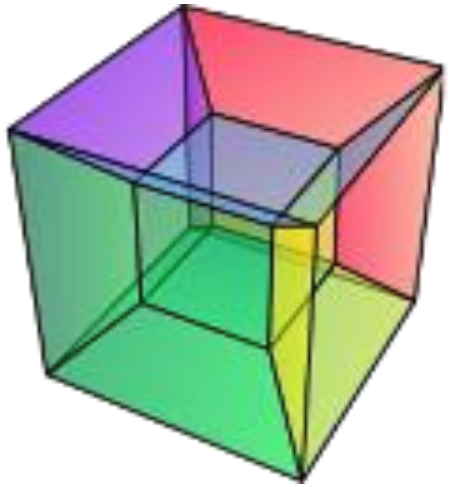
Fostering open and advanced data analysis in electron microscopy with HyperSpy

Eric Prestat[1,2]

[1]Department of Materials, University of Manchester, Manchester, UK

[2]SuperSTEM Laboratory, SciTech Daresbury Campus, Daresbury, UK



Outline

- Python and the scientific community
- What is HyperSpy?
- How HyperSpy is useful for data analysis in electron microscopy?
- What is coming next?

- Open source is not only about opening the source code, it is also about:
  - Accessibility to the users
  - Sustainable library development
  - Building a community

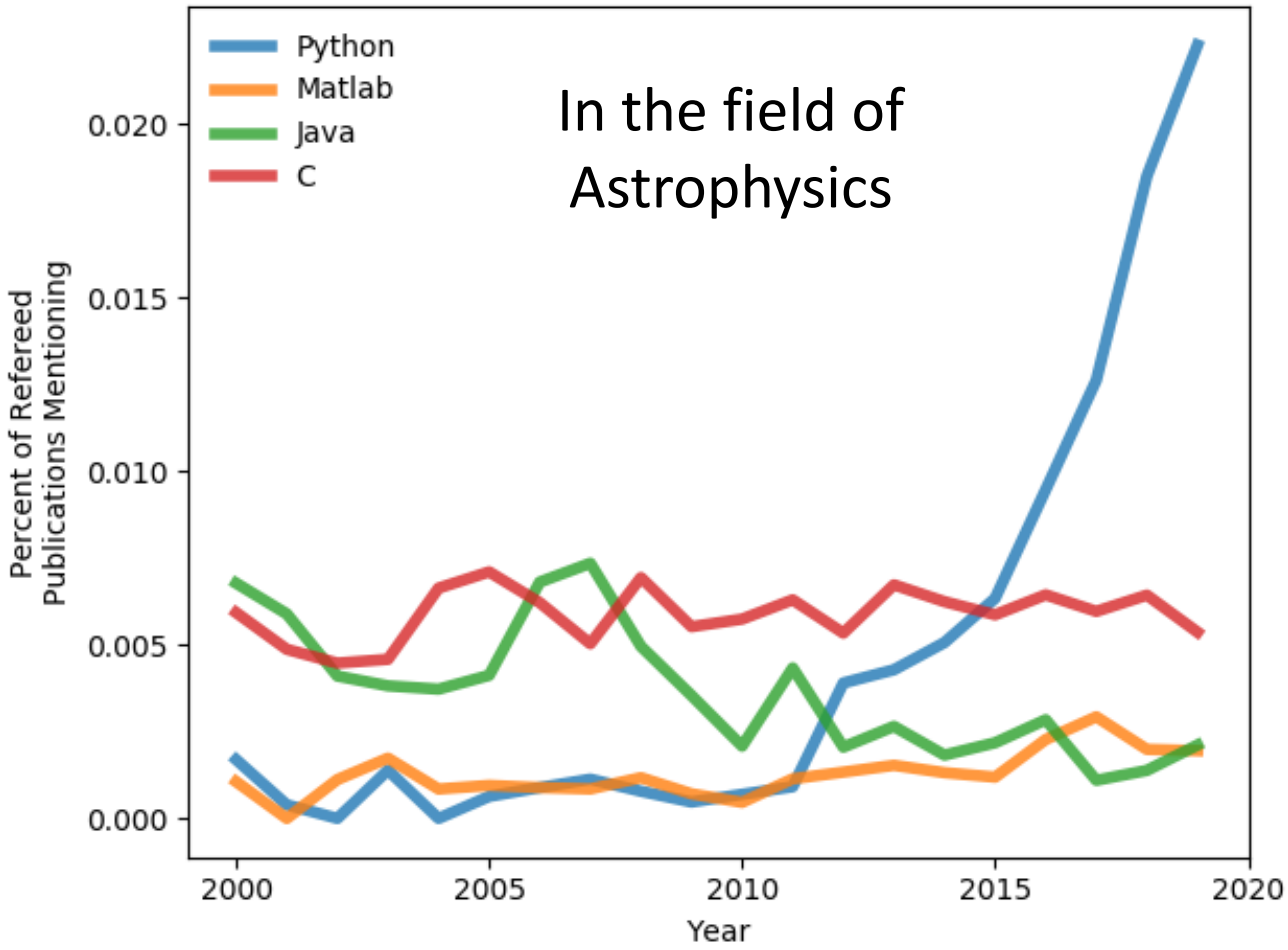
Why is Python so popular in the scientific community?

The University of Manchester

SuperSTEM DARESBURY

- Interoperability with other languages
- Simplicity and dynamic nature
- Syntax emphasizes readability and explicitness

- Consequences
  - Powerful scientific libraries
  - User becomes contributors



The Unexpected effectiveness of Python in Science, PyCon 2017
http://vanderplas.com/speaking.html

Adapted from https://gist.github.com/jni
https://gist.github.com/jni/3339985a016572f178d3c2f18e27ec0d
Data from https://ui.adsabs.harvard.edu

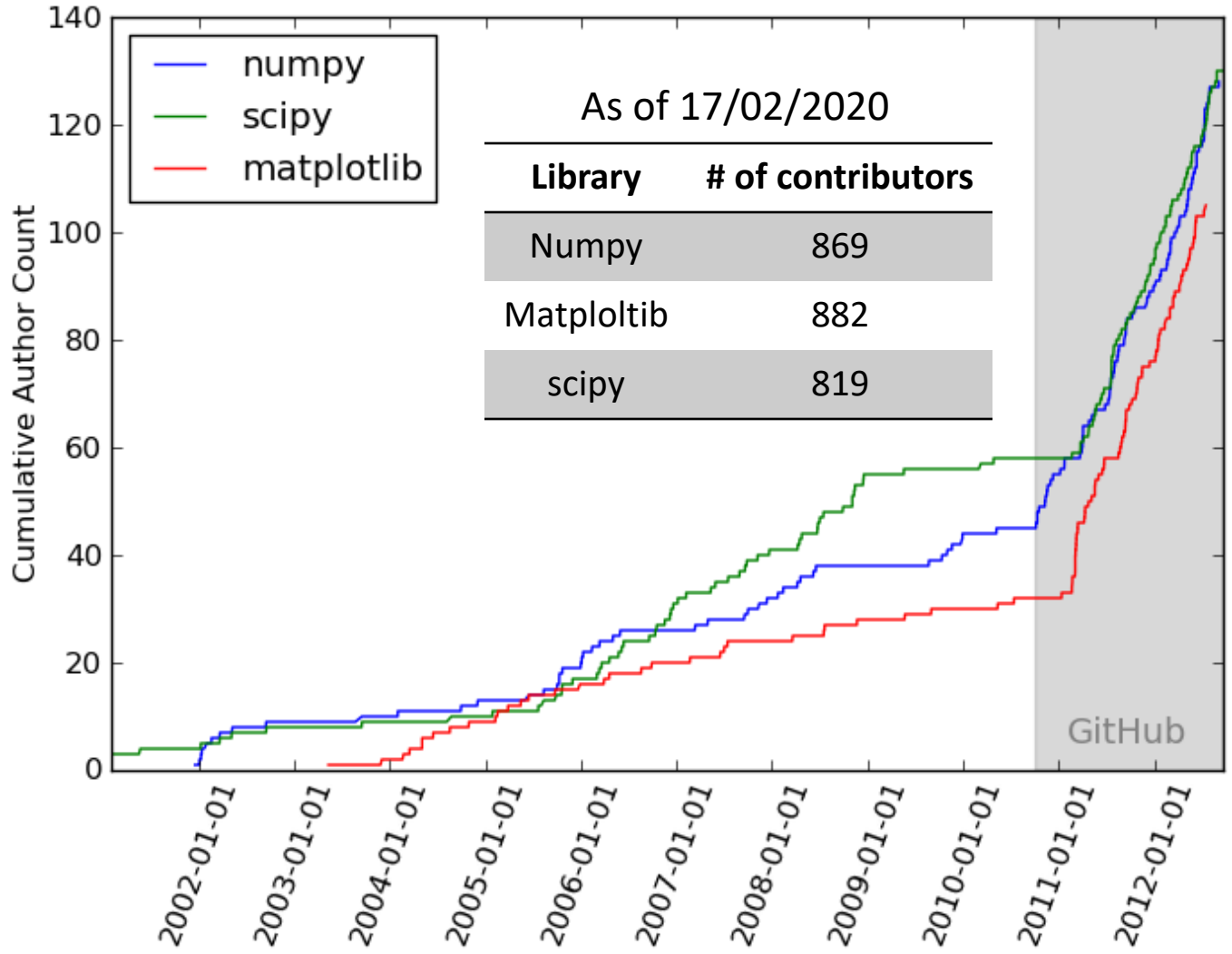
Standard and practices in the python community

