

# Advanced usages of flux

---

Emanuele Della Valle

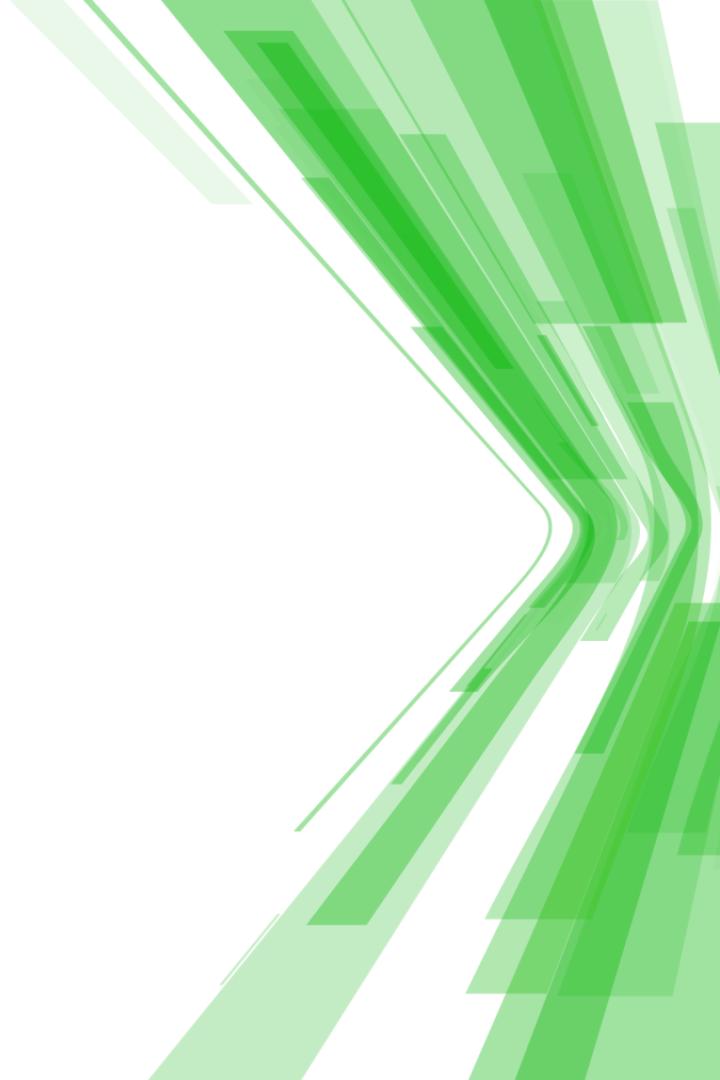
Prof. @ Politecnico di Milano

Founder & Partner @ Quantia Consulting

