



EUROPEAN
SPALLATION
SOURCE

Jupyter Examples for Neutron Scattering Facilities

Jonathan Taylor
European Spallation Source

Where is my Data?

How do I get it?

How do I look at it?

What does it mean?

Is it right?

REDUCING THE BOTTLENECK EFFECT:
"What we're trying to do here is **expedite the time to discovery.** Scientists should be able to **focus on their science** without having to become experts in data management."

—Shawn McKee
research scientist in physics



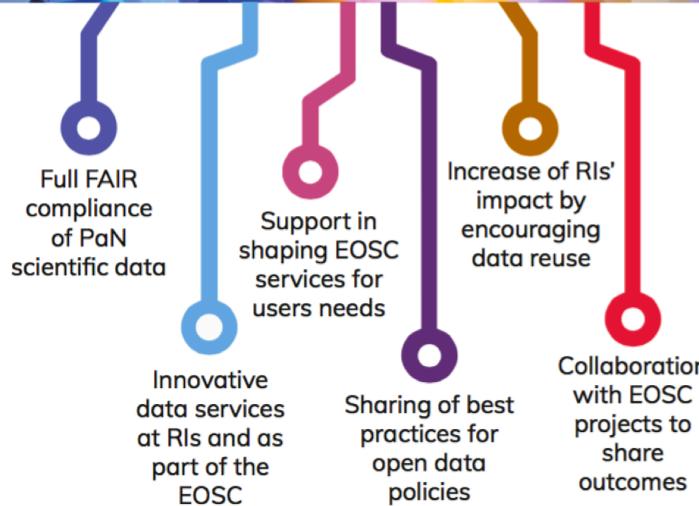
* Facility users are not expected to be experts in scientific computing

Summary



- Jupyter is a powerful tool for scientific computing at user facilities.
- Facility users need a range of tools
- Facility users have a range of experience
- Jupyter provides a unique data management function - can capture rich meta data
- Some examples of what can be developed in a short (ish) space of time.

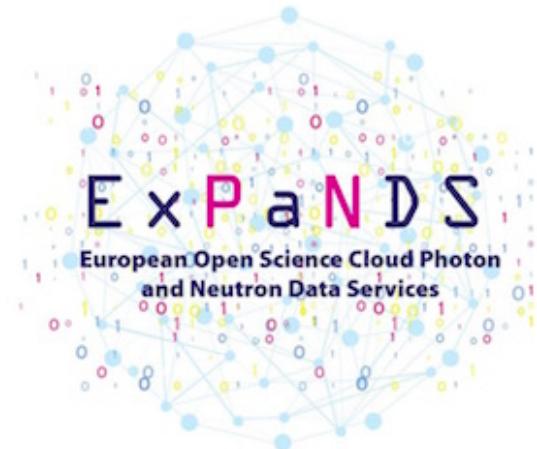
PaNOSC



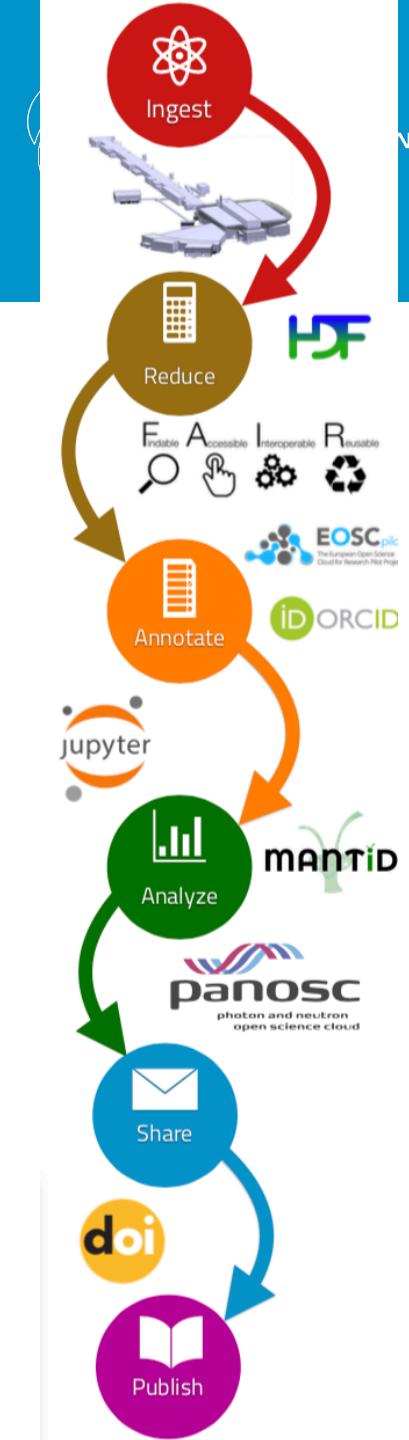
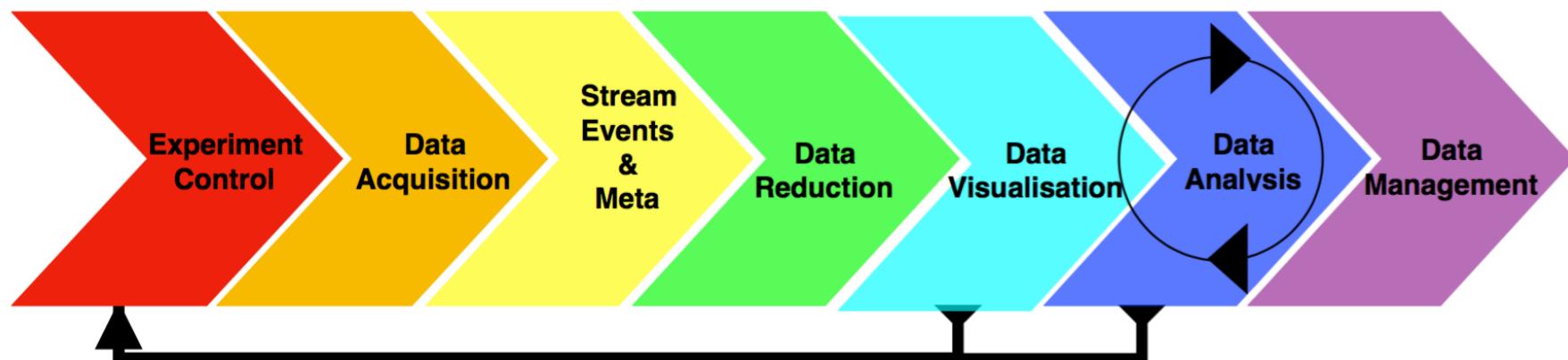
 Analyse

- PaNOSC (Photon and Neutron Open Science Cloud) links petabytes of data between CERIC, ELI, ESRF, ESS, ILL, XFEL
- Scientists can analyse and visualise cross-facility data online with Jupyter notebooks
- Hit the ground running, no need to wait for lengthy data downloads, or slow software installation
- Be ready for EOSC, the European Open Science Cloud

Vision- PaNOSC will help fasttrack new Photon and Neutron facilities, improve data services for existing users, create a new class of virtual users of open data, help build and integrate the PaN RIs to the EOSC.