The University of Manchester

SuperSTEM DARESBURY

- Open source development workflow
  - Continuous Integration
  - Review process
  - Consensus based
- High standard for documentation
- “Easy” packaging and distribution

1. Importance of the community
2. Sustainable development

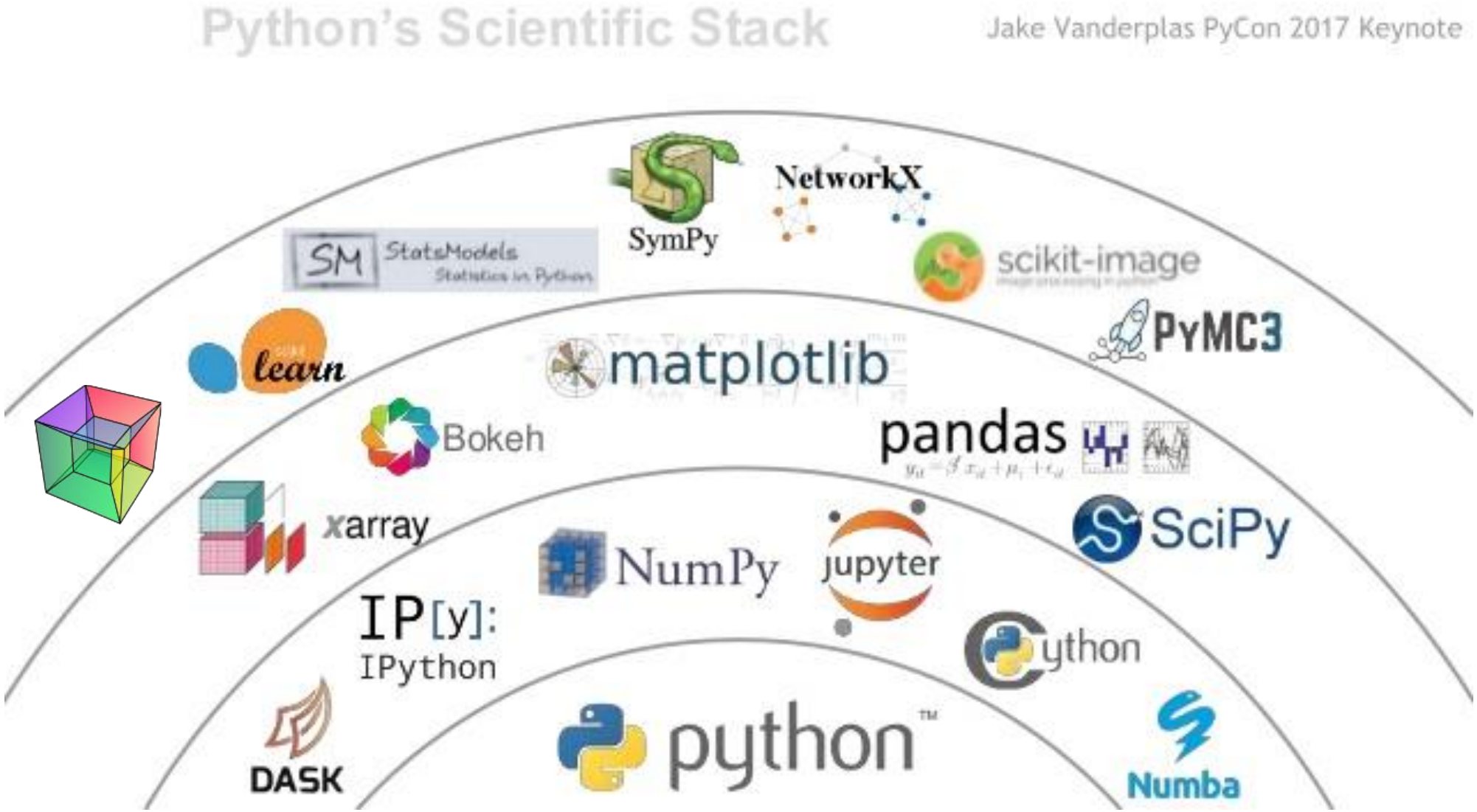


As of 17/02/2020

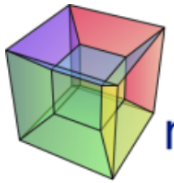
Library	# of contributors
Numpy	869
Matploltib	882
scipy	819

From Pythonic perambulations: Why Python is the last language you'll have to learn: https://jakevdp.github.io/blog/2012/09/20/why-python-is-the-last/

The python scientific stack



What is hyperspy?



HyperSpy
multi-dimensional data analysis

Home · Download · Documentation · Demos · News · Support · Citing · Credits

HyperSpy: multi-dimensional data analysis toolbox

HyperSpy is an open source Python library which provides tools to facilitate the interactive data analysis of multi-dimensional datasets that can be described as multi-dimensional arrays of a given signal (e.g. a 2D array of spectra a.k.a spectrum image).

HyperSpy aims at making it easy and natural to apply analytical procedures that operate on an individual signal to multi-dimensional arrays, as well as providing easy access to analytical tools that exploit the multi-dimensionality of the dataset.

Its modular structure makes it easy to add features to analyze different kinds of signals.

