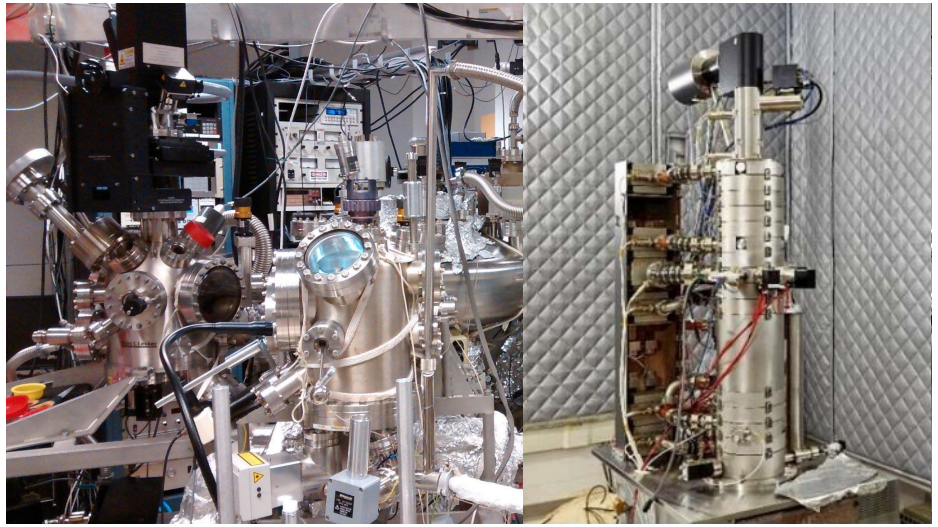
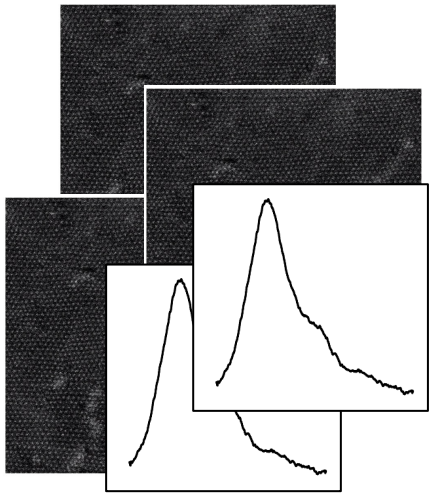
Edge/Cloud Compute Framework for Autonomous Microscopy

Human-centric → AI-powered

Instrument



Images and spectra

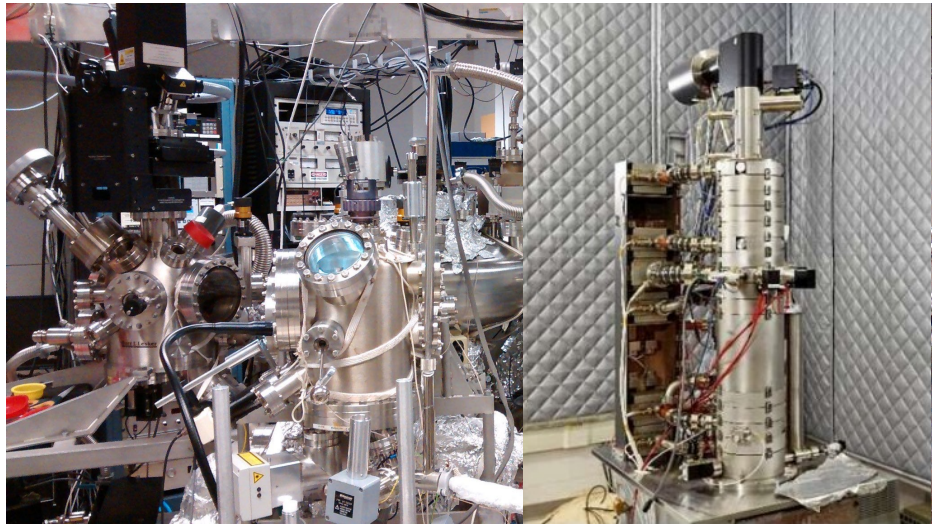


Human scientist

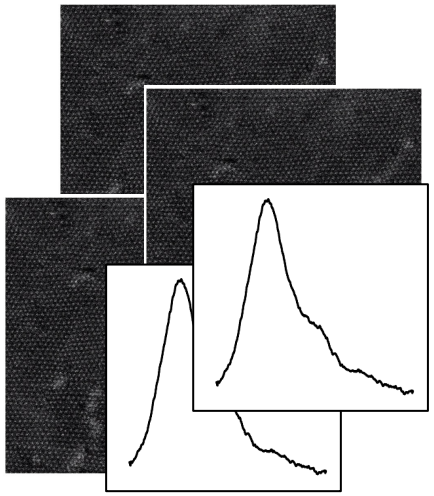
Decision about the next measurement

Human-centric → AI-powered

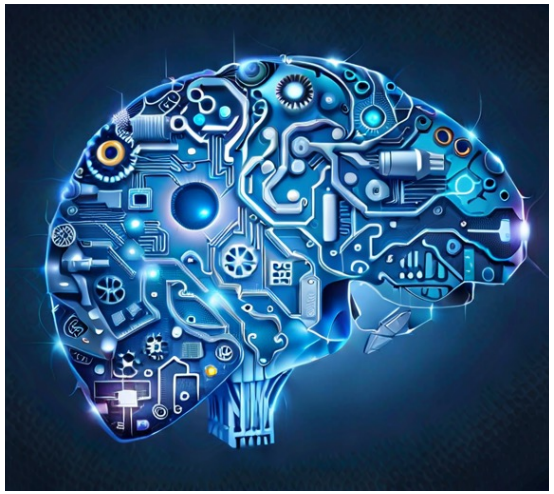
Instrument



Images and spectra



"AI"



Awesome pattern recognition powers

Decision about the next measurement

… require awesome compute powers

Active learning

Active learning

Inputs: Array of initial measurements D, Array of unmeasured points X_*, Surrogate model *Model* (GP, BNN, DKL), Acqusition function *acq*, Number of steps N_{steps}

for 1,…, N_{steps} **do**

Computationally costly; requires state-of-the-art GPUs

Use MCMC/SGD to obtain posterior samples for *Model* parameters given D

Compute posterior predictive distribution, $\bar{\mathbf{f}}_*$, and uncertainty, $U[\mathbf{f}_*]$, over X_*

