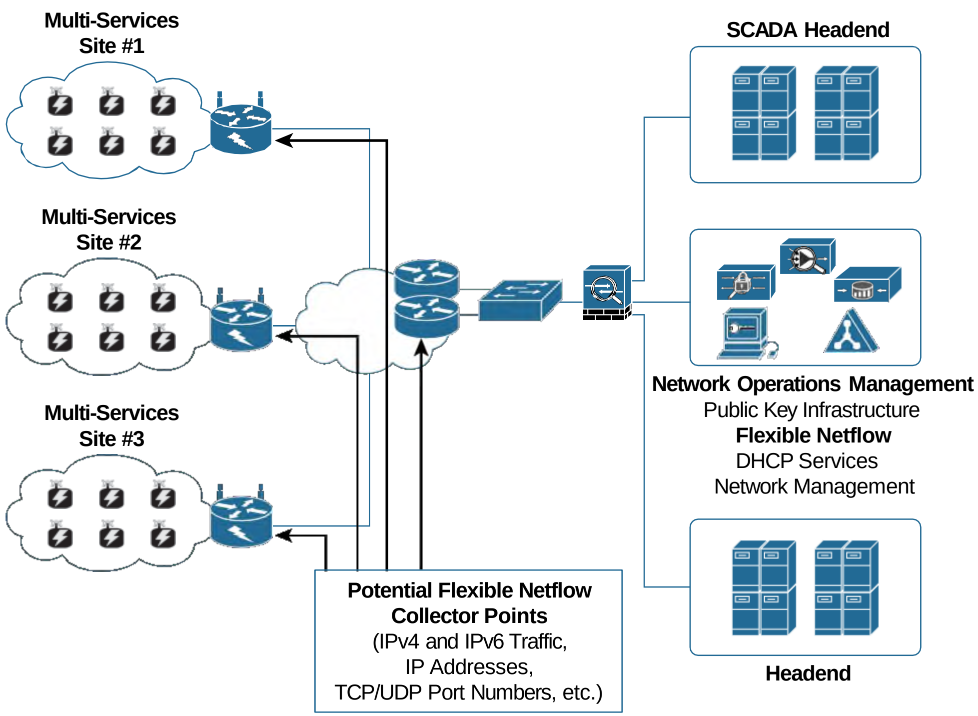


Πανεπιστήμιο Κύπρου

- Streaming analytics may be performed directly at the edge, in the fog, or in the cloud data center
- Value of data when aggregated increases massively
- But there is also value in retracking from the edge to the network to gain a wider understanding of largescale systems
- **Fog analytics** allows to see beyond one device



- Network analytics is concerned with discovering patterns in the communication flows from network traffic
- Analyze details of communications patterns made by protocols and correlate this across the network
- Understand what is normal behavior and identify anomalies
    - Connectivity and routing issues
    - Cyberattacks / Emergency situations



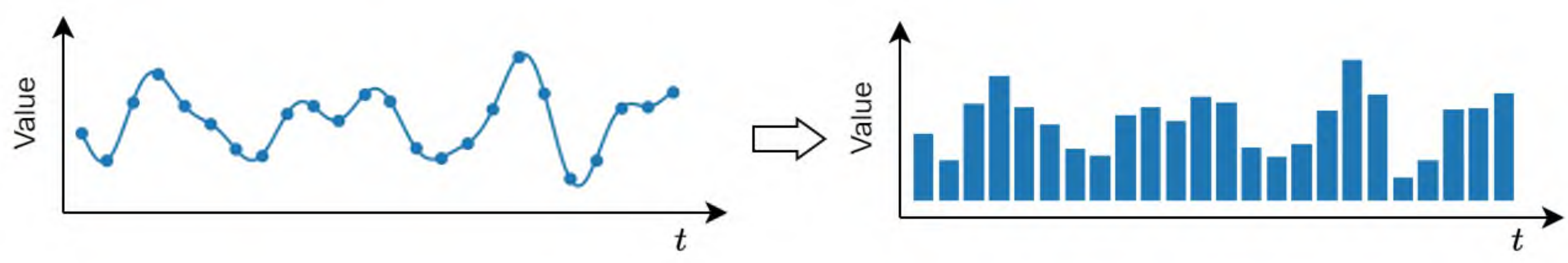
- Enable capabilities to cope with
  - capacity planning for scalability
  - security monitoring
- Drivers of the adoption of standardize architectures
- Flow statistics can be collected:
  - **Network traffic monitoring and profiling**: Flow collection from the network layer provides global and distributed near-real-time monitoring capabilities
  - **Application traffic monitoring and profiling**: Gain detailed time-based view of IoT access services, such as the application-layer protocols, including MQTT, CoAP, and DNP3
  - **Capacity planning**: Used to track and anticipate IoT traffic growth and help in the planning of upgrades
  - **Security analysis**: Generate low volumes of traffic typically and always send their data to the same server(s), any change in network traffic behavior may indicate a cybersecurity event
  - **Accounting**: Field networks are often physically isolated and leverage public cellular services and VPNs for backhaul. Traffic can be leveraged to optimize the billing

