

Data Analysis and Rapid Prototyping using Dashboards

Raimund Kammering, Deutsches Elektronen-Synchrotron DESY Hamburg, Germany



Why Dashboards

Not only since the Corona Pandemic are Dashboards popular in the **Data Science community**

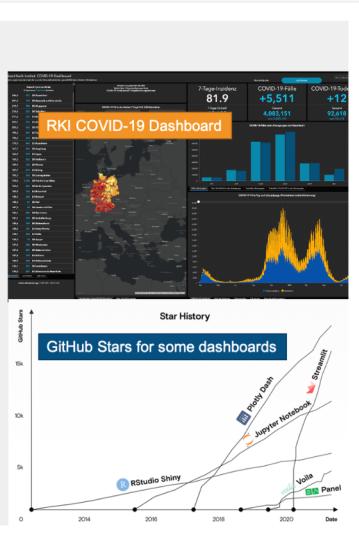
Running in nearly every web browser, they offer **easy access** also from many **different locations**

Classical accelerator control system user interfaces (UI) are mainly aiming for **fast** and **responsive** user experience

These UIs mostly use **predefined views** of the relevant data and thereby often **do not offer much flexibility** in terms of e.g. **data processing** or correlating over different domains

Looking at these **statistical properties**, languages like **python**, **R** or **Julia** provide large well established frameworks/libraries

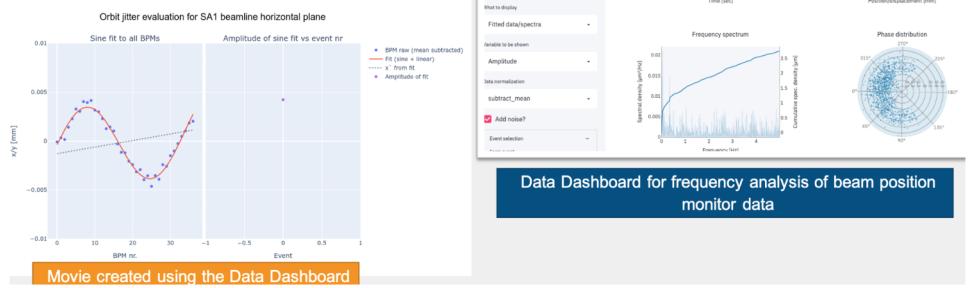
Using such libraries the step to **integrate complex routines/algorithms** in these days dashboard frameworks is **straight forward**



Data Evaluation Dashboard

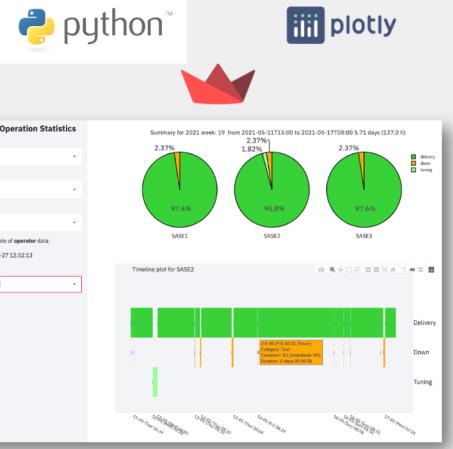
As a next project we **ported** a originally in **MATLAB** developed data evaluation to **python** and implemented the **UI** again using **streamlit** and **plotly**

Again this step could easily be done and let to an these days **routinely used** high level **diagnostics** for the European XFEL accelerator



First experience - the Statistics Dashboard

As a test case we developed a workflow to monitor the **statistics** of the different **operation states** at the European XFEL



Using **python** the data is captured from the electronic logbook used for the operation of the facility

This data is now processed to derive the relevant statistical properties for the visualization

To offer this results online to all relevant personal a **streamlit** dashboard showing the graphics using the **plotly** graphics library

This way the web interface offers not only the in streamlit implemented widgets for e.g. selection of time or beamline, but due to plotly's interactive support, plenty of other features like links to the logbook or additional information in tooltips

Conclusions - outlook

Both dashboards shown here, offer very convenient and nice user interfaces

The ease to alter 'some lines' of code, deploy the changes and have them immediately visible for all users, allows to **rapidly develop** and extend the dashboard

The **rich and comfortable graphics libraries** available in python in combination with **streamlits** widgets offers up-to-date and **powerful visualizations**

Also for educational purposes such an approach looks promising (see movie before)

In contrast to jupyter notebooks, dashboards do not offer direct modifications but a fixed interface, which is in the shown cases exactly what one wants

```

streamlit_example.py
1 import streamlit as st
2 import plotly.express as px
3 import pandas as pd
4 import pydoocs as pdo
5
6 st.write('Hello Dashboard World')
7
8

