The Internet of Things: Sensor Data Processing

Course website: http://site.unibo.it/iot

Prof. Luciano Bononi
luciano.bononi@unibo.it

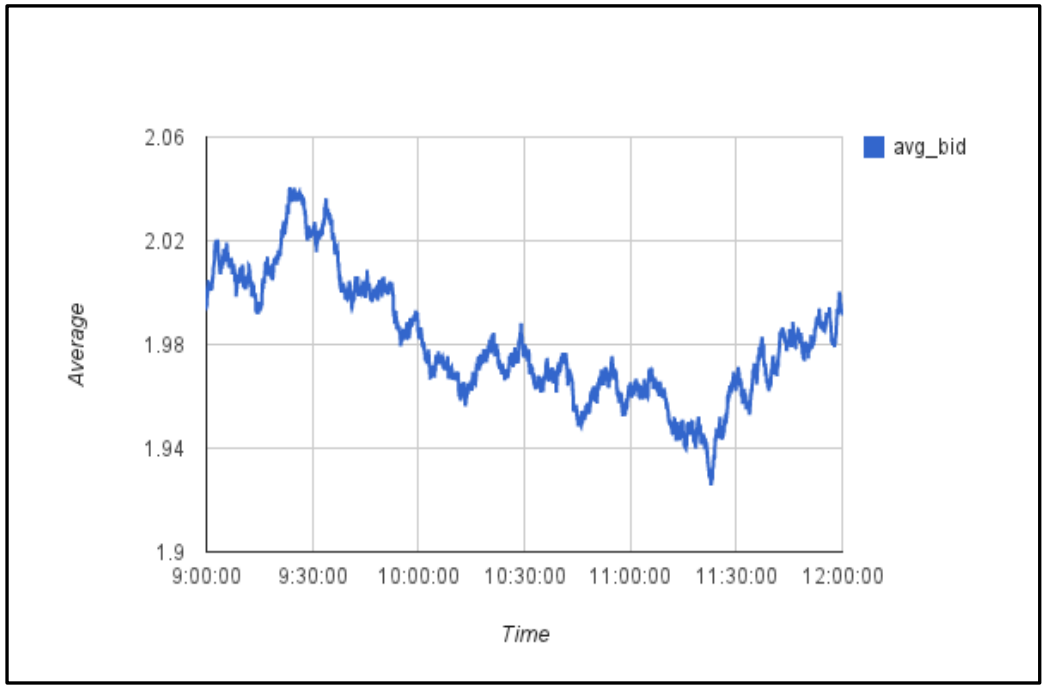
Prof. Marco Di Felice
marco.difelice3@unibo.it

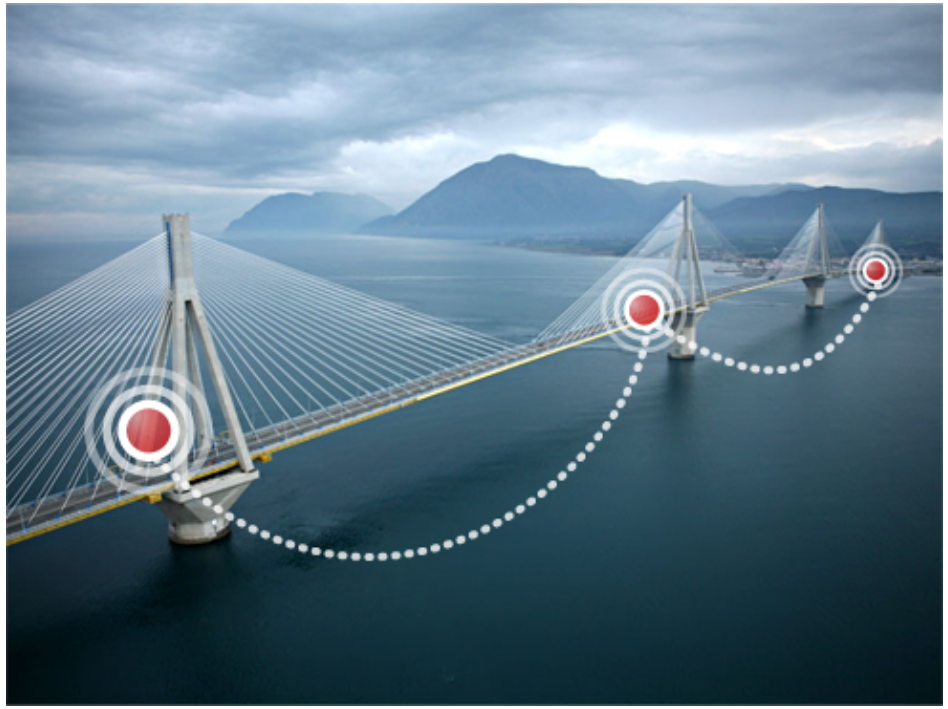
MASTER DEGREE IN COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, UNIVERSITY OF BOLOGNA, ITALY

IoT Sensor Data Management

IoT Sensor Data Management

✧Time-series → Sequence of timestamp plus values





IoT Sensor Data Management

✧**Time-series** → **Sequence of timestamp plus values**

- ❑ Data are **immutable**.
- ❑ Writing in **append**.
- ❑ Reading **contiguous sequence** of samples data.
- ❑ Highly **compressible data**.
- ❑ **Deleting** usually across large time period
- ❑ **High precision** for short period of time.
- ❑ Single value is not **so important**

IoT Sensor Data Management

✧ **Time-series Database** → Dedicated DBMS optimized for managing large volumes of time-series data

- ❑ Optimized data storage and sharding
- ❑ Operational support (e.g. range-based queries)
- ❑ Time-granularity management
- ❑ Time-series analytics and mining

IoT Sensor Data Processing

For ease of disposition, in this presentation we will distinguish between:

✧ **Time-series Processing** → Techniques and tools to manipulate the time-series (e.g. data aggregation).

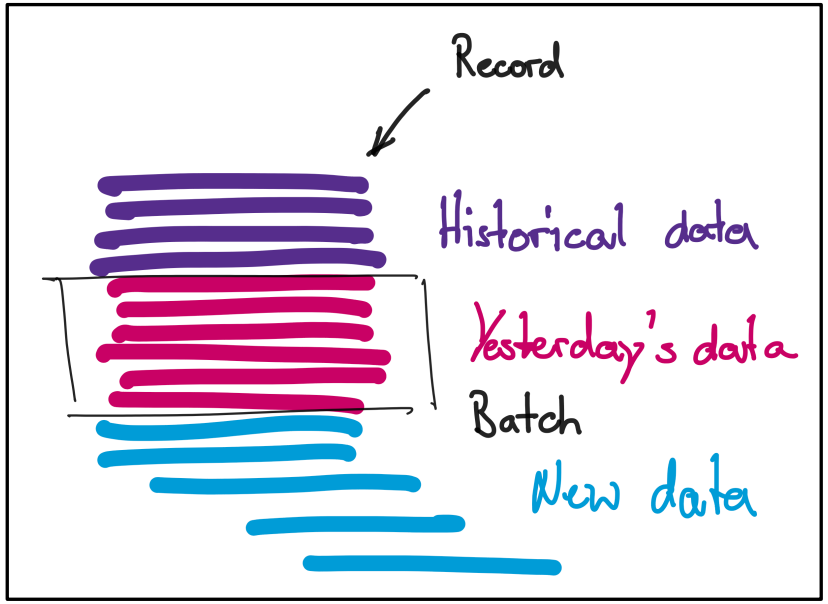
✧**Time-series Analytics** → Techniques and tools to extract information from the time-series (e.g. forecasting).