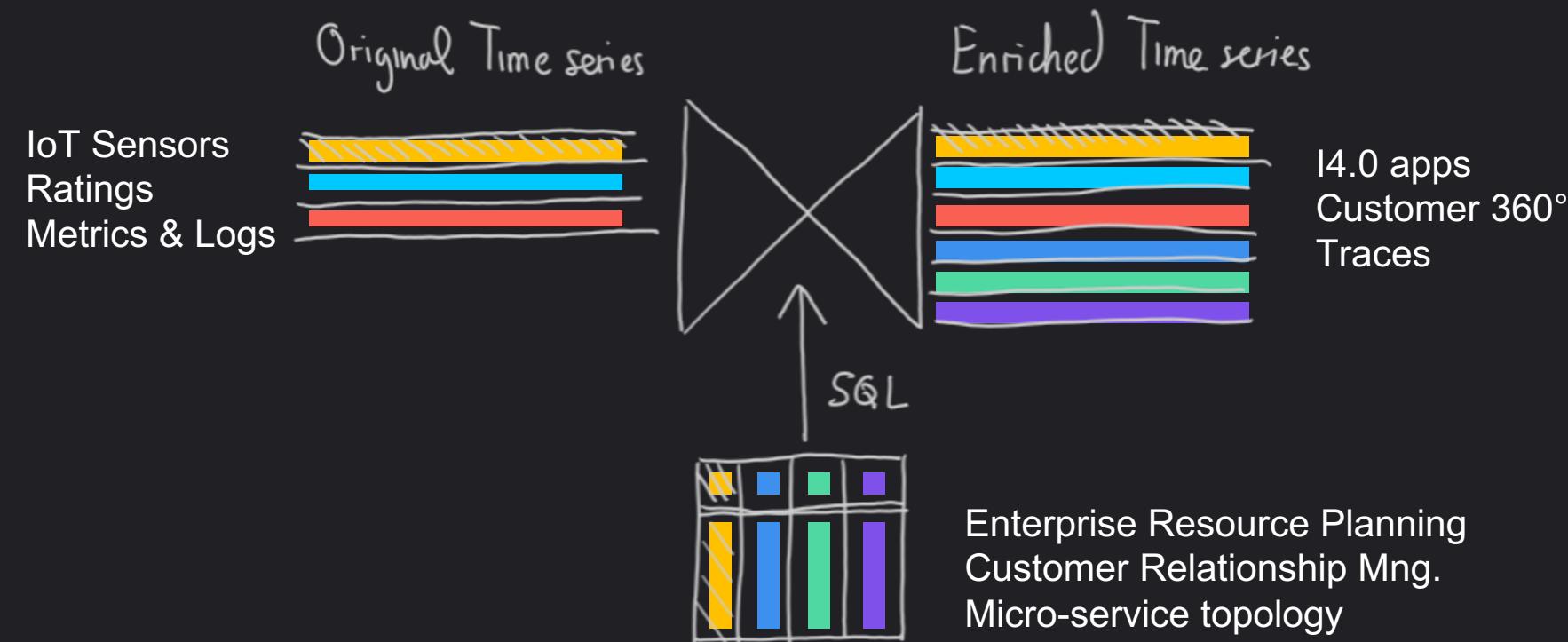
Marco Balduini



# Time Series Enrichment



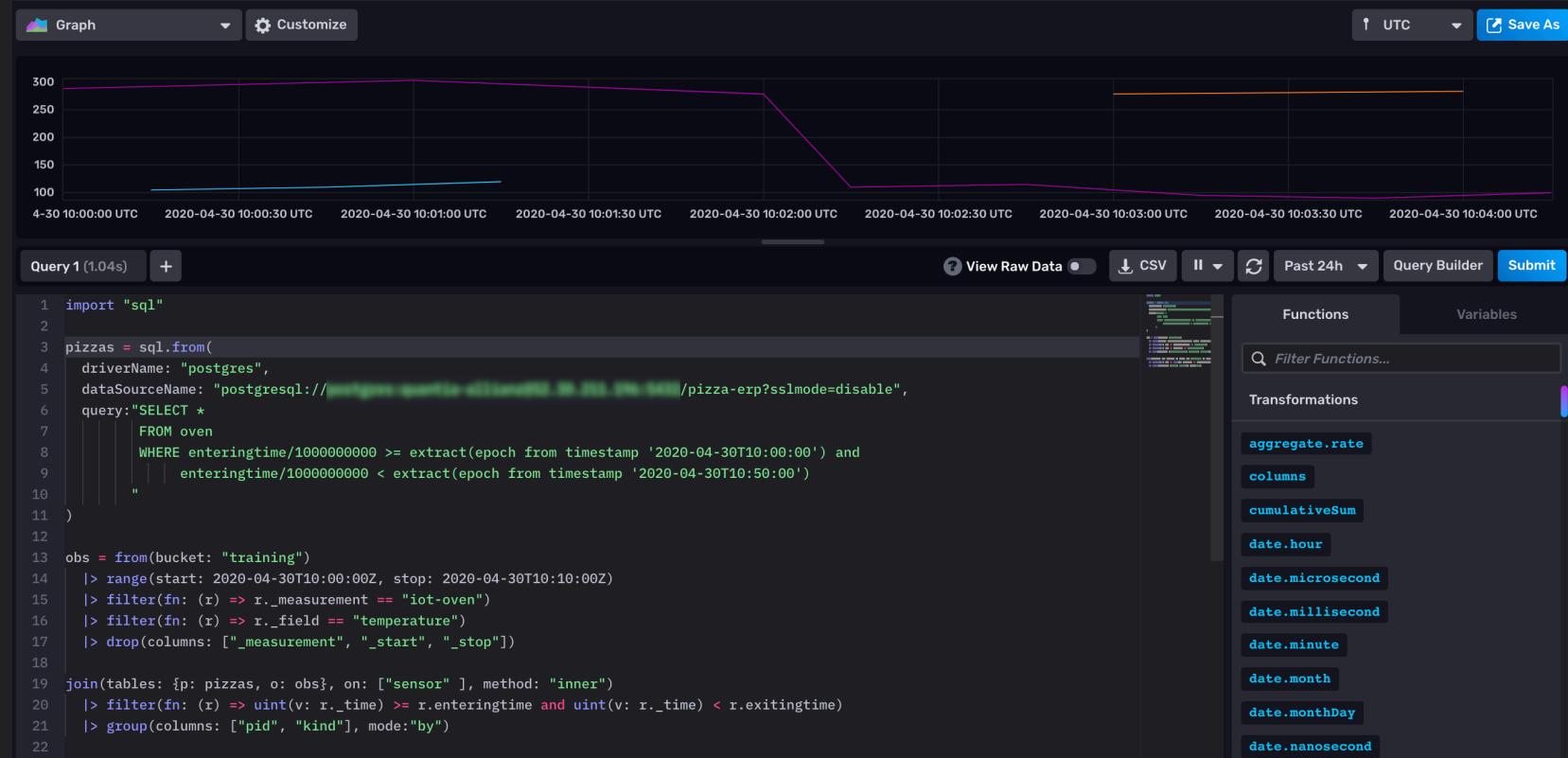
# Time Series Enrichment – the intuition



