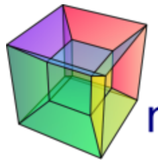
Welcome to the 2021 M&M Short Course X-15

Data Analysis in Materials Science with



HyperSpy
multi-dimensional data analysis

Presented by (a selection of) the HyperSpy developers:

- Håkon Wiik Ånes
- Carter Francis
- Eric Prestat
- Joshua Taillon

NIST Disclaimer

Certain commercial equipment, instruments, materials, vendors, and software are identified in this talk for example purposes and to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

Some logistics before we get started...

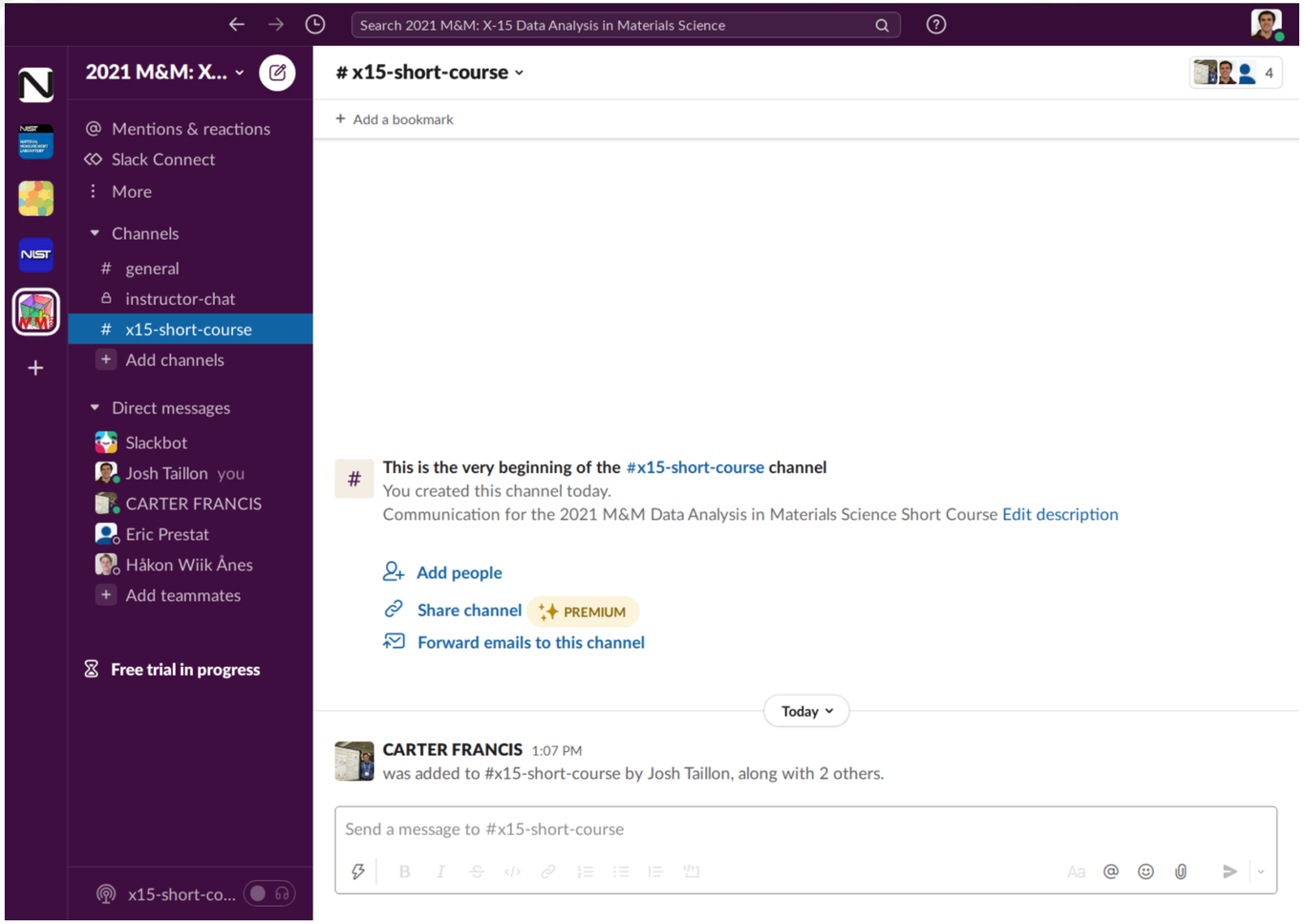
- Apologies in advance for difficulties due to the virtual platform, but we'll do our best to make sure you have a good experience!
- Each session will be led by one instructor, but throughout the day, the other instructors will be available via Slack to answer questions, facilitate discussion, etc.
- You should have received an invitation to the Slack workspace we'll be using; please type in the chat if you are not able to access it and we will try to get you set-up correctly

- Unlike an in-person session, we will not be able to help debug an installation during the course
- If you're having trouble following along on your local computer, please use the "Binder" we have provided at the link:
    - https://mybinder.org/v2/gh/usnistgov/hyperspy_tutorial/2021-08-01_MandM_short_course