NOTE. In several use-cases, processing and analytics are combined together (e.g. first aggregate/filter the time-series, then apply machine learning techniques).

IoT Sensor Data Processing

Batch Processing → Offline data processing.

- ✓ Consider **stored** time-series (e.g. from InfluxDB)
- ✓ **Group** data points together within a specific time interval
- ✓ **Aggregate/manipulate** the data–points.
- ✓ Write results back to a sink (e.g. again to InfluxDB)

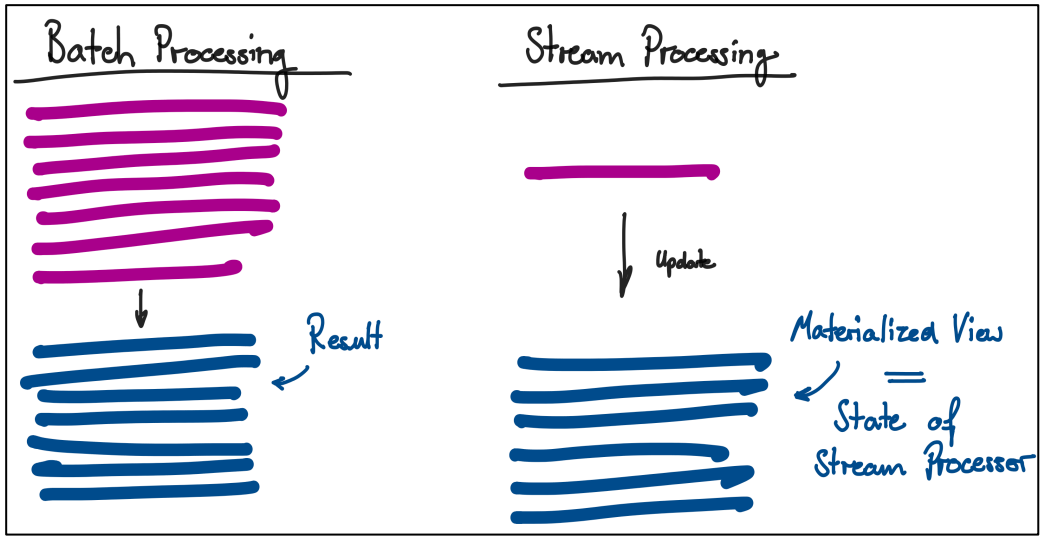


Source: https://blog.philipp-brunenberg.de/entries/batch-stream/

IoT Sensor Data Processing

Stream Processing → Online data processing.

- ✓ Process streams of data as they occurr
- ✓ Every new data point is an **event**
- ✓ Process the event immediately, and likely generate new events (e.g. trigger an alarm)
- ✓ **Velocity** is the main requirement.



Source: https://blog.philipp-brunenberg.de/entries/batch-stream/

IoT Sensor Data Processing

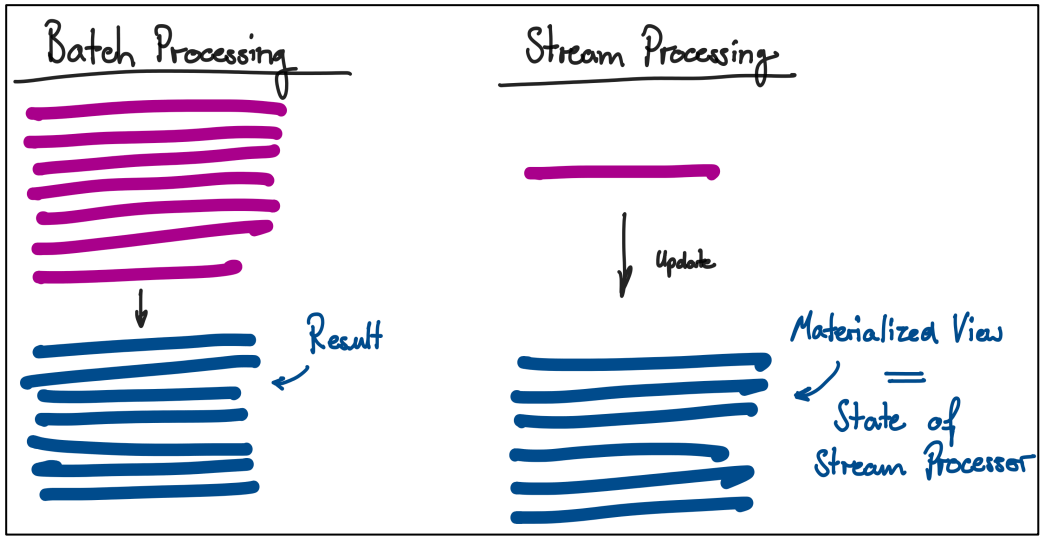
Stream Processing → Online data processing.

NON REAL-TIME OPERATIONS

- ✓ Sensor data **cleaning** and **validation**
- ✓ **Aggregate** information for monitoring

REAL-TIME OPERATIONS

- ✓ **Detect** events of interest and **trigger** appropriate action
- ✓ Real-time **predictive** and optimized operations

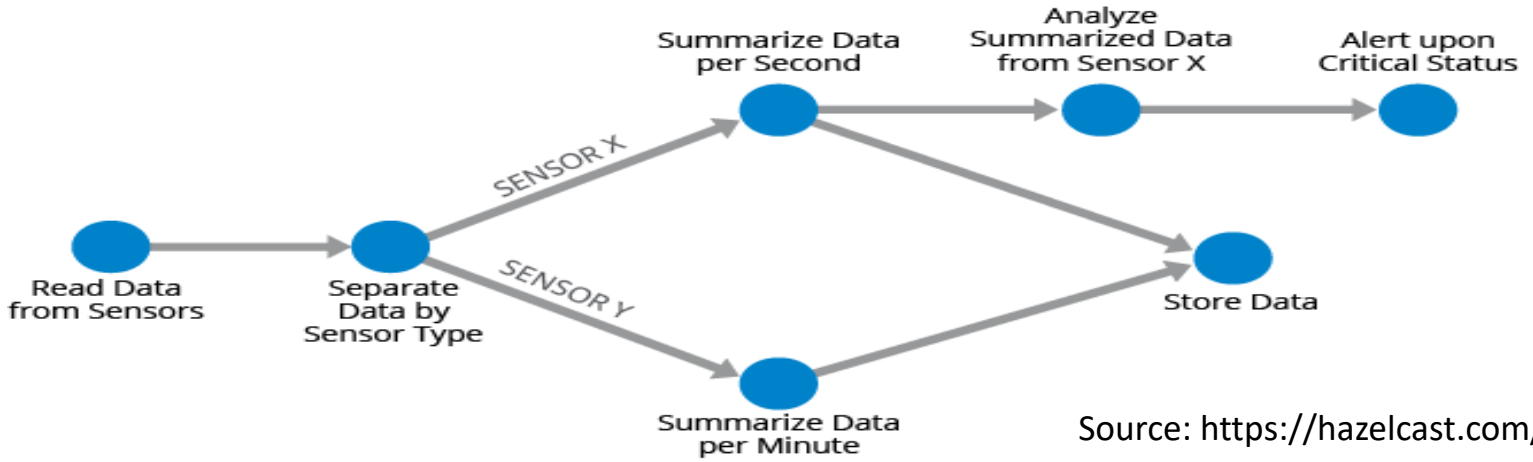


Source: https://blog.philipp-brunenberg.de/entries/batch-stream/

IoT Sensor Data Processing

Data processing flows can be represented by a **DAG** (**Direct Acyclic Graph**).