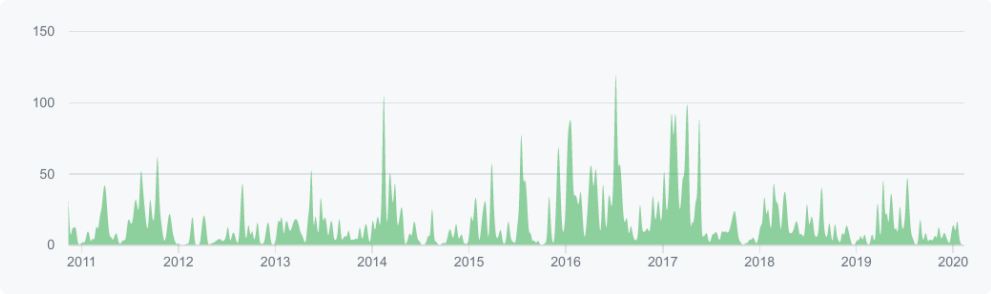
Highlights

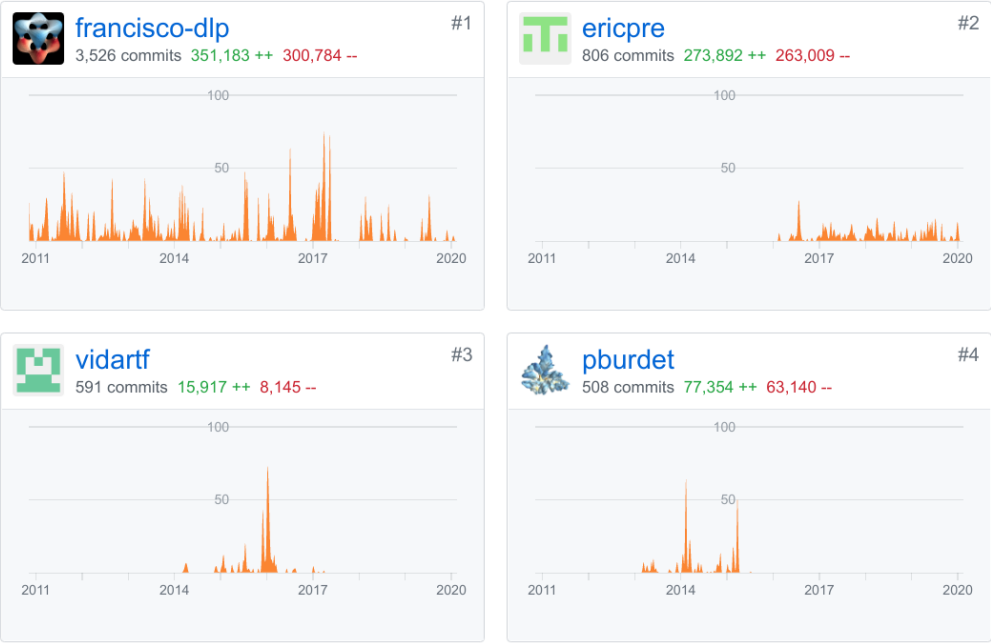
- Two families of named and scaled axes: *signal* and *navigation*.
- Visualization tools for multi-dimensional spectra and images.
- Easy access multi-dimensional curve fitting and blind source separation.
- Built on top of NumPy, SciPy, matplotlib and scikit-learn.
- Modular design for easy extensibility.

https://hyperspy.org

- Open source Python library
- Develop by electron microscopists
- Generic design
  - Multidimensional and hyperspectral dataset
- Powerful syntax

HyperSpy contributors and community





- Some contributors changed field/jobs
- Sustainable development

Francisco de la Peña[1,2,3,4], Eric Prestat[5,6], Vidar Tonaas Fauske[7], Pierre Burdet[1], Petras Jokubauskas[8] , Magnus Nord[7,9,10], Tomas Ostasevicius[1], Katherine E. MacArthur[11], Mike Sarahan[6], Joshua Taillon[12], Duncan N. Johnstone[1], Jonas Lähnemann[13], Alberto Eljarrat[14,15], Vadim Migunov[11], Thomas Aarholt[16,17], Jan Caron[11], Tom Furnival[1], Stefano Mazzucco[2], Suhas Somnath[18], Michael Walls[2], Thomas Slater[6,20], Paul Quinn[20], Florian Winkler[11], Gaël Donval[4,19], Robert McLeod[21] + ….

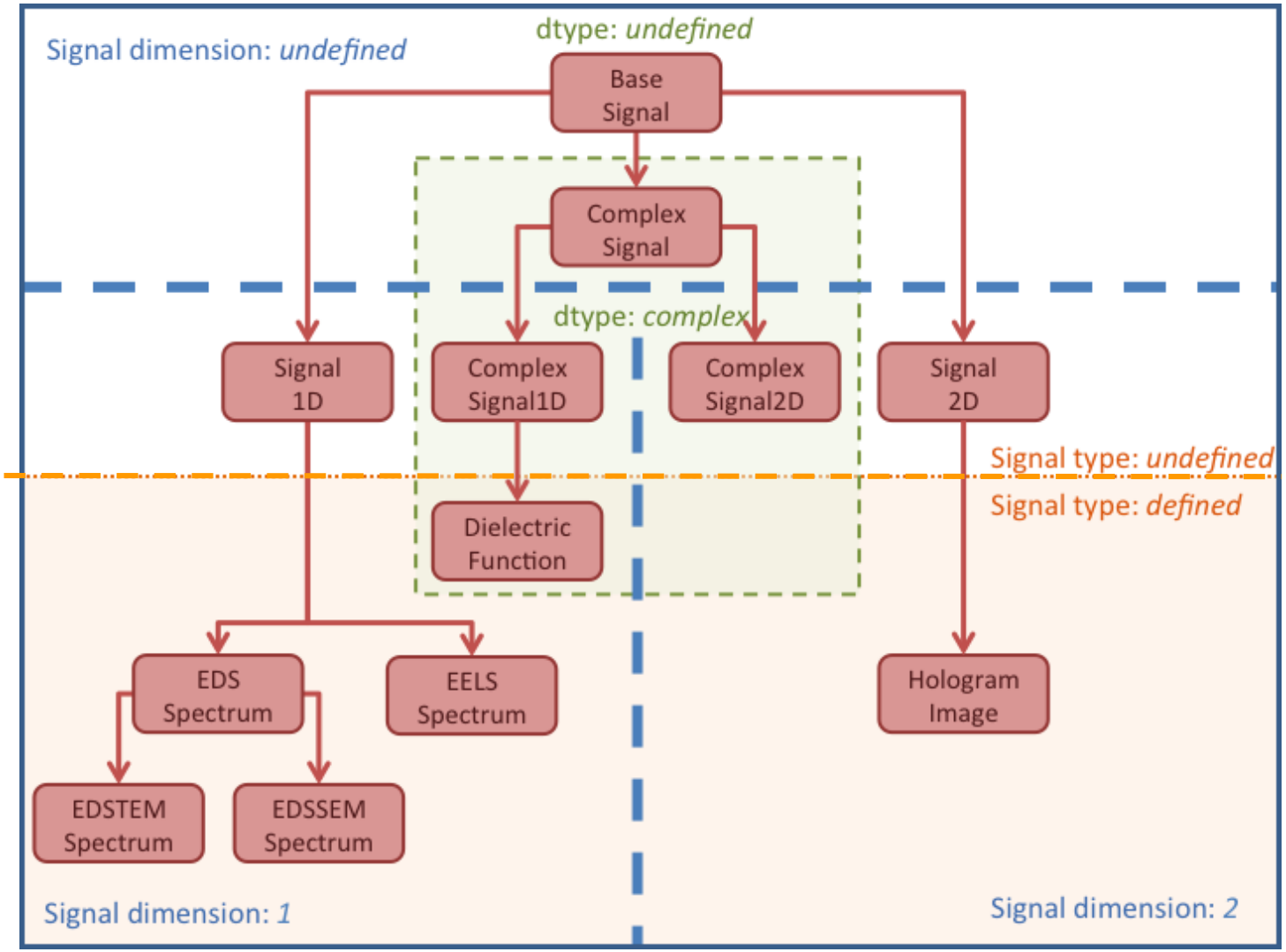
[1] Department of Materials Science and Metallurgy, University of Cambridge, Cambridge, U.K.
[2] Laboratoire de Physique des Solides, Paris-Sud University, Orsay, France
[3] Unité Matériaux et Transformations, University Lille1, Lille, France
[4] CEA Grenoble, Grenoble, France
[5] Department of Materials, The University of Manchester, Manchester, U.K.
[6] SuperSTEM, SciTech Daresbury Campus, Warrington, U.K.
[7] Department of Physics, NTNU, Trondheim, Norway
[8] Institute of Geochemistry, Mineralogy and Petrology, University of Warsaw, Poland
[9] School of Physics and Astronomy, University of Glasgow, Glasgow, U.K.
[10] EMAT, University of Antwerp, Antwerp, Belgium
[11] Forschungszentrum Jülich GmbH, Jülich, Germany
[12] Material Measurement Laboratory, NIST, Gaithersburg, MD, U.S.A.
[13] Paul-Drude-Institut, Hausvogteipl. 5-7, 10117 Berlin, Germany
[14] Laboratory of Electron NanoScopies, Universitat de Barcelona, Barcelona, Spain
[15] Department of Physics, Humboldt University of Berlin, Newtonstr. 15, 12489 Berlin, Germany
[16] Department of Materials, University of Oxford, Oxford, United Kingdom
[17] Department of Physics, University of Oslo, Oslo, Norway
[18] Oak Ridge National Laboratory, U.S.A.
[19] Institut des Matériaux Jean Rouxel (IMN), Université de Nantes, Nantes, France
[20] Diamond Light Source, Didcot, U.K.
[21] Entropy Reduction Algorithmics, Victoria, Canada

+ many more: in total 40 contributors (~10 is a one off)

https://github.com/hyperspy/hyperspy/graphs/contributors

HyperSpy core concepts

- Signal objects
- Navigation and signal space
  - Plotting
  - Mapping operation
- Generic signal type:
  - Signal1D, Signal2D
- Specialised signal type:
  - EDS, EELS, etc.

