Introduction to NWB & DANDI

Luiz Tauffer
Research Software Engineer, CatalystNeuro

Alessandra Trapani
Research Software Engineer, CatalystNeuro

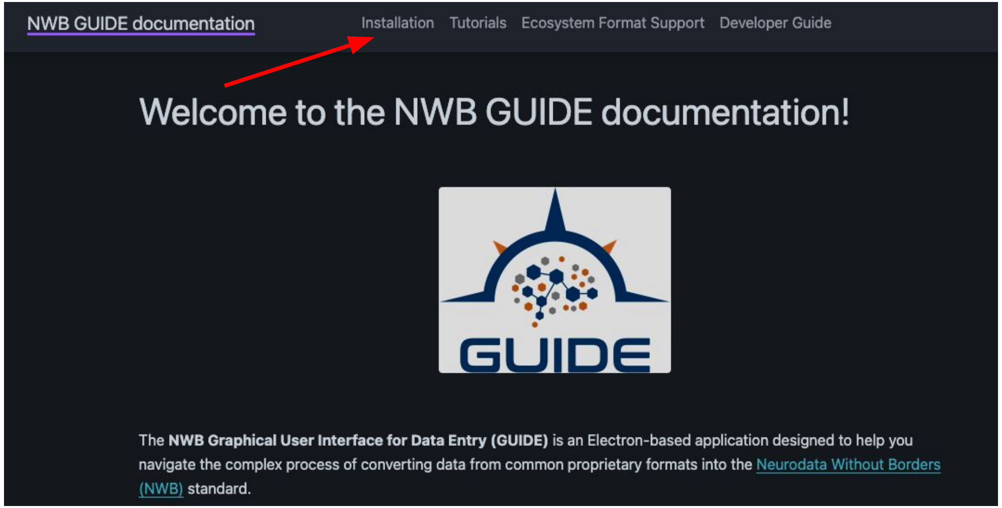
Schedule

10:15 - 10:45	Overview of NWB and DANDI
10:45 - 11:00	Q&A
11:00 - 12:15	NWB GUIDE (Interactive Tutorial) ● Easily convert data ● Inspect NWB File ● Explore NWB File
12:20	Group photo
12:30-13:30	Lunch
13:30 - 14:30	Advanced NWB conversion
14:30 - 15:30	Analysis tools in the NWB ecosystem

NWB GUIDE Tutorial

Installation

- https://nwb-guide.readthedocs.io/en/v0.0.15/index.html



Need for neurophysiology data standard and archive

- Data are expensive to collect
  - Money, time, animal use
- Sharing neurophysiology data within a lab and with collaborators is tedious
- Sharing scientific software is also difficult
- It is often easier to just collect new data and build new tools
- Difficult for labs to devote time and effort to sharing data and software

New NIH Data Management and Sharing Policy

For NIH projects starting Jan 25, 2023:

> “Shared scientific data should be made accessible as soon as possible, and no later than the time of an associated publication, or the end of performance period, whichever comes first.” [1]

Applications must include Data Management and Sharing Plan, which indicates use of a data standard and archive [1]

> “For BRAIN Initiative applications … applicants are required to share the data they collect using the BRAIN Initiative informatics infrastructure (both data archives and relevant data standards)” [2]

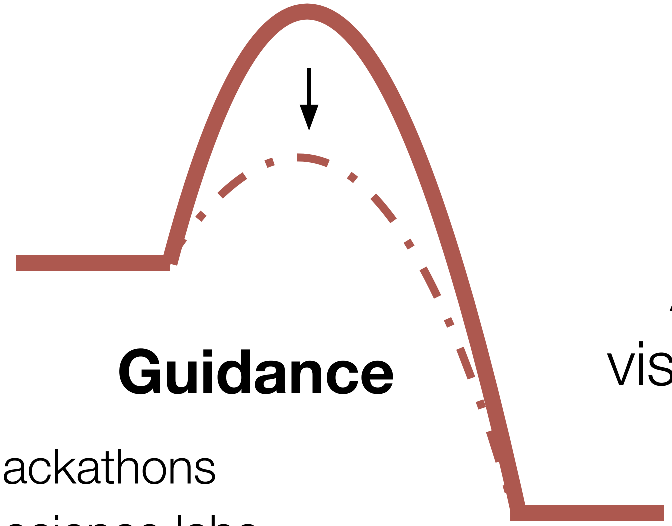
[1] https://sharing.nih.gov/data-management-and-sharing-policy/sharing-scientific-data/data-sharing-approaches
[2] https://grants.nih.gov/grants/guide/notice-files/NOT-MH-19-010.html

The energy barrier in data standardization

↑ Sticks

Funding ultimatives

No standardization



↓ Carrots

Analysis and visualization tools

Standardization

Guidance

- Documentation, tutorials, hackathons
- Working directly with neuroscience labs
- Building and distributing open source tools

Neurodata Without Borders and the DANDI Archive

NEURODATA WITHOUT BORDERS


Oliver Ruebel


Ryan Ly


Ben Dichter

BERKELEY LAB

CATALYST NEURO

A standard for sharing, archiving, and analyzing neurophysiology and behavioral data

DANDI


Satra Ghosh


Yaroslav Halchenko

MIT DARTMOUTH

kitware Delivering Innovation

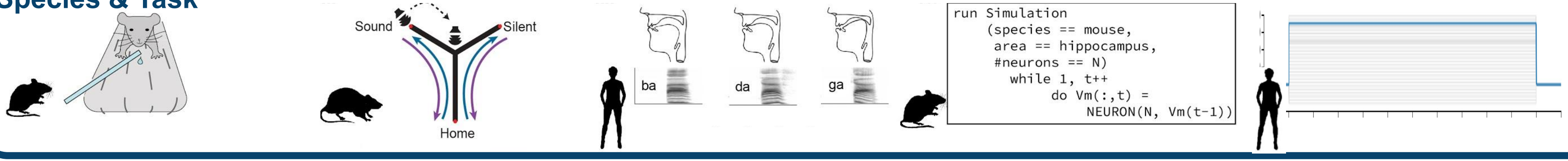
CATALYST NEURO

An archive for sharing and collaboration space for analyzing neurophysiology data

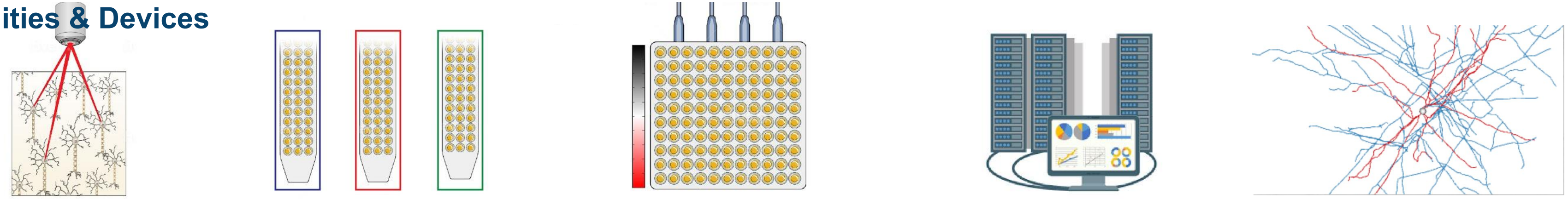
NIH The BRAIN Initiative®

Challenge: Diversity of experiments and rate of innovation

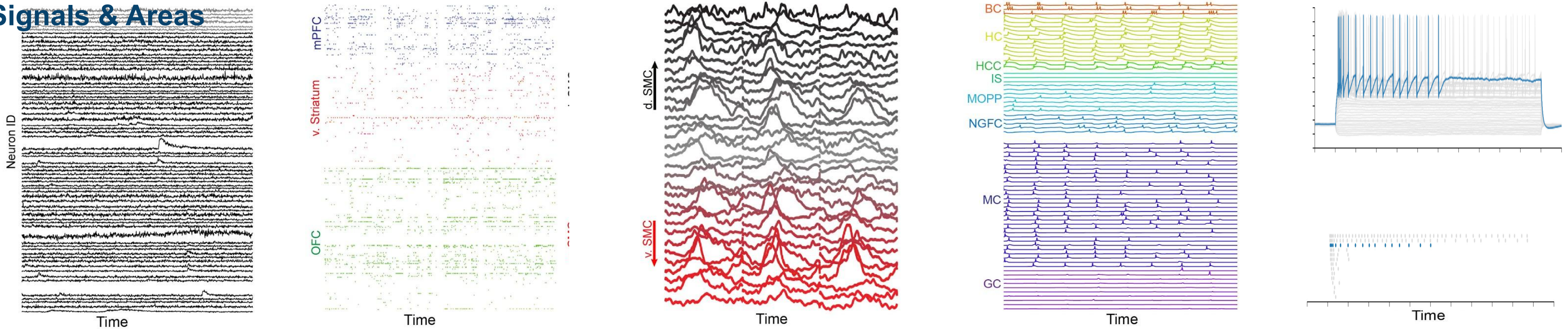
Species & Task



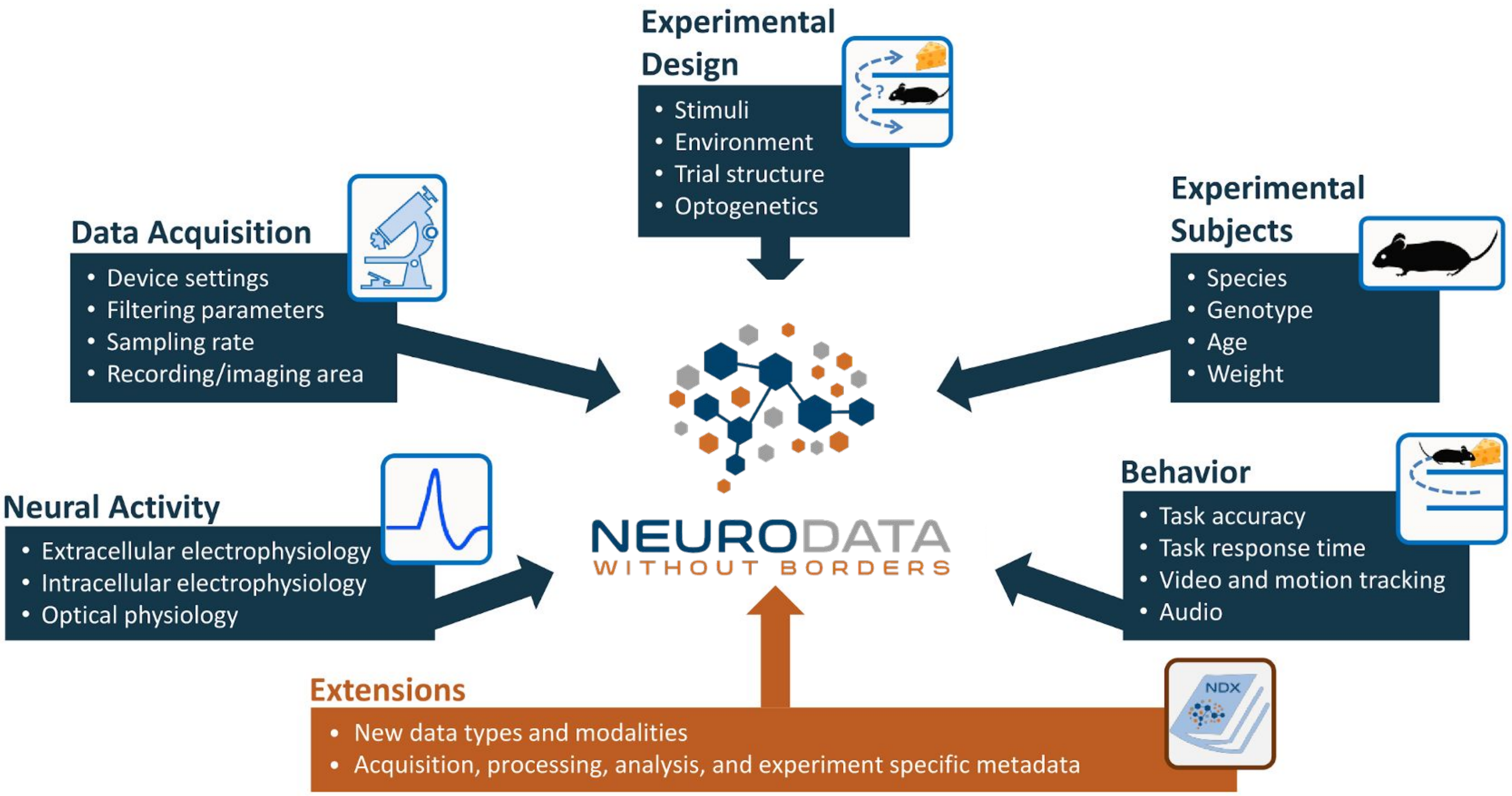
Modalities & Devices



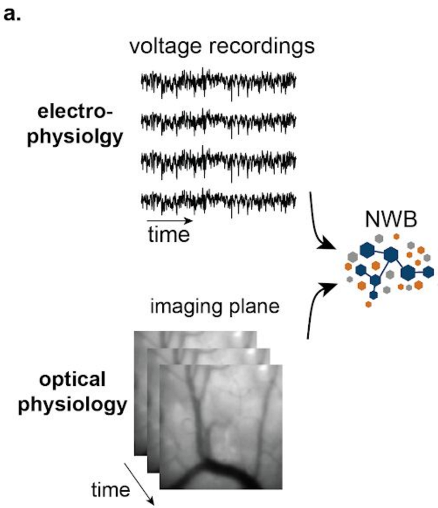
Signals & Areas



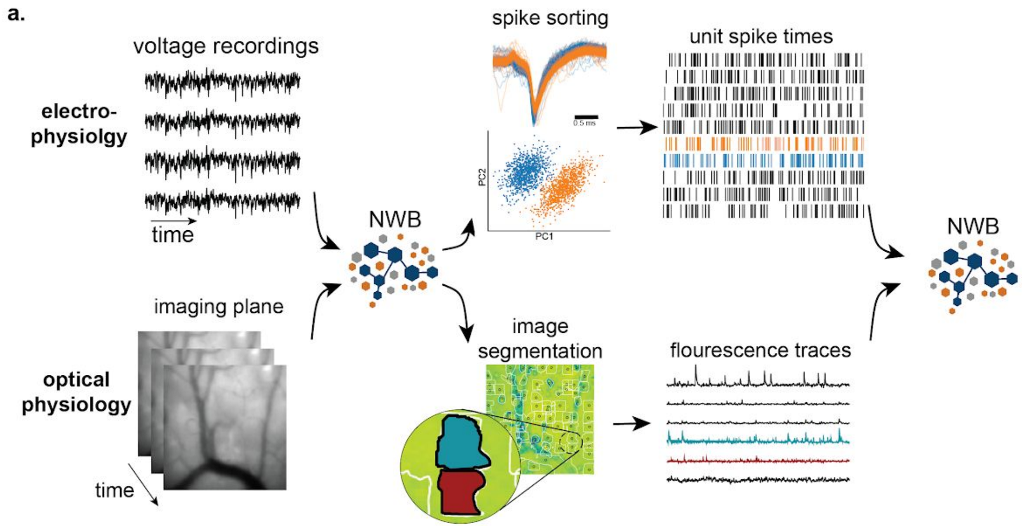
NWB defines a unified data format for neurophysiology



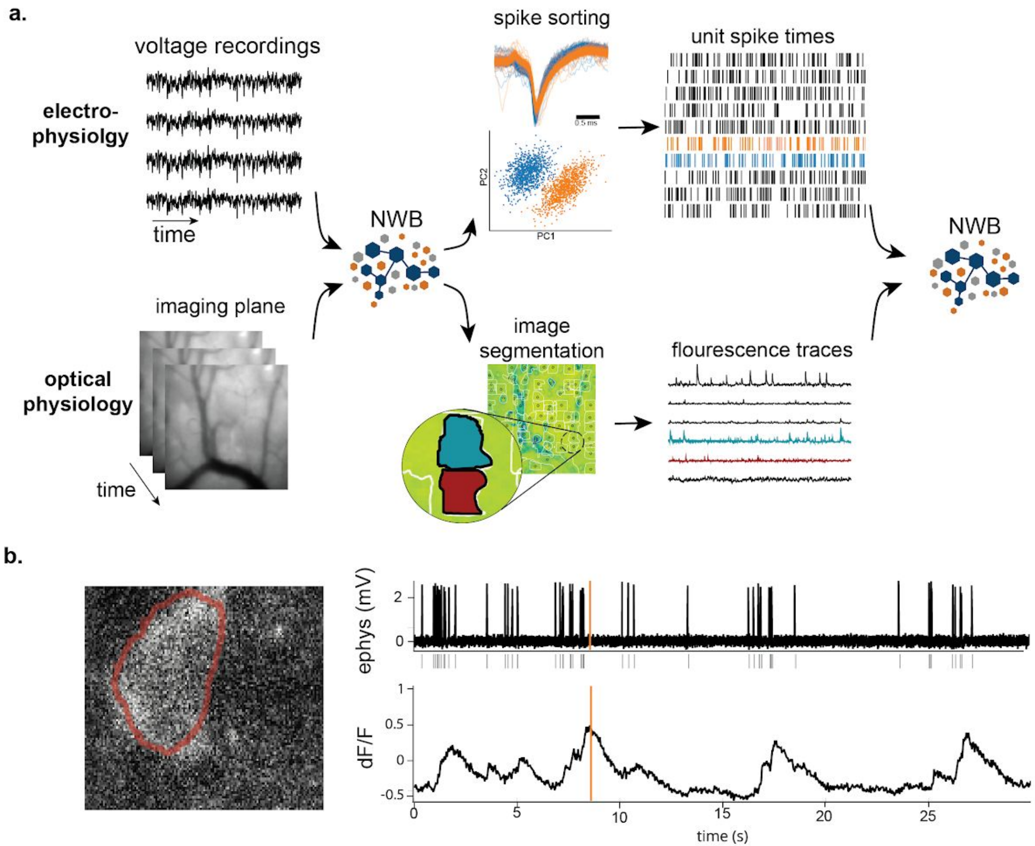
NWB enables unified storage of multimodal raw and processed data



NWB enables unified storage of multimodal raw and processed data



NWB enables unified storage of multimodal raw and processed data



NWB organizes your neurophysiology data and metadata

NEURODATA WITHOUT BORDERS

- general
  - subject `(neurodata_type: Subject)`
    - species : Mus musculus
  - experimenter :
    - Ryan Ly
    - Ben Dichter
  - extracellular_ephys
    - electrodes :

id	x	y	z	location	…
0	1.1	0.5	0.1	SSp-II1	…
1	1.2	0.5	0.3	SSp-II1	…
2	1.3	0.5	0.5	SSp-II2/3	…

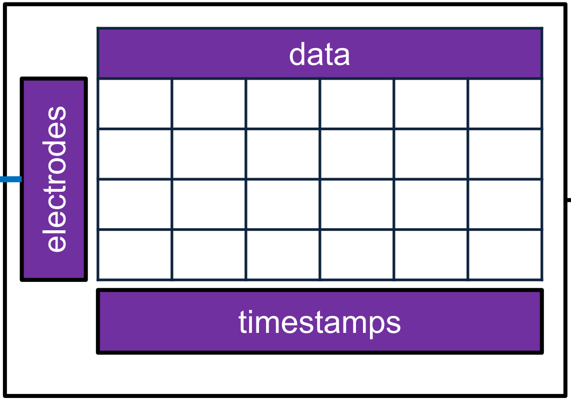
- acquisition
- processing
- units
- intervals

…

NWB organizes your neurophysiology data and metadata

NEURODATA WITHOUT BORDERS

☐ general
☐ acquisition
    ☐ ElectricalSeries `(neurodata_type: ElectricalSeries)`



- Description
- Unit of measurement
- Conversion factor
- Channel conversion factor
- Offset
- Sampling rate

☐ processing
☐ units
☐ intervals
…

NWB organizes your neurophysiology data and metadata

NEURODATA WITHOUT BORDERS

☐ general
☐ acquisition
☐ processing
    ☐ units (neurodata_type: Units)

id	spike_times	waveform_mean	waveform_sd	electrodes	…
0	[0.03, 0.14]	[0.01, 0.03, …]	[0.04, 0.06, …]	[0, 1, 2]	…
1	[0.6, 0.71, 0.82]	[0.12, 0.13, …]	[0.06, 0.08, …]	[3, 5, 6, 8]	…
2	…	…	…	…	…

☐ intervals
…

NWB provides structured neurodata types for most of neurophysiology data

Neurophysiology Data

- Extracellular Electrophysiology
  - Electrical series
  - Units (spikes)
  - Spike event series ...
- Intracellular Electrophysiology
  - Voltage clamp recording
  - Current clamp recording
  - Sweep table ...
- Optical Physiology
  - Two-photon image series
  - ROIs
  - DF/F ...
- Behavior
  - Position
  - Eye position
  - Pupil diameter ...
- Stimuli
  - Images
  - Videos
  - Stimulation
- Experimental Metadata
  - Trials
  - Experimenters
  - Devices ...

NWB Extensions (NDX)

- NWB supports the **formal extension** of the data standard to define new data types and metadata
- Extensions are immediately usable and shared directly with the data
- NDX Catalog: Catalog and tools for developing and publishing extensions
- Extension proposals: Formal process for integration of extensions with NWB

NDX Template

Neurodata Extensions Catalog

HDMF DocUtils

NDX Catalog

Acquisition types

- ndx-miniscope
- ndx-simulation-output
- ndx-ecog
- ndx-fret
- ndx-nirs
- ndx-photometry
- ndx-ecg
- ndx-photostim

Task types

- ndx-hierarchical-behavioral-data
- ndx-sound
- ndx-odor-metadata
- ndx-whisk

Processing

- ndx-extract

Lab-specific

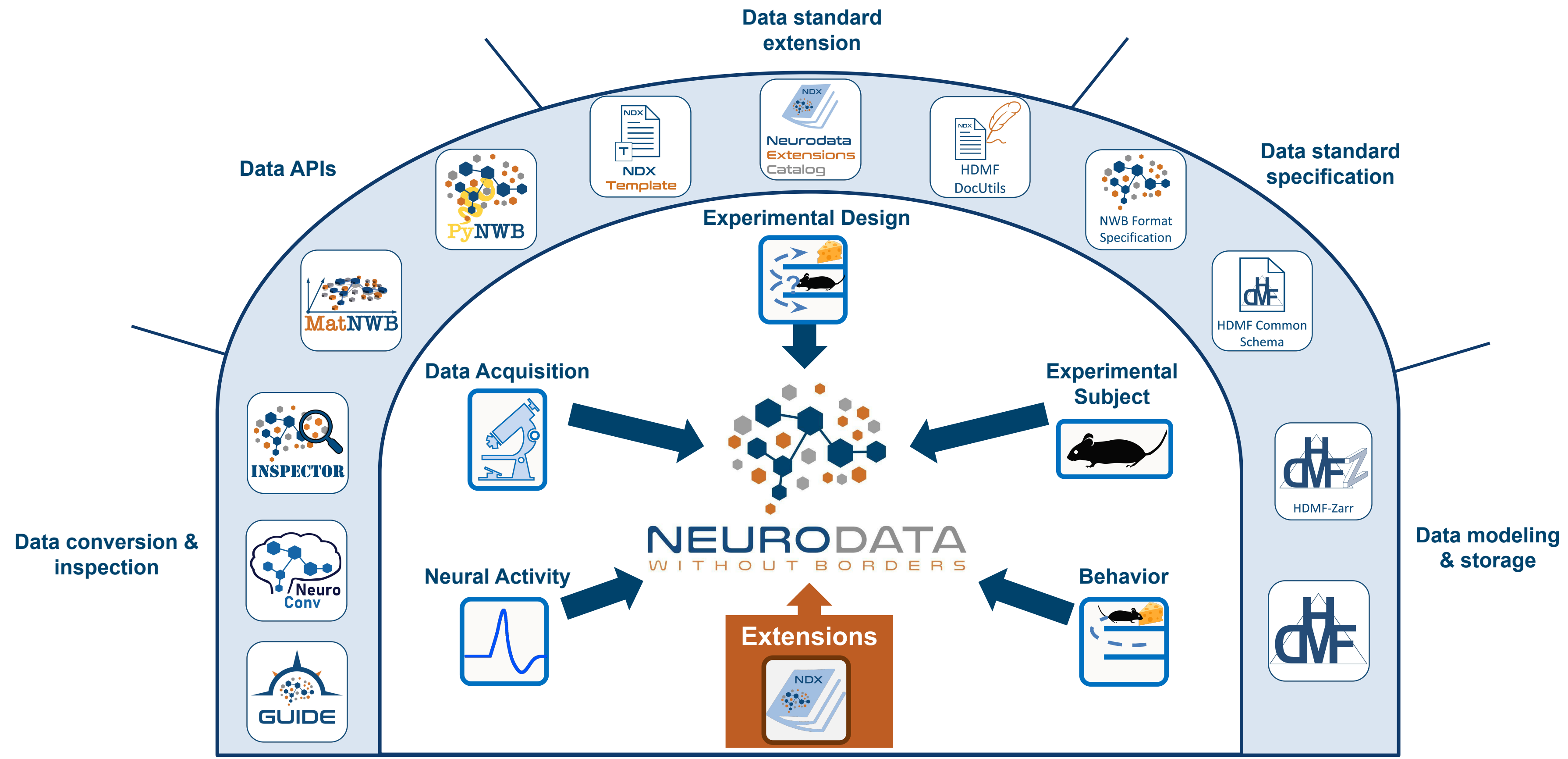
- ndx-franklab-novela

NWB Data Storage

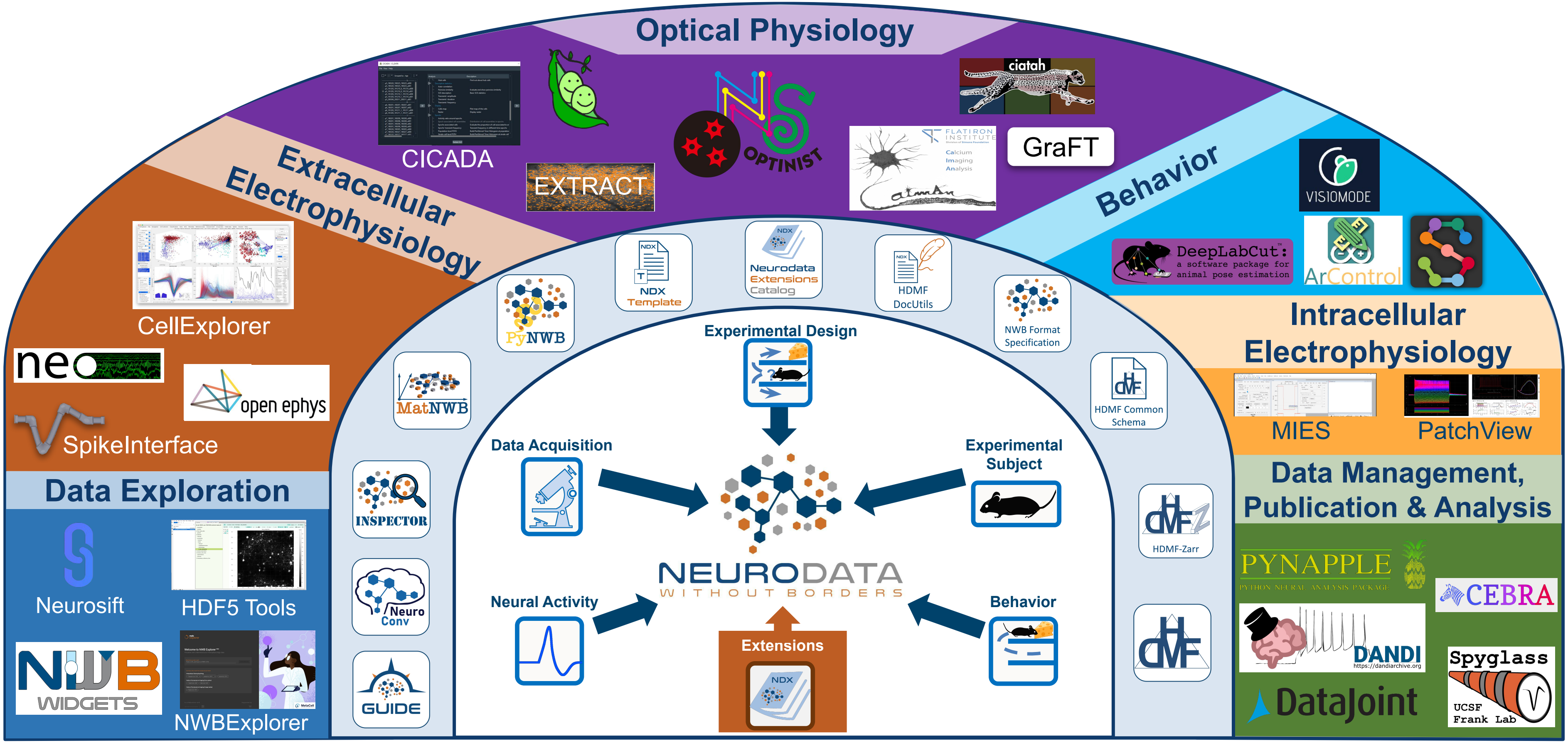
- NWB uses HDF5 as its main file storage backend:
  - Annotate complex data using Groups (like folders) Datasets (arrays) and Attributes in a single file
  - Supports advanced I/O features (e.g., lazy load, chunking, compression, links and references, parallel I/O, etc.)
  - Supported across platforms and languages (Matlab, Python, C/C++, Julia, R, Javascript…)
  - Targets long-term support
- NWB also supports Zarr
- We are working with the community to explore and support integration with other data management and storage technologies (e.g., DataJoint)

HDMF-Zarr

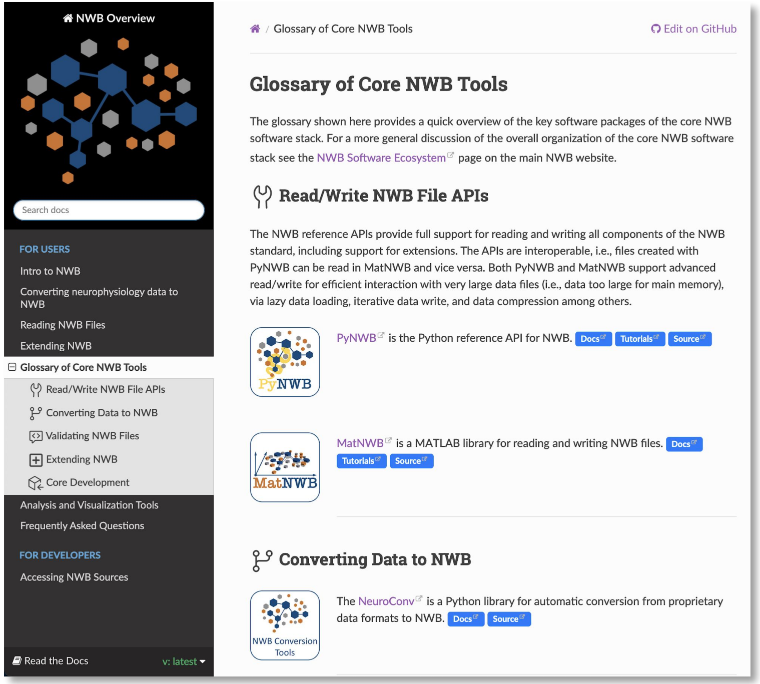
An ecosystem for neuroscience data standardization

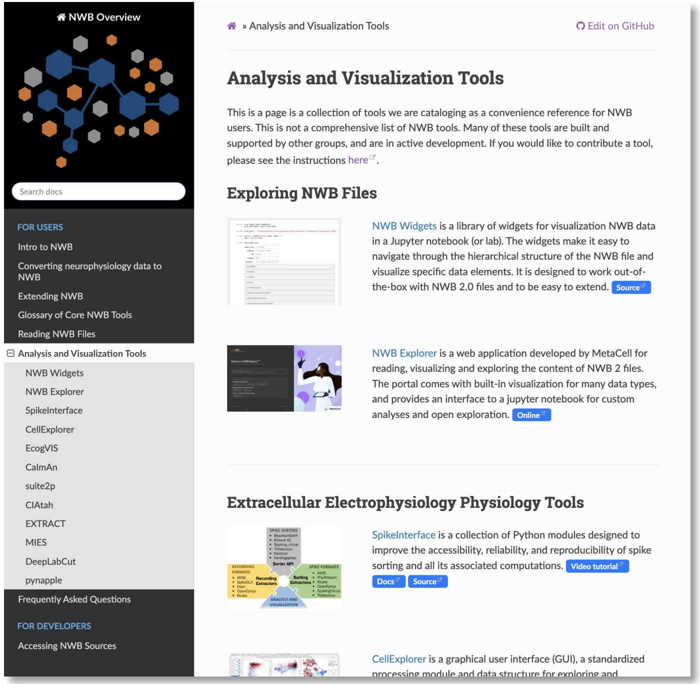


A unified data standard and software ecosystem for neurophysiology

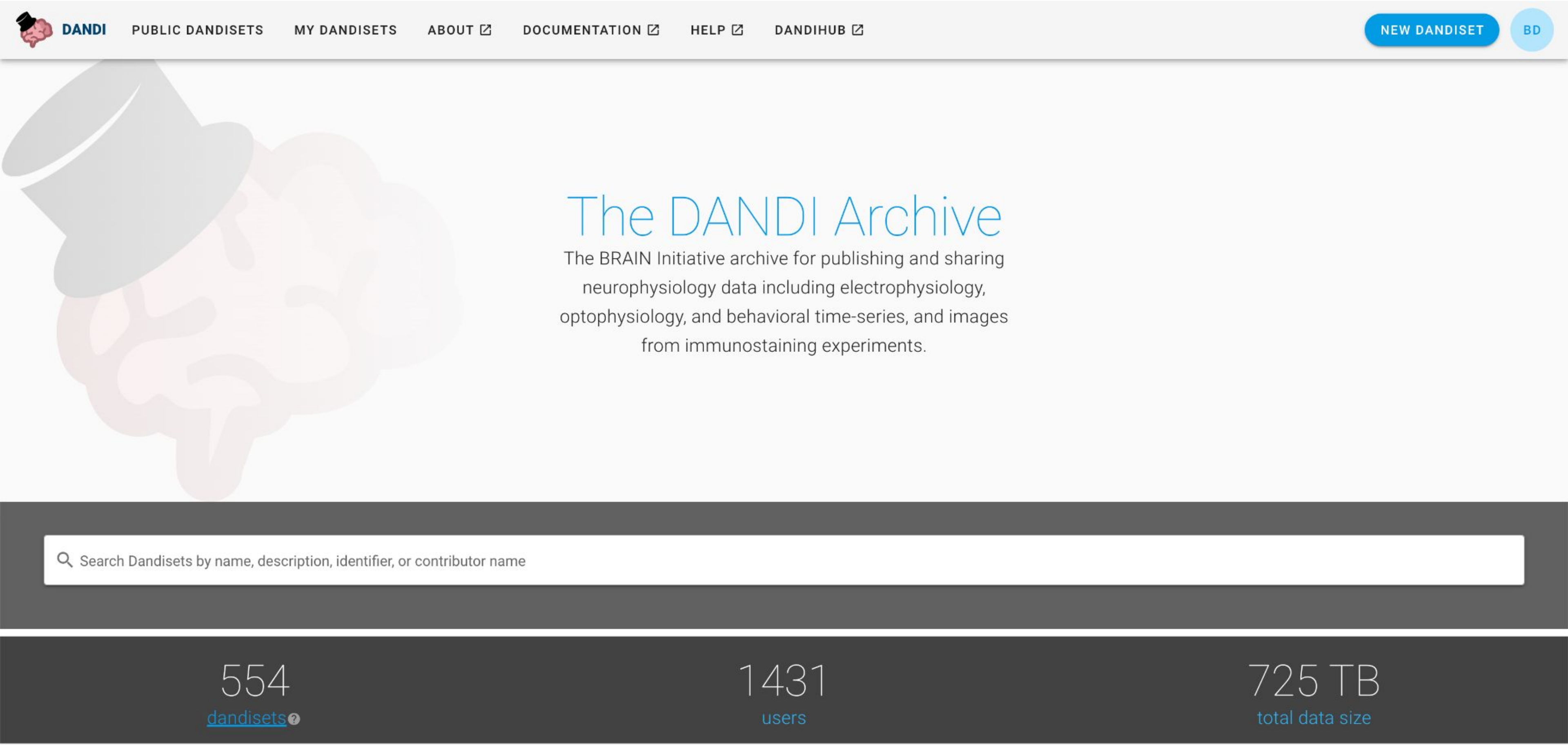


Tools Overview: https://nwb-overview.readthedocs.io





An Archive and Collaboration Space for Neurophysiology Projects



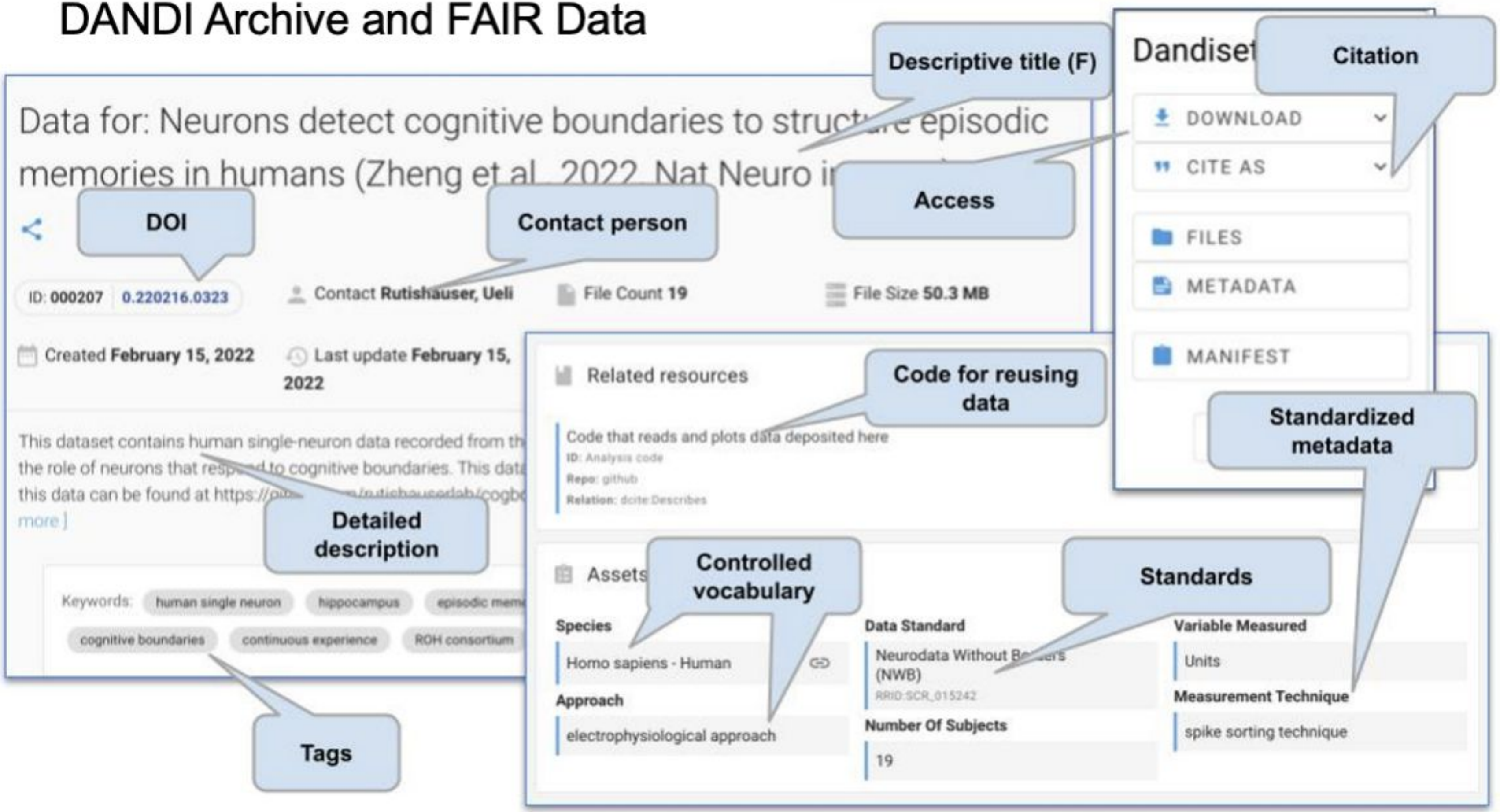
Contributing data to the DANDI Archive

- Stores neurophysiology data and associated behavioral data in NWB format
  - Validates NWB, checks for best practices, automatically extracts metadata
- Upload 100s of TBs for **FREE**
  - Talk to us if ≥10 TB
  - 5 TB per file max
  - Can handle all raw data for most experiments
  - AWS Open Data Sponsorship Program
- Open licenses: CC-0 or CC-BY
- Can upload data in embargo mode
- Create DOIs by “publishing” and re-publishing

Link to Terms

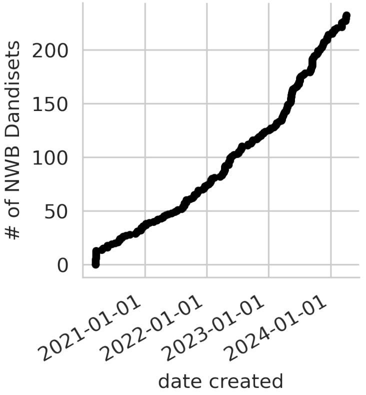
Link to Policies

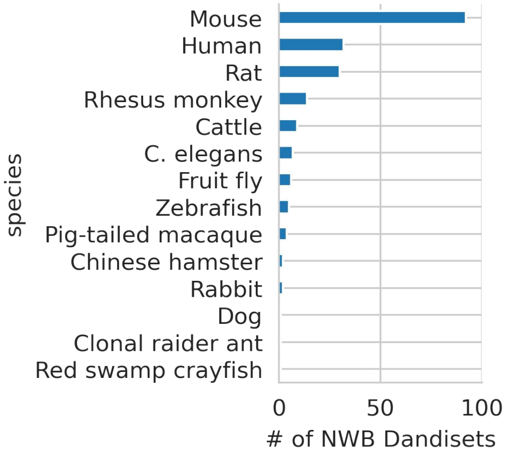
DANDI Archive and FAIR Data

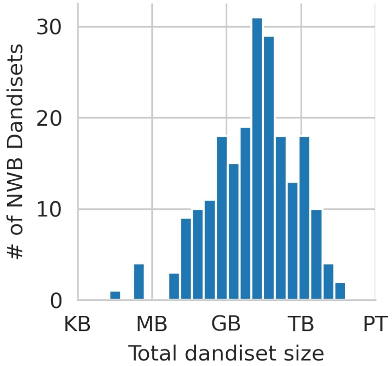


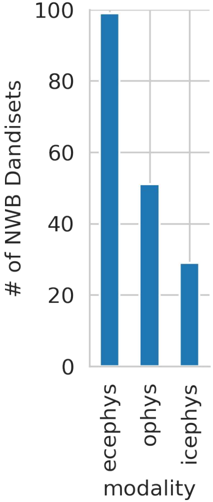
A Guide to the BRAIN Initiative Cell Census Network Data Ecosystem. (In press)

Neurophysiology datasets on DANDI







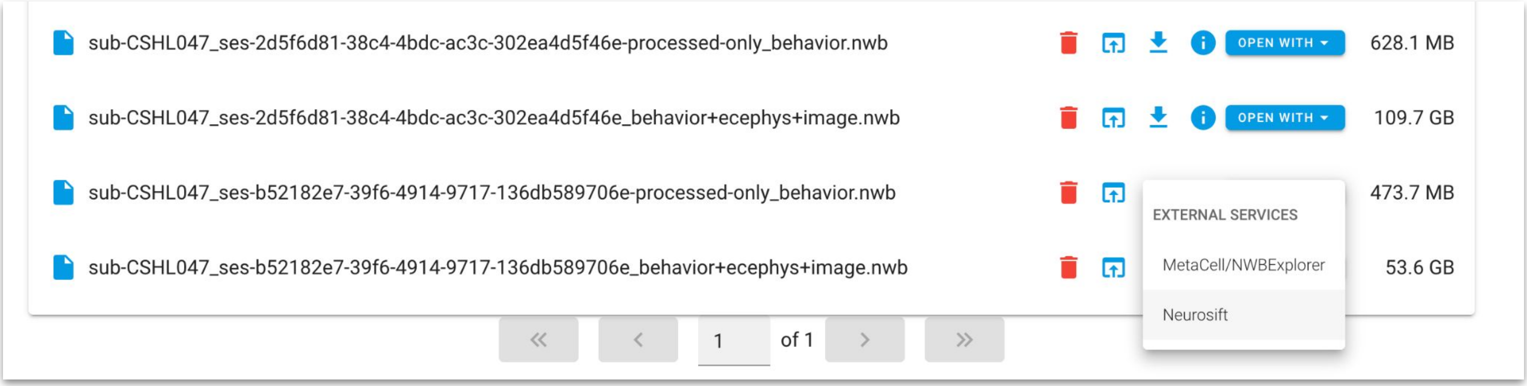


Highlights:

- **IBL - Brain Wide Map** (000409): 33.6 TB dataset from the International Brain Laboratory including Neuropixel electrophysiology, eye tracking, and behavioral monitoring for a visual decision making task
- **MICrONS Two Photon Functional Imaging** (000402): 1.3 TB of optical physiology data for a visual task. The functional data was co-registered with electron microscopy (EM) data.
- **Allen Institute OpenScope** (000036, 000037, 000488, 000535): Several studies from the Allen Institute Open Scope program that use optical physiology recordings of the visual cortex.

Neurosift: Explore NWB data in the browser

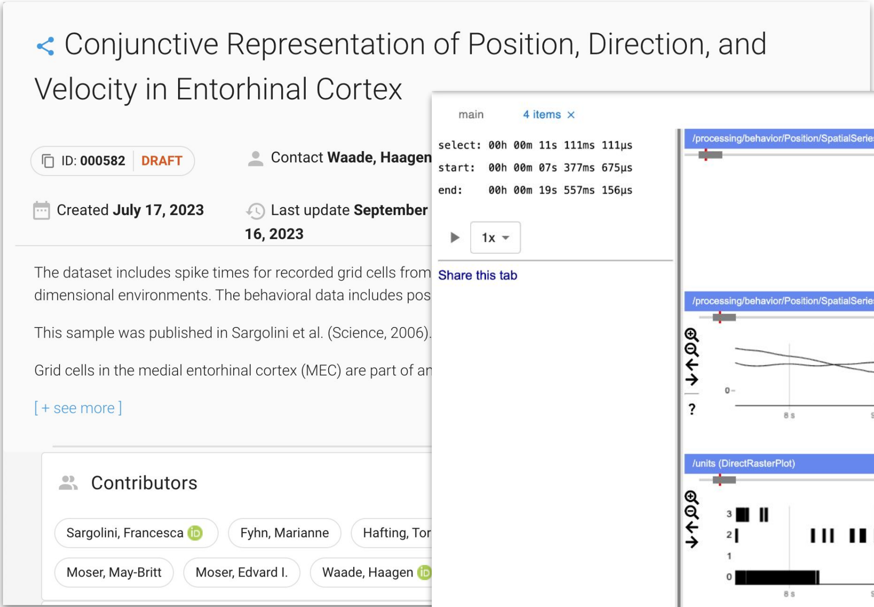
- Intelligent, interactive visualizations of NWB data
- Runs in browser- no need to install anything or run external services
- Streams data directly from the archive
- Integrated into the DANDI Archive website

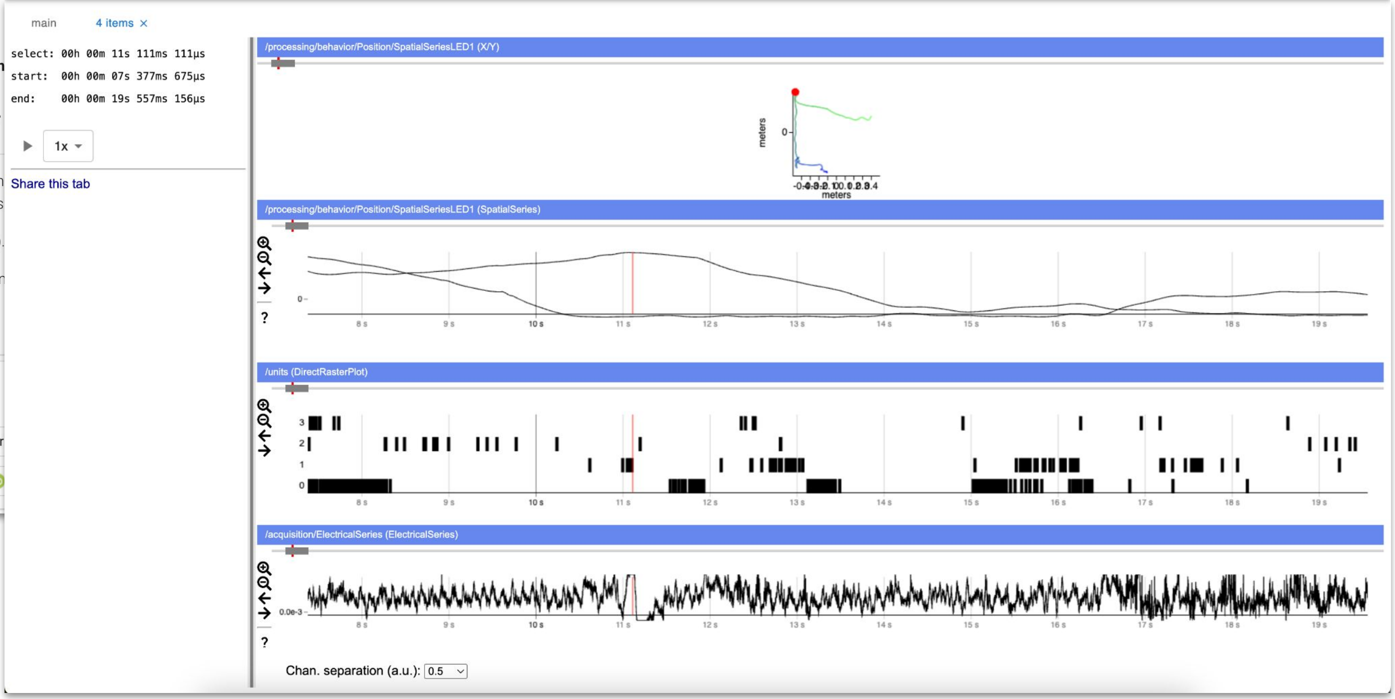


Jeremy Magland, Flatiron

Position tracking and trace views

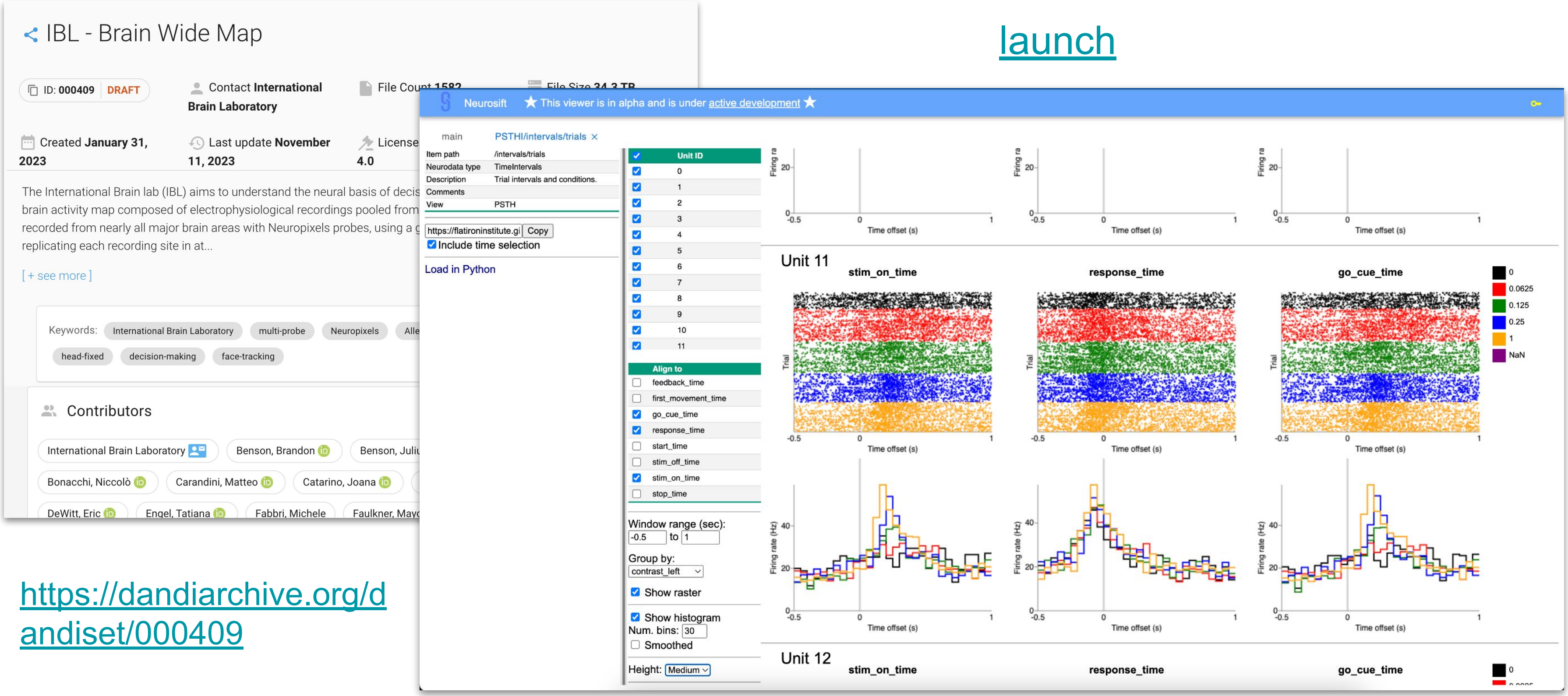
[launch](#)





https://dandiarchive.org/dandiset/000582

Peri-stimulus time histogram

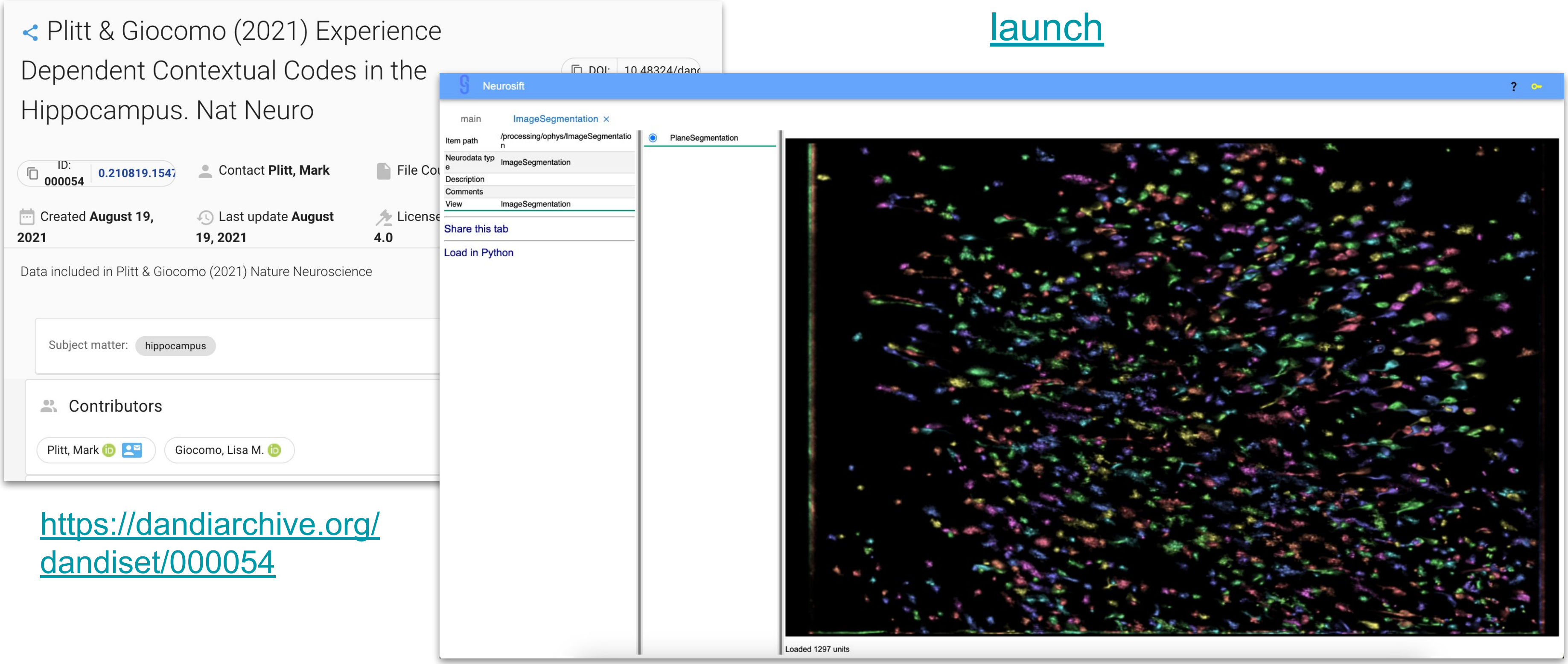


launch

https://dandiarchive.org/dandiset/000409

Optical physiology segmentation

[launch](#)



https://dandiarchive.org/dandiset/000054

Analyzing data on DANDI Hub

- DANDI Hub: a FREE cloud compute resource (https://hub.dandiarchive.org)
- Analyze datasets in Python, MATLAB, Julia, R, …
- Fast access to stream and download DANDI data
- Different server options, including GPU
- Share analysis notebooks with the community

Publications reusing public NWB data

Publications:

1. Gala et al. 2021. Consistent cross-modal identification of cortical neurons with coupled autoencoders. Nature Computational Science.
    a. Used DANDI:000020
2. Keshtkaran et al. 2022. A large-scale neural network training framework for generalized estimation of single-trial population dynamics. Nature Methods.
    a. Used DANDI:000070, DANDI:000128, DANDI:000129
3. Schneider et al. 2023. Learnable latent embeddings for joint behavioural and neural analysis. Nature.
    a. Used DANDI:000127 and several other public datasets
4. Easthope et al. 2023. Cortical control of posture in fine motor skills: evidence from inter-utterance rest position. Frontiers in Human Neuroscience.
    a. Used NWB dataset shared on Figshare
5. Cai et al. 2023. FIOLA: an accelerated pipeline for fluorescence imaging online analysis. Nature Methods.
    a. Used DANDI:000054
6. Patel et al. 2023. High-performance neural population dynamics modeling enabled by scalable computational infrastructure. Journal of Open Source Software.
    a. Used DANDI:000128
7. Arbabi et al. 2023. Investigating microglia-neuron crosstalk by characterizing microglial contamination in human and mouse patch-seq datasets. iScience.
    a. Used DANDI:000020, DANDI:000209

Preprints:

8. Ye et al. 2021. Representation learning for neural population activity with Neural Data Transformers.
    a. Used DANDI:000070
9. Nguyen et al. 2023. Fast Temporal Wavelet Graph Neural Networks. arXiv.
    a. Used DANDI:000055
10. Azabou et al. 2023. A Unified, Scalable Framework for Neural Population Decoding.
    a. Used DANDI:000688, DANDI:000070, DANDI:000128, DANDI:000129, and other public datasets
11. Bernaerts et al. 2023. Combined statistical-mechanistic modeling links ion channel genes to physiology of cortical neuron types.
    a. Used DANDI:000008
12. Perkins et al. 2023. Simple decoding of behavior from a complicated neural manifold.
    a. Used DANDI:000127, DANDI:000128, DANDI:000129, DANDI:000130, DANDI:000138, DANDI:000139, DANDI:000140
13. Vetter et al. 2023. Generating realistic neurophysiological time series with denoising diffusion probabilistic models.
    a. Used DANDI:000055 and other public datasets

Styles of data reuse

1. **Showcase.** Demonstrate new analysis or software on various existing datasets
2. **Pilot**. Try an analysis on existing data before designing your own experiment
3. **Follow-up**. Explore questions that were left unanswered in the initial study
4. **Confirmative**. Find similar results in data from another lab
5. **Contrastive**. Show that your results are particular to your experiment preparation
6. **Aggregative**. Gain statistical power by combining data across labs
7. **Educational**. Train and inspire the next generation of scientists using real data

Questions?

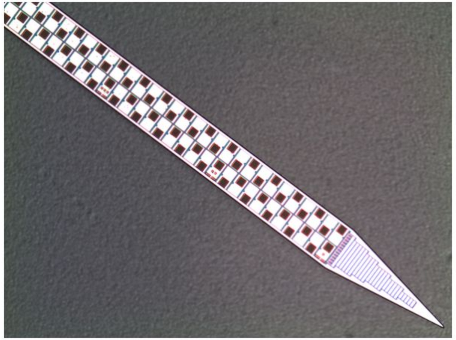
Schedule

10:15 - 10:45	Overview of NWB and DANDI
10:45 - 11:00	Q&A
11:00 - 12:15	NWB GUIDE (Interactive Tutorial) ● Easily convert data ● Inspect NWB File ● Explore NWB File
12:20	Group photo
12:30-13:30	Lunch
13:30 - 14:30	Advanced NWB conversion
14:30 - 15:30	Analysis tools in the NWB ecosystem

Converting to NWB

Bottleneck: Converting to NWB

- Wrangling diverse data formats into one standard is still a bottleneck for many labs
- Standardizing neurophysiology data is challenging due to:
    - **Diversity** - many different proprietary formats
    - **Complexity** - experiments often use multiple data sour
      (each stream in a unique format)
    - **Volume** - terabyte (TB) scale and growing



Neuropixels 1 probe
neuropixels.org

Converting data to NWB

GUIDE

No-code automated conversion via graphical interface app (some input data formats, beta)

NeuroConv

Low-code automated conversion for common input data formats

PyNWB

MatNWB

Custom code using NWB APIs

Automated ⟷ **Flexible**

NWB GUIDE hands-on tutorial

To submit any feedback on the tutorial or your experience with the GUIDE, feel free to open an issue on our GitHub page.

If you do not see your data format listed on the Data Formats page or our support table , please submit a request using this form.

Schedule

10:15 - 10:45	Overview of NWB and DANDI
10:45 - 11:00	Q&A
11:00 - 12:15	Interactive Tutorials

- ➢ Easily convert data with NWB Guide
- ➢ File validation (Inspector) and exploration (Neurosift) within NWB GUIDE

12:20	Group photo
12:30-13:30	Lunch
13:30 - 14:30	Advanced NWB conversion
14:30 - 15:30	Analysis tools in the NWB ecosystem

Hands-on session: Convert custom data using PyNWB

- You can use a Dandi hub instance or your local machine
- Navigate to https://dandiarchive.org/
- Start a Dandihub instance (Tiny or Base should be enough)
- git clone https://github.com/catalystneuro/stratneuro_2024
- Open folder stratneuro_2024/tutorial_pynwb
- Open **nwb_conversion_with_pynwb.ipynb**

Advanced NWB conversion with PyNWB and Neuroconv

NWB provides a robust and well designed schema for storage of multimodality neurophysiology data, and GUIDE can provide a direct way to convert simple experiments using the most popular acquisition systems, but:

- Some experiments might require custom data conversion
- Multimodality might require custom synchronization of signals
- NWB is flexible enough to allow for modality extensions
- The NWB schema might be a bit difficult to use for the first time

Converting data to NWB

GUIDE

No-code automated conversion via graphical interface app

NeuroConv

Low-code automated conversion for common input data formats

PyNWB

MatNWB

Custom code using NWB APIs

Automated ⟷ **Flexible**

NeuroConv

Electrophysiology

Optical physiology

Behavior

```python
from datetime import datetime
from dateutil import tz
from pathlib import Path
from neuroconv.converters import SpikeGLXConverterPipe

converter =
SpikeGLXConverterPipe(folder_path="spikeglx/Noise4Sam_g0")

metadata = converter.get_metadata()
session_start_time =
metadata["NWBFile"]["session_start_time"].replace(tzinfo=tz.gettz("
US/Pacific"))
metadata["NWBFile"].update(session_start_time=session_start_time)

converter.run_conversion(nwbfile_path="my_spikeglx_session.nwb",
metadata=metadata)
```

A few lines of code…

Automated NWB conversions with NeuroConv

- Convert data to NWB with minimal time and effort
- Supports 44+ common neurophys data formats
- Automatically extracts format-specific metadata
- Seamlessly integrated data engineering
    - Converts on the TB scale
    - Can reduce file size by ~35%
    - Optimizes chunking for streaming from DANDI

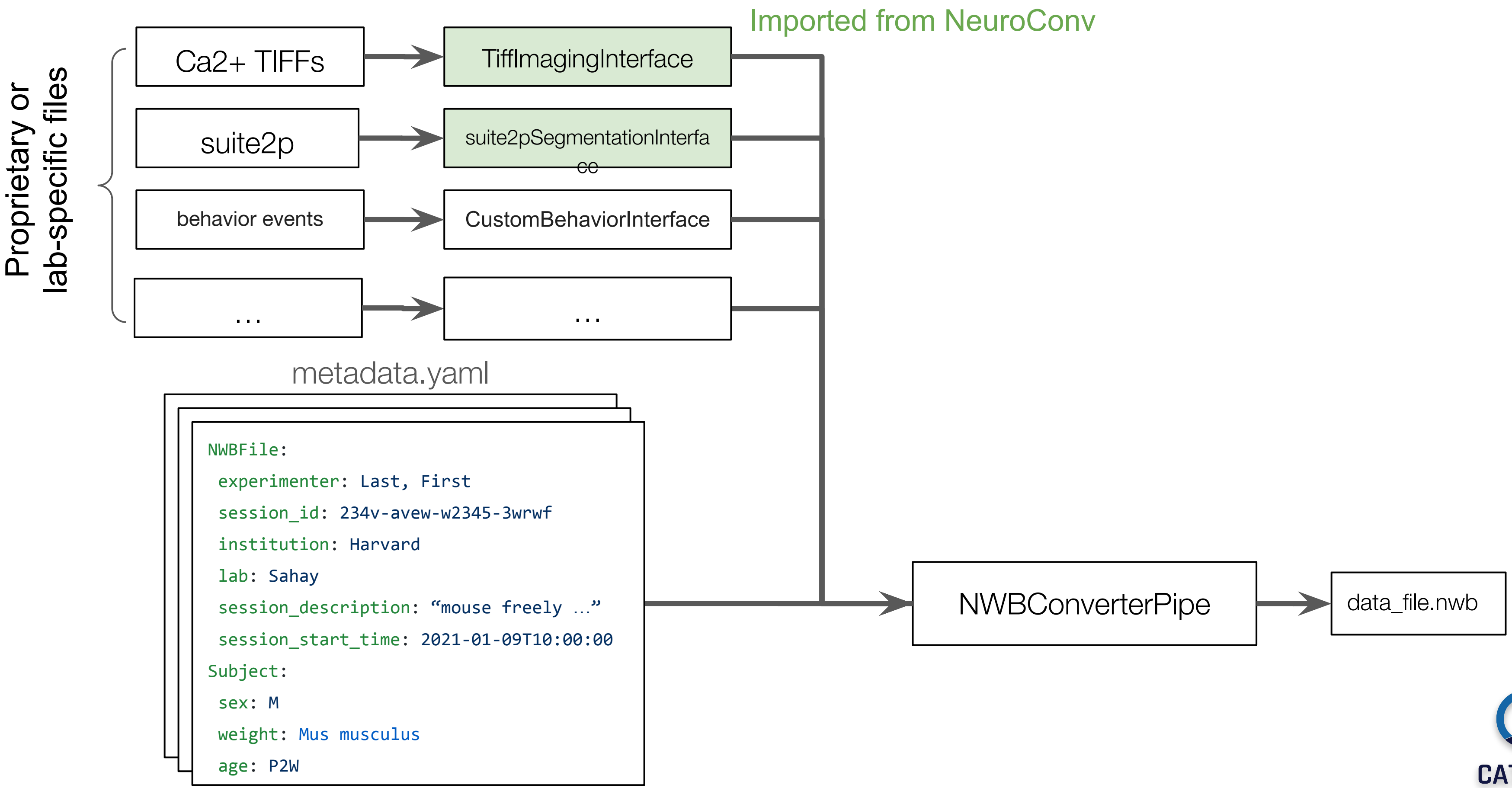
NeuroConv

Simple code to convert a SpikeGLX recording to NWB

```
from datetime import datetime
from dateutil import tz
from pathlib import Path
from neuroconv.converters import SpikeGLXConverterPipe

# Create a converter object with the path to your SpikeGLX data
converter = SpikeGLXConverterPipe(folder_path="spikeglx/Noise4Sam_g0")

# For data provenance, we add time zone information of the acquisition to the metadata
metadata = converter.get_metadata()
session_start_time = metadata["NWBFile"]["session_start_time"].replace(tzinfo=tz.gettz("US/Pacific"))
metadata["NWBFile"].update(session_start_time=session_start_time)

# Choose a path for saving the NWB file and run the conversion
converter.run_conversion(nwbfile_path="my_spikeglx_session.nwb", metadata=metadata)
```

Demo code for all supported data types in [NeuroConv Conversion Gallery]

Strategy: Modularize by data stream



CATALYST NEURO

NeuroConv supports 44 common input formats

Extracellular electrophysiology

Recording

- AlphaOmega
- Axona
- Biocam
- Blackrock
- EDF
- Intan
- MaxOne
- MCSRaw
- MEArec
- Neuralynx
- NeuroScope
- OpenEphys
- Plexon
- Spike2
- Spikegadgets
- SpikeGLX
- TDT

Sorting

- Blackrock
- Cell Explorer
- KiloSort
- Neuralynx
- NeuroScope
- Phy
- Plexon

Intracellular electrophysiology

- ABF

Optical physiology

Imaging

- Bruker
- HDF5
- Micro-Manager
- Miniscope
- Scanbox
- ScanImage
- Tiff

Segmentation

- Caiman
- CNMFE
- EXTRACT
- Suite2P

Behavior

- Audio
- DeepLabCut
- FicTrac
- LightningPose
- SLEAP
- Videos

Text for Trials Table

- CSV
- Excel

Use NWB APIs to write (and read) NWB data

- Most useful for converting custom data
- Also trial times, responses, and properties
- APIs support:
  - Iteratively reading and writing data
  - Custom chunking
  - Custom compression
  - Writing data in NWB extensions
- Lots of documentation and tutorials!

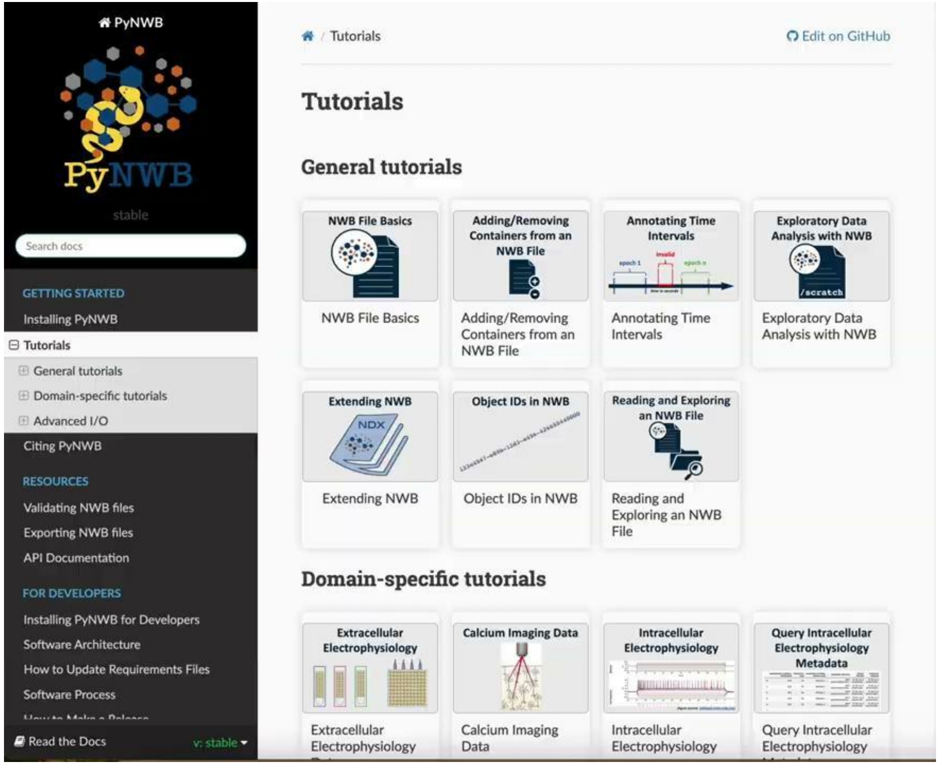
PyNWB

pynwb.readthedocs.io

MatNWB

github.com/NeurodataWithoutBorders/matnwb

PyNWB tutorials



MatNWB tutorials

Intro to MatNWB

Basic File Reading | a demo showcase for basic visualization from a DANDI dataset.

Extracellular Electrophysiology | YouTube walkthrough

Calcium Imaging | YouTube walkthrough

Intracellular Electrophysiology

Behavior

Optogenetics

Dynamic tables

Images

Advanced data write | YouTube walkthrough

Using Dynamically Loaded Filters

Remote read

Scratch Space

Hands-on session: Convert custom data using PyNWB

- You can use a Dandi hub instance or your local machine
- Navigate to https://dandiarchive.org/
- Start a Dandihub instance (Base should be enough)
- git clone https://github.com/catalystneuro/stratneuro_2024
- Open folder stratneuro_2024/tutorial_pynwb
- Open **nwb_conversion_with_pynwb.ipynb**

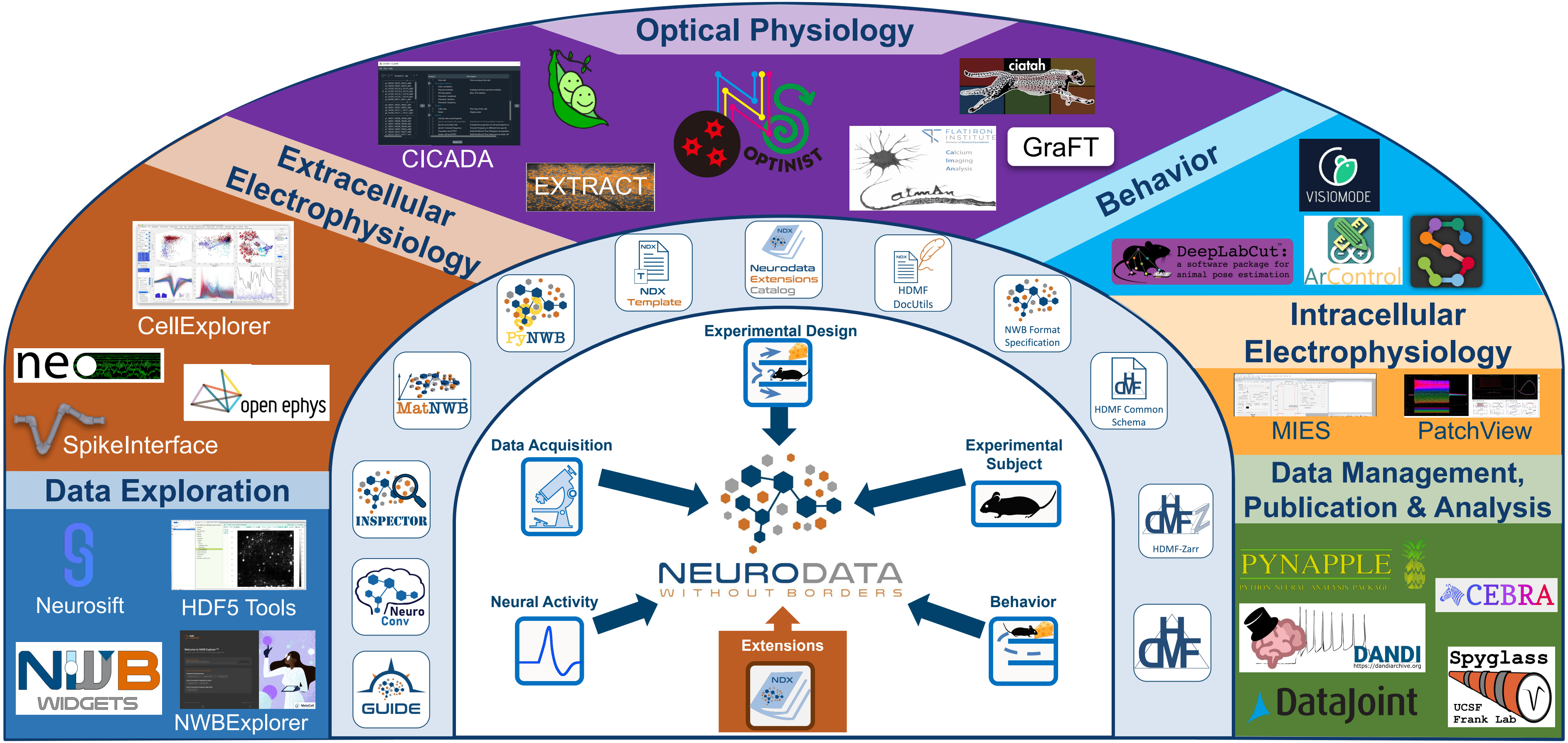
Schedule

10:15 - 10:45	Overview of NWB and DANDI
10:45 - 11:00	Q&A
11:00 - 12:15	Interactive Tutorials

➢ Easily convert data with NWB Guide
➢ File validation (Inspector) and exploration (Neurosift) within NWB GUIDE

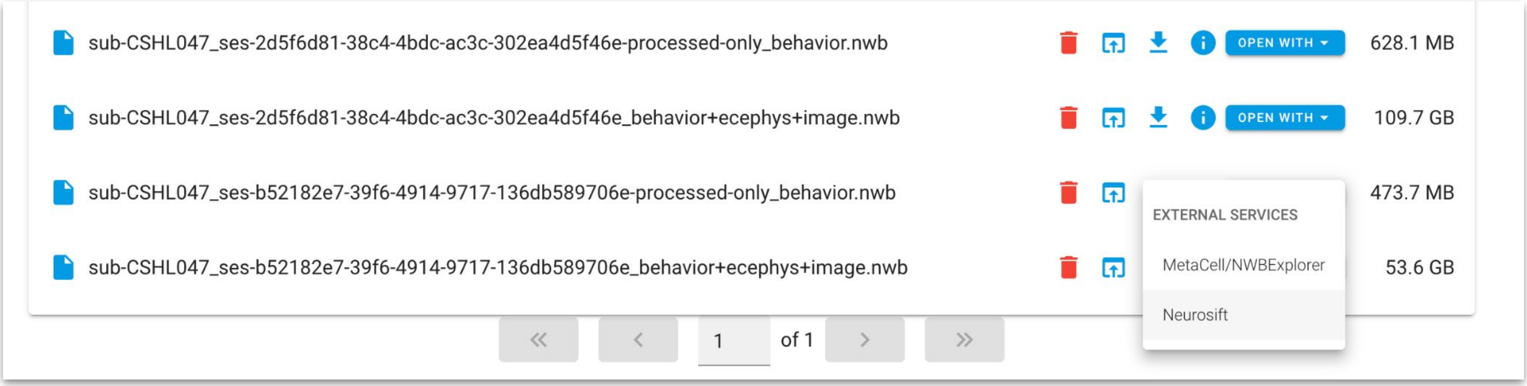
12:20	Group photo
12:30-13:30	Lunch
13:30 - 14:30	Advance NWB conversion
14:30 - 15:30	Analysis tools in the NWB ecosystem

Data Analysis within NWB ecosystem



Neurosift - Explore NWB data in the browser

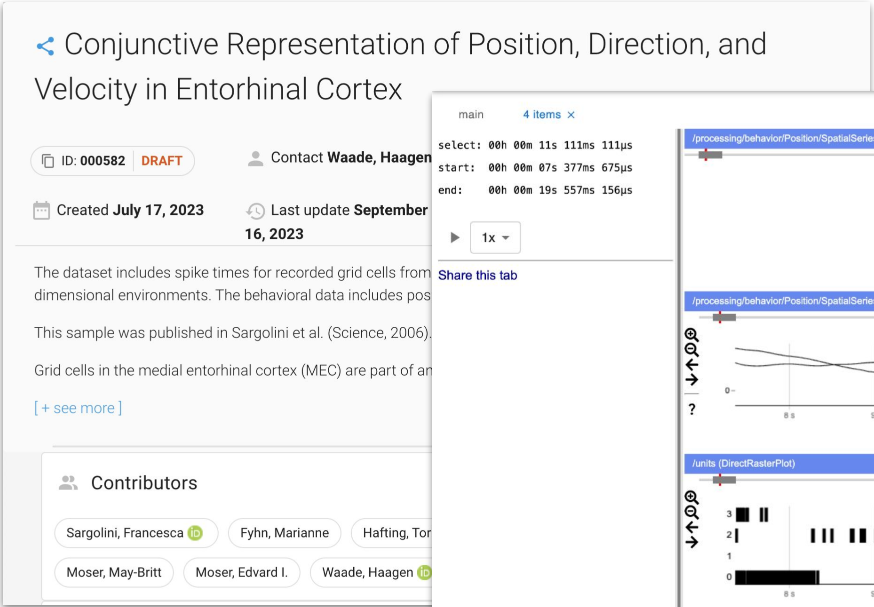
- Intelligent, interactive visualizations of NWB data
- Runs in browser- no need to install anything or run external services
- Streams data directly from the archive
- Integrated into the DANDI Archive website

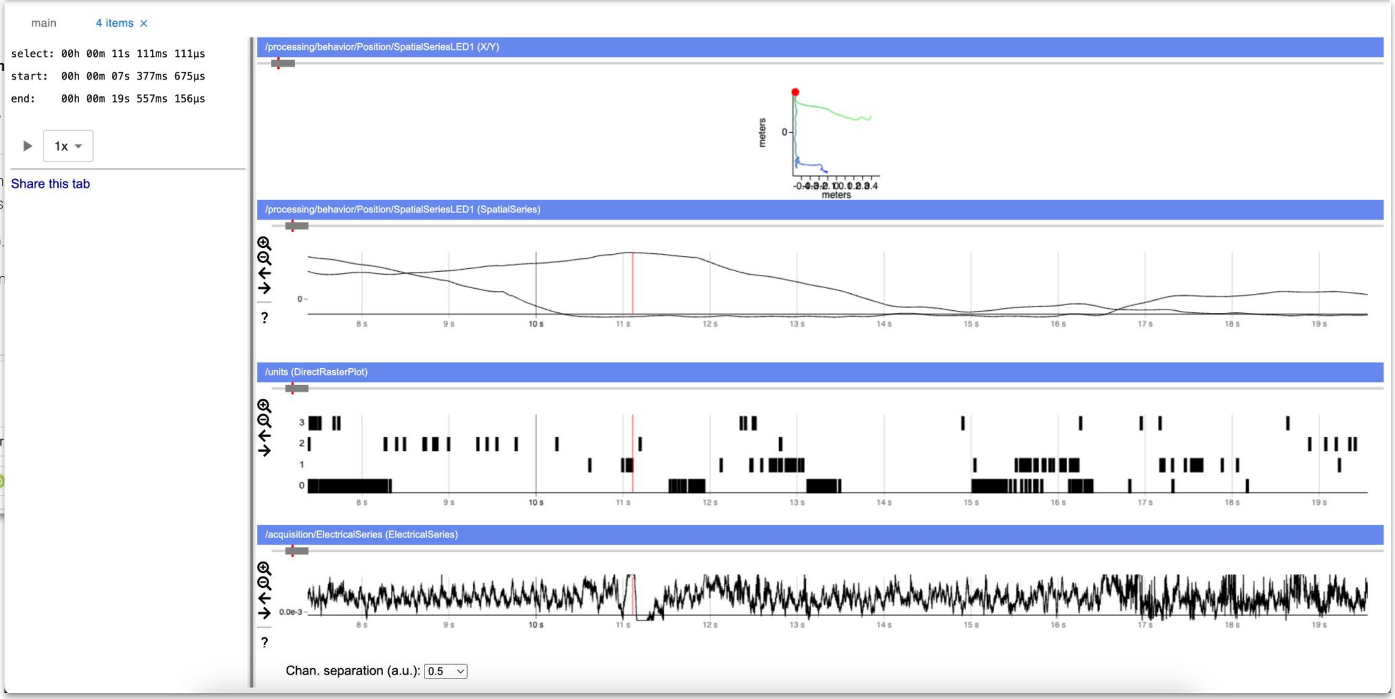


Jeremy Magland, Flatiron

Neurosift - Position tracking and trace views

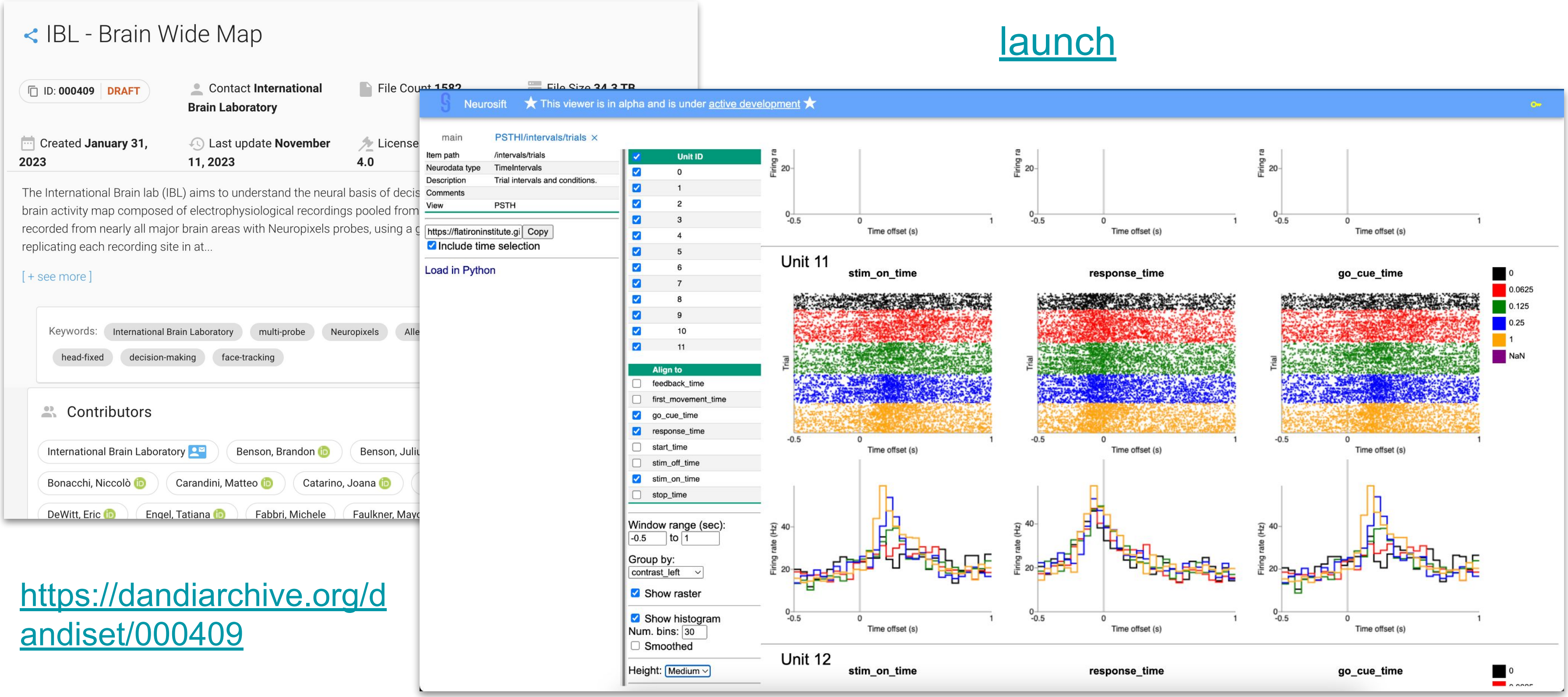
[launch]





https://dandiarchive.org/dandiset/000582

Neurosift - Peri-stimulus time histogram



[launch]

https://dandiarchive.org/dandiset/000409

Dendro - Analyse neurophys data in the cloud

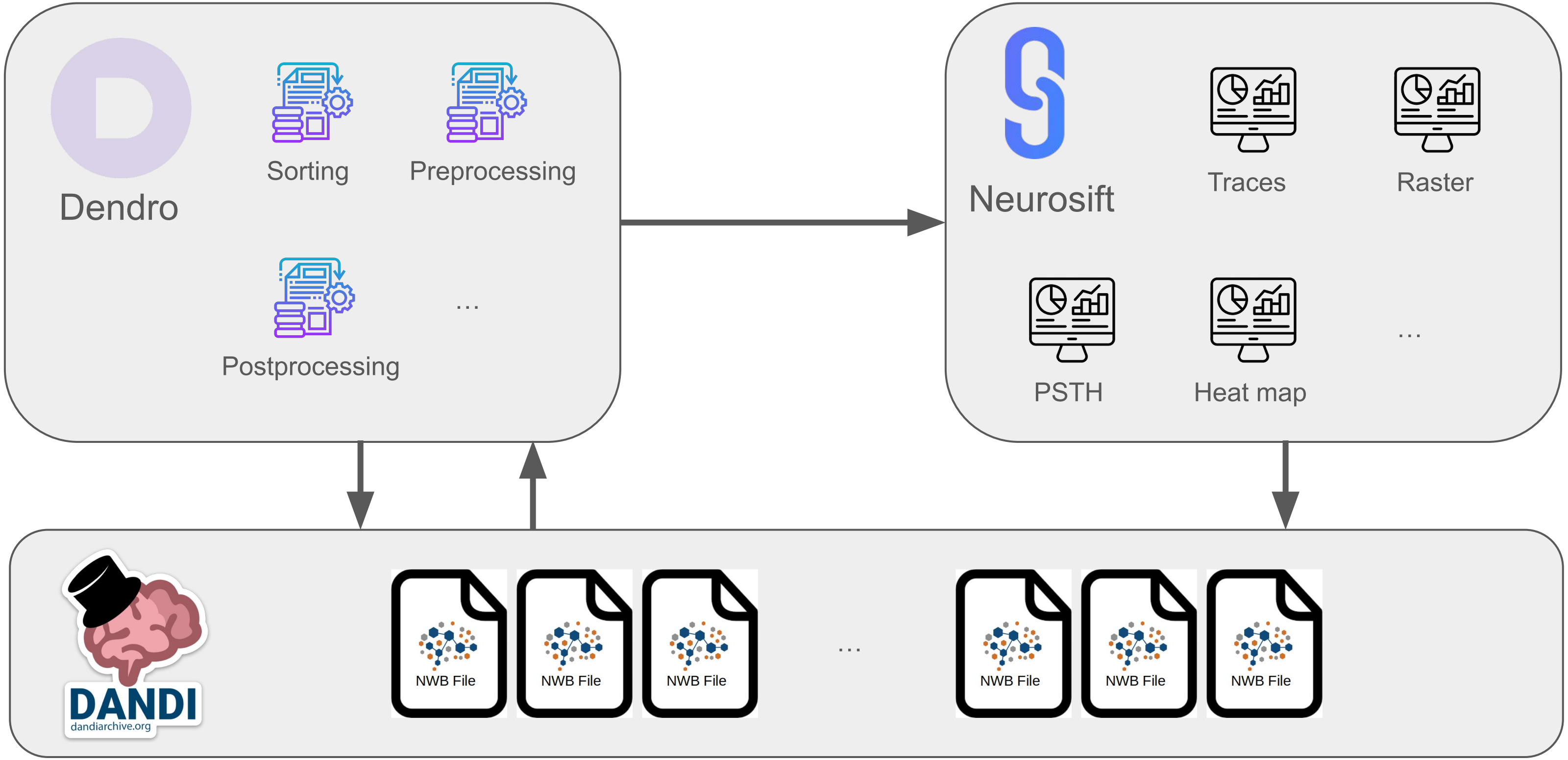
Dendro is an open source web application that aims at:

- Facilitating the usage of complex neurodata processing algorithms
- Promoting reusability of experimental data
- Promoting reproducibility of results with full data processing provenance

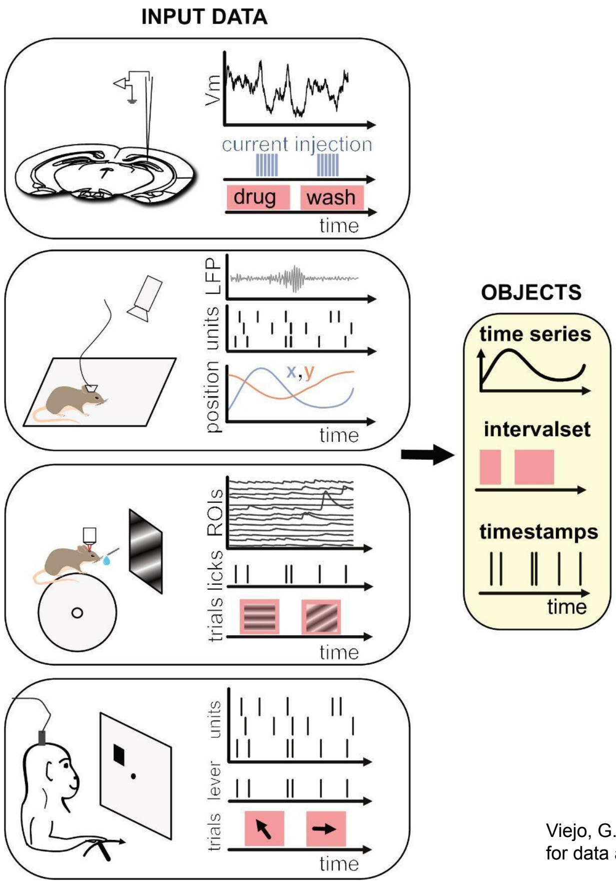
https://dendro.vercel.app

https://flatironinstitute.github.io/dendro-docs/

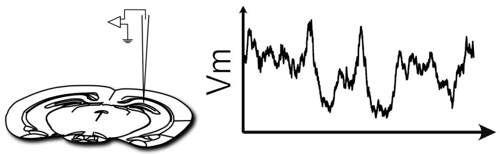
Cloud Ecosystem: NWB + DANDI + Dendro + Neurosift

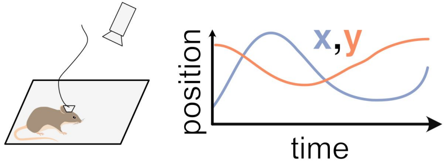


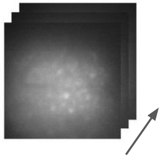
Pynapple



Viejo, G., Levenstein, D., Carrasco, S. S., Mehrotra, D., Mahallati, S., Vite, G. R., ... & Peyrache, A. (2023). Pynapple, a toolbox for data analysis in neuroscience. *eLife*, *12*, RP85786.







Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0           0.397043
1.0           1.55294
2.0           0.455892
3.0          -1.17359
4.0          -0.110113
...
95.0         -0.573408
96.0         -0.0110915
97.0         -1.58027
98.0          0.998846
99.0          0.542692
dtype: float64, shape: (100,)
```

TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:     columns = ['x', 'y'])

In [21]: tsdframe
Out[21]:
Time (s)             x          y
----------   ---------  ---------
0.0          -0.029719  -0.273102
1.0           0.181754   3.25403
2.0          -0.495068  -0.524877
3.0          -1.20696   -0.033936
4.0          -0.664662   2.20862
...
95.0          0.942969   0.180585
96.0          2.15161    0.661736
97.0          0.751956  -1.72922
98.0         -1.45054    1.52954
99.0          0.199145   0.582944
dtype: float64, shape: (100, 2)
```

TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)

In [35]: tsdtensor
Out[35]:
Time (s)
----------  ----------------------
0.0         [[-0.87 ... -0.87] ...]
1.0         [[1.14 ... 1.14] ...]
2.0         [[0.25 ... 0.25] ...]
3.0         [[1.29 ... 1.29] ...]
4.0         [[-0.91 ... -0.91] ...]
...
95.0        [[-2.01 ... -2.01] ...]
96.0        [[-0.08 ... -0.08] ...]
97.0        [[0.53 ... 0.53] ...]
98.0        [[-0.62 ... -0.62] ...]
99.0        [[-0.55 ... -0.55] ...]
dtype: float64, shape: (100, 15, 15)
```

FLATIRON INSTITUTE

Numpy array

```
In [15]: tsd.index.values
Out[15]:
array([ 0.,   1.,   2.,   3.,
```

```
In [16]: tsd.values
Out[16]:
array([ 0.39704278,  1.55294416,
```

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0           0.397043
1.0           1.55294
2.0           0.455892
3.0          -1.17359
4.0          -0.110113
...
95.0         -0.573408
96.0         -0.0110915
97.0         -1.58027
98.0          0.998846
99.0          0.542692
dtype: float64, shape: (100,)
```

Numpy array

```
In [15]: tsd.index.values
Out[15]:
array([ 0.,  1.,  2.,  3.,
```

```
In [16]: tsd.values
Out[16]:
array([ 0.39704278,  1.55294416,
```

```
In [11]: tsd = nap.Tsd(t=t, d=d)

In [12]: tsd
Out[12]:
Time (s)
----------  ----------
0.0           0.397043
1.0           1.55294
2.0           0.455892
3.0          -1.17359
4.0          -0.110113
...
95.0         -0.573408
96.0         -0.0110915
97.0         -1.58027
98.0          0.998846
99.0          0.542692
dtype: float64, shape: (100,)
```

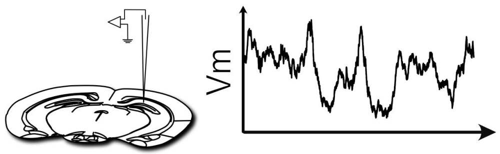
Numpy functions

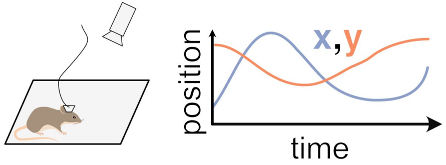
np.mean

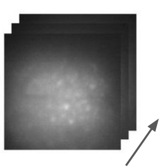
np.add

np.min

...







Tsd: 1-dimension

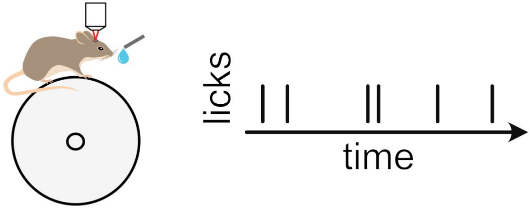
```
In [11]: tsd = nap.Tsd(t=t, d=d)
```

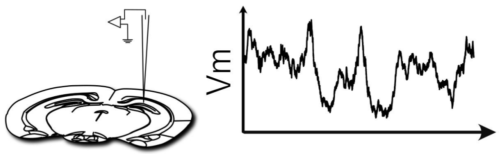
TsdFrame: 2-dimensions

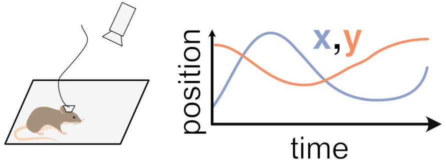
```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:       columns = ['x', 'y'])
```

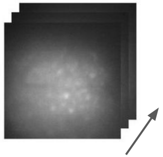
TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```









FLATIRON INSTITUTE

Tsd: 1-dimension

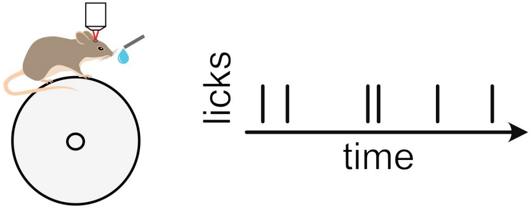
```
In [11]: tsd = nap.Tsd(t=t, d=d)
```

TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:         columns = ['x', 'y'])
```

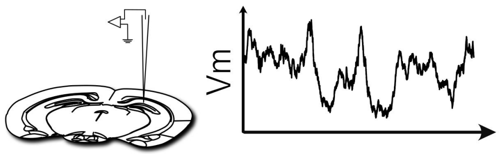
TsdTensor: n-dimensions

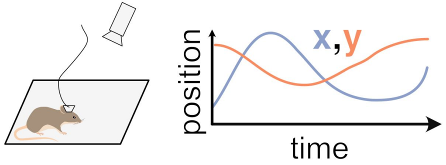
```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```

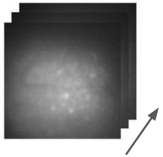


Ts: Timestamps

```
In [6]: nap.Ts(t)
Out[6]:
Time (s)
33.539693925
43.282779525
72.041005727
92.79257003
93.164316742
shape: 5
```







FLATIRON INSTITUTE

Tsd: 1-dimension

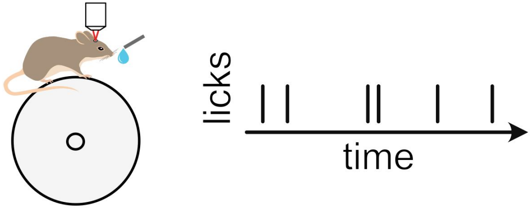
```
In [11]: tsd = nap.Tsd(t=t, d=d)
```

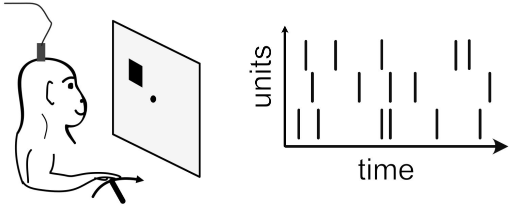
TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:        columns = ['x', 'y'])
```

TsdTensor: n-dimensions

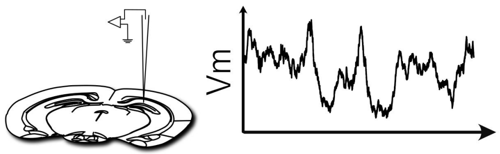
```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```





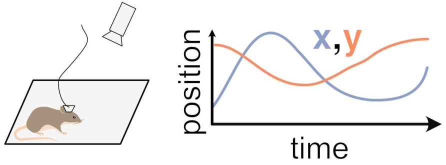
Ts: Timestamps

```
In [6]: nap.Ts(t)
Out[6]:
Time (s)
33.539693925
43.282779525
72.041005727
92.79257003
93.164316742
shape: 5
```



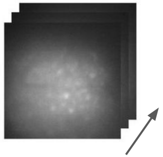
Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)
```



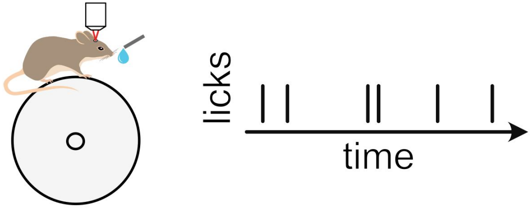
TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:      columns = ['x', 'y'])
```



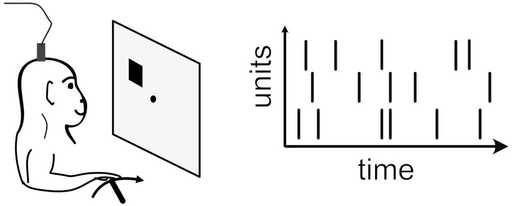
TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```



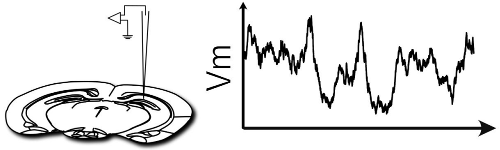
Ts: Timestamps

```
In [6]: nap.Ts(t)
Out[6]:
Time (s)
33.539693925
43.282779525
72.041005727
92.79257003
93.164316742
shape: 5
```



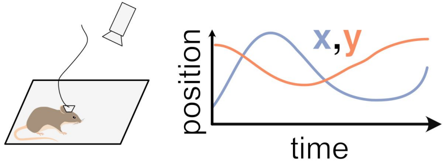
TsGroup: group of timestamps

```
In [18]: nap.TsGroup(data=data)
Out[18]:
  Index     rate
-------  ------
      0   10.02
      1    5.01
      2    2.06
```



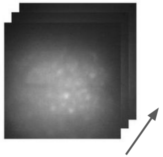
Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)
```



TsdFrame: 2-dimensions

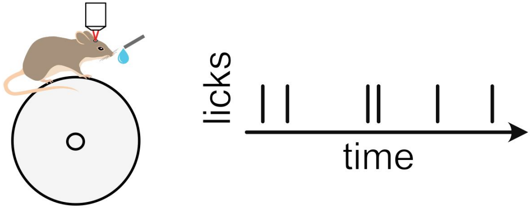
```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:         columns = ['x', 'y'])
```



FLATIRON INSTITUTE

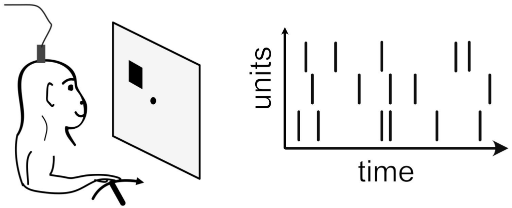
TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```



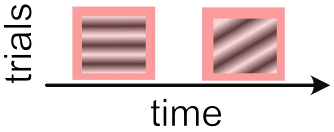
Ts: Timestamps

```
In [6]: nap.Ts(t)
```

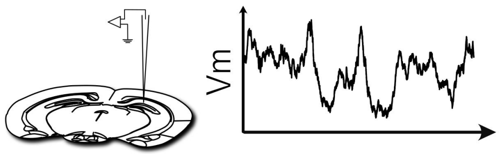


TsGroup: group of timestamps

```
In [18]: nap.TsGroup(data=data)
```

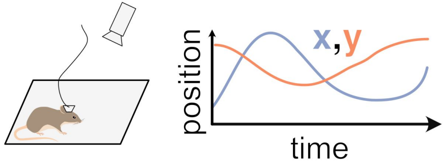


IntervalSet: set of epochs



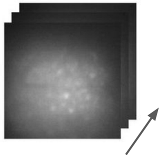
Tsd: 1-dimension

```
In [11]: tsd = nap.Tsd(t=t, d=d)
```



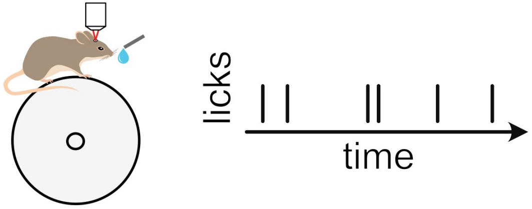
TsdFrame: 2-dimensions

```
In [20]: tsdframe = nap.TsdFrame(t=t, d=d,
    ...:         columns = ['x', 'y'])
```



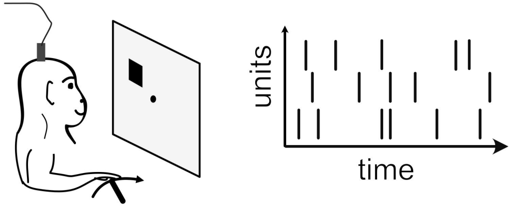
TsdTensor: n-dimensions

```
In [34]: tsdtensor = nap.TsdTensor(t=t, d=d)
```



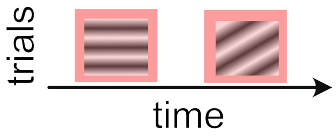
Ts: Timestamps

```
In [6]: nap.Ts(t)
```



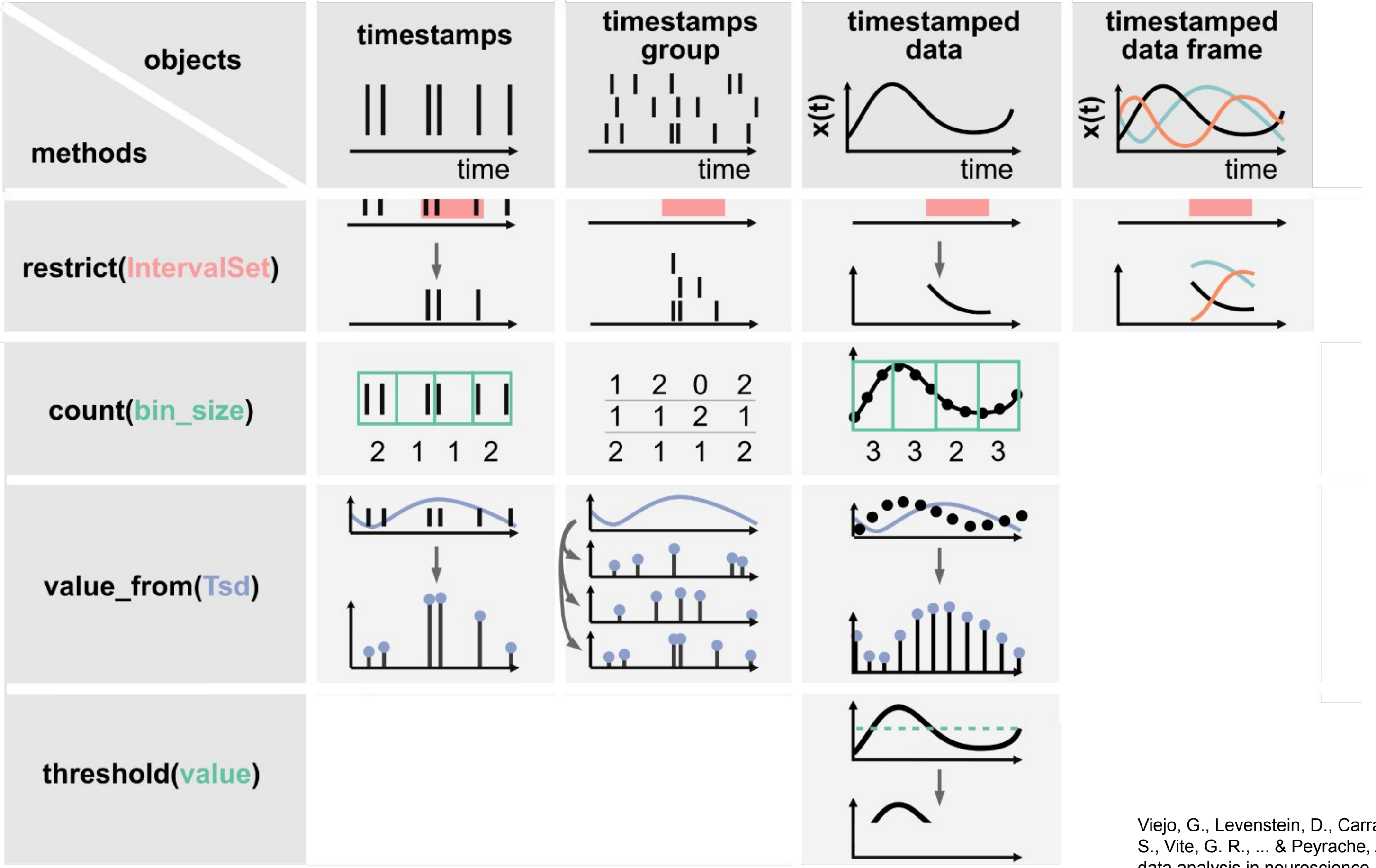
TsGroup: group of timestamps

```
In [18]: nap.TsGroup(data=data)
```

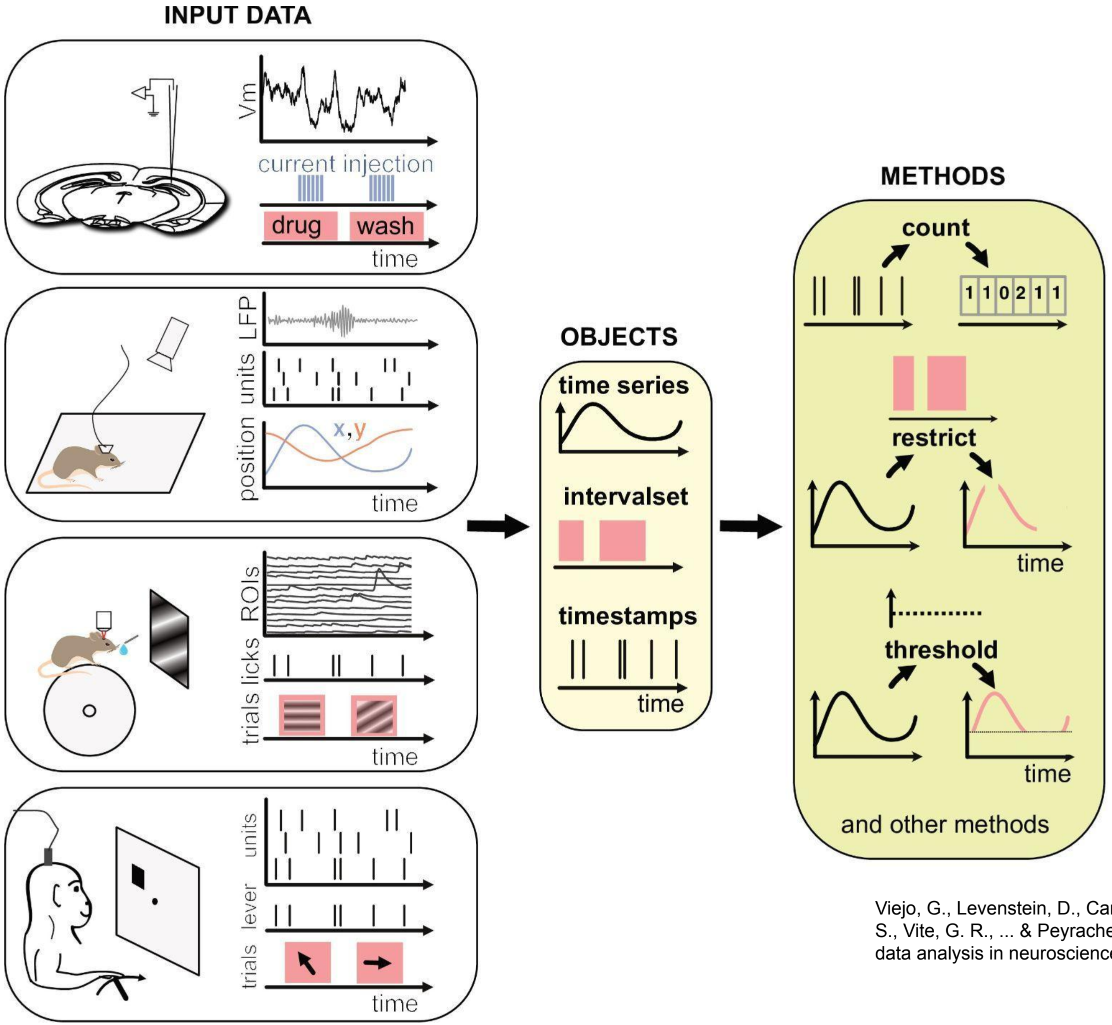


IntervalSet: set of epochs

```
In [23]: nap.IntervalSet(
    ...:     start=start,
    ...:     end = end)
Out[23]:
   start   end
0    0.0   1.0
1    3.0   5.0
2    9.0  12.0
```



Viejo, G., Levenstein, D., Carrasco, S. S., Mehrotra, D., Mahallati, S., Vite, G. R., ... & Peyrache, A. (2023). Pynapple, a toolbox for data analysis in neuroscience. *eLife*, *12*, RP85786.



Viejo, G., Levenstein, D., Carrasco, S. S., Mehrotra, D., Mahallati, S., Vite, G. R., ... & Peyrache, A. (2023). Pynapple, a toolbox for data analysis in neuroscience. *eLife*, *12*, RP85786.

NWB interface

In [1]:
```python
import pynapple as nap

nwb = nap.load_file("A2929-200711.nwb")
```

In [2]:
```python
nwb
```

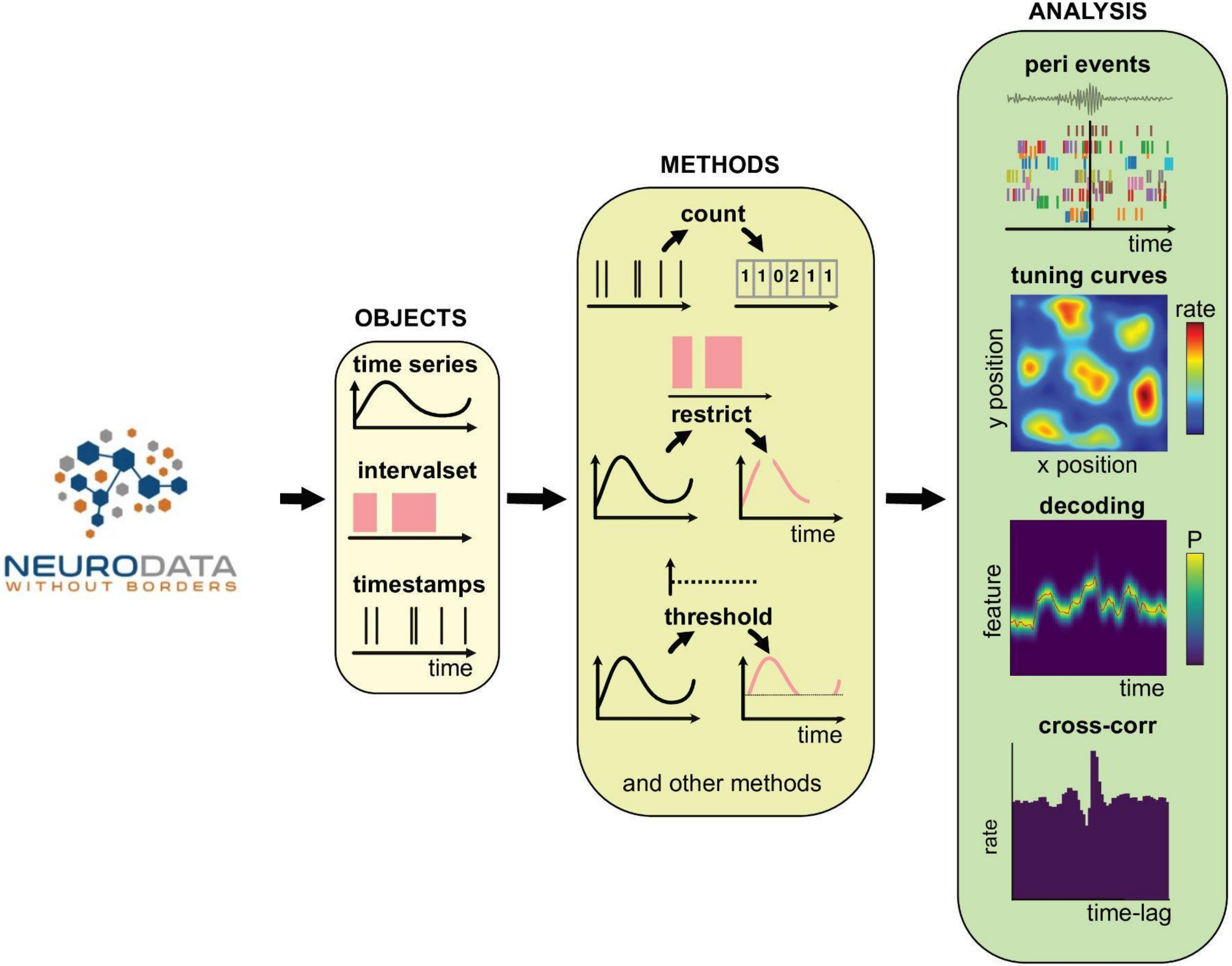
A2929-200711.nwb

Keys	Type
position_time_support	IntervalSet
epochs	IntervalSet
z	Tsd
y	Tsd
x	Tsd
rz	Tsd
ry	Tsd
rx	Tsd

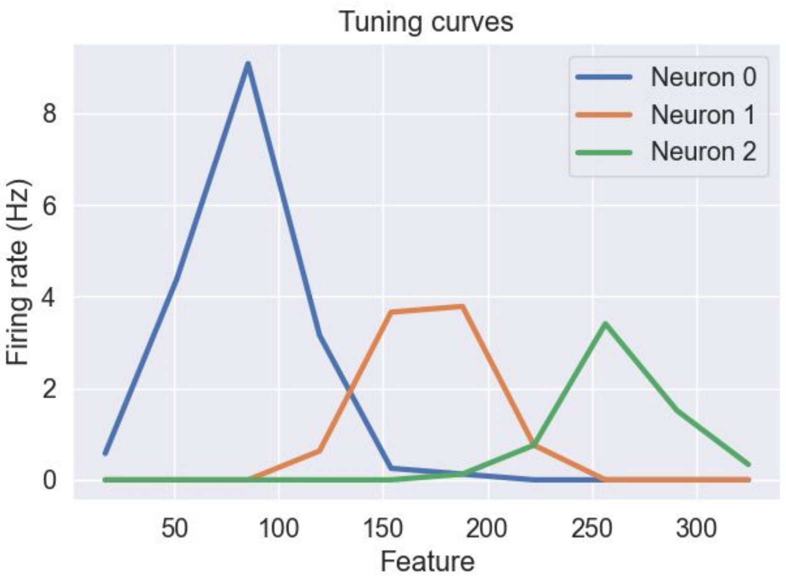
In [3]:
```python
nwb["position_time_support"]
```

Out[3]:

	start	end
0	670.6407	1199.99495



FLATIRON INSTITUTE




```python
tuning_curve = nap.compute_1d_tuning_curves(
    spikes,
    feature,
    nb_bins=10,
    ep=ep)
```

Hands-on session: NWB data analysis in the cloud

- Open folder stratneuro_2024/tutorial_nwb_analysis/
- Open **nwb_stream_analysis.ipynb**
- We will stream and run data analysis in remote NWB files

Want help?

CATALYST NEURO

We are an international team of neuroscientists

We use data and software engineering to connect neurophysiology data producers, tool builders, and users

CatalystNeuro Clients for NWB Adoption

1. Hillel Adesnik, UC Berkeley
2. Misha Ahrens, Janelia
3. Richard Axel, Columbia
4. Carlos Brody, Princeton
5. Bing Brunton, UW
6. Beth Buffalo, UW
7. Gyorgy Buzsaki, NYU
8. Eddie Chang, UCSF
9. Thomas Clandinin, Stanford
10. Bob Datta, Harvard
11. Saskia de Vries, Allen Institute
12. Michale Fee, MIT
13. Dan Feldman, UC Berkeley
14. Robert Froemke, NYU
15. Lisa Giocomo, Stanford
16. Michael Higley, Yale
17. International Brain Lab
18. Dieter Jaeger, Emory
19. Mehrdad Jazayeri, MIT
20. Talia Lerner, Northwestern
21. Dayu Lin, NYU
22. Attila Losonczy, Columbia
23. Tony Movshon, NYU
24. Mala Murthy, Princeton
25. Lydia Ng, Allen Institute
26. Daniel O'Connor, JHU
27. Lucas Pinto, Northwestern
28. Mark Schnitzer, Stanford
29. Eyal Seidemann, UT Austin
30. Marshall Hussain Schuler, JHU
31. Ivan Soltesz, Stanford
32. Sergey Stavisky, UC Davis
33. Krishna Shenoy, Stanford
34. David Tank, Princeton
35. Robert Turner, Pitt
36. Kay Tye, Salk

…

Schedule a free consultation

CATALYST NEURO