## Use case for the Linear Pizza Oven

- Enrich the IoT observations with the ID of the pizza and the kind of pizza contained in the Enterprise Resource Planning RDBMS

# Time Series Enrichment in flux

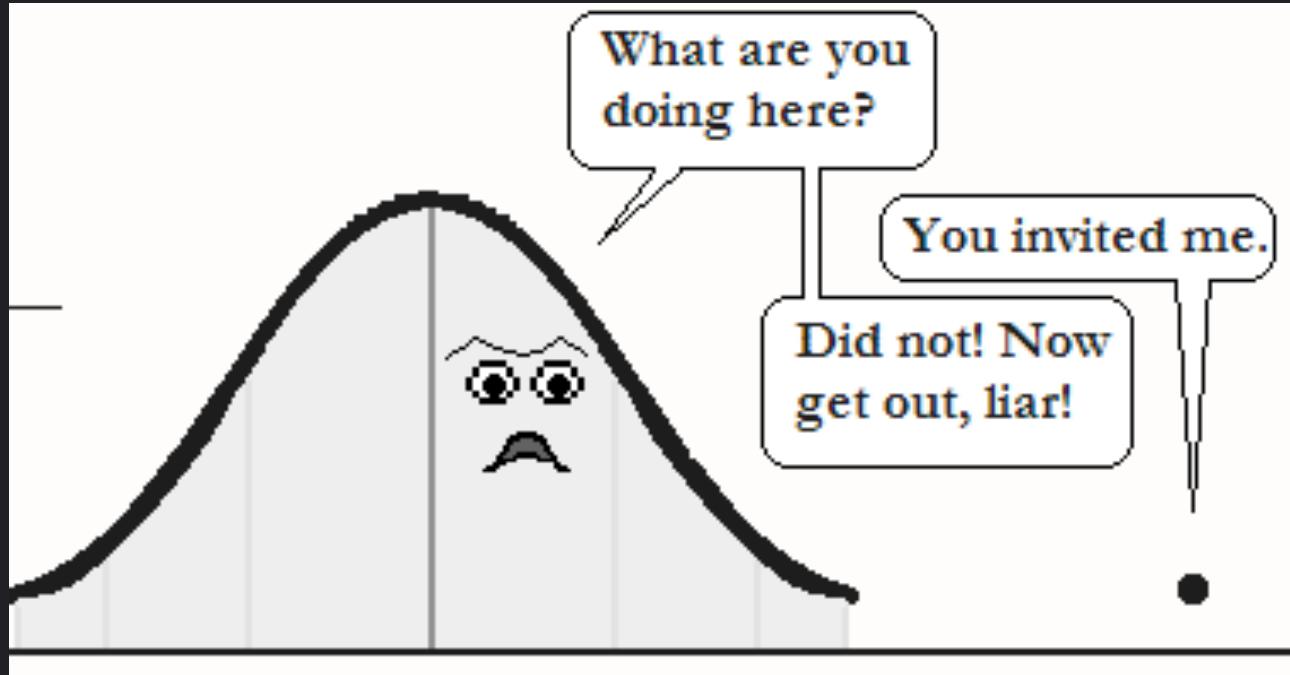


# Anomaly Detection

# Anomaly Detection

- Anomaly detection is a challenging data analysis task
- Anomalies can represent
  - spurious data to clean out
  - important patterns to detect
- This generality makes anomaly detection a powerful tool used in network security, remote sensing, fault detection, and many other domains

## Anomaly Detection – the intuition

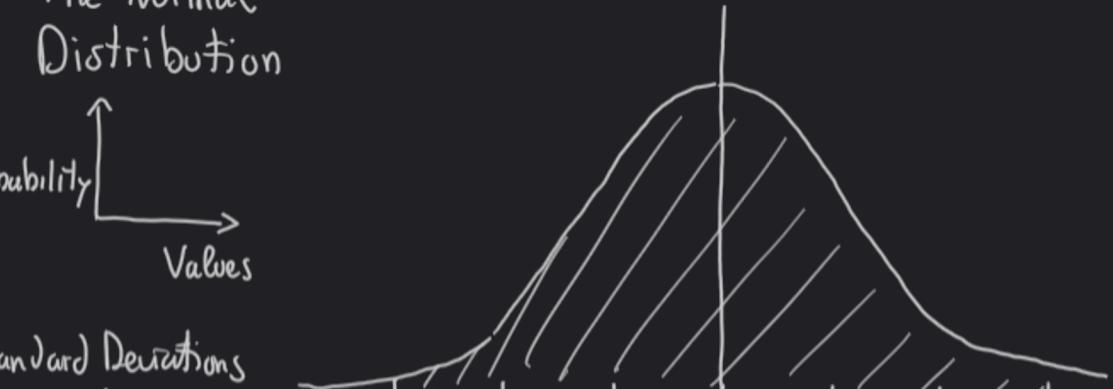


[image: <https://cerijayne.files.wordpress.com/2011/10/outliersss.png>]

Considering the distribution of the data point in a time series, anomalous points are **anomalies**, a.k.a. *outliers*