```

Why Dashboards

Not only since the Corona Pandemic are Dashboards popular in the **Data Science community**

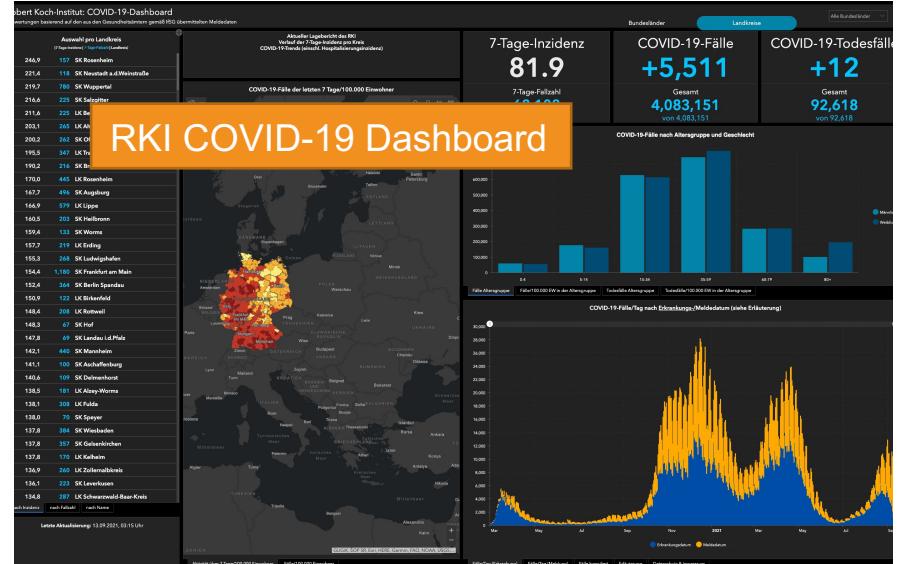
Running in nearly every web browser, they offer **easy access** also from many **different locations**

Classical accelerator control system user interfaces (UI) are mainly aiming for **fast** and **responsive** user experience

These UIs mostly use **predefined views** of the relevant data and thereby often **do not offer much flexibility** in terms of e.g. **data processing** or correlating over different domains

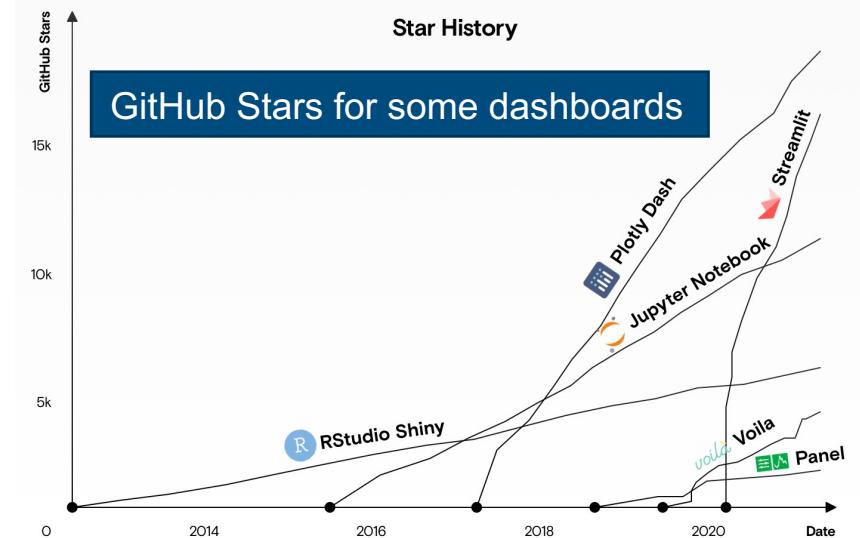
Looking at these **statistical properties**, languages like **python**, **R** or **Julia** provide large well established frameworks/libraries

Using such libraries the step to **integrate complex routines/algorithms** in these days dashboard frameworks is **straight forward**



