

---

# **Allen SDK Documentation**

***Release dev***

**Allen Institute for Brain Science**

**Apr 29, 2020**



---

## Contents

---

<b>1</b>	<b>Install Guide</b>	<b>1</b>
1.1	Quick Start Using Pip . . . . .	1
1.2	Other Distribution Formats . . . . .	1
1.3	Required Dependencies . . . . .	1
1.4	Optional Dependencies . . . . .	2
1.5	Installation with Docker (Optional) . . . . .	2
<b>2</b>	<b>Data Resources</b>	<b>3</b>
2.1	Brain Observatory . . . . .	3
2.2	Cell Types . . . . .	6
2.3	Mouse Connectivity . . . . .	9
2.4	Reference Space . . . . .	11
2.5	API Access . . . . .	11
2.6	Visual Coding – Neuropixels . . . . .	14
<b>3</b>	<b>Models</b>	<b>23</b>
3.1	Generalized LIF Models . . . . .	23
3.2	Biophysical Models . . . . .	29
<b>4</b>	<b>Examples</b>	<b>39</b>
<b>5</b>	<b>Authors</b>	<b>41</b>
<b>6</b>	<b>allensdk package</b>	<b>43</b>
6.1	Subpackages . . . . .	43
6.2	Submodules . . . . .	374
6.3	Module contents . . . . .	374
<b>7</b>	<b>Allen Brain Observatory</b>	<b>375</b>
<b>8</b>	<b>Allen Cell Types Database</b>	<b>377</b>
<b>9</b>	<b>Allen Mouse Brain Connectivity Atlas</b>	<b>379</b>
<b>10</b>	<b>What’s New - 1.7.0 (April 29, 2020)</b>	<b>381</b>
<b>11</b>	<b>What’s New - 1.6.0 (March 23, 2020)</b>	<b>383</b>

<b>12 What's New - 1.5.0 (February 10, 2020)</b>	<b>385</b>
<b>13 Previous Release Notes</b>	<b>387</b>
<b>Bibliography</b>	<b>389</b>
<b>Python Module Index</b>	<b>391</b>
<b>Index</b>	<b>397</b>

# CHAPTER 1

---

## Install Guide

---

This guide is a resource for using the Allen SDK package. It is maintained by the [Allen Institute for Brain Science](#).

The Allen SDK was developed and tested with Python 2.7.13 and Python 3.6.4, installed as part of [Anaconda Python](#) distribution version 4.3.13. We do not guarantee consistent behavior with other Python versions.

### 1.1 Quick Start Using Pip

First ensure you have [pip](#) installed. It is included with the Anaconda distribution.

```
pip install allensdk
```

To uninstall the SDK:

```
pip uninstall allensdk
```

### 1.2 Other Distribution Formats

The Allen SDK is also available from the Github source repository.

### 1.3 Required Dependencies

- [NumPy](#)
- [SciPy](#)
- [matplotlib](#)
- [h5py](#)
- [pandas](#)

- pynrrd
- Jinja2

## 1.4 Optional Dependencies

- pytest
- coverage

## 1.5 Installation with Docker (Optional)

[Docker](#) is an open-source technology for building and deploying applications with a consistent environment including required dependencies. The AllenSDK is not distributed as a Docker image, but example Dockerfiles are available.

1. Ensure you have Docker installed.
2. Use Docker to build one of the images.

Anaconda:

```
docker pull alleninstitute/allensdk
```

Other docker configurations are also available under docker directory in the source repository.

3. Run the docker image:

```
docker run -i -t -p 8888:8888 -v /data:/data alleninstitute/allensdk /bin/bash
cd allensdk
make test
```

4. Start a Jupyter Notebook:

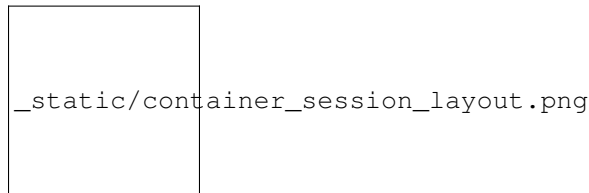
```
cd allensdk/doc_template/examples_root/examples/nb
jupyter-notebook --ip=* --no-browser
```

The Allen SDK features Python code to support data and model access for the Allen Cell Types Database. Resources for other Allen Brain Atlas data resources will come in future updates.

### 2.1 Brain Observatory

The [Allen Brain Observatory](#) is a database of the visually-evoked functional responses of neurons in mouse visual cortex based on 2-photon fluorescence imaging. Characterized responses include orientation tuning, spatial and temporal frequency tuning, temporal dynamics, and spatial receptive field structure.

The data is organized into experiments and experiment containers. An experiment container represents a group of experiments with the same targeted imaging area, imaging depth, and Cre line. The individual experiments within an experiment container have different stimulus protocols, but cover the same imaging field of view.



**Note:** Version 1.3 of scipy fixed an error in its 2 sample Kolmogorov-Smirnoff test implementation. The new version produces more accurate p values for small and medium-sized samples. This change impacts speed tuning analysis p values (as returned by *StimulusAnalysis.get\_speed\_tuning*). If you access precalculated analysis results via *BrainObservatoryCache.get\_ophys\_experiment\_analysis*, you will see values calculated using an older version of scipy's *ks\_2samp*. To access values calculated from the new version, install *scipy*  $\geq 1.3.0$  in your environment and construct a *StimulusAnalysis* object from a *BrainObservatoryNwbDataSet* (as returned by *BrainObservatoryCache.get\_ophys\_experiment\_data*).

**Note:** Data collected after September 2016 uses a new session C stimulus designed to better-characterize spatial receptive fields in higher visual areas. The original locally sparse noise stimulus used 4.65 visual degree pixels. Session C2 broke that stimulus into two separate stimulus blocks: one with 4.65 degree pixels and one with 9.3 degree pixels.

Note that the `stimulus_info` module refers to these as `locally_sparse_noise_4deg` and `locally_sparse_noise_8deg`, respectively.

For more information on experimental design and a data overview, please visit the [Allen Brain Observatory data portal](#).

## 2.1.1 Data Processing

For all data in Allen Brain Observatory, we perform the following processing:

1. Segment cell masks from each experiment's 2-photon fluorescence video
2. Associate cells from experiments belonging to the same experiment container and assign unique IDs
3. Extract each cell's mean fluorescence trace
4. Extract mean fluorescence traces from each cell's surrounding neuropil
5. Demix traces from overlapping ROIs
6. Estimate neuropil-corrected fluorescence traces
7. Compute dF/F
8. Compute stimulus-specific tuning metrics

All traces and masks for segmented cells in an experiment are stored in a Neurodata Without Borders (NWB) file. Stored traces include the raw fluorescence trace, neuropil trace, demixed trace, and dF/F trace. Code for extracting neuropil-corrected fluorescence traces, computing dF/F, and computing tuning metrics is available in the SDK.

**New in June 2017:** Trace demixing is a new addition as of June 2017. All past data was reprocessed using the new demixing algorithm. We have also developed a new module to better characterize a cell's receptive field. Take a look at the [receptive field analysis example notebook](#)

For more information about data processing, please [read the technical whitepapers](#).

## 2.1.2 Getting Started

The Brain Observatory [Jupyter notebook](#) has many code samples to help get started with the available data:

- Download experimental metadata by visual area, imaging depth, and Cre line
- Find cells with specific response properties, like direction tuning
- Download data for an experiment
- Plot raw fluorescence traces, neuropil-corrected traces, and dF/F
- Find the ROI mask for a given cell
- Run neuropil correction
- Get pupil location and size

The code used to analyze and visualize data in the [Allen Brain Observatory data portal](#) is available as part of the SDK. Take a look at this [Jupyter notebook](#) to find out how to:

- Plot cell's response to its preferred stimulus condition
- Compute a cell's on/off receptive field based on the locally sparse noise stimulus

More detailed documentation is available demonstrating how to:

- Read and visualize the stimulus presentation tables in the NWB files
- Understand the layout of Brain Observatory NWB files



- Map previous cell specimen IDs to current cell specimen IDs

### 2.1.3 Precomputed Cell Metrics

A large table of precomputed metrics are available for download to support population analysis and filtering. The table below describes all of the metrics in the table. The `get_cell_specimens()` method will download this table as a list of dictionaries which can be converted to a pandas DataFrame as shown in this [Jupyter notebook](#).

Stimulus	Metric	Field Name
drifting gratings	orientation selectivity	osi_dg
	direction selectivity	dsi_dg
	preferred direction	pref_dir_dg
	preferred temporal frequency	pref_tf_dg
	response p value	p_dg
	global ori. selectivity	g_osi_dg
	global dir. selectivity	g_dsi_dg
	response reliability	reliability_dg
	running modulation	run_mod_dg
	running modulation p value	p_run_mod_dg
	pref. condition mean df/f	peak_dff_dg
	TF discrimination index	tfdi_dg
static gratings	orientation selectivity	osi_sg
	preferred orientation	pref_ori_sg
	preferred spatial frequency	pref_sf_sg
	preferred phase	pref_phase_sg
	mean time to peak response	time_to_peak_sg
	response p value	p_sg
	global ori. selectivity	g_osi_sg
	reponse reliability	reliability_sg
	running modulation	run_mod_sg
	running modulation p value	p_run_mod_sg
	pref. condition mean df/f	peak_dff_ns
	SF discrimination index	sfdi_sg
natural scenes	mean time to peak response	time_to_peak_ns
	preferred scene index	pref_scene_ns
	response p value	p_ns
	image selectivity	image_sel_ns
	running modulation	run_mod_ns
	running modulation p value	p_run_mod_ns
	pref. condition mean df/f	peak_dff_ns
natural movie 1	response reliability (session A)	reliability_nm1_a
	response reliability (session B)	reliability_nm1_b
	response reliability (session C)	reliability_nm1_c
natural movie 2	response reliability	reliability_nm2
natural movie 3	response reliability	reliability_nm3
locally sparse noise	RF area (on subunit)	rf_area_on_lsn
	RF area (off subunit)	rf_area_off_lsn
	RF center (on subunit)	rf_center_on_x, rf_center_on_y
	RF center (off subunit)	rf_center_off_x, rf_center_off_y
	RF chi^2	rf_chi2_lsn
	RF on-off subunit distance	rf_distance_lsn
	RF on-off subunit overlap index	rf_overlap_lsn

## 2.2 Cell Types

The Allen Cell Types data set is a database of mouse and human neuronal cell types based on multimodal characterization of single cells to enable data-driven approaches to classification and is fully integrated with other Allen Brain Atlas resources. The database currently includes:

- **electrophysiology**: whole cell current clamp recordings made from Cre-positive neurons
- **morphology**: 3D bright-field images of the complete structure of neurons from the visual cortex

This page describes how the SDK can be used to access data in the Cell Types Database. For more information, please visit the Cell Types Database [home page](#) and the [API documentation](#).

### 2.2.1 Examples

The Cell Types [Jupyter notebook](#) has many code samples to help get started with analysis:

- Download and plot stimuli and responses from an NWB file for a cell
- Download and plot a cell's morphological reconstruction
- Download and plot precomputed electrophysiology features
- Download precomputed morphology features to a table
- Compute electrophysiology features for a single sweep

### 2.2.2 Cell Types Cache

The `CellTypesCache` class provides a Python interface for downloading data in the Allen Cell Types Database into well known locations so that you don't have to think about file names and directories. The following example demonstrates how to download meta data for all cells with 3D reconstructions, then download the reconstruction and electrophysiology recordings for one of those cells:

```
from allensdk.core.cell_types_cache import CellTypesCache

ctc = CellTypesCache(manifest_file='cell_types/manifest.json')

# a list of cell metadata for cells with reconstructions, download if necessary
cells = ctc.get_cells(require_reconstruction=True)

# open the electrophysiology data of one cell, download if necessary
data_set = ctc.get_ephys_data(cells[0]['id'])

# read the reconstruction, download if necessary
reconstruction = ctc.get_reconstruction(cells[0]['id'])
```

`CellTypesCache` takes care of knowing if you've already downloaded some files and reads them from disk instead of downloading them again. All data is stored in the same directory as the *manifest\_file* argument to the constructor.

### 2.2.3 Feature Extraction

The `EphysFeatureExtractor` class calculates electrophysiology features from cell recordings. `extract_cell_features()` can be used to extract the precise feature values available in the Cell Types Database:

```

from allensdk.core.cell_types_cache import CellTypesCache
from allensdk.ephys.extract_cell_features import extract_cell_features
from collections import defaultdict

# initialize the cache
ctc = CellTypesCache(manifest_file='cell_types/manifest.json')

# pick a cell to analyze
specimen_id = 324257146

# download the ephys data and sweep metadata
data_set = ctc.get_ephys_data(specimen_id)
sweeps = ctc.get_ephys_sweeps(specimen_id)

# group the sweeps by stimulus
sweep_numbers = defaultdict(list)
for sweep in sweeps:
    sweep_numbers[sweep['stimulus_name']].append(sweep['sweep_number'])

# calculate features
cell_features = extract_cell_features(data_set,
                                     sweep_numbers['Ramp'],
                                     sweep_numbers['Short Square'],
                                     sweep_numbers['Long Square'])

```

## 2.2.4 File Formats

This section provides a short description of the file formats used for Allen Cell Types data.

### Morphology SWC Files

Morphological neuron reconstructions are available for download as SWC files. The SWC file format is a white-space delimited text file with a standard set of headers. The file lists a set of 3D neuronal compartments, each of which has:

Column	Data Type	Description
id	string	compartment ID
type	integer	compartment type
x	float	3D compartment position (x)
y	float	3D compartment position (y)
z	float	3D compartment position (z)
radius	float	compartment radius
parent	string	parent compartment ID

Comment lines begin with a '#'. Reconstructions in the Allen Cell Types Database can contain the following compartment types:

Type	Description
0	unknown
1	soma
2	axon
3	basal dendrite
4	apical dendrite

The Allen SDK comes with a `swc` Python module that provides helper functions and classes for manipulating SWC files. Consider the following example:

```
import allensdk.core.swc as swc

# if you ran the examples above, you will have a reconstruction here
file_name = 'cell_types/specimen_485909730/reconstruction.swc'
morphology = swc.read_swc(file_name)

# subsample the morphology 3x. root, soma, junctions, and the first child of the root_
↳are preserved.
sparse_morphology = morphology.sparsify(3)

# compartments in the order that they were specified in the file
compartment_list = sparse_morphology.compartment_list

# a dictionary of compartments indexed by compartment id
compartments_by_id = sparse_morphology.compartment_index

# the root soma compartment
soma = morphology.soma

# all compartments are dictionaries of compartment properties
# compartments also keep track of ids of their children
for child in morphology.children_of(soma):
    print(child['x'], child['y'], child['z'], child['radius'])
```

## Neurodata Without Borders

The electrophysiology data collected in the Allen Cell Types Database is stored in the [Neurodata Without Borders](#) (NWB) file format. This format, created as part of the [NWB initiative](#), is designed to store a variety of neurophysiology data, including data from intra- and extracellular electrophysiology experiments, optophysiology experiments, as well as tracking and stimulus data. It has a defined schema and metadata labeling system designed so software tools can easily access contained data.

The Allen SDK provides a basic Python class for extracting data from Allen Cell Types Database NWB files. These files store data from intracellular patch-clamp recordings. A stimulus current is presented to the cell and the cell's voltage response is recorded. The file stores both stimulus and response for several experimental trials, here called "sweeps." The following code snippet demonstrates how to extract a sweep's stimulus, response, sampling rate, and estimated spike times:

```
from allensdk.core.nwb_data_set import NwbDataSet

# if you ran the examples above, you will have a NWB file here
file_name = 'cell_types/specimen_485909730/ephys.nwb'
data_set = NwbDataSet(file_name)

sweep_numbers = data_set.get_sweep_numbers()
sweep_number = sweep_numbers[0]
sweep_data = data_set.get_sweep(sweep_number)

# spike times are in seconds relative to the start of the sweep
spike_times = data_set.get_spike_times(sweep_number)

# stimulus is a numpy array in amps
stimulus = sweep_data['stimulus']
```

(continues on next page)

(continued from previous page)

```
# response is a numpy array in volts
reponse = sweep_data['response']

# sampling rate is in Hz
sampling_rate = sweep_data['sampling_rate']

# start/stop indices that exclude the experimental test pulse (if applicable)
index_range = sweep_data['index_range']
```

## HDF5 Overview

NWB is implemented in [HDF5](#). HDF5 files provide a hierarchical data storage that mirrors the organization of a file system. Just as a file system has directories and files, and HDF5 file has groups and datasets. The best way to understand an HDF5 (and NWB) file is to open a data file in an HDF5 browser. [HDFView](#) is the recommended browser from the makers of HDF5.

There are HDF5 manipulation libraries for many languages and platforms. MATLAB and Python in particular have strong HDF5 support.

## 2.3 Mouse Connectivity

The Allen Mouse Brain Connectivity Atlas consists of high-resolution images of axonal projections targeting different anatomic regions or various cell types using Cre-dependent specimens. Each data set is processed through an informatics data analysis pipeline to obtain spatially mapped quantified projection information.

This page describes how to use the SDK to access experimental projection data and metadata. For more information, please visit the Connectivity Atlas [home page](#) and the [API documentation](#)

### 2.3.1 Structure-Level Projection Data

All AAV projection signal in the Allen Mouse Connectivity Atlas has been registered to the expert-annotated Common Coordinate Framework (CCF) and summarized to structures in the adult mouse structure ontology. Most commonly used for analysis are measures of the density of projection signal in all brain areas for every experiment. This data is available for download and is described in more detail on the [structure unionizes page](#).

### 2.3.2 Voxel-Level Projection Data

The CCF-registered AAV projection signal is also available for download as a set of 3D volumes for each experiment. The following data volumes are available for download:

- **projection density**: sum of detected projection pixels / sum of all pixels in voxel
- **injection\_fraction**: fraction of pixels belonging to manually annotated injection site
- **injection\_density**: density of detected projection pixels within the manually annotated injection site
- **data\_mask**: binary mask indicating if a voxel contains valid data. Only valid voxels should be used for analysis.

### 2.3.3 Code Examples

The Mouse Connectivity Jupyter notebook has many code samples to help get started with analysis:

- Download experimental metadata by injection structure and transgenic line
- Download projection signal statistics at a structure level
- Build a structure-to-structure matrix of projection signal values
- Download and visualize gridded projection signal volumes

### 2.3.4 Mouse Connectivity Cache

The `MouseConnectivityCache` class saves all of the data you can download via the `MouseConnectivityApi` in well known locations so that you don't have to think about file names and directories. It also takes care of knowing if you've already downloaded some files and reads them from disk instead of downloading them again. The following example demonstrates how to download meta data for all experiments with injections in the isocortex and download the projection density volume for one of them:

```
from allensdk.core.mouse_connectivity_cache import MouseConnectivityCache

# tell the cache class what resolution (in microns) of data you want to download
mcc = MouseConnectivityCache(resolution=25)

# use the structure tree class to get information about the isocortex structure
structure_tree = mcc.get_structure_tree()
isocortex_id = structure_tree.get_structures_by_name(['Isocortex'])[0]['id']

# a list of dictionaries containing metadata for non-Cre experiments
experiments = mcc.get_experiments(file_name='non_cre.json',
                                  injection_structure_ids=[isocortex_id])

# download the projection density volume for one of the experiments
pd = mcc.get_projection_density(experiments[0]['id'])
```

### 2.3.5 File Formats

This section provides a short description of the file formats used for data in the Allen Mouse Connectivity Atlas.

#### NRRD Files

All of the volumetric data in the connectivity atlas are stored as **NRRD (Nearly Raw Raster Data)** files. A NRRD file consists of a short ASCII header followed by a binary array of data values.

To read these in Python, we recommend the `pynrrd` package. Usage is straightforward:

```
import nrrd

file_name = 'mouse_connectivity/experiment_644250774/projection_density_25.nrrd'
data_array, metadata = nrrd.read(file_name)
```

## 2.4 Reference Space

Allen Institute atlases and data are registered, when possible, to one of several common reference spaces. Working in such a space allows you to easily compare data across subjects and experimental modalities.

This page documents how to use the Allen SDK to interact with a reference space. For more information and a list of reference spaces, see the [atlas drawings and ontologies API documentation](#) and the [3D reference models API documentation](#). For details about the construction of the Common Coordinate Framework space, see the [CCFv3 whitepaper](#).

### 2.4.1 Structure Tree

Brain structures in our reference spaces are arranged in trees. The leaf nodes of the tree describe the very fine anatomical divisions of the space, while nodes closer to the root correspond to gross divisions. The `StructureTree` class provides an interface for interacting with a structure tree.

To download a structure tree, use the `allensdk.api.queries.ontologies_api.OntologiesApi` class as seen in [this example](#)

### 2.4.2 Annotation Volumes

An annotation volume is a 3d raster image that segments the reference space into structures. Each voxel in the annotation volume is assigned an integer value that describes the finest structure to which that point in space definitely belongs.

To download a nrrd formatted annotation volume at a specified isometric resolution, use the `allensdk.api.queries.mouse_connectivity_api` class. There is [an example](#) in the notebook.

### 2.4.3 ReferenceSpace Class

The `allensdk.core.reference_space.ReferenceSpace` class contains methods for working with our reference spaces. Some use cases might include:

- Building an indicator mask for one or more structures
- Viewing the annotation
- Querying the structure graph

Please see the [example notebook](#) for more code samples.

## 2.5 API Access

The `allensdk.api` package is designed to help retrieve data from the [Allen Brain Atlas API](#). `api` contains methods to help formulate API queries and parse the returned results. There are several pre-made subclasses available that provide pre-made queries specific to certain data sets. Currently there are several subclasses in Allen SDK:

- `CellTypesApi`: data related to the Allen Cell Types Database
- `BiophysicalApi`: data related to biophysical models
- `GlifApi`: data related to GLIF models
- `AnnotatedSectionDataSetsApi`: search for experiments by intensity, density, pattern, and age

- *GridDataApi*: used to download 3-D expression grid data
- *ImageDownloadApi*: download whole or partial two-dimensional images
- *MouseConnectivityApi*: common operations for accessing the Allen Mouse Brain Connectivity Atlas
- *OntologiesApi*: data about neuroanatomical regions of interest
- *ConnectedServices*: schema of Allen Institute Informatics Pipeline services available through the RmaApi
- *RmaApi*: general-purpose HTTP interface to the Allen Institute API data model and services
- *SvgApi*: annotations associated with images as scalable vector graphics (SVG)
- *SynchronizationApi*: data about image alignment
- *TreeSearchApi*: list ancestors or descendants of structure and specimen trees

## 2.5.1 RMA Database and Service API

One API subclass is the *RmaApi* class. It is intended to simplify constructing an RMA query.

The *RmaApi* is a base class for much of the `allensdk.api.queries` package, but it may be used directly to customize queries or to build queries from scratch.

Often a query will simply request a table of data of one type:

```
from allensdk.api.queries.rma_api import RmaApi

rma = RmaApi()

data = rma.model_query('Atlas',
                       criteria="[name$il'*Mouse*']")
```

This will construct the RMA query url, make the query and parse the resulting JSON into an array of Python dicts with the names, ids and other information about the atlases that can be accessed via the API.

Using the criteria, include and other parameter, specific data can be requested.

```
associations = ''.join(['[id$eq1]',
                       'structure_graph(ontology)', ', ',
                       'graphic_group_labels'])

atlas_data = rma.model_query('Atlas',
                             include=associations,
                             criteria=associations,
                             only=['atlases.id',
                                   'atlases.name',
                                   'atlases.image_type',
                                   'ontologies.id',
                                   'ontologies.name',
                                   'structure_graphs.id',
                                   'structure_graphs.name',
                                   'graphic_group_labels.id',
                                   'graphic_group_labels.name'])
```

Note that a ‘class’ name is used for the first parameter. ‘Association’ names are used to construct the include and criteria parameters nested using parentheses and commas. In the only clause, the ‘table’ form is used, which is generally a plural lower-case version of the class name. The only clause selects specific ‘fields’ to be returned. The schema that includes the classes, fields, associations and tables can be accessed in JSON form using:



```
# http://api.brain-map.org/api/v2/data.json
schema = rma.get_schema()
for entry in schema:
    data_description = entry['DataDescription']
    clz = list(data_description.keys())[0]
    info = list(data_description.values())[0]
    fields = info['fields']
    associations = info['associations']
    table = info['table']
    print("class: %s" % (clz))
    print("fields: %s" % ('.'.join(f['name'] for f in fields)))
    print("associations: %s" % ('.'.join(a['name'] for a in associations)))
    print("table: %s\n" % (table))
```

## 2.5.2 Using Pandas to Process Query Results

When it is difficult to get data in exactly the required form using only an RMA query, it may be helpful to perform additional operations on the client side. The pandas library can be useful for this.

Data from the API can be read directly into a pandas [Dataframe](#) object.

```
import pandas as pd

structures = pd.DataFrame(
    rma.model_query('Structure',
                    criteria='[graph_id$eq1]',
                    num_rows='all'))
```

Indexing subsets of the data (certain columns, certain rows) is one use of pandas: specifically `.loc`:

```
names_and_acronyms = structures.loc[:, ['name', 'acronym']]
```

and [Boolean indexing](#)

```
mea = structures[structures.acronym == 'MEA']
mea_id = mea.iloc[0,:].id
mea_children = structures[structures.parent_structure_id == mea_id]
print(mea_children['name'])
```

[Concatenate](#), [merge](#) and [join](#) are used to add columns or rows:

When an RMA call contains an include clause, the associated data will be represented as a python dict in a single column. The column may be converted to a proper Dataframe and optionally dropped.

```
criteria_string = "structure_sets[name$eq'Mouse Connectivity - Summary']"
include_string = "ontology"
summary_structures = \
    pd.DataFrame(
        rma.model_query('Structure',
                        criteria=criteria_string,
                        include=include_string,
                        num_rows='all'))

ontologies = \
    pd.DataFrame(
        list(summary_structures.ontology)).drop_duplicates()
flat_structures_dataframe = summary_structures.drop(['ontology'], axis=1)
```

Alternatively, it can be accessed using normal python dict and list operations.

```
print(summary_structures.ontology[0]['name'])
```

Pandas Dataframes can be written to a CSV file using `to_csv` and read using `load_csv`.

```
summary_structures[['id',
                    'parent_structure_id',
                    'acronym']].to_csv('summary_structures.csv',
                                      index_label='structure_id')
reread = pd.read_csv('summary_structures.csv')
```

Iteration over a Dataframe of API data can be done in several ways. The `.itertuples` method is one way to do it.

```
for id, name, parent_structure_id in summary_structures[['name',
                                                         'parent_structure_id']].
    .itertuples():
    print("%d %s %d" % (id, name, parent_structure_id))
```

## 2.5.3 Caching Queries on Disk

`wrap()` has several parameters for querying the API, saving the results as CSV or JSON and reading the results as a pandas dataframe.

```
from allensdk.api.cache import Cache

cache_writer = Cache()
do_cache=True
structures_from_api = \
    cache_writer.wrap(rma.model_query,
                     path='summary.csv',
                     cache=do_cache,
                     model='Structure',
                     criteria='[graph_id$eq1]',
                     num_rows='all')
```

If you change `do_cache` to `False` and run the code again it will read the data from disk rather than executing the query.

## 2.6 Visual Coding – Neuropixels

The Visual Coding – Neuropixels project uses high-density extracellular electrophysiology (**Ecephys**) probes to record spikes from a wide variety of regions in the mouse brain. Our experiments are designed to study the activity of the visual cortex and thalamus in the context of passive visual stimulation, but these data can be used to address a wide variety of topics.

Spike-sorted data and metadata are available via the AllenSDK as [Neurodata Without Borders](#) files. However, if you're using the AllenSDK to interact with the data, no knowledge of the NWB data format is required.

### 2.6.1 Getting Started

To jump right in, check out the [quick start guide \(download .ipynb\)](#), which will show you how to download the data, align spikes to a visual stimulus, and decode natural images from neural activity patterns. For a quick summary of experimental design and data access, see the [cheat sheet](#).

If you would like more example code, the [full example notebook \(download .ipynb\)](#) covers all of the ways to access data for each experiment.


Additional tutorials are available on the following topics:

1. [Data access \(download .ipynb\)](#)
2. [Unit quality metrics \(download .ipynb\)](#)
3. [LFP data analysis \(download .ipynb\)](#)
4. [Receptive field mapping \(download .ipynb\)](#)
5. [Optotagging \(download .ipynb\)](#)

For detailed information about the experimental design, data acquisition, and informatics methods, please refer to our [technical whitepaper](#). AllenSDK API documentation is [available here](#).

**A note on terminology:** Throughout the SDK, we refer to neurons as “units,” because we cannot guarantee that all the spikes assigned to one unit actually originate from a single cell. Unlike in two-photon imaging, where you can visualize each neuron throughout the entire experiment, with electrophysiology we can only “see” a neuron when it fires a spike. If a neuron moves relative to the probe, or if it’s far away from the probe, some of its spikes may get mixed together with those from other neurons. Because of this inherent ambiguity, we provide a variety of quality metrics to allow you to find the right units for your analysis. Even highly contaminated units contain potentially valuable information about brain states, so we didn’t want to leave them out of the dataset. But certain types of analysis require more stringent quality thresholds, to ensure that all of the included units are well isolated from their neighbors.

## 2.6.2 Data Processing



`_static/neuropixels_data_processing.png`

Neuropixels probes contain 374 or 383 channels that continuously detect voltage fluctuations in the surrounding neural tissue. Each channel is split into two separate data streams, or “bands,” on the probes. The “spike band” is digitized at 30 kHz, and contains information about action potentials fired by neurons directly adjacent to the probe. The “LFP band” is digitized at 2.5 kHz, and records the low-frequency (<1000 Hz) fluctuations that result from synchronized neural activity over a wider area.

To go from the raw spike-band data to NWB files, we perform the following processing steps:

1. Median-subtraction to remove common-mode noise from the continuous traces
2. High-pass filtering (>150 Hz) and whitening across blocks of 32 channels
3. Spike sorting with [Kilosort2](#), to detect spikes and assign them to individual units
4. Computing the mean waveform for each unit
5. Removing units with artifactual waveforms
6. Computing quality metrics for every unit
7. Computing stimulus-specific tuning metrics

For the LFP band, we:


1. Downsample the signals in space and time (every 4th channel and every 2nd sample)
2. High-pass filter at 0.1 Hz to remove the DC offset from each channel
3. Re-reference to channels outside of the brain to remove common-mode noise

The packaged NWB files contain:

1. Spike times, spike amplitudes, mean waveforms, and quality metrics for every unit
2. Information about the visual stimulus
3. Time series of the mouse's running speed, pupil diameter, and pupil position
4. LFP traces for channels in the brain
5. Experiment metadata

All code for data processing and packaging is available in the [ecephys\\_spike\\_sorting](#) and the `ecephys` section of the AllenSDK.