INTRO TO TIMESERIES ANALYSIS

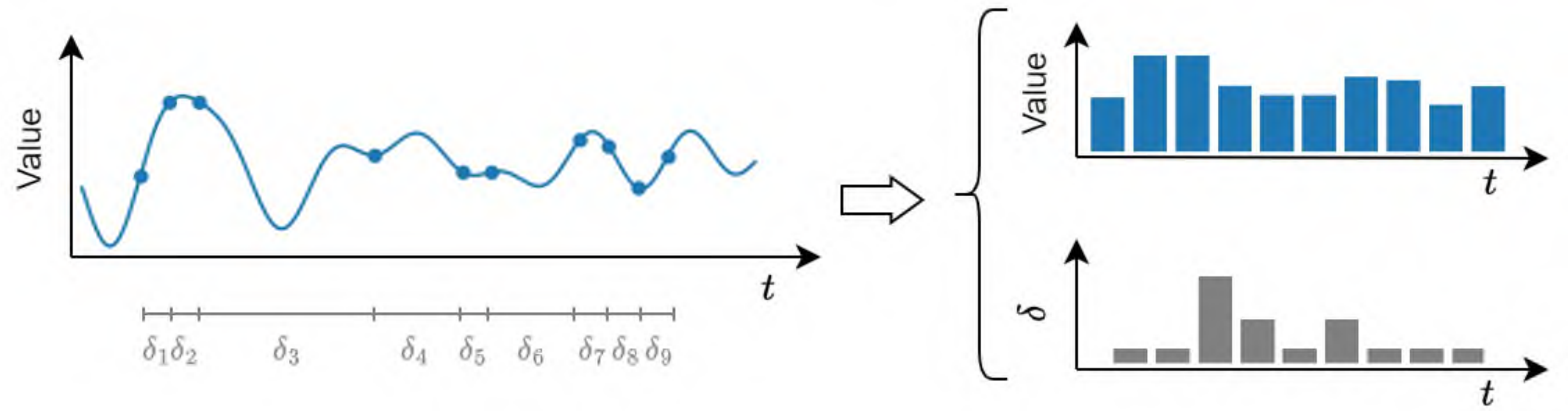
- Time series analysis provides insights into patterns over time that are invaluable
    - Understanding dynamics
    - Detecting events
    - forecasting
- Provide understanding of
    - Theoretical concepts
    - Translation to functional tools

What is a time series?