- ✓ Nodes → Process invokation units/data stream operators
- ✓ Links → Pipelines of operators

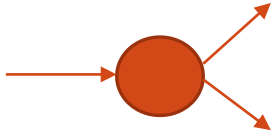


Source: https://hazelcast.com/glossary/directed-acyclic-graph/

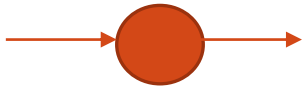
IoT Sensor Data Processing

Example of operators:

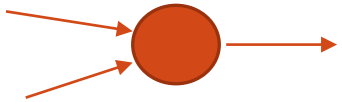
- **Stateless** operators → process each data point independently from the previously ones.



FILTER → Filter the data points according to one or more conditions



MAP → Transform each data-point (e.g. change unit scale)



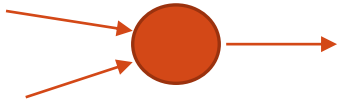
MERGE → Merge multiple data-points into one (e.g. average)

IoT Sensor Data Processing

Example of operators:

- **Stateful** operators → produce a single output from multiple data points, from the same or different series.

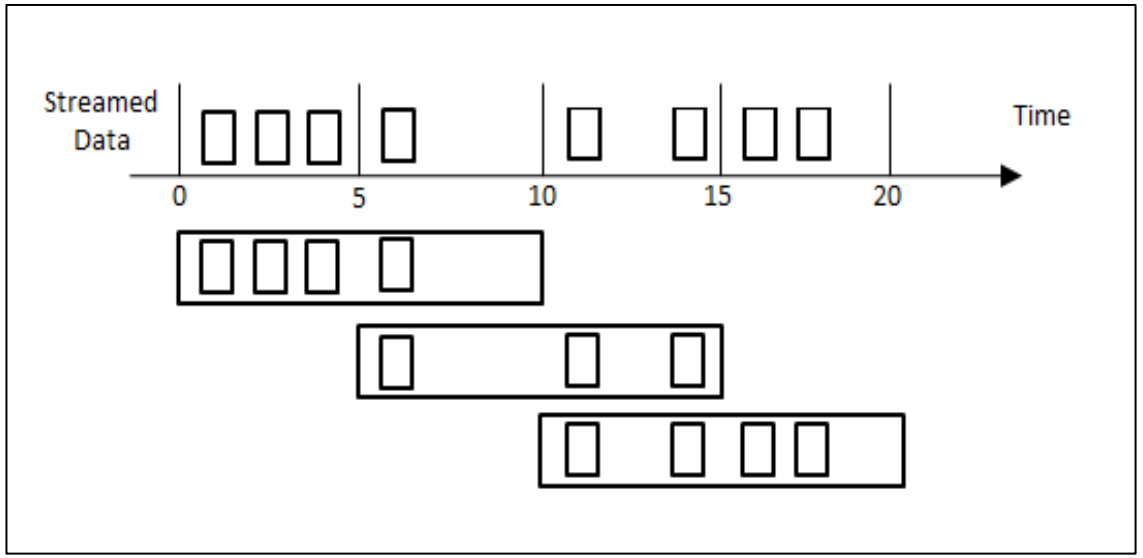
AGGREGATE → Aggregate multiple data-points from the same series (e.g. downsampling, compute min/max/avg values)



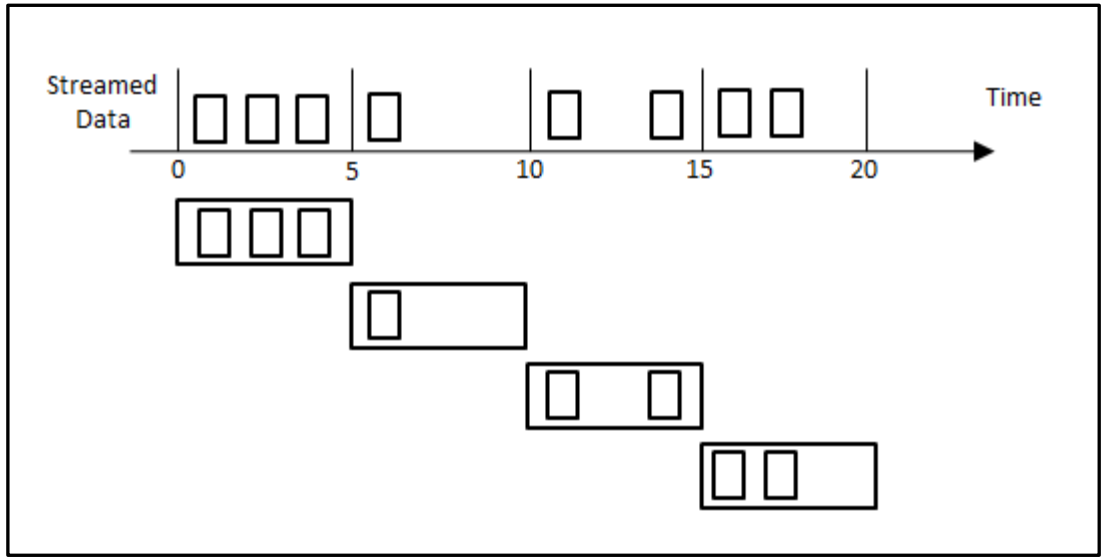
JOIN → Aggregate multiple data-points from different same series

IoT Sensor Data Processing

- **Stateful operators** require some sort of data buffering and windowing.