# Let's operationalize this intuition using the Z score

The Normal Distribution



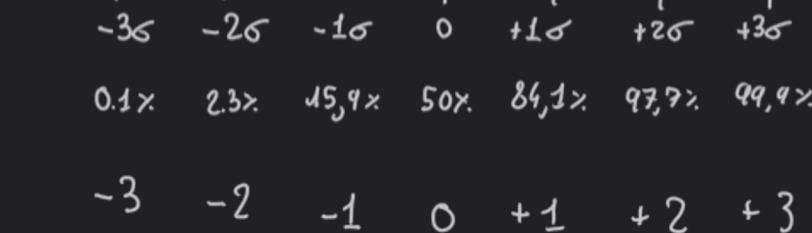
probability

Values

Standard Deviations  
from the Mean

Cumulative %

Z Score

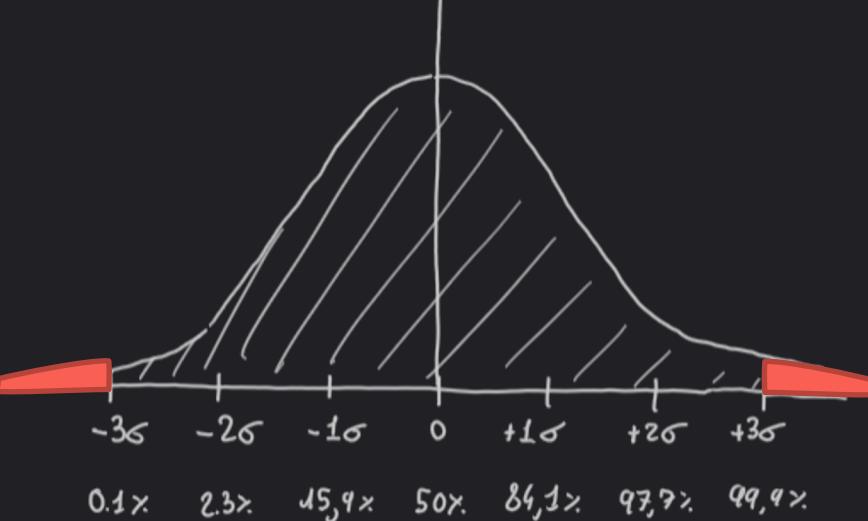


**Z score**

$$Z = \frac{X - \mu}{\sigma}$$

# Let's operationalize this intuition using the Z score

The Normal Distribution



probability  
Values

Standard Deviations from the Mean

Cumulative %

Z Score

-3    -2    -1    0    +1    +2    +3

**Z score**

$$Z = \frac{X - \mu}{\sigma}$$

**Anomaly**

$Abs(z) > 3$

0.2% of data

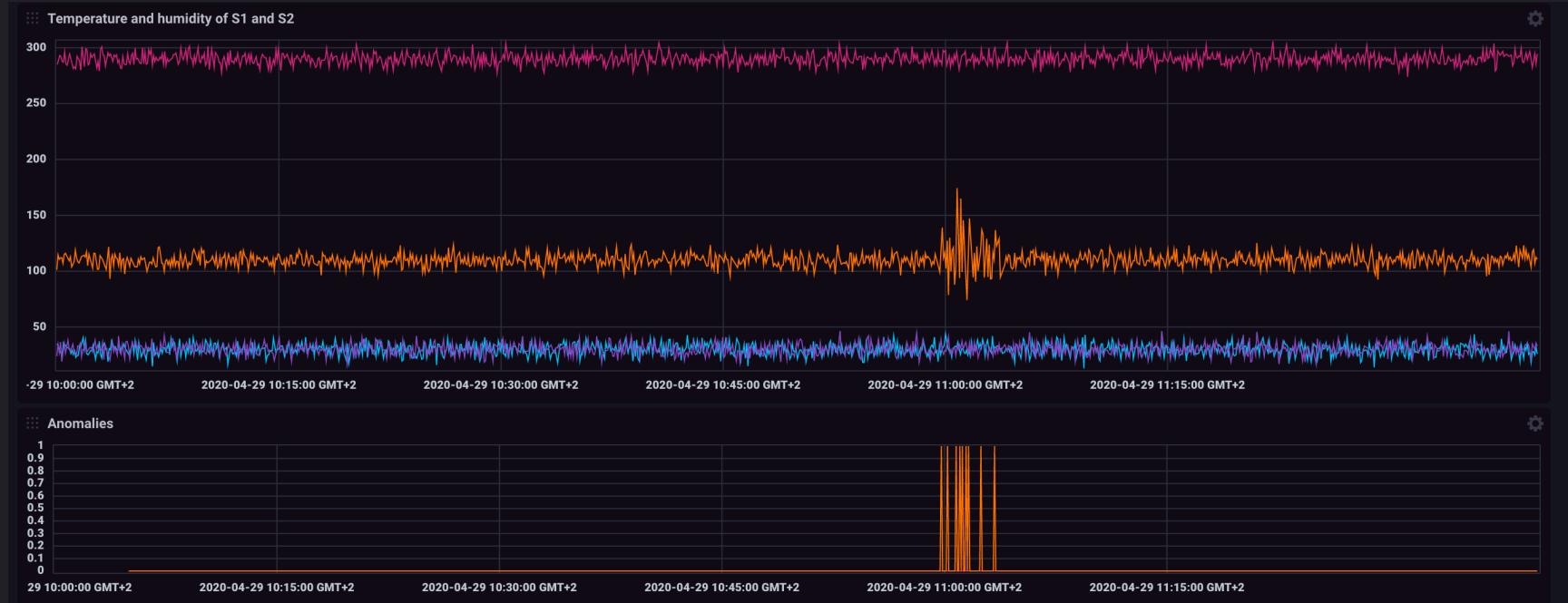
## Use case for the Linear Pizza Oven

- Detect if a failures of the temperature controller of the mozzarella melting area
  - Correct behavior:  $110 \pm 10 \text{ C}^\circ$
  - Anomalous: ?

## Use case for the Linear Pizza Oven

- Detect if a failures of the temperature controller of the mozzarella melting area
  - Correct behavior:  $110 \pm 10 \text{ C}^\circ$
  - Anomalous: let's compute the z-score where
    - the Mean is computed with `aggregateWindow(every: 5m, fn: mean)`
    - the StdDev is computed with `aggregateWindow(every: 5m, fn: stddev)`
    - the Z score is computed `joining` the times-series of all observations with the one of the Mean and the one the StdDev

# Use case for the Linear Pizza Oven – data viz



## Use case for the Linear Pizza Oven – solution 1/3

```
movingAvg = from(bucket: "training")  
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)  
|> aggregateWindow(every: 5m, fn: mean, createEmpty: false)  
|> filter(fn: (r) => r._stop != r._time)  
|> drop(columns: ["_start", "_stop", "_measurement"])  
|> rename(columns: {_value: "avg"})
```

