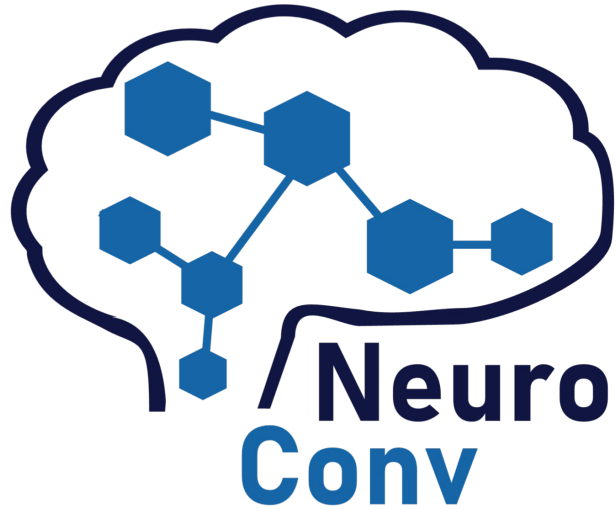


NWB User Days 2023



&



**We are an
international
team of
neuroscientists**



Paul Adkisson



Garrett Flynn



**Cody Baker,
Ph.D.**



Ben Dichter, Ph.D.



Melissa Hersh



**Julia Sprenger,
Ph.D.**



**Pierre Yger,
Ph.D.**



Samuel Garcia



Luiz Tauffer



**Ramon Mayorquin,
Ph.D.**



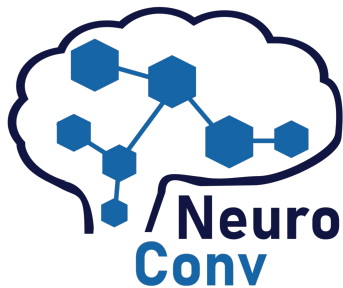
**Alessio Buccino,
Ph.D.**



**Anna (Szonja)
Weigl**



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



NeuroConv: A Python library for automatic conversions to NWB

Cody Baker
Szonja Weigl
Heberto Mayorquin
Alessio Buccino
Ben Dichter



THE
KAVLI
FOUNDATION

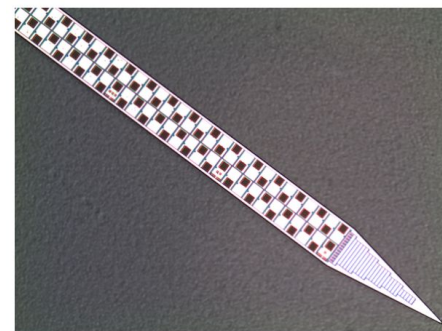


Challenges

- Converting neurophysiology data to NWB is challenging due to
 - ▶ **Diversity** - many different proprietary formats
 - ▶ **Complexity** - experiments often use multiple data sources (each stream in a unique format)
 - ▶ **Volume** - terabyte (TB) scale and growing

Challenges

- Converting neurophysiology data to NWB remains challenging due to
 - ▶ **Diversity** - many different proprietary formats
 - ▶ **Complexity** - experiments often use multiple data sources (each stream in a unique format)
 - ▶ **Volume** - terabyte (TB) scale and growing
- Cutting edge recording techniques continue to push the boundaries of temporal and spatial resolution
- Starting to see data sharing of...
 - ▶ multi-probe Neuropixels (version 1) recordings
 - ~30 kHz sampling rate can yield ~23 MB/s of data per probe
 - That's ~1 TB of data per probe every 12 hours!
 - ▶ whole-brain imaging (example resolution: ~1k x 2k px imaging plane)
 - ~40 scans per plane per second can yield ~160 MB/s of data
 - That's ~1 TB every 2 hours!



neuropixels.org

Challenges

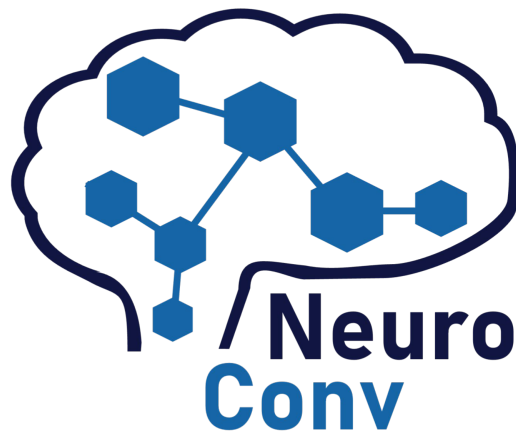
- Converting neurophysiology data to NWB remains challenging due to
 - ▶ **Diversity** - many different proprietary formats
 - ▶ **Complexity** - experiments often use multiple data sources (each stream in a unique format)
 - ▶ **Volume** - terabyte (TB) scale and growing
- Reminder: DANDI is happy to store all that raw data **for free!**



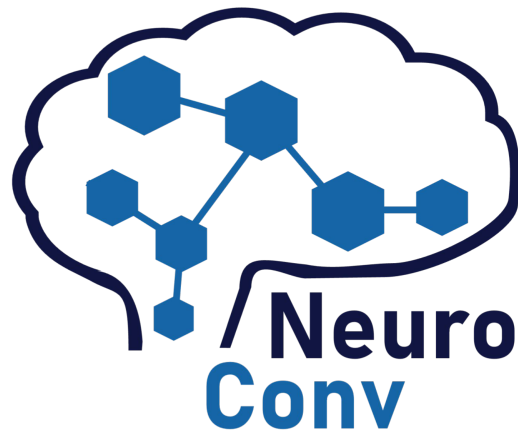
Challenges

NeuroConv is an open-source software package developed to simplify these challenges for converting neurophysiology data to NWB

- Currently supports 40+ common neurophysiology 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
- Allows easy combinations from multiple input sources



Installation



- Minimal installation

```
pip install neuroconv
```

- Format-specific installation

```
pip install neuroconv[<format name>]
```

E.g., `pip install neuroconv[spikeglx]`

- Full installation (if you plan to use EVERY format... mostly for developers)

```
pip install neuroconv[full]
```

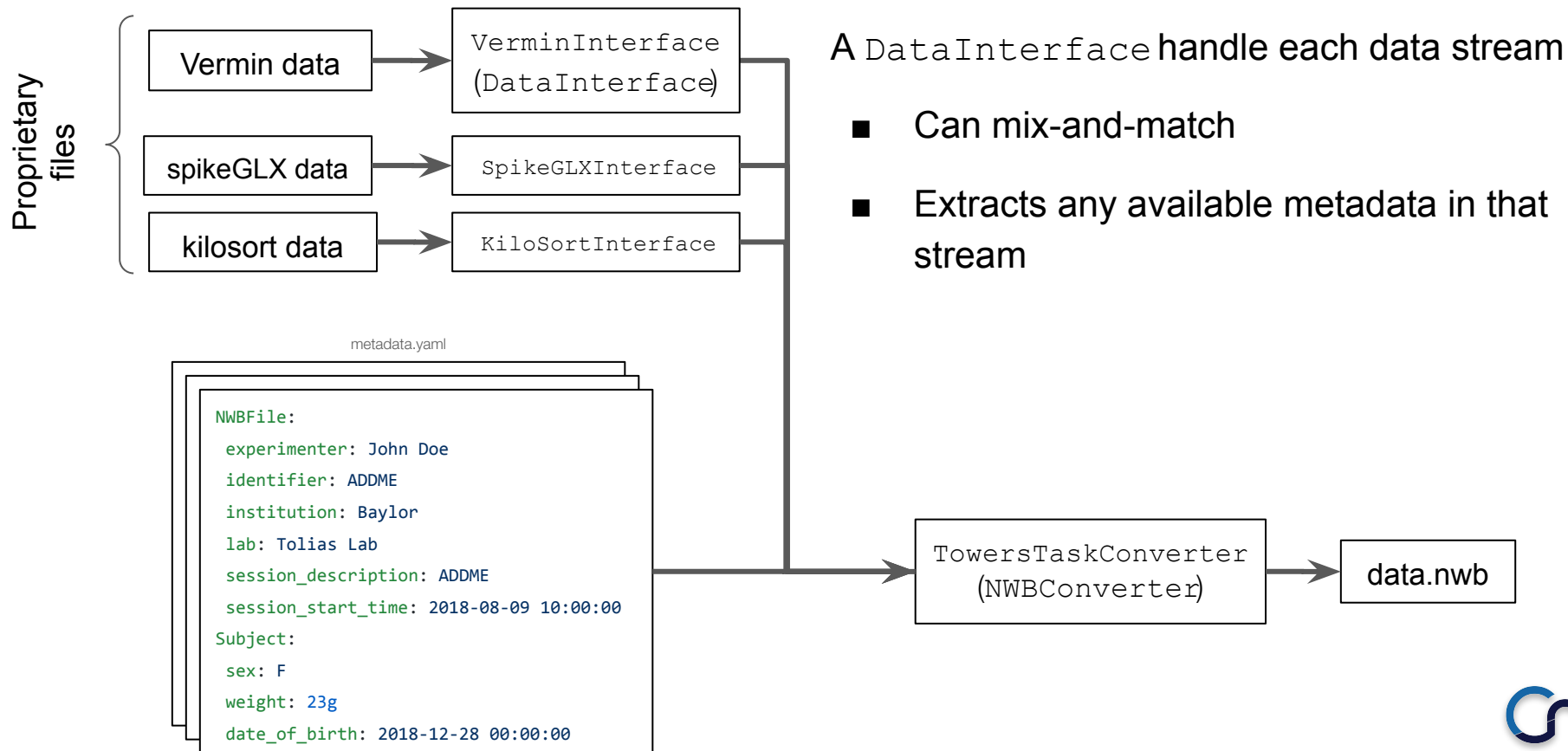
- Can always specify more than one

```
pip install neuroconv[tiff,extract,sleap]
```

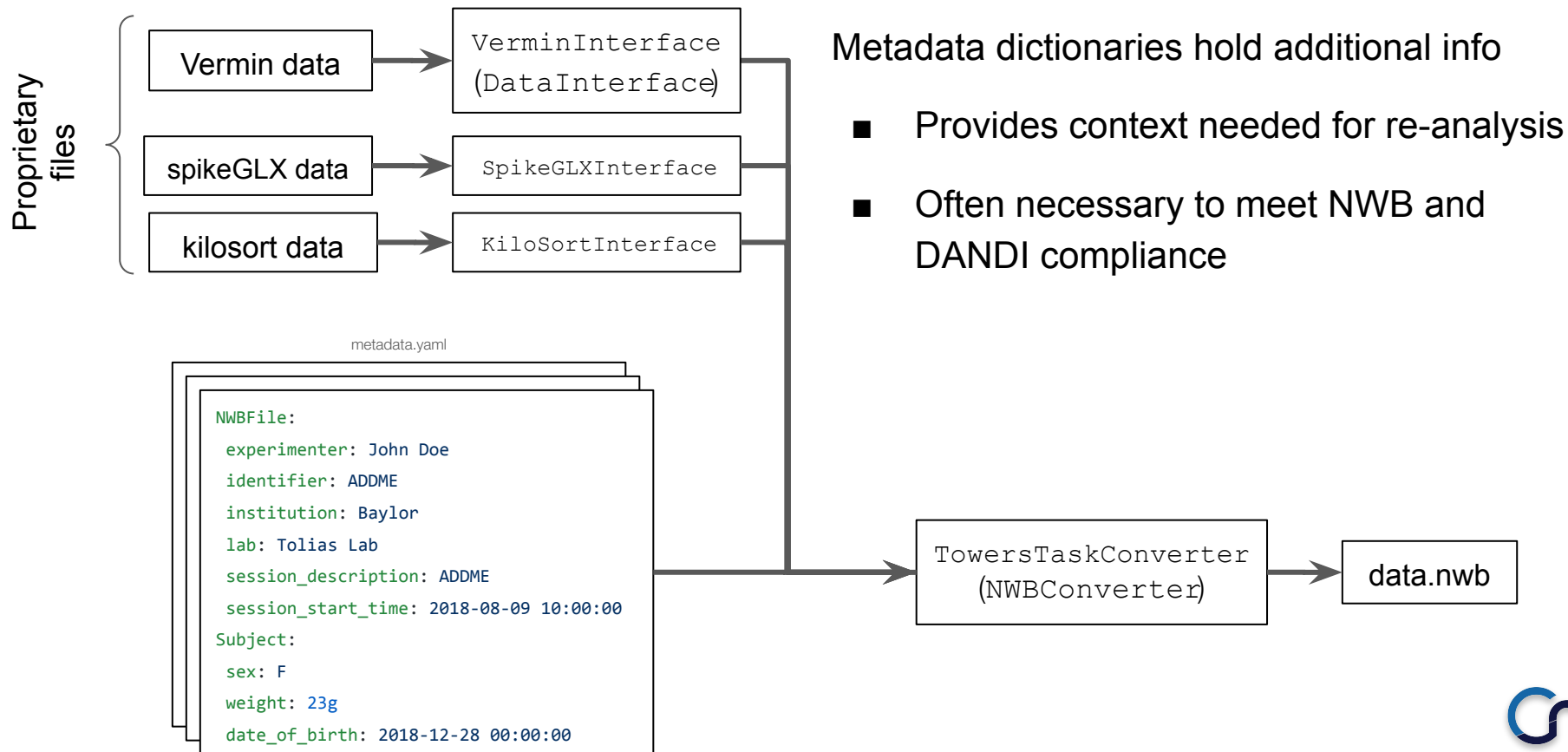
- Mac Users: must wrap in quotes when using extra requirements

```
pip install "neuroconv[full]"
```

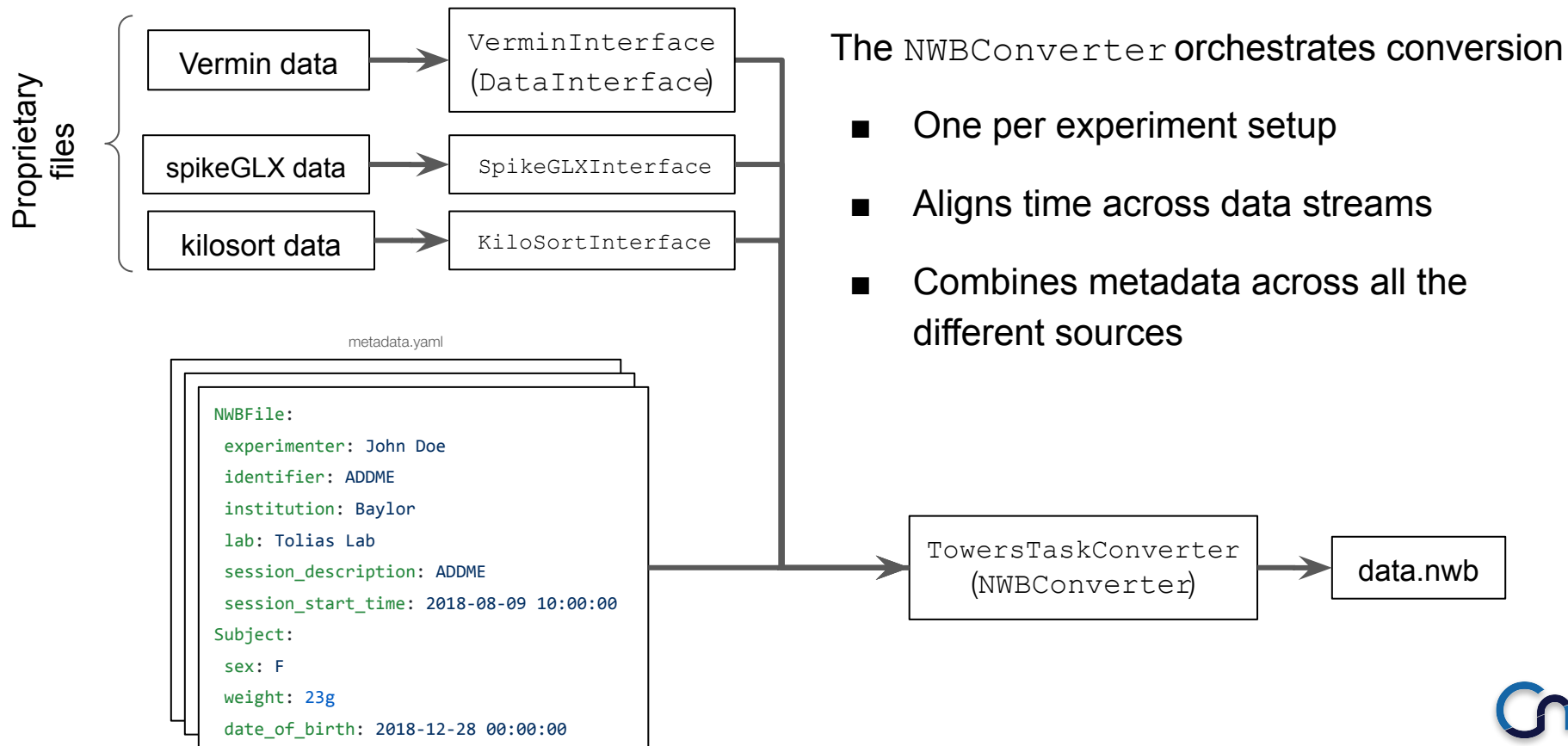

Strategy: Modularize by data stream



Strategy: Modularize by data stream



Strategy: Modularize by data stream



So what does it look like in practice?

Simple Snippet

```
from datetime import datetime
from neuroconv.datainterfaces import TiffImagingInterface

file_path = ... / "imaging_datasets" / "Tif" / "demoMovie.tif"
session_start_time = datetime(2020, 1, 1, 12, 30, 0)

interface = TiffImagingInterface(
    file_path="../../../imaging_datasets/Tif/demoMovie.tif",
    sampling_frequency=15.0
)
metadata = interface.get_metadata()
metadata["NWBFile"].update(session_start_time=session_start_time)
interface.run_conversion(
    nwbfile_path="../../../my_nwbfile.nwb", metadata=metadata
)
```

Demos

<https://neuroconv.readthedocs.io>

- + “Conversion Gallery”
- + “Catalogue”

Don't see your format listed? Request support for it

<https://github.com/catalystneuro/neuroconv/issues>

- + Need a small snippet of example data for our testing repository
- + Helps if you know of a Python library for reading the format



bit.ly/nwb-user-days-2023

No-code Specification File

For users less familiar with coding in Python, we have developed a text-only YAML language for specifying a full NWB conversion

- Only requires specification of file/folder paths and metadata not within the sources
- Example:
https://github.com/catalystneuro/neuroconv/blob/main/tests/test_on_data/conversion_specifications/GIN_conversion_specification_missing_nwbfile_names.yml
- Run conversion from the command line

```
neuroconv my_conversion_specification.yml
```

Note: Behavior Types

Most representations of behavior used by labs is ad-hoc (not proprietary or widespread)...

- NeuroConv supports video, audio, spreadsheet tables, and DeepLabCut + SLEAP for pose estimation
- Open to supporting more, possibly including arbitrary objects from `.mat`, `.pkl`, or `.npy` files, but we would have to work with you to ensure it meets your needs
- Usually this is the most custom part of the conversion and will require a little bit of actual PyNWB or MatNWB code...
- ...so be sure to check out the tutorial sessions for those after lunch

New Feature: Multi-stream Converters

A `DataInterface` represents the lowest level of connection between a single data stream and its corresponding NWB representation

Many formats have multiple streams, even some that are cross-modal (simultaneous behavior + neural recording)

So we have made two ‘subconverters’ (converters that can also be combined within another `NWBConverter` and act much like an interface)

```
from neuroconv.converters import MiniscopeConverter
```

```
miniscope_converter = MiniscopeConverter(  
    folder_path= ... / "miniscope_folder"  
) # Includes the behavior camera as well!
```

```
miniscope_converter.run_conversion()
```

```
from neuroconv.converters import SpikeGLXConverterPipe
```

```
spikeglx_converter = SpikeGLXConverterPipe(  
    folder_path= ... / "spikeglx_folder"  
)
```

```
spikeglx_converter.run_conversion()
```

Advanced Feature: Metadata Customization

All metadata that goes into the NWB file can be controlled via the `NeuroConv` `metadata` dictionary structure

No need to call `.update(key=value)` - fields can be referenced directly

```
from neuroconv.datainterfaces import TiffImagingInterface

interface = TiffImagingInterface(
    file_path= ... / "demoMovie.tif",
    sampling_frequency=15.0
)
metadata = interface.get_metadata()

metadata["NWBFile"]["session_start_time"] = datetime(2020, 1, 1, 12, 30, 0)

metadata["NWBFile"]["Subject"]["id"] = "MySubjectID001"
metadata["NWBFile"]["Subject"]["species"] = "Mus musculus" # Latin binomial
metadata["NWBFile"]["Subject"]["sex"] = "U" # Must be M, F, U, or O
metadata["NWBFile"]["Subject"]["age"] = "P1W3D" # Must be in ISO 8601

# Modality-specific structures are mostly auto-filled, can be viewed with
metadata["Ecephys"]
metadata["Ophys"]
metadata["Behavior"]
```


Advanced Feature: Metadata Schema

For those familiar with reading JSON, all NWB metadata in **NeuroConv** have a schema representation

When calling `.run_conversion()` on an `NWBConverter` the metadata structure you pass in will be validated against this schema

The **NWB GUIDE** interactively validates each entry as it is modified

```
from neuroconv.datainterfaces import TiffImagingInterface

interface = TiffImagingInterface(
    file_path= ... / "demoMovie.tif",
    sampling_frequency=15.0
)
metadata_schema = interface.get_metadata_schema()
```

```
...
    "required": [
        "Device",
        "ImagingPlane",
        "TwoPhotonSeries"
    ],
    "properties": {
        "Device": {
            "type": "array",
            "minItems": 1,
            "items": {
                "$ref": "#/properties/Ophys/properties/definitions/Device"
            },
            "default": [
                {
                    "name": "Microscope"
                }
            ]
        },
        "ImagingPlane": {
            "type": "array",
            "minItems": 1,
            "items": {
                "$ref": "#/properties/Ophys/properties/definitions/ImagingPlane"
            },
            "default": [
                {
                    "name": "ImagingPlane"
                }
            ]
        }
    }
}
```

New Advanced Feature: Path Expansion

It's common for experimenters to store source data in some structured way

Python f-strings: `f"{subject_id}/{session_date}/data_stream.ap.bin"`

A list of results of such an f-string used to encode file or folder paths can oftentimes be inverted to recovery the values of the variables (which can auto-populate NWB metadata!)

Demo and several examples:

https://neuroconv.readthedocs.io/en/main/user_guide/expand_path.html

New Advanced Feature: Temporal Alignment

All temporal data in NWB files must be aligned to the `session_start_time`, but there are several different synchronization strategies used to achieve this

Check out the [full documentation](#) of methods for all the details, but in a conversion script it might look something like this...

```
from datetime import datetime
from neuroconv.datainterfaces import SpikeGLXNIDQInterface, DeepLabCutInterface

nidq_interface = SpikeGLXNIDQInterface(file_path= ... / "some_spikeglx_session.nidq.ap.bin")
ttl_times = nidq_interface.get_event_times_from_ttl(channel_name="AUX1")

dlc_interface = DeepLabCutInterface(folder_path=... / "dlc_session")
dlc_interface.align_starting_time(starting_time=ttl_times[0])

...
```



NWB GUIDE: A Graphical User Interface for Data Entry

Cody Baker
Garrett Flynn
Oliver Ruebel
Ben Dichter



THE
KAVLI
FOUNDATION



Summary

NWB GUIDE is an open-source graphical interface developed around NeuroConv to remove the necessity of knowing how to program in Python

- Drag-and-drop data from local folders
- Interactive forms for attaching metadata
- Automated validation and upload to the DANDI Archive
- Currently only supports SpikeGLX and Phy formats
- Beta release at SFN 2023 will include all 40+ NeuroConv supported formats

Tutorial

NWB GUIDE has a built-in tutorial using some NeuroPixels (SpikeGLX) data with Phy-curated KiloSort output

To complete the entire workflow from start-to-finish

- Make an account on <https://dandiarchive.org>
 - + Requires an account on <https://github.com>
- Download the 'ephy_testing_data' folder from [this Google Drive](#)
- [Installation instructions](#)

Strategy for Hackathon

If you have SpikeGLX or Phy data, then the recommended method would be to try out the **NWB GUIDE** to create some base files to which you can add behavioral data to later

If you have any other [formats supported by NeuroConv](#), then start by ensuring it is able to convert your files without error

- If any errors do occur, please report them on the [NeuroConv GitHub issues](#)
- Szonja/Paul/Cody will be here to help, just raise your hand if you have any questions

If your proprietary format is not supported, then please [submit a request](#) for it on NeuroConv

- If you do not yet know how to read it in Python, take this time to sit down with one of our expert developers to see if we can help