Star History

GitHub Stars for some dashboards



First experience - the Statistics Dashboard

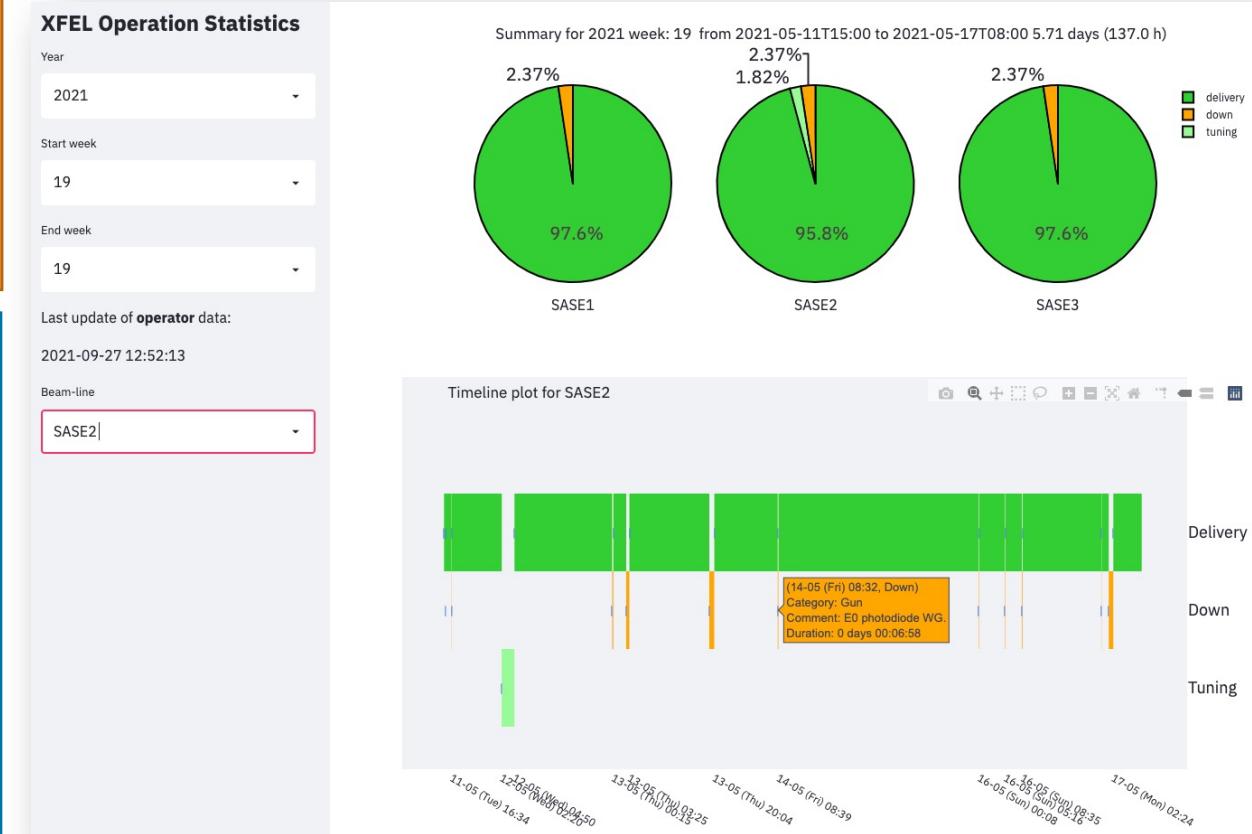
As a **test case** we developed a workflow to monitor the **statistics** of the different **operation states** at the European XFEL

Using **python** the data is captured from the electronic logbook used for the operation of the facility

This data is now processed using the python **pandas** to derive the relevant statistical properties for the visualization

To offer this results online to all relevant people a **streamlit** dashboard showing the graphics using the **plotly** graphics library has been created