## Use case for the Linear Pizza Oven – solution 2/3

```
movingStddev = from(bucket: "training")  
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)  
|> aggregateWindow(every: 5m, fn: mean, createEmpty: false)  
|> filter(fn: (r) => r._stop != r._time)  
|> drop(columns: ["_start", "_stop", "_measurement"])  
|> rename(columns: {_value: "stddev"})  
  
join1 = join(tables: {all: movingAvg, avg: movingStddev},  
            on: ["_time", "sensor", "_field"])
```

## Use case for the Linear Pizza Oven – solution 3/3

```
allData = from(bucket: "training")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> drop(columns: ["_start", "_stop", "_measurement"])

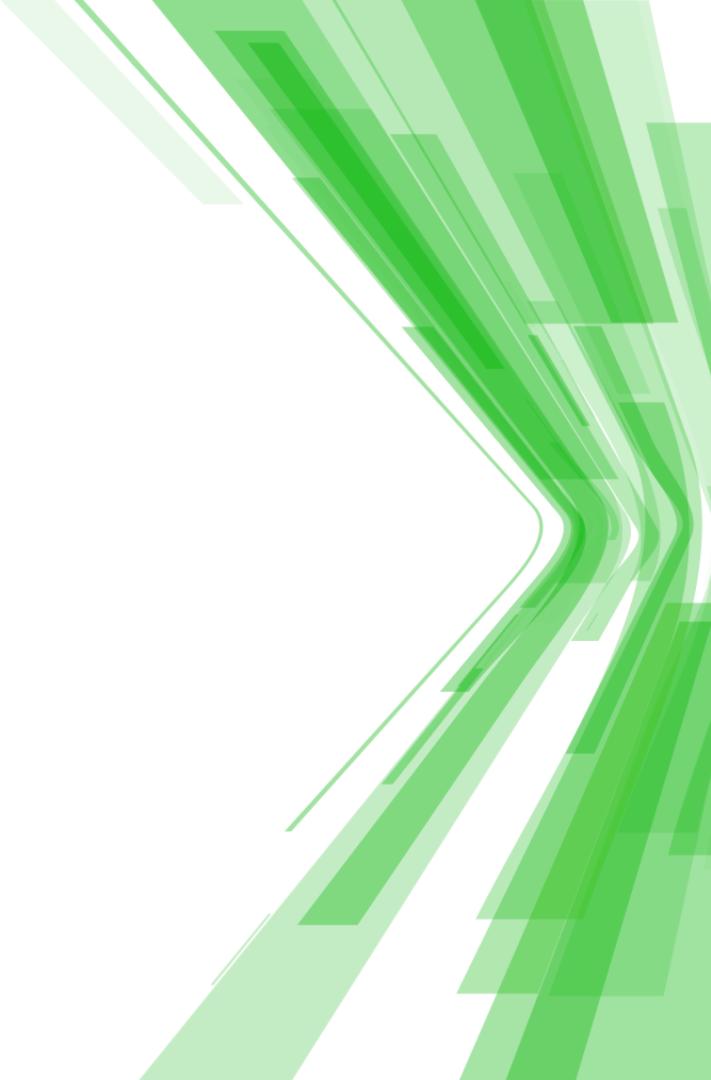
join(tables: {all: allData, j: join1}, on: ["sensor", "_field"])
|> filter(fn: (r) => uint(v: r._time_all)-uint(v: r._time_j) > 0)
|> filter(fn: (r) => uint(v: r._time_all)-uint(v: r._time_j) <=
           (5*60*1000000000))
|> map(fn: (r) => ({ r with zscore:
           if math.abs(x: (r._value - r.avg)/r.stddev)>3 then 1 else 0} ))
```

# A Network Security example

- Finding anomalies in the number of times users inserts a wrong password when they log into a system
  - Refuse the access to a user that inserts the right password at the first time if she gets in average the right password after 3 attempts
- Much more effective than any rule based approach