TUMBLING WINDOW



SLIDING WINDOW

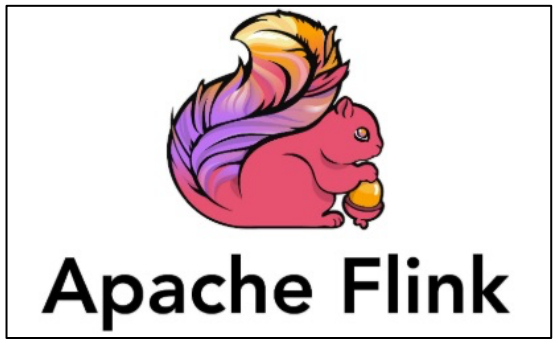
IoT Sensor Data Processing

- Some data stream-processing **frameworks**

APACHE PROJECTS

APACHE KAFKA

APACHE STORM



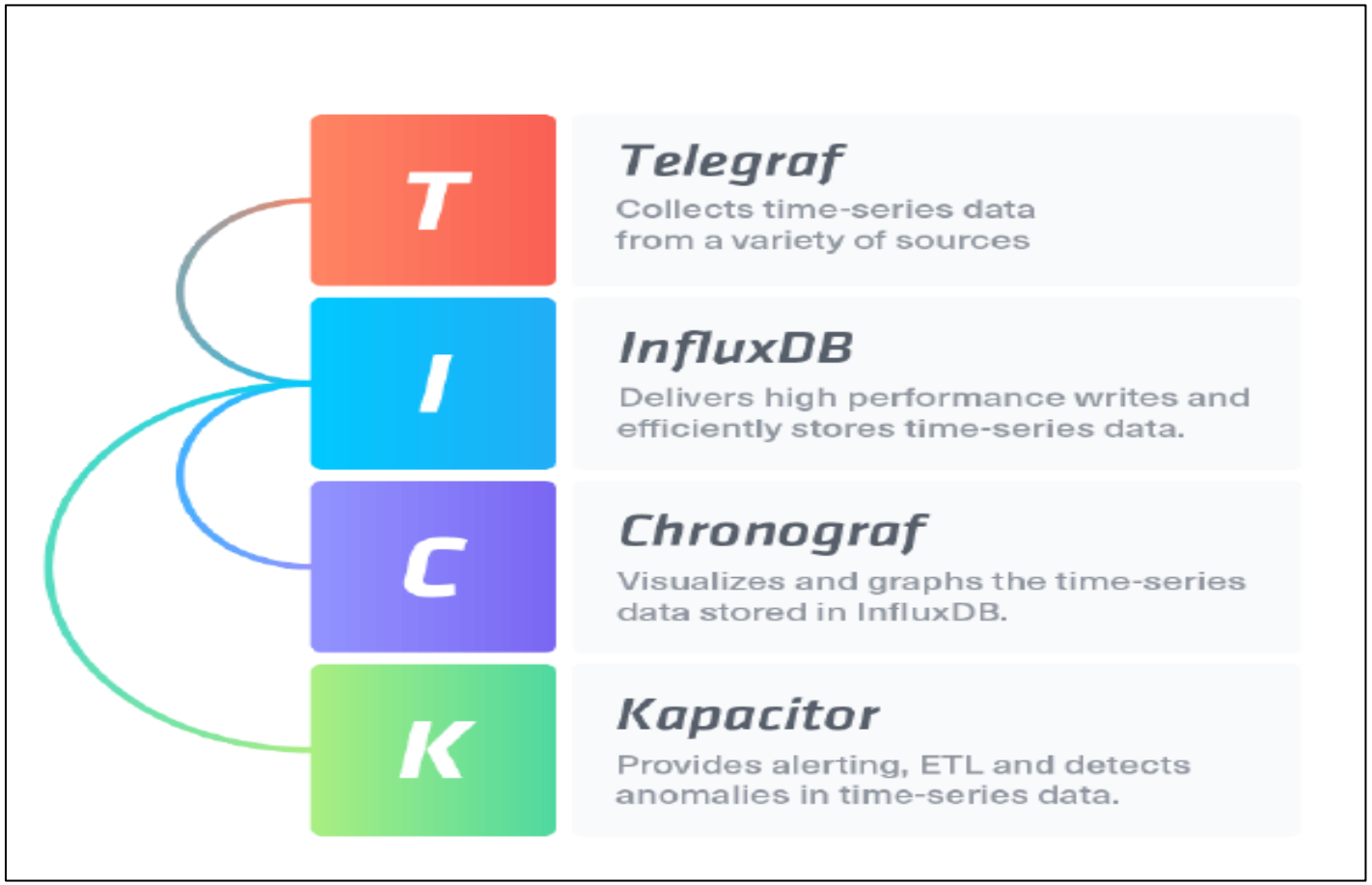
APACHE FLINK

APACHE SPARK

APACHE SAMZA

APACHE BEAM

IoT Sensor Data Processing



The **TICK** stack **from InfluxDB (tool version <2.0)**

IoT Sensor Data Processing

Novelties of the **InfluxDB 2.0** framework (https://www.influxdata.com) :

- TICK stack is integrated within a single tool, that supports time-series storage, processing and visualization
- A single language (**FLUX**) is used both for time-series data querying and for time-series processing
- Continuous **query** processing (hybrid data/stream processing)

IoT Sensor Data Processing

FLUX tutorial:
https://v2.docs.influxdata.com/v2.0/reference/flux/

Table → sequence of records with a common set of columns and a group key.

Stream → potentially unbounded set of tables.

Transformations → define a change to a stream. Transformations may consume an input stream and always produce a new output stream.

IoT Sensor Data Processing

FLUX tutorial:
https://v2.docs.influxdata.com/v2.0/reference/flux/

Table → sequence of records with a common set of columns and a group key.

Stream → potentially unbounded set of tables.

Transformations → define a change to a stream. Transformations may consume an input stream and always produce a new output stream.

IoT Sensor Data Processing

FLUX tutorial:
https://v2.docs.influxdata.com/v2.0/reference/flux/

Main language components:

- Types
- Variables
- Assignments
- Statements
- Expressions
- Packages
- Standard and external libraries
- …

IoT Sensor Data Processing

FLUX tutorial:
https://v2.docs.influxdata.com/v2.0/reference/flux/


```
bool     // boolean
int      // integer
uint     // unsigned integer
float    // floating point number
duration // duration of time
time     // time
string   // utf-8 encoded string
regexp   // regular expression
bytes    // sequence of byte values
type     // a type that itself describes a type
```

IoT Sensor Data Processing

Operator	Description	Examples	Meaning
`|>`	Pipe-forward	`data |> function()`	Tables contained in the "data" variable are piped into the function.
`<-`	Pipe-receive	`tables=<-`	The "tables" variable or parameter is assigned to data piped into the operation. *This operator is used for any data type passed into a function; not just table data.*
`=>`	Arrow	`(r) => r.tag1 == "tagvalue"`	The arrow passes an object or parameters into function operations.
`()`	Function call	`top(n:10)`	Call the `top` function setting the `n` parameter to `10` and perform the associated operations.

influxdata

FUNCTION OPERATORS (including DATA STREAMING OPERATORS)

IoT Sensor Data Processing

FLUX tutorial:
https://v2.docs.influxdata.com/v2.0/reference/flux/

Assign a bucket to a variable:

```
myvar= from (bucket: «temperature»);
```

Assign a data-stream pipeline to a variable:

```
myvar= from (bucket: «temperature»)
       |> range(start: -1h)
```

IoT Sensor Data Processing

FLUX tutorial:
https://v2.docs.influxdata.com/v2.0/reference/flux/

Example of **query** operators (stateless):

- `limit(n:XXX)`
- `sort(columns:[…])`
- `range(start: XXX)`
- `filter(fn: (r) => …)`

```
myvar= from (bucket: «sensorIOT»)
        |> range(start: -1h)
        |> filter(fn: (r) => r._measurement == «temperature»)
```

IoT Sensor Data Processing

FLUX tutorial:
https://v2.docs.influxdata.com/v2.0/reference/flux/

Example of **query** operators (stateful):

- `window(every:XXX)`
- `aggregateWindow(every: 1m, fn: XXX)`

Built-in function or user-defined

```
myvar= from (bucket: «sensorIOT»)
        |> window(every: 1m)
```

```
myvar= from (bucket: «sensorIOT»)
        |> aggregateWindow(every: 1m, fn: mean)
```

IoT Sensor Data Processing

FLUX tutorial:
https://v2.docs.influxdata.com/v2.0/reference/flux/

Example of **query** operators (stateless, manipulation):

- `map(fn: (r) => ({ MANIPULATION FUNCTION }))`

```
farToCelsius = (tables=<-) =>
            tables
           |> map(fn: (r) => ({ r with _value: r._value -32 / 1.8 }) )

data |> farToCelsius()
```

IoT Sensor Data Processing

FLUX tutorial:
https://v2.docs.influxdata.com/v2.0/reference/flux/

Example of **query** operators (conditional):

- `if <condition> then <action> else <alternative-action>`

```
myVar= from(bucket: "example-bucket")
       |> range(start: -5m)
       |> filter(fn: (r) => r._measurement == «mem»)
       |> map(fn: (r) => ({
          r with level: if r._value >= 95.0000001 then "critical"
                        else then "warning"
```

IoT Sensor Data Processing

FLUX tutorial:
https://v2.docs.influxdata.com/v2.0/reference/flux/

Task → scheduled Flux script that takes a stream of input data, modifies or analyzes it in some way, then stores the modified data in a new bucket or performs other actions.