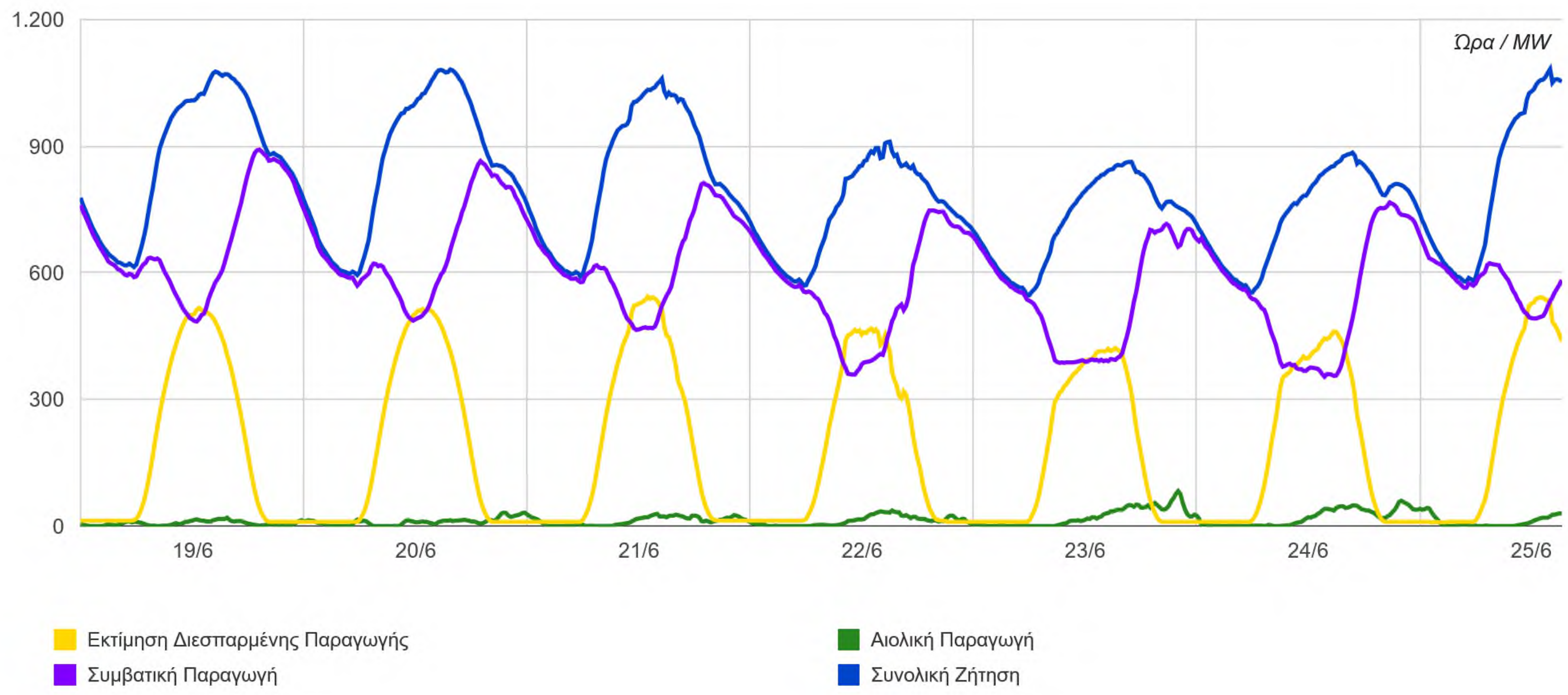
- A time series is a sequence of data points
- Usually, signal is sampled at given frequency
- Represented as sequence of sampled values



- Irregularly sampled signals are time series encoded with additional information stored into a data structure