### 2.6.3 Visual Stimulus Sets



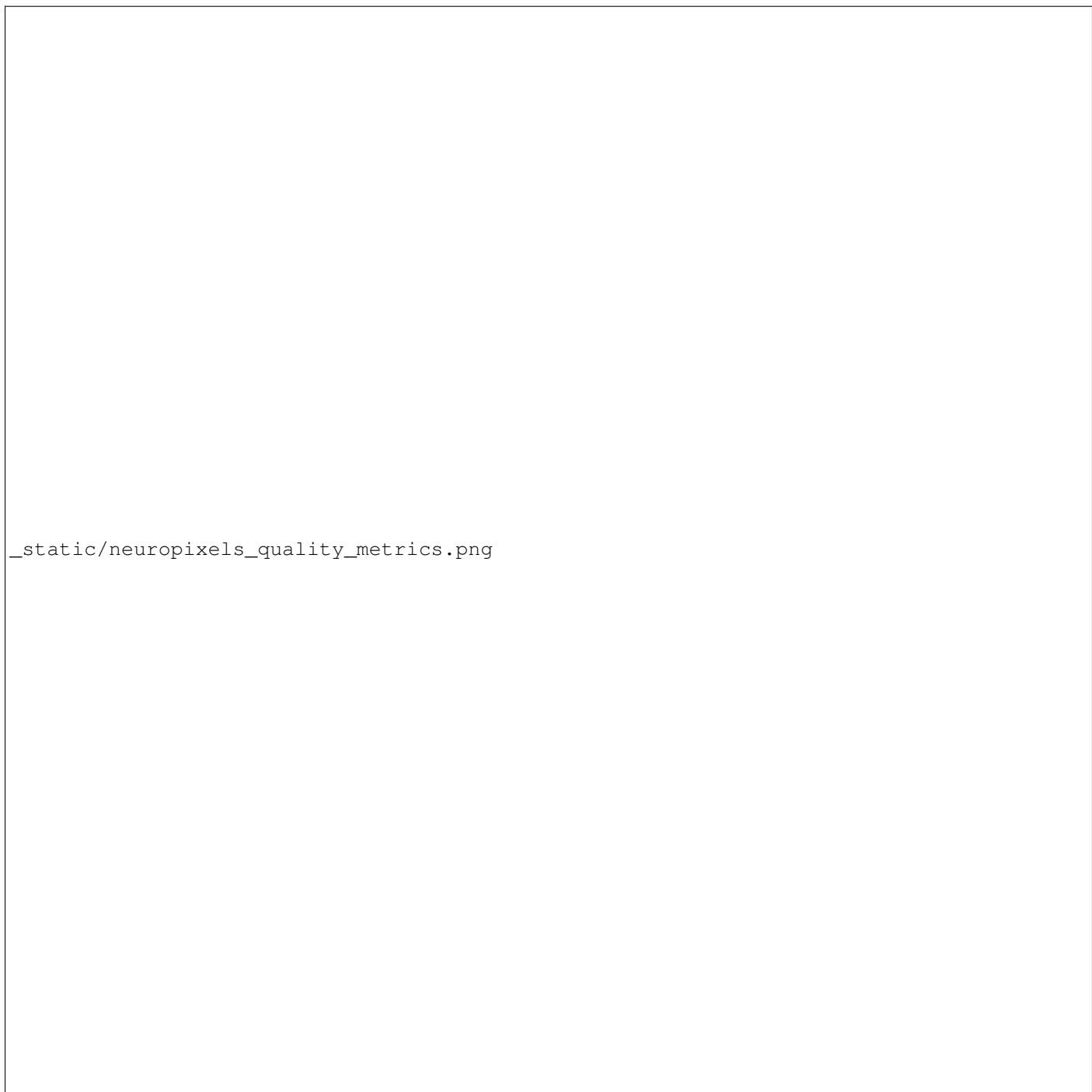
`_static/neuropixels_stimulus_sets.png`

A central aim of the Visual Coding – Neuropixels project is to measure the impact of visual stimuli on neurons throughout the mouse visual system. To that end, all mice viewed one of two possible stimulus sets, known as “Brain Observatory 1.1” or “Functional Connectivity”. Both stimulus sets began with a Gabor stimulus flashed at 81 different locations on the screen, used to map receptive fields of visually responsive units. Next, the mice were shown brief flashes of light or dark, to measure the temporal dynamics of the visual response.

The remainder of the visual stimulus set either consisted of the same stimuli shown in the two-photon experiments

(“Brain Observatory 1.1”), or a subset of those stimuli shown with a higher number of repeats. We also added a dot motion stimulus, to allow us to measure the speed tuning of units across the mouse visual system.

## 2.6.4 Quality Metrics



`_static/neuropixels_quality_metrics.png`

Every NWB file includes a table of quality metrics, which can be used to assess the completeness, contamination, and stability of units in the recording. By default, we won’t show you units below a pre-determined quality threshold; we hide any units that are not present for the whole session (`presence_ratio < 0.95`), that include many contaminating

spikes (`isi_violations > 0.5`), or are likely missing a large fraction of spikes (`amplitude_cutoff > 0.1`). However, even contaminated or incomplete units contain information about brain states, and may be of interest to analyze. Therefore, the complete units table can be accessed via special flags in the AllenSDK.

In general, we do not make a distinction between ‘single-unit’ and ‘multi-unit’ activity. There is no obvious place to draw a boundary in the overall distributions of quality metrics, and setting a strict cutoff (e.g. `isi_violations = 0`) will remove a lot of potentially valuable data. We prefer to leave it up to the end user to decide what level of contamination is tolerable. But that means you need to be aware that different units will have different levels of cleanliness.

It should also be noted that all of these metrics assume that the spike waveform is stable throughout the experiment. Given that the probe drifts, on average, about 40 microns over the course of the ~3 hour recordings, this assumption is almost never valid. The resulting changes in waveform shape can cause a unit’s quality to fluctuate. If you’re unsure about a unit’s quality, it can be helpful to plot its spike amplitudes over time. This can make it obvious if it’s drifting below threshold, or if it contains spikes from multiple neurons.

Documentation on the various quality metrics can be found in the [ecephys\\_spike\\_sorting](#) repository.

For a detailed discussion of the appropriate way to apply each of these metrics, please check out [this tutorial \(download .ipynb\)](#)

## 2.6.5 Precomputed Stimulus Metrics

Tables of precomputed metrics are available for download to support population analysis and filtering. The table below describes all of the available metrics. The `get_unit_analysis_metrics()` method will load this table as a [pandas DataFrame](#).

Stimulus	Metric	Field Name
drifting gratings	preferred orientation	<code>pref_ori_dg</code>
	preferred temporal frequency	<code>pref_tf_dg</code>
	global ori. selectivity	<code>g_osi_dg</code>
	global dir. selectivity	<code>g_dsi_dg</code>
	running modulation	<code>run_mod_dg</code>
	running modulation p-value	<code>p_run_mod_dg</code>
	firing rate	<code>firing_rate_dg</code>
	fano factor	<code>fano_dg</code>
	modulation index	<code>mod_idx_dg</code>
	f1/f0	<code>f1_f0_dg</code>
	lifetime sparseness	<code>lifetime_sparseness_dg</code>
	c50 (contrast tuning stimulus)	<code>c50_dg</code>
static gratings	preferred orientation	<code>pref_ori_sg</code>
	preferred spatial frequency	<code>pref_sf_sg</code>
	preferred phase	<code>pref_phase_sg</code>
	global ori. selectivity	<code>g_osi_sg</code>
	running modulation	<code>run_mod_sg</code>
	running modulation p-value	<code>p_run_mod_sg</code>
	firing rate	<code>firing_rate_sg</code>
	fano factor	<code>fano_sg</code>
	lifetime sparseness	<code>lifetime_sparseness_sg</code>
natural scenes	preferred image index	<code>pref_image_ns</code>
	image selectivity	<code>image_selectivity_ns</code>
	running modulation	<code>run_mod_ns</code>
	running modulation p-value	<code>p_run_mod_ns</code>
	firing rate	<code>firing_rate_ns</code>
	fano factor	<code>fano_factor_ns</code>

Continued on next page



Table 2 – continued from previous page

Stimulus	Metric	Field Name
	lifetime sparseness	lifetime_sparseness_ns
dot motion	preferred speed	pref_speed_dm
	preferred direction	pref_dir_dm
	running modulation	run_mod_dm
	running modulation p-value	p_run_mod_dm
	firing rate	firing_rate_dm
	fano factor	fano_factor_dm
	lifetime sparseness	lifetime_sparseness_dm
full-field flashes	on/off ratio	on_off_ratio_fl
	running modulation	run_mod_fl
	running modulation p-value	p_run_mod_fl
	firing rate	firing_rate_fl
	fano factor	fano_factor_fl
	lifetime sparseness	lifetime_sparseness_fl
gabor	RF area	area_rf
	RF elevation	elevation_rf
	RF azimuth	azimuth_rf
	RF p-value	p_value_rf
	running modulation	run_mod_rf
	running modulation p-value	p_run_mod_rf
	firing rate	firing_rate_rf
	fano factor	fano_factor_rf
	lifetime sparseness	lifetime_sparseness_rf



The Allen SDK currently focuses on models generated from electrophysiology data in the Allen Cell Types Database. There are two classes of models available for download: biophysical models and generalize leaky integrate-and-fire models.

### 3.1 Generalized LIF Models

The Allen Cell Types Database contains Generalized Leaky Integrate and Fire (GLIF) models that simulate the firing behavior of neurons at five levels of complexity. Review the GLIF technical [white paper](#) for details on these models and how their parameters were optimized.

The Allen SDK GLIF simulation module is an explicit time-stepping simulator that evolves a neuron's simulated voltage over the course of an input current stimulus. The module also tracks the neuron's simulated spike threshold and registers action potentials whenever voltage surpasses threshold. Action potentials initiate reset rules that update voltage, threshold, and (optionally) trigger afterspike currents.

The GLIF simulator in this package has a modular architecture that enables users to choose from a number of dynamics and reset rules that update the simulation's voltage, spike threshold, and afterspike currents during the simulation. The GLIF package contains a built-in set of rules, however developers can plug in custom rule implementations provided they follow a simple argument specification scheme.

The Allen SDK GLIF simulator was developed and tested with Python 2.7.9, installed as part of [Anaconda Python](#) distribution version 2.1.0.

The rest of this page provides examples demonstrating how to download models, examples of simulating these models, and general GLIF model documentation.

---

**Note:** the GLIF simulator module is still under heavy development and may change significantly in the future.

---

### 3.1.1 Downloading GLIF Models

There are two ways to download files necessary to run a GLIF model. The first way is to visit <http://celltypes.brain-map.org> and find cells that have GLIF models available for download. The electrophysiology details page for a cell has a neuronal model download link. Specifically:

1. Click ‘More Options +’ and filter for GLIF models.
2. Click the electrophysiology thumbnail for a cell on the right hand panel.
3. Choose a GLIF model from the ‘Show model responses’ dropdown.
4. Scroll down to the model response click ‘Download model’.

One such link (for a simple LIF neuronal model, ID 566302806), would look like this:

```
http://api.brain-map.org/neuronal_model/download/566302806
```

This link returns .zip archive containing the neuron configuration file and sweep metadata required to simulate the model with stimuli applied to the cell. Specifically, the .zip archive will contain:

- **472423251\_neuron\_config.json**: JSON config file for the GLIF model
- **ephys\_sweeps.json**: JSON with metadata for sweeps presented to the cell
- **neuronal\_model.json**: JSON with general metadata for the cell

If you would like to reproduce the model traces seen in the Cell Types Database, you can download an NWB file containing both the stimulus and cell response traces via a ‘Download data’ link on the cell’s electrophysiology page. See the [NWB](#) description section for more details on the NWB file format.

You can also download all of these files, including the cell’s NWB file, using the *GlifApi* class:

```
from allensdk.api.queries.glif_api import GlifApi
from allensdk.core.cell_types_cache import CellTypesCache
import allensdk.core.json_utilities as json_utilities

neuronal_model_id = 566302806

# download model metadata
glif_api = GlifApi()
nm = glif_api.get_neuronal_models_by_id([neuronal_model_id])[0]

# download the model configuration file
nc = glif_api.get_neuron_configs([neuronal_model_id])[neuronal_model_id]
neuron_config = glif_api.get_neuron_configs([neuronal_model_id])
json_utilities.write('neuron_config.json', neuron_config)

# download information about the cell
ctc = CellTypesCache()
ctc.get_ephys_data(nm['specimen_id'], file_name='stimulus.nwb')
ctc.get_ephys_sweeps(nm['specimen_id'], file_name='ephys_sweeps.json')
```

### 3.1.2 Running a GLIF Simulation

To run a GLIF simulation, the most important file you need is the `neuron_config` JSON file. You can use this file to instantiate a simulator and feed in your own stimulus:

```
import allensdk.core.json_utilities as json_utilities
from allensdk.model.glif.glif_neuron import GlifNeuron

# initialize the neuron
neuron_config = json_utilities.read('neuron_config.json')['566302806']
neuron = GlifNeuron.from_dict(neuron_config)

# make a short square pulse. stimulus units should be in Amps.
stimulus = [ 0.0 ] * 100 + [ 10e-9 ] * 100 + [ 0.0 ] * 100

# important! set the neuron's dt value for your stimulus in seconds
neuron.dt = 5e-6

# simulate the neuron
output = neuron.run(stimulus)

voltage = output['voltage']
threshold = output['threshold']
spike_times = output['interpolated_spike_times']
```

**Note:** The GLIF simulator does not simulate during action potentials. Instead it inserts NaN values for a fixed number of time steps when voltage surpasses threshold. The simulator skips `neuron.spike_cut_length` time steps after voltage surpasses threshold.

To reproduce the model's traces displayed on the Allen Cell Types Database web page, the Allen SDK provides the `allensdk.core.model.glif.simulate_neuron` module for simulating all sweeps presented to a cell and storing them in the NWB format:

```
import allensdk.core.json_utilities as json_utilities

from allensdk.model.glif.glif_neuron import GlifNeuron
from allensdk.model.glif.simulate_neuron import simulate_neuron

neuron_config = json_utilities.read('neuron_config.json')['566302806']
ephys_sweeps = json_utilities.read('ephys_sweeps.json')
ephys_file_name = 'stimulus.nwb'

neuron = GlifNeuron.from_dict(neuron_config)

sweep_numbers = [ s['sweep_number'] for s in ephys_sweeps if s['stimulus_units'] ==
↳ 'Amps' ]
sweep_numbers = sweep_numbers[:1] # for the sake of a speedy example, just run the_
↳ first one
simulate_neuron(neuron, sweep_numbers, ephys_file_name, ephys_file_name, 0.05)
```

**Warning:** These stimuli are sampled at a very high resolution (200kHz), and a given cell can have many sweeps. This process can take over an hour.

The `simulate_neuron` function call simulates all sweeps in the NWB file. Because the same NWB file is being used for both input and output, the cell's response traces will be overwritten as stimuli are simulated. `simulate_neuron` optionally accepts a value which will be used to overwrite these NaN values generated during action potentials (in this case 0.05 Volts).

If you would like to run a single sweep instead of all sweeps, try the following:

```
import allensdk.core.json_utilities as json_utilities
from allensdk.model.glif.glif_neuron import GlifNeuron
from allensdk.core.nwb_data_set import NwbDataSet

neuron_config = json_utilities.read('neuron_config.json')['566302806']
ephys_sweeps = json_utilities.read('ephys_sweeps.json')
ephys_file_name = 'stimulus.nwb'

# pull out the stimulus for the current-clamp first sweep
ephys_sweep = next( s for s in ephys_sweeps
                    if s['stimulus_units'] == 'Amps' )
ds = NwbDataSet(ephys_file_name)
data = ds.get_sweep(ephys_sweep['sweep_number'])
stimulus = data['stimulus']

# initialize the neuron
# important! update the neuron's dt for your stimulus
neuron = GlifNeuron.from_dict(neuron_config)
neuron.dt = 1.0 / data['sampling_rate']

# simulate the neuron
output = neuron.run(stimulus)

voltage = output['voltage']
threshold = output['threshold']
spike_times = output['interpolated_spike_times']
```

---

**Note:** The dt value provided in the downloadable GLIF neuron configuration files does not correspond to the sampling rate of the original stimulus. Stimuli were subsampled and filtered for parameter optimization. Be sure to overwrite the neuron's dt with the correct sampling rate.

---

If you would like to plot the outputs of this simulation using numpy and matplotlib, try:

```
import numpy as np
import matplotlib.pyplot as plt

voltage = output['voltage']
threshold = output['threshold']
interpolated_spike_times = output['interpolated_spike_times']
spike_times = output['interpolated_spike_times']
interpolated_spike_voltages = output['interpolated_spike_voltage']
interpolated_spike_thresholds = output['interpolated_spike_threshold']
grid_spike_indices = output['spike_time_steps']
grid_spike_times = output['grid_spike_times']
after_spike_currents = output['AScurrents']

# create a time array for plotting
time = np.arange(len(stimulus))*neuron.dt

plt.figure(figsize=(10, 10))

# plot stimulus
plt.subplot(3,1,1)
plt.plot(time, stimulus)
plt.xlabel('time (s)')
```

(continues on next page)

(continued from previous page)

```

plt.ylabel('current (A)')
plt.title('Stimulus')

# plot model output
plt.subplot(3,1,2)
plt.plot(time, voltage, label='voltage')
plt.plot(time, threshold, label='threshold')

if grid_spike_indices is not None:
    plt.plot(interpolated_spike_times, interpolated_spike_voltages, 'x',
             label='interpolated spike')

    plt.plot((grid_spike_indices-1)*neuron.dt, voltage[grid_spike_indices-1], '.',
             label='last step before spike')

plt.xlabel('time (s)')
plt.ylabel('voltage (V)')
plt.legend(loc=3)
plt.title('Model Response')

# plot after spike currents
plt.subplot(3,1,3)
for ii in range(np.shape(after_spike_currents)[1]):
    plt.plot(time, after_spike_currents[:,ii])
plt.xlabel('time (s)')
plt.ylabel('current (A)')
plt.title('After Spike Currents')

plt.tight_layout()
plt.show()

```

**Note:** There both interpolated spike times and grid spike times. The grid spike is the first time step where the voltage is higher than the threshold. Note that if you try to plot the voltage at the grid spike indices the output will be NaN. The interpolated spike is the calculated intersection of the threshold and voltage between the time steps.

### 3.1.3 GLIF Configuration

Instances of the *GlifNeuron* class require many parameters for initialization. Fixed neuron parameters are stored directly as properties on the class instance:

Parameter	Description	Units	Type
EI	resting potential	Volts	float
dt	time duration of each simulation step	seconds	float
R_input	input resistance	Ohms	float
C	capacitance	Farads	float
asc_vector	afterspike current coefficients	Amps	np.array
spike_cut_length	spike duration	time steps	int
th_inf	instantaneous threshold	Volts	float
th_adapt	adapted threshold	Volts	float

Some of these fixed parameters were optimized to fit Allen Cell Types Database electrophysiology data. Optimized coefficients for these parameters are stored by name in the `neuron.coeffs` dictionary. For more details on which

parameters were optimized, please see the technical [white paper](#).

The `GlifNeuron` class has six methods that can be customized: three rules for updating voltage, threshold, and afterspike currents during the simulation; and three rules for updating those values when a spike is detected (voltage surpasses threshold).

Method Type	Description
<code>voltage_dynamics_method</code>	Update simulation voltage for the next time step.
<code>threshold_dynamics_method</code>	Update simulation threshold for the next time step.
<code>AScurrent_dynamics_method</code>	Update afterspike current coefficients for the next time step.
<code>voltage_reset_method</code>	Reset simulation voltage after a spike occurs.
<code>threshold_reset_method</code>	Reset simulation threshold after a spike occurs.
<code>AScurrent_reset_method</code>	Reset afterspike current coefficients after a spike occurs.

The GLIF neuron configuration files available from the Allen Brain Atlas API use built-in methods, however you can supply your own custom method if you like:

```
# define your own custom voltage reset rule
# this one linearly scales the input voltage
def custom_voltage_reset_rule(neuron, voltage_t0, custom_param_a, custom_param_b):
    return custom_param_a * voltage_t0 + custom_param_b

# initialize a neuron from a neuron config file
neuron_config = json_utilities.read('neuron_config.json')['566302806']
neuron = GlifNeuron.from_dict(neuron_config)

# configure a new method and overwrite the neuron's old method
method = neuron.configure_method('custom', custom_voltage_reset_rule,
                                { 'custom_param_a': 0.1, 'custom_param_b': 0.0 })
neuron.voltage_reset_method = method

output = neuron.run(stimulus)
```

Notice that the function is allowed to take custom parameters (here `custom_param_a` and `custom_param_b`), which are configured on method initialization from a dictionary. For more details, see the documentation for the `GlifNeuron` and `GlifNeuronMethod` classes.

### 3.1.4 Built-in Dynamics Rules

The job of a dynamics rule is to describe how the simulator should update the voltage, spike threshold, and afterspike currents of the simulator at a given simulation time step.

#### Voltage Dynamics Rules

These methods update the output voltage of the simulation. They all expect a voltage, afterspike current vector, and current injection value to be passed in by the `GlifNeuron`. All other function parameters must be fixed using the `GlifNeuronMethod` class. They all return an updated voltage value.

```
allensdk.model.glif.glif_neuron_methods.dynamics_voltage_linear_forward_euler()
```

#### Threshold Dynamics Rules

These methods update the spike threshold of the simulation. They all expect the current threshold and voltage values of the simulation to be passed in by the `GlifNeuron`. All other function parameters must be fixed using the `GlifNeuronMethod` class. They all return an updated threshold value.

```
allensdk.model.glif.glif_neuron_methods.dynamics_threshold_three_components_exact()
```



```
allensdk.model.glif.glif_neuron_methods.dynamics_threshold_spike_component()
allensdk.model.glif.glif_neuron_methods.dynamics_threshold_inf()
```

### Afterspike Current Dynamics Rules

These methods expect current afterspike current coefficients, current time step, and time steps of all previous spikes to be passed in by the GlifNeuron. All other function parameters must be fixed using the GlifNeuronMethod class. They all return an updated afterspike current array.

```
allensdk.model.glif.glif_neuron_methods.dynamics_AScurrent_exp()
allensdk.model.glif.glif_neuron_methods.dynamics_AScurrent_none()
```

## 3.1.5 Built-in Reset Rules

The job of a reset rule is to describe how the simulator should update the voltage, spike threshold, and afterspike currents of the simulator after the simulator has detected that the simulated voltage has surpassed threshold.

### Voltage Reset Rules

These methods update the output voltage of the simulation after voltage has surpassed threshold. They all expect a voltage to be passed in by the GlifNeuron. All other function parameters must be fixed using the GlifNeuronMethod class. They all return an updated voltage value.

```
allensdk.model.glif.glif_neuron_methods.reset_voltage_zero()
allensdk.model.glif.glif_neuron_methods.reset_voltage_v_before()
```

### Threshold Reset Rules

These methods update the spike threshold of the simulation after a spike has been detected. They all expect the current threshold and the reset voltage value of the simulation to be passed in by the GlifNeuron. All other function parameters must be fixed using the GlifNeuronMethod class. They all return an updated threshold value.

```
allensdk.model.glif.glif_neuron_methods.reset_threshold_inf()
allensdk.model.glif.glif_neuron_methods.reset_threshold_three_components()
```

### Afterspike Reset Rules

These methods expect current afterspike current coefficients to be passed in by the GlifNeuron. All other function parameters must be fixed using the GlifNeuronMethod class. They all return an updated afterspike current array.

```
allensdk.model.glif.glif_neuron_methods.reset_AScurrent_none()
allensdk.model.glif.glif_neuron_methods.reset_AScurrent_sum()
```

## 3.2 Biophysical Models

The Allen Cell Types Database contains biophysical models that characterize the firing behavior of neurons measured in slices through current injection by a somatic whole-cell patch clamp electrode. These models contain a set of 10 active conductances placed at the soma and use the reconstructed 3D morphologies of the modeled neurons. The biophysical modeling [technical white paper](#) contains details on the specific construction of these models and the optimization of the model parameters to match the experimentally-recorded firing behaviors.

The biophysical models are run with the [NEURON](#) simulation environment. The Allen SDK package contains libraries that assist in downloading and setting up the models available on the Allen Institute web site for users to run using NEURON. The examples and scripts provided run on Linux using the bash shell.

### 3.2.1 Prerequisites

You must have NEURON with the Python interpreter enabled and the Allen SDK installed.

The Allen Institute perisomatic biophysical models were generated using NEURON [version v7.4.rel-1370](#). Instructions for compiling NEURON with the Python interpreter are available from the NEURON team under the heading [Installation with Python as an alternative interpreter](#). The Allen SDK is compatible with Python version 2.7.9, included in the Anaconda 2.1.0 distribution.

Instructions for optional [Docker installation](#) are also available.

---

**Note:** Building and installing NEURON with the Python wrapper enabled is not always easy. This page targets users that have a background in NEURON usage and installation.

---

### 3.2.2 Downloading Biophysical Models

There are two ways to download files necessary to run a biophysical model. The first way is to visit <http://celltypes.brain-map.org> and find cells that have biophysical models available for download. The electrophysiology details page for a cell has a neuronal model download link. Specifically:

1. Click ‘More Options+’
2. Check ‘Models -> Biophysical - perisomatic’ or ‘Biophysical - all active’
3. Use the Filters, Cell Location and Cell Feature Filters to narrow your results.
4. Click on a Cell Summary to view the Mouse Experiment Electrophysiology.
5. Click the “download data” link to download the NWB stimulus and response file.
6. Click “show model response” and select ‘Biophysical - perisomatic’ or ‘Biophysical - all active’.
7. Scroll down and click the ‘Biophysical - perisomatic’ or ‘Biophysical - all active’ “download model” link.

This may also be done programmatically. The neuronal model id can be found to the left of the corresponding ‘Biophysical - perisomatic’ or ‘Biophysical - all active’ “download model” link.

```
from allensdk.api.queries.biophysical_api import \
    BiophysicalApi

bp = BiophysicalApi()
bp.cache_stimulus = True # change to False to not download the large stimulus NWB file
neuronal_model_id = 472451419 # get this from the web site as above
bp.cache_data(neuronal_model_id, working_directory='neuronal_model')
```

More help can be found in the [online help](#) for the Allen Cell Types Database web application.

### 3.2.3 Directory Structure

The structure of the directory created looks like this. It includes stimulus files, model parameters, morphology, cellular mechanisms and application configuration.

```

neuronal_model
|-- manifest.json
|-- 472451419_fit.json
|-- Nr5a1-Cre_Ai14_IVSCC_-169248.04.02.01.nwb
|-- Nr5a1-Cre_Ai14_IVSCC_-169248.04.02.01_403165543_m.swc
|-- modfiles
|   |--CaDynamics.mod
|   |--Ca_HVA.mod
|   |--Ca_LVA.mod
|   |--Ih.mod
|   `--...etc.
|
|--x86_64
`---work

```

### 3.2.4 Running the Simulation (Linux shell prompt)

All of the sweeps available from the web site are included in manifest.json and will be run by default. This can take some time.

```

cd neuronal_model
nrnivmodl ./modfiles # compile the model (only needs to be done once)
python -m allensdk.model.biophysical.runner manifest.json

```

### 3.2.5 Selecting a Specific Sweep

The sweeps are listed in manifest.json. You can remove all of the sweep numbers that you do not want run.

### 3.2.6 Simulation Main Loop

The top level script is in the `run()` method of the `allensdk.model.biophysical.runner` module. The implementation of the method is discussed here step-by-step:

First configure NEURON based on the configuration file, which was read in from the command line at the very bottom of the script.

`run()`:

```

# configure NEURON -- this will infer model type (perisomatic vs. all-active)
utils = Utils.create_utils(description)
h = utils.h

```

The next step is to get the path of the morphology file and pass it to NEURON.

```

# configure model
manifest = description.manifest
morphology_path = description.manifest.get_path('MORPHOLOGY')
utils.generate_morphology(morphology_path.encode('ascii', 'ignore'))
utils.load_cell_parameters()

```

Then read the stimulus and recording configuration and configure NEURON

```
# configure stimulus and recording
stimulus_path = description.manifest.get_path('stimulus_path')
nwb_out_path = manifest.get_path("output")
output = NwbDataSet(nwb_out_path)
run_params = description.data['runs'][0]
sweeps = run_params['sweeps']
junction_potential = description.data['fitting'][0]['junction_potential']
mV = 1.0e-3
```

Loop through the stimulus sweeps and write the output.

```
# run sweeps
for sweep in sweeps:
    utils.setup_iclamp(stimulus_path, sweep=sweep)
    vec = utils.record_values()

    h.finitialize()
    h.run()

    # write to an NWB File
    output_data = (numpy.array(vec['v']) - junction_potential) * mV
    output.set_sweep(sweep, None, output_data)
```

### 3.2.7 Customization

Much of the code in the perisomatic simulation is not core Allen SDK code. The `runner.py` script largely reads the configuration file and calls into methods in the `Utils` class. `Utils` is a subclass of the `HocUtils` class, which provides access to objects in the NEURON package. The various methods called by the `runner.py` script are implemented here, including: `generate_morphology()`, `load_cell_parameters()`, `setup_iclamp()`, `read_stimulus()` and `record_values()`.

`Utils`:

```
from allensdk.model.biophys_sim.neuron.hoc_utils import HocUtils

.....

class Utils(HocUtils):
    .....

    def __init__(self, description):
        super(Utils, self).__init__(description)
    .....
```

To create a biophysical model using your own software or data, simply model your directory structure on one of the downloaded simulations or one of the examples below. Add your own `runner.py` and `utils.py` module to the simulation directory.

Compile the `.mod` files using NEURON's `nrnivmodl` command (Linux shell):

```
nrnivmodl modfiles
```

Then call your runner script directly, passing in the manifest file to your script:

```
python runner.py manifest.json
```

The output from your simulation and any intermediate files will go in the work directory.

### 3.2.8 Examples

A minimal example (`simple_example.tgz`) and a multicell example (`multicell_example.tgz`) are available to download as a starting point for your own projects.

Each example provides its own `utils.py` file along with a main script (Linux shell) and supporting configuration files.

`simple_example.tgz`:

```
tar xvzf simple_example.tgz
cd simple
nrnivmodl modfiles
python simple.py
```

`multicell_example.tgz`:

```
tar xvzf multicell_example.tgz
cd multicell
nrnivmodl modfiles
python multi.py
python multicell_diff.py
```

### 3.2.9 Exporting Output to Text Format or Image

This is an example of using the AllenSDK to save a response voltage to other formats.

```
from allensdk.core.dat_utilities import \
    DatUtilities
from allensdk.core.nwb_data_set import \
    NwbDataSet
import numpy as np
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

nwb_file = '313862020.nwb'
sweep_number = 52
dat_file = '313862020_%d.dat' % (sweep_number)

nwb = NwbDataSet(nwb_file)
sweep = nwb.get_sweep(sweep_number)

# read v and t as numpy arrays
v = sweep['response']
dt = 1.0e3 / sweep['sampling_rate']
num_samples = len(v)
t = np.arange(num_samples) * dt

# save as text file
data = np.transpose(np.vstack((t, v)))
with open (dat_file, "w") as f:
    np.savetxt(f, data)

# save image using matplotlib
fig, ax = plt.subplots(nrows=1, ncols=1)
ax.plot(t, v)
```

(continues on next page)

(continued from previous page)

```
ax.set_title("Sweep %s" % (sweep_number))
fig.savefig('out.png')
```

### 3.2.10 Model Description Files

#### Basic Structure

A model description file is simply a JSON object with several sections at the top level and an array of JSON objects within each section.

```
{
  "cell_section": [
    {
      "name": "cell 1",
      "shape": "pyramidal",
      "position": [ 0.1, 0.2, 0.3 ]
    },
    {
      "name": "cell 2",
      "shape": "glial",
      "position": [ 0.1, 0.2, 0.3 ]
    }
  ],
  "extra": [
    { "what": "wood",
      "who": "woodchuck"
    }
  ]
}
```

Even if a section contains no objects or only one object the array brackets must be present.

#### Objects Within Sections

While no restrictions are enforced on what kinds of objects are stored in a section, some rules of thumb make the file easier to work with.

1. All objects within a section are the same structure. Common operations on a section are to display it as a table, iterate over it, load from or write to a spreadsheet or csv file. These operations are all easier if the section is fairly homogeneous.
2. Objects are not deeply nested. While some shallow nesting is often useful, deep nesting such as a tree structure is not recommended. It makes interoperability with other tools and data formats more difficult.
3. Arrays are allowed, though they should not be deeply nested either.
4. Object member values should be literals. Do not use pickled classes, for example.

#### Comment Lines

The JSON specification does not allow comments. However, the Allen SDK library applies a preprocessing stage to remove C++-style comments, so they can be used in description files.

Multi-line comments should be surrounded by `/* */` and single-line comments start with `//`. Commented description files will not be recognized by strict json parsers unless the comments are stripped.

commented.json:

```
{
  /*
   * multi-line comment
   */
  "section1": [
    {
      "name": "simon" // single line comment
    }
  ]
}
```

## Split Description Files by Section

A model description can be split into multiple files by putting some sections in one file and other sections into another file. This can be useful if you want to put a topology of cells and connections in one file and experimental conditions and stimulus in another file. The resulting structure in memory will behave the same way as if the files were not split. This allows a small experiment to be described in a single file and large experiments to be more modular.

cells.json:

```
{
  "cell_section": [
    {
      "name": "cell 1",
      "shape": "pyramidal",
      "position": [ 0.1, 0.2, 0.3 ]
    },
    {
      "name": "cell 2",
      "shape": "glial",
      "position": [ 0.1, 0.2, 0.3 ]
    }
  ]
}
```

extras.json:

```
{
  "extra": [
    {
      "what": "wood",
      "who": "woodchuck"
    }
  ]
}
```

## Split Sections Between Description Files

If two description files containing the same sections are combined, the resulting description will contain objects from both files. This feature allows sub-networks to be described in separate files. The sub-networks can then be composed into a larger network with an additional description of the interconnections.

network1.json:

```
/* A self-contained sub-network */
{
  "cells": [
    { "name": "cell1" },
    { "name": "cell2" }
  ],
  /* intra-network connections */
  "connections": [
    { "source": "cell1", "target" : "cell2" }
  ]
}
```

network2.json:

```
/* Another self-contained sub-network */
{
  "cells": [
    { "name": "cell3" },
    { "name": "cell4" }
  ],
  "connections": [
    { "source": "cell3", "target" : "cell4" }
  ]
}
```

interconnect.json:

```
{
  // the additional connections needed to
  // connect the network1 and network2
  // into a ring topology.
  "connections": [
    { "source": "cell2", "target": "cell3" },
    { "source": "cell4", "target": "cell1" }
  ]
}
```

### 3.2.11 Resource Manifest

JSON has many advantages. It is widely supported, readable and easy to parse and edit. As data sets get larger or specialized those advantages diminish. Large or complex models and experiments generally need more than a single model description file to completely describe an experiment. A manifest file is a way to describe all of the resources needed within the Allen SDK description format itself.