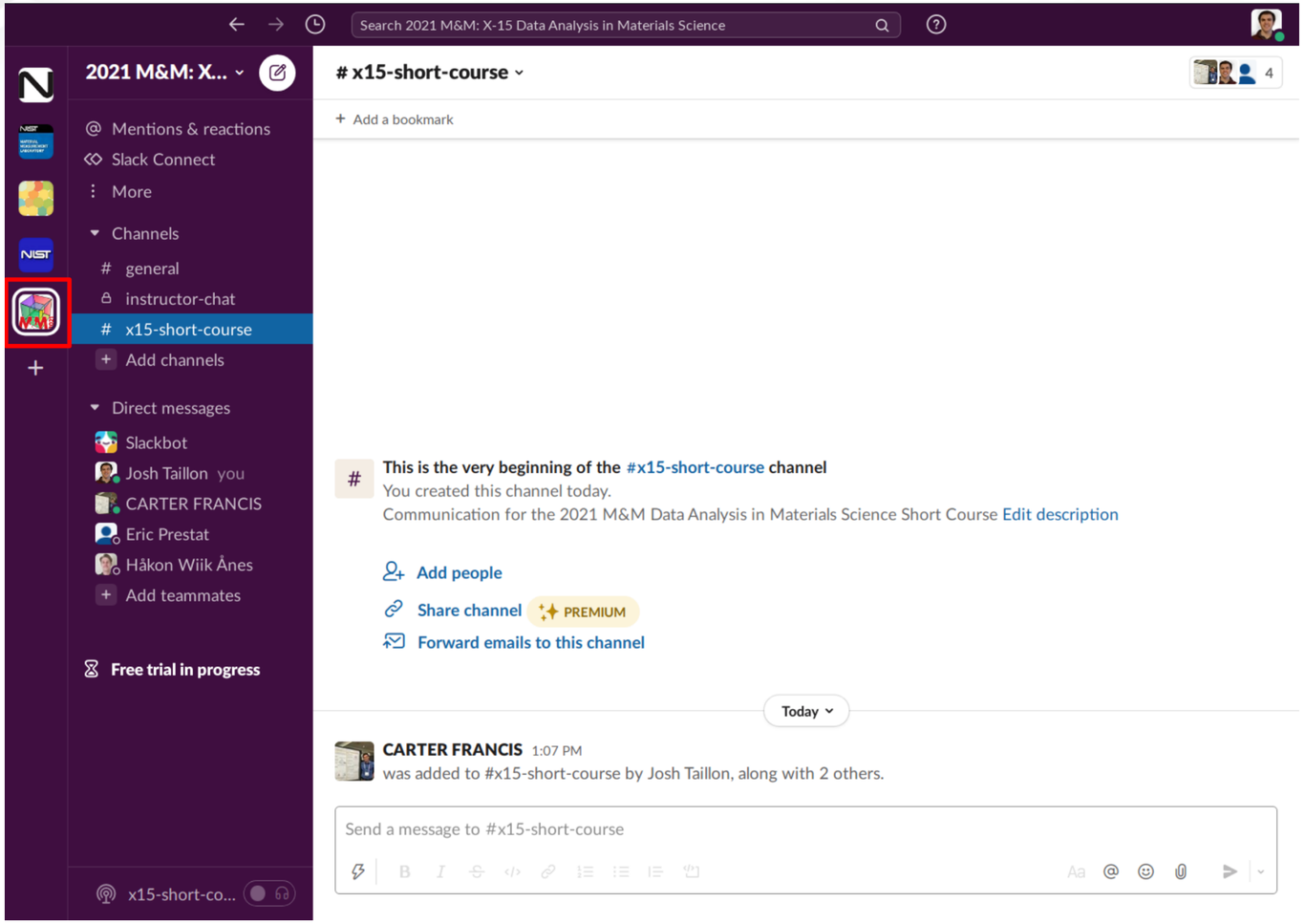
Helpful Resources:

- Agenda and course materials: https://pages.nist.gov/hyperspy_tutorial
    - Also contains helpful links to Binder, resources, installation help, etc.
- Slack download: https://slack.com/downloads/
- Repository of notebook and data files (to download manually): https://github.com/usnistgov/hyperspy_tutorial