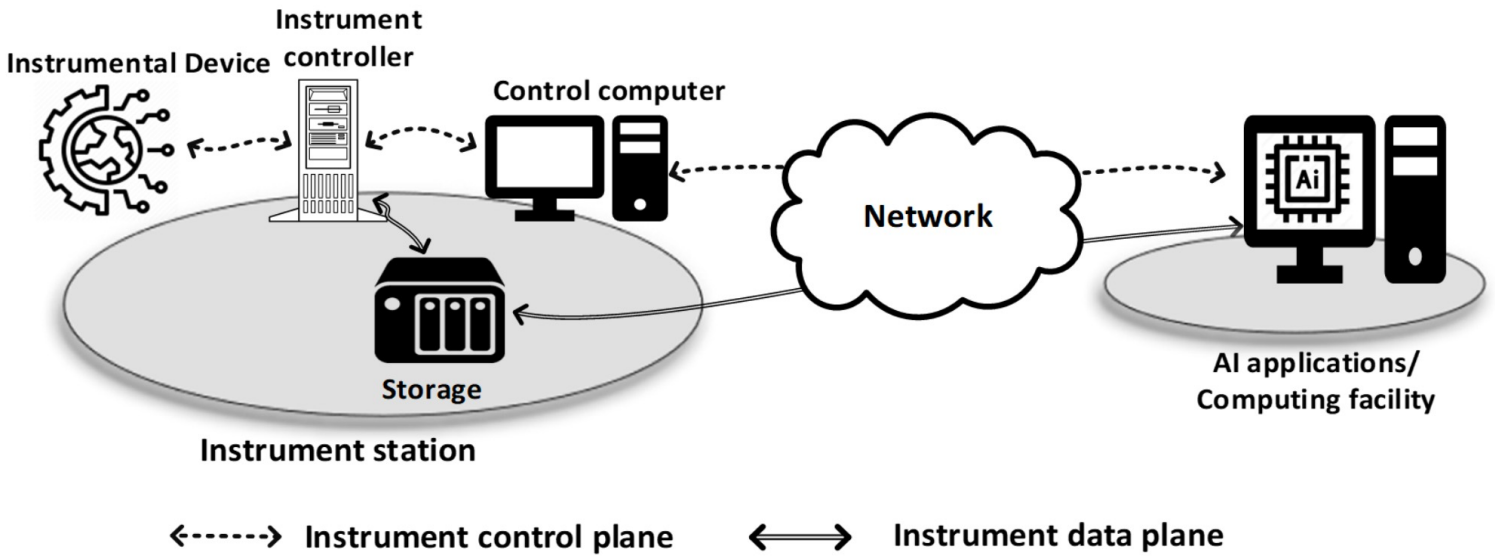
Compute acquisition function, $a = acq(\bar{\mathbf{f}}_*, U[\mathbf{f}_*])$

Derive the next evaluation point, $x_{next} = \arg\max a$

Perform a measurement in x_{next}; update D and X_*

end

Our solution

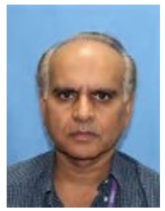


Key features:

- Control channel for remote instrument steering
- Data channel for streaming measurements across the ecosystem
- Cross-facility analytics and science workflow orchestration

A. Al-Najjar, N. S. Rao, et al., IEEE 18th International Conference on e-Science (e-Science), pp. 267–277, IEEE, 2022.

Anees Al-Najjar



Nageswara Rao

Pyro client-server modules

A Pyro Server running on an instrument compute controller

```
import Pyro4
import ..
.
.                                          } Importing Pyro4 and the Instrumentation Python packages
@Pyro4.expose
class Embedded_Server(object):
   def __init__(self):
     #define science experiment parameters

   def control_module_i(self, arg*,kwarg**):
     .
     .                                     } Build a class of science experiment functions to be published by Pyro
     return (attr_1,..,attr_n)

Pyro4.Daemon.serveSimple(
{
 Embedded_Server:'Instrument_Server',      } The Pyro Daemon turns the Python object (class Embedded_Server) into a Pyro object (Instrument_Server) to publish it and listen to incoming requests.
},host=ipAddressServer,
 port=int(connectionPort),
 ns=False, verbose=True)
```

