Advanced Data Analysis

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

Exploring flux

Custom Functions

Recall: What Is Flux?

- Flux is a functional data scripting and query language
- Written to be:
  - Useable: easy to learn
  - Readable: developers read more code than we write
  - **Composable: developers can build onto the language**
  - Testable: queries are code
  - Contributable: open source contributions matter
  - Shareable: developers read more code than we write

Applying functions to each raw

- The map() function applies a function to each record in the input tables. The modified records are assigned to new tables based on he group key of the input table.
- Example

```
from(bucket:"foo")
    |> range(start: -1h)
    |> filter(fn: (r) => r._measurement == "samples")
    |> map(fn: (r) => ({ _value: r._value * r._value }))
    |> filter(fn: (r) => r._value > 23.2)
```

Defining and using a costume function

- Syntax

  <<function name>> = (<<variable>>*) => <<implementation>>

- Example

```
squared = (r) => r*r
from(bucket:"foo")
    |> range(start: -1h)
    |> filter(fn: (r) => r._measurement == "samples")
    |> map(fn: (r) => ({ _value: squared(r._value)}))
    |> filter(fn: (r) => r._value > 23.2)
```

Defining a costume pipe forwardable function

- Syntax

  <<function name>> **= (tables=<-) => tables |>** << implementation >>

- Example

```
allSquared = (tables=<-) =>
        tables => map(fn: (r) => squared(r._value))
from(bucket:"foo")
        |> range(start: -1h)
        |> filter(fn: (r) => r._measurement == "samples")
        |> allSquared()
        |> filter(fn: (r) => r._value > 23.2)
```

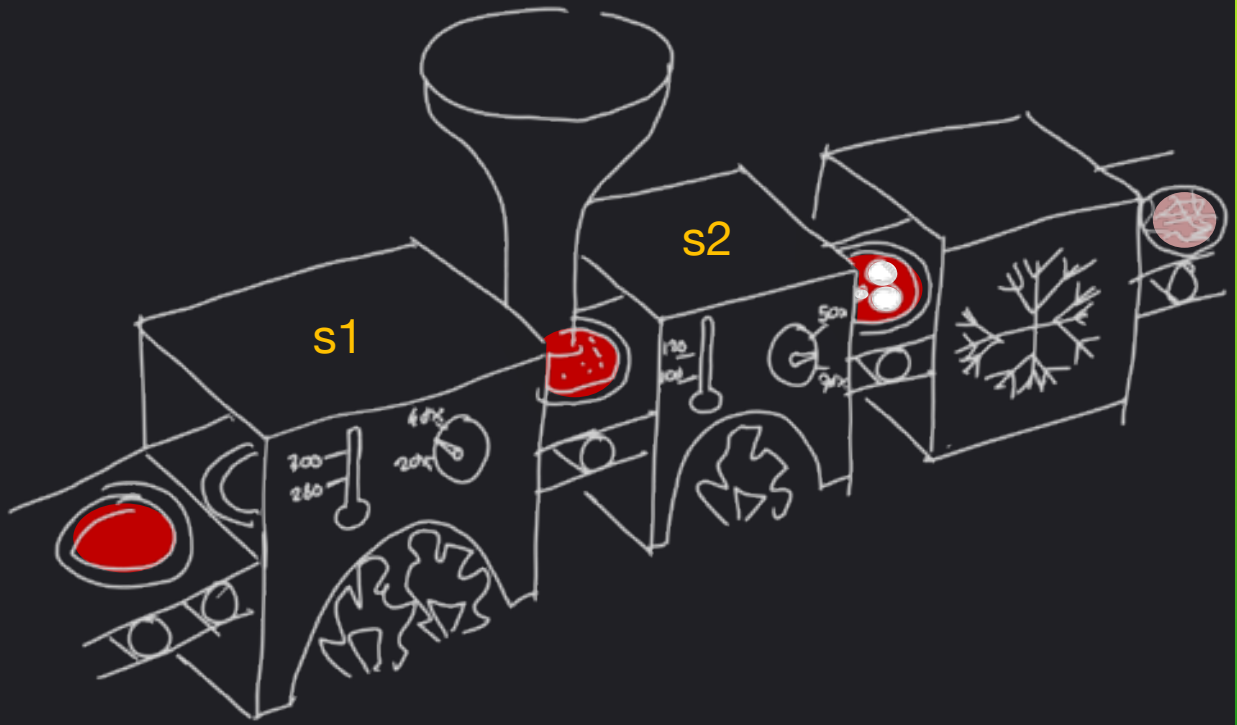
Let’s get dirty!