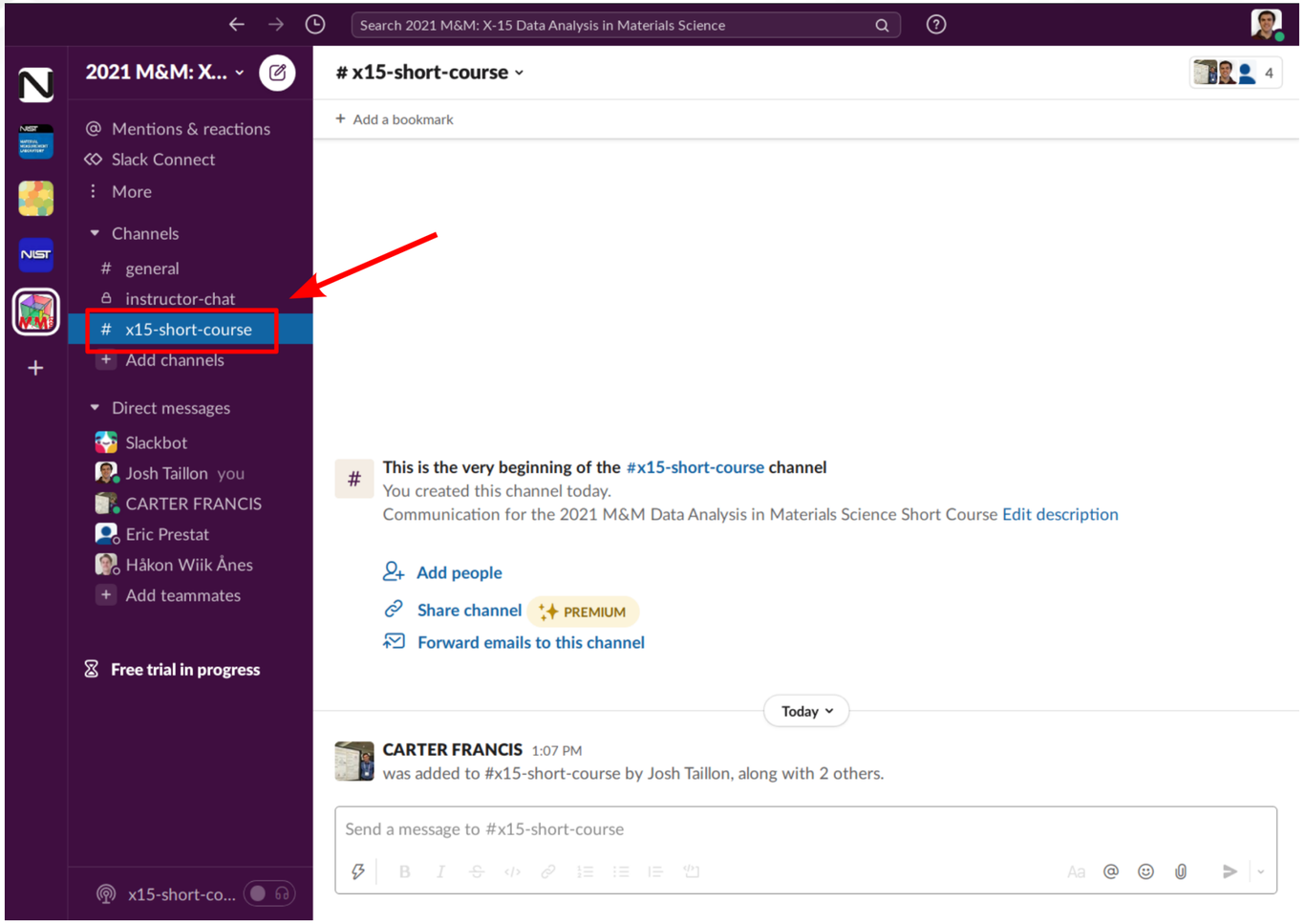
Using Slack — General Interface



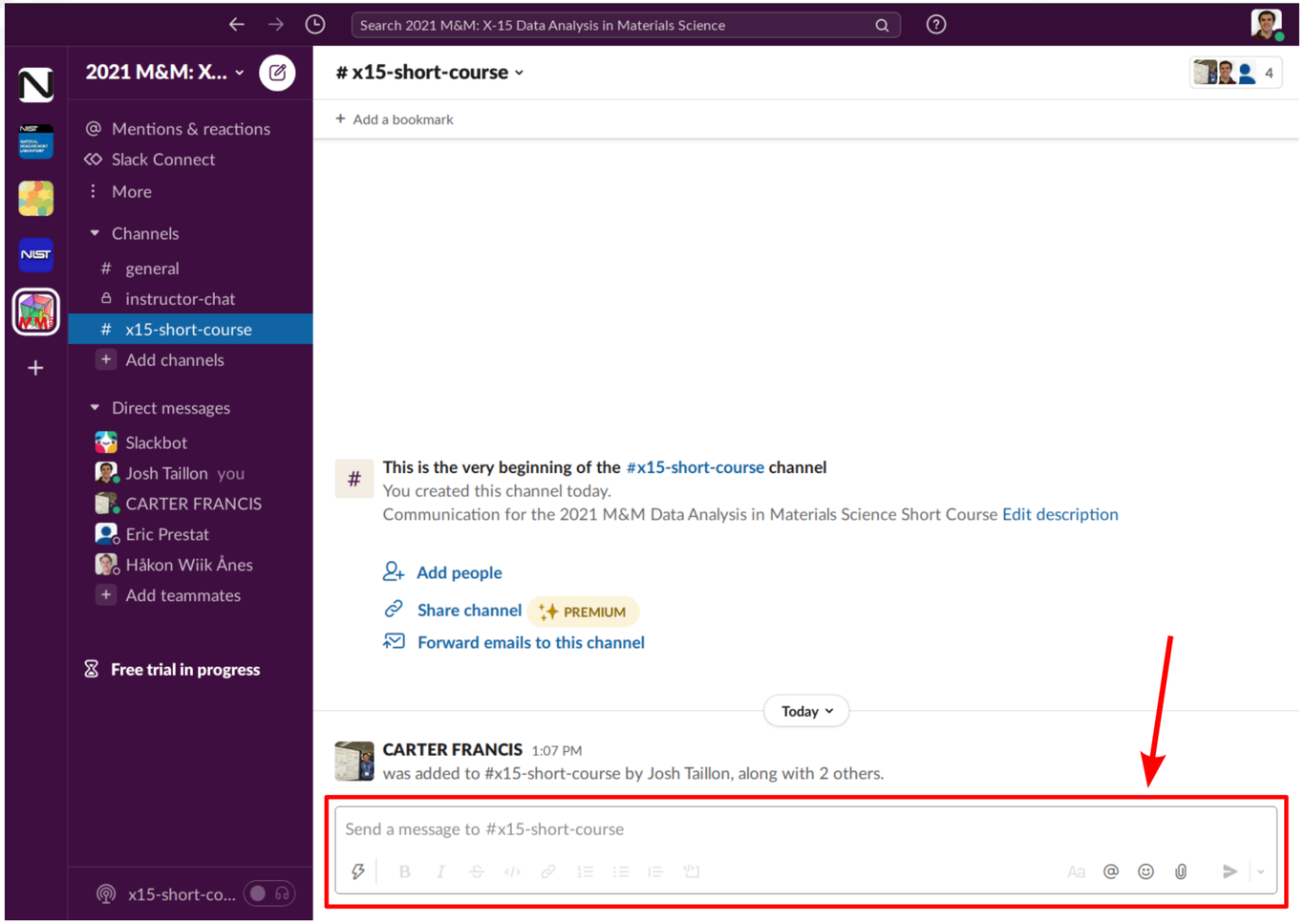
Using Slack — Selecting our workspace



Using Slack — Selecting the right channel



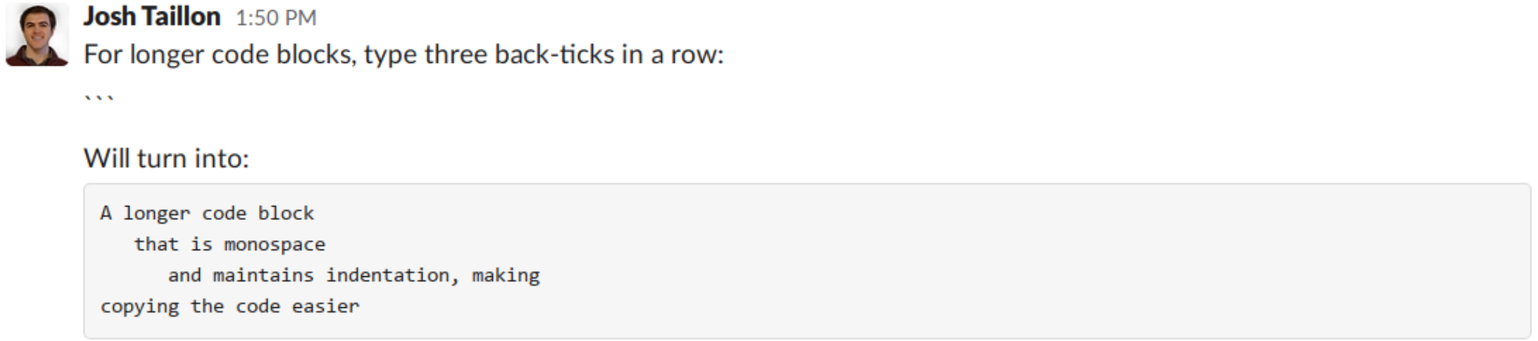
Using Slack — Sending a message



Some Slack tips

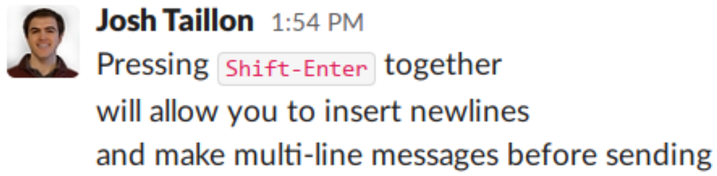
- When writing code into the message box, a few tips will help make your messages easier to read and understand
  - Using back-ticks to make code "monospaced":

  - Using three back-ticks to type or copy/paster longer code blocks:

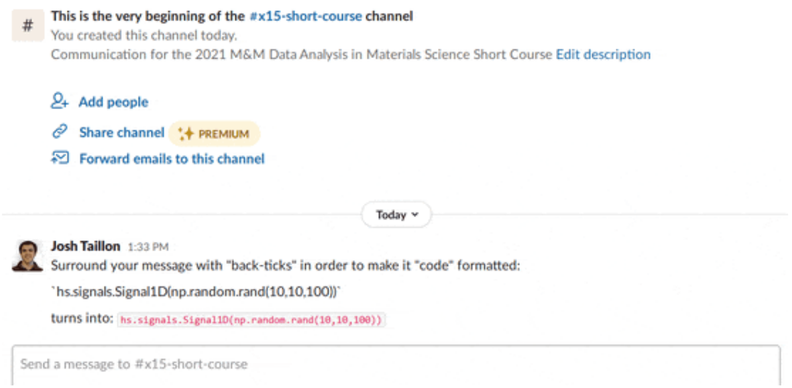


Some Slack tips

- When writing code into the message box, a few tips will help make your messages easier to read and understand
  - Pressing `Shift-Enter` together will insert a new line without sending your message:



  - Messages can be edited or deleted by highlighting over it, clicking the three dots, and then selecting the right option:



A note about Binder:

- Binder (https://mybinder.org) is a tool that allows you to execute Python code using your browser without having to install any software

- You access a server owned by Binder via the web-browser, the code runs on their servers and displays back to you

- This makes it very easy to get familiar with Python/HyperSpy, but will not help you if you need to do your own analysis
    - Only pre-defined "templates" are available, and the server has limited resources
    - It may also be fairly slow depending on the connection speed between you and Binder's servers

Session 1: An Introduction to Python and HyperSpy:

The multi-dimensional data analysis toolbox

Josh Taillon

August 1, 2021

What is this "Python" that I've heard of?

What is python™ ?

- High-level, *general purpose* programming language with "batteries included"
  - *i.e.*, it comes with enough features to do most basic programming tasks
- Simple enough for quick scripts; featured enough for complex projects
- Used extensively on the web, in applications, and throughout science
- Syntax emphasizes readability and explicitness

A "Hello, World!" comparison:

Java:

```java
public class HelloWorld {

  public static void main(String[] args)
    System.out.println("Hello, World!");
  }
}
```

C++:

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

A "Hello, World!" comparison:

Python:

```
print("Hello, World!")
```

Matlab:

```
disp("Hello, World!")
```

Okay, but what does this have to do with science?

The "scientific Python" ecosystem provides almost any functionality you may need:

Linear algebra, optimization, machine learning...

- Numerical computing with `NumPy` (like what you would use Matlab for):

- Signal processing, numerical integration, optimization, etc. with `SciPy`:

- Machine learning with `scikit-learn` (and others):

Visualization, statistics, computer algebra...

- Scientific visualization with `matplotlib`:

- Data frames/series and statistical analysis with `pandas`:

- Symbolic computation with `SymPy`:

A better (?) way to create and publish your work

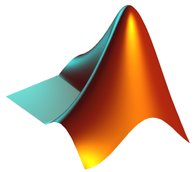
- Literate programming using a notebook interface with *Jupyter*:

- Creation of interactive computational analysis documents (not just scripts):
  - Facilitates reproducibility
  - Makes analyses more accessible (anyone can recreate your figure when they want!)
- Easy to contribute back to the "open source" scientific ecosystem

Can't I just use Matlab?

End To End development to execution	Slightly more code required than Matlab to do same procedures
Vast array of open source scientific packages	Quality of third-party packages varies quite a bit
Great for general programming and application development	Package management (dependencies) can be difficult (getting easier with tools like conda)
Can be a “glue” language to connect R, C++, and others	IDE is not “built-in” like Matlab
Free (as in speech and as in beer)!	By default, some operations are slower

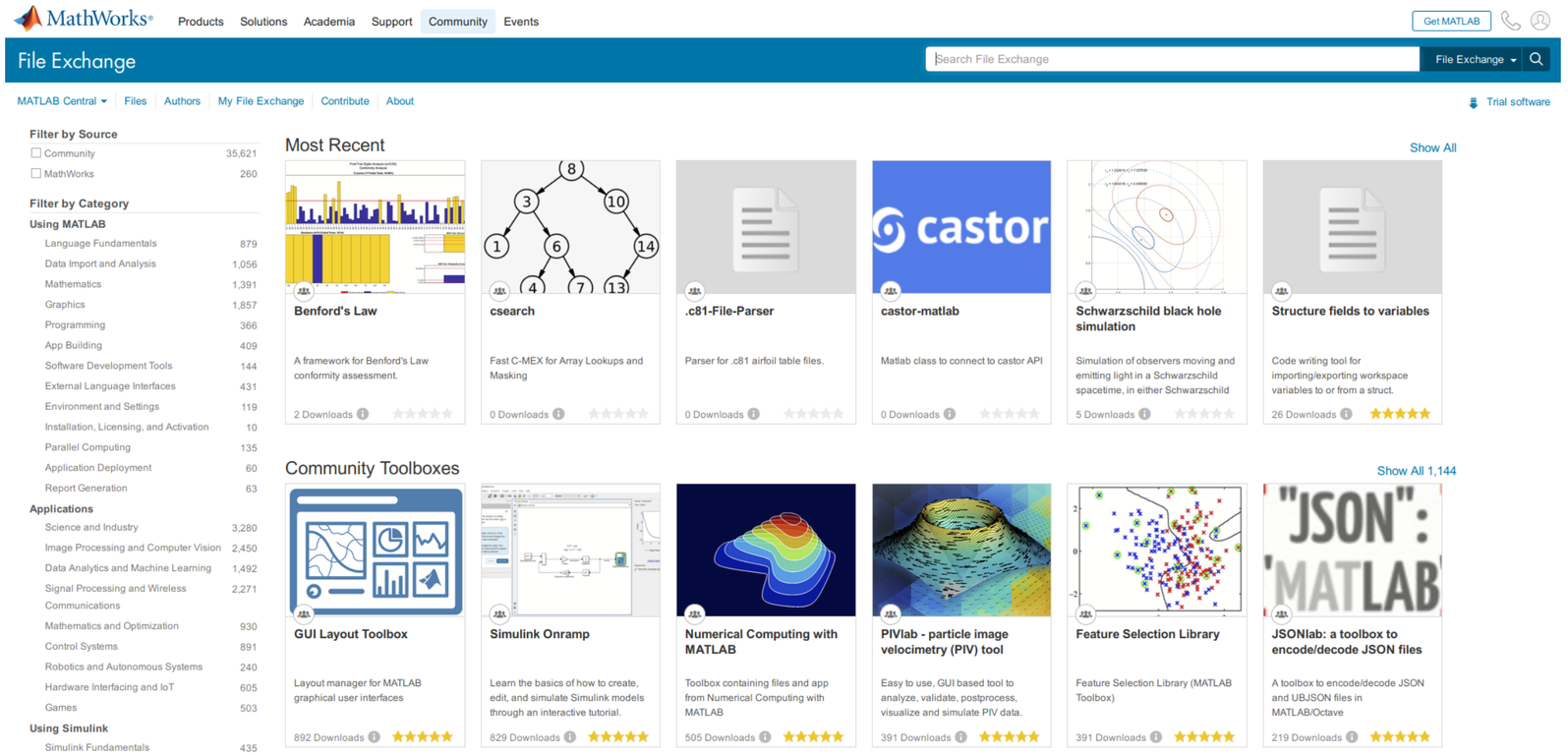
VS.