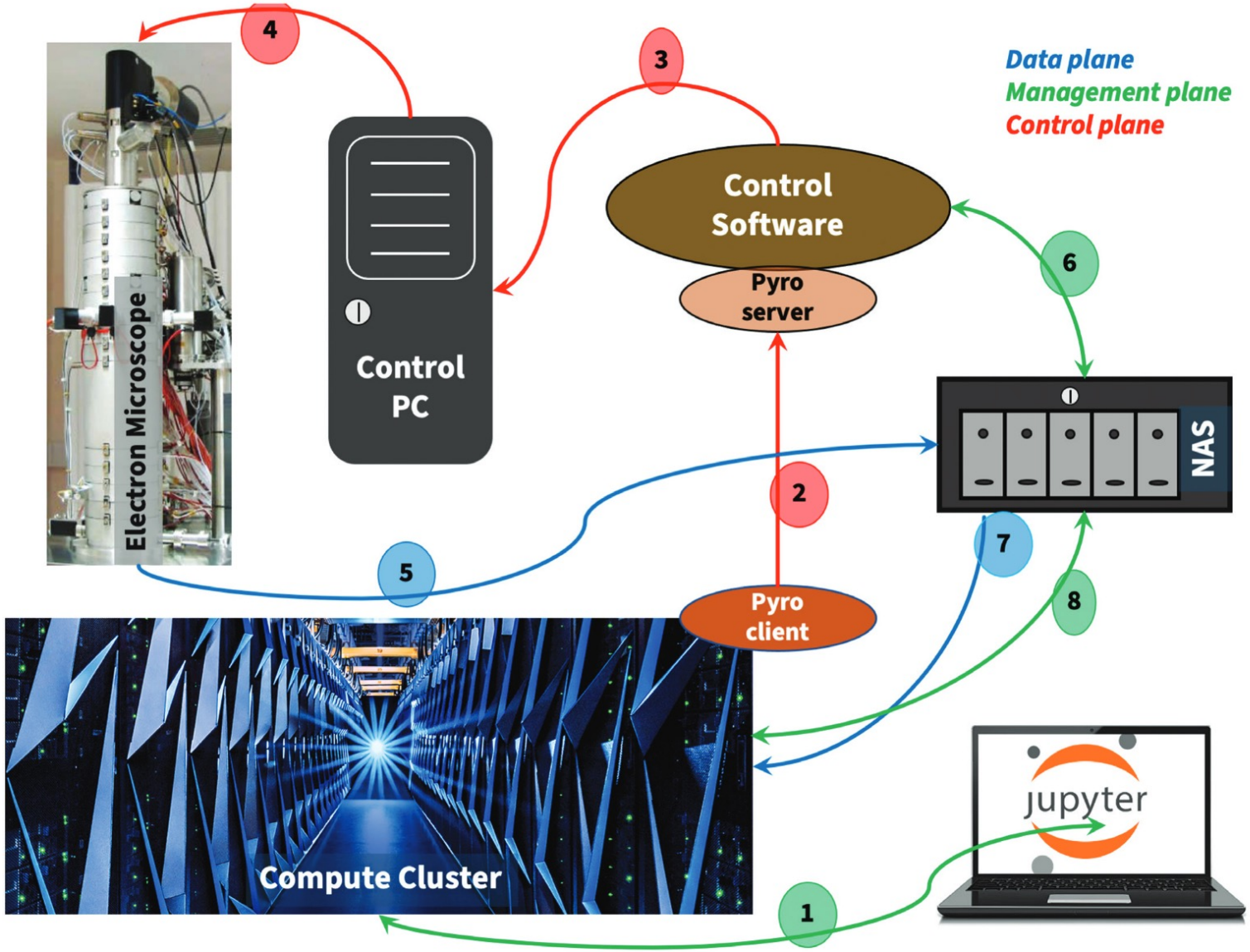
A Pyro client application running on a remote node.

```
import Pyro4                    Initiate the connection with the Pyro server based on its IP address and TCP port number
.
.
module_call =
Pyro4.core.Proxy('PYRO:Instrument_Server@'+
 ipAddressServer + ':' + connectionPort)

Out_1,..,out_n=\
module_call.control_module_i(arg*,kwarg**)
.
.                               Remotely call an experiment function and pass arguments
```


OAK RIDGE National Laboratory

Our solution



Debangshu Mukherjee

Anees Al-Najjar

Nageswara Rao

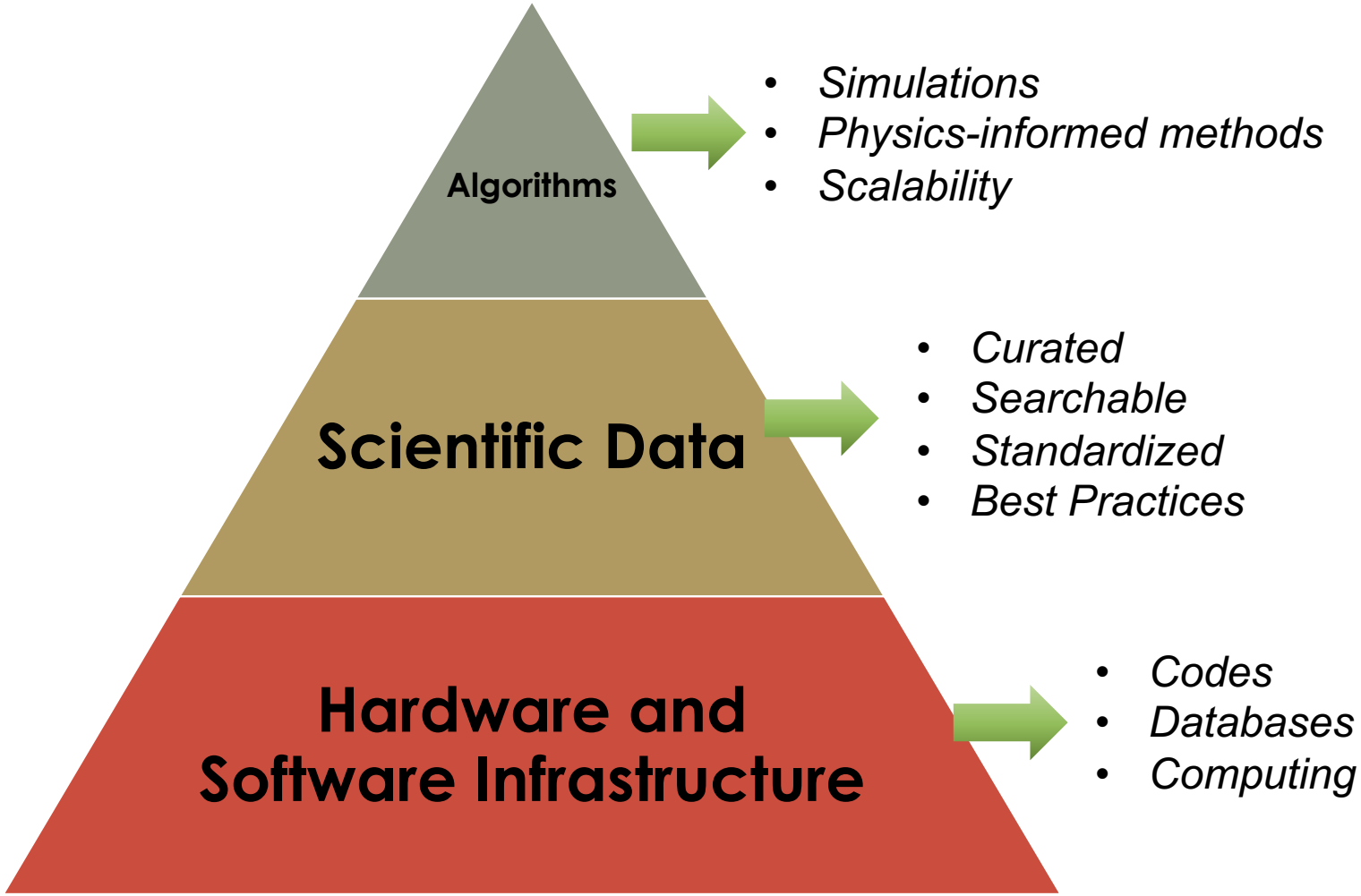
D. Mukherjee et al., *Microscopy Today*, *30*(6), 10-19 (2022)

Pycroscopy, Data & Code Infrastructure

Why bother with infrastructure?