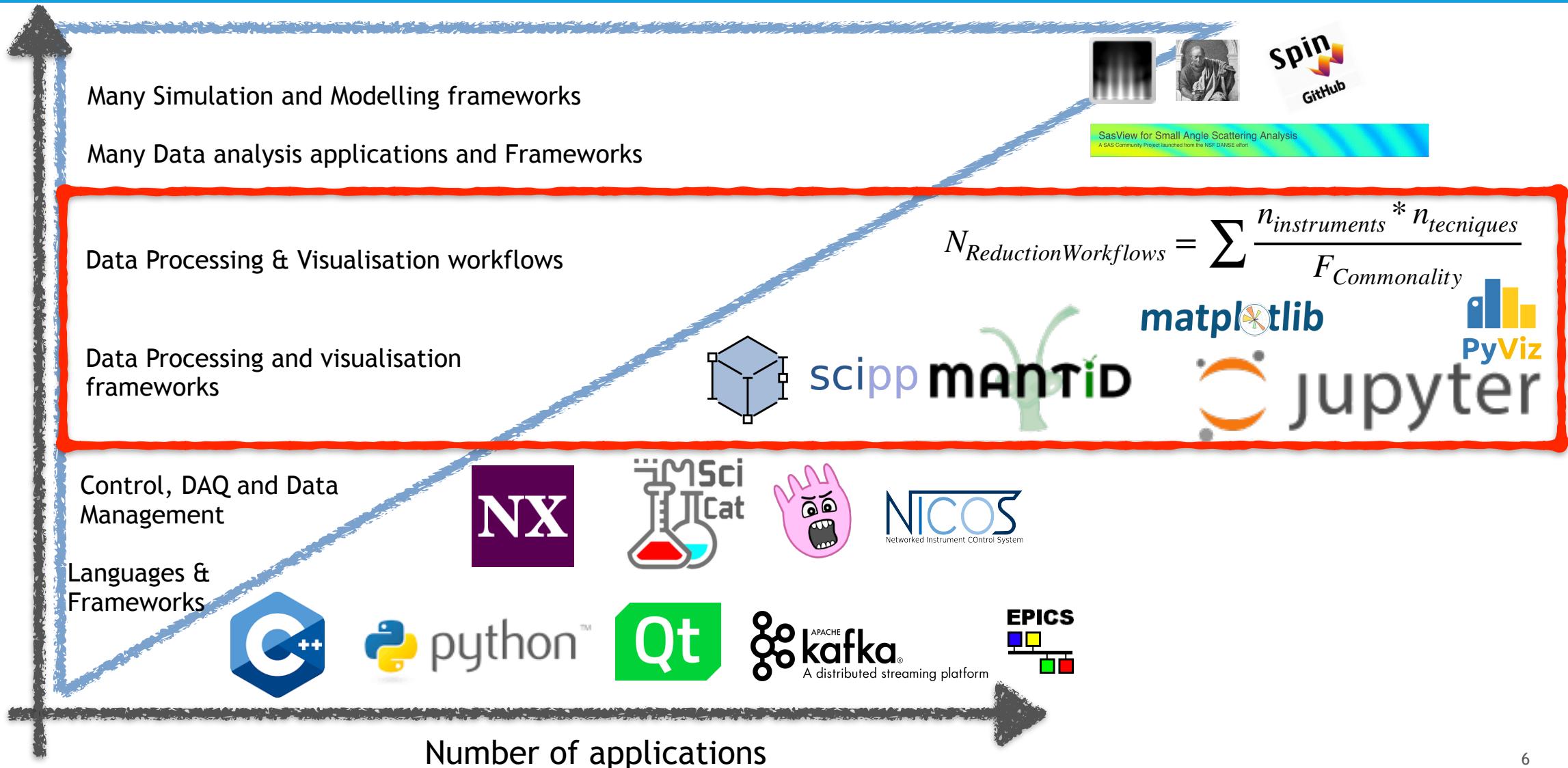
Data Pipelines



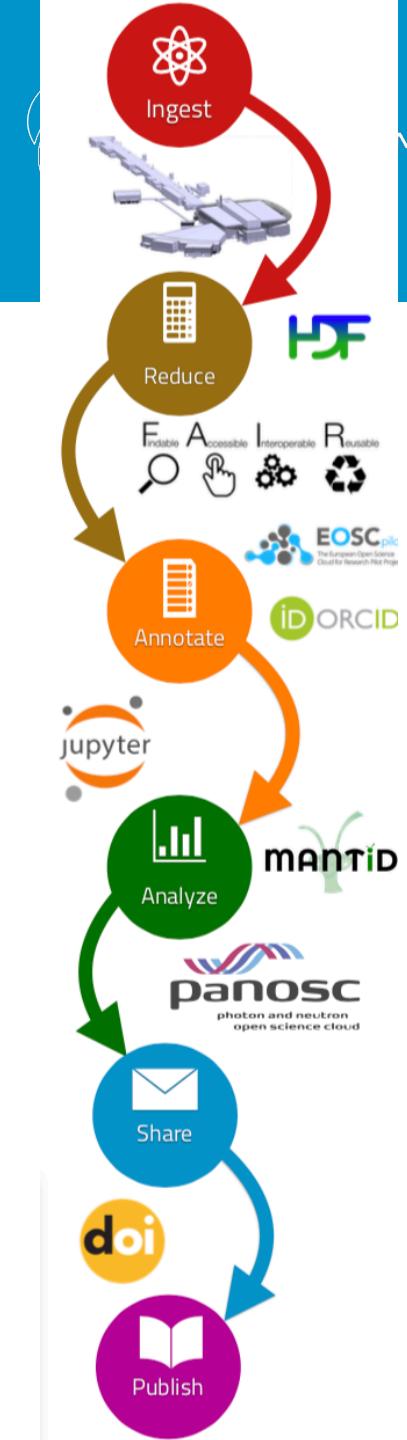
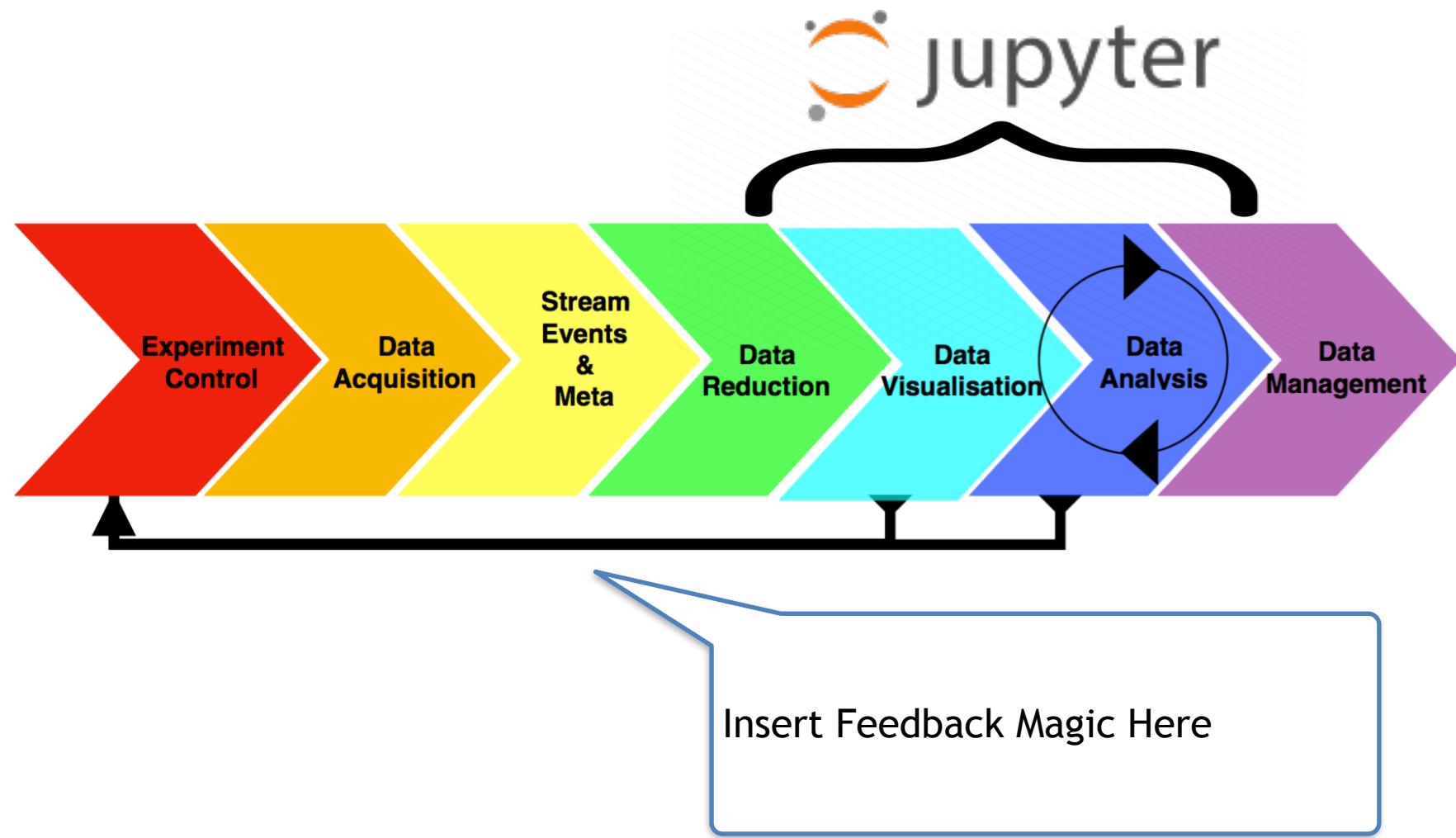
Scientific Application Stack ESS (& Neutron scattering)



Level of community development



Data Pipelines



The jupyter EcoSystem



TOOLBOX · 30 OCTOBER 2018

Why Jupyter is data scientists' computational notebook of choice

An improved architecture and enthusiastic user base are driving uptake of the open-source web tool.



Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.



Language of choice

Jupyter supports over 40 programming languages, including Python, R, Julia, and Scala.



Share notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the [Jupyter Notebook Viewer](#).



Interactive output

Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

Advantages.