Scripts tend to be short due to high level of integration	Can not execute code; must be compiled into another language at runtime
Easy visualization and GUI development	Expensive! Must be licensed per-user ($$)
Well-tested and supported (commercial product)	Does not integrate well with other languages large projects are *extremely* difficult to manage
Built-in IDE and powerful debugger	Much functionality locked away in extra licensed packages ($)
Multi-threading easier than in Python	Cannot develop stand-alone applications (must distribute Matlab runtime)

Adapted from: https://towardsdatascience.com/r-vs-python-vs-matlab-vs-octave-c28cd059aa69

File Exchange is good, but...



Distributing code is simple in the Python community

(and integrated directly into Python tools)

- GitHub: Community standard is to (at least) release code on GitHub (or similar service)
- python Package Index: Placing into the PyPI, the **Py**thon **P**ackage **I**ndex enables installing with simple:
  - `pip install my-awesome-package`
- ANACONDA: Anaconda enables multiple environments and complex dependency management
  - `conda install my-awesome-package`

What does it mean to be "open source"?

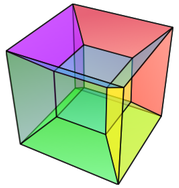
> *…something people can modify and share because its design is publicly accessible*
>
> — *opensource.com*

What about "open science"?

> *…science must be done in an open, and reproducible fashion where all components of research are open*
>
> — *Marcus Hanwell (also opensource.com)*

Open-source and Python in the microscopy ecosystem:

General Purpose		Others	
HyperSpy	http://hyperspy.org/	AbTEM	https://abtem.readthedocs.io
Nion Swift	https://nionswift.readthedocs.io	HRTEMFringe Analyzer	https://github.com/ialxn/HRTEMFringeAnalyzer
pycroscopy	https://github.com/pycroscopy/pycroscopy	Atomap	https://atomap.org/
DigitalMicrograph	https://www.gatan.com/run-python-script	py_multislice	https://github.com/HamishGBrown/py_multislice
		Kikuchipy	https://kikuchipy.org
Pixelated STEM		Prismatic	https://prism-em.com/
pyXem	https://pyxem.github.io/pyxem-website/	tomopy	https://tomopy.readthedocs.io/en/latest/
py4DSTEM	https://py4dstem.readthedocs.io/	tomotools	https://github.com/usnistgov/tomotools
LiberTEM	https://libertem.github.io/LiberTEM/	tomviz	https://tomviz.org/
fpd	https://gitlab.com/fpdpy/fpd/		

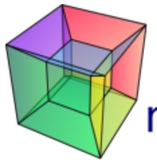


What is HyperSpy, anyway?

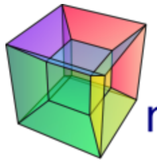
What is  HyperSpy multi-dimensional data analysis ?

- Open-source Python library for interactive data analysis of multi-dimensional datasets
- Leverages the Scientific Python ecosystem for much of its functionality:
  -       matplotlib
- Accessed like any other Python library:

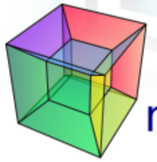
```
import numpy as np      ---->      import hyperspy.api as hs
```

Why would you use  HyperSpy multi-dimensional data analysis ?

- Makes it easy to operate on multi-dimensional arrays as you would a single spectrum (or image)
- Easy access to cutting-edge signal processing tools
- Modular structure makes it easy to add custom features or extend into dedicated packages
- Use of Jupyter notebooks encourages reproducible and sharable analyses (FAIR data)

Why would you use  HyperSpy multi-dimensional data analysis ?

- Beyond generic signal processing, provides many tools specifically for electron microscopy:
    - Provides facilities for easy access to proprietary software formats
    - EDS:
        - Background removal
        - Net intensity line map extraction
        - Quantification (k-factor, ζ-factors, ionization cross sections)

Why would you use  HyperSpy multi-dimensional data analysis ?

- Beyond generic signal processing, provides many tools specifically for electron microscopy:
    - EELS:
        - Background removal
        - Curve fitting for quantification (including ELNES)
        - All of Egerton's famous methods
    - "Advanced" methods:
        - Multivariate statistical analysis
        - General curve fitting
        - Dimensionality reduction/signal separation

Why  python™?

- Quickly becoming the *de facto* standard of scientific computing
- Free (as in speech and as in beer)
  - No pesky licenses to checkout
- Vast array of scientific libraries available:
  - `pip install <just about anything>`
- Thanks to `numpy` and other libraries, similar (or often better) performance than MATLAB

History of HyperSpy

- Developed by Francisco de la Peña in 2007 — 2012 as part of his Ph.D. Thesis at Université Paris-Sud in Orsay, France

- See Chapter 5 of the dissertation for a historical introduction

- Originally called EELSLab:

Talk EELSLab: a Python toolbox for (hyper)spectroscopy data analysis

Presented by Francisco de la Peña in Poster and Demo Session 2011

Abstract

EELS*Lab*: a Python toolbox for (hyper)spectroscopy data analysis

F. de la Peña[1, 3, *], M. Sarahan[2], S. Mazzucco[4, 5], L-F. Zagonel[1, **], M. Walls[1]

1) Laboratoire de Physique des Solides, Bât. 510, Université Paris-Sud. 91405 Orsay Cedex

2) SuperSTEM, STFC Daresbury Laboratories. Keckwick Lane, Warrington WA4 4AD – UK

3) CEA-LETI, MINATEC 17, avenue des Martyrs, 38054 GRENOBLE Cedex 9 – France

4) Center for Nanoscale Science and Technology, National Institute of Science and Technology, 100 Bureau Drive, Gaithersburg, MD 20899-6203, USA

5) Institute for Research in Electronics and Applied Physics (Bldg. 223), Paint Branch Drive, University of Maryland, College Park, MD 20742-3511, USA

* Current address: department of Materials Science and Metallurgy, University of Cambridge. Pembroke Street, Cambridge, CB2 3QZ – UK

** Current address: Associação Brasileira de Tecnologia de Luz Sincrotron, Laboratório de Microscopia Eletrônica.13083-970 - Campinas, SP – Brasil

Modern scientific instruments from several disciplines now yield multidimensional spectroscopic data. As an example, a modern transmission electron microscope (TEM) can acquire spectral data from sub-atomic volumes that ultimately could reveal the position and nature of each atom of a material [1]. However, such datasets are usually quite large and difficult to work with.