Components:

- ✓ Options (e.g. scheduling interval)
- ✓ Data Source
- ✓ Stream Processing Script
- ✓ Destination

IoT Sensor Data Processing

```
option task = { name: "taskDemo",
                every: 1m,
                offset: 0m}
```

TASK OPTIONS


```
data = from(bucket: "iotDemo")
       |>  range(start: -task.every)
       |> filter(fn: (r) => r._measurement =="temperature")
```

DATA SOURCE

```
data
|> aggregateWindow( every: 5m, fn: mean )
```

DATA PROCESSING

```
data
|> to(bucket: "iotDemo_downsampled")
```

DATA OUTPUT

IoT Sensor Data Processing

For ease of disposition, in this presentation we will distinguish between:

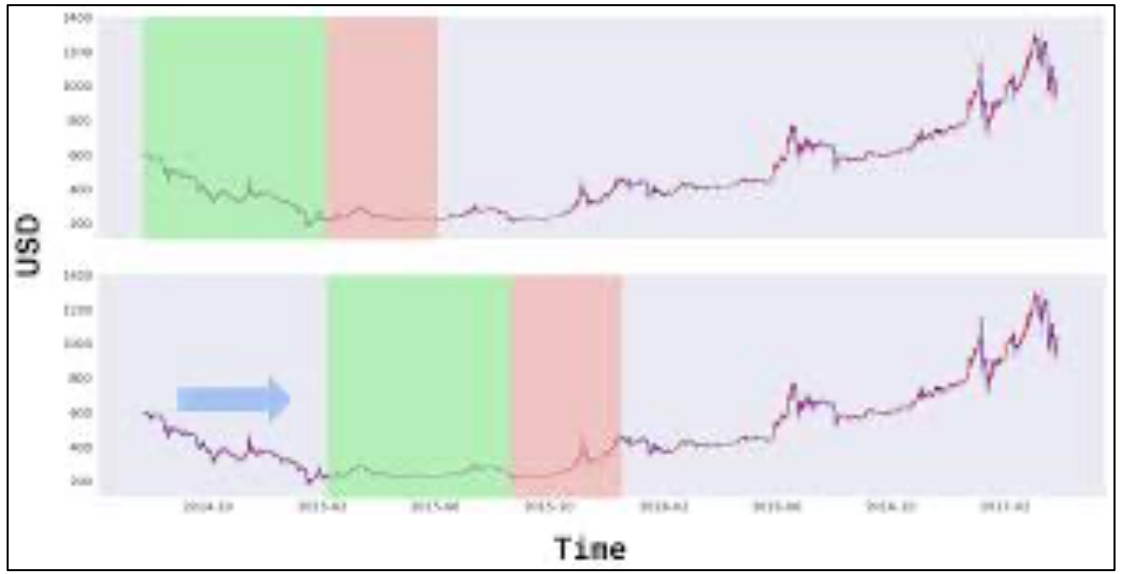
✧ **Time-series Processing** → Techniques and tools to manipulate the time-series (e.g. data aggregation).

✧**Time-series Analytics** → Techniques and tools to extract information from the time-series (e.g. forecasting).

NOTE. In several use-cases, processing and analytics are combined together (e.g. first aggregate/filter the time-series, then apply machine learning techniques).

IoT Sensor Data Processing

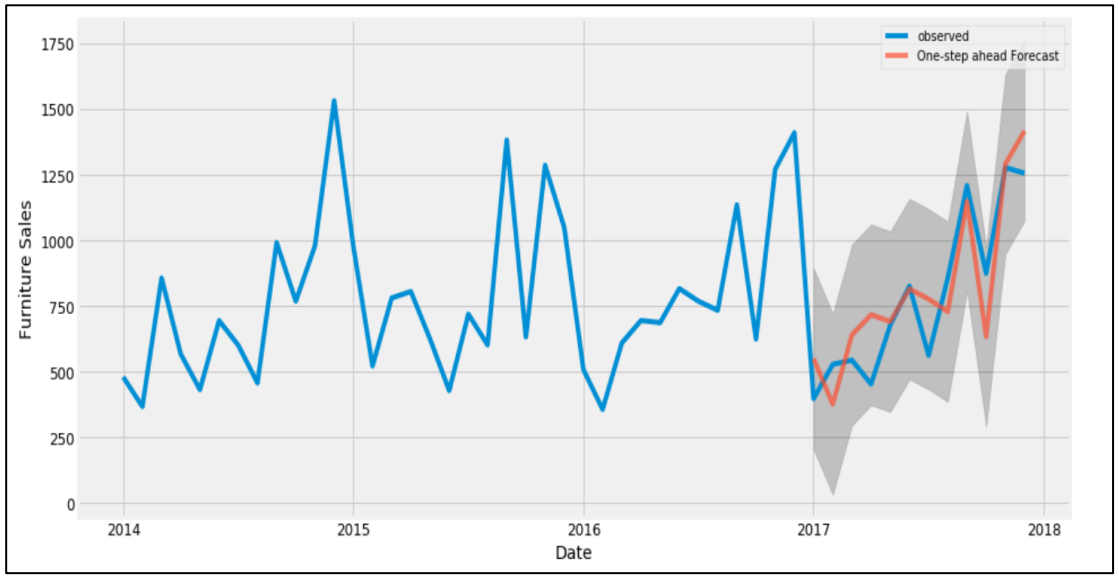
Source: https://www.mdpi.com/1911-8074/12/1/17/htm



TIME-SERIES CLASSIFICATION

- Assign a label to a time-series or to a subset

Source: https://towardsdatascience.com/an-end-to-end-project-on-time-series-analysis-and-forecasting-with-python-4835e6bf050b