8

Continuous Linear Pizza Oven

Learning goals:

- Map functions
- Custom functions



Task

Extract the measurements from the cooking base area and correct them by subtracting a delta of 5°C to each value

- Use an inline map
- Create a custom function to be used in the inline map
- Create a custom function that contains a map

Take home message

- Flux is extensible exploiting map() and the custom functions
- The `r with` clause in a map keep the original row structure
- A custom function can work on multiple lines or on each line of a table
  - The `<-` pipe-receive expression allows the function to capture the input tables

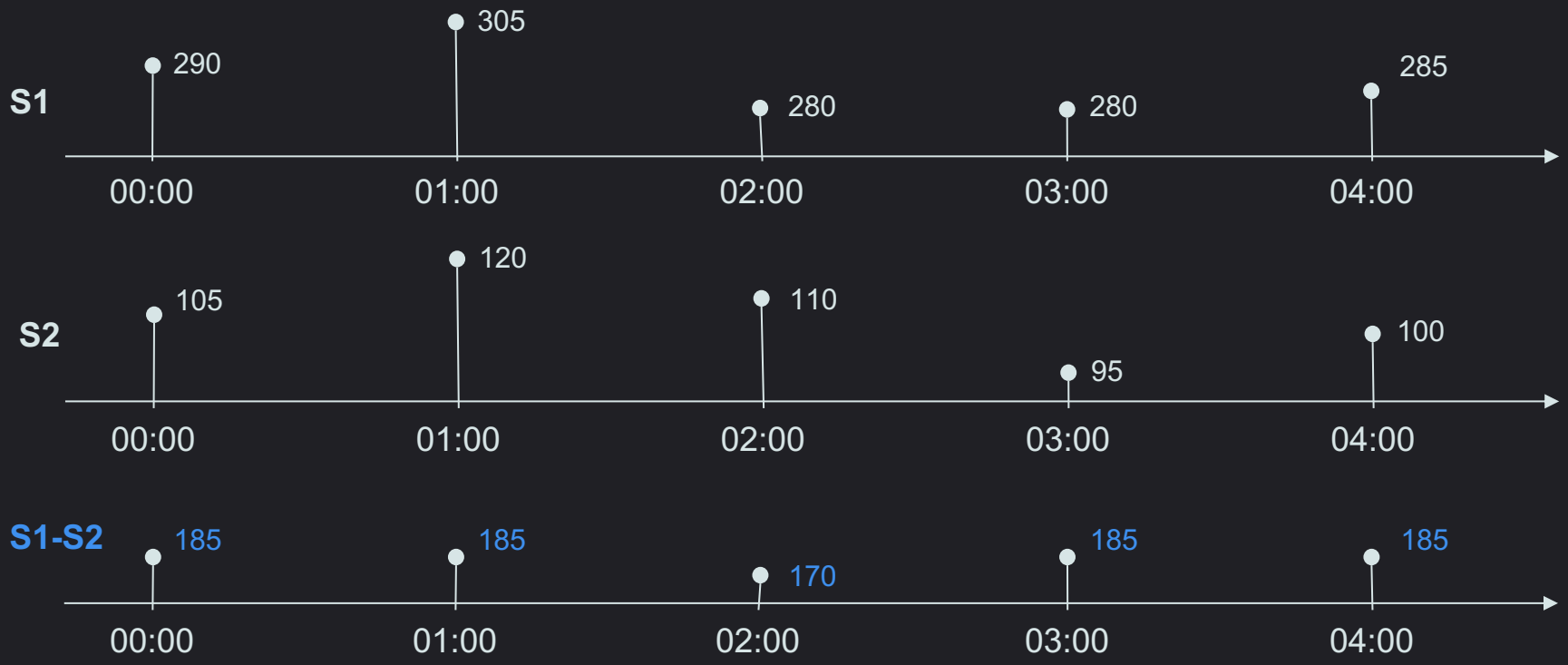
Exploring flux

Joining Time Series

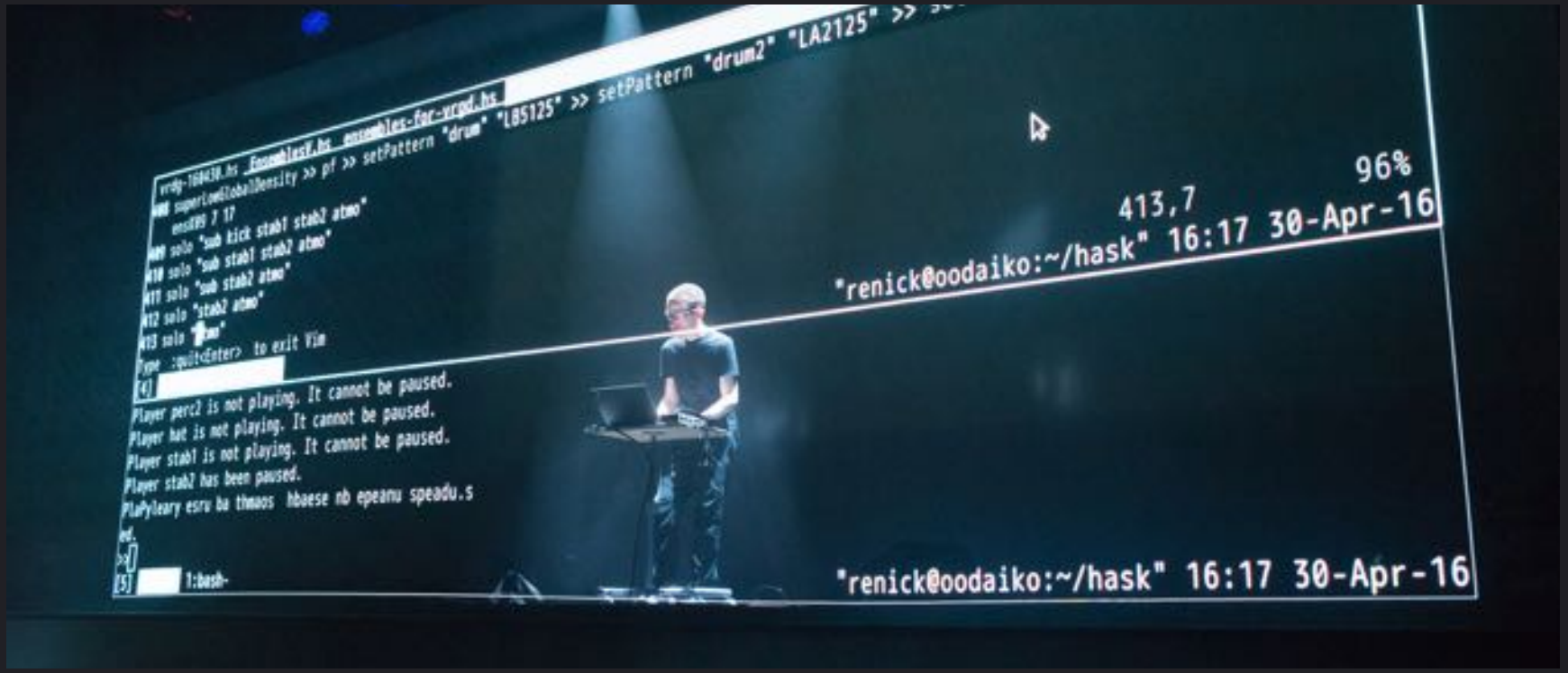
Joining two time series

The ideal case

The easy case



Let’s do some live coding



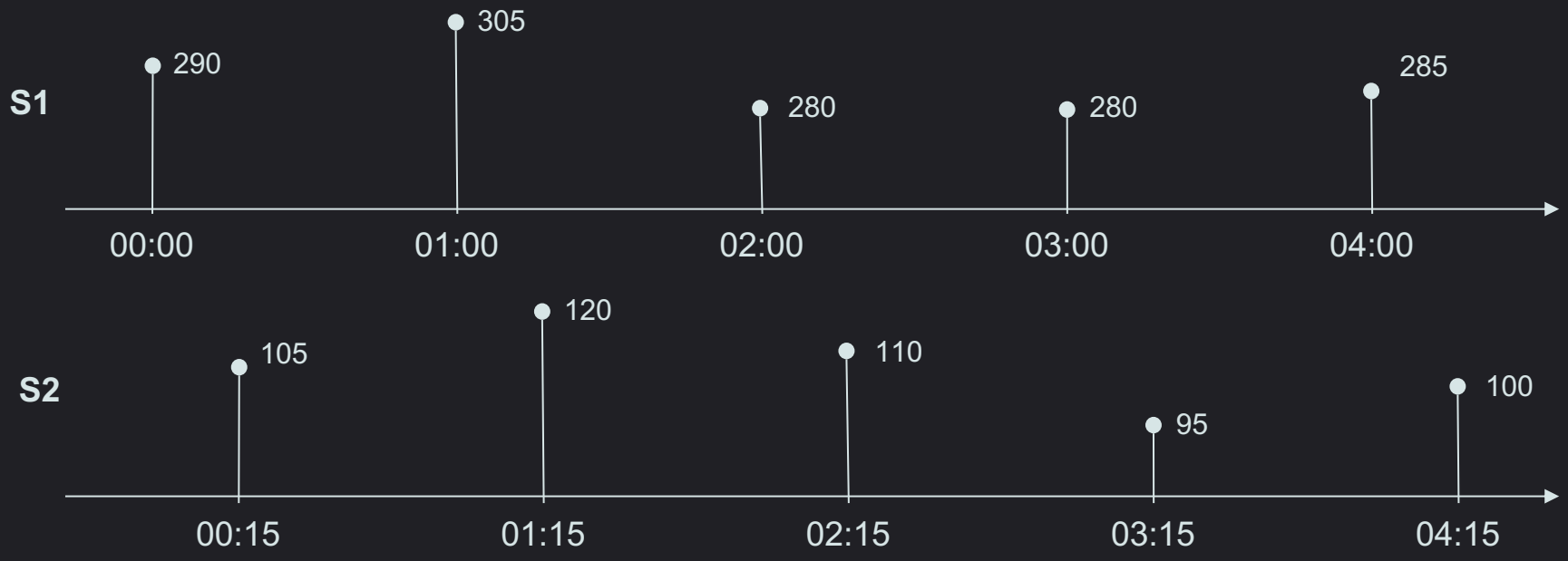
Here we go

```
temp = from(bucket: "training")
  |> range(start: 2019-10-01T00:00:00Z, stop: 2019-10-01T00:05:00Z)
  |> filter(fn: (r) => r._measurement == "iot-oven")
  |> filter(fn: (r) => r._field == "temperature")
tempS1 = temp |> filter(fn: (r) => r.sensor == "S1")
tempS2 = temp |> filter(fn: (r) => r.sensor == "S2")

join = join(  tables: {s1:tempS1, s2: tempS2},  on: ["_time"] )

join |> map(fn: (r) => ({ _time: r._time,
                          _tempDiff: r._value_s1 - r._value_s2 }))
```

It does not work with our data … why?