- Clean Flexible Architecture

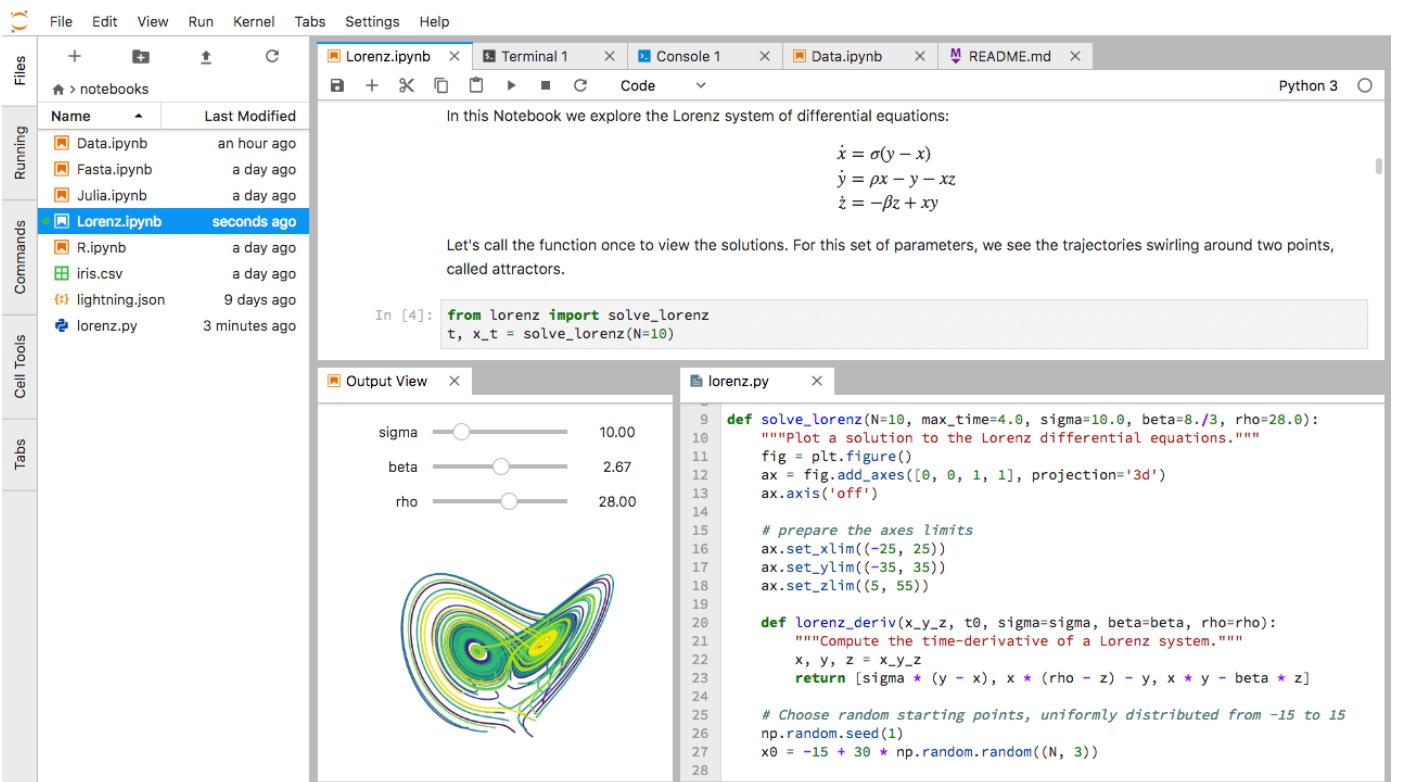
- Compute & storage can be close
- No need to transfer large data volumes
- Facility and user developed methods can coexist

- Web Interface

- Good UX
- High level of community commitment / development
- Code development and execution
- 1D, 2D, 3D Visualisation of data

- Data Management / FAIR

- Persistence for analysis workflow & rich meta data
 - Text, Math, Citations, Links ...
- Interfaces to catalogue both directions and storage



The screenshot shows a Jupyter Notebook environment. On the left, a sidebar lists files: notebooks (Data.ipynb, Fasta.ipynb, Julia.ipynb), Running (Lorenz.ipynb, R.ipynb, iris.csv, lightning.json), Commands (lorenz.py), Cell Tools, and Tabs. The main area has tabs for Lorenz.ipynb, Terminal 1, Console 1, Data.ipynb, README.md. The Lorenz.ipynb tab shows code for exploring the Lorenz system of differential equations:

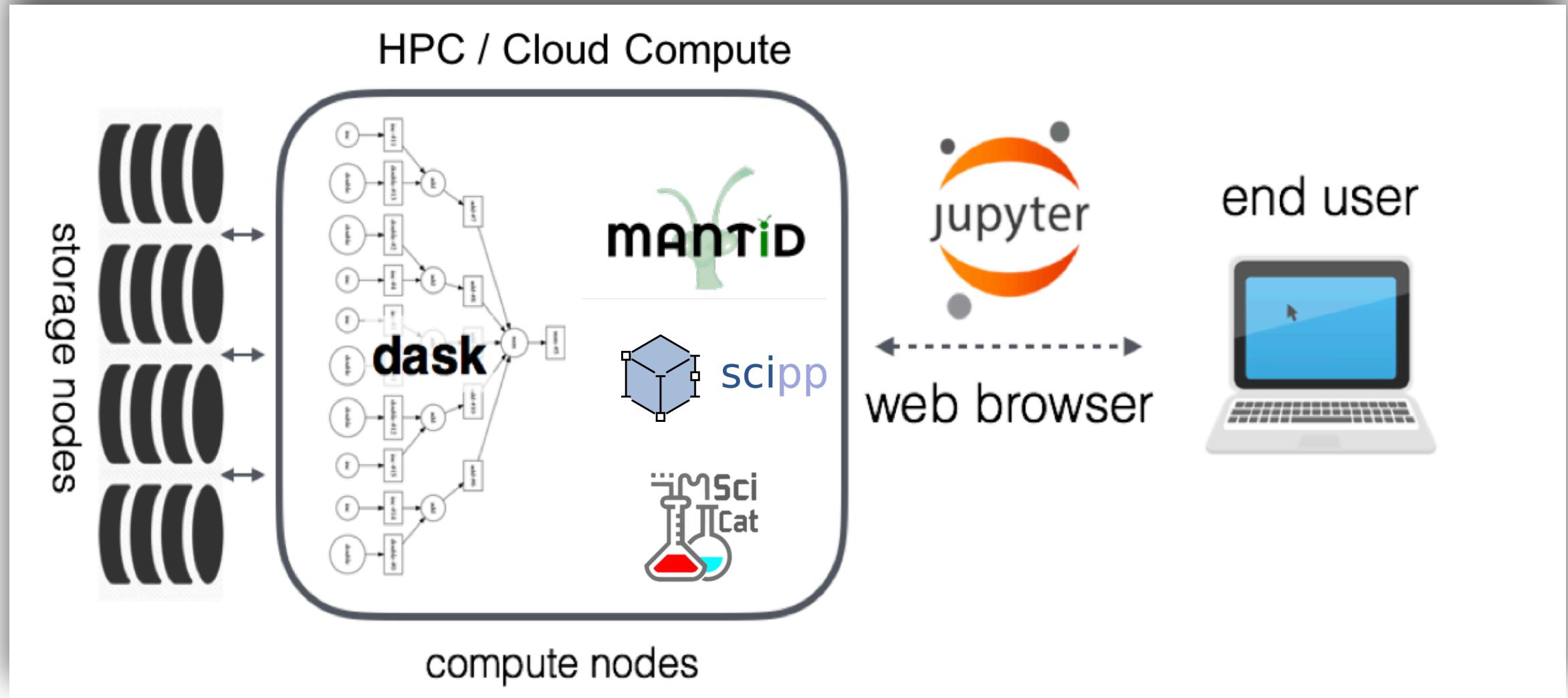
```
In this Notebook we explore the Lorenz system of differential equations:
 $\dot{x} = \sigma(y - x)$ 
 $\dot{y} = \rho x - y - xz$ 
 $\dot{z} = -\beta z + xy$ 

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

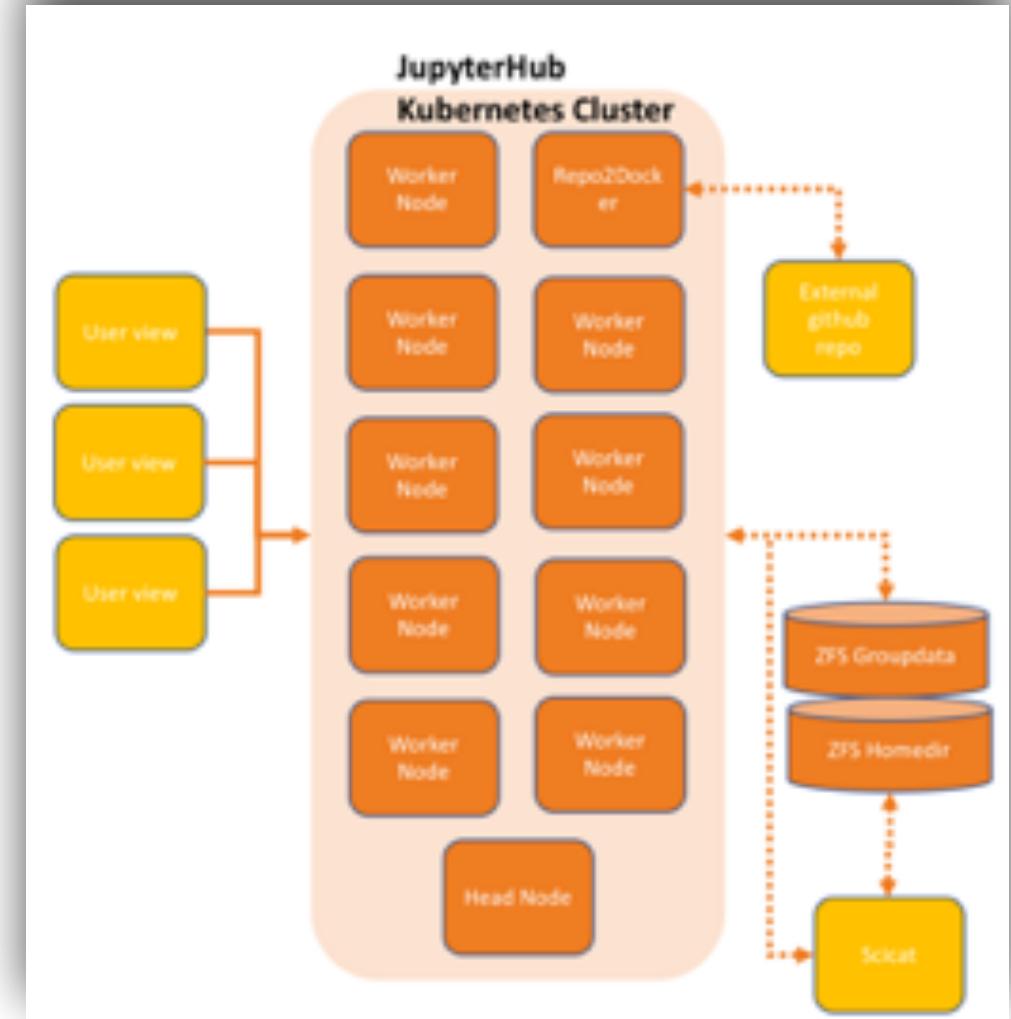
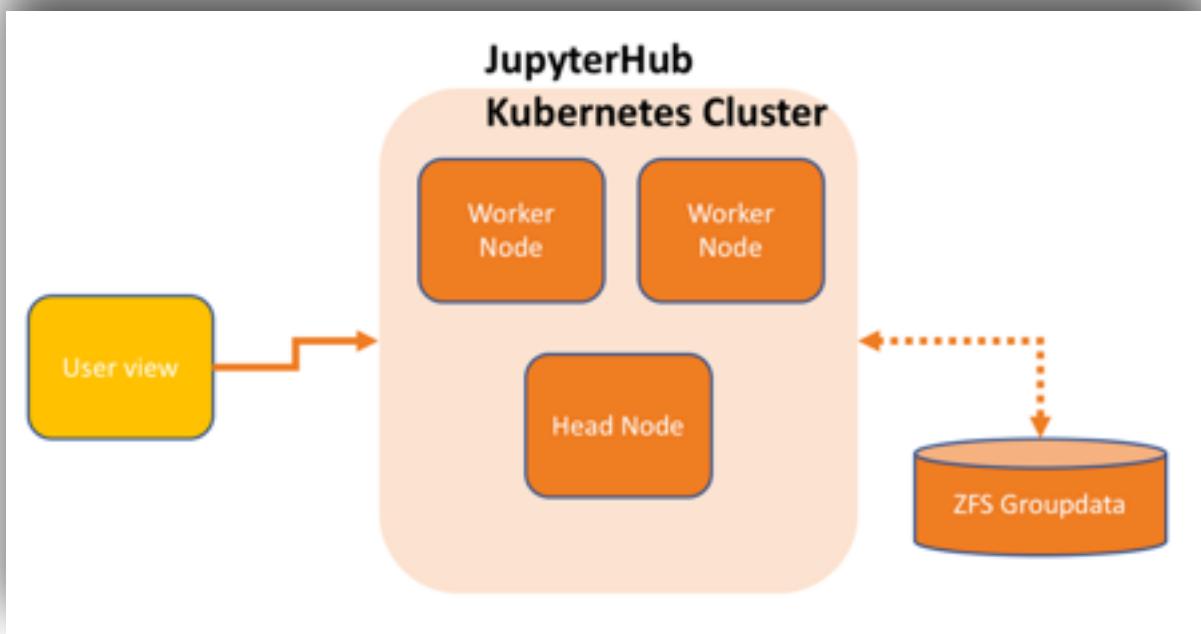
In [4]: from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

Below the code, there's an Output View showing sliders for parameters sigma (10.00), beta (2.67), and rho (28.00). To the right is a 3D plot of the Lorenz attractor, and further right is the lorenz.py file containing the source code for the Lorenz system.

How could this look for ESS

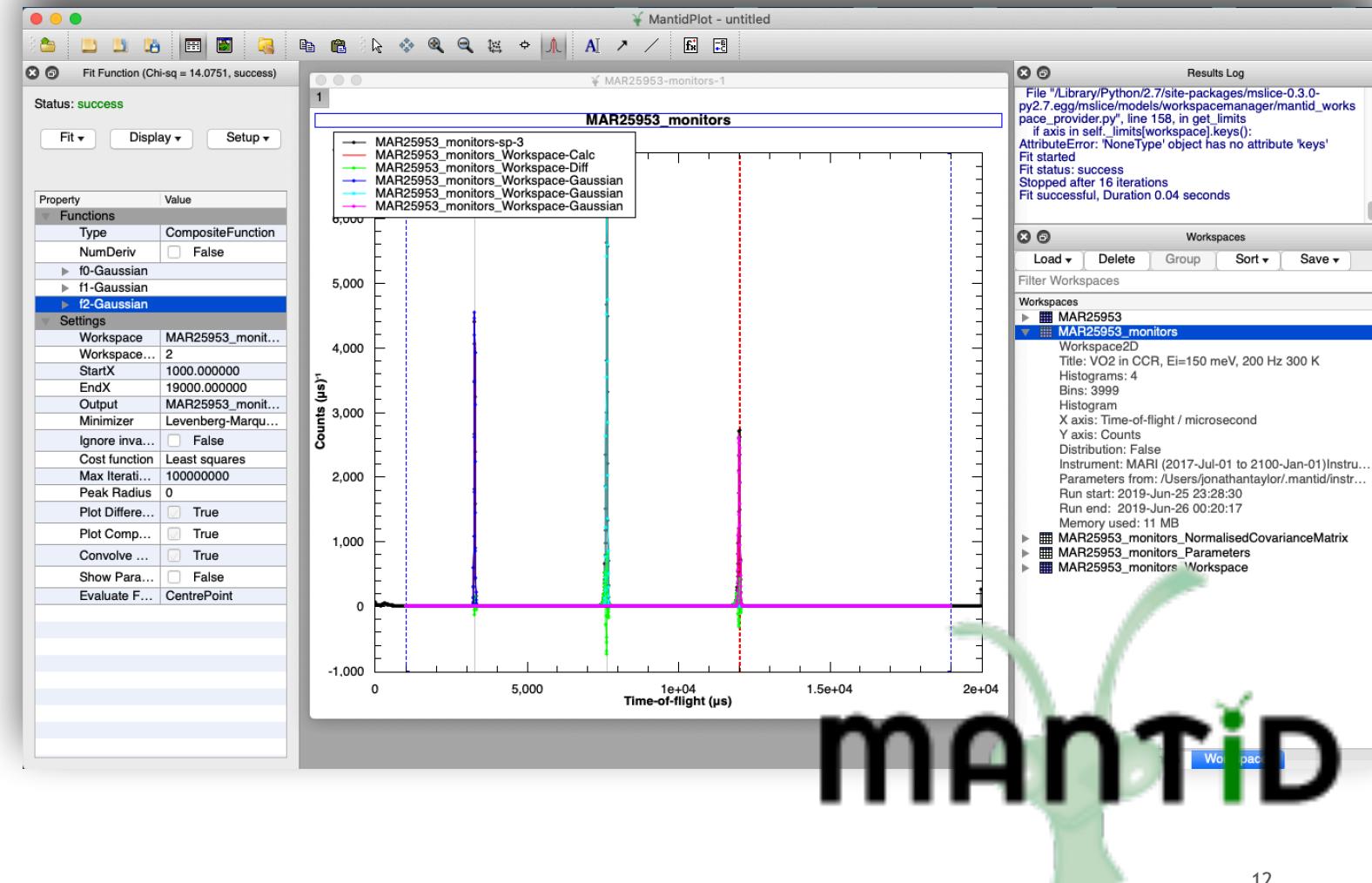
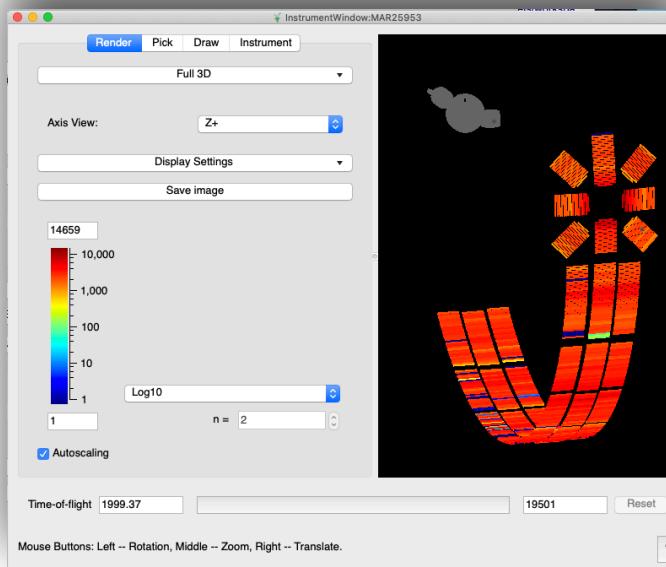


Jupyter Hub at DMSC



What Are (Most) Users Comfortable With.

- GUI for interacting with data.
- Relies on code and data format to persist data history
- No opportunity to persist Rich information
- Some complexity with provisioning architecture



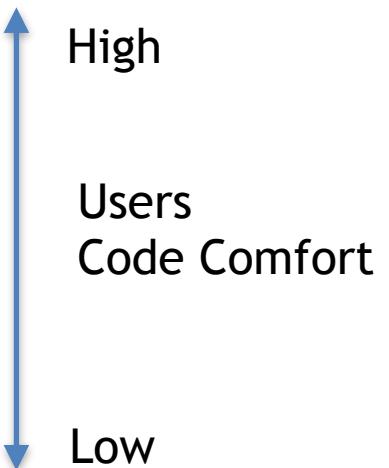
Questions



- Not all users are confident with a code / script based environment
- Can it be used as a direct replacement for traditional GUI delivery for facility reduction & analysis?

Jupyter examples for Neutron data.

- Some example concepts for deploying data processing using Jupyter for facility users
- Use Standard Notebooks - Populated with pre defined functions and scripts
- Decorate a note book with widgets to steer the user in a familiar directions
- Execute Notebooks as web applications to provide a GUI look and feel



Examples are all from MARI @ ISIS

- Code cell allows data reduction from Time of Flight to Energy Transfer