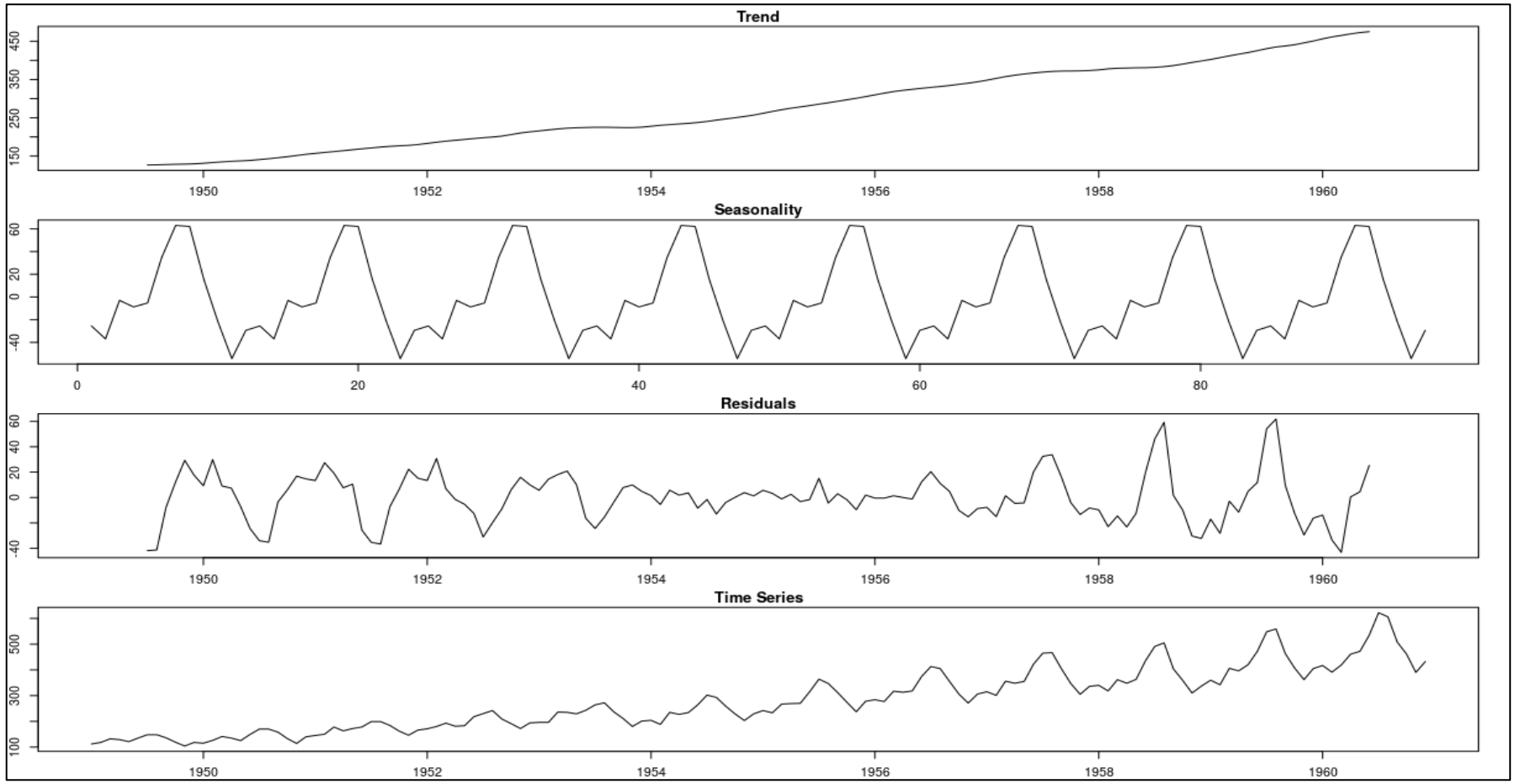
TIME-SERIES FORECASTING

- Predict future values based on previously observed values.

IoT Sensor Data Processing

- **Secular Trend**
  - Smooth long-term direction of a time series
- **Cyclic**
  - Recurring events, but not periodic in nature.
- **Seasonal**
  - Patterns of change in a time series within a period. These patterns tend to repeat themselves each period.
- **Irregular Variations**
  - Further classified as episodic or residual, cannot be predicted

IoT Sensor Data Processing



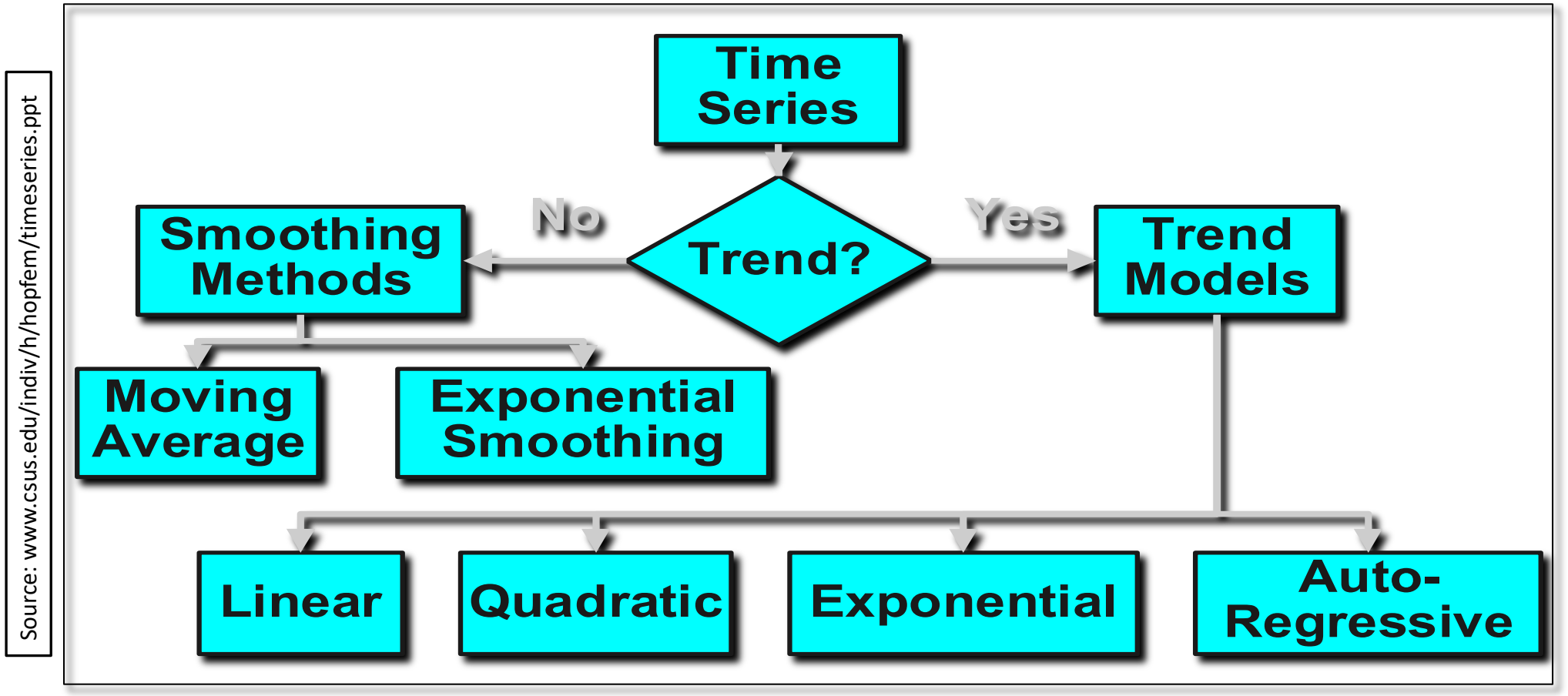
TREND

SEASONALITY

RESIDUALS

TIME-SERIES

IoT Sensor Data Processing



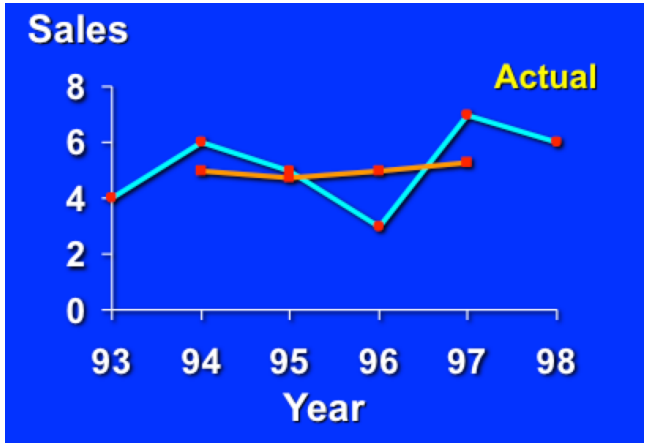
Source: www.csus.edu/indiv/h/hopfem/timeseries.ppt