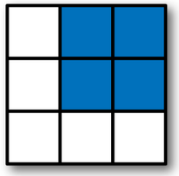


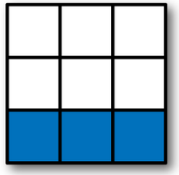
"*Spectrum*" type dataset

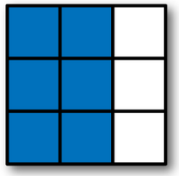
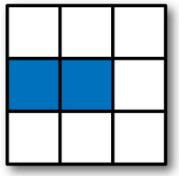
"*Image*" type dataset

HyperSpy main features

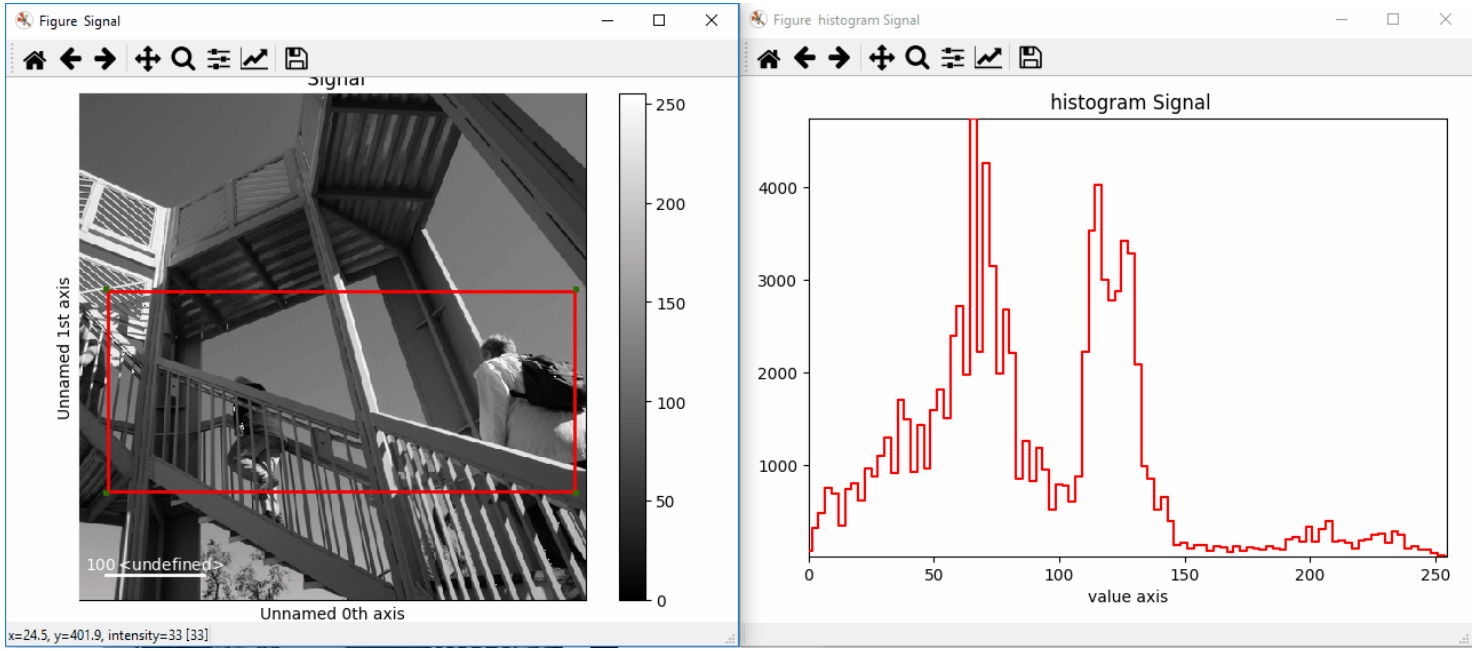
- Indexing and slicing
  - How to select the specific part of the dataset we are interesting in

	Expression	Shape
	`arr[:2, 1:]`	`(2, 2)`

	`arr[2]`
	`arr[2, :]`
	`arr[2:, :]`
		`(3,)`
		`(3,)`
		`(1, 3)`

	`arr[:, :2]`	`(3, 2)`
	`arr[1, :2]` `arr[1:2, :2]`	`(2,)` `(1, 2)`

- Interactive operation and ROIs



Indexing and slicing syntax

The University of Manchester

SuperSTEM DARESBURY

- Indexing and slicing
  - How to select the specific part of the dataset we are interesting in

In [1]:
```
%matplotlib qt
import hyperspy.api as hs
import numpy as np
```

In [2]:
```
# 20 x 30 x 40 x 1024 signal 1D
data = np.arange(20*30*40*1024).reshape((20, 30, 40, 1024))
s = hs.signals.Signal1D(data)
s.axes_manager[0].units = 'micrometer'
s.axes_manager[0].scale = 0.01
```

Indexing and slicing syntax

The University of Manchester

SuperSTEM DARESBURY

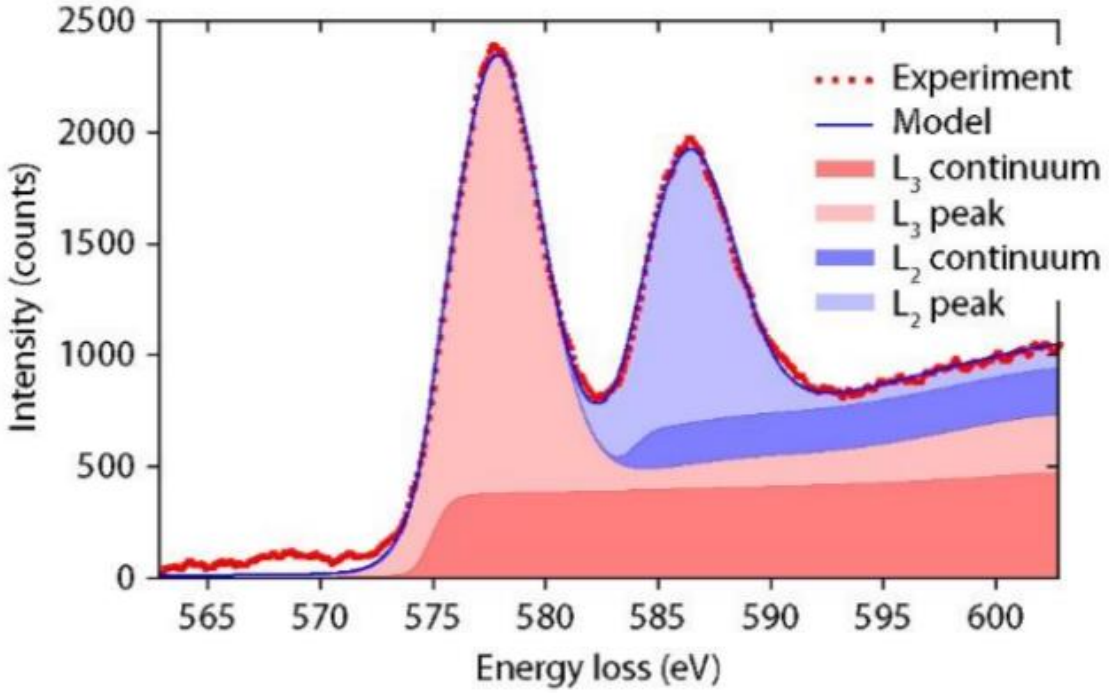
- Indexing and slicing
  - How to select the specific part of the dataset we are interesting in
- Follow Python syntax `[i1:i2]`
- Allow
  - Independent indexing of signal and navigation dimensions
  - Indexing in calibrated values (using decimal numbers)
  - Indexing with units (using string)

```
In [1]: %matplotlib qt
        import hyperspy.api as hs
        import numpy as np

In [2]: # 20 x 30 x 40 x 1024 signal 1D
        data = np.arange(20*30*40*1024).reshape((20, 30, 40, 1024))
        s = hs.signals.Signal1D(data)
        s.axes_manager[0].units = 'micrometer'
        s.axes_manager[0].scale = 0.01

In [3]: # slice in signal dimension from index 128 to 256
        s.isig[128:256]

Out[3]: <Signal1D, title: , dimensions: (40, 30, 20|128)>

In [4]: # slice in the first navigation dimension from index 0 to 5
        s.inav[:5]

Out[4]: <Signal1D, title: , dimensions: (5, 30, 20|1024)>

In [5]: # slice in the first navigation dimension using
        # calibration values from 0.1 to 0.3
        s.inav[0.1:0.3]

Out[5]: <Signal1D, title: , dimensions: (20, 30, 20|1024)>

In [6]: # Slice in the first navigation dimension using
        # different units
        s.inav[:"125 nm"]

Out[6]: <Signal1D, title: , dimensions: (12, 30, 20|1024)>
```