EELS*Lab* [2] has been developed as a tool to facilitate hyperspectral data analysis. Originally it was intended for TEM data analysis, but it has been successfully used in other domains. Specifically, it provides easy access to multidimensional curve fitting, peak analysis and machine learning algorithms, as well as a viewing framework for navigating data and reading and writing capabilities for several popular hyperspectral formats.

This talk will discuss how Python has been used to implement these features, with demonstrations of applications to both spatially-resolved spectroscopic data (so-called spectrum images) [3] and to structural analysis of atomic resolution image stacks [4]. The blend of intuitiveness, power, and availability of high-quality scientific libraries that Python offers has allowed the creation of a simple, natural tool that scientists from many disciplines can both use and easily extend into new scientific domains.

References

[1] Sandra Van Aert et al., « Three-dimensional atomic imaging of crystalline nanoparticles », *Nature* 470, n°. 7334 (17th of February, 2011): 374-377.

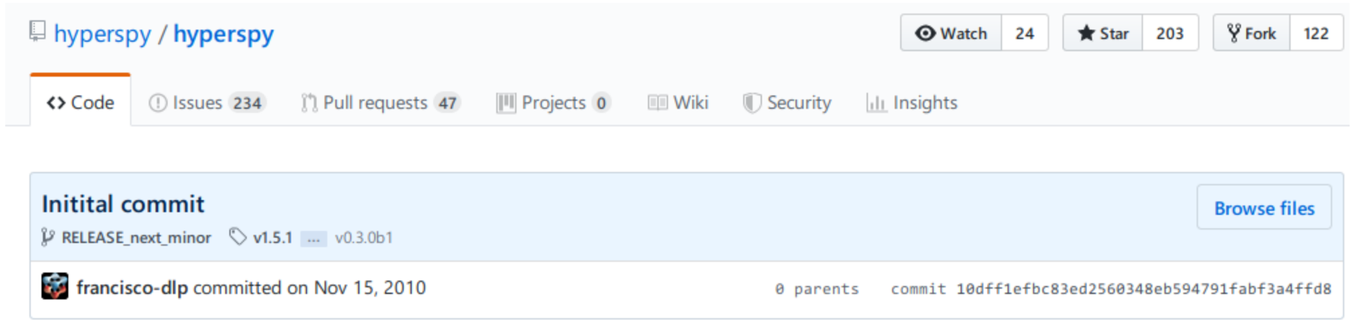
[2] http://www.eelslab.org

[3] R. Arenal et al., « Extending the analysis of EELS spectrum-imaging data, from elemental to bond mapping in complex nanostructures », *Ultramicroscopy* 109, n°. 1 (December 2008): 32-38.

[4] Michael C. Sarahan et al., « Point defect characterization in HAADF-STEM images using multivariate statistical analysis », *Ultramicroscopy* 111, n°. 3 (February 2011): 251-257.

History of HyperSpy

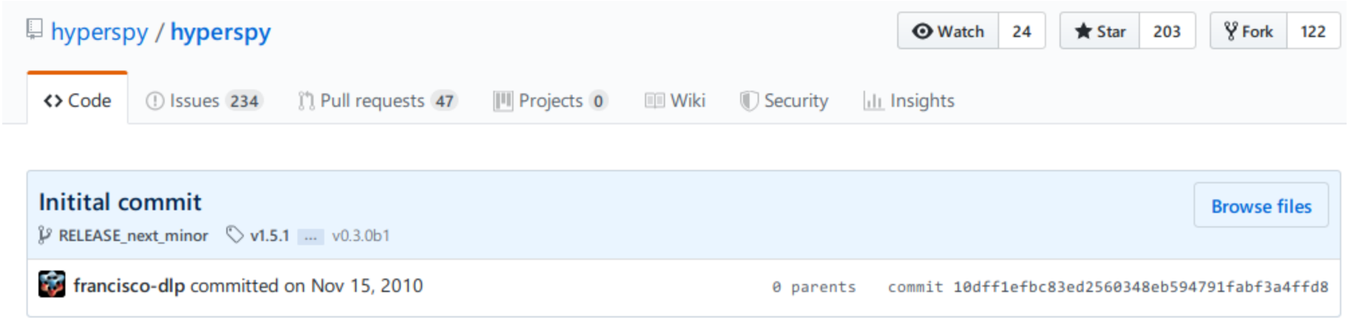
- Open-sourced (on Github) in 2010:



- Renamed to HyperSpy in 2011

History of HyperSpy

- Open-sourced (on Github) in 2010:

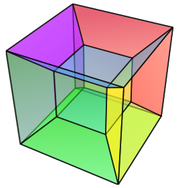


- Renamed to HyperSpy in 2011

- Now... over 630 citations, 43 releases, 53 contributors, used in (at least) 89 other Github projects, and rapidly growing!

Design philosophy of HyperSpy

- HyperSpy is a Python library, rather than standalone program
    - Part of the greater scientific Python ecosystem
- Data storage is in an open hierarchical format (HDF5)
    - Saves all metadata by default (including most processing steps)
- Analysis typically done via reproducible notebooks
- Feature development is completely open-source (GPLv3)