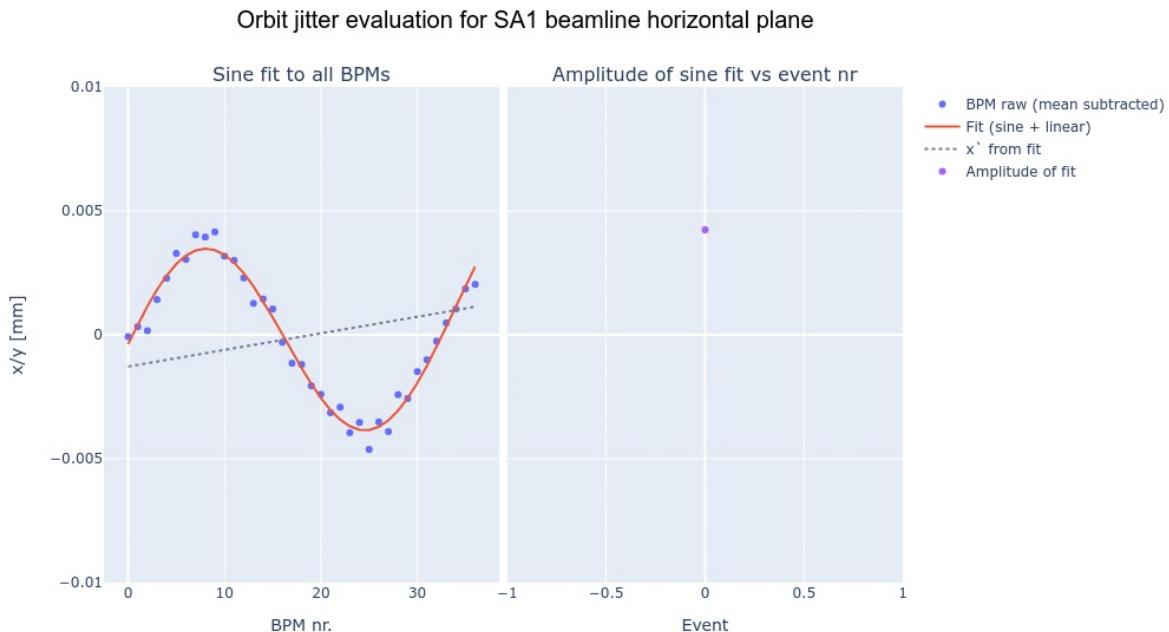
This way the web interface offers not only the in streamlit implemented **widgets** for e.g. selection of time or beamline, but due to plotly's **interactive** support, plenty of other features like **links** to the logbook or additional information in rich **tooltips**



Data Evaluation Dashboard

As a next project we ported a originally in **MATLAB** developed data evaluation to **python** and implemented the UI again using **streamlit** and **plotly**

Again this step could easily be done and led to an these days **routinely used high level diagnostics** for the European XFEL accelerator



Movie created using the Data Dashboard

Beam-line/File selection

Beam-line: SA1

Machine file: bpm_y_20210903T154500...

Experiment file: SA1_fxe_run43.txt

Get data

What to show

What to display

Fitted data/spectra

Variable to be shown

Amplitude

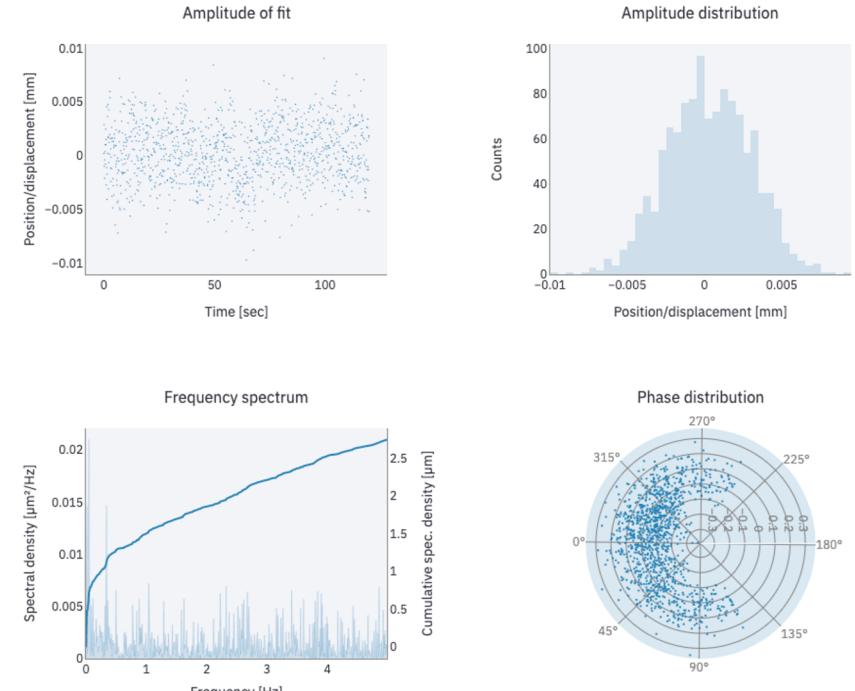
Data normalization

subtract_mean

Add noise?

Event selection

XFEL DOOCS DAQ Data Dashboard



Data Dashboard for frequency analysis of beam position monitor data

Conclusions - outlook

Both dashboards shown here, offer very convenient and nice user interfaces

The ease to alter ‘some lines’ of code, deploy the changes and have them immediately visible for all users, allows to **rapidly develop** and extend the dashboard

The **rich** and comfortable **graphics libraries** available in python in combination with **streamlits** widgets offers up-to-date and **powerful visualizations**

Also for educational purposes such an approach looks promising (see movie before)

In contrast to jupyter notebooks, dashboards do not offer direct modifications but a fixed interface, which is in the shown cases exactly what one wants