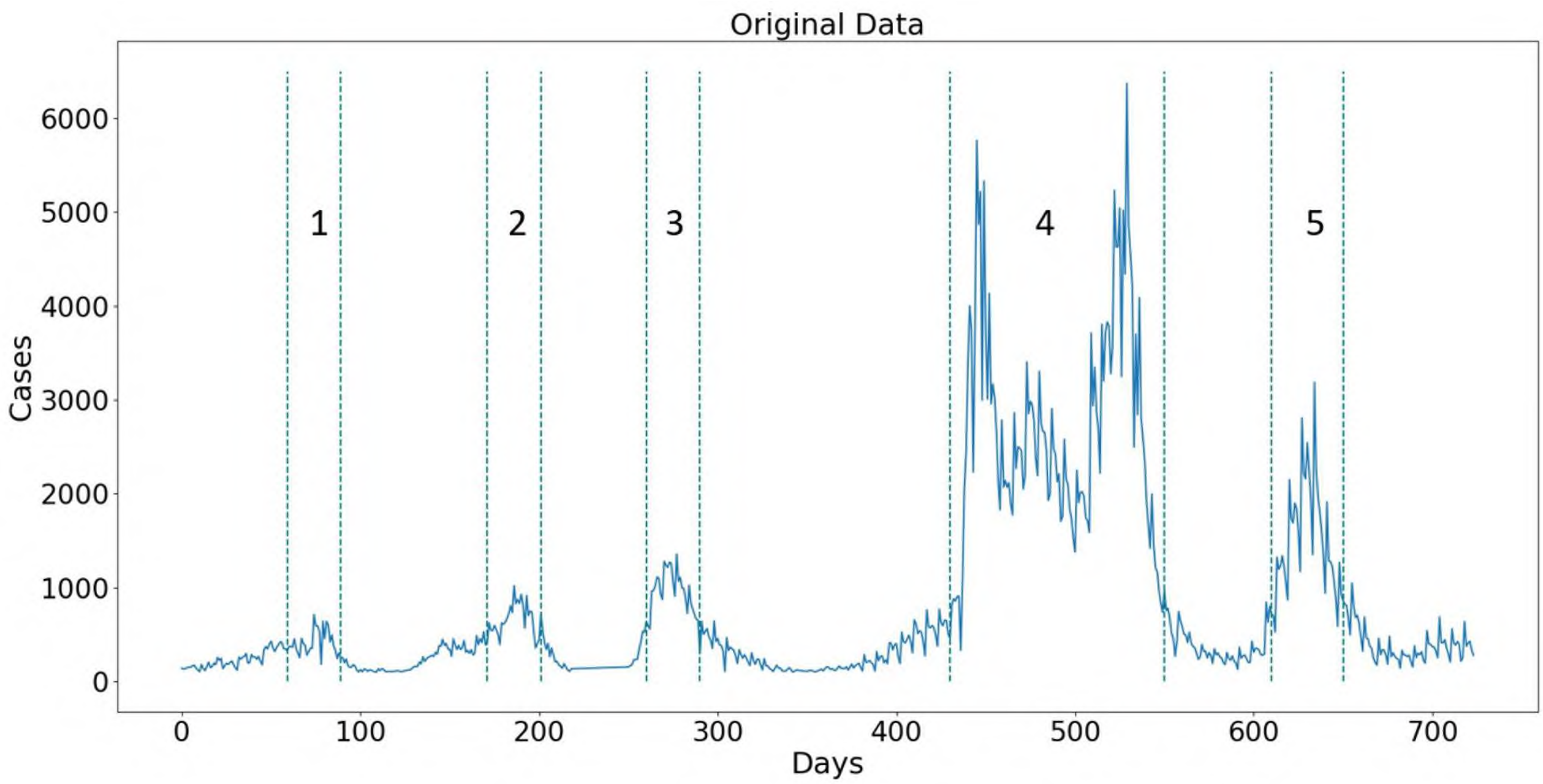
TIME SERIES DATA



https://tsoc.org.cy/electrical-system/archive-total-daily-system-generation-on-the-transmission-system/?startdt=19-06-2024&enddt=%2B15days

Πανεπιστήμιο Κύπρου

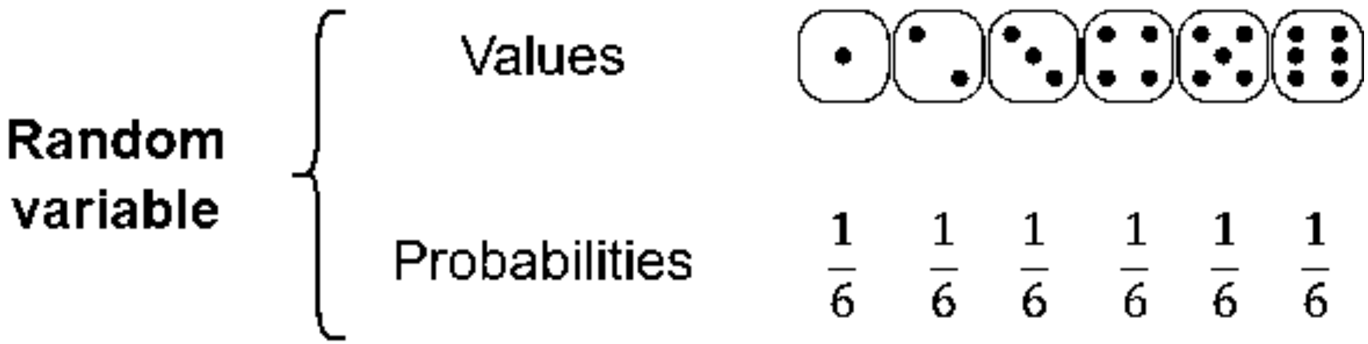
- COVID-19 Positive Cases (15/10/20 -18/10/22)