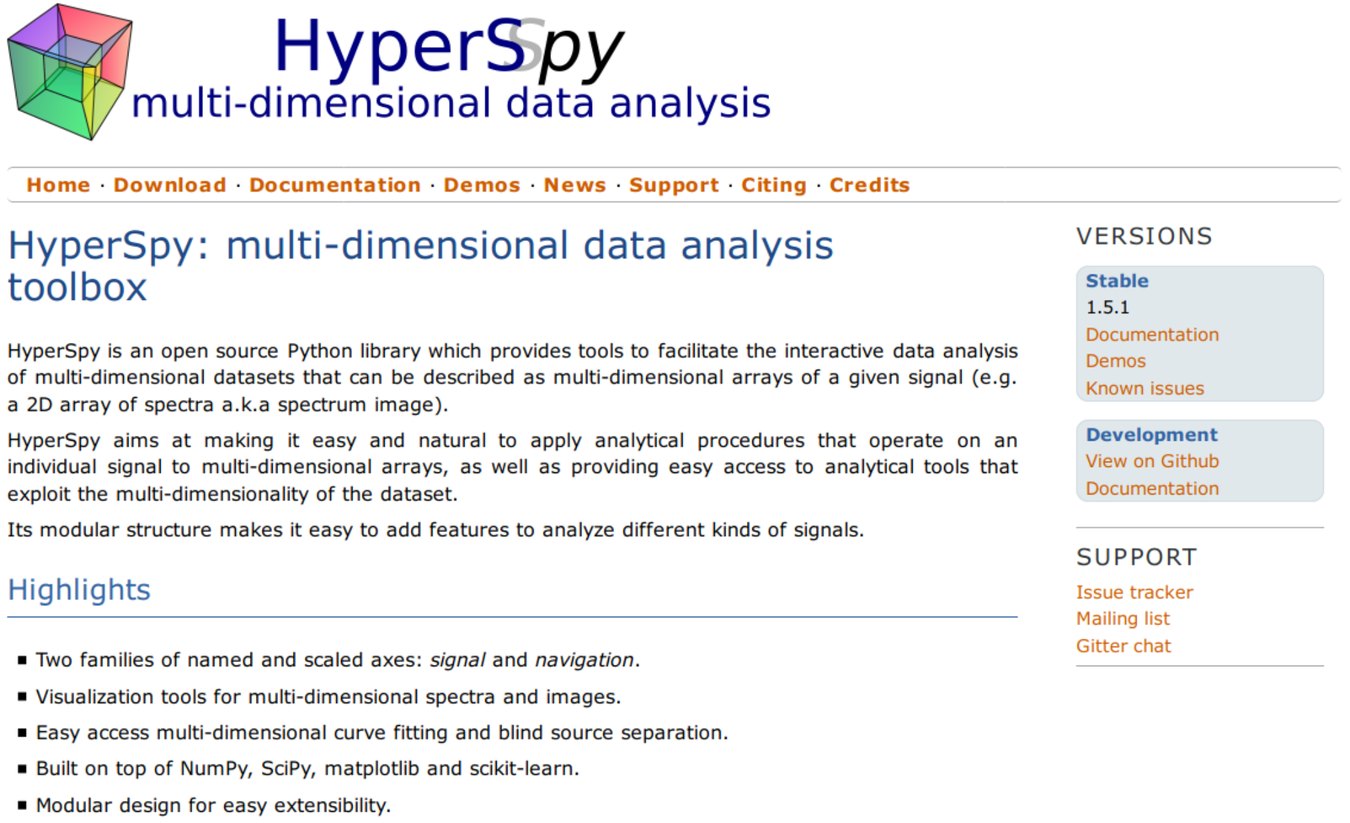
Welcome to the community!

The power of community

- One of the *best* parts of HyperSpy is the community that surrounds it (personal opinion)
- This software is made by researchers, for researchers
- HyperSpy is built from collaboration:
    - By collaborating, we advance faster and avoid reinventing the wheel

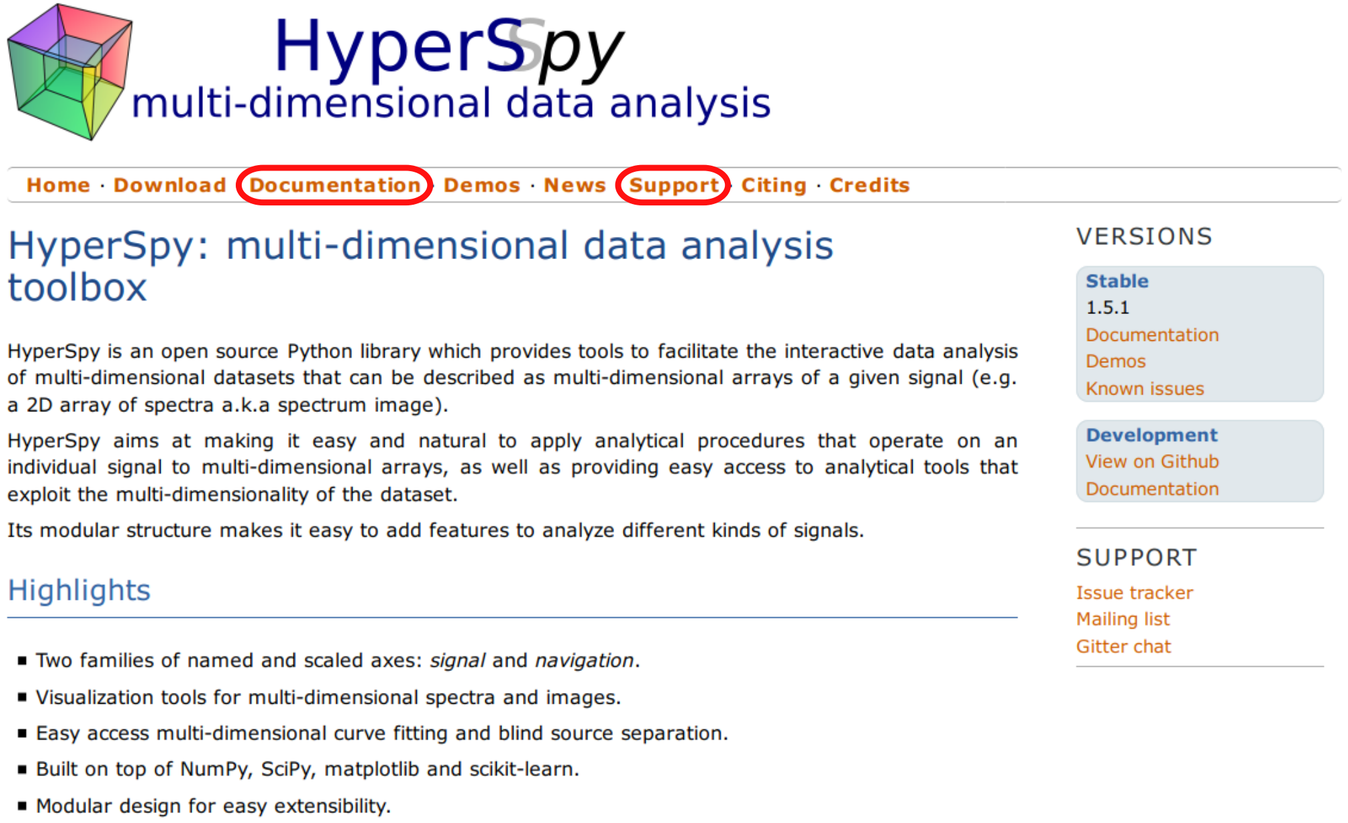
How to get help?