- Almost all of machine learning relies extensively on having access to good quality data
- In most laboratories, this data is acquired via multiple instruments in different formats, and not findable or accessible, and often lacks necessary metadata for ML labeling
- As such, in many cases, ML in science is impossible especially in the experimental domains, without the necessary investments in data standardization and storage
- Similarly, reproducibility of workflows relies on strongly tested codebases, not one-off scripts.
- Pycroscopy is our method at attempting to alleviate this problem

The pyramid of machine learning



- *Simulations*
- *Physics-informed methods*
- *Scalability*

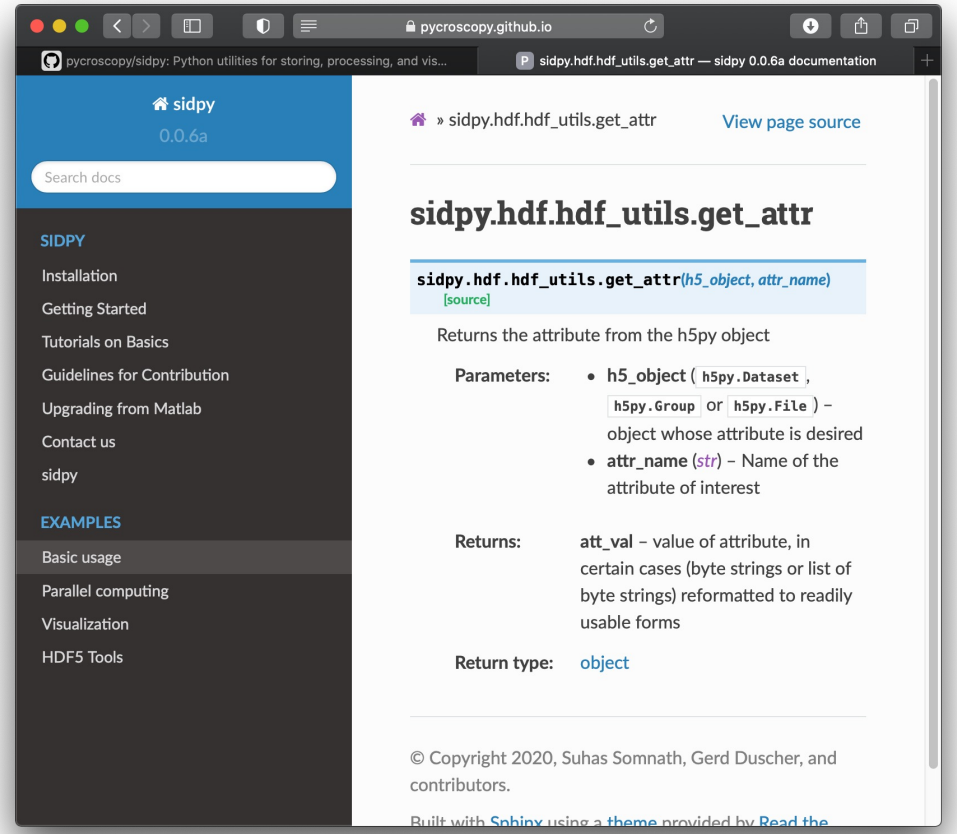
- *Curated*
- *Searchable*
- *Standardized*
- *Best Practices*

- *Codes*
- *Databases*
- *Computing*

pycroscopy

github.com/pycroscopy

An ecosystem for microscopy data ingestion, analytics and visualization



pycroscopy

A general-purpose package for microscopy imaging and spectroscopy data analytics, including regisytration, image cleaning, unmixing, ect.

scifireaders

For ingesting a variety of microscopy f les for output to sidpy dataset objects

pyusid

Python package for reading and visualizing our universal spectral imaging dataset format

pynsid

Python package for reading writing and visualizing our N-dimensional spctral imaging datase format

sidpy

Python utilities for storing, visualizing, and Spectroscopic and Imaging Data (SID)

bglib

Utilities to analyze, f t and visualize Band-excitation and G-mode imaging and spectroscopy data promarily for SNMS SPM users

atomai

Deep learning toolkit for analysis of atomiclly resolved imageing and spectroscopy datasets

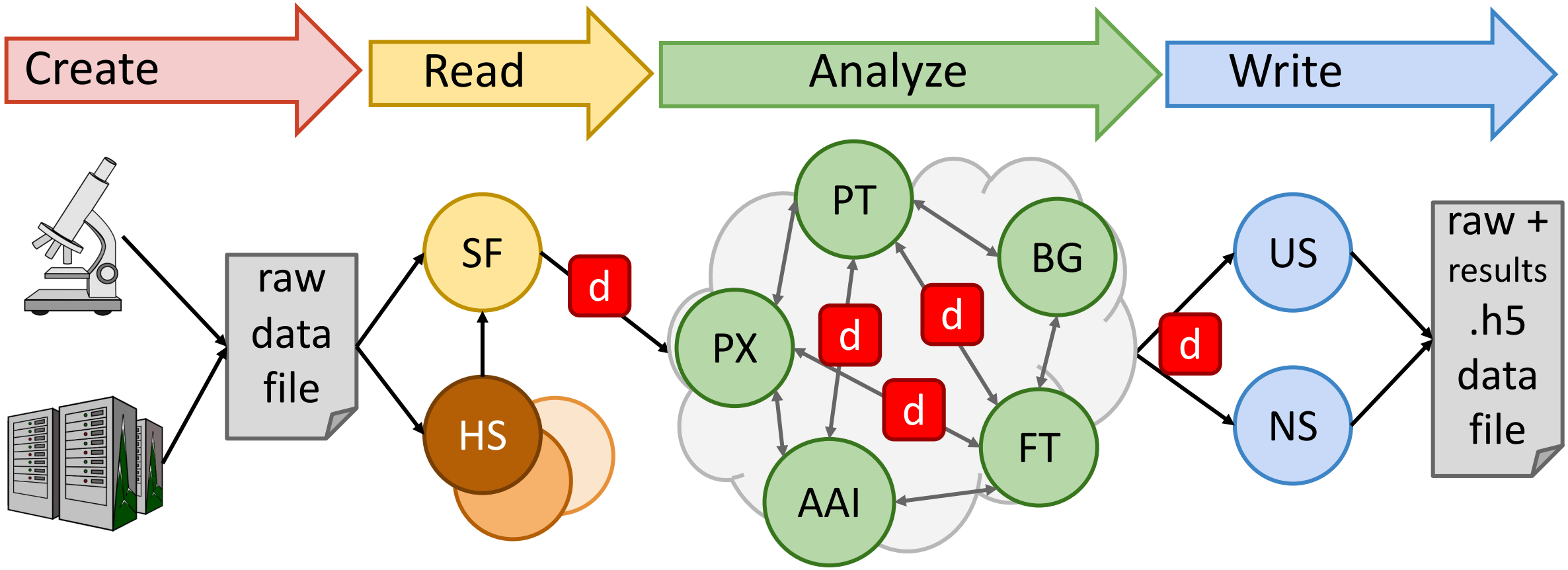
stemtools

Python based codes for analysis of 4D-STEM and abreeation corrected vanilla STEM datasets

pytemlib

Python tools for simulation, registraion, analysis and visualization of TEM datasets

Pycroscopy philosophy



Data from measurements or simulations are read into **`sidpy.Dataset`** (d) objects directly by **`SciFiReaders`** (SF). Data are processed using multiple science packages in the Pycroscopy ecosystem that interoperate via **`Dataset`** objects. **`Dataset`** objects are written to HDF5 files via **`pyUSID`** (US) or **`pyNSID`** (NS).

Standardization of Microscopy data

Slide by S. Somnath

Micro Raman Microscope

Atomic Force Microscope (AFM)

AFM with Infrared spectroscopy (AFM-IR)

Scanning Tunneling Microscope (STM)

Scanning Transmission Electron Microscope (STEM)

AFM with Raman spectroscopy

Multitude of File Formats

.wdf

.ibw

.cdb

- Proprietary
- Incompatible

≠

.asc

.dm3

.mat

Slide by S. Somnath

Disjoint & Unorganized Communities

- Clustering
- Fit spectra …

- Filter Image
- Register Image
- …

- Fit Spectra
- SVD Filtering …

- FFT Filtering
- SVD Filtering …

- FFT Filtering
- Classify Images …

- Register Images
- Clustering

Slide by S. Somnath

Cannot Share Code Efficiently

- HIGHLY instrument-specific code
- Different programming languages
- Often licensed / costly software like Matlab
- Most popular sharing method = email!
- No centralized repository

Slide by S. Somnath

The Solution

Instrument Tier

Automated, standardized, modularized data acquisition

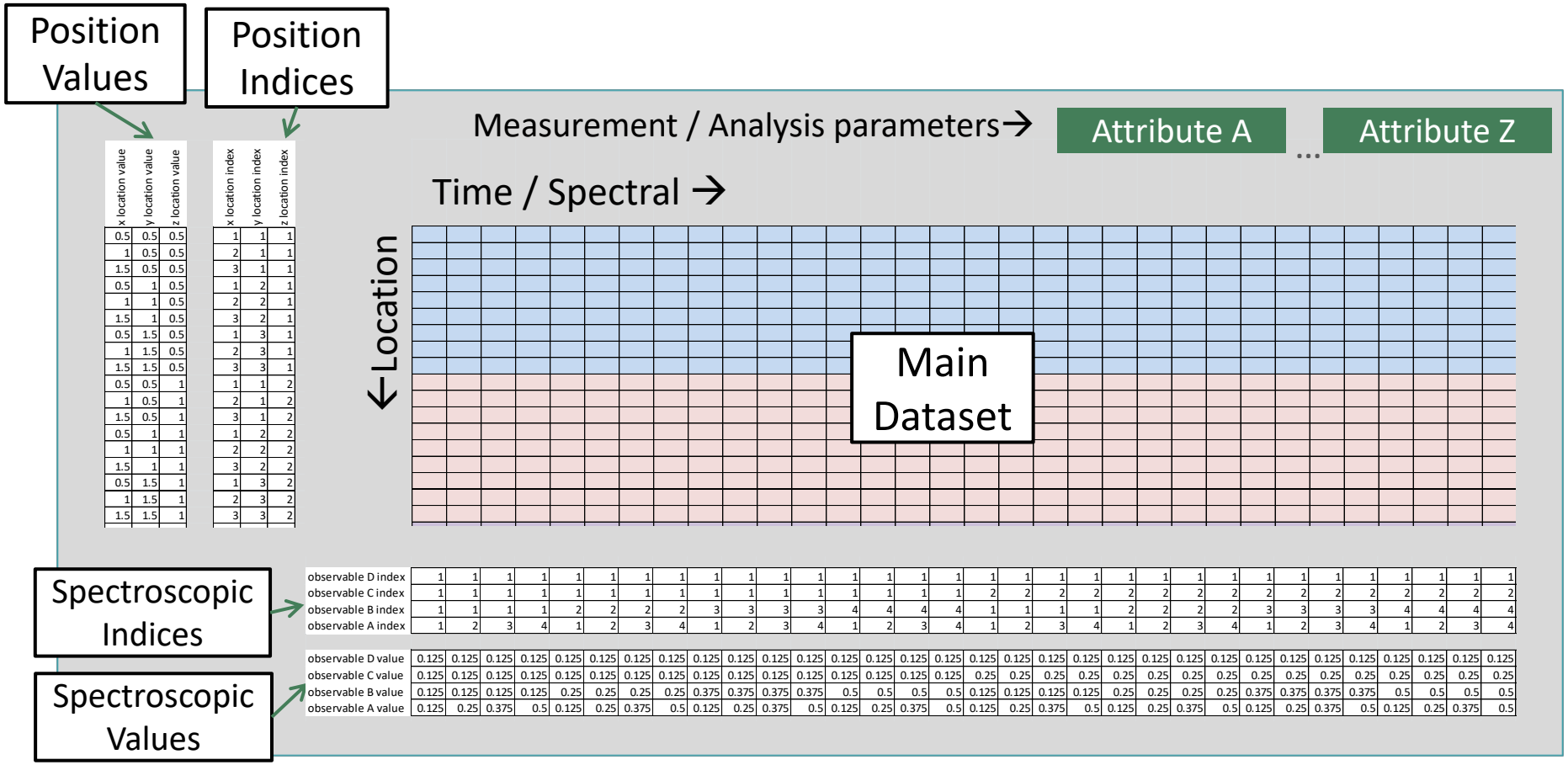
Instrument-agnostic, self-describing, model in an open friendly file format