Πανεπιστήμιο Κύπρου

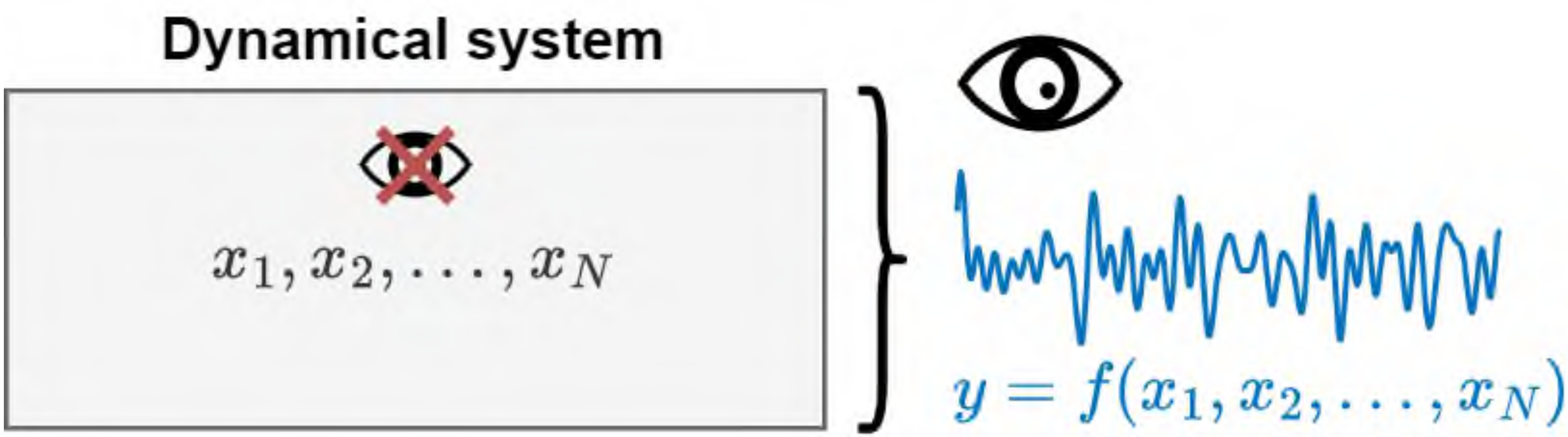
- The main objective of time series analysis is:
  - To understand and characterize the underlying process that generates the observed data.
  - To forecast the evolution of the process, i.e., predict the next observed values.
- Two main perspectives to look at a time series leading to different analysis approaches
  - Statistical
  - Dynamical systems

- Assumed to be a sequence of random variables that have some correlation or part of a distribution
- Sequence is a realization (observed values) of a stochastic process
- Statistical time series approaches focus on finding the parameters of the stochastic process that most likely produced the observed time series



Πανεπιστήμιο Κύπρου

- Assume that there is a system governed by unknown variables
- Observe one time series generated by the system
    - Unknown variable of the system
    - Unknown function of the system
- Objective is to reconstruct the dynamics of the system



- Composed of three parts:
  - Trend: the long-term direction
  - Seasonality: the periodic behavior
  - Residuals: the irregular fluctuations
- Trend captures the general direction of the time series
  - For example, increasing number of passengers over the years despite seasonal fluctuations
- Trend can be increasing, decreasing, or constant
- It can increase/decrease in different ways over time (linearly, exponentially, etc)

- Periodic fluctuations in time series data that occur at regular intervals due to seasonal factors
- Consistent / predictable patterns over a specific period (e.g., daily, monthly, quarterly, yearly)
- It can be driven by many factors:
    - Naturally occurring events such as weather fluctuations caused by time of year
    - Business or administrative procedures, such as start and end of a school year
    - Social or cultural behavior, e.g., holidays

- Residuals are the random fluctuations left over after trend and seasonality are removed from the original time series
- Capture short term, unpredictable measurements
    - Noise
    - Uncertainty (measurements/ model / actuator)
    - Faults / failures

- Time series components can be decomposed with the following models:
    1. Additive decomposition
    2. Multiplicative decomposition
    3. Pseudoadditive decomposition
- Additive models assume that the observed time series is the sum of its components

$$X(t) = T(t) + S(t) + R(t)$$

- where T(t), S(t) and R(t) is the trend, seasonality and residual
- Additive models are used when the magnitudes of the seasonal and residual values do not depend on the level of the trend

- Multiplicative models assume the observed time series is a product of its components

$$X(t) = T(t) \cdot S(t) \cdot R(t)$$

- Can be transformed in additive model by log transformation

$$\log X(t) = \log(T(t)) + \log(S(t)) + \log(R(t))$$

- Used when the magnitudes of seasonal and residual values depends on the trend

- Pseudoadditive models combine elements of the additive and multiplicative models.
- Useful when:
  - Time series values are very small or zero
  - Multiplicative models struggle with zero values, but you still need to model multiplicative seasonality
- Some features are multiplicative (e.g., seasonal effects) and other are additive (e.g., residuals).
- Complex seasonal patterns or data that do not completely align with additive or multiplicative model.
- Particularly relevant for modeling series that are extremely weather-dependent, have sharply pronounced seasonal fluctuations and trend-cycle movements