HyperSpy main features

- Basic machine learning
  - Principal component analysis (PCA)
  - Independent component Analysis (ICA)
  - Non-negative matrix factorization (NMF)
- Model fitting
- Big data processing
  - Data larger than computer memory

Fine structure fitting
of Cr-L edge



Model consisting of two Gaussian and two arctangent functions and convolved with the low-loss to model plural scattering.

Kravets *et al.,* Sci. Rep. 7:2878 (2017)
https://dx.doi.org/10.1038/s41598-017-02976-7

Syntax

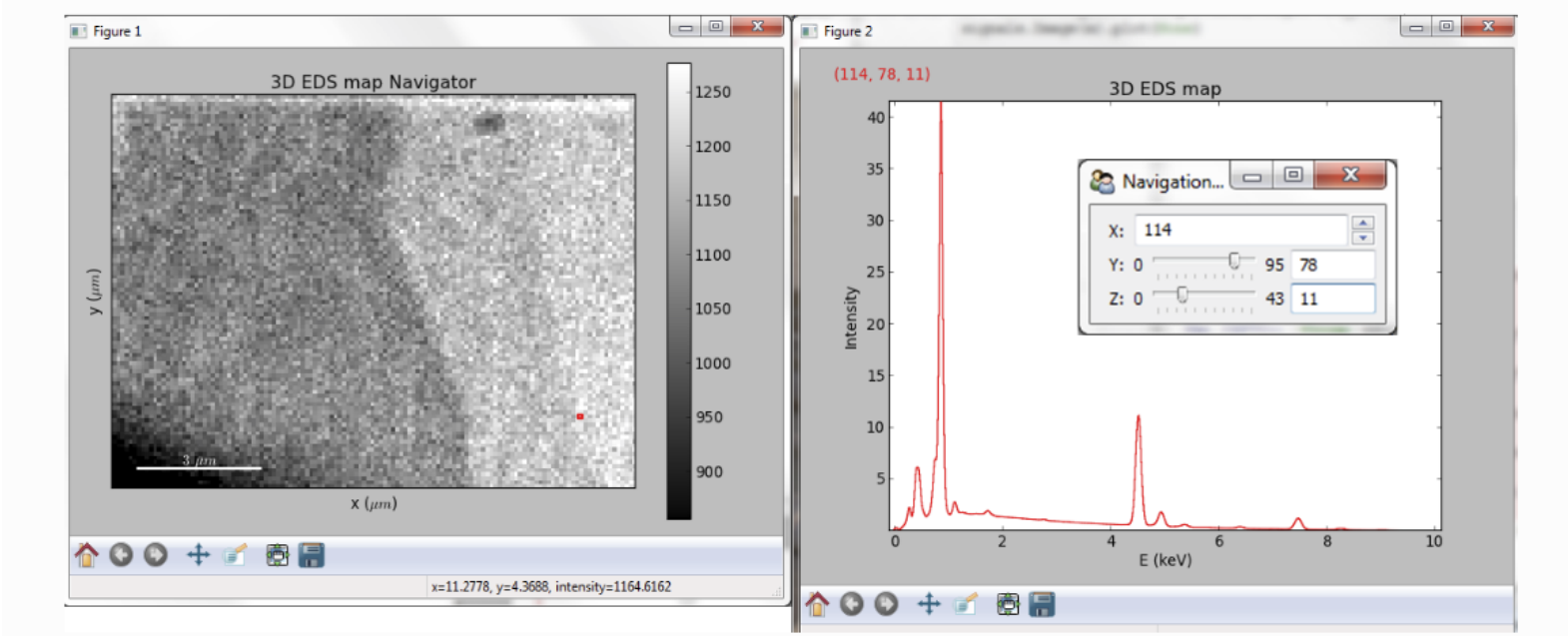
- Example to load a stack of EDS spectrum image (FIB slice and view)

```
In [1]: %matplotlib qt
        import hyperspy.api as hs

In [2]: s = hs.load("Ni_superalloy_0*.rpl", stack=True).as_signal1D(0)
```

Syntax

- Example to load a stack of EDS spectrum image (FIB slice and view)

```
In [1]: %matplotlib qt
        import hyperspy.api as hs

In [2]: s = hs.load("Ni_superalloy_0*.rpl", stack=True).as_signal1D(0)

In [3]: s.plot()
```

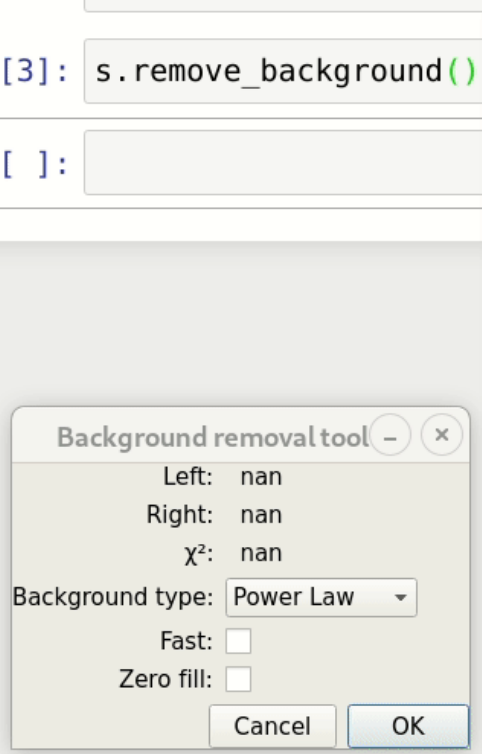


Visualisation of a 3D spectrum image with sliders.

Interactive tools

- Remove background of an EELS

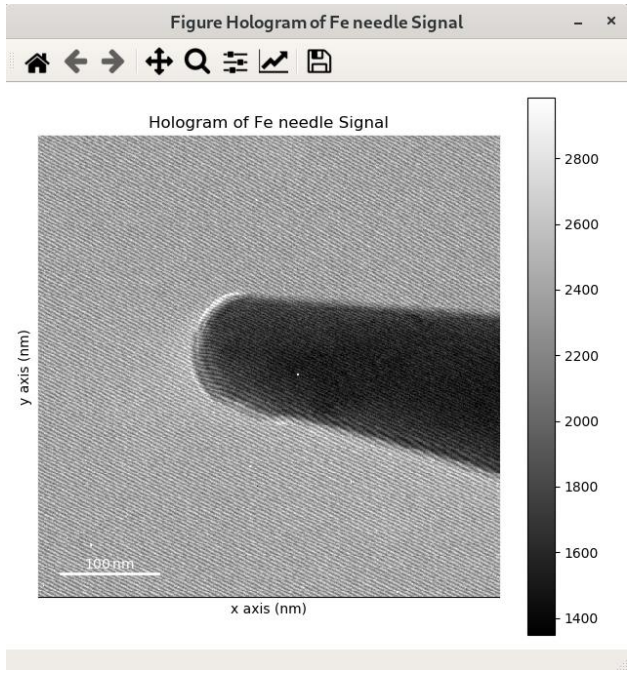
```
In [1]: %matplotlib qt
        import hyperspy.api as hs

In [2]: s = hs.datasets.artificial_data.get_core_loss_eels_signal(True)

In [3]: s.remove_background()

In [ ]:
```



Making processing easy but ….