IoT Sensor Data Processing

✧ Moving Average Method

- ❑ Elementary **forecasting** solution
- ❑ The *n*-period forecast is simply the average of the observations in the most recent *n* periods:

$$A_t = (x_t + x_{t-1} + \ldots + x_{t-n+1}) / n$$



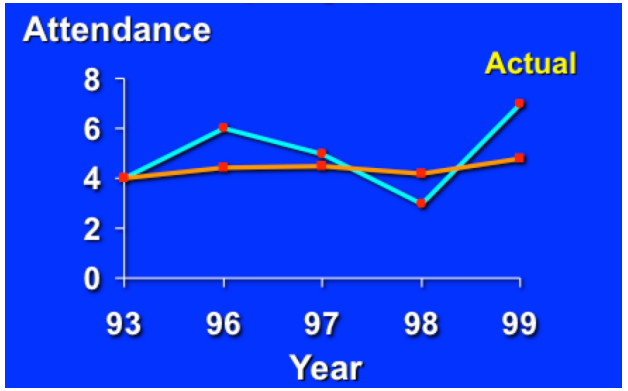
Source: www.csus.edu/indiv/h/hopfem/timeseries.ppt

IoT Sensor Data Processing

✧ Exponential Smoothing

- ❑ Form of weighted moving average
- ❑ Use a smoothing costant (W), range between 0 and 1
- ❑ Record only **actual value** and **past prediction**

$$S_t = \alpha x_t + (1 - \alpha) S_{t-1}$$



Source: R.B. Clough – UNH M. E. Henrie - UAA

IoT Sensor Data Processing

✧ Linear Time-Series Forecasting Model

- ❑ Causal models establish a cause-and-effect relationship between dependent variable to be forecast (Y) and independent variables (x_i)
- ❑ A common tool of causal modeling is **multiple linear regression**:

$$Y = a + b_1 x_1 + b_2 x_2 + \ldots + b_k x_k$$

IoT Sensor Data Processing

✧ Linear Time-Series Forecasting Model

- ❑ The trend line equation is then:

$$Y = ax + b$$

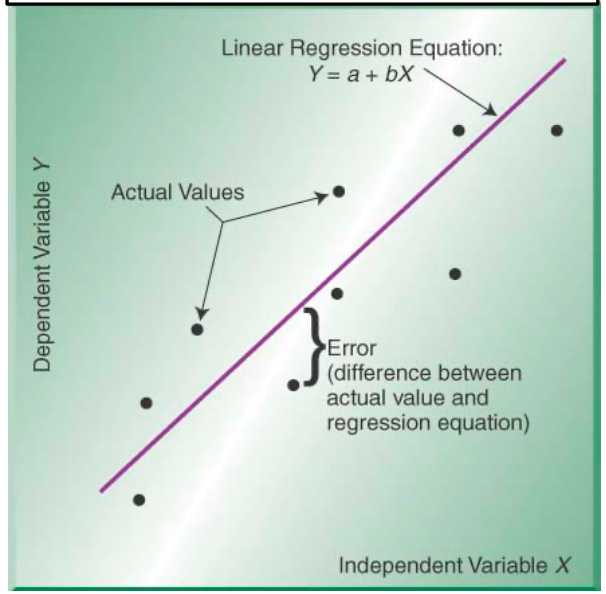
- ❑ The slope of the line can be computed as follows:

$$\mathbf{b} = \frac{\sum \mathbf{XY} - \mathbf{n}\overline{\mathbf{X}}\overline{\mathbf{Y}}}{\sum \mathbf{X}^2 - \mathbf{n}\overline{\mathbf{X}}^2}$$

- ❑ The intercept can be computed as follows

$$\mathbf{a} = \overline{\mathbf{Y}} - \mathbf{b}\overline{\mathbf{X}}$$

Source: R.B. Clough – UNH M. E. Henrie - UAA



IoT Sensor Data Processing

✧ Linear Time-Series Forecasting Model

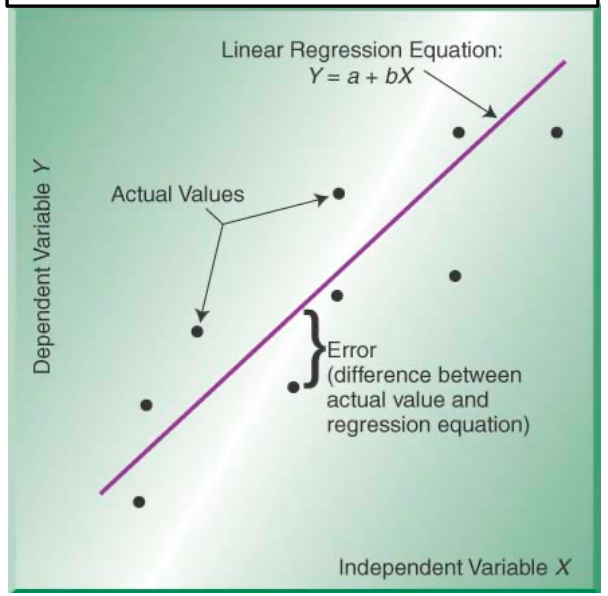
- ❑ The coefficient (r^2) equation measures the amount of variation in the dependent variable that is explained by the regression line. :

Source: R.B. Clough – UNH M. E. Henrie - UAA

$$r = \frac{n\left(\sum XY\right) - \left(\sum X\right)\left(\sum Y\right)}{\sqrt{n\left(\sum X^2\right) - \left(\sum X\right)^2} * \sqrt{n\left(\sum Y^2\right) - \left(Y\right)^2}}$$

- ❑ The higher is r^2, the better the model is

Source: R.B. Clough – UNH M. E. Henrie - UAA



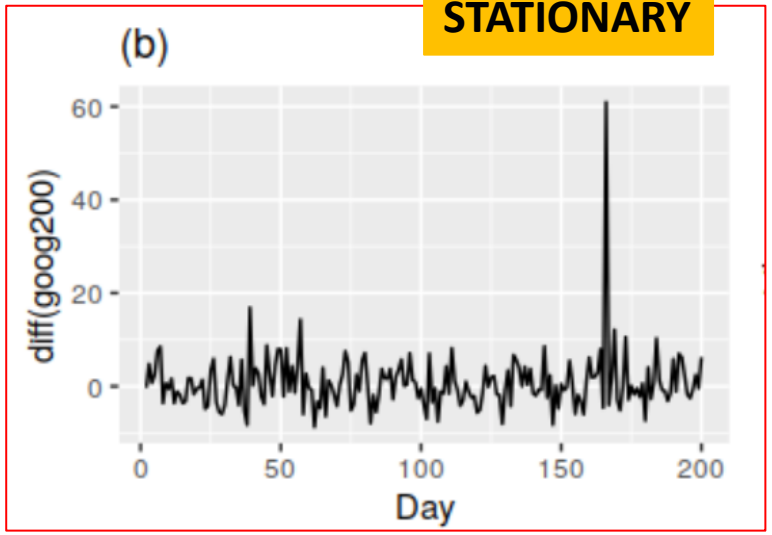
IoT Sensor Data Processing

✧ Time-series Stationary

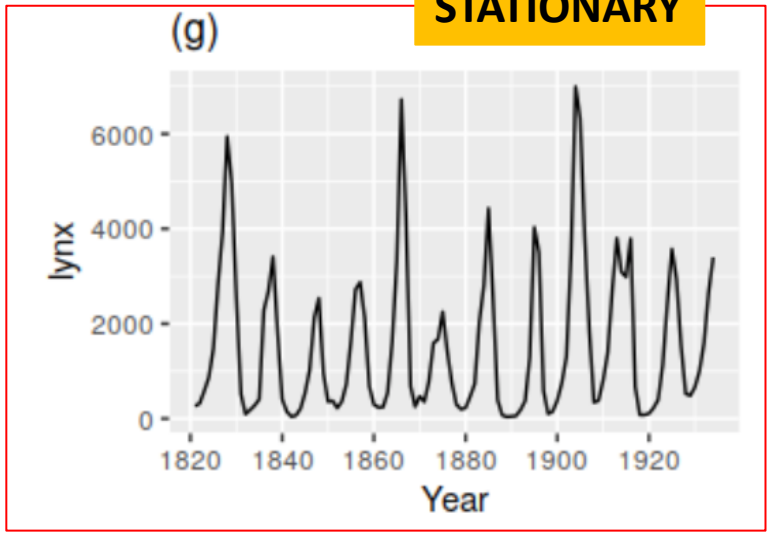
Source: https://otexts.com/fpp2/stationarity.html

✓ **Informal definition** → mean, variance and autocorrelation of the time-series do not change over time.

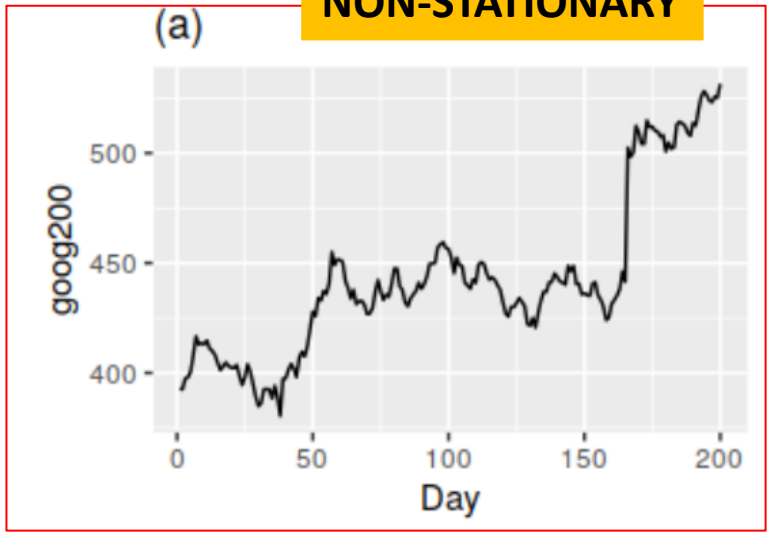
STATIONARY



STATIONARY



NON-STATIONARY



IoT Sensor Data Processing

✧ Auto Regression (AR) Model

STATIONARY TIME-SERIES

- ❑ Past values have an effect on current values
- ❑ P-th order regression:

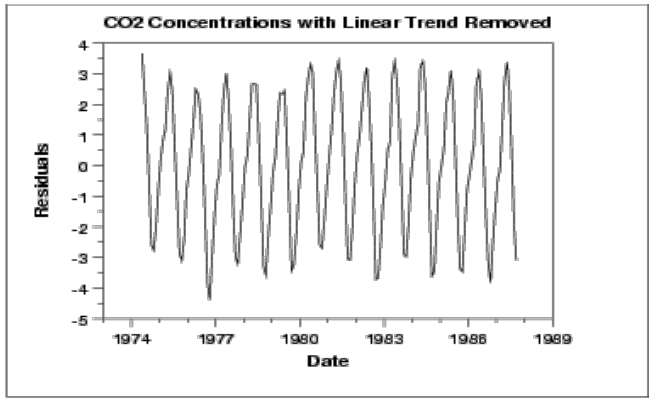
$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \varepsilon_t$$

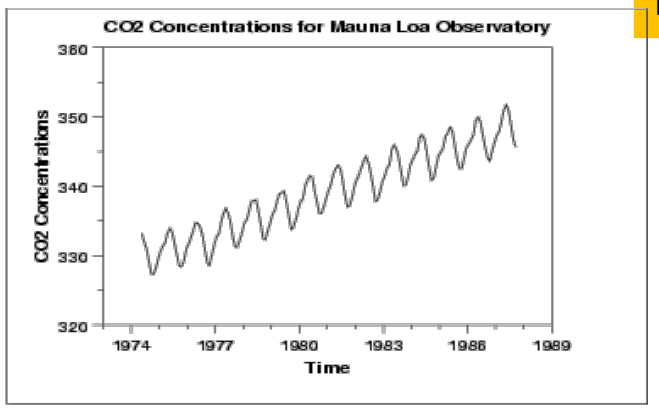
✧ Moving Average (MA) Model

- ❑ Current value depends on previous error terms
- ❑ Q-th order regression:

$$Y_t = \varepsilon_t + \alpha_1 \cdot \varepsilon_{t-1} + \alpha_2 \cdot \varepsilon_{t-2} + \cdots + \alpha_q \cdot \varepsilon_{t-q}$$

IoT Sensor Data Processing





NON-STATIONARY TIME-SERIES

Source: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm

- Transforming a *non-stationary* time-series into a *stationary* time-series

Differencing technique → Stabilise the **mean** of a time series by removing changes over time

$$y'_t = y_t - y_{t-1}.$$

FIRST ORDER DIFFERENCING

$$\begin{aligned} y''_t &= y'_t - y'_{t-1} \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2}. \end{aligned}$$

SECOND ORDER DIFFERENCING

IoT Sensor Data Processing

NON-STATIONARY TIME-SERIES

- **ARIMA(p,q,d)** *Autoregressive Integrated Moving Average Model*
  - p → number of **autoregressive** terms (AR order)
  - q → number of **nonseasonal differences** (differencing order)
  - d → number of **moving-average** terms (MA order)

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

y'_t is the d-differenced value at time t

IoT Sensor Data Processing

✧ Mean Absolute Deviation (MAD)

- ❑ Measures the total error in forecast (without sign)

$$\mathbf{MAD} = \frac{\sum |\mathbf{actual} - \mathbf{forecast}|}{\mathbf{n}}$$

✧ Mean Square Error (MSE)

- ❑ Similar to MAD, but gives higher penalty to larger errors

$$\mathbf{MSE} = \frac{\sum (\mathbf{actual} - \mathbf{forecast})^2}{\mathbf{n}}$$