$$X(t) = T(T) + T(t) \cdot (S(t) - 1) + T(t) \cdot (R(t) - 1)$$

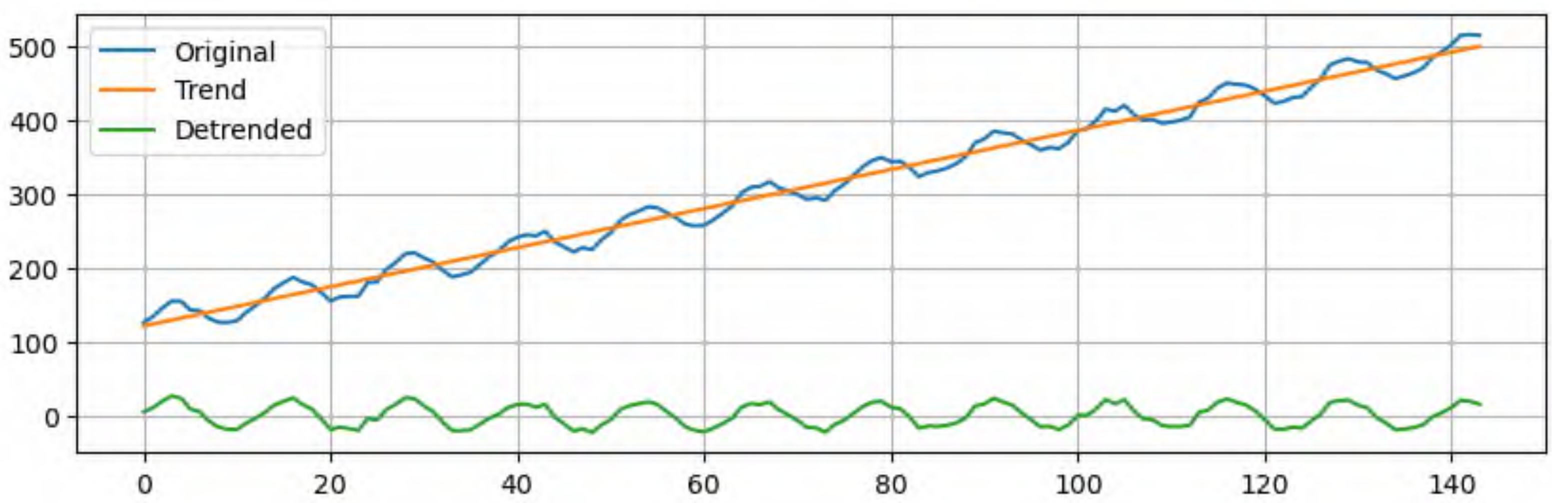
Πανεπιστήμιο Κύπρου

1. Estimate a linear trend (compute 1st order polynomial)

```
slope, intercept = np.polyfit(np.arange(len(additive)),
additive, 1) # estimate line coefficient
trend = np.arange(len(additive)) * slope + intercept # linear
trend
```

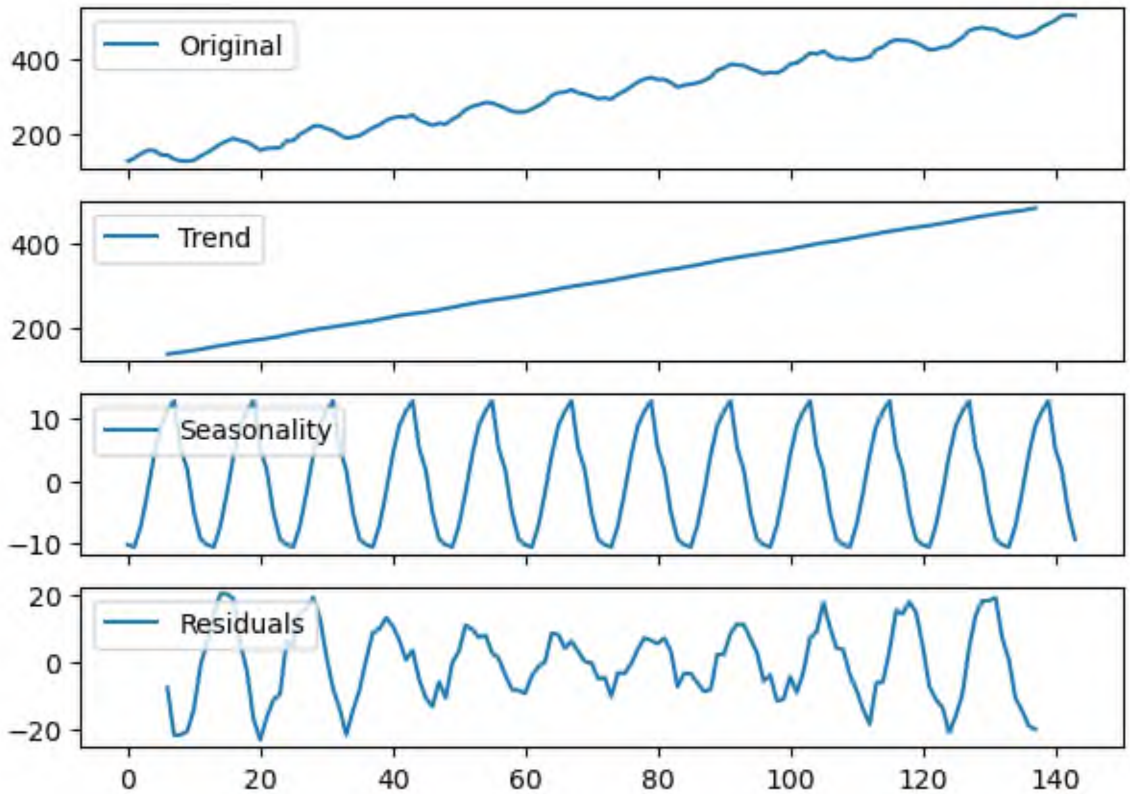
2. Detrend time series by subtracting linear component

```
detrended = additive - trend # remove the trend
```