The reality is always more difficult than expected

A close look to the distribution of our data over time



NOTE: S1 metrics are not synchronized with S2 ones

Let’s discuss about alternatives and code them in flux

- Join on time approximating
  - assuming a fixed delta → `timeShift()`
  - assuming a maximum error → `trucateTimeColumn()`
  - minimizing assumptions → `aggregateWindow()`

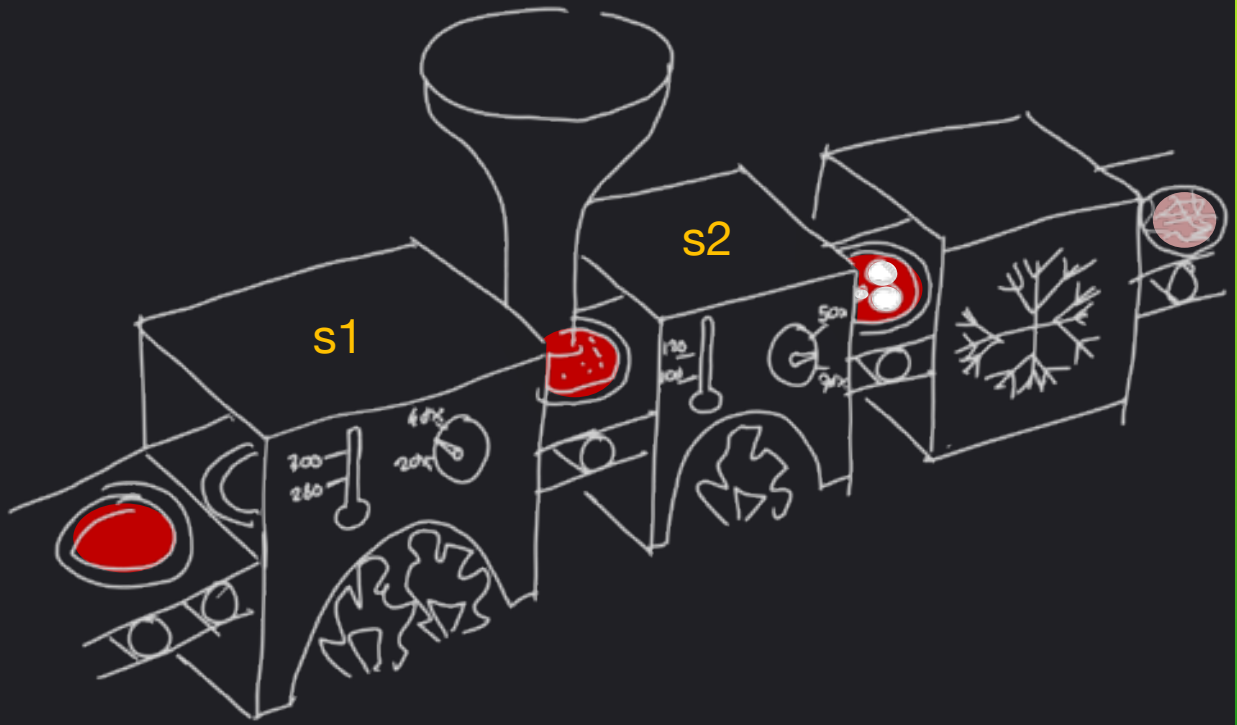
Let’s get dirty!

9

Continuous Linear Pizza Oven

Learning goals:

- Join data



Task

- Extract the difference between the humidity levels of the base cooking area and the mozzarella melting area. Find if the differences are lower than 20% or greater than 30%

Take home message

- Data From different time series can be joined on time
- In few cases, you can use the exact join
- In general, you need to manipulate data before joining them
  - Time shift data
  - Truncate time column
  - Aggregate over (synchronized) windows

Data Analysis Cont.

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