- Raising awareness about what it actually does
- Example of FFT calculation:
  - Most users will not be aware of the difference between FFT and power spectrum

Hologram of Fe needle

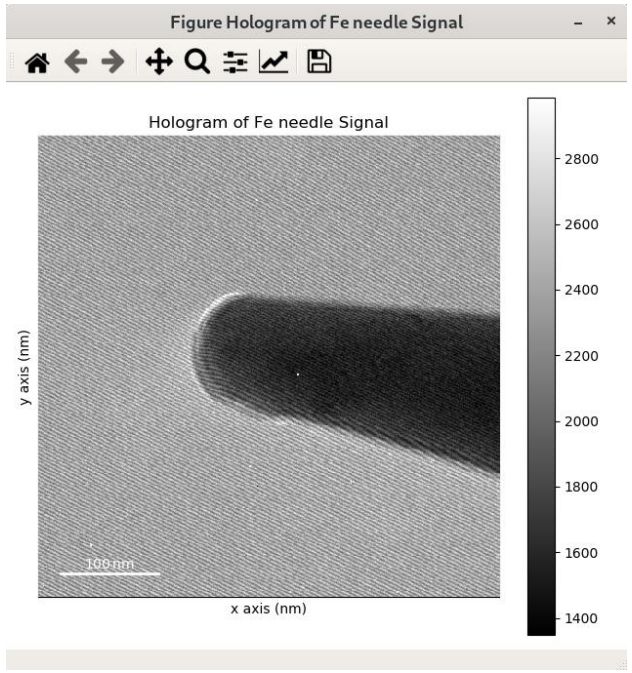



```
In [1]: %matplotlib qt
        import hyperspy.api as hs

In [2]: ima = hs.datasets.example_signals.object_hologram()

In [3]: fft = ima.fft()
```

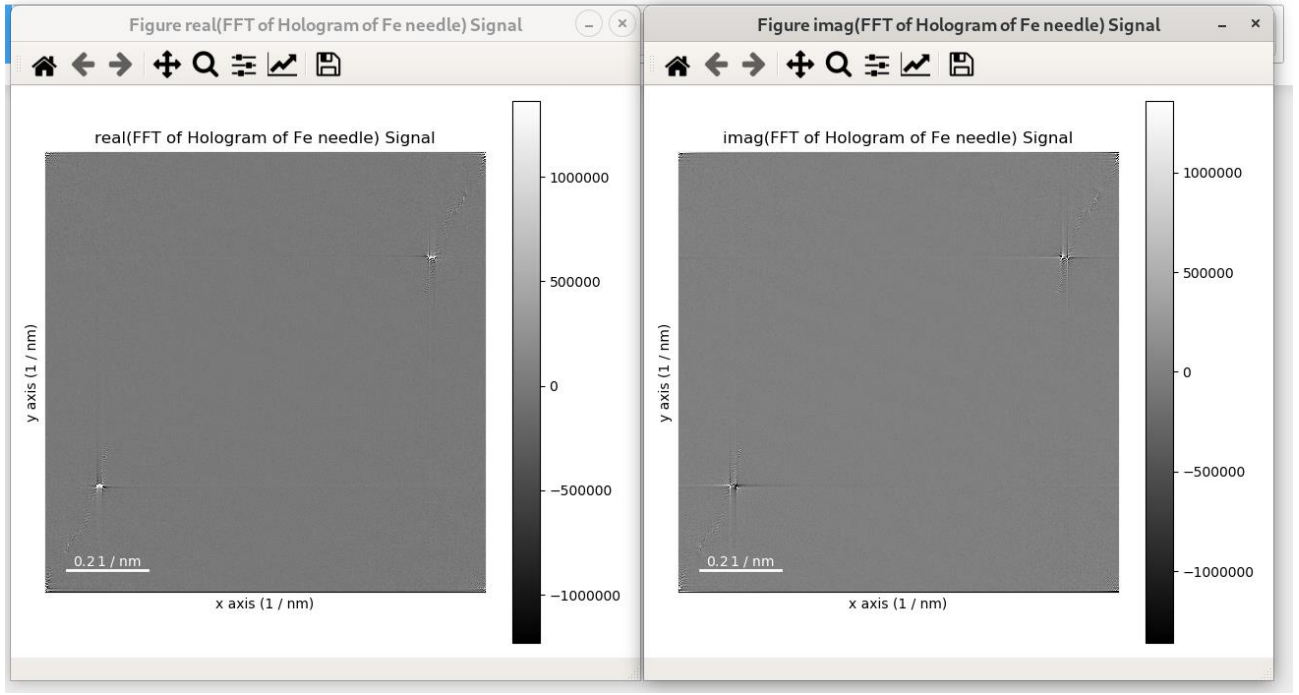
Making processing easy but ….

- Raising awareness about what it actually does
- Example of FFT calculation:
  - Most users will not be aware of the difference between FFT and power spectrum

Hologram of Fe needle



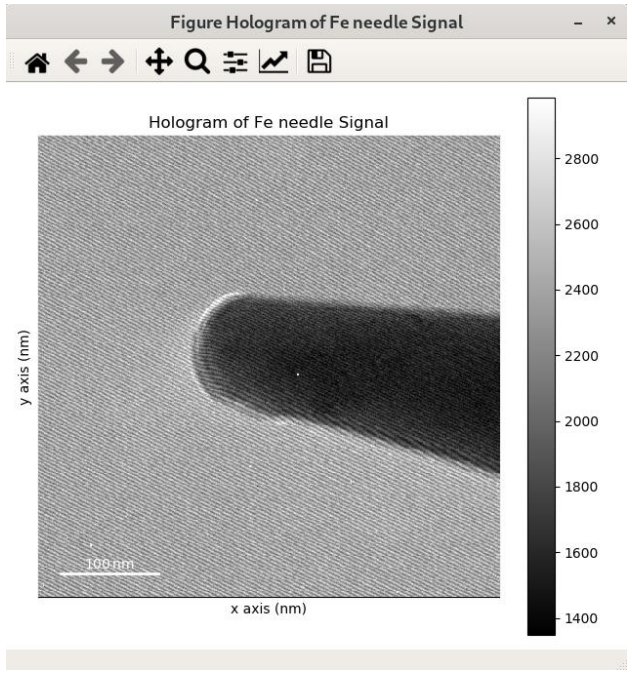

```
In [1]: %matplotlib qt
        import hyperspy.api as hs

In [2]: ima = hs.datasets.example_signals.object_hologram()

In [3]: fft = ima.fft()

In [4]: fft.plot()
```



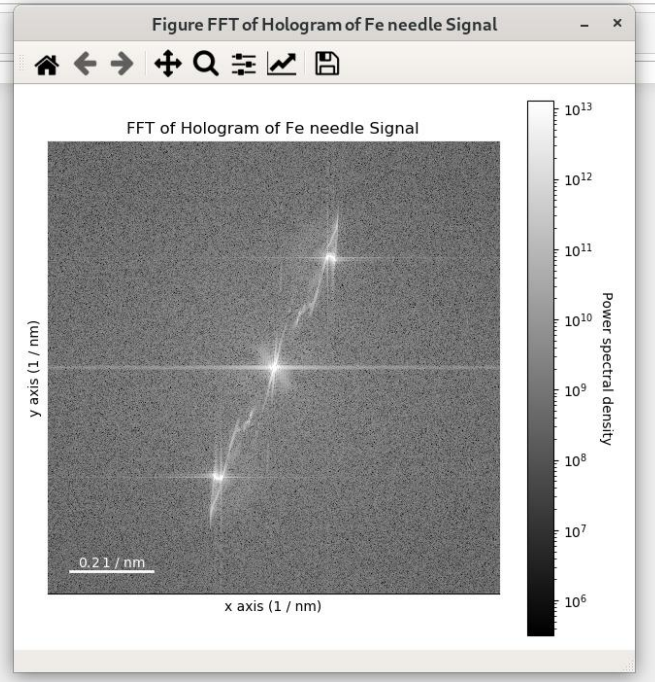
Making processing easy but ….