The manifest section is named “manifest” by default, though it is configurable. The objects in the manifest section each specify a directory, file, or file pattern. Files and directories may be organized in a parent-child relationship.

#### A Simple Manifest

This is a simple manifest file that specifies the BASEDIR directory using “.”, meaning the current directory:

```
{
  "manifest": [
```

(continues on next page)



(continued from previous page)

```

    {
      "key": "BASEDIR",
      "type": "dir",
      "spec": "."
    }
  ] }
}
```

## Parent Child Relationships

Adding the optional “parent\_key” member to a manifest object creates a parent-child relation. In this case WORKDIR will be found in “./work”:

```

{
  "manifest": [
    {
      "key": "BASEDIR",
      "type": "dir",
      "spec": "."
    },
    {
      "key": "WORKDIR",
      "type": "dir",
      "spec": "/work",
      "parent_key": "BASEDIR"
    }
  ] }
}
```

## File Spec Patterns

Files can be specified using the type “file” instead of “dir”. If a sequence of many files is needed, the spec may contain patterns to indicate where the sequence number (%d) or string (%s) will be interpolated:

```

{
  "manifest": [
    {
      "key": "BASEDIR",
      "type": "dir",
      "spec": "."
    },
    {
      "key": "voltage_out_cell_path",
      "type": "file",
      "spec": "v_out-cell-%d.dat",
      "parent_key": "BASEDIR"
    }
  ] }
}
```

## Split Manifest Files

Manifest files can be split like any description file. This allows the specification of a general directory structure in a shared file and specific files in a separate configuration (i.e. stimulus and working directory)

## Extensions

To date, manifest description files have not been used to reference URLs that provide model data, but it is a planned future use case.

### 3.2.12 Further Reading

- [NEURON](#)
- [Python](#)
- [JSON](#)

## CHAPTER 4

---

### Examples

---

Take a look at the table below for a list of SDK example notebooks and scripts.

Description	Link
Introduction to the Mouse Connectivity Atlas	<a href="#">Jupyter notebook (download .ipynb)</a>
Introduction to the Cell Types Database	<a href="#">Jupyter notebook (download .ipynb)</a>
Introduction to the Brain Observatory	<a href="#">Jupyter notebook (download .ipynb)</a>
Brain Observatory Stimulus Manipulation	<a href="#">Jupyter notebook (download .ipynb)</a>
Brain Observatory Tuning Analysis	<a href="#">Jupyter notebook (download .ipynb)</a>
Brain Observatory Receptive Field Analysis	<a href="#">Jupyter notebook (download .ipynb)</a>
Brain Observatory Cell Specimen ID Mapping	<a href="#">Jupyter notebook (download .ipynb)</a>
Brain Observatory Monitor	<a href="#">Jupyter notebook (download .ipynb)</a>
Dynamic Brain Workshop 2015 experiment detail	<a href="#">Jupyter notebook (download .ipynb)</a>
Stimulating a biophysical model with a square pulse	<a href="#">Jupyter notebook (download .ipynb)</a>
Using a Reference Space	<a href="#">Jupyter notebook (download .ipynb)</a>
Downloading Images	<a href="#">Jupyter notebook (download .ipynb)</a>
Visual Coding Neuropixels Quick Start	<a href="#">Jupyter notebook (download .ipynb)</a>
Visual Coding Neuropixels Reference	<a href="#">Jupyter notebook (download .ipynb)</a>



## CHAPTER 5

---

### Authors

---

Michael Buice @mabuice  
Nicholas Cain @nicain  
Tom Chartrand @tmchartrand  
Kael Dai @kaeldai  
Saskia de Vries @saskiad  
David Feng @dyf  
Tim Fliss @tfliss  
Marina Garrett @matchings  
Richard Gerkin @rgerkin  
Nathan Gouwens @gouwens  
Nile Graddis @nilegraddis  
Sergey Gratiy @sgratiy  
@jennan @jennan  
Xiaoxuan Jia @jiaxx  
Justin Kiggins @neuromusic  
Joe Knox @jknox13  
Peter Ledochowitsch @ledochowitsch  
Nicholas Mei @njmei  
Chris Mochizuki @mochic  
Gabe Ocker @gocker  
Michael Oliver @the-moliver  
Doug Ollerenshaw @dougollerenshaw

Jed Perkins @jfperkins

Alex Piet @alexpiet

Nick Ponvert @nickponvert

Kat Schelonka @kschelonka

Shu Shi @shus2018

Josh Siegle @jsiegle

Corinne Teeter @corinneteeter

Shreejoy Tripathy @stripathy

Werner Van Geit @wvangeit

Michael Wang @aimichaelwang

Derric Williams @derricw

## 6.1 Subpackages

### 6.1.1 allensdk.api package

#### Subpackages

allensdk.api.queries package

#### Submodules

allensdk.api.queries.annotated\_section\_data\_sets\_api module

**class** allensdk.api.queries.annotated\_section\_data\_sets\_api.**AnnotatedSectionDataSetsApi** (*base*  
Bases: *allensdk.api.queries.rma\_api.RmaApi*)

See: [Searching Annotated SectionDataSets](#)

**get\_annotated\_section\_data\_sets** (*self*, *structures*, *intensity\_values=None*, *den-*  
*sity\_values=None*, *pattern\_values=None*,  
*age\_names=None*)

For a list of target structures, find the SectionDataSet that matches the parameters for intensity\_values, density\_values, pattern\_values, and Age.

#### Parameters

**structure\_graph\_id** [dict of integers] what to retrieve

**intensity\_values** [array of strings, optional] 'High', 'Low', 'Medium' (default)

**density\_values** [array of strings, optional] 'High', 'Low'

**pattern\_values** [array of strings, optional] 'Full'

**age\_names** [array of strings, options] for example 'E11.5', '13.5'

**Returns**

**data** [dict] The parsed JSON response message.

**Notes**

This method uses the non-RMA Annotated SectionDataSet endpoint.

```
get_annotated_section_data_sets_via_rma (self, structures, intensity_values=None,  
                                           density_values=None, pattern_values=None,  
                                           age_names=None)
```

For a list of target structures, find the SectionDataSet that matches the parameters for intensity\_values, density\_values, pattern\_values, and Age.

**Parameters**

**structure\_graph\_id** [dict of integers] what to retrieve

**intensity\_values** [array of strings, optional] intensity values, 'High', 'Low', 'Medium' (default)

**density\_values** [array of strings, optional] density values, 'High', 'Low'

**pattern\_values** [array of strings, optional] pattern values, 'Full'

**age\_names** [array of strings, options] for example 'E11.5', '13.5'

**Returns**

**data** [dict] The parsed JSON response message.

**Notes**

This method uses the RMA endpoint to search annotated SectionDataSet data.

```
get_compound_annotated_section_data_sets (self, queries, fmt='json')
```

Find the SectionDataSet that matches several annotated\_section\_data\_sets queries linked together with a Boolean 'and' or 'or'.

**Parameters**

**queries** [array of dicts] dicts with args like build\_query

**fmt** [string, optional] 'json' or 'xml'

**Returns**

**data** [dict] The parsed JSON response message.

**allensdk.api.queries.biophysical\_api module**

```
class allensdk.api.queries.biophysical_api.BiophysicalApi (base_uri=None)
```

Bases: *allensdk.api.queries.rma\_template.RmaTemplate*

```
BIOPHYSICAL_MODEL_TYPE_IDS = (491455321, 329230710)
```

```
build_rma (self, neuronal_model_id, fmt='json')
```

Construct a query to find all files related to a neuronal model.

**Parameters**

**neuronal\_model\_id** [integer or string representation] key of experiment to retrieve.



**fmt** [string, optional] json (default) or xml

### Returns

**string** RMA query url.

**cache\_data** (*self*, *neuronal\_model\_id*, *working\_directory=None*)

Take a an experiment id, query the Api RMA to get well-known-files download the files, and store them in the working directory.

### Parameters

**neuronal\_model\_id** [int or string representation] found in the neuronal\_model table in the api

**working\_directory** [string] Absolute path name where the downloaded well-known files will be stored.

**create\_manifest** (*self*, *fit\_path=""*, *model\_type=""*, *stimulus\_filename=""*, *swc\_morphology\_path=""*, *marker\_path=""*, *sweeps=[]*)

Generate a json configuration file with parameters for a a biophysical experiment.

### Parameters

**fit\_path** [string] filename of a json configuration file with cell parameters.

**stimulus\_filename** [string] path to an NWB file with input currents.

**swc\_morphology\_path** [string] file in SWC format.

**sweeps** [array of integers] which sweeps in the stimulus file are to be used.

**get\_neuronal\_models** (*self*, *specimen\_ids*, *num\_rows='all'*, *count=False*, *model\_type\_ids=None*, *\*\*kwargs*)

Fetch all of the biophysically detailed model records associated with a particular specimen\_id

### Parameters

**specimen\_ids** [list] One or more integer ids identifying specimen records.

**num\_rows** [int, optional] how many records to retrieve. Default is 'all'.

**count** [bool, optional] If True, return a count of the lines found by the query. Default is False.

**model\_type\_ids** [list, optional] One or more integer ids identifying categories of neuronal model. Defaults to all-active and perisomatic biophysical\_models.

### Returns

**List of dict** Each element is a biophysical model record, containing a unique integer id, the id of the associated specimen, and the id of the model type to which this model belongs.

**get\_well\_known\_file\_ids** (*self*, *neuronal\_model\_id*)

Query the current RMA endpoint with a neuronal\_model id to get the corresponding well known file ids.

### Returns

**list** A list of well known file id strings.

**is\_well\_known\_file\_type** (*self*, *wkf*, *name*)

Check if a structure has the expected name.

### Parameters

**wkf** [dict] A well-known-file structure with nested type information.

**name** [string] The expected type name

See also:

`read_json` where this helper function is used.

`read_json(self, json_parsed_data)`

Get the list of well\_known\_file ids from a response body containing nested sample, microarray\_slides, well\_known\_files.

#### Parameters

**json\_parsed\_data** [dict] Response from the Allen Institute Api RMA.

#### Returns

**list of strings** Well known file ids.

```
rma_templates = {'model_queries': [{'name': 'models_by_specimen', 'description': 's
```

### allensdk.api.queries.brain\_observatory\_api module

```
class allensdk.api.queries.brain_observatory_api.BrainObservatoryApi (base_uri=None,
                                                                    dat-
                                                                    acube_uri=None)
```

Bases: `allensdk.api.queries.rma_template.RmaTemplate`

**CELL\_MAPPING\_ID** = 590985414

**NWB\_FILE\_TYPE** = 'NWBophys'

**OPHYS\_ANALYSIS\_FILE\_TYPE** = 'OphysExperimentCellRoiMetricsFile'

**OPHYS\_EVENTS\_FILE\_TYPE** = 'ObservatoryEventsFile'

`dataframe_query(self, data, filters, primary_key)`

Given a list of dictionary records and a list of filter dictionaries, filter the records using Pandas and return the filtered set of records.

#### Parameters

**data: list of dicts** List of dictionaries

**filters: list of dicts** Each dictionary describes a filtering operation on a field in the dictionary. The general form is { 'field': <field>, 'op': <operation>, 'value': <filter\_value(s)> }. For example, you can apply a threshold on the "osi\_dg" column with something like this: { 'field': 'osi\_dg', 'op': '>', 'value': 1.0 }. See `_QUERY_TEMPLATES` for a full list of operators.

`dataframe_query_string(self, filters)`

Convert a list of cell metric filter dictionaries into a Pandas query string.

`filter_cell_specimens(self, cell_specimens, ids=None, experiment_container_ids=None, include_failed=False, filters=None)`

Filter a list of cell specimen records returned from the `get_cell_metrics` method according some of their properties.

#### Parameters

**cell\_specimens: list of dicts** List of records returned by the `get_cell_metrics` method.

**ids: list of integers** Return only records for cells with cell specimen ids in this list

**experiment\_container\_ids: list of integers** Return only records for cells that belong to experiment container ids in this list

**include\_failed: bool** Whether to include cells from failed experiment containers

**filters: list of dicts** Custom query used to reproduce filter sets created in the Allen Brain Observatory web application. The general form is a list of dictionaries each of which describes a filtering operation based on a metric. For more information, see `dataframe_query`.

**filter\_experiment\_containers** (*self*, *containers*, *ids=None*, *targeted\_structures=None*, *imaging\_depths=None*, *cre\_lines=None*, *reporter\_lines=None*, *transgenic\_lines=None*, *include\_failed=False*, *simple=False*)

**filter\_experiments\_and\_containers** (*self*, *objs*, *ids=None*, *targeted\_structures=None*, *imaging\_depths=None*, *cre\_lines=None*, *reporter\_lines=None*, *transgenic\_lines=None*, *include\_failed=False*)

**filter\_ophys\_experiments** (*self*, *experiments*, *ids=None*, *experiment\_container\_ids=None*, *targeted\_structures=None*, *imaging\_depths=None*, *cre\_lines=None*, *reporter\_lines=None*, *transgenic\_lines=None*, *stimuli=None*, *session\_types=None*, *include\_failed=False*, *require\_eye\_tracking=False*, *simple=False*)

**get\_cell\_metrics** (*self*, *cell\_specimen\_ids=None*, *\*args*, *\*\*kwargs*)

Get cell metrics by id

#### Parameters

**cell\_metrics\_ids** [integer or list of integers, optional] only select specific cell metric records.

#### Returns

**dict** [cell metric metadata]

**get\_cell\_specimen\_id\_mapping** (*self*, *file\_name*, *mapping\_table\_id=None*)

Download mapping table from old to new cell specimen IDs.

The mapping table is a CSV file that maps cell specimen ids that have changed between processing runs of the Brain Observatory pipeline.

#### Parameters

**file\_name** [string] Filename to save locally.

**mapping\_table\_id** [integer] ID of the mapping table file. Defaults to the most recent mapping table.

#### Returns

**pandas.DataFrame** Mapping table as a DataFrame.

**get\_column\_definitions** (*self*, *api\_class\_name=None*)

Get column definitions

#### Parameters

**api\_class\_names** [string or list of strings, optional] only select specific column definition records.

#### Returns

**dict** [column definition metadata]

**get\_experiment\_container\_metrics** (*self*, *experiment\_container\_metric\_ids=None*)

Get experiment container metrics by id

#### Parameters

**isi\_experiment\_ids** [integer or list of integers, optional] only select specific experiments.

**Returns**

**dict** [isi experiment metadata]

**get\_experiment\_containers** (*self*, *experiment\_container\_ids=None*)

Get experiment container by id

**Parameters**

**experiment\_container\_ids** [integer or list of integers, optional] only select specific experiment containers.

**Returns**

**dict** [experiment container metadata]

**get\_isi\_experiments** (*self*, *isi\_experiment\_ids=None*)

Get ISI Experiments by id

**Parameters**

**isi\_experiment\_ids** [integer or list of integers, optional] only select specific experiments.

**Returns**

**dict** [isi experiment metadata]

**get\_ophys\_experiments** (*self*, *ophys\_experiment\_ids=None*)

Get OPhys Experiments by id

**Parameters**

**ophys\_experiment\_ids** [integer or list of integers, optional] only select specific experiments.

**Returns**

**dict** [ophys experiment metadata]

**get\_stimulus\_mappings** (*self*, *stimulus\_mapping\_ids=None*)

Get stimulus mappings by id

**Parameters**

**stimulus\_mapping\_ids** [integer or list of integers, optional] only select specific stimulus mapping records.

**Returns**

**dict** [stimulus mapping metadata]

**list\_column\_definition\_class\_names** (*self*)

Get column definitions

**Returns**

**list** [api class name strings]

**list\_isi\_experiments** (*self*, *isi\_ids=None*)

List ISI experiments available through the Allen Institute API

**Parameters**

**neuronal\_model\_ids** [integer or list of integers, optional] only select specific isi experiments.

**Returns**

```

    dict [neuronal model metadata]

    rma_templates = {'brain_observatory_queries': [{'name': 'list_isi_experiments', 'des
    save_ophys_experiment_analysis_data(self, ophys_experiment_id, file_name)
    save_ophys_experiment_data(self, ophys_experiment_id, file_name)
    save_ophys_experiment_event_data(self, ophys_experiment_id, file_name)
    save_ophys_experiment_eye_gaze_data(self, ophys_experiment_id: int, ophys_session_id:
        int, file_name: str)
    simplify_experiment_containers(self, containers)
    simplify_ophys_experiments(self, exps)

allensdk.api.queries.brain_observatory_api.find_container_tags(container)
    Custom logic for extracting tags from donor conditions. Filtering out tissueocyte tags.

allensdk.api.queries.brain_observatory_api.find_experiment_acquisition_age(exp)
allensdk.api.queries.brain_observatory_api.find_specimen_cre_line(specimen)
allensdk.api.queries.brain_observatory_api.find_specimen_reporter_line(specimen)
allensdk.api.queries.brain_observatory_api.find_specimen_transgenic_lines(specimen)

```

**allensdk.api.queries.cell\_types\_api module**

```

class allensdk.api.queries.cell_types_api.CellTypesApi (base_uri=None)
    Bases: allensdk.api.queries.rma_api.RmaApi

    HUMAN = 'Homo Sapiens'
    MARKER_FILE_TYPE = '3DNeuronMarker'
    MOUSE = 'Mus musculus'
    NWB_FILE_TYPE = 'NWBDownload'
    SWC_FILE_TYPE = '3DNeuronReconstruction'

    filter_cells(self, cells, require_morphology, require_reconstruction, reporter_status, species)
        Filter a list of cell specimens to those that optionally have morphologies or have morphological recon-
        structions.

```

**Parameters**

```

    cells: list List of cell metadata dictionaries to be filtered

    require_morphology: boolean Filter out cells that have no morphological images.

    require_reconstruction: boolean Filter out cells that have no morphological reconstruc-
        tions.

    reporter_status: list Filter for cells that have a particular cell reporter status

    species: list Filter for cells that belong to one or more species. If None, return all. Must be
        one of [ CellTypesApi.MOUSE, CellTypesApi.HUMAN ].

    filter_cells_api(self, cells, require_morphology=False, require_reconstruction=False, re-
        porter_status=None, species=None, simple=True)

```

**get\_cell** (*self*, *id*)

Query the API for a one cells in the Cell Types Database.

**Returns**

**list** Meta data for one cell.

**get\_ephys\_features** (*self*)

Query the API for the full table of EphysFeatures for all cells.

**get\_ephys\_sweeps** (*self*, *specimen\_id*)

Query the API for a list of sweeps for a particular cell in the Cell Types Database.

**Parameters**

**specimen\_id: int** Specimen ID of a cell.

**Returns**

**list: List of sweep dictionaries belonging to a cell**

**get\_morphology\_features** (*self*)

Query the API for the full table of morphology features for all cells

**Notes**

by default the tags column is removed because it isn't useful

**list\_cells** (*self*, *id=None*, *require\_morphology=False*, *require\_reconstruction=False*, *reporter\_status=None*, *species=None*)

Query the API for a list of all cells in the Cell Types Database.

**Parameters**

**id: int** ID of a cell. If not provided returns all matching cells.

**require\_morphology: boolean** Only return cells that have morphology images.

**require\_reconstruction: boolean** Only return cells that have morphological reconstructions.

**reporter\_status: list** Return cells that have a particular cell reporter status.

**species: list** Filter for cells that belong to one or more species. If None, return all. Must be one of [ CellTypesApi.MOUSE, CellTypesApi.HUMAN ].

**Returns**

**list** Meta data for all cells.

**list\_cells\_api** (*self*, *id=None*, *require\_morphology=False*, *require\_reconstruction=False*, *reporter\_status=None*, *species=None*)

**save\_ephys\_data** (*self*, *specimen\_id*, *file\_name*)

Save the electrophysiology recordings for a cell as an NWB file.

**Parameters**

**specimen\_id: int** ID of the specimen, from the Specimens database model in the Allen Institute API.

**file\_name: str** Path to save the NWB file.

**save\_reconstruction** (*self*, *specimen\_id*, *file\_name*)

Save the morphological reconstruction of a cell as an SWC file.

**Parameters**

**specimen\_id: int** ID of the specimen, from the Specimens database model in the Allen Institute API.

**file\_name: str** Path to save the SWC file.

**save\_reconstruction\_markers** (*self, specimen\_id, file\_name*)

Save the marker file for the morphological reconstruction of a cell. These are comma-delimited files indicating points of interest in a reconstruction (truncation points, early tracing termination, etc).

**Parameters**

**specimen\_id: int** ID of the specimen, from the Specimens database model in the Allen Institute API.

**file\_name: str** Path to save the marker file.

**simplify\_cells\_api** (*self, cells*)

**allensdk.api.queries.connected\_services module**

**class** allensdk.api.queries.connected\_services.**ConnectedServices**

Bases: object

A class representing a schema of informatics web services.

**Notes**

See [Connected Services and Pipes](#) for a human-readable list of services and parameters.

The URL format is documented at [Service Pipelines](#).

Connected Services only include API services that are accessed via the RMA endpoint using an rma::services stage.

**ARRAY** = 'array'

**BOOLEAN** = 'boolean'

**FLOAT** = 'float'

**INTEGER** = 'integer'

**STRING** = 'string'

**build\_url** (*self, service\_name, kwargs*)

Create a single stage RMA url from a service name and parameters.

**classmethod** **schema** ()

Dictionary of service names and parameters.

**Notes**

See [Connected Services and Pipes](#) for a human-readable list of connected services and their parameters.

### allensdk.api.queries.glif\_api module

**class** allensdk.api.queries.glif\_api.GlifApi (*base\_uri=None*)

Bases: *allensdk.api.queries.rma\_template.RmaTemplate*

**GLIF\_TYPES** = [395310498, 395310469, 395310475, 395310479, 471355161]

**NWB\_FILE\_TYPE** = None

**cache\_stimulus\_file** (*self, output\_file\_name*)

DEPRECATED Download the NWB file for the current neuronal model and save it to a file.

#### Parameters

**output\_file\_name: string** File name to store the NWB file.

**get\_ephys\_sweeps** (*self*)

DEPRECATED Retrieve ephys sweep information out of downloaded metadata for a neuronal model

#### Returns

**list** A list of sweeps metadata dictionaries

**get\_neuron\_config** (*self, output\_file\_name=None*)

DEPRECATED Retrieve a model configuration file from the API, optionally save it to disk, and return the contents of that file as a dictionary.

#### Parameters

**output\_file\_name: string** File name to store the neuron configuration (optional).

**get\_neuron\_configs** (*self, neuronal\_model\_ids=None*)

**get\_neuronal\_model** (*self, neuronal\_model\_id*)

DEPRECATED Query the current RMA endpoint with a neuronal\_model id to get the corresponding well known files and meta data.

#### Returns

**dict** A dictionary containing

**get\_neuronal\_model\_templates** (*self*)

**get\_neuronal\_models** (*self, ephys\_experiment\_ids=None*)

**get\_neuronal\_models\_by\_id** (*self, neuronal\_model\_ids=None*)

**list\_neuronal\_models** (*self*)

DEPRECATED Query the API for a list of all GLIF neuronal models.

#### Returns

**list** Meta data for all GLIF neuronal models.

**rma\_templates** = {'glif\_queries': [{'name': 'neuronal\_model\_templates', 'description':

### allensdk.api.queries.grid\_data\_api module

**class** allensdk.api.queries.grid\_data\_api.GridDataApi (*resolution=None,*

*base\_uri=None*)

Bases: *allensdk.api.queries.rma\_api.RmaApi*

HTTP Client for the Allen 3-D Expression Grid Data Service.

See: [Downloading 3-D Expression Grid Data](#)



```
DATA_MASK = 'data_mask'
```

```
DENSITY = 'density'
```

```
ENERGY = 'energy'
```

```
INJECTION_DENSITY = 'injection_density'
```

```
INJECTION_ENERGY = 'injection_energy'
```

```
INJECTION_FRACTION = 'injection_fraction'
```

```
INTENSITY = 'intensity'
```

```
PROJECTION_DENSITY = 'projection_density'
```

```
PROJECTION_ENERGY = 'projection_energy'
```

```
download_alignment3d(self, section_data_set_id, num_rows='all', count=False, **kwargs)
```

Download the parameters of the 3D affine transformation mapping this section data set's image-space stack to CCF-space (or vice-versa).

#### Parameters

**section\_data\_set\_id** [int] download the parameters for this data set.

#### Returns

**dict** : parameters of this section data set's alignment3d

```
download_deformation_field(self, section_data_set_id, header_path=None,
                             voxel_path=None, voxel_type='DeformationFieldVoxels',
                             header_type='DeformationFieldHeader')
```

Download the local alignment parameters for this dataset. This a 3D vector image (3 components) describing a deformable local mapping from CCF voxels to this section data set's affine-aligned image stack.

#### Parameters

**section\_data\_set\_id** [int]

Download the deformation field for this data set

**header\_path** [str, optional] If supplied, the deformation field header will be downloaded to this path.

**voxel\_path** [str, optional] If supplied, the deformation field voxels will be downloaded to this path.

**voxel\_type** [str] WellKnownFileType of this dataset's data file

**header\_type** [str] WellKnownFileType of this dataset's header file

```
download_expression_grid_data(self, section_data_set_id, include=None, path=None)
```

Download in zipped metaimage format.

#### Parameters

**section\_data\_set\_id** [integer] What to download.

**include** [list of strings, optional] Image volumes. 'energy' (default), 'density', 'intensity'.

**path** [string, optional] File name to save as.

#### Returns

**file** [3-D expression grid data packaged into a compressed archive file (.zip).]

**download\_gene\_expression\_grid\_data** (*self, section\_data\_set\_id, volume\_type, path*)

Download a metainage file containing registered gene expression grid data

**Parameters**

**section\_data\_set\_id** [int] Download data from this experiment

**volume\_type** [str] Download this type of data (options are GridDataApi.ENERGY, GridDataApi.DENSITY, GridDataApi.INTENSITY)

**path** [str] Download to this path

**download\_projection\_grid\_data** (*self, section\_data\_set\_id, image=None, resolution=None, save\_file\_path=None*)

Download in NRRD format.

**Parameters**

**section\_data\_set\_id** [integer] What to download.

**image** [list of strings, optional] Image volume. 'projection\_density', 'projection\_energy', 'injection\_fraction', 'injection\_density', 'injection\_energy', 'data\_mask'.

**resolution** [integer, optional] in microns. 10, 25, 50, or 100 (default).

**save\_file\_path** [string, optional] File name to save as.

**Notes**

See [Downloading 3-D Projection Grid Data](#) for additional documentation.

## **allensdk.api.queries.image\_download\_api module**

**class** allensdk.api.queries.image\_download\_api.**ImageDownloadApi** (*base\_uri=None*)

Bases: *allensdk.api.queries.rma\_template.RmaTemplate*

HTTP Client to download whole or partial two-dimensional images from the Allen Institute with the Section-Image, AtlasImage and ProjectionImage Download Services.

See [Downloading an Image](#) for more documentation.

**COLORMAPS** = {'aba': 8, 'aibsmmap\_alt': 9, 'blue': 6, 'colormap': 10, 'expression':

**atlas\_image\_query** (*self, atlas\_id, image\_type\_name=None*)

List atlas images belonging to a specified atlas

**Parameters**

**atlas\_id** [integer, optional] Find images from this atlas.

**image\_type\_name** [string, optional] Restrict response to images of this type. If not provided, the query will get it from the atlas id.

**Returns**

**list of dict** : Each element is an AtlasImage record.

## Notes

See [Downloading Atlas Images and Graphics](#) for additional documentation. `allensdk.api.queries.ontologies_api.OntologiesApi.get_atlases()` can also be used to list atlases along with their ids.

**download\_atlas\_image** (*self*, *atlas\_image\_id*, *file\_path=None*, *\*\*kwargs*)

**download\_image** (*self*, *image\_id*, *file\_path=None*, *endpoint=None*, *\*\*kwargs*)

Download whole or partial two-dimensional images from the Allen Institute with the SectionImage or AtlasImage service.

### Parameters

**image\_id** [integer] SubImage to download.

**file\_path** [string, optional] where to put it, defaults to image\_id.jpg

**downsample** [int, optional] Number of times to downsample the original image.

**quality** [int, optional] jpeg quality of the returned image, 0 to 100 (default)

**expression** [boolean, optional] Request the expression mask for the SectionImage.

**view** [string, optional] 'expression', 'projection', 'tumor\_feature\_annotation' or 'tumor\_feature\_boundary'

**top** [int, optional] Index of the topmost row of the region of interest.

**left :int, optional** Index of the leftmost column of the region of interest.

**width** [int, optional] Number of columns in the output image.

**height** [int, optional] Number of rows in the output image.

**range** [list of ints, optional] Filter to specify the RGB channels. low,high,low,high,low,high

**colormap** [list of floats, optional] Filter to specify the RGB channels. [lower\_threshold,colormap] gain 0-1, colormap id is a string from ImageDownload-Api.COLORMAPS

**rgb** [list of floats, optional] Filter to specify the RGB channels. [red,green,blue] 0-1

**contrast** [list of floats, optional] Filter to specify contrast parameters. [gain,bias] 0-1

**annotation** [boolean, optional] Request the annotated AtlasImage

**atlas** [int, optional] Specify the desired Atlas' annotations.

**projection** [boolean, optional] Request projection for the specified image.

**downsample\_dimensions** [boolean, optional] Indicates if the width and height should be adjusted to account for downsampling.

### Returns

**None** the file is downloaded and saved to the path.

## Notes

By default, an unfiltered full-sized image with the highest quality is returned as a download if no parameters are provided.

'downsample=1' halves the number of pixels of the original image both horizontally and vertically.  
range\_list = kwargs.get('range', None)

Specifying ‘downsample=2’ quarters the height and width values.

Quality must be an integer from 0, for the lowest quality, up to as high as 100. If it is not specified, it defaults to the highest quality.

Top is specified in full-resolution (largest tier) pixel coordinates. `SectionImage.y` is the default value.

Left is specified in full-resolution (largest tier) pixel coordinates. `SectionImage.x` is the default value.

Width is specified in tier-resolution (desired tier) pixel coordinates. `SectionImage.width` is the default value. It is automatically adjusted when downsampled.

Height is specified in tier-resolution (desired tier) pixel coordinates. `SectionImage.height` is the default value. It is automatically adjusted when downsampled.

The range parameter consists of 6 comma delimited integers that define the lower (0) and upper (4095) bound for each channel in red-green-blue order (i.e. “range=0,1500,0,1000,0,4095”). The default range values can be determined by referring to the following fields on the Equalization model associated with the `SectionDataSet`: `red_lower`, `red_upper`, `green_lower`, `green_upper`, `blue_lower`, `blue_upper`. For more information, see the [Image Controls](#) section of the Allen Mouse Brain Connectivity Atlas: [Projection Dataset](#) help topic. See: ‘Image Download Service’ <<http://help.brain-map.org/display/api/Downloading+an+Image>>\_

**download\_projection\_image** (*self*, *projection\_image\_id*, *file\_path=None*, *\*\*kwargs*)

**download\_section\_image** (*self*, *section\_image\_id*, *file\_path=None*, *\*\*kwargs*)

**get\_section\_data\_sets\_by\_product** (*self*, *product\_ids*, *include\_failed=False*,  
*num\_rows='all'*, *count=False*, *\*\*kwargs*)

List all of the section data sets produced as part of one or more products

#### Parameters

**product\_ids** [list of int] Integer specifiers for Allen Institute products. A product is a set of related data.

**include\_failed** [bool, optional] If True, find both failed and passed datasets. Default is False

**num\_rows** [int, optional] how many records to retrieve. Default is ‘all’.

**count** [bool, optional] If True, return a count of the lines found by the query. Default is False.

#### Returns

**list of dict** : Each returned element is a section data set record.

#### Notes

See <http://api.brain-map.org/api/v2/data/query.json?criteria=model::Product> for a list of products.

**get\_section\_image\_ranges** (*self*, *section\_image\_ids*, *num\_rows='all'*, *count=False*,  
*as\_lists=True*, *\*\*kwargs*)

Section images from the Mouse Connectivity Atlas are displayed on [connectivity.brain-map.org](http://connectivity.brain-map.org) after having been linearly windowed and leveled. This method obtains parameters defining channelwise upper and lower bounds of the windows used for one or more images.