```
In [9]: Hide Prompts □ Hide Code □ Hide Outputs □

from Direct.DirectEnergyConversion import *
import Direct.dgreduce as dgreduce

iliad_setup=dgreduce.setup
iliad=dgreduce.arb_units
iliad_abs=dgreduce.abs_units

iliad_reducer = dgreduce.getReducer

#@out.capture()
def reduceSimple(runNumber):
    inst='mar'
    iliad_setup(inst)
    ext='.raw'
    #mapfile='mari_res2013'
    mapfile=pathToMapFile+'mari_121_cor'

    #det_cal_file must be specified if the reduction sends out put to a workspace
    cal_file=pathToData+'MAR19717.raw'
    #hard mask file
    #mask_file='/home/mari/Users/taylor/mask.msk'
    #load vanadium file
    whitebeamfile=pathToData+'MAR19717.raw'
    LoadRaw(Filename=whitebeamfile,OutputWorkspace="wb_wksp",LoadLogFiles="0")

    runs=pathToData+'MAR'+str(runNumber)+'.raw'; ei=12; rebin_params='-10,.05,11'; sum=False
    LoadRaw(Filename=runs,OutputWorkspace="run_wksp",LoadLogFiles="0")
    w1=iliad("wb_wksp","run_wksp",ei,rebin_params,mapfile,fixei=False,det_cal_file=cal_file,norm_method='current')
```

Data processing - Standard scripts

- Decorate with widgets
- Run data reduction
- Visualise data

Voila: Notebook - Chromium

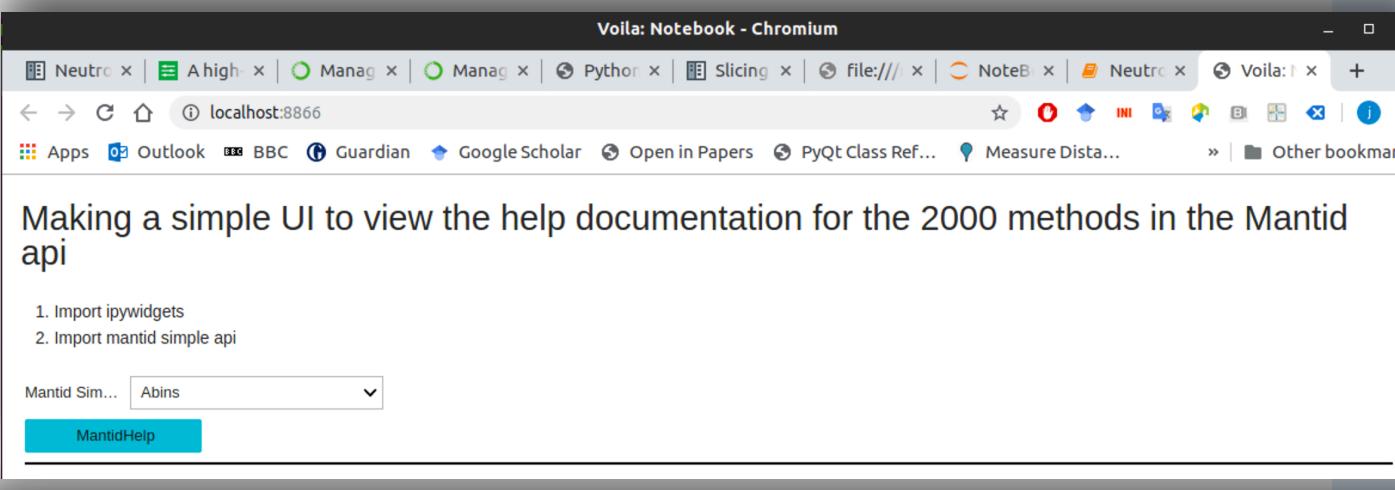
localhost:8866

Making a simple UI to view the help documentation for the 2000 methods in the Mantid api

1. Import ipywidgets
2. Import mantid simple api

Mantid Sim... Abins

MantidHelp



```
In [8]:  
  
%matplotlib inline  
import matplotlib.pyplot as plt  
  
mtd.importAll()  
workSpaceList = %who_ls Workspace2D  
workSpaceListWidget = widgets.Dropdown(  
    options=workSpaceList,  
    description='WorkSpace:',  
    disabled=False,  
)  
  
buttonSelectWorkspace = widgets.Button(description="Select Workspace",layout=widgets.Layout(width='auto', grid_area="#display(buttonPlotSpectrum)")  
buttonPlotSpectrum = widgets.Button(description="plot spectrum",layout=widgets.Layout(width='auto', grid_area='main')  
#display(buttonPlotSpectrum)  
buttonPlot2D = widgets.Button(description="plot 2D",layout=widgets.Layout(width='auto', grid_area='main'))  
#display(buttonPlot2D)  
clearPlots = widgets.Button(description="ClearPlots",layout=widgets.Layout(width='auto', grid_area='main'))  
#display(buttonPlot2D)  
text_field_spec = widgets.Text(value='Spectrum_number')  
PlotOutput = widgets.Output(layout={'border': '1px solid black'})  
TextOutput = widgets.Output(layout={'border': '1px solid black'})  
  
list_widgets = [  
    widgets.VBox([workSpaceListWidget,buttonSelectWorkspace,TextOutput]),  
    widgets.VBox([buttonPlotSpectrum,text_field_spec]),  
    buttonPlot2D, widgets.VBox([clearPlots, PlotOutput])]  
  
accordion = widgets.Accordion(children=list_widgets)  
accordion.set_title(0, 'Select Data')  
accordion.set_title(1, '1D Plot')  
accordion.set_title(2, '2D Plot')
```

In [6]:

Reduce Data

Clear

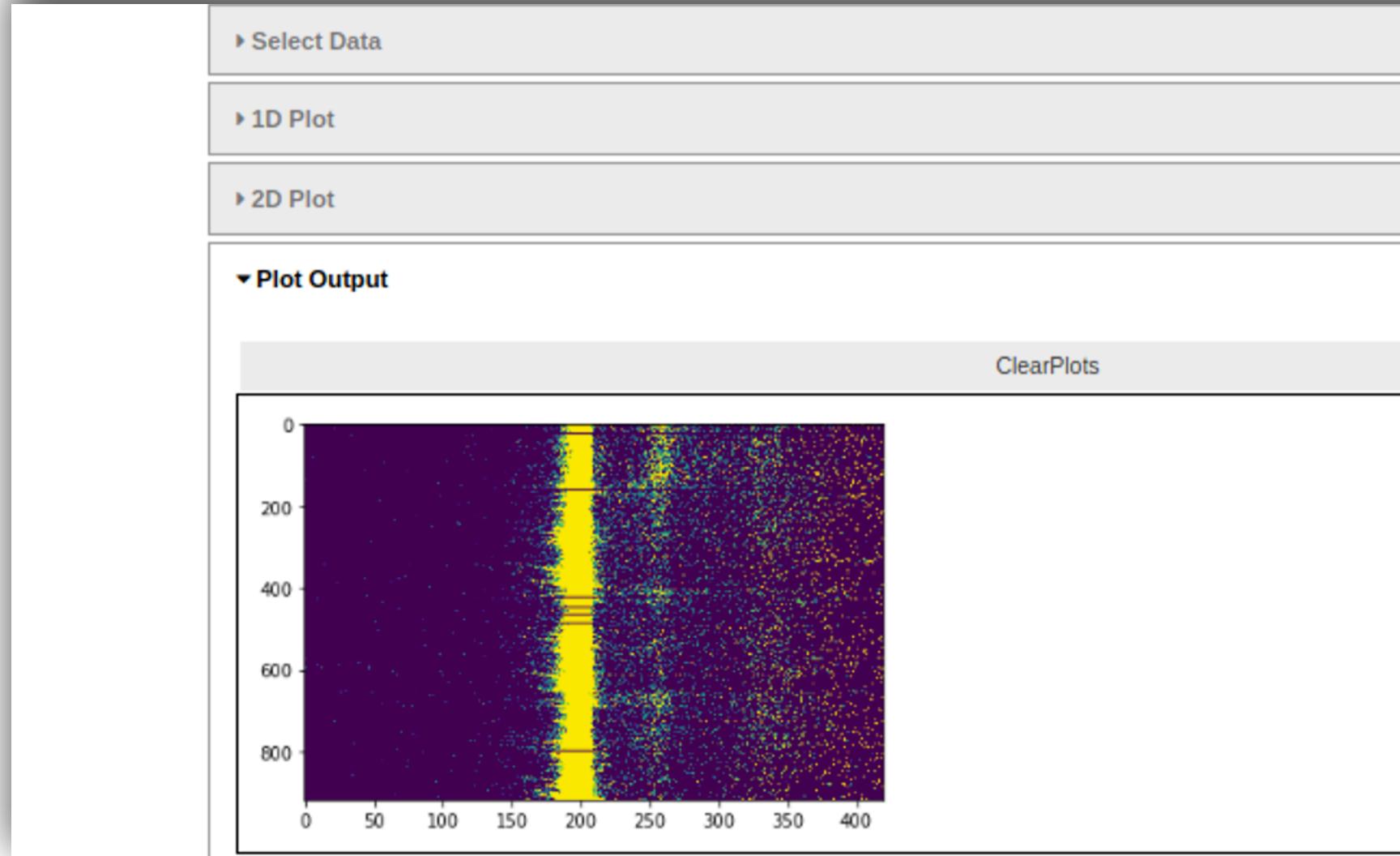
20000

Hide Prompts Hide Code Hide Outputs



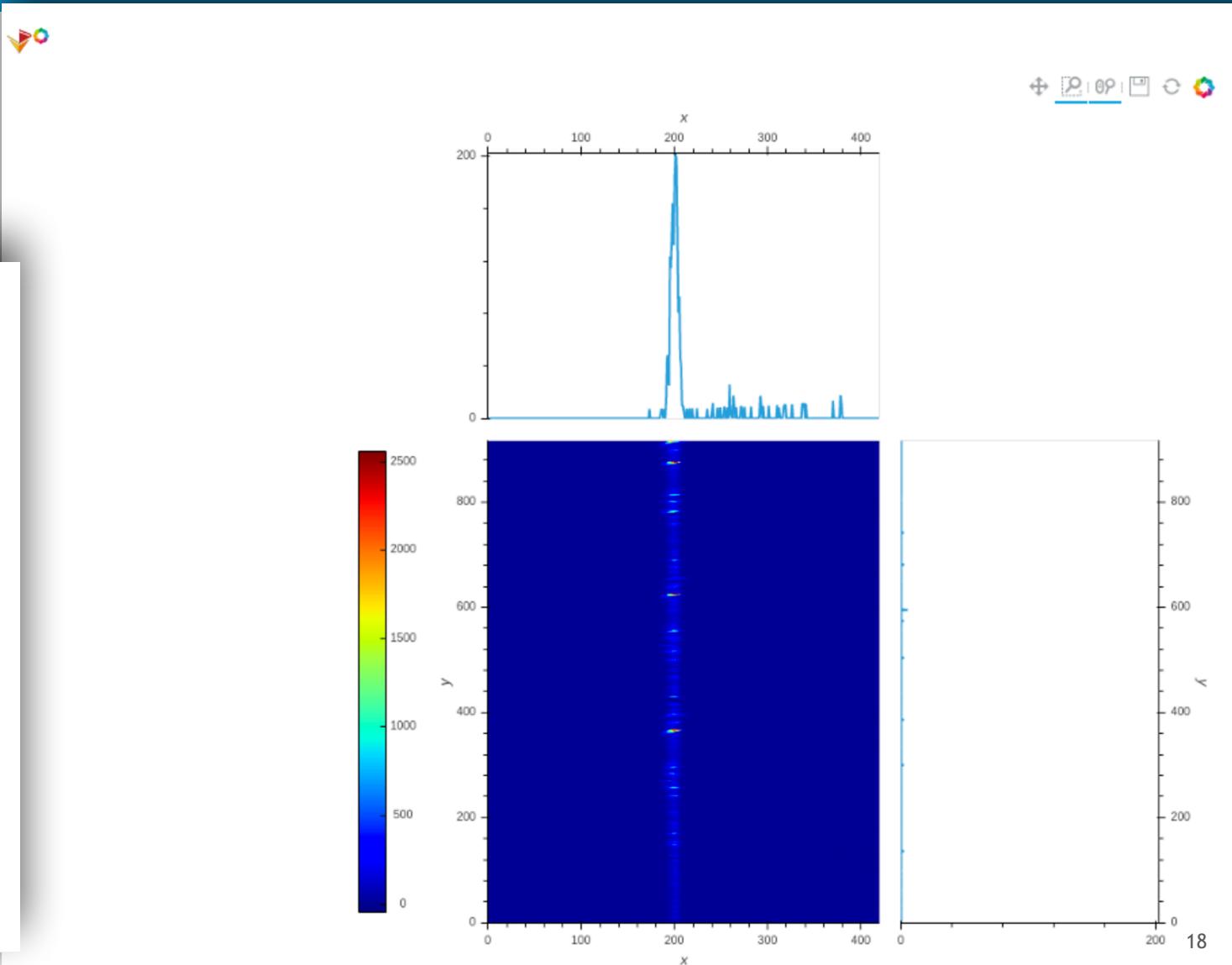
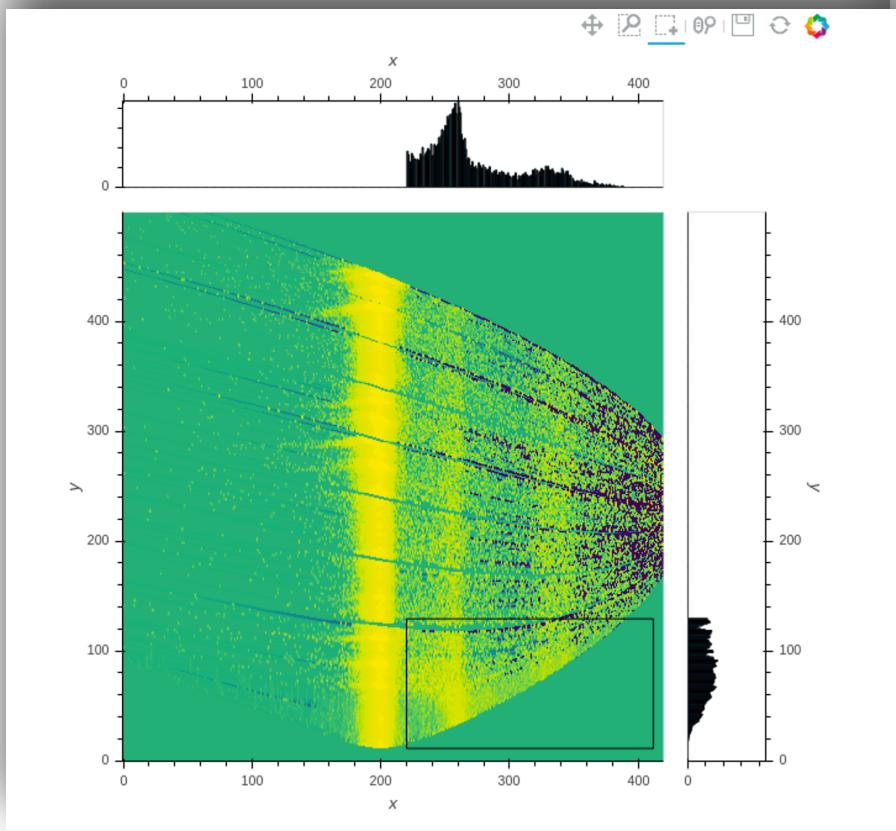
Interactive Data visualisation

- Data Selection
- Plot 1D spectrum
- Plot 2D Area



Holoviews multiplots

- Simple multiplot
- 2D Area + Mouse select for x, y
- 1 D integrations

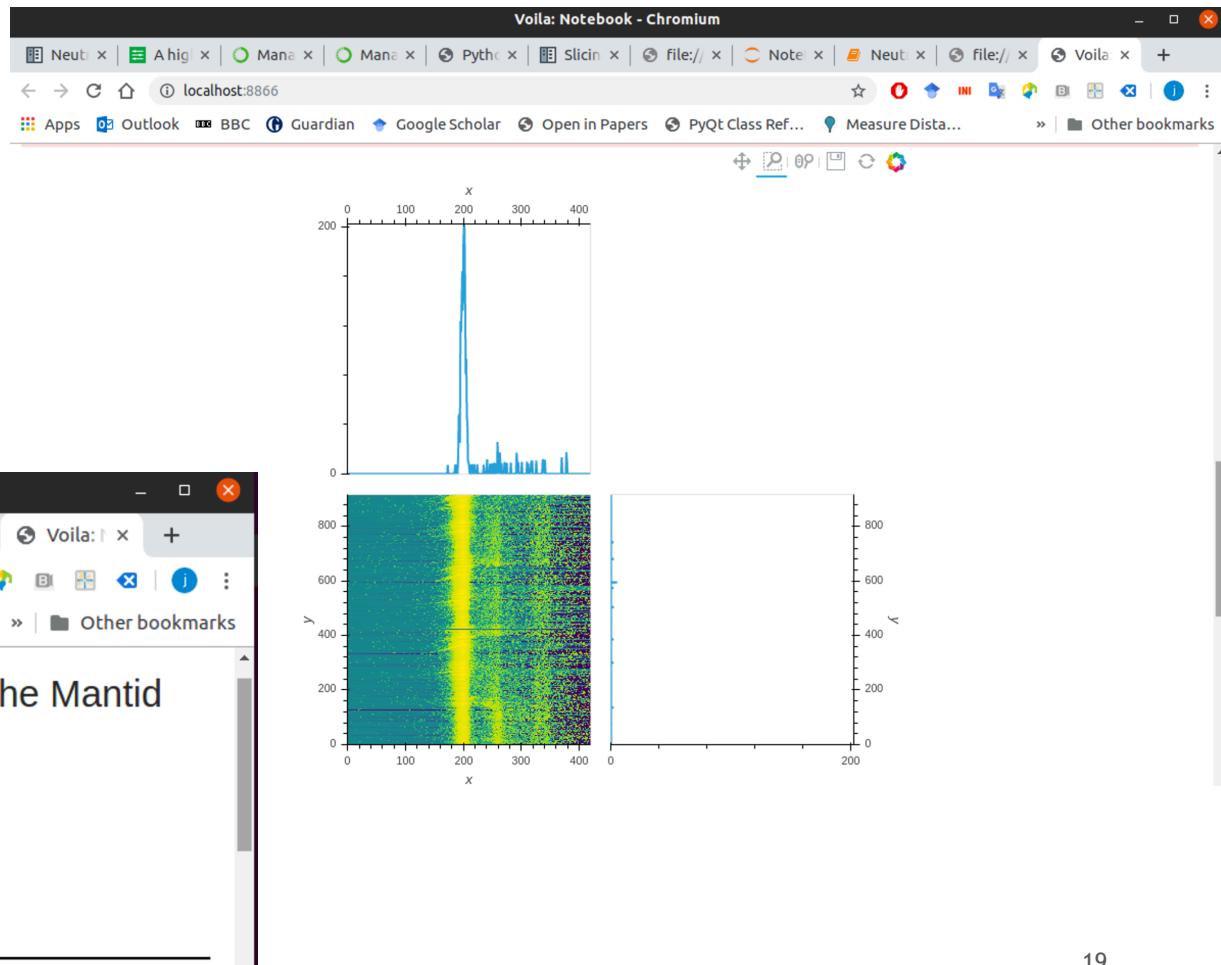


Tools deployed as interactive web apps

- Same notebook rendered as Web Page
- Deploys simple to use UIs

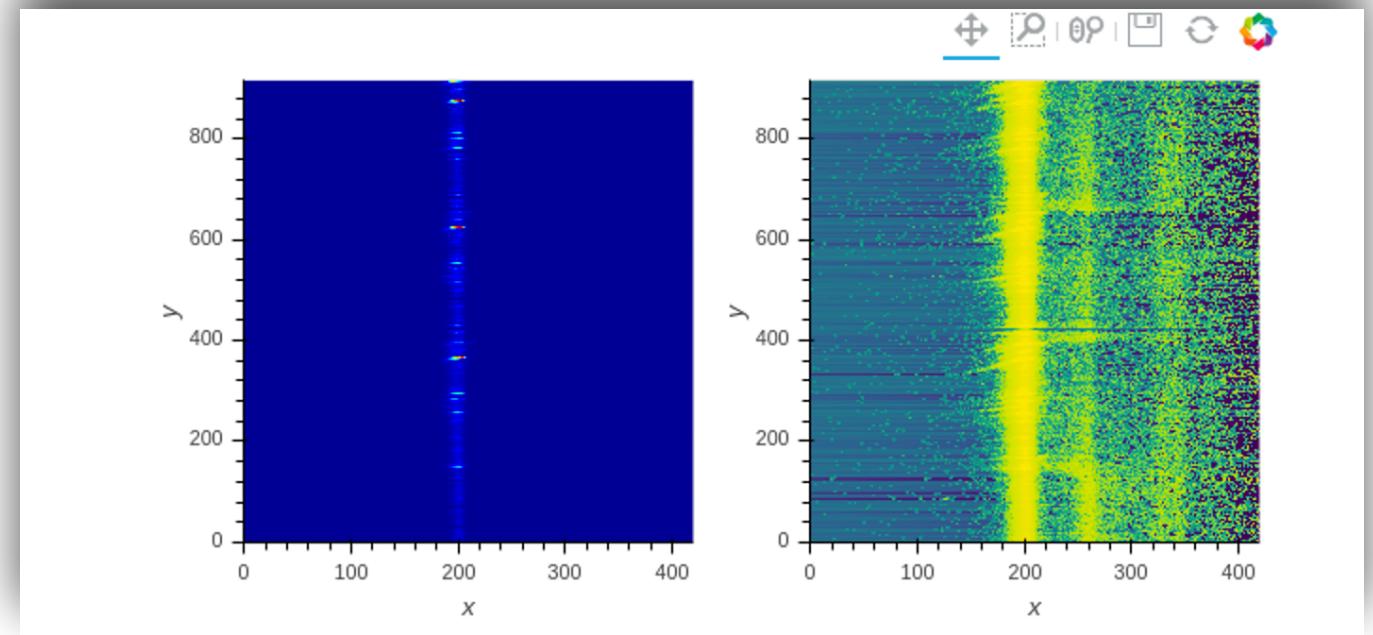


Voila turns Jupyter notebooks into standalone web applications.



Data Visualisation

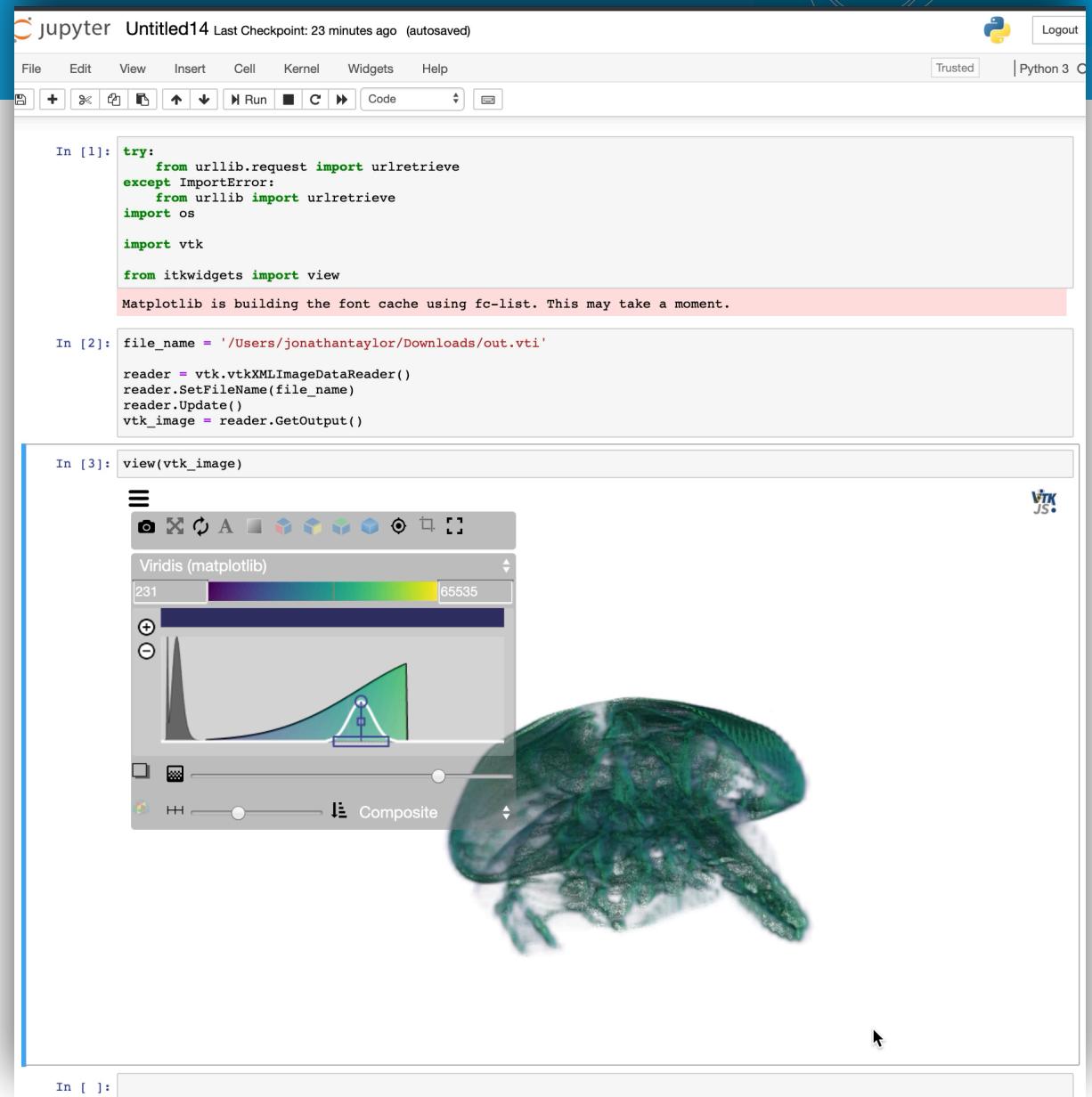
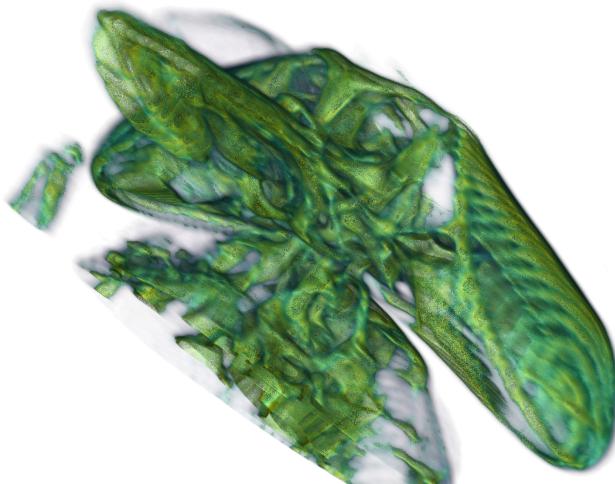
- So many really useful tools
- MPL
- Plotly
- HoloViews
- DataShader
- Users are not constrained to a single library



Inelastic Neutron scattering data visualised using Data Shader

Visualisation of Imaging data

- ITKWidgets & VTK
- Data From PSI imaging group
- Neutron CT of fruit fly
- Can be used for nD neutron data



```
jupyter Untitled14 Last Checkpoint: 23 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help
+ % Run C Code Trusted Python 3 C

In [1]: try:
           from urllib.request import urlretrieve
       except ImportError:
           from urllib import urlretrieve
       import os

       import vtk

       from itkwidgets import view

       Matplotlib is building the font cache using fc-list. This may take a moment.

In [2]: file_name = '/Users/jonathanjtaylor/Downloads/out.vti'

       reader = vtk.vtkXMLImageDataReader()
       reader.SetFileName(file_name)
       reader.Update()
       vtk_image = reader.GetOutput()

In [3]: view(vtk_image)
```

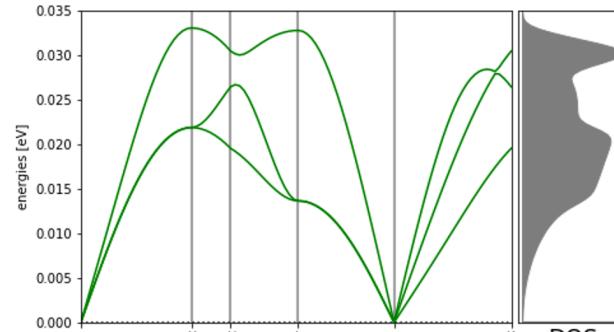
Jupyter for data analysis - example

- Most analysis codes are exposed through Python
- Example :
 - Atomic simulation environment ASE
 - Simple calculation of Density of states and phonon dispersion relations for Aluminium

In [26]:

```
# Plot the band structure and DOS:  
%matplotlib inline  
import matplotlib.pyplot as plt  
  
fig = plt.figure(1, figsize=(7, 4))  
ax = fig.add_axes([.12, .07, .67, .85])  
  
emax = 0.035  
bs.plot(ax=ax, show=False, emin=0.0, emax=emax)  
aa=bs.todict()  
aa.keys  
dosax = fig.add_axes([.8, .07, .17, .85])  
dosax.fill_between(dos.weights[0], dos.energy, y2=0, color='grey', edgecolor='k', lw=1)  
  
dosax.set_xlim(0, emax)  
dosax.set_yticks([])  
dosax.set_xticks([])  
dosax.set_xlabel("DOS", fontsize=18)
```

Out[26]: Text(0.5, 0, 'DOS')



Jupyter & SciCat

- Access the facility data catalogue.
- SciCat can be accessed directly from your jupyter notebook!
- You can search for metadata about different experiments at ESS

The screenshot shows a Jupyter Notebook interface with the title "Search SciCat". The notebook contains the following code:

```
In [91]: import requests
import json
import urllib

def search_scicat(text, max_number_results):
    fields = {'text': text}
    limit = {'limit': max_number_results, 'order': "creationTime:desc"}
    fields_encode = urllib.parse.quote(json.dumps(fields))
    limit_encode = urllib.parse.quote(json.dumps(limit))
    dataset_url = \
        "https://scicatapi.esss.dk/api/v3/Datasets/anonymousquery?fields=" + \
        fields_encode+"&limits="+limit_encode
    r=requests.get(dataset_url).json()
    print(len(r), "result found!")
    return r
```

Search for e.g. hexagonal boron nitride (hbn)

```
In [92]: r=search_scicat("hbn",1)
```

1 result found!

Show the sample description

```
In [93]: print(r[0]["scientificMetadata"].get("sample_description"))
```

hBN target with 1.0 mm diameter hole

Print the location of the data

```
In [90]: print(r[0]["sourceFolder"])
```

/nfs/groups/beamlines/v20/67JH32

Summary



- Jupyter will be used at ESS for deployment of a broad range of scientific computing
- Widgets / JupyterLab (not mentioned) provide a flexible UX
- Development of UI is fast when based on well written python compatible frameworks