- HyperSpy website (www.hyperspy.org):



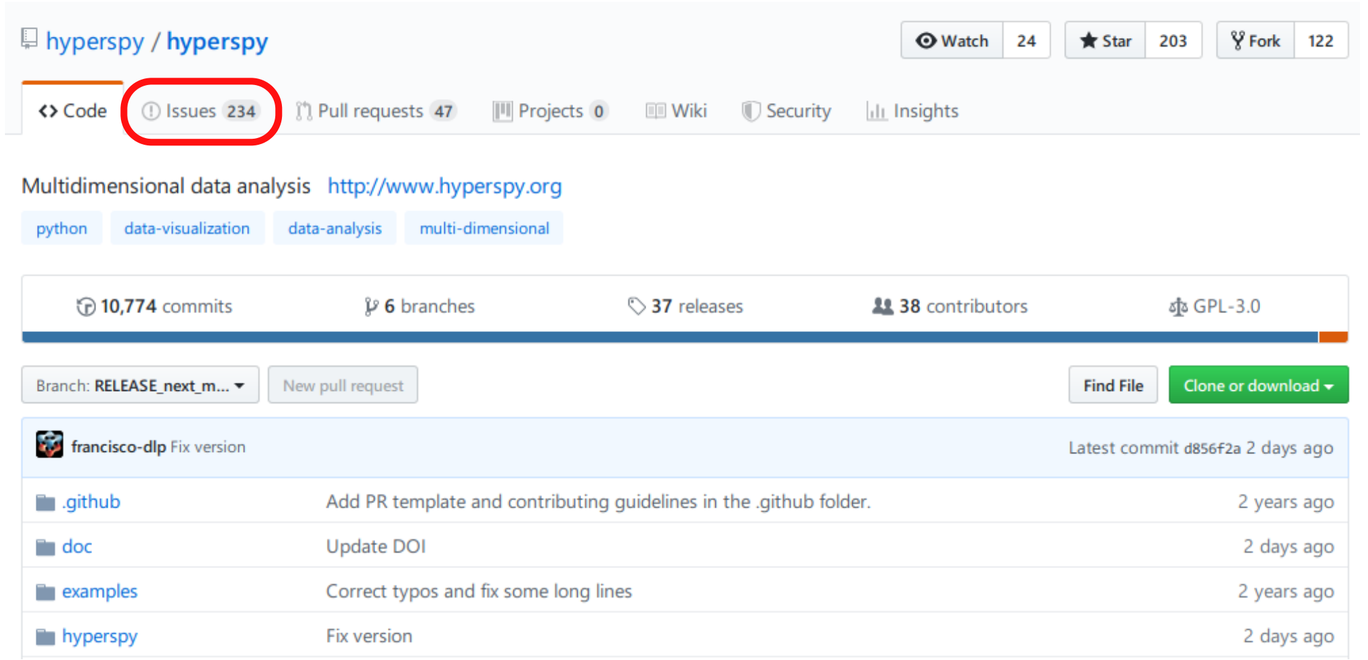
How to get help?

- HyperSpy website (www.hyperspy.org):



How to get help?

- HyperSpy repository (https://github.com/hyperspy/hyperspy):

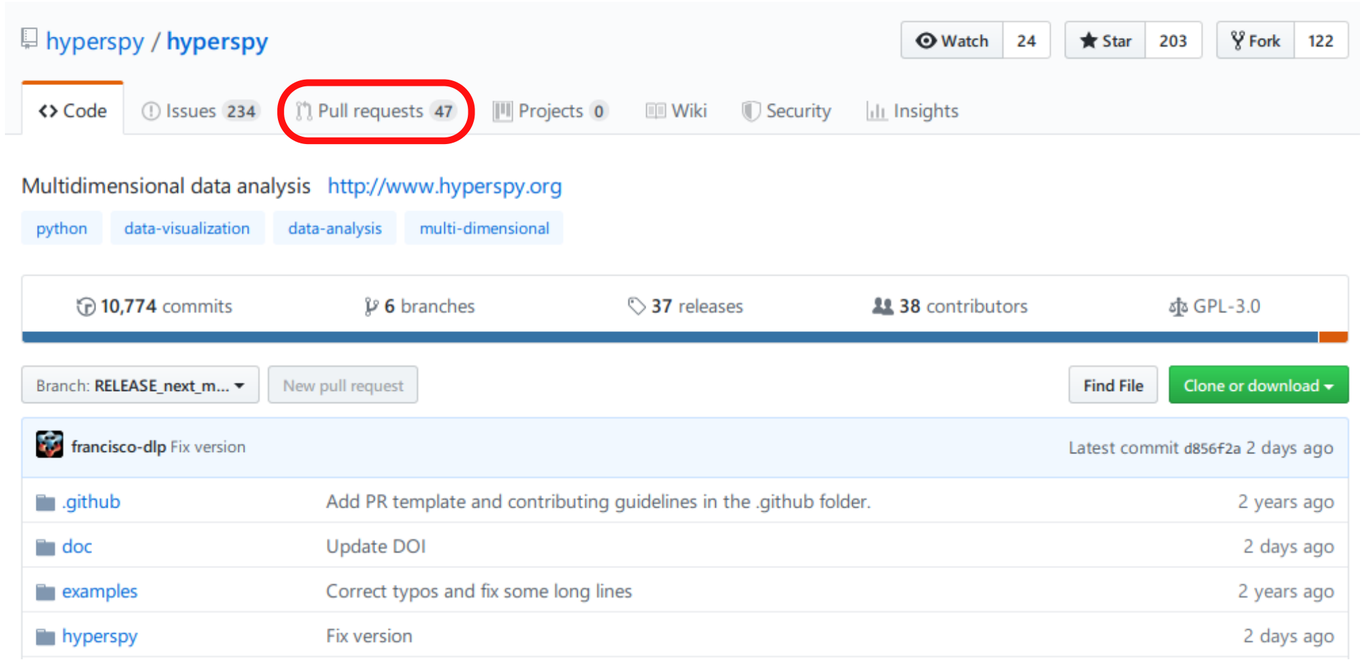


How to get (more) help?

- Sessions like this (good job!)
- Expansive user guide and documentation: https://hyperspy.org/hyperspy-doc/current/user_guide/intro.html
- Tutorials and demos: https://github.com/hyperspy/hyperspy-demos
- User group list: hyperspy-users@googlegroups.com
- Gitter chat: https://gitter.im/hyperspy/hyperspy
- Developer guide (if you're into that sort of thing): https://hyperspy.org/hyperspy-doc/current/dev_guide/intro.html

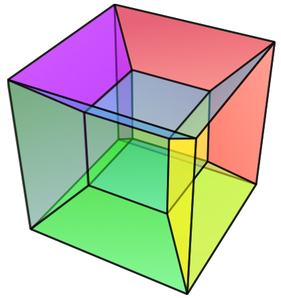
Anyone can make HyperSpy better!

- HyperSpy repository (https://github.com/hyperspy/hyperspy):



What about general Python questions?

- There are vast numbers of resources about Python online (and in print):
  - Official documentation: https://docs.python.org
  - Getting started with scientific Python: https://scipy-lectures.org
  - NumPy for Matlab users: https://docs.scipy.org/doc/numpy/user/numpy-for-matlab-users.html
- StackOverflow for general programming/Python questions: https://stackoverflow.com/



Thank you!

Any questions?

joshua.taillon@nist.gov