#### Parameters

**section\_image\_ids** [list of int] Each element is a unique identifier for a section image.

**num\_rows** [int, optional] how many records to retrieve. Default is ‘all’.

**count** [bool, optional] If True, return a count of the lines found by the query. Default is False.

**as\_lists** [bool, optional] If True, return the window parameters in a list, rather than a dict (this is the format of the range parameter on ImageDownloadApi.download\_image). Default is False.

### Returns

**list of dict or list of list :** For each section image id provided, return the window bounds for each channel.

```
rma_templates = {'image_queries': [{'name': 'section_image_ranges', 'description':
```

```
section_image_query (self, section_data_set_id, num_rows='all', count=False, **kwargs)
```

List section images belonging to a specified section data set

### Parameters

**atlas\_id** [integer, optional] Find images from this section data set.

**num\_rows** [int] how many records to retrieve. Default is 'all'

**count** [bool] If True, return a count of the lines found by the query.

### Returns

**list of dict :** Each element is an SectionImage record.

## Notes

The SectionDataSet model is used to represent single experiments which produce an array of images. This includes Mouse Connectivity and Mouse Brain Atlas experiments, among other projects. You may see references to the ids of experiments from those projects. These are the same as section data set ids.

## allensdk.api.queries.mouse\_atlas\_api module

```
class allensdk.api.queries.mouse_atlas_api.MouseAtlasApi (base_uri=None)
```

Bases: *allensdk.api.queries.reference\_space\_api.ReferenceSpaceApi*, *allensdk.api.queries.grid\_data\_api.GridDataApi*

Downloads Mouse Brain Atlas grid data, reference volumes, and metadata.

```
DEVMOUSE_ATLAS_PRODUCTS = (3,)
```

```
HUMAN_ORGANISM = (1,)
```

```
MOUSE_ATLAS_PRODUCTS = (1,)
```

```
MOUSE_ORGANISM = (2,)
```

```
download_expression_density (self, path, experiment_id)
```

```
download_expression_energy (self, path, experiment_id)
```

```
download_expression_intensity (self, path, experiment_id)
```

```
get_genes (self, organism_ids=None, chromosome_ids=None, **kwargs)
```

Download a list of genes

### Parameters

**organism\_ids** [list of int, optional] Filter genes to those appearing in these organisms. Defaults to mouse (2).

**chromosome\_ids** [list of int, optional] Filter genes to those appearing on these chromosomes. Defaults to all.

#### Returns

**list of dict:** Each element is a gene record, with a nested chromosome record (also a dict).

**get\_section\_data\_sets** (*self*, *gene\_ids=None*, *product\_ids=None*, *\*\*kwargs*)

Download a list of section data sets (experiments) from the Mouse Brain Atlas project.

#### Parameters

**gene\_ids** [list of int, optional] Filter results based on the genes whose expression was characterized in each experiment. Default is all.

**product\_ids** [list of int, optional] Filter results to a subset of products. Default is the Mouse Brain Atlas.

#### Returns

**list of dict :** Each element is a section data set record, with one or more gene records nested in a list.

### allensdk.api.queries.mouse\_connectivity\_api module

**class** allensdk.api.queries.mouse\_connectivity\_api.**MouseConnectivityApi** (*base\_uri=None*)

Bases: [allensdk.api.queries.reference\\_space\\_api.ReferenceSpaceApi](#), [allensdk.api.queries.grid\\_data\\_api.GridDataApi](#)

HTTP Client for the Allen Mouse Brain Connectivity Atlas.

See: [Mouse Connectivity API](#)

**PRODUCT\_IDS** = [5, 31]

**build\_reference\_aligned\_image\_channel\_volumes\_url** (*self*, *data\_set\_id*)

Construct url to download the red, green, and blue channels aligned to the 25um adult mouse brain reference space volume.

#### Parameters

**data\_set\_id** [integerallensdk.api.queries] aka attachable\_id

#### Notes

See: [Reference-aligned Image Channel Volumes](#) for additional documentation.

**calculate\_injection\_centroid** (*self*, *injection\_density*, *injection\_fraction*, *resolution=25*)

Compute the centroid of an injection site.

#### Parameters

**injection\_density:** **np.ndarray** The injection density volume of an experiment

**injection\_fraction:** **np.ndarray** The injection fraction volume of an experiment

**download\_data\_mask** (*self*, *path*, *experiment\_id*, *resolution*)

**download\_injection\_density** (*self*, *path*, *experiment\_id*, *resolution*)

**download\_injection\_fraction** (*self*, *path*, *experiment\_id*, *resolution*)

**download\_projection\_density** (*self*, *path*, *experiment\_id*, *resolution*)

**download\_reference\_aligned\_image\_channel\_volumes** (*self*, *data\_set\_id*,  
*save\_file\_path*=None)

#### Returns

The well known file is downloaded

**experiment\_correlation\_search** (*self*, *\*\*kwargs*)

Select a seed experiment and a domain over which the similarity comparison is to be made.

#### Parameters

**row** [integer] SectionDataSet.id to correlate against.

**structures** [list of integers or strings, optional] Integer Structure.id or String Structure.acronym.

**hemisphere** [string, optional] Use 'right' or 'left'. Defaults to both hemispheres.

**transgenic\_lines** [list of integers or strings, optional] Integer TransgenicLine.id or String TransgenicLine.name. Specify ID 0 to exclude all TransgenicLines.

**injection\_structures** [list of integers or strings, optional] Integer Structure.id or String Structure.acronym.

**primary\_structure\_only** [boolean, optional]

**product\_ids** [list of integers, optional] Integer Product.id

**start\_row** [integer, optional] For paging purposes. Defaults to 0.

**num\_rows** [integer, optional] For paging purposes. Defaults to 2000.

#### Notes

See [Correlation Search](#) and [service::mouse\\_connectivity\\_correlation](#).

**experiment\_injection\_coordinate\_search** (*self*, *\*\*kwargs*)

User specifies a seed location within the 3D reference space. The service returns a rank list of experiments by distance of its injection site to the specified seed location.

#### Parameters

**seed\_point** [list of floats] The coordinates of a point in 3-D SectionDataSet space.

**transgenic\_lines** [list of integers or strings, optional] Integer TransgenicLine.id or String TransgenicLine.name. Specify ID 0 to exclude all TransgenicLines.

**injection\_structures** [list of integers or strings, optional] Integer Structure.id or String Structure.acronym.

**primary\_structure\_only** [boolean, optional]

**product\_ids** [list of integers, optional] Integer Product.id

**start\_row** [integer, optional] For paging purposes. Defaults to 0.

**num\_rows** [integer, optional] For paging purposes. Defaults to 2000.

## Notes

See [Injection Coordinate Search](#) and `service::mouse_connectivity_injection_coordinate`.

**experiment\_source\_search** (*self*, *\*\*kwargs*)

Search over the whole projection signal statistics dataset to find experiments with specific projection profiles.

### Parameters

**injection\_structures** [list of integers or strings] Integer Structure.id or String Structure.acronym.

**target\_domain** [list of integers or strings, optional] Integer Structure.id or String Structure.acronym.

**injection\_hemisphere** [string, optional] 'right' or 'left', Defaults to both hemispheres.

**target\_hemisphere** [string, optional] 'right' or 'left', Defaults to both hemispheres.

**transgenic\_lines** [list of integers or strings, optional] Integer TransgenicLine.id or String TransgenicLine.name. Specify ID 0 to exclude all TransgenicLines.

**injection\_domain** [list of integers or strings, optional] Integer Structure.id or String Structure.acronym.

**primary\_structure\_only** [boolean, optional]

**product\_ids** [list of integers, optional] Integer Product.id

**start\_row** [integer, optional] For paging purposes. Defaults to 0.

**num\_rows** [integer, optional] For paging purposes. Defaults to 2000.

## Notes

See [Source Search](#), [Target Search](#), and `service::mouse_connectivity_injection_structure`.

**experiment\_spatial\_search** (*self*, *\*\*kwargs*)

Displays all SectionDataSets with projection signal density  $\geq 0.1$  at the seed point. This service also returns the path along the most dense pixels from the seed point to the center of each injection site..

### Parameters

**seed\_point** [list of floats] The coordinates of a point in 3-D SectionDataSet space.

**transgenic\_lines** [list of integers or strings, optional] Integer TransgenicLine.id or String TransgenicLine.name. Specify ID 0 to exclude all TransgenicLines.

**section\_data\_sets** [list of integers, optional] Ids to filter the results.

**injection\_structures** [list of integers or strings, optional] Integer Structure.id or String Structure.acronym.

**primary\_structure\_only** [boolean, optional]

**product\_ids** [list of integers, optional] Integer Product.id

**start\_row** [integer, optional] For paging purposes. Defaults to 0.

**num\_rows** [integer, optional] For paging purposes. Defaults to 2000.



## Notes

See [Spatial Search](#) and `service::mouse_connectivity_target_spatial`.

**get\_experiment\_detail** (*self*, *experiment\_id*)

Retrieve the experiments data.

**get\_experiments** (*self*, *structure\_ids*, *\*\*kwargs*)

Fetch experiment metadata from the Mouse Brain Connectivity Atlas.

### Parameters

**structure\_ids** [integer or list, optional] injection structure

### Returns

**url** [string] The constructed URL

**get\_experiments\_api** (*self*)

Fetch experiment metadata from the Mouse Brain Connectivity Atlas via the ApiConnectivity table.

### Returns

**url** [string] The constructed URL

**get\_manual\_injection\_summary** (*self*, *experiment\_id*)

Retrieve manual injection summary.

**get\_projection\_image\_info** (*self*, *experiment\_id*, *section\_number*)

Fetch meta-information of one projection image.

### Parameters

**experiment\_id** [integer]

**section\_number** [integer]

## Notes

See: [image examples](#) under [Experimental Overview and Metadata](#) for additional documentation. Download the image using `allensdk.api.queries.image_download_api.ImageDownloadApi.download_section_image()`

**get\_reference\_aligned\_image\_channel\_volumes\_url** (*self*, *data\_set\_id*)

Retrieve the download link for a specific data set. Notes — See [Reference-aligned Image Channel Volumes](#) for additional documentation.

**get\_structure\_unionizes** (*self*, *experiment\_ids*, *is\_injection=None*, *structure\_name=None*, *structure\_ids=None*, *hemisphere\_ids=None*, *normalized\_projection\_volume\_limit=None*, *include=None*, *debug=None*, *order=None*)

## allensdk.api.queries.ontologies\_api module

**class** `allensdk.api.queries.ontologies_api.OntologiesApi` (*base\_uri=None*)

Bases: `allensdk.api.queries.rma_template.RmaTemplate`

See: [Atlas Drawings and Ontologies](#)

**get\_atlases** (*self*)

**get\_atlases\_table** (*self*, *atlas\_ids=None*, *brief=True*)

List Atlases available through the API with associated ontologies and structure graphs.

**Parameters**

**atlas\_ids** [integer or list of integers, optional] only select specific atlases

**brief** [boolean, optional] True (default) requests only name and id fields.

**Returns**

**dict** [atlas metadata]

**Notes**

This query is based on the [table of available Atlases](#). See also: [Class: Atlas](#)

**get\_structure\_graphs** (*self*)

**get\_structure\_sets** (*self*, *structure\_set\_ids=None*)

**get\_structures** (*self*, *structure\_graph\_ids=None*, *structure\_graph\_names=None*,  
*structure\_set\_ids=None*, *structure\_set\_names=None*, *order=*  
*der=['structures.graph\_order']*, *num\_rows='all'*, *count=False*, *\*\*kwargs*)

Retrieve data about anatomical structures.

**Parameters**

**structure\_graph\_ids** [int or list of ints, optional] database keys to get all structures in particular graphs

**structure\_graph\_names** [string or list of strings, optional] list of graph names to narrow the query

**structure\_set\_ids** [int or list of ints, optional] database keys to get all structures in a particular set

**structure\_set\_names** [string or list of strings, optional] list of set names to narrow the query.

**order** [list of strings] list of RMA order clauses for sorting

**num\_rows** [int] how many records to retrieve

**Returns**

**dict** the parsed json response containing data from the API

**Notes**

Only one of the methods of limiting the query should be used at a time.

**get\_structures\_with\_sets** (*self*, *structure\_graph\_ids*, *order=*  
*der=['structures.graph\_order']*,  
*num\_rows='all'*, *count=False*, *\*\*kwargs*)

Download structures along with the sets to which they belong.

**Parameters**

**structure\_graph\_ids** [int or list of int] Only fetch structure records from these graphs.

**order** [list of strings] list of RMA order clauses for sorting

**num\_rows** [int] how many records to retrieve

**Returns**

**dict** the parsed json response containing data from the API

**rma\_templates** = {'ontology\_queries': [{'name': 'structures\_by\_graph\_ids', 'descripti

**unpack\_structure\_set\_ancestors** (*self*, *structure\_dataframe*)

Convert a slash-separated *structure\_id\_path* field to a list.

#### Parameters

**structure\_dataframe** [DataFrame] structure data from the API

#### Returns

**None** A new column is added to the dataframe containing the ancestor list.

### allensdk.api.queries.reference\_space\_api module

**class** allensdk.api.queries.reference\_space\_api.**ReferenceSpaceApi** (*base\_uri=None*)

Bases: *allensdk.api.queries.rma\_api.RmaApi*

**ARA\_NISSL** = 'ara\_nissl'

**AVERAGE\_TEMPLATE** = 'average\_template'

**CCF\_2015** = 'annotation/ccf\_2015'

**CCF\_2016** = 'annotation/ccf\_2016'

**CCF\_2017** = 'annotation/ccf\_2017'

**CCF\_VERSION\_DEFAULT** = 'annotation/ccf\_2017'

**DEVMOUSE\_2012** = 'annotation/devmouse\_2012'

**MOUSE\_2011** = 'annotation/mouse\_2011'

**VOXEL\_RESOLUTION\_100\_MICRONS** = 100

**VOXEL\_RESOLUTION\_10\_MICRONS** = 10

**VOXEL\_RESOLUTION\_25\_MICRONS** = 25

**VOXEL\_RESOLUTION\_50\_MICRONS** = 50

**build\_volumetric\_data\_download\_url** (*self*, *data\_path*, *file\_name*, *voxel\_resolution=None*,  
*release=None*, *coordinate\_framework=None*)

Construct url to download 3D reference model in NRRD format.

#### Parameters

**data\_path** [string] 'average\_template', 'ara\_nissl', 'annotation/ccf\_{year}', 'annotation/mouse\_2011', or 'annotation/devmouse\_2012'

**voxel\_resolution** [int] 10, 25, 50 or 100

**coordinate\_framework** [string] 'mouse\_ccf' (default) or 'mouse\_annotation'

#### Notes

See: [3-D Reference Models](#) for additional documentation.

**download\_annotation\_volume** (*self*, *ccf\_version*, *resolution*, *file\_name*)

Download the annotation volume at a particular resolution.

#### Parameters

**ccf\_version: string** Which reference space version to download. Defaults to “annotation/ccf\_2017”

**resolution: int** Desired resolution to download in microns. Must be 10, 25, 50, or 100.

**file\_name: string** Where to save the annotation volume.

**Note: the parameters must be used as positional parameters, not keywords**

**download\_mouse\_atlas\_volume** (*self, age, volume\_type, file\_name*)

Download a reference volume (annotation, grid annotation, atlas volume) from the mouse brain atlas project

#### Parameters

**age** [str] Specify a mouse age for which to download the reference volume

**volume\_type** [str] Specify the type of volume to download

**file\_name** [str] Specify the path to the downloaded volume

**download\_structure\_mask** (*self, structure\_id, ccf\_version, resolution, file\_name*)

Download an indicator mask for a specific structure.

#### Parameters

**structure\_id** [int] Unique identifier for the annotated structure

**ccf\_version** [string] Which reference space version to download. Defaults to “annotation/ccf\_2017”

**resolution** [int] Desired resolution to download in microns. Must be 10, 25, 50, or 100.

**file\_name** [string] Where to save the downloaded mask.

**download\_structure\_mesh** (*self, structure\_id, ccf\_version, file\_name*)

Download a Wavefront obj file containing a triangulated 3d mesh built from an annotated structure.

#### Parameters

**structure\_id** [int] Unique identifier for the annotated structure

**ccf\_version** [string] Which reference space version to download. Defaults to “annotation/ccf\_2017”

**file\_name** [string] Where to save the downloaded mask.

**download\_template\_volume** (*self, resolution, file\_name*)

Download the registration template volume at a particular resolution.

#### Parameters

**resolution: int** Desired resolution to download in microns. Must be 10, 25, 50, or 100.

**file\_name: string** Where to save the registration template volume.

**download\_volumetric\_data** (*self, data\_path, file\_name, voxel\_resolution=None, save\_file\_path=None, release=None, coordinate\_framework=None*)

Download 3D reference model in NRRD format.

#### Parameters

**data\_path** [string] ‘average\_template’, ‘ara\_nissl’, ‘annotation/ccf\_{year}’, ‘annotation/mouse\_2011’, or ‘annotation/devmouse\_2012’

**file\_name** [string] server-side file name. ‘annotation\_10.nrrd’ for example.

**voxel\_resolution** [int] 10, 25, 50 or 100

**coordinate\_framework** [string] 'mouse\_ccf' (default) or 'mouse\_annotation'

## Notes

See: [3-D Reference Models](#) for additional documentation.

## allensdk.api.queries.rma\_api module

**class** allensdk.api.queries.rma\_api.**RmaApi** (*base\_uri=None*)

Bases: [allensdk.api.api.Api](#)

See: [RESTful Model Access \(RMA\)](#)

**ALL** = 'all'

**COUNT** = 'count'

**CRITERIA** = 'rma::criteria'

**DEBUG** = 'debug'

**EQ** = '\$eq'

**EXCEPT** = 'except'

**EXCPT** = 'excpt'

**FALSE** = 'false'

**INCLUDE** = 'rma::include'

**IS** = '\$is'

**MODEL** = 'model::'

**NUM\_ROWS** = 'num\_rows'

**ONLY** = 'only'

**OPTIONS** = 'rma::options'

**ORDER** = 'order'

**PIPE** = 'pipe::'

**PREVIEW** = 'preview'

**SERVICE** = 'service::'

**START\_ROW** = 'start\_row'

**TABULAR** = 'tabular'

**TRUE** = 'true'

**build\_query\_url** (*self, stage\_clauses, fmt='json'*)

Combine one or more RMA query stages into a single RMA query.

### Parameters

**stage\_clauses** [list of strings] subqueries

**fmt** [string, optional] json (default), xml, or csv

**Returns**

**string** complete RMA url

**build\_schema\_query** (*self*, *clazz=None*, *fmt='json'*)

Build the URL that will fetch the data schema.

**Parameters**

**clazz** [string, optional] Name of a specific class or None (default).

**fmt** [string, optional] json (default) or xml

**Returns**

**url** [string] The constructed URL

**Notes**

If a class is specified, only the schema information for that class will be requested, otherwise the url requests the entire schema.

**debug\_clause** (*self*, *debug\_value=None*)

Construct a debug clause for use in an rma::options clause. Parameters ——— debug\_value : string or boolean

True, False, None (default) or 'preview'

**Returns**

**clause** [string] The query clause for inclusion in an RMA query URL.

**Notes**

True will request debugging information in the response. False will request no debugging information. None will return an empty clause. 'preview' will request debugging information without the query being run.

**filter** (*self*, *key*, *value*)

serialize a single RMA query filter clause.

**Parameters**

**key** [string] keys for narrowing a query.

**value** [string] value for narrowing a query.

**Returns**

**string** a single filter clause for an RMA query string.

**filters** (*self*, *filters*)

serialize RMA query filter clauses.

**Parameters**

**filters** [dict] keys and values for narrowing a query.

**Returns**

**string** filter clause for an RMA query string.

**get\_schema** (*self*, *clazz=None*)

Retrieve schema information.

**model\_query** (*self*, \*args, \*\*kwargs)

Construct and execute a model stage of an RMA query string.

#### Parameters

- model** [string] The top level data type
- filters** [dict] key, value comparisons applied to the top-level model to narrow the results.
- criteria** [string] raw RMA criteria clause to choose what object are returned
- include** [string] raw RMA include clause to return associated objects
- only** [list of strings, optional] to be joined into an rma::options only filter to limit what data is returned
- except** [list of strings, optional] to be joined into an rma::options except filter to limit what data is returned
- except** [list of strings, optional] synonym for except parameter to avoid a reserved word conflict.
- tabular** [list of string, optional] return columns as a tabular data structure rather than a nested tree.
- count** [boolean, optional] False to skip the extra database count query.
- debug** [string, optional] 'true', 'false' or 'preview'
- num\_rows** [int or string, optional] how many database rows are returned (may not correspond directly to JSON tree structure)
- start\_row** [int or string, optional] which database row is start of returned data (may not correspond directly to JSON tree structure)

#### Notes

See [RMA Path Syntax](#) for a brief overview of the normalized RMA syntax. Normalized RMA syntax differs from the legacy syntax used in much of the RMA documentation. Using the &debug=true option with an RMA URL will include debugging information in the response, including the normalized query.

**model\_stage** (*self*, *model*, \*\*kwargs)

Construct a model stage of an RMA query string.

#### Parameters

- model** [string] The top level data type
- filters** [dict] key, value comparisons applied to the top-level model to narrow the results.
- criteria** [string] raw RMA criteria clause to choose what object are returned
- include** [string] raw RMA include clause to return associated objects
- only** [list of strings, optional] to be joined into an rma::options only filter to limit what data is returned
- except** [list of strings, optional] to be joined into an rma::options except filter to limit what data is returned
- tabular** [list of string, optional] return columns as a tabular data structure rather than a nested tree.
- count** [boolean, optional] False to skip the extra database count query.

**debug** [string, optional] ‘true’, ‘false’ or ‘preview’

**num\_rows** [int or string, optional] how many database rows are returned (may not correspond directly to JSON tree structure)

**start\_row** [int or string, optional] which database row is start of returned data (may not correspond directly to JSON tree structure)

## Notes

See [RMA Path Syntax](#) for a brief overview of the normalized RMA syntax. Normalized RMA syntax differs from the legacy syntax used in much of the RMA documentation. Using the `&debug=true` option with an RMA URL will include debugging information in the response, including the normalized query.

**only\_except\_tabular\_clause** (*self*, *filter\_type*, *attribute\_list*)

Construct a clause to filter which attributes are returned for use in an `rma::options` clause.

### Parameters

**filter\_type** [string] ‘only’, ‘except’, or ‘tabular’

**attribute\_list** [list of strings] for example [‘acronym’, ‘products.name’, ‘structure.id’]

### Returns

**clause** [string] The query clause for inclusion in an RMA query URL.

## Notes

The title of tabular columns can be set by adding ‘+as+<title>’ to the attribute. The tabular filter type requests a response that is row-oriented rather than a nested structure. Because of this, the tabular option can mask the lazy query behavior of an `rma::include` clause. The tabular option does not mask the inner-join behavior of an `rma::include` clause. The tabular filter is required for .csv format RMA requests.

**options\_clause** (*self*, *\*\*kwargs*)

build `rma::options` clause.

### Parameters

**only** [list of strings, optional]

**except** [list of strings, optional]

**tabular** [list of string, optional]

**count** [boolean, optional]

**debug** [string, optional] ‘true’, ‘false’ or ‘preview’

**num\_rows** [int or string, optional]

**start\_row** [int or string, optional]

**order\_clause** (*self*, *order\_list=None*)

Construct a debug clause for use in an `rma::options` clause.

### Parameters

**order\_list** [list of strings] for example [‘acronym’, ‘products.name+asc’, ‘structure.id+desc’]

### Returns

**clause** [string] The query clause for inclusion in an RMA query URL.



## Notes

Optionally adding ‘+asc’ (default) or ‘+desc’ after an attribute will change the sort order.

**pipe\_stage** (*self*, *pipe\_name*, *parameters*)

Connect model and service stages via their JSON responses.

## Notes

See: [Service Pipelines](#) and [Connected Services and Pipes](#)

**quote\_string** (*self*, *the\_string*)

Wrap a clause in single quotes.

### Parameters

**the\_string** [string] a clause to be included in an rma query that needs to be quoted

### Returns

**string** input wrapped in single quotes

**service\_query** (*self*, *\*args*, *\*\*kwargs*)

Construct and Execute a single-stage RMA query to send a request to a connected service.

### Parameters

**service\_name** [string] Name of a documented connected service.

**parameters** [dict] key-value pairs as in the online documentation.

## Notes

See: [Service Pipelines](#) and [Connected Services and Pipes](#)

**service\_stage** (*self*, *service\_name*, *parameters=None*)

Construct an RMA query fragment to send a request to a connected service.

### Parameters

**service\_name** [string] Name of a documented connected service.

**parameters** [dict] key-value pairs as in the online documentation.

## Notes

See: [Service Pipelines](#) and [Connected Services and Pipes](#)

**tuple\_filters** (*self*, *filters*)

Construct an RMA filter clause.

## Notes

See [RMA Path Syntax - Square Brackets for Filters](#) for additional documentation.

### allensdk.api.queries.rma\_pager module

```
class allensdk.api.queries.rma_pager.RmaPager
    Bases: object

    static pager (fn, *args, **kwargs)

allensdk.api.queries.rma_pager.pageable (total_rows=None, num_rows=None)
```

### allensdk.api.queries.rma\_template module

```
class allensdk.api.queries.rma_template.RmaTemplate (base_uri=None,
                                                    query_manifest=None)
    Bases: allensdk.api.queries.rma_api.RmaApi
    See: Atlas Drawings and Ontologies

    template_query (self, template_name, entry_name, **kwargs)
    to_filter_rhs (self, rhs)
```

### allensdk.api.queries.svg\_api module

```
class allensdk.api.queries.svg_api.SvgApi (base_uri=None)
    Bases: allensdk.api.api.Api

    build_query (self, section_image_id, groups=None, download=False)
        Build the URL that will fetch meta data for the specified structure.

        Parameters

        section_image_id [integer] Key of the object to be retrieved.

        groups [array of integers] Keys of the group labels to filter the svg types that are returned.

        Returns

        url [string] The constructed URL

    download_svg (self, section_image_id, groups=None, file_path=None)
        Download the svg file

    get_svg (self, section_image_id, groups=None)
        Get the svg document.
```

### allensdk.api.queries.synchronization\_api module

```
class allensdk.api.queries.synchronization_api.SynchronizationApi (base_uri=None)
    Bases: allensdk.api.api.Api

    HTTP client for image synchronization services uses the image alignment results from the Informatics Data Processing Pipeline. Note: all locations on SectionImages are reported in pixel coordinates and all locations in 3-D ReferenceSpaces are reported in microns.

    See Image to Image Synchronization for additional documentation.

    get_image_to_atlas (self, section_image_id, x, y, atlas_id)
        For a specified Atlas, find the closest annotated SectionImage and (x,y) location as defined by a seed SectionImage and seed (x,y) location.
```

**Parameters**

**section\_image\_id** [integer] Seed for spatial sync.

**x** [float] Pixel coordinate of the seed location in the seed SectionImage.

**y** [float] Pixel coordinate of the seed location in the seed SectionImage.

**atlas\_id** [int] Target Atlas for image sync.

**Returns**

**dict** The parsed json response

**get\_image\_to\_image** (*self, section\_image\_id, x, y, section\_data\_set\_ids*)

For a list of target SectionDataSets, find the closest SectionImage and (x,y) location as defined by a seed SectionImage and seed (x,y) pixel location.

**Parameters**

**section\_image\_id** [integer] Seed for spatial sync.

**x** [float] Pixel coordinate of the seed location in the seed SectionImage.

**y** [float] Pixel coordinate of the seed location in the seed SectionImage.

**section\_data\_set\_ids** [list of integers] Target SectionDataSet IDs for image sync.

**Returns**

**dict** The parsed json response

**get\_image\_to\_image\_2d** (*self, section\_image\_id, x, y, section\_image\_ids*)

For a list of target SectionImages, find the closest (x,y) location as defined by a seed SectionImage and seed (x,y) location.

**Parameters**

**section\_image\_id** [integer] Seed for image sync.

**x** [float] Pixel coordinate of the seed location in the seed SectionImage.

**y** [float] Pixel coordinate of the seed location in the seed SectionImage.

**section\_image\_ids** [list of ints] Target SectionImage IDs for image sync.

**Returns**

**dict** The parsed json response

**get\_image\_to\_reference** (*self, section\_image\_id, x, y*)

For a specified SectionImage and (x,y) location, return the (x,y,z) location in the ReferenceSpace of the associated SectionDataSet.

**Parameters**

**section\_image\_id** [integer] Seed for image sync.

**x** [float] Pixel coordinate on the specified SectionImage.

**y** [float] Pixel coordinate on the specified SectionImage.

**Returns**

**dict** The parsed json response

**get\_reference\_to\_image** (*self, reference\_space\_id, x, y, z, section\_data\_set\_ids*)

For a list of target SectionDataSets, find the closest SectionImage and (x,y) location as defined by a (x,y,z) location in a specified ReferenceSpace.

**Parameters**

**reference\_space\_id** [integer] Seed for spatial sync.

**x** [float] Coordinate (in microns) of the seed location in the seed ReferenceSpace.

**y** [float] Coordinate (in microns) of the seed location in the seed ReferenceSpace.

**z** [float] Coordinate (in microns) of the seed location in the seed ReferenceSpace.

**section\_data\_set\_ids** [list of ints] Target SectionDataSets IDs for image sync.

**Returns**

**dict** The parsed json response

**get\_structure\_to\_image** (*self, section\_data\_set\_id, structure\_ids*)

For a list of target structures, find the closest SectionImage and (x,y) location as defined by the centroid of each Structure.

**Parameters**

**section\_data\_set\_id** [integer] primary key

**structure\_ids** [list of integers] primary key

**Returns**

**dict** The parsed json response

**allensdk.api.queries.tree\_search\_api module**

**class** allensdk.api.queries.tree\_search\_api.**TreeSearchApi** (*base\_uri=None*)

Bases: [allensdk.api.api.Api](#)

See [Searching a Specimen or Structure Tree](#) for additional documentation.

**get\_tree** (*self, kind, db\_id, ancestors=None, descendants=None*)

Fetch meta data for the specified structure or specimen.

**Parameters**

**kind** [string] ‘Structure’ or ‘Specimen’

**db\_id** [integer] The id of the structure or specimen to search.

**ancestors** [boolean, optional] whether to include ancestors in the response (defaults to False)

**descendants** [boolean, optional] whether to include descendants in the response (defaults to False)

**Returns**

**dict** parsed json response data

**Module contents****Submodules****allensdk.api module**

**class** allensdk.api.**Api** (*api\_base\_url\_string=None*)

Bases: object

**cleanup\_truncated\_file** (*self*, *file\_path*)

Helper for removing files.

#### Parameters

**file\_path** [string] Absolute path including the file name to remove.

**construct\_well\_known\_file\_download\_url** (*self*, *well\_known\_file\_id*)

Join data api endpoint and id.

#### Parameters

**well\_known\_file\_id** [integer or string representing an integer] well known file id

#### Returns

**string** the well-known-file download url for the current api server

See also:

[\*retrieve\\_file\\_over\\_http\*](#) Can be used to retrieve the file from the url.

**default\_api\_url** = 'http://api.brain-map.org'

**do\_query** (*self*, *url\_builder\_fn*, *json\_traversal\_fn*, *\*args*, *\*\*kwargs*)

Bundle an query url construction function with a corresponding response json traversal function.

#### Parameters

**url\_builder\_fn** [function] A function that takes parameters and returns an rma url.

**json\_traversal\_fn** [function] A function that takes a json-parsed python data structure and returns data from it.

**post** [boolean, optional kwarg] True does an HTTP POST, False (default) does a GET

**args** [arguments] Arguments to be passed to the url builder function.

**kwargs** [keyword arguments] Keyword arguments to be passed to the rma builder function.

#### Returns

**any type** The data extracted from the json response.

### Examples

A simple Api subclass example.

**do\_rma\_query** (*self*, *rma\_builder\_fn*, *json\_traversal\_fn*, *\*args*, *\*\*kwargs*)

Bundle an RMA query url construction function with a corresponding response json traversal function.

**..note::** **Deprecated in AllenSDK 0.9.2** *do\_rma\_query* will be removed in AllenSDK 1.0, it is replaced by *do\_query* because the latter is more general.

#### Parameters

**rma\_builder\_fn** [function] A function that takes parameters and returns an rma url.

**json\_traversal\_fn** [function] A function that takes a json-parsed python data structure and returns data from it.

**args** [arguments] Arguments to be passed to the rma builder function.

**kwargs** [keyword arguments] Keyword arguments to be passed to the rma builder function.

**Returns**

**any type** The data extracted from the json response.

**Examples**

A simple `Api` subclass example.

```
download_url = 'http://download.alleninstitute.org'
```

```
json_msg_query(self, url, dataframe=False)
```

**Common case where the url is fully constructed** and the response data is stored in the 'msg' field.

**Parameters**

**url** [string] Where to get the data in json form

**dataframe** [boolean] True converts to a pandas dataframe, False (default) doesn't

**Returns**

**dict or DataFrame** returned data; type depends on dataframe option

```
load_api_schema(self)
```

Download the RMA schema from the current RMA endpoint

**Returns**

**dict** the parsed json schema message

**Notes**

This information and other [Allen Brain Atlas Data Portal Data Model](#) documentation is also available as a [Class Hierarchy](#) and [Class List](#).

```
read_data(self, parsed_json)
```

Return the message data from the parsed query.

**Parameters**

**parsed\_json** [dict] A python structure corresponding to the JSON data returned from the API.

**Notes**

See [API Response Formats - Response Envelope](#) for additional documentation.

```
retrieve_file_over_http(self, url, file_path, zipped=False)
```

Get a file from the data api and save it.

**Parameters**

**url** [string] `Url[R099781a1d33c-1]_` from which to get the file.

**file\_path** [string] Absolute path including the file name to save.

**zipped** [bool, optional] If true, assume that the response is a zipped directory and attempt to extract contained files into the directory containing `file_path`. Default is False.

**See also:**

`construct_well_known_file_download_url` Can be used to construct the url.

## References

[1]

**retrieve\_parsed\_json\_over\_http** (*self, url, post=False*)

Get the document and put it in a Python data structure

### Parameters

**url** [string] Full API query url.

**post** [boolean] True does an HTTP POST, False (default) encodes the URL and does a GET

### Returns

**dict** Result document as parsed by the JSON library.

**retrieve\_xml\_over\_http** (*self, url*)

Get the document and put it in a Python data structure

### Parameters

**url** [string] Full API query url.

### Returns

**string** Unparsed xml string.

**set\_api\_urls** (*self, api\_base\_url\_string*)

Set the internal RMA and well known file download endpoint urls based on a api server endpoint.

### Parameters

**api\_base\_url\_string** [string] url of the api to point to

**set\_default\_working\_directory** (*self, working\_directory*)

Set the working directory where files will be saved.

### Parameters

**working\_directory** [string] the absolute path string of the working directory.

`allensdk.api.api.stream_file_over_http` (*url, file\_path, timeout=(9.05, 31.1)*)

Supply an http get request and stream the response to a file.

### Parameters

**url** [str] Send the request to this url

**file\_path** [str] Stream the response to this path

**timeout** [float or tuple of float, optional] Specify a timeout for the request. If a tuple, specify separate connect and read timeouts.

`allensdk.api.api.stream_zip_directory_over_http` (*url, directory, members=None, timeout=(9.05, 31.1)*)

Supply an http get request and stream the response to a file.

### Parameters

**url** [str] Send the request to this url

**directory** [str] Extract the response to this directory

**members** [list of str, optional] Extract only these files

**timeout** [float or tuple of float, optional] Specify a timeout for the request. If a tuple, specify separate connect and read timeouts.

### allensdk.api.cache module

**class** allensdk.api.cache.Cache (manifest=None, cache=True, version=None, \*\*kwargs)  
Bases: object

**add\_manifest\_paths** (self, manifest\_builder)

Add cache-class specific paths to the manifest. In derived classes, should call super.

**build\_manifest** (self, file\_name)

Creation of default path specifications.

#### Parameters

**file\_name** [string] where to save it

**static** cache\_csv ()

**static** cache\_csv\_dataframe ()

**static** cache\_csv\_json ()

**static** cache\_json ()

**static** cache\_json\_dataframe ()

**static** cacher (fn, \*args, \*\*kwargs)

make an rma query, save it and return the dataframe.

#### Parameters

**fn** [function reference] makes the actual query using kwargs.

**path** [string] where to save the data

**strategy** [string or None, optional] 'create' always generates the data, 'file' loads from disk, 'lazy' queries the server if no file exists, None generates the data and bypasses all caching behavior

**pre** [function] dfljson->dfljson, takes one data argument and returns filtered version, None for pass-through

**post** [function] dfljson->?, takes one data argument and returns Object

**reader** [function, optional] path -> data, default NOP

**writer** [function, optional] path, data -> None, default NOP

**kwargs** [objects] passed through to the query function

#### Returns

**Object or None** data type depends on fn, reader and/or post methods.

**static** csv\_writer (pth, gen)

**get\_cache\_path** (self, file\_name, manifest\_key, \*args)

Helper method for accessing path specs from manifest keys.

#### Parameters

**file\_name** [string]

**manifest\_key** [string]



**args** [ordered parameters]

#### Returns

**string or None** path

**static json\_remove\_keys** (*data, keys*)

**static json\_rename\_columns** (*data, new\_old\_name\_tuples=None*)

Convenience method to rename columns in a pandas dataframe.

#### Parameters

**data** [dataframe] edited in place.

**new\_old\_name\_tuples** [list of string tuples (new, old)]

**load\_csv** (*self, path, rename=None, index=None*)

Read a csv file as a pandas dataframe.

#### Parameters

**rename** [list of string tuples (new old), optional] columns to rename

**index** [string, optional] post-rename column to use as the row label.

**load\_json** (*self, path, rename=None, index=None*)

Read a json file as a pandas dataframe.

#### Parameters

**rename** [list of string tuples (new old), optional] columns to rename

**index** [string, optional] post-rename column to use as the row label.

**load\_manifest** (*self, file\_name, version=None*)

Read a keyed collection of path specifications.

#### Parameters

**file\_name** [string] path to the manifest file

#### Returns

**Manifest**

**manifest\_dataframe** (*self*)

Convenience method to view manifest as a pandas dataframe.

**static nocache\_dataframe** ()

**static nocache\_json** ()

**static pathfinder** (*file\_name\_position,* *secondary\_file\_name\_position=None,*  
*path\_keyword=None*)

helper method to find path argument in legacy methods written prior to the @cacheable decorator. Do not use for new @cacheable methods.

#### Parameters

**file\_name\_position** [integer] zero indexed position in the decorated method args where file path may be found.

**secondary\_file\_name\_position** [integer] zero indexed position in the decorated method args where the file path may be found.

**path\_keyword** [string] kwarg that may have the file path.

## Notes

This method is only intended to provide backward-compatibility for some methods that otherwise do not follow the path conventions of the `@cacheable` decorator.

**static remove\_keys** (*data*, *keys=None*)

DataFrame version

**static rename\_columns** (*data*, *new\_old\_name\_tuples=None*)

Convenience method to rename columns in a pandas dataframe.

### Parameters

**data** [dataframe] edited in place.

**new\_old\_name\_tuples** [list of string tuples (new, old)]

**wrap** (*self*, *fn*, *path*, *cache*, *save\_as\_json=True*, *return\_dataframe=False*, *index=None*, *rename=None*, *\*\*kwargs*)

make an rma query, save it and return the dataframe.

### Parameters

**fn** [function reference] makes the actual query using kwargs.

**path** [string] where to save the data

**cache** [boolean] True will make the query, False just loads from disk

**save\_as\_json** [boolean, optional] True (default) will save data as json, False as csv

**return\_dataframe** [boolean, optional] True will cast the return value to a pandas dataframe, False (default) will not

**index** [string, optional] column to use as the pandas index

**rename** [list of string tuples, optional] (new, old) columns to rename

**kwargs** [objects] passed through to the query function

### Returns

**dict or DataFrame** data type depends on return\_dataframe option.

## Notes

Column renaming happens after the file is reloaded for json

`allensdk.api.cache.cacheable` (*strategy=None*, *pre=None*, *writer=None*, *reader=None*, *post=None*, *pathfinder=None*)

decorator for rma queries, save it and return the dataframe.

### Parameters

**fn** [function reference] makes the actual query using kwargs.

**path** [string] where to save the data

**strategy** [string or None, optional] 'create' always gets the data from the source (server or generated), 'file' loads from disk, 'lazy' creates the data and saves to file if no file exists, None queries the server and bypasses all caching behavior

**pre** [function] dfljson->dfljson, takes one data argument and returns filtered version, None for pass-through

**post** [function] dfljson->?, takes one data argument and returns Object

**reader** [function, optional] path -> data, default NOP

**writer** [function, optional] path, data -> None, default NOP

**kwargs** [objects] passed through to the query function

### Returns

**dict or DataFrame** data type depends on dataframe option.

### Notes

Column renaming happens after the file is reloaded for json

`allensdk.api.cache.get_default_manifest_file(cache_name)`

`allensdk.api.cache.memoize(f)`

Creates an unbound cache of function calls and results. Note that arguments of different types are not cached separately (so `f(3.0)` and `f(3)` are not treated as distinct calls)

Arguments to the cached function must be hashable.

View the cache size with `f.cache_size()`. Clear the cache with `f.cache_clear()`. Access the underlying function with `f.__wrapped__`.

### allensdk.api.caching\_utilities module

```
allensdk.api.caching_utilities.call_caching(fetch: Callable[[Q], ~Q], write:
                                         Callable[[~Q], NoneType], read:
                                         Union[Callable[[Q], ~P], NoneType] =
                                         None, pre_write: Union[Callable[[~Q],
                                         ~Q], NoneType] = None, cleanup:
                                         Union[Callable[[Q], NoneType], None-
                                         Type] = None, lazy: bool = True, num_tries:
                                         int = 1, failure_message: str = "") →
                                         Union[~P, NoneType]
```

Access data, caching on a local store for future accesses.

### Parameters

**fetch** : Function which pulls data from a remote/expensive source.

**write** : Function which stores data in a local/inexpensive store.

**read** : Function which pulls data from a local/inexpensive store.

**pre\_write** : Function applied to obtained data after fetching, but before writing.

**cleanup** : Function for fixing a failed fetch. e.g. unlinking a partially downloaded file. Exceptions raised by cleanup are not themselves handled

**lazy** : If True, attempt to read the data from the local/inexpensive store before fetching it. If False, forcibly fetch from the remote/expensive store.

**num\_tries** : How many fetches to attempt before (re)raising an exception. A fetch is failed if reading the result raises an exception.

**failure\_message** : Provides additional context in the event of a failed download. Emitted when retrying, and when a fetch failure occurs after tries are exhausted

### Returns

### The result of calling read

```
allensdk.api.caching_utilities.one_file_call_caching(path: Union[pathlib.Path, str],
                                                    fetch: Callable[[], ~Q], write:
                                                    Callable[[Union[pathlib.Path,
                                                                    str], ~Q], NoneType], read:
                                                    Union[Callable[[Union[pathlib.Path,
                                                                    str]], ~P], NoneType]
                                                    = None, pre_write:
                                                    Union[Callable[[~Q], ~Q],
                                                    NoneType] = None, cleanup:
                                                    Union[Callable[[], NoneType],
                                                    NoneType] = None, lazy: bool
                                                    = True, num_tries: int = 1,
                                                    failure_message: str = ") →
                                                    Union[~P, NoneType]
```

A call\_caching variant where the local store is a single file. See call\_caching for complete documentation.

### Parameters

**path** : Path at which the data will be stored

## Module contents

Subclasses of allensdk.api.api.Api to implement specific queries to the [Allen Brain Atlas Data Portal](#).

## 6.1.2 allensdk.brain\_observatory package

### Subpackages

**allensdk.brain\_observatory.behavior package**

### Subpackages

**allensdk.brain\_observatory.behavior.behavior\_ophys\_api package**

### Submodules

**allensdk.brain\_observatory.behavior.behavior\_ophys\_api.behavior\_ophys\_nwb\_api module**

**class** allensdk.brain\_observatory.behavior.behavior\_ophys\_api.behavior\_ophys\_nwb\_api.BehaviorOphysNwbApi

Bases: *allensdk.brain\_observatory.nwb.nwb\_api.NwbApi*, *allensdk.brain\_observatory.behavior.behavior\_ophys\_api.BehaviorOphysApiBase*

**get\_average\_projection** (*self*, *image\_api=None*) → SimpleITK.SimpleITK.Image

**get\_cell\_specimen\_table** (*self*) → pandas.core.frame.DataFrame

**get\_corrected\_fluorescence\_traces** (*self*) → pandas.core.frame.DataFrame

**get\_dff\_traces** (*self*) → pandas.core.frame.DataFrame

**get\_ticks** (*self*) → numpy.ndarray

```

get_max_projection (self, image_api=None) → SimpleITK.SimpleITK.Image
get_metadata (self) → dict
get_motion_correction (self) → pandas.core.frame.DataFrame
get_ophys_timestamps (self) → numpy.ndarray
get_rewards (self) → numpy.ndarray
get_running_data_df (self, **kwargs)
get_segmentation_mask_image (self, image_api=None) → SimpleITK.SimpleITK.Image
get_stimulus_templates (self, **kwargs)
get_stimulus_timestamps (self) → numpy.ndarray
get_task_parameters (self) → dict
get_trials (self) → pandas.core.frame.DataFrame
save (self, session_object)

allensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_nwb_api.compare_fields(A, B, reraise=

```

## Module contents

```

class allensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysApiBase
    Bases: object

    get_average_projection (self)
    get_cell_specimen_table (self)
    get_corrected_fluorescence_traces (self)
    get_dff_traces (self)
    get_eye_tracking_data (self)
    get_licks (self)
    get_max_projection (self)
    get_metadata (self)
    get_motion_correction (self)
    get_ophys_experiment_id (self) → int
    get_ophys_timestamps (self)
    get_rewards (self)
    get_running_data_df (self)
    get_running_speed (self)
    get_segmentation_mask_image (self)
    get_stimulus_presentations (self)

```

```
get_stimulus_templates(self)
get_stimulus_timestamps(self)
get_task_parameters(self)
get_trials(self)
```

## allensdk.brain\_observatory.behavior.internal package

### Submodules

#### allensdk.brain\_observatory.behavior.internal.behavior\_base module

```
class allensdk.brain_observatory.behavior.internal.behavior_base.BehaviorBase
    Bases: abc.ABC
```

Abstract base class implementing required methods for interacting with behavior session data.

Child classes should be instantiated with a fetch API that implements these methods.

```
get_licks(self) → pandas.core.frame.DataFrame
    Get lick data from.pkl file.
```

#### Returns

**np.ndarray** A dataframe containing lick timestamps.

```
get_rewards(self) → pandas.core.frame.DataFrame
    Get reward data from.pkl file.
```

#### Returns

**pd.DataFrame** A dataframe containing timestamps of delivered rewards.

```
get_running_data_df(self) → pandas.core.frame.DataFrame
    Get running speed data.
```

#### Returns

**pd.DataFrame** Dataframe containing various signals used to compute running speed.

```
get_running_speed(self) → allensdk.brain_observatory.running_speed.RunningSpeed
    Get running speed using timestamps from self.get_stimulus_timestamps.
```

NOTE: Do not correct for monitor delay.

#### Returns

**RunningSpeed (NamedTuple with two fields)**

**timestamps** [np.ndarray] Timestamps of running speed data samples

**values** [np.ndarray] Running speed of the experimental subject (in cm / s).

```
get_stimulus_presentations(self) → pandas.core.frame.DataFrame
    Get stimulus presentation data.
```

NOTE: Uses timestamps that do not account for monitor delay.

#### Returns

**pd.DataFrame** Table whose rows are stimulus presentations (i.e. a given image, for a given duration, typically 250 ms) and whose columns are presentation characteristics.

**get\_stimulus\_templates** (*self*) → Dict[str, numpy.ndarray]

Get stimulus templates (movies, scenes) for behavior session.

#### Returns

**Dict[str, np.ndarray]** A dictionary containing the stimulus images presented during the session. Keys are data set names, and values are 3D numpy arrays.

**get\_stimulus\_timestamps** (*self*) → numpy.ndarray

Get stimulus timestamps from pkl file.

NOTE: Located with behavior\_session\_id

#### Returns

**np.ndarray** Timestamps associated with stimulus presentations on the monitor that do not account for monitor delay.

**get\_task\_parameters** (*self*) → dict

Get task parameters from pkl file.

#### Returns

**dict** A dictionary containing parameters used to define the task runtime behavior.

**get\_trials** (*self*) → pandas.core.frame.DataFrame

Get trials from pkl file

#### Returns

**pd.DataFrame** A dataframe containing behavioral trial start/stop times, and trial data

## allensdk.brain\_observatory.behavior.internal.behavior\_ophys\_base module

**class** allensdk.brain\_observatory.behavior.internal.behavior\_ophys\_base.**BehaviorOphysBase**

Bases: [allensdk.brain\\_observatory.behavior.internal.behavior\\_base.BehaviorBase](#)

Abstract base class implementing required methods for interacting with behavior+ophys session data.

Child classes should be instantiated with a fetch API that implements these methods. Both fetch API and session object should inherit from this base.

**get\_average\_projection** (*self*) → allensdk.brain\_observatory.behavior.image\_api.Image

Get an image whose values are the average obtained values at each pixel of the ophys movie over time.

#### Returns

**allensdk.brain\_observatory.behavior.image\_api.Image:** Array-like interface to avg projection image data and metadata.

**get\_cell\_specimen\_table** (*self*) → pandas.core.frame.DataFrame

Get a cell specimen dataframe containing ROI information about cells identified in an ophys experiment.

#### Returns

**pd.DataFrame** Cell ROI information organized into a dataframe. Index is the cell ROI IDs.

**get\_corrected\_fluorescence\_traces** (*self*) → pandas.core.frame.DataFrame

Get motion-corrected fluorescence traces.

#### Returns

**pd.DataFrame** Motion-corrected fluorescence traces organized into a dataframe. Index is the cell ROI IDs.

**get\_dff\_traces** (*self*) → pandas.core.frame.DataFrame  
Get a table of delta fluorescence over fluorescence traces.

**Returns**

**pd.DataFrame** The traces of dff (normalized fluorescence) organized into a dataframe. Index is the cell ROI IDs.

**get\_max\_projection** (*self*) → allensdk.brain\_observatory.behavior.image\_api.Image  
Get an image whose values are the maximum obtained values at each pixel of the ophys movie over time.

**Returns**

**allensdk.brain\_observatory.behavior.image\_api.Image:** Array-like interface to max projection image data and metadata.

**get\_metadata** (*self*) → dict  
Get behavior+ophys session metadata.

**Returns**

**dict** A dictionary of session-specific metadata.

**get\_motion\_correction** (*self*) → pandas.core.frame.DataFrame  
Get motion correction trace data.

**Returns**

**pd.DataFrame** A dataframe containing trace data used during motion correction computation.

**get\_ophys\_timestamps** (*self*) → numpy.ndarray  
Get optical physiology frame timestamps.

**Returns**

**np.ndarray** Timestamps associated with frames captured by the microscope.

**get\_raw\_stimulus\_timestamps** (*self*) → numpy.ndarray  
Get raw stimulus timestamps.

**Returns**

**np.ndarray** Timestamps associated with stimulus presentations on the monitor without accounting for monitor delay.

**get\_stimulus\_presentations** (*self*) → pandas.core.frame.DataFrame  
Get stimulus presentation data.

NOTE: Uses monitor delay corrected stimulus timestamps.

**Returns**

**pd.DataFrame** Table whose rows are stimulus presentations (i.e. a given image, for a given duration, typically 250 ms) and whose columns are presentation characteristics.

**get\_stimulus\_timestamps** (*self*) → numpy.ndarray  
Get stimulus timestamps.

**Returns**

**np.ndarray** Timestamps associated with stimulus presentations on the monitor after accounting for monitor delay.



**allensdk.brain\_observatory.behavior.internal.behavior\_project\_base module**

**class** allensdk.brain\_observatory.behavior.internal.behavior\_project\_base.**BehaviorProjectBase**

Bases: abc.ABC

**get\_behavior\_only\_session\_data** (*self*, *behavior\_session\_id*: int) → allensdk.brain\_observatory.behavior.behavior\_data\_session.BehaviorDataSession  
Returns a BehaviorDataSession object that contains methods to analyze a single behavior session. :param behavior\_session\_id: id that corresponds to a behavior session :type behavior\_session\_id: int :rtype: BehaviorDataSession

**get\_behavior\_only\_session\_table** (*self*) → pandas.core.frame.DataFrame  
Returns a pd.DataFrame table with all behavior session\_ids to the user with additional metadata. :rtype: pd.DataFrame

**get\_natural\_movie\_template** (*self*, *number*: int) → Iterable[bytes]  
Download a template for the natural scene stimulus. This is the actual image that was shown during the recording session. :param number: identifier for this movie (note that this is an int, so to get the template for natural\_movie\_three should pass 3)

**Returns** iterable yielding a tiff file as bytes

**get\_natural\_scene\_template** (*self*, *number*: int) → Iterable[bytes]  
Download a template for the natural movie stimulus. This is the actual movie that was shown during the recording session. :param number: identifier for this scene :type number: int :returns: An iterable yielding an npy file as bytes

**get\_session\_data** (*self*, *ophys\_session\_id*: int) → allensdk.brain\_observatory.behavior.behavior\_ophys\_session.BehaviorOphysSession  
Returns a BehaviorOphysSession object that contains methods to analyze a single behavior+ophys session. :param ophys\_session\_id: id that corresponds to a behavior session :type ophys\_session\_id: int :rtype: BehaviorOphysSession

**get\_session\_table** (*self*) → pandas.core.frame.DataFrame  
Return a pd.DataFrame table with all ophys\_session\_ids and relevant metadata.

**Module contents****allensdk.brain\_observatory.behavior.sync package****Submodules****allensdk.brain\_observatory.behavior.sync.process\_sync module**

allensdk.brain\_observatory.behavior.sync.process\_sync.**calculate\_delay** (*sync\_data*,  
*stim\_vsync\_fall*,  
*sample\_frequency*)

allensdk.brain\_observatory.behavior.sync.process\_sync.**filter\_digital** (*rising*,  
*falling*,  
*threshold*=0.0001)

Removes short transients from digital signal.

**Rising and falling should be same length and units** in seconds.

**Kwargs:** threshold (float): transient width

## Module contents

Created on Sunday July 15 2018

@author: marinag

`allensdk.brain_observatory.behavior.sync.frame_time_offset` (*data: Dict[str, Any]*)  
→ float

Contained in the behavior “pickle” file is a series of time between consecutive vsync frames (*intervalsms*). This information required to get the timestamp (via frame number) for events that occurred outside of a trial (e.g. licks). However, we don’t have the value in the trial log time stream when the first vsync frame actually occurred – so we estimate it with a linear regression (frame number x time). All trials in the *trial\_log* have events for *trial\_start* and *trial\_end*, so these are used to fit the regression. A linear regression is used rather than just subtracting the time from the first trial, since there can be some jitter given the 60Hz refresh rate.

### Parameters

**data: dict** behavior pickle well-known file data

### Returns

**float** Time offset to add to the vsync stream to sync it with the *trial\_log* time stream. The “zero-th” frame time.

`allensdk.brain_observatory.behavior.sync.get_behavior_monitoring` (*dataset: al-*  
*lensdk.brain\_observatory.sync\_dataset.Dataset,*  
*permissive: bool = False*) →  
*Union[numpy.ndarray, NoneType]*

Report the timestamps of each frame of the behavior monitoring video

### Parameters

**dataset** [describes experiment timing]

**permissive** [If True, None will be returned if timestamps are not found. If False, a KeyError will be raised]

### Returns

**array of timestamps (floating point; seconds; relative to experiment start)** or None. If None, no behavior monitoring timestamps were found in this sync dataset.

`allensdk.brain_observatory.behavior.sync.get_eye_tracking` (*dataset: al-*  
*lensdk.brain\_observatory.sync\_dataset.Dataset,*  
*permissive: bool = False*) →  
*Union[numpy.ndarray, NoneType]*

Report the timestamps of each frame of the eye tracking video

### Parameters

**dataset** [describes experiment timing]

**permissive** [If True, None will be returned if timestamps are not found. If False, a KeyError will be raised]

**Returns**

**array of timestamps (floating point; seconds; relative to experiment start)** or `None`. If `None`, no eye tracking timestamps were found in this sync dataset.

```
allensdk.brain_observatory.behavior.sync.get_lick_times(dataset: al-
                                                         lensdk.brain_observatory.sync_dataset.Dataset,
                                                         permissive: bool = False)
                                                         → Union[numpy.ndarray,
                                                         NoneType]
```

Report the timestamps of each detected lick

**Parameters**

**dataset** [describes experiment timing]

**permissive** [If True, `None` will be returned if timestamps are not found. If False, a `KeyError` will be raised]

**Returns**

**array of timestamps (floating point; seconds; relative to experiment start)** or `None`. If `None`, no lick timestamps were found in this sync dataset.

```
allensdk.brain_observatory.behavior.sync.get_ophys_frames(dataset: al-
                                                           lensdk.brain_observatory.sync_dataset.Dataset,
                                                           permissive: bool
                                                           = False) →
                                                           numpy.ndarray
```

Report the timestamps of each optical physiology video frame

**Parameters**

**dataset** [describes experiment timing]

**Returns**

**array of timestamps (floating point; seconds; relative to experiment start).**

**permissive** [If True, `None` will be returned if timestamps are not found. If False, a `KeyError` will be raised]

**Notes**

use rising edge for Scientifica, falling edge for Nikon <http://confluence.corp.alleninstitute.org/display/IT/Ophys+Time+Sync> This function uses rising edges

```
allensdk.brain_observatory.behavior.sync.get_raw_stimulus_frames(dataset: al-
                                                                  lensdk.brain_observatory.sync_dataset
                                                                  permissive:
                                                                  bool =
                                                                  False) →
                                                                  numpy.ndarray
```

Report the raw timestamps of each stimulus frame. This corresponds to the time at which the psychopy window's flip method returned, but not necessarily to the time at which the stimulus frame was displayed.

**Parameters**

**dataset** [describes experiment timing]

**permissive** [If True, `None` will be returned if timestamps are not found. If False, a `KeyError` will be raised]

**Returns**

**array of timestamps (floating point; seconds; relative to experiment start).**

```
allensdk.brain_observatory.behavior.sync.get_stim_photodiode (dataset: al-  
                                                             lensdk.brain_observatory.sync_dataset.Data  
                                                             permissive: bool  
                                                             = False) →  
                                                             Union[List[float],  
                                                             NoneType]
```

Report the timestamps of each detected sync square transition (both black -> white and white -> black) in this experiment.

**Parameters**

**dataset** [describes experiment timing]

**permissive** [If True, None will be returned if timestamps are not found. If False, a KeyError will be raised]

**Returns**

**array of timestamps (floating point; seconds; relative to experiment start) or None.** If None, no photodiode timestamps were found in this sync dataset.

```
allensdk.brain_observatory.behavior.sync.get_stimulus_rebase_function (data,  
                                                                       stim-  
                                                                       u-  
                                                                       lus_timestamps_no_monitor_a
```

Create a rebase function to align times for licks and stimulus timestamps in the “pickle” log with the same events in the event “sync” log.

```
allensdk.brain_observatory.behavior.sync.get_sync_data (sync_path: str, permissive:  
                                                         bool = False) → Dict[str,  
                                                         Union[List, numpy.ndarray,  
                                                         NoneType]]
```

Convenience function for extracting several timestamp arrays from a sync file.

**Parameters**

**sync\_path** [The hdf5 file here ought to be a Visual Behavior sync output] file. See allensdk.brain\_observatory.sync\_dataset for more details of this format.

**permissive** [If True, None will be returned if timestamps are not found. If False, a KeyError will be raised]

**Returns**

**A dictionary with the following keys. All timestamps in seconds:** ophys\_frames : timestamps of each optical physiology frame lick\_times : timestamps of each detected lick ophys\_trigger : The time at which ophys acquisition was started eye\_tracking : timestamps of each eye tracking video frame behavior\_monitoring : timestamps of behavior monitoring video frame stim\_photodiode : timestamps of each photodiode transition stimulus\_times\_no\_delay : raw stimulus frame timestamps

**Some values may be None. This indicates that the corresponding timestamps were not located in this sync file.**

```
allensdk.brain_observatory.behavior.sync.get_trigger(dataset: al-
lensdk.brain_observatory.sync_dataset.Dataset,
permissive: bool = False) →
Union[numpy.ndarray, None-
Type]
```

**Returns (as a 1-element array) the time at which optical physiology** acquisition was started.

#### Parameters

**dataset** [describes experiment timing]

**permissive** [If True, None will be returned if timestamps are not found. If ] False, a Key-Error will be raised

#### Returns

**timestamps (floating point; seconds; relative to experiment start)** or None. If None, no timestamps were found in this sync dataset.

#### Notes

**Ophys frame timestamps can be recorded before acquisition start when** experimenters are setting up the recording session. These do not correspond to acquired ophys frames.

### allensdk.brain\_observatory.behavior.write\_nwb package

#### Module contents

#### Submodules

### allensdk.brain\_observatory.behavior.behavior\_data\_session module

```
class allensdk.brain_observatory.behavior.behavior_data_session.BehaviorDataSession (api:
Optional[Type]
=
None)
```

Bases: object

#### **behavior\_session\_id**

Unique identifier for this experimental session. :rtype: int

#### **cache\_clear** (self) → None

Convenience method to clear the api cache, if applicable.

#### **classmethod from\_lims** (behavior\_session\_id: int) → 'BehaviorDataSession'

#### **classmethod from\_nwb\_path** (nwb\_path: str, \*\*api\_kwargs: Any) → 'BehaviorDataSession'

#### **licks**

Get lick data from pkl file.

#### Returns

**np.ndarray** A dataframe containing lick timestamps.

**list\_api\_methods** (*self*) → List[Tuple[str, str]]

Convenience method to expose list of API *get* methods. These methods can be accessed by referencing the API used to initialize this BehaviorDataSession via its *api* instance attribute. :rtype: list of tuples, where the first value in the tuple is the method name, and the second value is the method docstring.

**metadata**

Return metadata about the session. :rtype: dict

**ophys\_experiment\_ids**

The unique identifiers for the ophys experiment(s) associated with this behavior session (if one exists) :rtype: int

**ophys\_session\_id**

The unique identifier for the ophys session associated with this behavior session (if one exists) :rtype: int

**rewards**

Get reward data from pkl file.

**Returns**

**pd.DataFrame** A dataframe containing timestamps of delivered rewards.

**running\_data\_df**

Get running speed data.

**Returns**

**pd.DataFrame** Dataframe containing various signals used to compute running speed.

**running\_speed**

Get running speed using timestamps from self.get\_stimulus\_timestamps.

NOTE: Do not correct for monitor delay.

**Returns**

**RunningSpeed (NamedTuple with two fields)**

**timestamps** [np.ndarray] Timestamps of running speed data samples

**values** [np.ndarray] Running speed of the experimental subject (in cm / s).

**stimulus\_presentations**

Get stimulus presentation data.

NOTE: Uses timestamps that do not account for monitor delay.

**Returns**

**pd.DataFrame** Table whose rows are stimulus presentations (i.e. a given image, for a given duration, typically 250 ms) and whose columns are presentation characteristics.

**stimulus\_templates**

Get stimulus templates (movies, scenes) for behavior session.

**Returns**

**Dict[str, np.ndarray]** A dictionary containing the stimulus images presented during the session. Keys are data set names, and values are 3D numpy arrays.

**stimulus\_timestamps**

Get stimulus timestamps from pkl file.

NOTE: Located with behavior\_session\_id

**Returns**

**np.ndarray** Timestamps associated with stimulus presentations on the monitor that do not account for monitor delay.

**task\_parameters**

Get task parameters from pkl file.

**Returns**

**dict** A dictionary containing parameters used to define the task runtime behavior.

**trials**

Get trials from pkl file

**Returns**

**pd.DataFrame** A dataframe containing behavioral trial start/stop times, and trial data

## allensdk.brain\_observatory.behavior.behavior\_ophys\_analysis module

**class** allensdk.brain\_observatory.behavior.behavior\_ophys\_analysis.**BehaviorOphysAnalysis** (session, api)

Bases: *allensdk.core.lazy\_property.lazy\_property\_mixin.LazyPropertyMixin*

**plot\_example\_traces\_and\_behavior** (self, N=10)

allensdk.brain\_observatory.behavior.behavior\_ophys\_analysis.**plot\_example\_traces\_and\_behavior**

allensdk.brain\_observatory.behavior.behavior\_ophys\_analysis.**plot\_trace** (timestamps, trace, ax=None, xlabel='time (seconds)', ylabel='fluorescence', title='roi')

**allensdk.brain\_observatory.behavior.behavior\_ophys\_session module**

```
class allensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession (api=None, eye_tracking=True, float_precision=3.0, eye_tracking_resolution=100, eye_tracking_buffer=2)
```

Bases: `allensdk.brain_observatory.session_api_utils.ParamsMixin`

Represents data from a single Visual Behavior Ophys imaging session. Can be initialized with an api that fetches data, or by using class methods `from_lims` and `from_nwb_path`.

**average\_projection**

2D image of the microscope field of view, averaged across the experiment :rtype: pandas.DataFrame

**cache\_clear** (*self*) → None

Convenience method to clear the api cache, if applicable.

**cell\_specimen\_table**

Cell roi information organized into a dataframe; index is the cell roi ids. :rtype: pandas.DataFrame

**corrected\_fluorescence\_traces**

The motion-corrected fluorescence traces organized into a dataframe; index is the cell roi ids. :rtype: pandas.DataFrame

**deserialize\_image** (*self*, *sitk\_image*)

Convert SimpleITK image returned by the api to an Image class:

**Args:** *sitk\_image* (SimpleITK image): image object returned by the api

**Returns** *img* (allensdk.brain\_observatory.behavior.image\_api.Image)

**dff\_traces**

Traces of dff organized into a dataframe; index is the cell roi ids. :rtype: pandas.DataFrame

**eye\_tracking**

A dataframe containing ellipse fit parameters for the eye, pupil and corneal reflection (cr). Fits are derived from tracking points from a DeepLabCut model applied to video frames of a subject's right eye. Raw tracking points and raw video frames are not exposed by the SDK.

Notes: - All columns starting with '**pupil\_**' represent ellipse fit parameters

relating to the pupil.

- All columns starting with '**eye\_**' represent ellipse fit parameters relating to the eyelid.
- All columns starting with '**cr\_**' represent ellipse fit parameters relating to the corneal reflection, which is caused by an infrared LED positioned near the eye tracking camera.
- All positions are in units of pixels.
- All areas are in units of pixels^2
- All values are in the coordinate space of the eye tracking camera, NOT the coordinate space of the stimulus display (i.e. this is not gaze location), with (0, 0) being the upper-left corner of the eye-tracking image.



- The ‘likely\_blink’ column is True for any row (frame) where the pupil fit failed OR eye fit failed OR an outlier fit was identified.
- All ellipse fits are derived from tracking points that were output by a DeepLabCut model that was trained on hand-annotated data from a subset of imaging sessions on optical physiology rigs.
- Raw DeepLabCut tracking points are not publicly available.

**Return type** pandas.DataFrame

**classmethod from\_lims** (*ophys\_experiment\_id: int, eye\_tracking\_z\_threshold: float = 3.0, eye\_tracking\_dilation\_frames: int = 2*) → ‘BehaviorOphysSession’

**classmethod from\_nwb\_path** (*nwb\_path: str, \*\*api\_kwargs: Any*) → ‘BehaviorOphysSession’

**get\_average\_projection** (*self*)

Returns an image whose values are the average obtained values at each pixel of the ophys movie over time.

**Returns**

**allensdk.brain\_observatory.behavior.image\_api.Image:** array-like interface to max projection image data and metadata

**get\_cell\_specimen\_ids** (*self*)

**get\_cell\_specimen\_indices** (*self, cell\_specimen\_ids*)

**get\_dff\_traces** (*self, cell\_specimen\_ids=None*)

**get\_max\_projection** (*self*)

Returns an image whose values are the maximum obtained values at each pixel of the ophys movie over time.

**Returns**

**allensdk.brain\_observatory.behavior.image\_api.Image:** array-like interface to max projection image data and metadata

**get\_performance\_metrics** (*self, engaged\_trial\_reward\_rate\_threshold=2*)

**get\_reward\_rate** (*self*)

**get\_roi\_masks** (*self, cell\_specimen\_ids=None*)

Obtains boolean masks indicating the location of one or more cell’s ROIs in this session.

**Parameters**

**cell\_specimen\_ids** [array-like of int, optional] ROI masks for these cell specimens will be returned. The default behavior is to return masks for all cell specimens.

**Returns**

**result** [xr.DataArray]

**dimensions are:**

- **cell\_specimen\_id** : which cell’s roi is described by this mask?
- **row** : index within the underlying image
- **column** : index within the image

values are 1 where an ROI was present, otherwise 0.

**get\_rolling\_performance\_df** (*self*)

**get\_segmentation\_mask\_image** (*self*)

Returns an image with value 1 if the pixel was included in an ROI, and 0 otherwise

**Returns**

**allensdk.brain\_observatory.behavior.image\_api.Image:** array-like interface to segmentation\_mask image data and metadata

**licks**

A dataframe containing lick timestamps. :rtype: pandas.DataFrame

**max\_projection**

2D max projection image. :rtype: allensdk.brain\_observatory.behavior.image\_api.Image

**metadata**

Dictionary of session-specific metadata. :rtype: dict

**motion\_correction**

A dataframe containing trace data used during motion correction computation :rtype: pandas.DataFrame

**ophys\_experiment\_id**

Unique identifier for this experimental session. :rtype: int

**ophys\_timestamps**

Timestamps associated with frames captured by the microscope :rtype: numpy.ndarray

**rewards**

A dataframe containing timestamps of delivered rewards. :rtype: pandas.DataFrame

**running\_data\_df**

Dataframe containing various signals used to compute running speed :rtype: pandas.DataFrame

**running\_speed**

**Running speed of mouse. NamedTuple with two fields**

**timestamps** [numpy.ndarray] Timestamps of running speed data samples

**values** [np.ndarray] Running speed of the experimental subject (in cm / s).

**Return type** *allensdk.brain\_observatory.running\_speed.RunningSpeed*

**segmentation\_mask\_image**

An image with pixel value 1 if that pixel was included in an ROI, and 0 otherwise :rtype: allensdk.brain\_observatory.behavior.image\_api.Image

**stimulus\_presentations**

Table whose rows are stimulus presentations (i.e. a given image, for a given duration, typically 250 ms) and whose columns are presentation characteristics. :rtype: pandas.DataFrame

**stimulus\_templates**

A dictionary containing the stimulus images presented during the session keys are data set names, and values are 3D numpy arrays. :rtype: dict

**stimulus\_timestamps**

Timestamps associated with stimulus presentations on the monitor (corrected for monitor delay). :rtype: numpy.ndarray

**task\_parameters**

A dictionary containing parameters used to define the task runtime behavior. :rtype: dict

**trials**

A dataframe containing behavioral trial start/stop times, and trial data :rtype: pandas.DataFrame

**allensdk.brain\_observatory.behavior.behavior\_project\_cache module**

```
class allensdk.brain_observatory.behavior.behavior_project_cache.BehaviorProjectCache (fetch_
    Op-
    tional[
    =
    None,
    fetch_1
    int
    =
    2,
    man-
    i-
    fest:
    Union,
    path-
    lib.Path,
    None]
    =
    None,
    ver-
    sion:
    Op-
    tional[
    =
    None,
    cache:
    bool
    =
    True)
```

Bases: *allensdk.api.cache.Cache*

**BEHAVIOR\_ANALYSIS\_LOG\_KEY** = 'behavior\_analysis\_log'

**BEHAVIOR\_SESSIONS\_KEY** = 'behavior\_sessions'

**MANIFEST\_CONFIG** = {'behavior\_analysis\_log': {'parent\_key': 'BASEDIR', 'spec': 'beha

**MANIFEST\_VERSION** = '0.0.1-alpha'

**OPHYS\_ANALYSIS\_LOG\_KEY** = 'ophys\_analysis\_log'

**OPHYS\_EXPERIMENTS\_KEY** = 'ophys\_experiments'

**OPHYS\_SESSIONS\_KEY** = 'ophys\_sessions'

**add\_manifest\_paths** (*self*, *manifest\_builder*)

Add cache-class specific paths to the manifest. In derived classes, should call super.

```
classmethod from_lims (manifest: Union[str, pathlib.Path, NoneType] = None,
    version: Union[str, NoneType] = None, cache:
    bool = True, fetch_tries: int = 2, lims_credentials:
    Union[allensdk.core.authentication.DbCredentials, NoneType] = None,
    mtrain_credentials: Union[allensdk.core.authentication.DbCredentials,
    NoneType] = None, host: Union[str, NoneType] = None, scheme:
    Union[str, NoneType] = None, asynchronous: bool = True) →
    'BehaviorProjectCache'
```

Construct a BehaviorProjectCache with a lims api. Use this method to create a BehaviorProjectCache

instance rather than calling BehaviorProjectCache directly.

### Parameters

**manifest** [str or Path] full path at which manifest json will be stored

**version** [str] version of manifest file. If this mismatches the version recorded in the file at manifest, an error will be raised.

**cache** [bool] Whether to write to the cache

**fetch\_tries** [int] Maximum number of times to attempt a download before giving up and raising an exception. Note that this is total tries, not retries

**lims\_credentials** [DbCredentials] Optional credentials to access LIMS database. If not set, will look for credentials in environment variables.

**mtrain\_credentials: DbCredentials** Optional credentials to access mtrain database. If not set, will look for credentials in environment variables.

**host** [str] Web host for the app\_engine. Currently unused. This argument is included for consistency with EcephysProjectCache.from\_lims.

**scheme** [str] URI scheme, such as “http”. Currently unused. This argument is included for consistency with EcephysProjectCache.from\_lims.

**asynchronous** [bool] Whether to fetch from web asynchronously. Currently unused.

### Returns

=====

**BehaviorProjectCache** BehaviorProjectCache instance with a LIMS fetch API

**get\_behavior\_session\_data** (*self*, *behavior\_session\_id*: int, *fixed*: bool = False)

Note – This method mocks the behavior of a cache. No files are actually downloaded for local access. Instead, it adds the session id to a csv log. If the “fixed” parameter is true, then the API will first check to ensure that the log is present in the record before pulling the data.

**get\_behavior\_session\_table** (*self*, *suppress*: Union[List[str], NoneType] = None) → pandas.core.frame.DataFrame

Return summary table of all behavior\_session\_ids in the database. :param suppress: optional list of columns to drop from the resulting

dataframe.

**Return type** pd.DataFrame

**get\_experiment\_table** (*self*, *suppress*: Union[List[str], NoneType] = None) → pandas.core.frame.DataFrame

Return summary table of all ophys\_experiment\_ids in the database. :param suppress: optional list of columns to drop from the resulting

dataframe.

**Return type** pd.DataFrame

**get\_session\_data** (*self*, *ophys\_experiment\_id*: int, *fixed*: bool = False)

Note – This method mocks the behavior of a cache. No files are actually downloaded for local access. Instead, it adds the session id to a csv log. If the “fixed” parameter is true, then the API will first check to ensure that the log is present in the record before pulling the data.

**get\_session\_table** (*self*, *suppress*: *Union[List[str], NoneType] = None*, *by*: *str = 'ophys\_session\_id'*) → *pandas.core.frame.DataFrame*  
 Return summary table of all ophys\_session\_ids in the database. :param suppress: optional list of columns to drop from the resulting

dataframe.

**Parameters by** (*str*) – (default="ophys\_session\_id"). Column to index on, either "ophys\_session\_id" or "ophys\_experiment\_id". If by="ophys\_experiment\_id", then each row will only have one experiment id, of type int (vs. an array of 1+more).

**Return type** *pd.DataFrame*

## allensdk.brain\_observatory.behavior.behavior\_project\_lims\_api module

**class** *allensdk.brain\_observatory.behavior.behavior\_project\_lims\_api.BehaviorProjectLimsApi*

Bases: *allensdk.brain\_observatory.behavior.internal.behavior\_project\_base.BehaviorProjectBase*

**classmethod default** (*lims\_credentials*: *Union[allensdk.core.authentication.DbCredentials, NoneType] = None*, *mtrain\_credentials*: *Union[allensdk.core.authentication.DbCredentials, NoneType] = None*, *app\_kwargs*: *Union[Dict[str, Any], NoneType] = None*) → 'BehaviorProjectLimsApi'

Construct a BehaviorProjectLimsApi instance with default postgres and app engines.

### Parameters

**lims\_credentials: Optional[DbCredentials]** Credentials to pass to the postgres connector to the lims database. If left unspecified, will check environment variables for the appropriate values.

**mtrain\_credentials: Optional[DbCredentials]** Credentials to pass to the postgres connector to the mtrain database. If left unspecified, will check environment variables for the appropriate values.

**app\_kwargs: Dict** Dict of arguments to pass to the app engine. Currently unused.

### Returns

**BehaviorProjectLimsApi**

**get\_behavior\_only\_session\_data** (*self*, *behavior\_session\_id*: *int*) → *allensdk.brain\_observatory.behavior.behavior\_data\_session.BehaviorDataSession*  
 Returns a BehaviorDataSession object that contains methods to analyze a single behavior session. :param behavior\_session\_id: id that corresponds to a behavior session :type behavior\_session\_id: int :rtype: BehaviorDataSession

**get\_behavior\_only\_session\_table** (*self*, *behavior\_session\_ids*: *Union[List[int], NoneType] = None*) → *pandas.core.frame.DataFrame*  
 Returns a *pd.DataFrame* table with all behavior session\_ids to the user with additional metadata.

Can't return age at time of session because there is no field for acquisition date for behavior sessions (only in the stimulus.pkl file) :rtype: *pd.DataFrame*

**get\_experiment\_table** (*self*, *ophys\_experiment\_ids*: *Union[List[int], NoneType] = None*) → *pandas.core.frame.DataFrame*  
 Return a *pd.DataFrame* table with all ophys\_experiment\_ids and relevant metadata. This is the most specific and most informative level to examine the data. Return columns:

ophys\_experiment\_id, ophys\_session\_id, behavior\_session\_id, container\_id, project\_code, container\_workflow\_state, experiment\_workflow\_state, session\_name, session\_type, equipment\_name, date\_of\_acquisition, isi\_experiment\_id, specimen\_id, sex, age\_in\_days, full\_genotype, reporter\_line, driver\_line, imaging\_depth, targeted\_structure, published\_at

**Parameters** `ophys_experiment_ids` – optional list of ophys\_experiment\_ids to include

**Return type** `pd.DataFrame`

**get\_natural\_movie\_template** (*self*, *number: int*) → `Iterable[bytes]`

Download a template for the natural scene stimulus. This is the actual image that was shown during the recording session. :param number: identifier for this movie (note that this is an int,

so to get the template for natural\_movie\_three should pass 3)

**Returns** iterable yielding a tiff file as bytes

**get\_natural\_scene\_template** (*self*, *number: int*) → `Iterable[bytes]`

Download a template for the natural movie stimulus. This is the actual movie that was shown during the recording session. :param number: identifier for this scene :type number: int :returns: An iterable yielding an npy file as bytes

**get\_session\_data** (*self*, *ophys\_session\_id: int*) → `allensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorO`

Returns a BehaviorOphysSession object that contains methods to analyze a single behavior+ophys session. :param ophys\_session\_id: id that corresponds to a behavior session :type ophys\_session\_id: int :rtype: BehaviorOphysSession

**get\_session\_table** (*self*, *ophys\_session\_ids: Union[List[int], NoneType] = None*) → `pan-  
das.core.frame.DataFrame`

Return a pd.DataFrame table with all ophys\_session\_ids and relevant metadata. Return columns: ophys\_session\_id, behavior\_session\_id,

ophys\_experiment\_id, project\_code, session\_name, session\_type, equipment\_name, date\_of\_acquisition, specimen\_id, full\_genotype, sex, age\_in\_days, reporter\_line, driver\_line

**Parameters** `ophys_session_ids` – optional list of ophys\_session\_ids to include

**Return type** `pd.DataFrame`

## **allensdk.brain\_observatory.behavior.criteria module**

Functions for calculating mtrain state transitions. If criteria are met, return true. Otherwise, return false.

`allensdk.brain_observatory.behavior.criteria.consistency_is_key` (*session\_summary*)

need some way to judge consistency of various parameters

- dprime
- num trials
- hit rate
- fa rate
- lick timing

`allensdk.brain_observatory.behavior.criteria.consistent_behavior_within_session` (*session\_summa*)

need some way to measure consistent performance within a session

- compare peak to overall dprime?

- variance in rolling window dprime?

`allensdk.brain_observatory.behavior.criteria.meets_engagement_criteria(session_summary)`

Returns true if engagement criteria were met for the past 3 days, else false. Args:

`session_summary` (pd.DataFrame): Pandas dataframe with daily values for 'dprime\_peak' and 'num\_engaged\_trials', ordered ascending by training day, for at least 3 days. If dataframe is not properly ordered, criterion may not be correctly calculated. This function does not sort the data to preserve prior behavior (sorting column was not required by mtrain function) The mtrain implementation created the required columns if they didn't exist, so a more informative error is raised here to assist end-users in debugging.

**Returns:** bool: True if criterion is met, False otherwise

`allensdk.brain_observatory.behavior.criteria.mostly_useful(trials)`

Returns True if fewer than half the trial time on the last day were aborted trials.

Args: `trials` (pd.DataFrame): Pandas dataframe with columns 'training\_day', 'trial\_type', and 'trial\_length'.

**Returns:** bool: True if criterion is met, False otherwise

`allensdk.brain_observatory.behavior.criteria.n_complete(threshold, count)`

For compatibility with original API. If count >= threshold, return True. Otherwise return False. Args:

`threshold` (numeric): Threshold for the count to meet. `count` (numeric): The count to compare to the threshold.

**Returns:** True if count >= threshold, otherwise False.

`allensdk.brain_observatory.behavior.criteria.no_response_bias(session_summary)`

the mouse meets this criterion if their last session exhibited a response bias between 10% and 90%

Args: `session_summary` (pd.DataFrame): Pandas dataframe with daily values for 'response\_bias', ordered ascending by training day, for at least 1 day. If dataframe is not properly ordered, criterion may not be correctly calculated. This function does not sort the data to preserve prior behavior (sorting column was not required by mtrain function). The mtrain implementation created the required columns if they didn't exist, so a more informative error is raised here to assist end-users in debugging.

**Returns:** bool: True if criterion is met, False otherwise

`allensdk.brain_observatory.behavior.criteria.summer_over(trials)`

Returns true if the maximum value of 'training\_day' in the trials dataframe is >= 40, else false.

`allensdk.brain_observatory.behavior.criteria.two_out_of_three_aint_bad(session_summary)`

Returns true if 2 of the last 3 days showed a peak d-prime above 2.

Args: `session_summary` (pd.DataFrame): Pandas dataframe with daily values for 'dprime\_peak', ordered ascending by training day, for at least the past 3 days. If dataframe is not properly ordered, criterion may not be correctly calculated. This function does not sort the data to preserve prior behavior (sorting column was not required by mtrain function). The mtrain implementation created the required columns if they didn't exist, so a more informative error is raised here to assist end-users in debugging.

**Returns:** bool: True if criterion is met, False otherwise

`allensdk.brain_observatory.behavior.criteria.whole_lotta_trials(session_summary)`

Mouse meets this criterion if the last session has more than 300 trials. Args:

`session_summary` (pd.DataFrame): Pandas dataframe with daily values for 'num\_contingent\_trials', ordered ascending by training day, for at least 1 day. If dataframe is not properly ordered, criterion may not be correctly calculated. This function does not sort the data to preserve prior behavior (sorting column was not required by `mtrain` function). The `mtrain` implementation created the required columns if they didn't exist, so a more informative error is raised here to assist end-users in debugging.

**Returns:** bool: True if criterion is met, False otherwise

`allensdk.brain_observatory.behavior.criteria.yesterday_was_good(session_summary)`

Returns true if the last day showed a peak d-prime above 2 Args:

`session_summary` (pd.DataFrame): Pandas dataframe with daily values for 'dprime\_peak', ordered ascending by training day, for at least 1 day. If dataframe is not properly ordered, criterion may not be correctly calculated. This function does not sort the data to preserve prior behavior (sorting column was not required by `mtrain` function). The `mtrain` implementation created the required columns if they didn't exist, so a more informative error is raised here to assist end-users in debugging.

**Returns:** bool: True if criterion is met, False otherwise

### `allensdk.brain_observatory.behavior.dprime` module

`allensdk.brain_observatory.behavior.dprime.get_catch_responses` (*correct\_reject=None*,  
*false\_alarm=None*,  
*aborted=None*)

`allensdk.brain_observatory.behavior.dprime.get_dprime` (*hit\_rate*, *fa\_rate*, *sliding\_window=100*)

calculates the d-prime for a given hit rate and false alarm rate [https://en.wikipedia.org/wiki/Sensitivity\\_index](https://en.wikipedia.org/wiki/Sensitivity_index)

Parameters ——— `hit_rate` : float

rate of hits in the True class

**fa\_rate** [float] rate of false alarms in the False class

**limits** [tuple, optional] limits on extreme values, which distort. default: (0.01,0.99)

**d\_prime**

`allensdk.brain_observatory.behavior.dprime.get_false_alarm_rate` (*correct\_reject=None*,  
*false\_alarm=None*,  
*aborted=None*,  
*sliding\_window=100*)

`allensdk.brain_observatory.behavior.dprime.get_go_responses` (*hit=None*,  
*miss=None*,  
*aborted=None*)

`allensdk.brain_observatory.behavior.dprime.get_hit_rate` (*hit=None*, *miss=None*,  
*aborted=None*, *sliding\_window=100*)

`allensdk.brain_observatory.behavior.dprime.get_rolling_dprime` (*rolling\_hit\_rate*,  
*rolling\_fa\_rate*,  
*sliding\_window=100*)



```
allensdk.brain_observatory.behavior.dprime.get_trial_count_corrected_false_alarm_rate(corrected,
                                                                                   false_alarm_rate,
                                                                                   hit_rate,
                                                                                   miss_rate,
                                                                                   aborted_rate,
                                                                                   sliding_window)

allensdk.brain_observatory.behavior.dprime.get_trial_count_corrected_hit_rate(hit=None,
                                                                              miss=None,
                                                                              aborted=None,
                                                                              sliding_window=100)

allensdk.brain_observatory.behavior.dprime.trial_number_limit(p, N)
```

### allensdk.brain\_observatory.behavior.eye\_tracking\_processing module

```
allensdk.brain_observatory.behavior.eye_tracking_processing.compute_circular_area(df_row:
                                                                                  pan-
das.core.series.Series
                                                                                  →
                                                                                  float)
```

Calculate the area of the pupil as a circle using the max of the height/width as radius.

Note: This calculation assumes that the pupil is a perfect circle and any eccentricity is a result of the angle at which the pupil is being viewed.

#### Parameters

**df\_row** [pd.Series] A row from an eye tracking dataframe containing only “pupil\_width” and “pupil\_height”.

#### Returns

**float** The circular area of the pupil in pixels<sup>2</sup>.

```
allensdk.brain_observatory.behavior.eye_tracking_processing.compute_elliptical_area(df_row:
                                                                                  pan-
das.core.series.Series
                                                                                  →
                                                                                  float)
```

Calculate the area of corneal reflection (cr) or eye ellipse fits using the ellipse formula.

#### Parameters

**df\_row** [pd.Series] A row from an eye tracking dataframe containing either: “cr\_width”, “cr\_height” or “eye\_width”, “eye\_height”

#### Returns

**float** The elliptical area of the eye or cr in pixels<sup>2</sup>

```
allensdk.brain_observatory.behavior.eye_tracking_processing.determine_likely_blinks (eye_areas: pandas.core.series.Series, pupil_areas: pandas.core.series.Series, outliers: pandas.core.series.Series, dilation_frames: int)
    =
    2)
    →
    pandas.core.series.Series
```

Determine eye tracking frames which contain likely blinks or outliers

#### Parameters

**eye\_areas** [pd.Series] A pandas series of eye areas.

**pupil\_areas** [pd.Series] A pandas series of pupil areas.

**outliers** [pd.Series] A pandas series containing bool values of outlier rows.

**dilation\_frames** [int, optional] Determines the number of additional adjacent frames to mark as 'likely\_blink', by default 2.

#### Returns

**pd.Series** A pandas series of bool values that has the same length as the number of eye tracking dataframe rows (frames).

```
allensdk.brain_observatory.behavior.eye_tracking_processing.determine_outliers (data_df: pandas.core.frame.DataFrame, z_threshold: float)
    →
    pandas.core.series.Series
```

Given a dataframe and some z-score threshold return a pandas boolean Series where each entry indicates whether a given row contains at least one outlier (where outliers are calculated along columns).

#### Parameters

**data\_df** [pd.DataFrame] A dataframe containing only columns where outlier detection is desired. (e.g. "cr\_area", "eye\_area", "pupil\_area")

**z\_threshold** [float] z-score values higher than the z\_threshold will be considered outliers.

#### Returns

**pd.Series** A pandas boolean Series whose length == len(data\_df.index). True denotes that a row in the data\_df contains at least one outlier.

```
allensdk.brain_observatory.behavior.eye_tracking_processing.load_eye_tracking_hdf(eye_tracking_file,
path-
lib.Path)
→
pan-
das.core.frame
```

Load a DeepLabCut hdf5 file containing eye tracking data into a dataframe.

Note: The eye tracking hdf5 file contains 3 separate dataframes. One for corneal reflection (cr), eye, and pupil ellipse fits. This function loads and returns this data as a single dataframe.

#### Parameters

**eye\_tracking\_file** [Path] Path to an hdf5 file produced by the DeepLabCut eye tracking pipeline. The hdf5 file will contain the following keys: “cr”, “eye”, “pupil”. Each key has an associated dataframe with the following columns: “center\_x”, “center\_y”, “height”, “width”, “phi”.

#### Returns

**pd.DataFrame** A dataframe containing combined corneal reflection (cr), eyelid (eye), and pupil data. Column names for each field will be renamed by prepending the field name. (e.g. center\_x -> eye\_center\_x)

```
allensdk.brain_observatory.behavior.eye_tracking_processing.process_eye_tracking_data(eye_data,
pan-
das.core.frame_
pan-
das.core.frame_
z_thres:
float
=
3.0,
di-
la-
tion_fr
int
=
2)
→
pan-
das.core.frame
```

Processes and refines raw eye tracking data by adding additional computed feature columns.

#### Parameters

**eye\_data** [pd.DataFrame] A ‘raw’ eye tracking dataframe produced by load\_eye\_tracking\_hdf()

**frame\_times** [pd.Series] A series of frame times acquired from a behavior + ophy session ‘sync file’.

**z\_threshold** [float] z-score values higher than the z\_threshold will be considered outliers, by default 3.0.

**dilation\_frames** [int, optional] Determines the number of additional adjacent frames to mark as ‘likely\_blink’, by default 2.

#### Returns

**pd.DataFrame** A refined eye tracking dataframe that contains additional information about frame times, eye areas, pupil areas, and frames with likely blinks/outliers.

**Raises**

**RuntimeError** If the number of sync file frame times does not match the number of eye tracking frames.

### **allensdk.brain\_observatory.behavior.image\_api module**

```
class allensdk.brain_observatory.behavior.image_api.Image
    Bases: tuple
    Describes a 2D Image
    data [np.ndarray] Image data points
    spacing [tuple] Spacing describes the physical size of each pixel
    unit [str] Physical unit of the spacing (currently constrained to be isotropic)
    data
        Alias for field number 0
    spacing
        Alias for field number 1
    unit
        Alias for field number 2
class allensdk.brain_observatory.behavior.image_api.ImageApi
    Bases: object
    static deserialize (img)
    static serialize (data, spacing, unit)
```

### **allensdk.brain\_observatory.behavior.metadata\_processing module**

```
allensdk.brain_observatory.behavior.metadata_processing.get_task_parameters (data)
```

### **allensdk.brain\_observatory.behavior.mtrain module**

```
class allensdk.brain_observatory.behavior.mtrain.ExtendedTrialSchema (only=None,
                                                                    ex-
                                                                    clude=(),
                                                                    many=False,
                                                                    con-
                                                                    text=None,
                                                                    load_only=(),
                                                                    dump_only=(),
                                                                    par-
                                                                    tial=False,
                                                                    un-
                                                                    known=None)

    Bases: marshmallow.schema.Schema
    This schema describes the edf core trial structure
```

**opts** = <marshmallow.schema.SchemaOpts object>

**class** allensdk.brain\_observatory.behavior.mtrain.**FriendlyDate** (*format=None, \*\*kwargs*)

Bases: marshmallow.fields.Date

**class** allensdk.brain\_observatory.behavior.mtrain.**FriendlyDateTime** (*format=None, \*\*kwargs*)

Bases: marshmallow.fields.DateTime

allensdk.brain\_observatory.behavior.mtrain.**annotate\_change\_detect** (*trials*)  
adds *change* and *detect* columns to dataframe

#### Parameters

**trials** [pandas DataFrame] dataframe of trials

**inplace** [bool, optional] modify *trials* in place. if False, returns a copy. default: True

See also:

**io.load\_trials**

allensdk.brain\_observatory.behavior.mtrain.**annotate\_trials** (*trials*)  
performs multiple annotations:

- **annotate\_change\_detect**
- **fix\_change\_time**
- **explode\_response\_window**

#### Parameters

**trials** [pandas DataFrame] dataframe of trials

**inplace** [bool, optional] modify *trials* in place. if False, returns a copy. default: True

See also:

**io.load\_trials**

allensdk.brain\_observatory.behavior.mtrain.**assign\_session\_id** (*trials*)

**adds a column with a unique ID for the session defined as** a combination of the mouse ID and startdatetime

#### Parameters

**trials** [pandas DataFrame] dataframe of trials

**inplace** [bool, optional] modify *trials* in place. if False, returns a copy. default: True

See also:

**io.load\_trials**

allensdk.brain\_observatory.behavior.mtrain.**explode\_response\_window** (*trials*)  
explodes the *response\_window* column in lower & upper columns

#### Parameters

**trials** [pandas DataFrame] dataframe of trials

**inplace** [bool, optional] modify *trials* in place. if False, returns a copy. default: True

See also:

`io.load_trials`

`allensdk.brain_observatory.behavior.mtrain.fix_change_time(trials)`  
forces *None* values in the *change\_time* column to numpy NaN

#### Parameters

**trials** [pandas DataFrame] dataframe of trials

**inplace** [bool, optional] modify *trials* in place. if False, returns a copy. default: True

See also:

`io.load_trials`

### `allensdk.brain_observatory.behavior.rewards_processing` module

`allensdk.brain_observatory.behavior.rewards_processing.get_rewards(data, stimulus_rebase_function)`

### `allensdk.brain_observatory.behavior.running_processing` module

`allensdk.brain_observatory.behavior.running_processing.calc_deriv(x, time)`  
`allensdk.brain_observatory.behavior.running_processing.deg_to_dist(speed_deg_per_s)`  
takes speed in degrees per second converts to radians multiplies by radius (in cm) to get linear speed in cm/s  
`allensdk.brain_observatory.behavior.running_processing.get_running_df(data, time)`

### `allensdk.brain_observatory.behavior.schemas` module

**class** `allensdk.brain_observatory.behavior.schemas.OphysBehaviorMetaDataSchema` (*only=None, exclude=(), many=False, context=None, load\_only=(), dump\_only=(), partial=False, unknown=None*)

Bases: `allensdk.brain_observatory.behavior.schemas.RaisingSchema`

base schema for all timeseries

**neurodata\_type** = 'OphysBehaviorMetaData'

**opts** = <marshmallow.schema.SchemaOpts object>

```
class allensdk.brain_observatory.behavior.schemas.OphysBehaviorTaskParametersSchema (only=None,
ex-
clude=(),
many=False,
con-
text=None,
load_only=(),
dump_only=(),
par-
tial=False,
un-
known=None)
```

Bases: `allensdk.brain_observatory.behavior.schemas.RaisingSchema`

base schema for all timeseries

```
neurodata_type = 'OphysBehaviorTaskParameters'
```

```
opts = <marshmallow.schema.SchemaOpts object>
```

```
class allensdk.brain_observatory.behavior.schemas.RaisingSchema (only=None,
exclude=(),
many=False,
context=None,
load_only=(),
dump_only=(),
par-
tial=False, un-
known=None)
```

Bases: `marshmallow.schema.Schema`

```
class Meta
```

Bases: `object`

```
unknown = 'raise'
```

```
opts = <marshmallow.schema.SchemaOpts object>
```

## allensdk.brain\_observatory.behavior.session\_metrics module

```
allensdk.brain_observatory.behavior.session_metrics.num_contingent_trials (session_trials)
```

Returns the number of “go” and “catch” trials in a training session dataframe. Args:

`session_trials` (`pandas.DataFrame`): a `pandas.DataFrame` describing behavior training trials, with the string column “trial\_type” describing the type of trial.

Returns (int): Number of “go” and “catch” trials

```
allensdk.brain_observatory.behavior.session_metrics.response_bias (trials, de-
tect_col,
trial_types=('go',
'catch'))
```

Calculate the response bias for a subset of trial types from a behavioral training dataframe. Args:

**trials** (`pandas.DataFrame`): **Dataframe containing trial-level information** from a behavioral training session. Required columns: “trial\_type”, `detect_col`.

**detect\_col** (str): **Name of column containing boolean** or numeric codings (0/1) for whether or not the mouse had a response.

**trial\_types** (iterable<str>): Iterable containing string trial types to check for the response bias. Trials of types not included in this iterable will be ignored. Default=(“go”, “catch”)

**Return:** The response bias (or average value of the *detect\_col*) for trials in *trial\_types*.

### allensdk.brain\_observatory.behavior.stimulus\_processing module

```
allensdk.brain_observatory.behavior.stimulus_processing.convert_filepath_caseinsensitive (fil
allensdk.brain_observatory.behavior.stimulus_processing.get_images_dict (pkl)
allensdk.brain_observatory.behavior.stimulus_processing.get_stimulus_metadata (pkl)
allensdk.brain_observatory.behavior.stimulus_processing.get_stimulus_presentations (data,
                                                                    stim-
                                                                    u-
                                                                    lus_timesta
allensdk.brain_observatory.behavior.stimulus_processing.get_stimulus_templates (pkl)
allensdk.brain_observatory.behavior.stimulus_processing.get_visual_stimuli_df (data,
                                                                    time)
allensdk.brain_observatory.behavior.stimulus_processing.load_pickle (pstream)
allensdk.brain_observatory.behavior.stimulus_processing.unpack_change_log (change)
```

### allensdk.brain\_observatory.behavior.trial\_masks module

allensdk.brain\_observatory.behavior.trial\_masks.contingent\_trials (trials)  
GO & CATCH trials only

#### Parameters

**trials** [pandas DataFrame] dataframe of trials

#### Returns

**mask** [pandas Series of booleans, indexed to trials DataFrame]

allensdk.brain\_observatory.behavior.trial\_masks.reward\_rate (trials, thresh=2.0)  
masks trials where the reward rate (per minute) is below some threshold.

This de facto omits trials in which the animal was not licking for extended periods or periods when they were licking indiscriminantly.

#### Parameters

**trials** [pandas DataFrame] dataframe of trials

**thresh** [float, optional] threshold under which trials will not be included, default: 2.0

#### Returns

**mask** [pandas Series of booleans, indexed to trials DataFrame]

allensdk.brain\_observatory.behavior.trial\_masks.trial\_types (trials, trial\_types)  
only include trials of certain trial types

#### Parameters

**trials** [pandas DataFrame] dataframe of trials



**trial\_types** [list or other iterator]

#### Returns

**mask** [pandas Series of booleans, indexed to trials DataFrame]

### allensdk.brain\_observatory.behavior.trials\_processing module

allensdk.brain\_observatory.behavior.trials\_processing.**calculate\_reward\_rate** (*response\_latency=None, start-time=None, win-dow=0.75, trial\_window=25, initial\_trials=10*)

allensdk.brain\_observatory.behavior.trials\_processing.**categorize\_one\_trial** (*tr*)

allensdk.brain\_observatory.behavior.trials\_processing.**colormap** (*trial\_type, response\_type*)

allensdk.brain\_observatory.behavior.trials\_processing.**create\_extended\_trials** (*trials=None, meta-data=None, time=None, licks=None*)

allensdk.brain\_observatory.behavior.trials\_processing.**data\_to\_licks** (*data, time*)

allensdk.brain\_observatory.behavior.trials\_processing.**data\_to\_metadata** (*data, time*)

allensdk.brain\_observatory.behavior.trials\_processing.**find\_licks** (*reward\_times, licks, win-dow=3.5*)

allensdk.brain\_observatory.behavior.trials\_processing.**get\_change\_time\_frame\_response\_latency**

allensdk.brain\_observatory.behavior.trials\_processing.**get\_even\_sampling** (*data*)  
Get status of even\_sampling

#### Parameters

**data:** Mapping foraging2 experiment output data

#### Returns

**bool:** True if even\_sampling is enabled

allensdk.brain\_observatory.behavior.trials\_processing.**get\_extended\_trials** (*data, time=None*)

allensdk.brain\_observatory.behavior.trials\_processing.**get\_image\_info\_from\_trial** (*trial\_log, ti*)

allensdk.brain\_observatory.behavior.trials\_processing.**get\_mouse\_id** (*exp\_data*)

allensdk.brain\_observatory.behavior.trials\_processing.**get\_ori\_info\_from\_trial** (*trial\_log, ti*)

allensdk.brain\_observatory.behavior.trials\_processing.**get\_params** (*exp\_data*)

```
allensdk.brain_observatory.behavior.trials_processing.get_response_latency(change_event,  
                                                                           trial)  
allensdk.brain_observatory.behavior.trials_processing.get_response_type(trials)  
allensdk.brain_observatory.behavior.trials_processing.get_stimulus_attr_changes(stim_dict,  
                                                                                change_frame,  
                                                                                first_frame,  
                                                                                last_frame)
```

## Notes

- assumes only two stimuli are ever shown
- converts attr\_names to lowercase
- gets the net attr changes from the start of a trial to the end of a trial

```
allensdk.brain_observatory.behavior.trials_processing.get_time(exp_data)  
allensdk.brain_observatory.behavior.trials_processing.get_trial_image_names(trial,  
                                                                           stim-  
                                                                           uli)  
allensdk.brain_observatory.behavior.trials_processing.get_trial_lick_times(lick_times,  
                                                                           start_time,  
                                                                           stop_time)
```

extract lick times in time range

```
allensdk.brain_observatory.behavior.trials_processing.get_trial_reward_time(rebased_reward_time,  
                                                                           start_time,  
                                                                           stop_time)
```

extract reward times in time range

```
allensdk.brain_observatory.behavior.trials_processing.get_trial_timing(event_dict,  
                                                                           stim-  
                                                                           u-  
                                                                           lus_presentations_df,  
                                                                           licks,  
                                                                           go,  
                                                                           catch,  
                                                                           auto_rewarded,  
                                                                           hit,  
                                                                           false_alarm)
```

extract trial timing data

content of trial log depends on trial type depends on trial type and response type go, catch, auto\_rewarded, hit, false\_alarm must be passed as booleans to disambiguate trial and response type

on go or auto\_rewarded trials, extract the stimulus\_changed time on catch trials, extract the sham\_change time

on hit trials, extract the response time from the hit entry in event\_dict on false\_alarm trials, extract the response time from the false\_alarm entry in event\_dict

```
allensdk.brain_observatory.behavior.trials_processing.get_trials(data,  
                                                                licks_df,  
                                                                rewards_df,  
                                                                stimu-  
                                                                lus_presentations_df,  
                                                                rebase)
```

```
allensdk.brain_observatory.behavior.trials_processing.get_trials_v0(data,
                                                                    time)
allensdk.brain_observatory.behavior.trials_processing.local_time(iso_timestamp,
                                                                    time-
                                                                    zone=None)
allensdk.brain_observatory.behavior.trials_processing.resolve_initial_image(stimuli,
                                                                              start_frame)
```

Attempts to resolve the initial image for a given start\_frame for a trial

#### Parameters

**stimuli: Mapping** foraging2 shape stimuli mapping

**start\_frame: int** start frame of the trial

#### Returns

**initial\_image\_category\_name: str** stimulus category of initial image

**initial\_image\_group: str** group name of the initial image

**initial\_image\_name: str** name of the initial image

```
allensdk.brain_observatory.behavior.trials_processing.trial_data_from_log(trial)
```

Infer trial logic from trial log. Returns a dictionary.

- reward volume: volume of water delivered on the trial, in mL

Each of the following values is boolean:

Trial category values are mutually exclusive \* go: trial was a go trial (trial with a stimulus change) \* catch: trial was a catch trial (trial with a sham stimulus change)

stimulus\_change/sham\_change are mutually exclusive \* stimulus\_change: did the stimulus change (True on 'go' trials) \* sham\_change: stimulus did not change, but response was evaluated (True on 'catch' trials)

Each trial can be one (and only one) of the following: \* hit (stimulus changed, animal responded in response window) \* miss (stimulus changed, animal did not respond in response window) \* false\_alarm (stimulus did not change, animal responded in response window) \* correct\_reject (stimulus did not change, animal did not respond in response window) \* aborted (animal responded before change time) \* auto\_rewarded (reward was automatically delivered following the change. This will bias the animals choice and should not be categorized as hit/miss)

```
allensdk.brain_observatory.behavior.trials_processing.validate_trial_condition_exclusivity
```

ensure that only one of N possible mutually exclusive trial conditions is True

## allensdk.brain\_observatory.behavior.validation module

**exception** allensdk.brain\_observatory.behavior.validation.ValidationError

Bases: AssertionError

```
allensdk.brain_observatory.behavior.validation.get_raw_ophys_file_shape(raw_filepath)
```

```
allensdk.brain_observatory.behavior.validation.validate_last_trial_ends_adjacent_to_flash(ophys_experiment_id,
                                                                                          api=None)
```

```
allensdk.brain_observatory.behavior.validation.validate_ophys_dff_length(ophys_experiment_id,
                                                                            api=None)
```

```
allensdk.brain_observatory.behavior.validation.validate_ophys_timestamps(ophys_experiment_id,  
                                                                           api=None)
```

## Module contents

**allensdk.brain\_observatory.ecephys package**

### Subpackages

**allensdk.brain\_observatory.ecephys.align\_timestamps package**

### Submodules

**allensdk.brain\_observatory.ecephys.align\_timestamps.barcode module**

```
allensdk.brain_observatory.ecephys.align_timestamps.barcode.extract_barcodes_from_times(on_  
                                                                                          off_  
                                                                                          in_  
                                                                                          ter_  
                                                                                          bar  
                                                                                          bar  
                                                                                          cod  
                                                                                          nbits)
```

Read barcodes from timestamped rising and falling edges.

#### Parameters

- on\_times** [numpy.ndarray] Timestamps of rising edges on the barcode line
- off\_times** [numpy.ndarray] Timestamps of falling edges on the barcode line
- inter\_barcode\_interval** [numeric, optional] Minimum duration of time between barcodes.
- bar\_duration** [numeric, optional] A value slightly shorter than the expected duration of each bar
- barcode\_duration\_ceiling** [numeric, optional] The maximum duration of a single barcode
- nbits** [int, optional] The bit-depth of each barcode

#### Returns

- barcode\_start\_times** [list of numeric] For each detected barcode, the time at which that barcode started
- barcodes** [list of int] For each detected barcode, the value of that barcode as an integer.

## Notes

ignores first code in prod (ok, but not intended) ignores first on pulse (intended - this is needed to identify that a barcode is starting)

```
allensdk.brain_observatory.ecephys.align_timestamps.barcode.find_matching_index(master_barcode  
                                                                                  probe_barcode  
                                                                                  align_  
                                                                                  ment_type='sta)
```

Given a set of barcodes for the master clock and the probe clock, find the indices of a matching set, either

starting from the beginning or the end of the list.

#### Parameters

**master\_barcodes** [np.ndarray] barcode values on the master line. One per barcode

**probe\_barcodes** [np.ndarray] barcode values on the probe line. One per barcode

**alignment\_type** [string] 'start' or 'end'

#### Returns

**master\_barcode\_index** [int] matching index for master barcodes (None if not found)

**probe\_barcode\_index** [int] matching index for probe barcodes (None if not found)

`allensdk.brain_observatory.ecephys.align_timestamps.barcode.get_probe_time_offset` (*master\_time*

*mas-*  
*ter\_barcode*  
*probe\_time*  
*probe\_barcode*  
*acq\_start\_in*  
*lo-*  
*cal\_probe\_r*

Time offset between master clock and recording probes. For converting probe time to master clock.

#### Parameters

**master\_times** [np.ndarray] start times of barcodes (according to the master clock) on the master line. One per barcode.

**master\_barcodes** [np.ndarray] barcode values on the master line. One per barcode

**probe\_times** [np.ndarray] start times (according to the probe clock) of barcodes on the probe line. One per barcode

**probe\_barcodes** [np.ndarray] barcode values on the probe\_line. One per barcode

**acq\_start\_index** [int] sample index of probe acquisition start time

**local\_probe\_rate** [float] the probe's apparent sampling rate

#### Returns

**total\_time\_shift** [float] Time at which the probe started acquisition, assessed on the master clock. If < 0, the probe started earlier than the master line.

**probe\_rate** [float] The probe's sampling rate, assessed on the master clock

**master\_endpoints** [iterable] Defines the start and end times of the sync interval on the master clock

`allensdk.brain_observatory.ecephys.align_timestamps.barcode.linear_transform_from_interval`

Find a scale and translation which aligns two 1d segments

#### Parameters

**master** [iterable] Pair of floats defining the master interval. Order is [start, end].

**probe** [iterable] Pair of floats defining the probe interval. Order is [start, end].

#### Returns

**scale** [float] Scale factor. If > 1.0, the probe clock is running fast compared to the master clock. If < 1.0, the probe clock is running slow.

**translation** [float] If > 0, the probe clock started before the master clock. If > 0, after.

## Notes

**solves** (master + translation) \* scale = probe

for scale and translation

`allensdk.brain_observatory.ecephys.align_timestamps.barcode.match_barcodes` (*master\_times*,  
*mas-*  
*ter\_barcodes*,  
*probe\_times*,  
*probe\_barcodes*)

Given sequences of barcode values and (local) times on a probe line and a master line, find the time points on each clock corresponding to the first and last shared barcode.

If there's only one probe barcode, only the first matching timepoint is returned.

## Parameters

**master\_times** [np.ndarray] start times of barcodes (according to the master clock) on the master line. One per barcode.

**master\_barcodes** [np.ndarray] barcode values on the master line. One per barcode

**probe\_times** [np.ndarray] start times (according to the probe clock) of barcodes on the probe line. One per barcode

**probe\_barcodes** [np.ndarray] barcode values on the probe\_line. One per barcode

## Returns

**probe\_interval** [np.ndarray] Start and end times of the matched interval according to the probe\_clock.

**master\_interval** [np.ndarray] Start and end times of the matched interval according to the master clock

## `allensdk.brain_observatory.ecephys.align_timestamps.barcode_sync_dataset` module

**class** `allensdk.brain_observatory.ecephys.align_timestamps.barcode_sync_dataset.BarcodeSyncDataset`

Bases: `allensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset.EcephysSyncDataset`

### **barcode\_line**

Obtain the index of the barcode line for this dataset.

**extract\_barcodes** (*self*, **\*\*barcode\_kwargs**)

Read barcodes and their times from this dataset's barcode line.

## Parameters

**\*\*barcode\_kwargs** : Will be passed to `.barcode.extract_barcodes_from_times`

## Returns

**times** [np.ndarray] The start times of each detected barcode.

**codes** [np.ndarray] The values of each detected barcode

**get\_barcode\_table** (*self*, *\*\*barcode\_kwargs*)

A convenience method for getting barcode times and codes in a dictionary.

### Notes

This method is deprecated!

## allensdk.brain\_observatory.ecephys.align\_timestamps.channel\_states module

allensdk.brain\_observatory.ecephys.align\_timestamps.channel\_states.**extract\_barcodes\_from\_st**

Obtain barcodes from timestamped rising/falling edges.

### Parameters

**channel\_states** [numpy.ndarray] Rising and falling edges, denoted 1 and -1

**timestamps** [numpy.ndarray] Sample index of each event.

**sampling\_rate** [numeric] Samples / second

**\*\*barcode\_kwargs** : Additional parameters describing the barcodes.

allensdk.brain\_observatory.ecephys.align\_timestamps.channel\_states.**extract\_splits\_from\_stat**

Obtain barcodes from timestamped rising/falling edges.

### Parameters

**channel\_states** [numpy.ndarray] Rising and falling edges, denoted 1 and -1

**timestamps** [numpy.ndarray] Sample index of each event.

**sampling\_rate** [numeric] Samples / second

**\*\*barcode\_kwargs** : Additional parameters describing the barcodes.

## allensdk.brain\_observatory.ecephys.align\_timestamps.probe\_synchronizer module

**class** allensdk.brain\_observatory.ecephys.align\_timestamps.probe\_synchronizer.**ProbeSynchron**

Bases: object

```
classmethod compute (master_barcode_times, master_barcodes, probe_barcode_times,  
                     probe_barcodes, min_time, max_time, probe_start_index, lo-  
                     cal_probe_sampling_rate)
```

Compute a transform from probe samples to master times by aligning barcodes.

#### Parameters

**master\_barcode\_times** [np.ndarray] start times of barcodes (according to the master clock) on the master line. One per barcode.

**master\_barcodes** [np.ndarray] barcode values on the master line. One per barcode

**probe\_barcode\_times** [np.ndarray] start times (according to the probe clock) of barcodes on the probe line. One per barcode

**probe\_barcodes** [np.ndarray] barcode values on the probe\_line. One per barcode

**min\_time** [Float] time (in seconds) of first barcode to align

**max\_time** [Float] time (in seconds) of last barcode to align

**probe\_start\_index** [int] sample index of probe acquisition start time

**local\_probe\_sampling\_rate** [float] the probe's apparent sampling rate

#### Returns

**ProbeSynchronizer** : When called, applies the transform computed here to samples on the probe clock.

**sampling\_rate\_scale**

The ratio of the probe's sampling rate assessed on the global clock to the probe's locally assessed sampling rate.

## Module contents

**allensdk.brain\_observatory.ecephys.copy\_utility package**

## Module contents

**allensdk.brain\_observatory.ecephys.current\_source\_density package**

## Module contents

**allensdk.brain\_observatory.ecephys.ecephys\_project\_api package**

## Subpackages

**allensdk.brain\_observatory.ecephys.ecephys\_project\_api.warehouse\_patches package**

## Module contents

**allensdk.brain\_observatory.ecephys.ecephys\_project\_api.warehouse\_patches.load\_structure\_as**



```
allensdk.brain_observatory.ecephys.ecephys_project_api.warehouse_patches.replace_bad_struct
```

```
allensdk.brain_observatory.ecephys.ecephys_project_api.warehouse_patches.structure_assignme
```

## Submodules

### allensdk.brain\_observatory.ecephys.ecephys\_project\_api.ecephys\_project\_api module

```
class allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysPro
    Bases: object
```

```
    get_channels (self, channel_ids: Union[~ArrayLike, NoneType] = None, probe_ids:
                  Union[~ArrayLike, NoneType] = None, session_ids: Union[~ArrayLike, None-
                  Type] = None, published_at: Union[str, NoneType] = None)
```

```
    get_isi_experiments (self, *args, **kwargs)
```

```
    get_natural_movie_template (self, number) → Iterable
```

```
    get_natural_scene_template (self, number) → Iterable
```

```
    get_probe_lfp_data (self, probe_id: int) → Iterable
```

```
    get_probes (self, probe_ids: Union[~ArrayLike, NoneType] = None, session_ids: Union[~ArrayLike,
                  NoneType] = None, published_at: Union[str, NoneType] = None)
```

```
    get_session_data (self, session_id: int) → Iterable
```

```
    get_sessions (self, session_ids: Union[~ArrayLike, NoneType] = None, published_at: Union[str,
                  NoneType] = None)
```

```
    get_unit_analysis_metrics (self, unit_ids: Union[~ArrayLike, NoneType] = None, ece-
                              phys_session_ids: Union[~ArrayLike, NoneType] = None, ses-
                              sion_types: Union[~ArrayLike, NoneType] = None) → pan-
                              das.core.frame.DataFrame
```

```
    get_units (self, unit_ids: Union[~ArrayLike, NoneType] = None, channel_ids: Union[~ArrayLike,
                  NoneType] = None, probe_ids: Union[~ArrayLike, NoneType] = None, session_ids:
                  Union[~ArrayLike, NoneType] = None, published_at: Union[str, NoneType] = None)
```

### allensdk.brain\_observatory.ecephys.ecephys\_project\_api.ecephys\_project\_fixed\_api module

```
class allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_fixed_api.EcephysProjectFixedApi
    Bases: allensdk.brain_observatory.ecephys.ecephys_project_api.EcephysProjectApi
```

```
    get_channels (self, *args, **kwargs)
```

```
    get_isi_experiments (self, *args, **kwargs)
```

```
    get_natural_movie_template (self, number, *args, **kwargs)
```

```
    get_natural_scene_template (self, number, *args, **kwargs)
```

```
    get_probe_lfp_data (self, probe_id, *args, **kwargs)
```

```
get_probes (self, *args, **kwargs)
get_session_data (self, session_id, *args, **kwargs)
get_sessions (self, *args, **kwargs)
get_targeted_regions (self, *args, **kwargs)
get_units (self, *args, **kwargs)

exception allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_fixed_api
Bases: ValueError
```

## **allensdk.brain\_observatory.ecephys.ecephys\_project\_api.ecephys\_project\_lims\_api module**

```
class allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_lims_api.EcephysProjectLimsApi

Bases: allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi

STIMULUS_TEMPLATE_NAMESPACE = 'brain_observatory_1.1'

classmethod default (lims_credentials: Union[allensdk.core.authentication.DbCredentials,
NoneType] = None, app_kwargs=None, asynchronous=True)
Construct a “straightforward” lims api that can fetch data from lims2.
```

### **Parameters**

**lims\_credentials** [DbCredentials] Credentials and configuration for postgres queries against the LIMS database. If left unspecified will attempt to provide credentials from environment variables.

**app\_kwargs** [dict] High-level configuration for http requests. See `allensdk.brain_observatory.ecephys.ecephys_project_api.http_engine.HttpEngine` and `AsyncHttpEngine` for details.

**asynchronous** [bool] If true, (http) queries will be made asynchronously.

### **Returns**

#### **EcephysProjectLimsApi**

```
get_channels (self, channel_ids: Union[~ArrayLike, NoneType] = None, probe_ids:
Union[~ArrayLike, NoneType] = None, session_ids: Union[~ArrayLike,
NoneType] = None, published_at: Union[str, NoneType] = None) → pandas.core.frame.DataFrame
Download a table of ecephys channel records.
```

### **Parameters**

**channel\_ids** : A collection of integer identifiers for ecephys channels. If provided, results will be filtered to these channels.

**probe\_ids** : A collection of integer identifiers for ecephys probes. If provided, results will be filtered to channels on these probes.

**session\_ids** : A collection of integer identifiers for ecephys sessions. If provided, results will be filtered to channels recorded from during these sessions.

**published\_at** : A date (rendered as “YYYY-MM-DD”). If provided, only channels recorded from during sessions published before this date will be returned.

### **Returns**

a **pd.DataFrame** whose rows are ecephys channels.

**get\_natural\_movie\_template** (*self*, *number: int*) → Iterable[bytes]

Download a template for the natural movie stimulus. This is the actual movie that was shown during the recording session.

#### Parameters

**number** : identifier for this movie (note that this is an integer, so to get the template for natural\_movie\_three you should pass in 3)

#### Returns

An iterable yielding an **npz** file as bytes

**get\_natural\_scene\_template** (*self*, *number: int*) → Iterable[bytes]

Download a template for the natural scene stimulus. This is the actual image that was shown during the recording session.

#### Parameters

**number** : identifier for this scene

#### Returns

An iterable yielding a **tiff** file as bytes.

**get\_probe\_lfp\_data** (*self*, *probe\_id: int*) → Iterable[bytes]

Download an NWB file containing detailed data for the local field potential recorded from an ecephys probe.

#### Parameters

**probe\_id** : Download an NWB file for this probe's LFP

#### Returns

An iterable yielding an **NWB** file as bytes.

**get\_probes** (*self*, *probe\_ids: Union[~ArrayLike, NoneType] = None*, *session\_ids: Union[~ArrayLike, NoneType] = None*, *published\_at: Union[str, NoneType] = None*) → pandas.core.frame.DataFrame

Download a table of ecephys probe records.

#### Parameters

**probe\_ids** : A collection of integer identifiers for ecephys probes. If provided, results will be filtered to these probes.

**session\_ids** : A collection of integer identifiers for ecephys sessions. If provided, results will be filtered to probes recorded from during these sessions.

**published\_at** : A date (rendered as “YYYY-MM-DD”). If provided, only probes recorded from during sessions published before this date will be returned.

#### Returns

a **pd.DataFrame** whose rows are ecephys probes.

**get\_session\_data** (*self*, *session\_id: int*) → Iterable[bytes]

Download an NWB file containing detailed data for an ecephys session.

#### Parameters

**session\_id** : Download an NWB file for this session

#### Returns

**An iterable yielding an NWB file as bytes.**

**get\_sessions** (*self*, *session\_ids*: Union[~ArrayLike, NoneType] = None, *published\_at*: Union[str, NoneType] = None) → pandas.core.frame.DataFrame

Download a table of ecephys session records.

#### Parameters

**session\_ids** : A collection of integer identifiers for ecephys sessions. If provided, results will be filtered to these sessions.

**published\_at** : A date (rendered as “YYYY-MM-DD”). If provided, only sessions published before this date will be returned.

#### Returns

**a pd.DataFrame whose rows are ecephys sessions.**

**get\_unit\_analysis\_metrics** (*self*, *unit\_ids*: Union[~ArrayLike, NoneType] = None, *ecephys\_session\_ids*: Union[~ArrayLike, NoneType] = None, *session\_types*: Union[~ArrayLike, NoneType] = None) → pandas.core.frame.DataFrame

Fetch analysis metrics (stimulus set-specific characterizations of unit response patterns) for ecephys units. Note that the metrics returned depend on the stimuli that were presented during recording ( and thus on the session\_type)

**unit\_ids** : integer identifiers for a set of ecephys units. If provided, the response will only include metrics calculated for these units

**ecephys\_session\_ids** : integer identifiers for a set of ecephys sessions. If provided, the response will only include metrics calculated for units identified during these sessions

**session\_types** : string names identifying ecephys session types (e.g. “brain\_observatory\_1.1” or “functional\_connectivity”)

#### Returns

**a pandas dataframe indexed by ecephys unit id whose columns are metrics.**

**get\_units** (*self*, *unit\_ids*: Union[~ArrayLike, NoneType] = None, *channel\_ids*: Union[~ArrayLike, NoneType] = None, *probe\_ids*: Union[~ArrayLike, NoneType] = None, *session\_ids*: Union[~ArrayLike, NoneType] = None, *published\_at*: Union[str, NoneType] = None) → pandas.core.frame.DataFrame

Download a table of records describing sorted ecephys units.

#### Parameters

**unit\_ids** : A collection of integer identifiers for sorted ecephys units. If provided, only return records describing these units.

**channel\_ids** : A collection of integer identifiers for ecephys channels. If provided, results will be filtered to units recorded from these channels.

**probe\_ids** : A collection of integer identifiers for ecephys probes. If provided, results will be filtered to units recorded from these probes.

**session\_ids** : A collection of integer identifiers for ecephys sessions. If provided, results will be filtered to units recorded during these sessions.

**published\_at** : A date (rendered as “YYYY-MM-DD”). If provided, only units recorded during sessions published before this date will be returned.

**Returns**

a **pd.DataFrame** whose rows are ecephys channels.

**class** allensdk.brain\_observatory.ecephys.ecephys\_project\_api.ecephys\_project\_lims\_api.Split

Bases: tuple

**published\_at**

Alias for field number 0

**published\_at\_not\_null**

Alias for field number 1

**allensdk.brain\_observatory.ecephys.ecephys\_project\_api.ecephys\_project\_warehouse\_api** module

**class** allensdk.brain\_observatory.ecephys.ecephys\_project\_api.ecephys\_project\_warehouse\_api

Bases: *allensdk.brain\_observatory.ecephys.ecephys\_project\_api.ecephys\_project\_api.EcephysProjectApi*

**classmethod default** (asynchronous=True, \*\*kwargs)

**get\_channels** (self, channel\_ids=None, probe\_ids=None)

**get\_natural\_movie\_template** (self, number)

**get\_natural\_scene\_template** (self, number)

**get\_probe\_lfp\_data** (self, probe\_id)

**get\_probes** (self, probe\_ids=None, session\_ids=None)

**get\_session\_data** (self, session\_id, \*\*kwargs)

**get\_sessions** (self, session\_ids=None, has\_eye\_tracking=None, stimulus\_names=None)

**get\_unit\_analysis\_metrics** (self, unit\_ids=None, ecephys\_session\_ids=None, session\_types=None)

Download analysis metrics - precalculated descriptions of unitwise responses to visual stimulation.

**Parameters**

**unit\_ids** [array-like of int, optional] Unique identifiers for ecephys units. If supplied, only download metrics for these units.

**ecephys\_session\_ids** [array-like of int, optional] Unique identifiers for ecephys sessions. If supplied, only download metrics for units collected during these sessions.

**session\_types** [array-like of str, optional] Names of session types. e.g. “brain\_observatory\_1.1” or “functional\_connectivity”. If supplied, only download metrics for units collected during sessions of these types

**Returns**

**pd.DataFrame** : A table of analysis metrics, indexed by unit\_id.

**get\_units** (self, unit\_ids=None, channel\_ids=None, probe\_ids=None, session\_ids=None, \*a, \*\*k)

**movie\_re** = re.compile('.\*natural\_movie\_(?P<num>\\d+).np\\y')

**scene\_re** = re.compile('.\*/(?P<num>\\d+).tiff')

**stimulus\_templates**

**allensdk.brain\_observatory.ecephys.ecephys\_project\_api.http\_engine module**

**class** allensdk.brain\_observatory.ecephys.ecephys\_project\_api.http\_engine.**AsyncHttpEngine** (sc

Bases: *allensdk.brain\_observatory.ecephys.ecephys\_project\_api.http\_engine.HttpEngine*

**stream** (*self*, *route: str*) → Callable[[Callable[[AsyncIterator[bytes]], Awaitable[NoneType]]], Awaitable[NoneType]]

**Returns a coroutine which**

- makes an http request
- exposes internally an asynchronous iterator over the response
- takes a callback parameter, which should consume the iterator.

**Parameters**

**route** : the http route (under this object's host) to request against.

**Notes**

To use this method, you will need an appropriate consumer. For instance, If you want to write the streamed data to a local file, you can use `write_bytes_from_coroutine`.

**Examples**

```
>>> engine = AsyncHttpEngine("http", "examplehost")
>>> stream_coro = engine.stream("example/route")
>>> write_bytes_from_coroutine("example/file/path.txt", stream_coro)
```

```

class allensdk.brain_observatory.ecephys.ecephys_project_api.http_engine.HttpEngine (scheme:
                                                                    str,
                                                                    host:
                                                                    str,
                                                                    time-
                                                                    out:
                                                                    float
                                                                    =
                                                                    600,
                                                                    chunk-
                                                                    size:
                                                                    int
                                                                    =
                                                                    10240,
                                                                    **kwargs

```

Bases: object

**stream** (*self*, *route*)

Makes an http request and returns an iterator over the response.

#### Parameters

**route** : the http route (under this object's host) to request against.

```

allensdk.brain_observatory.ecephys.ecephys_project_api.http_engine.write_bytes_from_corouti

```

Utility for streaming http from an asynchronous requester to a file.

#### Parameters

**path** : Write to this file

**coroutine** :

**The source of the data. Needs to have a specific structure, namely:**

- the first-position parameter of the coroutine ought to accept a callback. This callback ought to itself be awaitable. - within the coroutine, this callback ought to be called with a single argument. That single argument should be an asynchronous iterator.

Please see AsyncHttpEngine.stream (and AsyncHttpEngine.\_stream\_coroutine) for an example.

```

allensdk.brain_observatory.ecephys.ecephys_project_api.http_engine.write_from_stream (path:
                                                                    str,
                                                                    stream:
                                                                    It-
                                                                    er-
                                                                    able[by

```

Write bytes to a file from an iterator

#### Parameters

**path** : write to this file

**stream** : iterable yielding bytes to be written

### **allensdk.brain\_observatory.ecephys.ecephys\_project\_api.rma\_engine module**

```
class allensdk.brain_observatory.ecephys.ecephys_project_api.rma_engine.AsyncRmaEngine(scheme: str, host: str, rma_prefix: str, rma_format: str, page_size: int, **kwargs)
```

Bases: `allensdk.brain_observatory.ecephys.ecephys_project_api.rma_engine.RmaEngine`, `allensdk.brain_observatory.ecephys.ecephys_project_api.http_engine.AsyncHttpEngine`

```
class allensdk.brain_observatory.ecephys.ecephys_project_api.rma_engine.RmaEngine(scheme: str, host: str, rma_prefix: str, rma_format: str, page_size: int, **kwargs)
```

Bases: `allensdk.brain_observatory.ecephys.ecephys_project_api.http_engine.HttpEngine`

**add\_page\_params** (self, url, start, count=None)

**format\_query\_string**

**get\_rma** (self, query: str)  
Makes a paging rma query

#### **Parameters**

**query** : The RMA query parameters

**get\_rma\_list** (self, query)

**get\_rma\_tabular** (self, query, try\_infer\_dtypes=True)

**exception** allensdk.brain\_observatory.ecephys.ecephys\_project\_api.rma\_engine.**RmaRequestError**  
Bases: Exception

**infer\_column\_types** (dataframe)  
RMA queries often come back with string-typed columns. This utility tries to infer numeric types.



## allensdk.brain\_observatory.ecephys.ecephys\_project\_api.utilities module

```
allensdk.brain_observatory.ecephys.ecephys_project_api.utilities.build_and_execute(query,
                                          base=None,
                                          engine=None,
                                          **kwargs)

allensdk.brain_observatory.ecephys.ecephys_project_api.utilities.build_environment(template_script,
                                          base=None)

allensdk.brain_observatory.ecephys.ecephys_project_api.utilities.execute_templated(environment_script,
                                          name,
                                          engine,
                                          engine_kwargs,
                                          **kwargs)

allensdk.brain_observatory.ecephys.ecephys_project_api.utilities.macros()

allensdk.brain_observatory.ecephys.ecephys_project_api.utilities.postgres_macros()

allensdk.brain_observatory.ecephys.ecephys_project_api.utilities.rma_macros()
```

## Module contents

### allensdk.brain\_observatory.ecephys.ecephys\_session\_api package

#### Submodules

### allensdk.brain\_observatory.ecephys.ecephys\_session\_api.ecephys\_nwb1\_session\_api module

```
class allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_nwb1_session_api.EcephysSession
```

Bases: `allensdk.brain_observatory.ecephys.ecephys_session_api.EcephysSessionApi`

An EcephysSession adaptor for reading NWB1.0 files.

Was created by sight using an assortment of existing NWB1 files. It is possible that parts of the NWB1 standard (!) is missing or not properly implemented.

NWB1 vs NWB2 issues: \* In NWB 1 there is no difference between global unit-ids and probe's local-index. A unit is unique to one channel \* Units are missing information about firing\_rate, isi\_violation, and quality.

- So that EcephysSession.\_build\_units() actually return values I had to set quality=good for all units
- NWB Stimulus\_presentations missing stimulus\_block, stimulus\_index and Image column - To get EcephysSession.conditionwise\_spikes() working had to make up a block number for every stimulus type
- NWB1 missing a 'valid\_data' tag for channels. Had to set to True otherwise EcephysSession won't see any channels
- There were no 'channels' table/group in NWB1. Instead we had to iterate through all the units and pull out the distinct channel info.

- In NWB2 each unit has a mean-waveform for every channel on the probe. In NWB1 A unit only has a single waveform
- The NWB1 identifier is a string

```
classmethod from_path (path, **kwargs)
get_channels (self) → pandas.core.frame.DataFrame
get_ecephys_session_id (self) → int
get_mean_waveforms (self) → Dict[int, numpy.ndarray]
get_probes (self) → pandas.core.frame.DataFrame
get_running_speed (self)
get_spike_times (self) → Dict[int, numpy.ndarray]
get_stimulus_presentations (self) → pandas.core.frame.DataFrame
get_units (self) → pandas.core.frame.DataFrame
processing_grp
running_speed_grp
class allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_nwb1_session_api.IDCRe
    Bases: object
    get_id (self, key)
```

#### **allensdk.brain\_observatory.ecephys.ecephys\_session\_api.ecephys\_nwb\_session\_api** module

```
class allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_nwb_session_api.Ecephys
```

```
Bases:          allensdk.brain_observatory.nwb.nwb_api.NwbApi,          allensdk.
brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.
EcephysSessionApi
```

```
get_channels (self) → pandas.core.frame.DataFrame
get_current_source_density (self, probe_id)
get_ecephys_session_id (self) → int
get_lfp (self, probe_id: int) → xarray.core.dataarray.DataArray
get_mean_waveforms (self) → Dict[int, numpy.ndarray]
```

```

get_metadata (self)
get_optogenetic_stimulation (self) → pandas.core.frame.DataFrame
get_probes (self) → pandas.core.frame.DataFrame
get_pupil_data (self, suppress_pupil_data: bool = True) → Union[pandas.core.frame.DataFrame,
    NoneType]
get_raw_running_data (self)
get_rig_metadata (self) → Union[dict, NoneType]
get_running_speed (self, include_rotation=False)
get_session_start_time (self)
get_spike_amplitudes (self) → Dict[int, numpy.ndarray]
get_spike_times (self) → Dict[int, numpy.ndarray]
get_stimulus_presentations (self)
get_units (self) → pandas.core.frame.DataFrame
test (self)
    A minimal test to make sure that this API's NWB file exists and is readable. Ecephys NWB files use
    the required session identifier field to store the session id, so this is guaranteed to be present for any
    uncorrupted NWB file.

```

Of course, this does not ensure that the file as a whole is correct.

```
allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_nwb_session_api.clobbering_r
```

## allensdk.brain\_observatory.ecephys.ecephys\_session\_api.ecephys\_session\_api module

```
class allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi
```

```

Bases: object
get_channels (self) → pandas.core.frame.DataFrame
get_ecephys_session_id (self) → int
get_invalid_times (self) → pandas.core.frame.DataFrame
get_lfp (self, probe_id: int) → xarray.core.dataarray.DataArray
get_mean_waveforms (self) → Dict[int, numpy.ndarray]
get_metadata (self)
get_optogenetic_stimulation (self) → pandas.core.frame.DataFrame
get_probes (self) → pandas.core.frame.DataFrame
get_pupil_data (self, suppress_pupil_data: bool) → Union[pandas.core.frame.DataFrame, None-
    Type]
get_rig_metadata (self) → Union[dict, NoneType]
get_running_speed (self) → allensdk.brain_observatory.running_speed.RunningSpeed
get_session_start_time (self) → datetime.datetime

```

```
get_spike_amplitudes (self) → Dict[int, numpy.ndarray]
get_spike_times (self) → Dict[int, numpy.ndarray]
get_stimulus_presentations (self) → pandas.core.frame.DataFrame
get_units (self) → pandas.core.frame.DataFrame
session_na = -1
test (self) → bool
```

## Module contents

### allensdk.brain\_observatory.ecephys.file\_io package

#### Submodules

#### allensdk.brain\_observatory.ecephys.file\_io.continuous\_file module

```
class allensdk.brain_observatory.ecephys.file_io.continuous_file.ContinuousFile (data_path,
                                                                                     times-
                                                                                     tamps_path,
                                                                                     to-
                                                                                     tal_num_chann
                                                                                     dtype=<class
                                                                                     'numpy.int16'>)
```

Bases: object

Represents a continuous (.dat) file, and its associated timestamps

```
get_lfp_channel_order (self)
    Returns the channel ordering for LFP data extracted from NPX files.

    None
```

```
load (self, memmap=False, memmap_thresh=10000000000.0)
    Reads lfp data and timestamps from the filesystem
```

**memmap** [bool, optional] If True, the returned data array will be a memory map of the file on disk. Default is True.

**memmap\_thresh** [float, optional] Files above this size in bytes will be memory-mapped, regardless of memmap setting

#### allensdk.brain\_observatory.ecephys.file\_io.ecephys\_sync\_dataset module

```
class allensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset.EcephysSyncDataset
```

Bases: *allensdk.brain\_observatory.sync\_dataset.Dataset*

```
extract_frame_times (self, strategy, photodiode_cycle=60, frame_keys=('frames', 'stim_vsync'),
                      photodiode_keys=('photodiode', 'stim_photodiode'))
```

```
extract_frame_times_from_photodiode (self, photodiode_cycle=60, frame_keys=('frames',
                                                                              'stim_vsync'),
                                     photodiode_keys=('photodiode',
                                                                              'stim_photodiode'))
```

```
extract_frame_times_from_vsyncs (self, photodiode_cycle=60, frame_keys=('frames',
                                                                    'stim_vsync'),
                                photodiode_keys=('photodiode',
                                                                    'stim_photodiode'))
```

```
extract_led_times (self, keys=('LED_sync', 'opto_trial'), fallback_line=18)
```

```
classmethod factory (path)
```

Build a new SyncDataset.

#### Parameters

**path** [str] Filesystem path to the h5 file containing sync information to be loaded.

**sample\_frequency**

### allensdk.brain\_observatory.ecephys.file\_io.stim\_file module

```
class allensdk.brain_observatory.ecephys.file_io.stim_file.CamStimOnePickleStimFile (data,
                                                                                      **kwargs)
```

Bases: object

```
angular_wheel_rotation
```

Extract the total rotation of the running wheel on each frame.

```
angular_wheel_velocity
```

Extract the mean angular velocity of the running wheel (degrees / s) for each frame.

```
classmethod factory (path, **kwargs)
```

```
frames_per_second
```

Framerate of stimulus presentation

```
pre_blank_sec
```

Time (s) before initial stimulus presentation

```
stimuli
```

List of dictionaries containing information about individual stimuli

```
vin
```

```
vsig
```

Running speed signal voltage

### Module contents

#### allensdk.brain\_observatory.ecephys.lfp\_subsampling package

#### Submodules

**allensdk.brain\_observatory.ecephys.lfp\_subsampling.subsampling module**

`allensdk.brain_observatory.ecephys.lfp_subsampling.subsampling.remove_lfp_noise` (*lfp*,  
*surface\_channel*,  
*channel\_numbers*,  
*channel\_max=384*,  
*channel\_limit=380*)

Subtract mean of channels out of brain to remove noise

**lfp** [numpy.ndarray] 2D array of LFP values (time x channels)

**surface\_channel** [int] Surface channel (relative to original probe)

**channel\_numbers** [numpy.ndarray] Channel numbers in 'lfp' array (relative to original probe)

Returns:

**lfp\_noise\_removed** [numpy.ndarray] New 2D array of LFP values

`allensdk.brain_observatory.ecephys.lfp_subsampling.subsampling.remove_lfp_offset` (*lfp*,  
*sampling\_frequency*,  
*cutoff\_frequency*,  
*filter\_order*)

High-pass filters LFP data to remove offset

**lfp** [numpy.ndarray] 2D array of LFP values (time x channels)

**sampling\_frequency** [float] Sampling frequency in Hz

**cutoff\_frequency** [float] Cutoff frequency for highpass filter

**filter\_order** [int] Butterworth filter order

Returns:

**lfp\_filtered** [numpy.ndarray] New 2D array of LFP values

`allensdk.brain_observatory.ecephys.lfp_subsampling.subsampling.select_channels` (*total\_channels*, *surface\_channel*, *surface\_padding*, *start\_channel\_offset*, *channel\_stride*, *channel\_order*, *noisy\_channels*, *dtype=float64*), *remove\_noisy\_channels*, *reference\_channels*, *dtype=float64*), *remove\_references*

Selects a subset of channels for spatial downsampling

**total\_channels** [int] Number of channels in the original data file

**surface\_channel** [int] Index of channel at brain surface

**surface\_padding** [int] Number of channels above surface to save

**start\_channel\_offset** [int] First channel to save

**channel\_stride** [int] Number of channels to skip in output

**channel\_order** [np.ndarray] Actual order of LFP channels (needed to account for the bug in NPX extraction)

**noisy\_channels** [numpy.ndarray] Array indicating noisy channels

**remove\_noisy\_channels** [bool] Flag to remove noisy channels

**reference\_channels** [numpy.ndarray] Array indicating reference channels

**remove\_references** [bool] Flag to remove reference channels

`allensdk.brain_observatory.ecephys.lfp_subsampling.subsampling.subsample_lfp` (*lfp\_raw*, *selected\_channels*, *subsampling\_factor*)

Subsamples LFP data

**lfp\_raw** [numpy.ndarray] 2D array of LFP values (time x channels)

**selected\_channels** [numpy.ndarray] Indices of channels to select (spatial subsampling)

**downsampling\_factor** [int] Factor by which to subsample in time

Returns:

**lfp\_subsampled** [numpy.ndarray] New 2D array of LFP values

`allensdk.brain_observatory.ecephys.lfp_subsampling.subsampling.subsample_timestamps` (*timestamp*  
*sub-*  
*sam-*  
*pling\_fac*

Subsamples an array of timestamps

**timestamps** [numpy.ndarray] 1D array of timestamp values

**downsampling\_factor** [int] Factor by which to subsample the timestamps

Returns:

**timestamps\_sub** [numpy.ndarray] New 1D array of timestamps

## Module contents

### allensdk.brain\_observatory.ecephys.nwb package

## Module contents

```
class allensdk.brain_observatory.ecephys.nwb.EcephysLabMetaData (**kwargs)
    Bases: pynwb.file.LabMetaData
```

**age\_in\_days**  
age of this subject, in days

**full\_genotype**  
long-form description of this subject's genotype

**namespace** = 'AIBS\_ecephys'

**neurodata\_type** = 'EcephysLabMetaData'

**sex**  
this subject's sex

**specimen\_name**  
full name of this specimen

**stimulus\_name**  
the name of the stimulus set used for this session

**strain**  
this subject's strain

**to\_dict** (*self*)

```
class allensdk.brain_observatory.ecephys.nwb.EcephysProbe (**kwargs)
    Bases: pynwb.ecephys.ElectrodeGroup
```

**has\_lfp\_data**

**lfp\_sampling\_rate**

**namespace** = 'AIBS\_ecephys'

**neurodata\_type** = 'EcephysProbe'

**sampling\_rate**



## allensdk.brain\_observatory.ecephys.optotagging\_table package

### Module contents

## allensdk.brain\_observatory.ecephys.stimulus\_analysis package

### Submodules

## allensdk.brain\_observatory.ecephys.stimulus\_analysis.dot\_motion module

**class** allensdk.brain\_observatory.ecephys.stimulus\_analysis.dot\_motion.**DotMotion** (*ecephys\_session*, *col\_dir='Dir'*, *col\_speeds='Sp'*, *trial\_duration=**\*\*kwargs*)

Bases: `allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.StimulusAnalysis`

A class for computing single-unit metrics from the dot motion stimulus of an ecephys session NWB file.

**To use, pass in a EcephysSession object::** `session = EcephysSession.from_nwb_path('/path/to/my.nwb')`  
`dm_analysis = DotMotion(session)`

**or, alternatively, pass in the file path::** `dm_analysis = DotMotion('/path/to/my.nwb')`

You can also pass in a unit filter dictionary which will only select units with certain properties. For example to get only those units which are on probe C and found in the VISp area:

```
dm_analysis = DotMotion(session, filter={'location': 'probeC', 'ecephys_structure_
↳acronym': 'VISp'})
```

**or a list of unit\_ids:** `dm_analysis = DotMotion(session, filter=[914580630, 914580280, 914580278])`

**To get a table of the individual unit metrics ranked by unit ID::** `metrics_table_df = dm_analysis.metrics()`

### METRICS\_COLUMNS

#### directions

#### known\_spontaneous\_keys

#### classmethod known\_stimulus\_keys()

Used for discovering the correct stimulus\_name key for a given StimulusAnalysis subclass (when stimulus\_key is not explicitly set). Should return a list of “stimulus\_name” strings.

#### metrics

Returns a pandas DataFrame of the stimulus response metrics for each unit.

#### name

Return the stimulus name.

#### null\_condition

Stimulus condition ID for null stimulus (not used, so set to -1)

#### number\_directions

#### number\_speeds

#### speeds

**allensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_gratings module**

**class** allensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_gratings.**DriftingGratings**

Bases: `allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.StimulusAnalysis`

A class for computing single-unit metrics from the drifting gratings stimulus of an ecephys session NWB file.

**To use, pass in a EcephysSession object::** `session = EcephysSession.from_nwb_path('/path/to/my.nwb')`  
`dg_analysis = DriftingGratings(session)`

**or, alternatively, pass in the file path::** `dg_analysis = DriftingGratings('/path/to/my.nwb')`

You can also pass in a unit filter dictionary which will only select units with certain properties. For example to get only those units which are on probe C and found in the VISp area:

```
dg_analysis = DriftingGratings(session, filter={'location': 'probeC', 'structure_
→acronym': 'VISp'})
```

**To get a table of the individual unit metrics ranked by unit ID::** `metrics_table_df = dg_analysis.metrics()`

**METRICS\_COLUMNS**

**conditionwise\_statistics\_contrast**

Conditionwise statistics for contrast stimulus

**contrastvals**

Array of grating temporal frequency conditions

**classmethod known\_stimulus\_keys()**

Used for discovering the correct stimulus\_name key for a given StimulusAnalysis subclass (when stimulus\_key is not explicitly set). Should return a list of “stimulus\_name” strings.

**make\_star\_plot(self, unit\_id)**

Make a 2P-style Star Plot based on presentationwise spike counts

**metrics**

Returns a pandas DataFrame of the stimulus response metrics for each unit.

**name**

Return the stimulus name.

**null\_condition**

Stimulus condition ID for null (blank) stimulus

**number\_contrast**

Number of grating temporal frequency conditions

**number\_ori**

Number of grating orientation conditions

**number\_tf**

Number of grating temporal frequency conditions

**orivals**

Array of grating orientation conditions

**plot\_raster** (*self*, *stimulus\_condition\_id*, *unit\_id*)

Plot raster for one condition and one unit

**plot\_response\_summary** (*self*, *unit\_id*, *bar\_thickness*=0.25)

Plot the spike counts across conditions

**stim\_table\_contrast**

**stimulus\_conditions\_contrast**

Stimulus conditions for contrast stimulus

**tfvals**

Array of grating temporal frequency conditions

`allensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.c50` (*contrasts*,  
*re-*  
*sponses*)

Computes C50, the halfway point between the maximum and minimum values in a curved fitted against a difference of gaussian for the contrast values and their responses (mean spike rates)

#### Parameters

**contrasts** [array of floats] list of different contrast stimuli

**responses** [array of floats] array of responses (spike rates)

#### Returns

**c50** [float]

`allensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.f1_f0` (*arr*,  
*tf*,  
*trial\_duration*)

Computes F1/F0 of a drifting grating response

#### Parameters

**arr** : DataArray with trials x bin-times

**tf** : temporal frequency of the stimulus

#### Returns

**f1\_f0** [float] metric

`allensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.modulation_index` (*resp*,  
*tf*,  
*sample\_rate*,  
*ple*)

Depth of modulation by each cycle of a drifting grating; similar to F1/F0

**ref:** **Matteucci et al. (2019) Nonlinear processing of shape information** in rat lateral extrastriate cortex. J Neurosci 39: 1649-1670

#### Parameters

**response\_psth** [array of floats] the binned responses of a unit for a given stimuli

**tf** [float] the temporal frequency

**sample\_rate** [float] the sampling rate of response\_psth

#### Returns

**modulation\_index** [float] the mi value

**allensdk.brain\_observatory.ecephys.stimulus\_analysis.flashes module**

```
class allensdk.brain_observatory.ecephys.stimulus_analysis.flashes.Flashes (ecephys_session,  
                                                                    col_color='color',  
                                                                    trial_duration=0.25,  
                                                                    **kwargs)
```

Bases: `allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.StimulusAnalysis`

A class for computing single-unit metrics from the full-field flash stimulus of an ecephys session NWB file.

**To use, pass in a EcephysSession object::** session = EcephysSession.from\_nwb\_path('/path/to/my.nwb')  
fl\_analysis = Flashes(session)

**or, alternatively, pass in the file path::** fl\_analysis = Flashes('/path/to/my.nwb')

You can also pass in a unit filter dictionary which will only select units with certain properties. For example to get only those units which are on probe C and found in the VISp area:

```
fl_analysis = Flashes(session, filter={'location': 'probeC', 'ecephys_structure_  
→acronym': 'VISp'})
```

**To get a table of the individual unit metrics ranked by unit ID::** metrics\_table\_df = fl\_analysis.metrics()

**METRICS\_COLUMNS****colors**

Array of 'color' conditions (black vs. white flash)

**classmethod known\_stimulus\_keys()**

Used for discovering the correct stimulus\_name key for a given StimulusAnalysis subclass (when stimulus\_key is not explicitly set). Should return a list of "stimulus\_name" strings.

**metrics**

Returns a pandas DataFrame of the stimulus response metrics for each unit.

**name**

Return the stimulus name.

**null\_condition**

Stimulus condition ID for null stimulus (not used, so set to -1)

**number\_colors**

Number of 'color' conditions (black vs. white flash)

**plot\_raster** (self, stimulus\_condition\_id, unit\_id)

Plot raster for one condition and one unit

**plot\_response** (self, unit\_id)

Plot a histogram for the two conditions

**allensdk.brain\_observatory.ecephys.stimulus\_analysis.natural\_movies module**

```
class allensdk.brain_observatory.ecephys.stimulus_analysis.natural_movies.NaturalMovies (ece  
                                                                    trial  
                                                                    **k
```

Bases: `allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.StimulusAnalysis`

A class for computing single-unit metrics from the natural movies stimulus of an ecephys session NWB file.

**To use, pass in a EcephysSession object::** session = EcephysSession.from\_nwb\_path('/path/to/my.nwb')  
nm\_analysis = NaturalMovies(session)

**or, alternatively, pass in the file path::** nm\_analysis = Flashes('/path/to/my.nwb')

You can also pass in a unit filter dictionary which will only select units with certain properties. For example to get only those units which are on probe C and found in the VISp area:

```
nm_analysis = NaturalMovies(session, filter={'location': 'probeC', 'ecephys_
↳structure_acronym': 'VISp'})
```

**To get a table of the individual unit metrics ranked by unit ID::** metrics\_table\_df = nm\_analysis.metrics()

TODO: Need to find a default trial\_duration otherwise class will fail

#### METRICS\_COLUMNS

**classmethod known\_stimulus\_keys()**

Used for discovering the correct stimulus\_name key for a given StimulusAnalysis subclass (when stimulus\_key is not explicitly set). Should return a list of “stimulus\_name” strings.

**metrics**

Returns a pandas DataFrame of the stimulus response metrics for each unit.

**name**

Return the stimulus name.

**null\_condition**

### allensdk.brain\_observatory.ecephys.stimulus\_analysis.natural\_scenes module

**class** allensdk.brain\_observatory.ecephys.stimulus\_analysis.natural\_scenes.NaturalScenes (ecephys.stimulus\_analysis.NaturalScenes) *(ecephys.stimulus\_analysis.NaturalScenes)*

Bases: *allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis.StimulusAnalysis*

A class for computing single-unit metrics from the natural scenes stimulus of an ecephys session NWB file.

**To use, pass in a EcephysSession object::** session = EcephysSession.from\_nwb\_path('/path/to/my.nwb')  
ns\_analysis = NaturalScenes(session)

**or, alternatively, pass in the file path::** ns\_analysis = NaturalScenes('/path/to/my.nwb')

You can also pass in a unit filter dictionary which will only select units with certain properties. For example to get only those units which are on probe C and found in the VISp area:

```
ns_analysis = NaturalScenes(session, filter={'location': 'probeC', 'ecephys_
↳structure_acronym': 'VISp'})
```

**To get a table of the individual unit metrics ranked by unit ID::** metrics\_table\_df = ns\_analysis.metrics()

#### METRICS\_COLUMNS

**frames**

**images**

Array of image labels

**images\_nonblank****classmethod known\_stimulus\_keys()**

Used for discovering the correct stimulus\_name key for a given StimulusAnalysis subclass (when stimulus\_key is not explicitly set). Should return a list of “stimulus\_name” strings.

**metrics**

Returns a pandas DataFrame of the stimulus response metrics for each unit.

**name**

Return the stimulus name.

**null\_condition**

Stimulus condition ID for null (blank) stimulus

**number\_images**

Number of images shown

**number\_nonblank**

Number of images shown (excluding blank condition)

`allensdk.brain_observatory.ecephys.stimulus_analysis.natural_scenes.image_selectivity(spike_num_s`

Quantifies how selective a cell is for images, based on Quian Quiroga et al., 2007. A value of 0 indicates the cell responds the same no matter what the image. While if the neuron only responds to a single image it will have a selectivity of  $1 - 2/N$  (1.0 and  $N$  goes to  $\infty$ ).

**Parameters**

**spike\_means** [array of floats] Averaged spiking responses to a series of images for a given neuron

**num\_steps** [int] Number of threshold values used to build response distribution (default to 1000 as in Quian paper)

**Returns**

**selectivity** [float] selectivity of neuron to images

**allensdk.brain\_observatory.ecephys.stimulus\_analysis.receptive\_field\_mapping module**

**class** `allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.ReceptiveFieldMapping`

Bases: `allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.StimulusAnalysis`

A class for computing single-unit metrics from the receptive field mapping stimulus of an ecephys session NWB file.

**To use, pass in a EcephysSession object::** `session = EcephysSession.from_nwb_path('/path/to/my.nwb')`  
`rf_analysis = ReceptiveFieldMapping(session)`

or, alternatively, pass in the file path:: `rf_analysis = ReceptiveFieldMapping('/path/to/my.nwb')`

You can also pass in a unit filter dictionary which will only select units with certain properties. For example to get only those units which are on probe C and found in the VISp area:

```
rf_analysis = ReceptiveFieldMapping(session, filter={'location': 'probeC',
↪ 'ecephys_structure_acronym': 'VISp'})
```

To get a table of the individual unit metrics ranked by unit ID:: `metrics_table_df = rf_analysis.metrics()`

#### **METRICS\_COLUMNS**

##### **azimuths**

Array of stimulus azimuths

##### **elevations**

Array of stimulus elevations

##### **get\_receptive\_field(*self*, *unit\_id*)**

Alias for `_get_rf`

##### **classmethod known\_stimulus\_keys()**

Used for discovering the correct `stimulus_name` key for a given `StimulusAnalysis` subclass (when `stimulus_key` is not explicitly set). Should return a list of “`stimulus_name`” strings.

##### **metrics**

Returns a pandas DataFrame of the stimulus response metrics for each unit.

##### **name**

Return the stimulus name.

##### **null\_condition**

Stimulus condition ID for null stimulus (not used, so set to -1)

##### **number\_azimuths**

Number of stimulus azimuths

##### **number\_elevations**

Number of stimulus elevations

##### **plot\_raster(*self*, *stimulus\_condition\_id*, *unit\_id*)**

Plot raster for one condition and one unit

##### **plot\_rf(*self*, *unit\_id*)**

Plot the spike counts across conditions

##### **receptive\_fields**

Spatial receptive fields for N units (9 x 9 x N matrix of responses)

`allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.convert_azimuth`

Converts a pixel-based azimuth into degrees relative to center of gaze

#### **Parameters**

**azimuth\_in\_pixels** [float]

**azimuth\_offset\_degrees**: float

#### **Returns**

**azimuth\_in\_degrees** [float]

```
allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.convert_elevation
```

Converts a pixel-based elevation into degrees relative to center of gaze

The receptive field computed by this class is oriented such that the pixel values are in the correct relative location when using `matplotlib.pyplot.imshow()`, which places (0,0) in the upper-left corner of the figure.

Therefore, we need to invert the elevation value prior to converting to degrees.

#### Parameters

**elevation\_in\_pixels** [float]

**elevation\_offset\_degrees:** float

#### Returns

**elevation\_in\_degrees** [float]

```
allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.convert_pixel_area
```

Converts a pixel-based area measure into degrees

Each pixel is a square with side of length `<degrees_to_pixels_ratio>`

So the area in degrees is `area_in_pixels * <degrees_to_pixels_ratio>^2`

#### Parameters

**area\_in\_pixels** [float]

#### Returns

**area\_in\_degrees** [float]

```
allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.convert_pixel_distance
```

Converts a pixel-based distance into degrees

#### Parameters

**value\_in\_pixels** [float]

**degrees\_to\_pixels\_ratio:** float

#### Returns

**value in degrees** [float]

```
allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.fit_2d_gaussian
```

Fits a receptive field with a 2-dimensional Gaussian distribution

#### Parameters

**matrix** [numpy.ndarray] 2D matrix of spike counts

#### Returns

**parameters - tuple** `peak_height` : peak of distribution `center_y` : y-coordinate of distribution center `center_x` : x-coordinate of distribution center `width_y` : width of distribution along x-axis `width_x` : width of distribution along y-axis

**success - bool** True if a fit was found, False otherwise

```
allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.gaussian_moments
```

Finds the moments of a 2D Gaussian distribution, given an input matrix



**Parameters****data** [numpy.ndarray] 2D matrix**Returns****peak\_height** : peak of distribution**center\_y** : y-coordinate of distribution center**center\_x** : x-coordinate of distribution center**width\_y** : width of distribution along x-axis**width\_x** : width of distribution along y-axis`allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.invert_rf (rf)`

Creates an inverted version of the receptive field

**Parameters****rf** - matrix of spike counts at each stimulus position**Returns****rf\_inverted** - new RF matrix`allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.is_rf_inverted`

Checks if the receptive field mapping stimulus is suppressing or exciting the cell

**Parameters****rf\_thresh** [matrix] matrix of spike counts at each stimulus position**Returns****if\_rf\_inverted** [bool] True if the receptive field is inverted`allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.rf_on_screen (rf)`

Checks whether the receptive field is on the screen, given the center location.

`allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.threshold_rf (rf)`

Creates a spatial mask based on the receptive field peak, and returns the x, y coordinates of the center of mass, as well as the area.

**Parameters****rf** [numpy.ndarray] 2D matrix of spike counts**threshold** [float] Threshold as ratio of the RF's standard deviation**Returns****threshold\_rf** [numpy.ndarray] Thresholded version of the original RF**center\_x** [float] x-coordinate of mask center of mass**center\_y** [float] y-coordinate of mask center of mass**area** [float] area of mask

**allensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_gratings module****class** allensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_gratings.**StaticGratings** (

Bases: `allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.StimulusAnalysis`

A class for computing single-unit metrics from the static gratings stimulus of an ecephys session NWB file.

**To use, pass in a EcephysSession object::** `session = EcephysSession.from_nwb_path('/path/to/my.nwb')`  
`sg_analysis = StaticGratings(session)`

**or, alternatively, pass in the file path::** `sg_analysis = StaticGratings('/path/to/my.nwb')`

You can also pass in a unit filter dictionary which will only select units with certain properties. For example to get only those units which are on probe C and found in the VISp area:

```
sg_analysis = StaticGratings(session, filter={'location': 'probeC', 'ecephys_
↪structure_acronym': 'VISp'})
```

**To get a table of the individual unit metrics ranked by unit ID::** `metrics_table_df = sg_analysis.metrics()`

**METRICS\_COLUMNS****classmethod** `known_stimulus_keys()`

Used for discovering the correct stimulus\_name key for a given StimulusAnalysis subclass (when stimulus\_key is not explicitly set). Should return a list of “stimulus\_name” strings.

**make\_fan\_plot** (*self*, *unit\_id*)

Make a 2P-style Fan Plot based on presentationwise spike counts

**metrics**

Returns a pandas DataFrame of the stimulus response metrics for each unit.

**name**

Return the stimulus name.

**null\_condition**

Stimulus condition ID for null (blank) stimulus

**number\_ori**

Number of grating orientation conditions

**number\_phase**

Number of grating phase conditions

**number\_sf**

Number of grating orientation conditions

**orivals**

Array of grating orientation conditions

**phasevals**

Array of grating phase conditions

**plot\_raster** (*self*, *stimulus\_condition\_id*, *unit\_id*)

Plot raster for one condition and one unit

**plot\_response\_summary** (*self*, *unit\_id*, *bar\_thickness=0.25*)

Plot the spike counts across conditions

**sfvals**

Array of grating spatial frequency conditions

`allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.exp_function` (*x*,  
*a*,  
*b*,  
*c*)

`allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.fit_sf_tuning` (*sf\_tuning\_r*,  
*sf\_values*,  
*pref\_sf\_index*)

Performs gaussian or exponential fit on the spatial frequency tuning curve at preferred orientation/phase for a given cell.

#### Parameters

- **sf\_tuning\_responses** – An array of len N, with each value the (averaged) response of a cell at a given spatial freq. stimulus.
- **sf\_values** – An array of len N, with each value the spatial freq. of the stimulus (corresponding to sf\_tuning\_response).
- **pref\_sf\_index** – The pre-determined preferred spatial frequency (sf\_values index) of the cell.

**Returns** index for the preferred sf from the curve fit, preferred sf from the curve fit, low cutoff sf from the curve fit, high cutoff sf from the curve fit

`allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.gauss_function` (*x*,  
*a*,  
*x0*,  
*sigma*)

`allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.get_sfdi` (*sf\_tuning\_responses*,  
*mean\_sweeps\_trials*,  
*bias=5*)

Computes spatial frequency discrimination index for cell

#### Parameters

- **sf\_tuning\_responses** – sf\_tuning\_responses: An array of len N, with each value the (averaged) response of a cell at a given spatial freq. stimulus.
- **mean\_sweeps\_trials** – The set of events (spikes) across all trials of varying
- **bias** –

**Returns** The sfdi value (float)

### `allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis` module

**class** `allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.StimulusAnalysis`

Bases: `object`

**METRICS\_COLUMNS**

**conditionwise\_psth**

For every unit and stimulus-condition construction a PSTH table. ie. the spike-counts at a each time-interval during a stimulus, averaged over all trials of the same stim condition.

Each PSTH will count and average spikes over a time-window as determined by class parameter ‘trial\_duration’ which ideally be a similar value as the duration of each stimulus (in seconds). The length of each time-bin is determined by the class parameter ