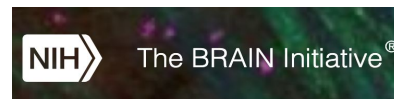


# Searching the DANDI Archive



## Support



1R24MH117295  
AWS Public dataset program



*Dandiset. n.*

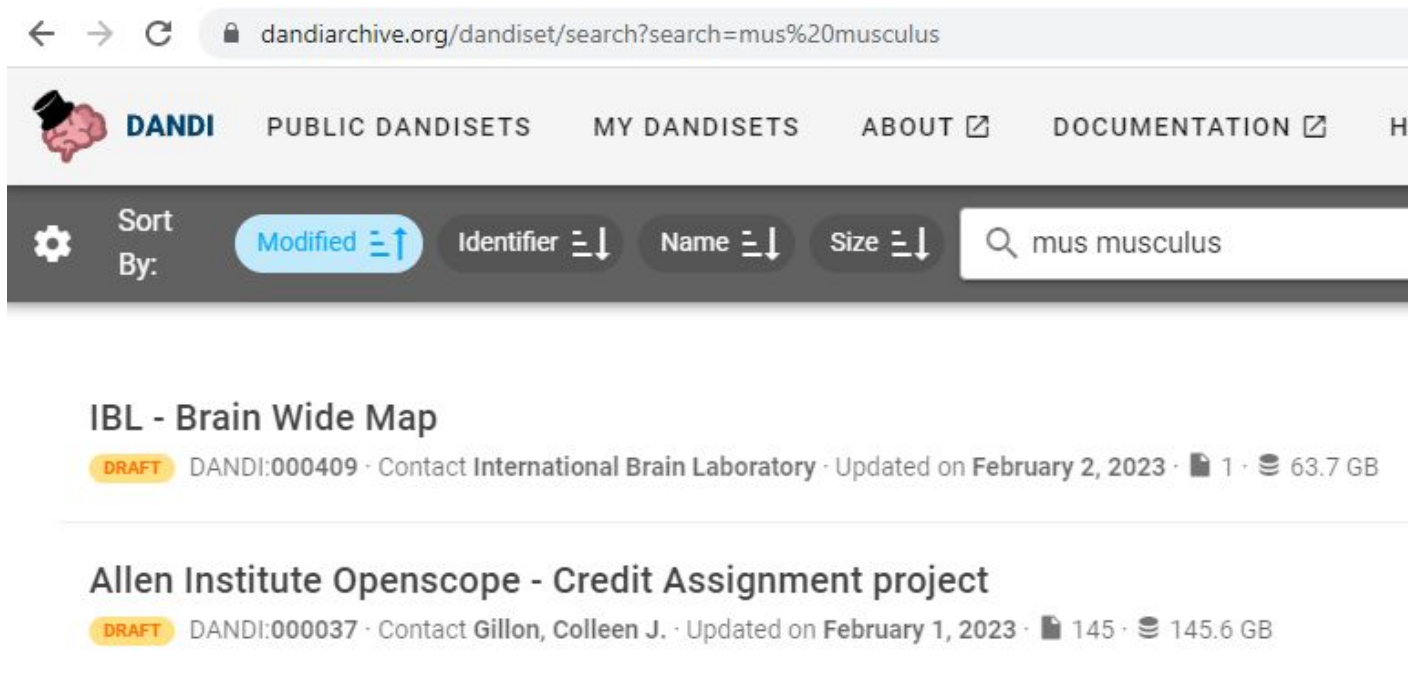
An organized collection of assets (files) with both file level and dataset level metadata generated from an experiment or a project.

*A dandiset is a **FAIR** collection.*

***F**indable   **A**ccessible   **I**nteroperable   **R**eusable*

# From the Web Interface

- From the main page <https://dandiarchive.org/dandiset>, type a query in the search bar



The screenshot shows the DANDI Archive web interface. The browser address bar displays the URL `dandiarchive.org/dandiset/search?search=mus%20musculus`. The DANDI logo is in the top left, followed by navigation links: PUBLIC DANDISSETS, MY DANDISSETS, ABOUT, and DOCUMENTATION. Below these is a search bar containing the text "mus musculus". To the left of the search bar are sorting options: "Sort By:" with a dropdown menu showing "Modified" (selected), "Identifier", "Name", and "Size". Below the search bar, two search results are listed:

- IBL - Brain Wide Map**  
DRAFT DANDI:000409 · Contact International Brain Laboratory · Updated on February 2, 2023 · 1 · 63.7 GB
- Allen Institute Openscope - Credit Assignment project**  
DRAFT DANDI:000037 · Contact Gillon, Colleen J. · Updated on February 1, 2023 · 145 · 145.6 GB

# From the Web Interface

- These top-level queries look at the highest level of dandiset metadata
  - Titles, keywords, dandiset description
  - Automatically extracted subject species, modality, and techniques
  - You can see these fields listed at the bottom of the main page of any dandiset



## Assets Summary

### Species

Mus musculus - House mouse



Rattus norvegicus - Norway rat



### Approach

electrophysiological approach

### Variable Measured

ElectrodeGroup

ElectricalSeries

### Measurement Technique

multi electrode extracellular electrophysiology  
recording technique

surgical technique

# From the Web Interface

- Good for...
  - A quick glance or casual browsing of general content
  - Finding the dandiset for a particular publication
    - usually the title, or linked as a Related Resource ('IsDescribedBy')
  - Finding all the dandisets belonging to a particular lab
    - search by name of a 'Contributor'
  - Finding all the dandisets that use a particular species
    - Latin binomial, e.g.: *Mus musculus*, *Rattus norvegicus*, etc.

# From the Web Interface

- Doesn't help with...
  - Presence or absence of raw, pre-processed, or post-processed data
  - Presence or absence of identified brain regions
    - unless the creator of the dandiset set these as keywords or 'subject matter'
  - Presence or absence of exact brain coordinates (like Allen CCFv3)
  - The huge variety of behavioral techniques
    - open exploration vs. maze task
    - virtual reality vs. simple stimulus presentation
    - trialized tasks or spontaneous events
    - and many, many more...

# Using the DANDI API in Python

- For finer-grain searchability, we can use the **ap**plied **p**rogramming interface (API) for DANDI to scan the metadata.yml files to programmatically obtain information
- Installation - preferably in a new conda environment

```
pip install dandi jupyter  
jupyter notebook
```

Or use the DANDI Hub



Notebook



# Using the DANDI API in Python

- For finer-grain searchability, we can use the **applied programming interface (API)** for DANDI to scan the `metadata.yml` files to programmatically obtain information
- Relevant methods
  - The `client` initiates communication with the archive

```
from dandi.dandiapi import DandiAPIClient
```

```
client = DandiAPIClient()
```



# Using the DANDI API in Python

- For finer-grain searchability, we can use the **applied programming interface (API)** for DANDI to scan the `metadata.yml` files to programmatically obtain information
- Relevant methods
  - The `client` can be queried to return currently public dandisets

```
dandisets = list(client.get_dandisets())  
dandiset = dandisets[0]  
dandiset  
> dandi.dandiapi.RemoteDandiset
```

# Using the DANDI API in Python

- For finer-grain searchability, we can use the **applied programming interface (API)** for DANDI to scan the metadata.yml files to programmatically obtain information
- Relevant methods
  - A `dandi.dandiapi.RemoteDandiset` can return its pre-parsed metadata

```
raw_metadata = dandiset.get_raw_metadata()  
raw_metadata  
> { < kind of messy > }
```

# Using the DANDI API in Python

- For finer-grain searchability, we can use the **applied programming interface (API)** for DANDI to scan the metadata.yml files to programmatically obtain information
- Relevant methods
  - A `dandi.dandiapi.RemoteDandiset` can return its pre-parsed metadata

```
import json
```

```
json.dumps(raw_metadata, indent=4)
```

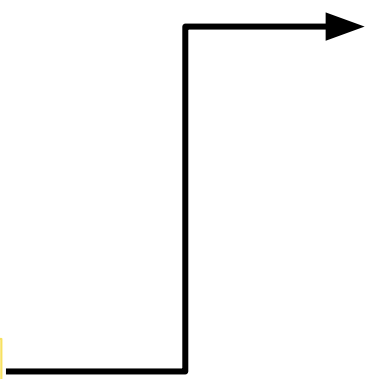
```
> {
```

```
    < A lot more readable >
```

```
}
```

# Using the DANDI API in Python

```
[ 'id', 'identifier',  
  'doi', 'repository',  
  'url', 'contributor',  
  'name', 'description',  
  'about', 'publishedBy',  
  'access', 'studyTarget',  
  'license', 'assetsSummary',  
  'version', 'datePublished',  
  '@context', 'schemaVersion',  
  'citation', 'ethicsApproval',  
  'keywords', 'wasGeneratedBy',  
  'protocol', 'relatedResource',  
  'schemaKey', 'manifestLocation']
```



A diagram consisting of a black arrow originates from the 'assetsSummary' field in the list on the left and points to the 'assetsSummary' field in the JSON object on the right.

```
"assetsSummary": [  
  "species",  
  "approach",  
  "schemaKey",  
  "dataStandard",  
  "numberOfBytes",  
  "numberOfFiles",  
  "numberOfSubjects",  
  "variableMeasured",  
  "measurementTechnique"  
]
```

# Using the DANDI API in Python

```
{
  "age": {
    "value": "P209DT55274S",
    "unitText": "ISO-8601 duration",
    "schemaKey": "PropertyValue",
    "valueReference": {
      "value": "dandi:BirthReference",
      "schemaKey": "PropertyValue"
    }
  },
  "sex": {
    "name": "Male",
    "schemaKey": "SexType",
    "identifier": "http://purl.obolibrary.org/obo/PATO\_0000384"
  },
  "species": {
    "name": "Mus musculus - House mouse",
    "schemaKey": "SpeciesType",
    "identifier": "http://purl.obolibrary.org/obo/NCBITaxon\_10062"
  },
  "genotype": "Emx1-Cre[tg/wt];Ai32[tg/wt]",
  "schemaKey": "Participant",
  "identifier": "San4"
},
  "assetsSummary": [
    "species",
    "approach",
    "schemaKey",
    "dataStandard",
    "numberOfBytes",
    "numberOfFiles",
    "numberOfSubjects",
    "variableMeasured",
    "measurementTechnique"
  ]
}
```



A diagram consisting of a black line that originates from the "species" field within the "assetsSummary" array on the right. The line extends to the left and then turns downwards as an arrow, pointing directly at the "species" field within the main object on the left. This visualizes a relationship or mapping between the two "species" fields.

# Using the DANDI API in Python

- For finer-grain searchability, we can use the dandi-api to scan the metadata.yml files to programmatically obtain information

- Relevant methods

- Each file from `dandiset.get_asset(...)` can return its pre-parsed metadata

```
all_assets = dandiset.get_assets()
```

```
first_asset_raw_metadata = all_assets[0].get_raw_metadata()
```

```
print(json.dumps(first_asset_raw_metadata, indent=4))
```

```
> {
```

```
    < A lot of information >
```

```
}
```

# Using the DANDI API in Python

- Simple dandiset-level examples

[https://github.com/NeurodataWithoutBorders/nwb\\_hackathons/blob/add\\_cosyne\\_dandi\\_search\\_tutorial/Cosyne\\_2023/tutorials/dandi%20search%20cookbook.ipynb](https://github.com/NeurodataWithoutBorders/nwb_hackathons/blob/add_cosyne_dandi_search_tutorial/Cosyne_2023/tutorials/dandi%20search%20cookbook.ipynb)

- Advanced asset-level examples

[https://github.com/NeurodataWithoutBorders/nwb\\_hackathons/blob/add\\_cosyne\\_dandi\\_search\\_tutorial/Cosyne\\_2023/tutorials/dandi%20search%20cookbook.ipynb](https://github.com/NeurodataWithoutBorders/nwb_hackathons/blob/add_cosyne_dandi_search_tutorial/Cosyne_2023/tutorials/dandi%20search%20cookbook.ipynb)

# Investigating an Individual NWB File on DANDI

- The dandi-api metadata aggregates information over the contents of the dandiset
- To investigate the contents of a single file, a good place to start is to try the NWB Widgets

```
pip install -U pynwb dandi  
pip install nwbwidgets  
jupyter notebook
```

And in the notebook...

```
from nwbwidgets import Panel  
  
Panel()
```



☐ Local dir

☐ Local file

☒ DANDI

☐ S3

Dandiset: 000409 - IBL - Brain Wide Map

File: sub-PL015/sub-PL015\_ses-1d4a7bd6-296a-48b9-b20e-bd0ac80750a5

✓ Load file

The International Brain lab (IBL) aims to understand the neural basis of decision-making in the mouse by gathering a whole-brain activity map composed of electrophysiological recordings pooled from multiple laboratories. We have systematically recorded from nearly all major brain areas with Neuropixels probes, using a grid system for unbiased sampling and replicating each recording site in at least two laboratories. These data

session\_description: The full description of the session/task protocol can be found in Appendix 2 of Inte

identifier: c33e2740-5475-463e-bd16-d1c38da37463

session\_start\_time: 2022-07-21 16:08:53.428769+01:00

timestamps\_reference\_time: 2022-07-21 16:08:53.428769+01:00

related\_publications: <https://doi.org/10.6084/m9.figshare.21400815.v6>, <https://doi.org/10.1101/2020.01.17.909838>

experiment\_description: IBL aims to understand the neural basis of decision-making in the mouse by gather

session\_id: 1d4a7bd6-296a-48b9-b20e-bd0ac80750a5

lab: Hausser

institution: University College London

protocol: \_iblrig\_tasks\_ephysChoiceWorld6.6.1

▸ file\_create\_date

▸ acquisition

▸ processing

▸ electrodes: metadata about extracellular electrodes

▸ electrode\_groups

▸ devices

# Reading directly from an identified file

- Once you have concluded your investigation and found some NWB files of interest, you can either...
  - download them to local device via command-line

```
dandi download DANDI:<six-digit-ID> # Will download all files
```

Or

```
dandi download <copy and paste individual file URL>
```

Then in Python (script or notebook)...

```
from pynwb import NWBHDF5IO
```

```
io = NWBHDF5IO(path='../path_to_single_file.nwb', load_namespaces=True)
```

```
nwbfile = io.read()
```

# Reading directly from an identified file

- Once you have concluded your investigation and found some NWB files of interest, you can either...
  - download them locally via command-line

```
dandi download DANDI:<six-digit-ID> # Will download all files
```

Or

```
dandi download <copy and paste individual file URL>
```

Then in MATLAB...

```
%% With MatNWB downloaded and added to your MATLAB session path...
```

```
nwbfile = nwbRead('.../path_to_single_file.nwb')
```

# Reading directly from an identified file

- Or stream from the cloud (most recommended for one-off analyses or quick calculations)

In Python...

```
import h5py
import fsspec
from dandi.dandiapi import DandiAPIClient
from pynwb import NWBHDF5IO

client = DandiAPIClient()

s3_url =
client.get_dandiset("<six-digit-ID>").get_asset_by_path(path="path/on/DANDI").api_url
cfs = fsspec.implementations.cached.CachingFileSystem(
    fs=fsspec.filesystem("http"), cache_storage="some/temporary/folder",
)
file_system = cfs.open(s3_url, "rb")
file = h5py.File(file_system)
io = NWBHDF5IO(file=file, load_namespaces=True)
nwbfile = io.read()
```

# Reading directly from an identified file

- Or stream from the cloud (most recommended for one-off analyses or quick calculations)

[In MATLAB\\*](#)...

```
%% The S3 path must be copy/pasted manually  
  
s3_url = 's3://dandiarchive/blobs/7ee/415/7ee41580-9b0b-44ca-8675-6959ddd8dc33'  
  
nwbfile = nwbRead(s3_url)
```

\* streaming speeds are usually much slower than in Python