Centralized repository for data processing, analysis

Interactive visualization + analysis + storage on the cloud

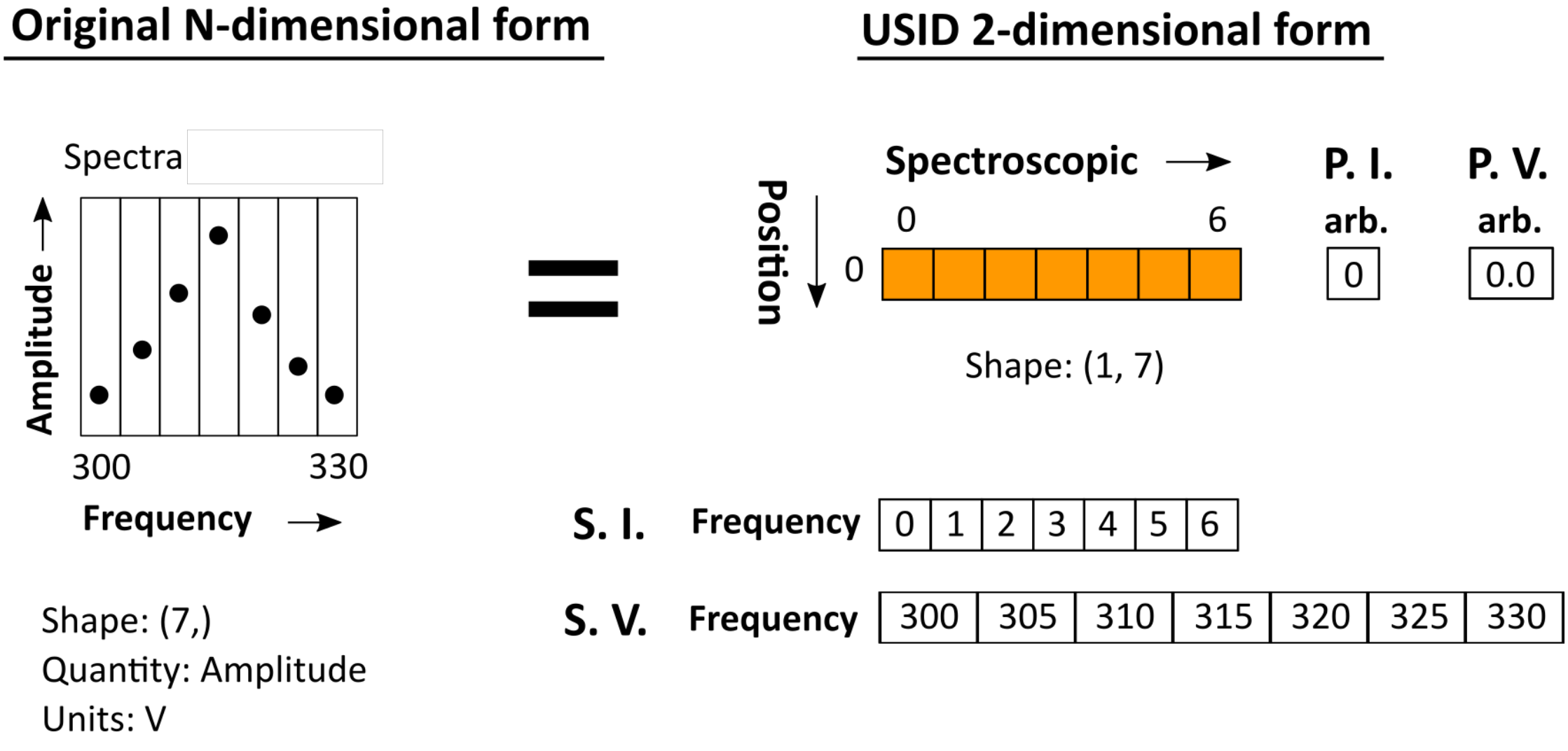
Universal Spectroscopic and Imaging Data (USID)

- Data stored as 2D matrix of (position x spectral values) regardless of dimensionality
- Ancillary datasets explain the data
- We recommend USID only when NSID is not suitable, due to higher difficulty for regular use.



Detailed information on pyUSID website

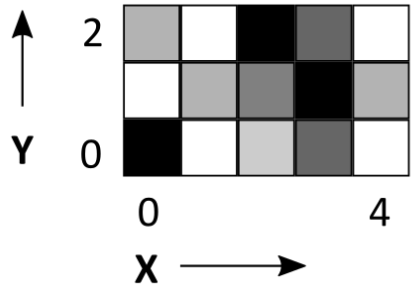
USID – 1D spectra



Slide by S. Somnath

USID – 2D Image

Slide by S. Somnath



Shape: (3, 5)

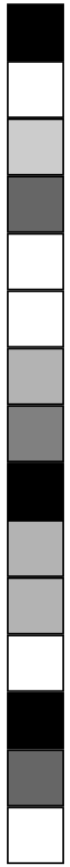
Original N-D form

Quantity: Intensity
Units: arb. units

=

Spectroscopic →

Position ↓



S. I. arb. 0

S. V. arb. 0.0

P. I.

X	Y
0	0
1	0
2	0
3	0
4	0
0	1
1	1
2	1
3	1
4	1
0	2
1	2
2	2
3	2
4	2

P. V.

X	Y
-250	0
-125	0
0	0
125	0
250	0
-250	3.5
-125	3.5
0	3.5
125	3.5
250	3.5
-250	7
-125	7
0	7
125	7
250	7

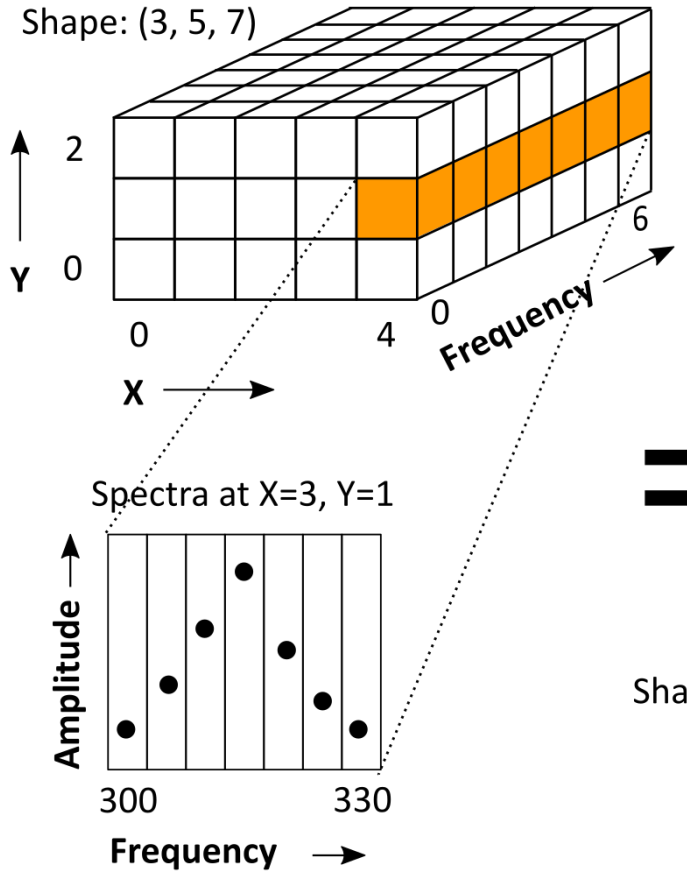
Shape: (15, 1)

USID 2D form

USID – Spectra on Grid (3D)

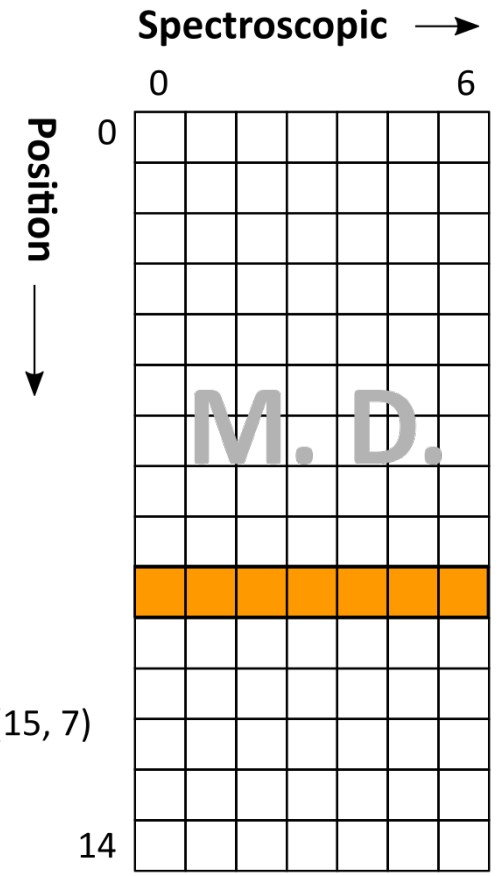
Slide by S. Somnath

Original N-dimensional form



Quantity: Amplitude
Units: V

USID 2-dimensional Form



P. I.

X	Y
0	0
1	0
2	0
3	0
4	0
0	1
1	1
2	1
3	1
4	1
0	2
1	2
2	2
3	2
4	2

P. V.

X	Y
-250	0
-125	0
0	0
125	0
250	0
-250	3.5
-125	3.5
0	3.5
125	3.5
250	3.5
-250	7
-125	7
0	7
125	7
250	7

S. I. Frequency

0	1	2	3	4	5	6

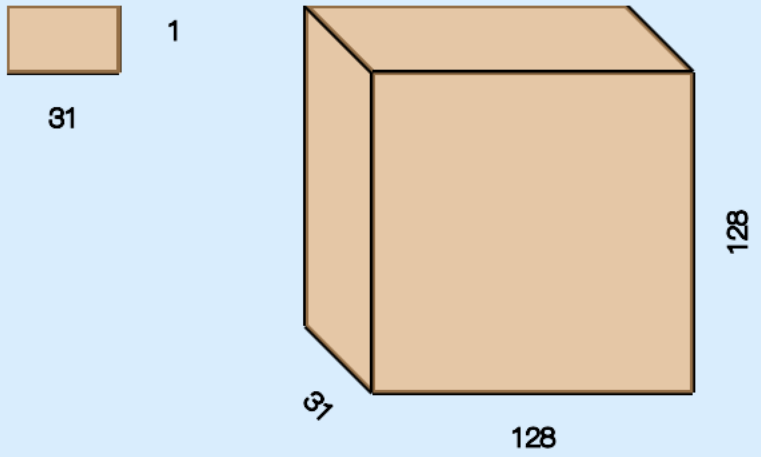
S. V. Frequency

300	305	310	315	320	325	330

NSID Model (implemented as sidpy.Dataset)

Dataset Object built on top of dask arrays

	Array	Chunk
Bytes	120.12 MiB	120.12 MiB
Shape	(31, 31, 128, 128)	(31, 31, 128, 128)
Count	1 Tasks	1 Chunks
Type	float64	numpy.ndarray



Benefits of the model:

- Easy to understand
- sidpy takes care of plotting (dataset.plot())
- Can easily perform parallel computations
- Easy to push to file including metadata
- Useful for data pipelining

- Maintain N-dimensional form
- All of the advantages of dask (large sizes, parallel compute)
- Additional data given for each dimension of dataset, such as name, quantity, units
- Metadata stored in dictionary
- Can be readily pushed to hdf5 files

Two Data models: USID and NSID

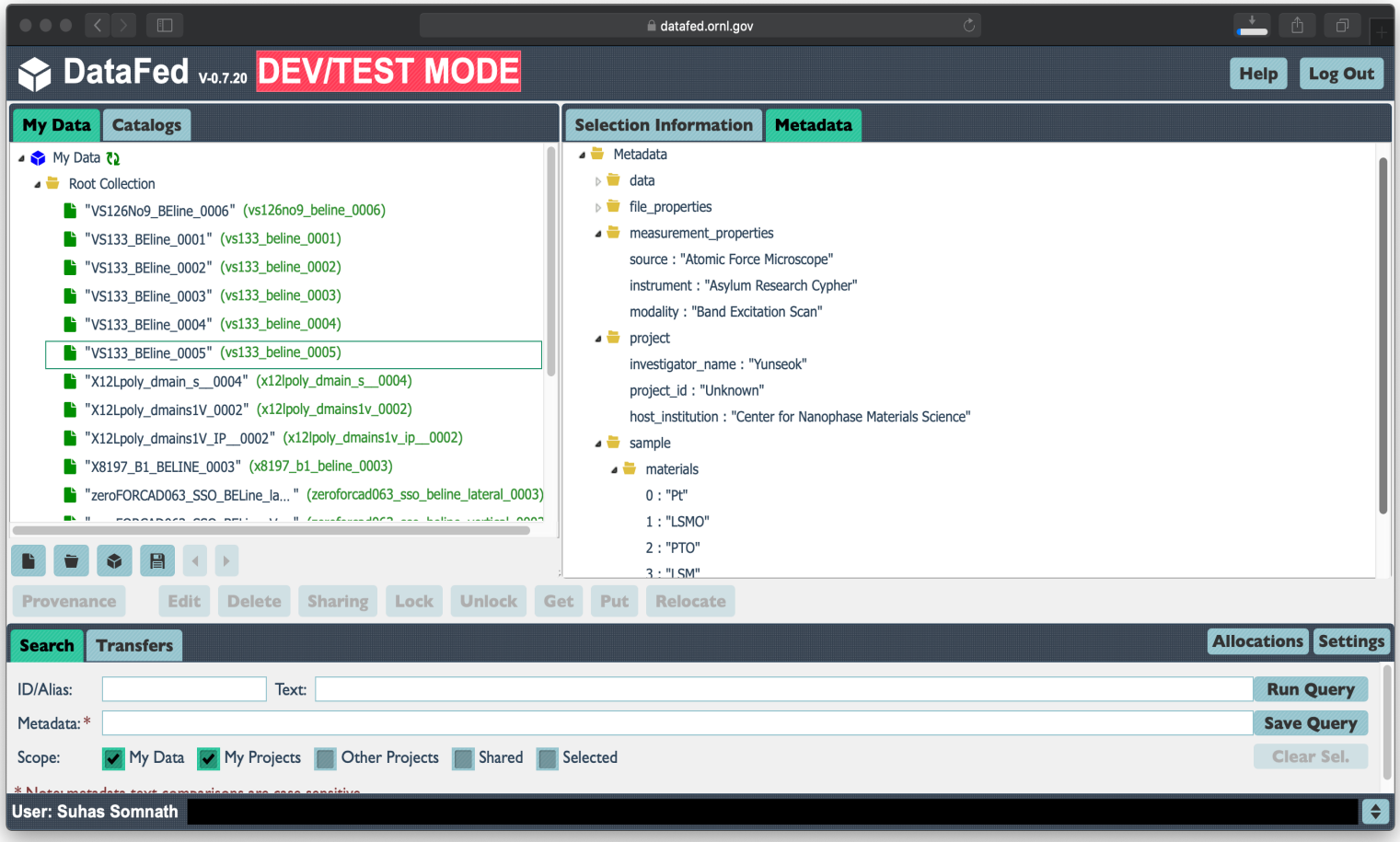
Universal spectral imaging dataset (USID) and n-dimensional spectral imaging dataset (NSID)

Feature	NSID	USID
Accommodate N-dimensional datasets of different shapes, sizes	Yes	Yes
Accommodates data without N-dimensional representation	No**	Yes
Stores metadata	Yes	Yes
Stores very large files	Yes	Yes
Enables Parallel processing	Yes	Yes
Traceable	Medium	High
Ease of use	High	Medium

** potentially, point clouds can be stored but this remains work in progress

DataFed

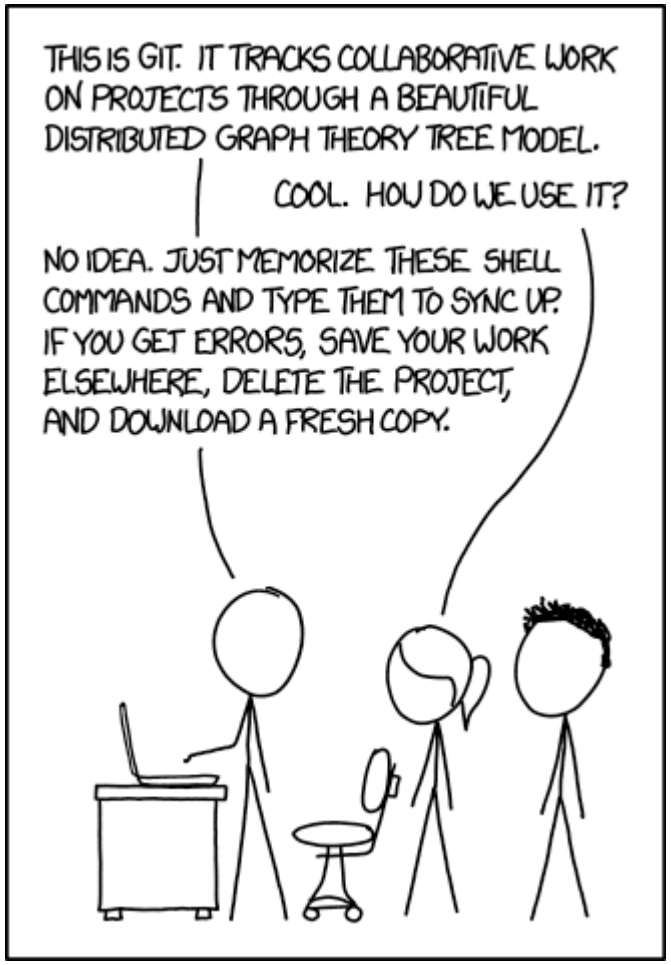
- Once the data is standardized, it can be easily stored and accessed, e.g. with ORNL's open DataFed (federated data storage solution).



Code repositories and version control

- Sharing scripts between users can be workable for immediate or short-term needs, but is not scalable nor lasting
- For reproducibility, it is better to have codes that reside in packages that are documented and well tested
- Most of you are familiar with python packages; but many are probably new to version control
- Version control systems such as git enable multiple people to work on a single software project at the same time to speed up development and ensure consistency
- Git is an open-source distributed version control system. It maintains a history of changes that have occurred in the project and allows for updates as well as reversions to older ‘commits’.

More about git



Git would take a significant amount of time to explain in detail. However, there are plenty of online tutorials for you to try, e.g.

https://www.atlassian.com/git/tutorials

For now, we will test some of the basic git functionality in an interactive demo.

Make sure you have a github account

Head on over to https://github.com/ramav87/ML_Summer_Course

We will practice forking a repository, making a change to the code and submitting a pull request

Pycroscopy code demo

Here we will go through sidpy and pycroscopy's core functions and features