# Advanced Time Series Analytics



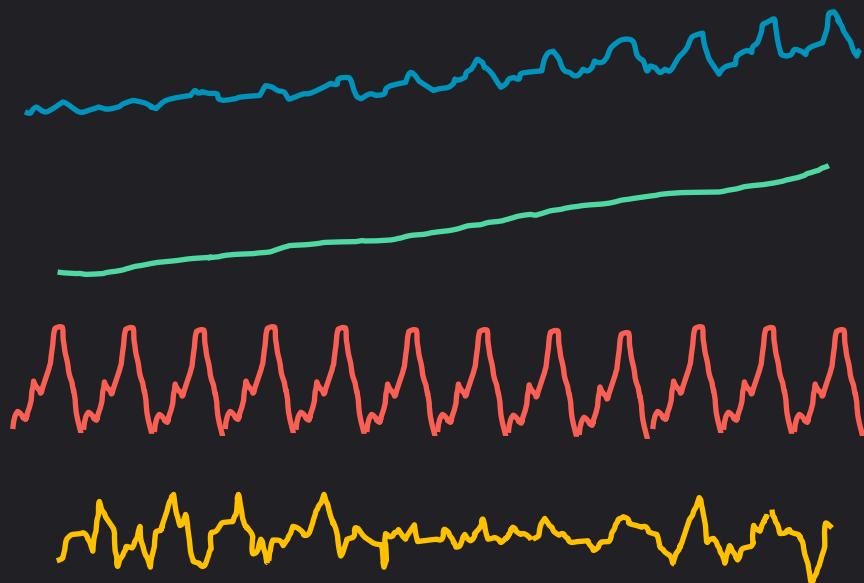
# Holt-Winter's method

Given a **time series**

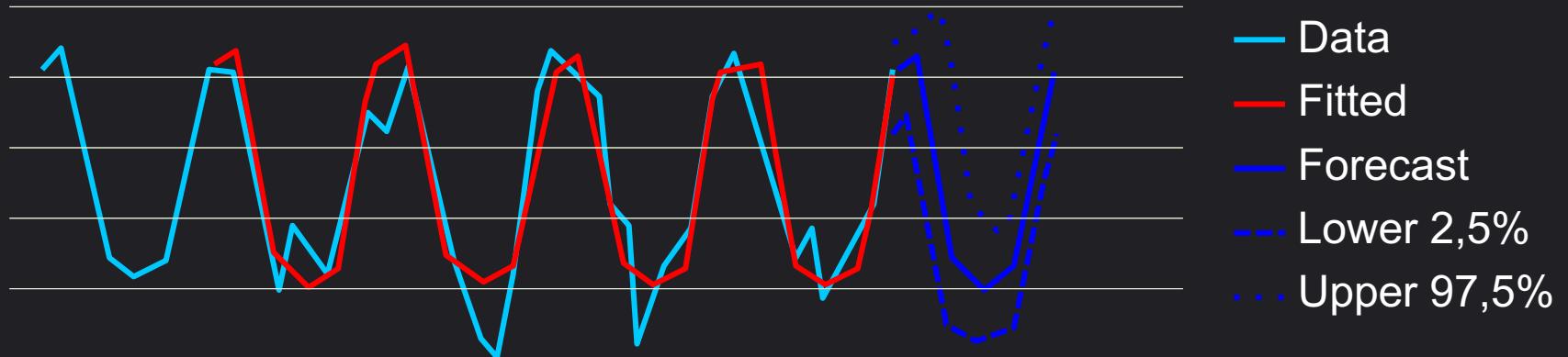
It is often possible to extract

- a **trend**: a long term variation of the time series.
- a **seasonality**: a short term recurring pattern

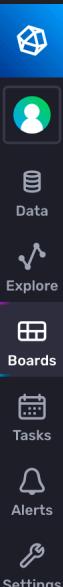
What is left is call **residual**



# Forecasting with Holt-Winter's method



# Holt-Winter's in Flux



## Time-series forecasting



## Holt-Winter's in Flux

```
from(bucket: "training")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> filter(fn: (r) => r._measurement == "fill_level")
|> filter(fn: (r) => r._field == "value")
|> filter(fn: (r) => r.tank == "B2")
|> aggregateWindow(every: 1m, fn: first)
|> holtWinters(n: 10, seasonality: 8, interval: 1m)
```

# What (else) can I do with Flux?

## What (else) can I do with flux?

- Map/Reduce
- Derivative, Difference, Integral
- Histograms
- Regular expression matching
- String functions
- Go Math library parity
- Quantile
- Conditional Expressions
- Cross-measurement calculations
- Conditional expressions
- Sorting
- Data Generation
- In-language test assertions
- ...
- And more every day!



# Advanced usages of flux

---

**Emanuele Della Valle** Prof. @ Politecnico di Milano & Partner @ Quantia Consulting  
**Marco Balduini** Founder & CEO @ Quantia Consulting