- Raising awareness about what it actually does
- Example of FFT calculation:
  - Most users will not be aware of the difference between FFT and power spectrum

Hologram of Fe needle




```
In [1]: %matplotlib qt
        import hyperspy.api as hs

In [2]: ima = hs.datasets.example_signals.object_hologram()

In [5]: fft = ima.fft(shift=True)

In [6]: fft.plot(power_spectrum=True)

In [ ]:
```

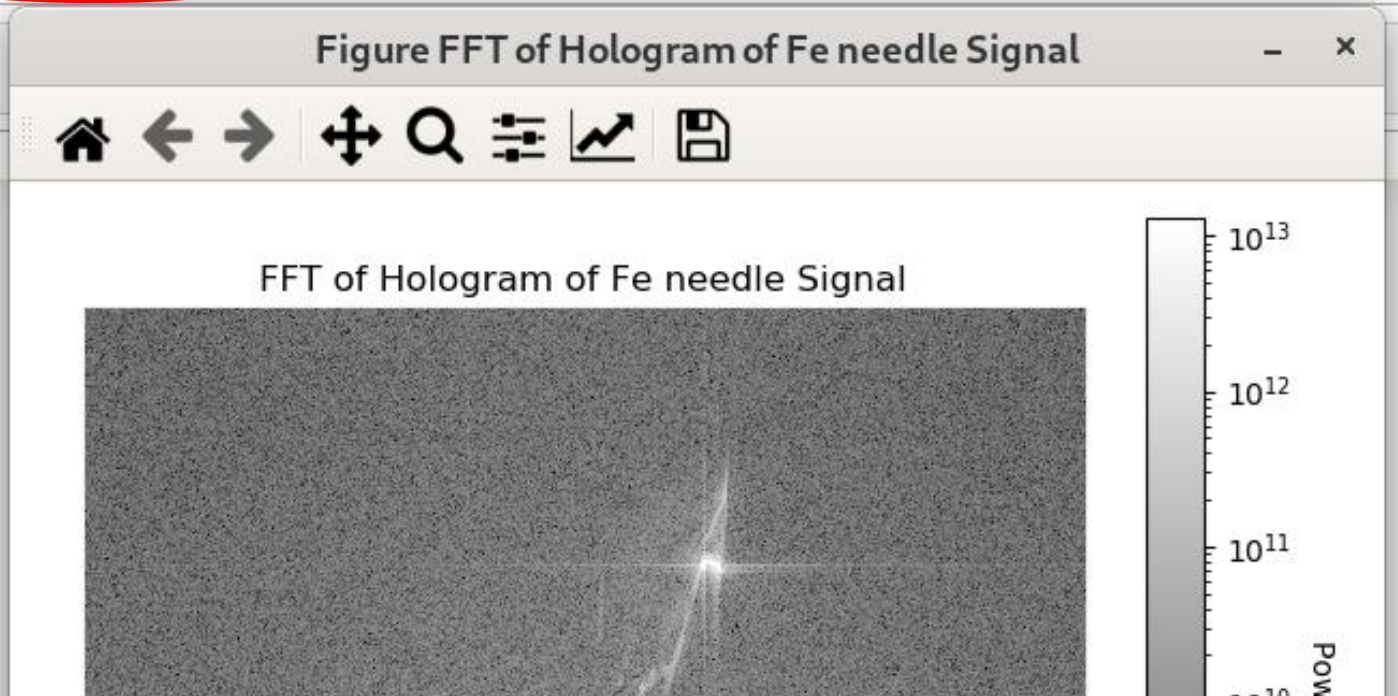


Making processing easy but ….

The University of Manchester

SuperSTEM DARESBURY

```
In [1]: %matplotlib qt
        import hyperspy.api as hs

In [2]: ima = hs.datasets.example_signals.object_hologram()

In [5]: fft = ima.fft(shift=True)

In [6]: fft.plot(power_spectrum=True)

In [ ]:
```



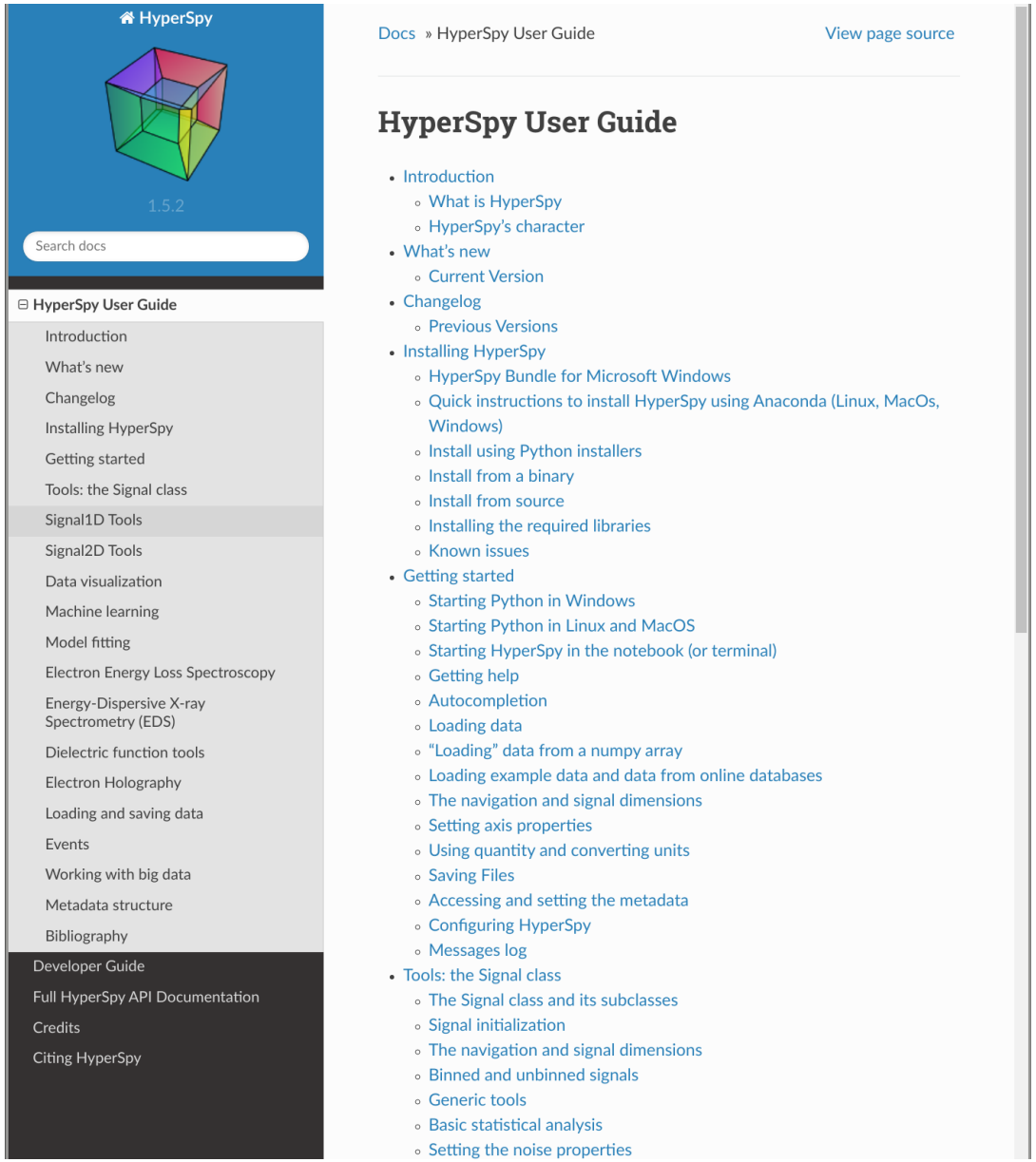
Documentation

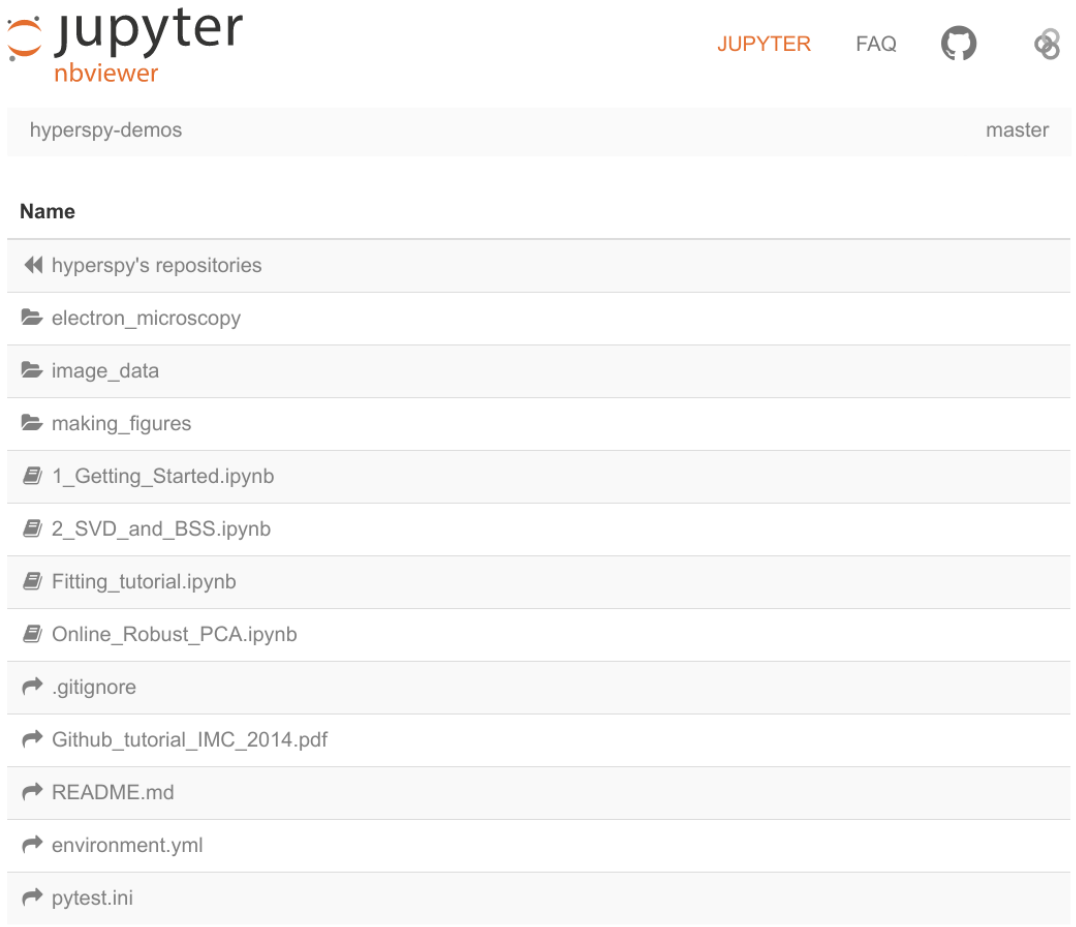
The University of Manchester

SuperSTEM DARESBURY

- PDF documentation
  - 180 pages of User Guide
  - 348 pages of “docstring”: documentation of functions, etc.

Documentation and notebook demos





http://hyperspy.org/hyperspy-doc/current/index.html

Input/Output capabilities

The University of Manchester

SuperSTEM DARESBURY

- Input/Output support for many format used in electron microscopy
- Read data *lazily* for data larger than computer memory

Format	Read	Write	lazy
Gatan's dm3	Yes	No	Yes
Gatan's dm4	Yes	No	Yes
FEI's emi and ser	Yes	No	Yes
HDF5	Yes	Yes	Yes
Image: jpg	Yes	Yes	Yes
TIFF	Yes	Yes	Yes
MRC	Yes	No	Yes
MRCZ	Yes	Yes	Yes
EMSA/MSA	Yes	Yes	No
NetCDF	Yes	No	No

Format	Read	Write	lazy
Ripple	Yes	Yes	Yes
SEMPER unf	Yes	Yes	Yes
Blockfile	Yes	Yes	Yes
DENS heater log	Yes	No	No
Bruker's bcf	Yes	No	Yes
Bruker's spx	Yes	No	No
EMD (NCEM)	Yes	Yes	Yes
EMD (FEI)	Yes	No	Yes
Protochips log	Yes	No	No
EDAX .spc and .spd	Yes	No	Yes

The HyperSpy eco-system today

The University of Manchester

SuperSTEM DARESBURY

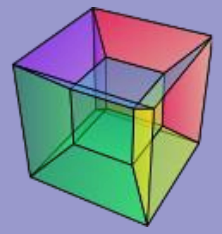
- Extension registration: other libraries can create their specific Signal

HyperSpy-ipywidget

HyperSpy-traitsui

HyperSpyUI

EELS holography Big data EDS Multi-dimensional IO Visualisation and interactivity Machine learning Curve-fitting



Atomic position fitting Quantitative STEM **Atomap** Strain mapping

4D STEM Magnetic measurement **pixSTEM** Differential contrast imaging

cathodoluminescence **LumiSpy** photoluminescence

EBSD **Kikuchipy** Strain mapping Orientation mapping

Strain mapping Orientation mapping **pyXem** Electron cristallography Scanning Electron diffraction

inpainting **inpystem** reconstruction techniques

segmentation **ParticleSpy** Particles analysis

Electron Correlation Microscopy **EMpyer** Angular Correlations Fluctuation Electron Microscopy

Python libraries for 4D STEM data analysis

The University of Manchester

SuperSTEM DARESBURY

- pyXem
- pixstem
- EMpyer
- py4DSTEM
- pycroscopy
- stemtools
- Various nionswift plugins
- …
- LiberTEM

All these open source libraries are doing very similarly things

Optimised for small computer clusters and performance

Coming soon: HyperSpy 2.0

The University of Manchester

SuperSTEM DARESBURY

- Split specialised signals
  - EDS, EELS, holography analysis move to separate packages
- HyperSpy to keep
  - Indexing, slicing, ROIs
  - Model fitting
  - Machine learning
  - Visualisation and interactivity tools
- Will enable independent development of field specific libraries
  - Hope this would encourage collaboration within the community
- Will be mostly transparent to the end users
  - Extension registration mechanism already in place

Coming soon: HyperSpy 2.0

- Split Input/Output module into a separate package
  - Will become independent from HyperSpy
  - Easier for the community to contribute
  - Easier to share with other libraries
  - Easier to maintain
  - Merge with other file reader/writer of other libraries
- Most likely change license to be permissive
  - Company can use it as part of their product

https://github.com/hyperspy/hyperspy/issues/1978
https://github.com/hyperspy/hyperspy/pull/2174

Summary

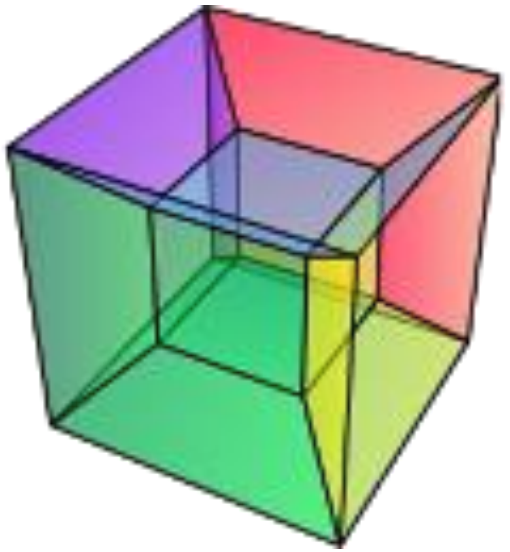
- Python and its scientific stack provide a great environment for data analysis
- HyperSpy aims at providing a framework for multidimensional and hyperspectral dataset
  - Powerful indexing and slicing
  - Interactivity and visualisation tools
  - Navigation | signal spaces
  - Model fitting
  - Basic machine learning
- HyperSpy extensions
  - Benefit from HyperSpy main capabilities
  - Focus on field specific data analysis

Base for multidimensional analysis

The importance of sustainability and the community

To learn more about HyperSpy

- Workshop organised regularly
  - Workshop at Diamond (UK) 2nd-3rd March 2020
  - Sunday Short Course at M&M 2020
  - Pre-congress workshop at EMC 2020
- Materials available online
  - https://github.com/hyperspy/hyperspy-demos
  - www.superstem.org/acmm2020hyperspy
- Various workshop run locally by users



Thank you for attention