1. seasonal_decompose function
    a) specify type of model (additive or multiplicative)
    b) main period

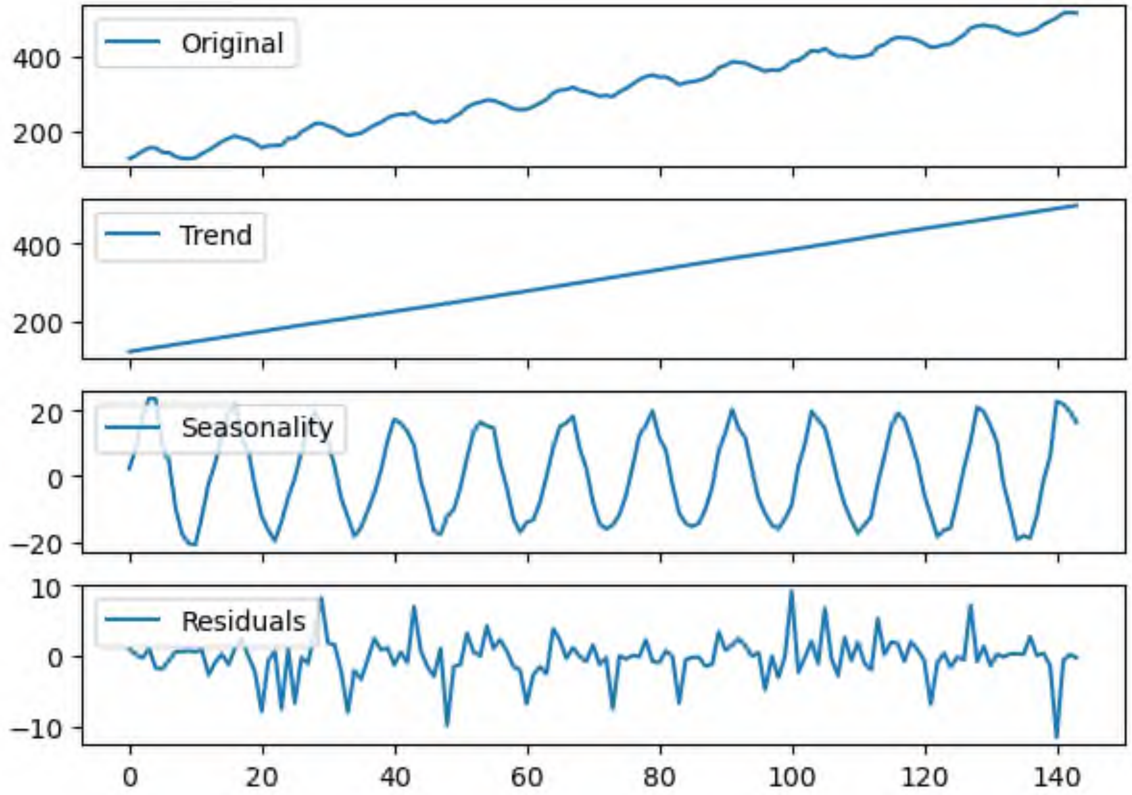
```
additive_decomposition = seasonal_decompose(x, model='additive', period=12)
```



Πανεπιστήμιο Κύπρου

- LOESS is an alternative approach employed by function STL
    - Seasonal and Trend decomp. using LOESS

```
stl_decomposition = STL(endog=additive, period=12, robust=True).fit()
```



Πανεπιστήμιο Κύπρου

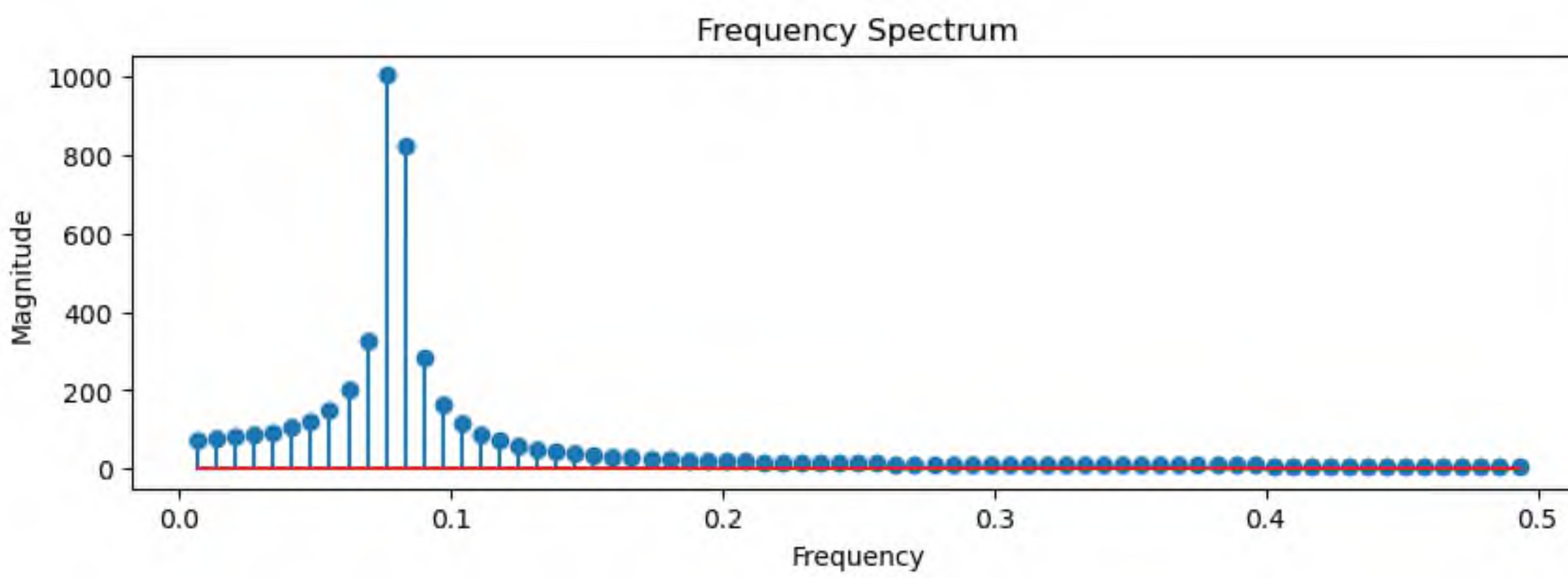
Which method to use?

- Use seasonal_decompose when:
    - Data has a clear and stable seasonal pattern and trend
    - Simpler model with fewer parameters to set
    - The seasonal amplitude is constant over time (suggesting an additive model) or varies proportionally with the trend (suggesting a multiplicative model)

- Use STL when:
    - Data exhibits complex seasonality that may change over time
    - Handle outliers effectively without them distorting the trend and seasonal components
    - Non-linear trends and seasonality, and better adjustment over the decomposition process

- Autocorrelation function
- Use Fast Fourier Transform (FFT) on a **detrended** signal

```
period, freqs, magnitudes = fft_analysis(seasonal)
```



- Autocorrelation function
- Use Fast Fourier Transform (FFT) on a **detrended** signal

```
def fft_analysis(signal):

    # Linear detrending
    slope, intercept = np.polyfit(np.arange(len(signal)), signal, 1)
    trend = np.arange(len(signal)) * slope + intercept
    detrended = signal - trend

    fft_values = fft(detrended)
    frequencies = np.fft.fftfreq(len(fft_values))

    # Remove negative frequencies and sort
    positive_frequencies = frequencies[frequencies > 0]
    magnitudes = np.abs(fft_values)[frequencies > 0]

    # Identify dominant frequency
    dominant_frequency = positive_frequencies[np.argmax(magnitudes)]
    print(f"Dominant Frequency: {dominant_frequency:.3f}")

    # Convert frequency to period (e.g., days, weeks, months, etc.)
    dominant_period = 1 / dominant_frequency
    print(f"Dominant Period: {dominant_period:.2f} time units")

    return dominant_period, positive_frequencies, magnitudes
```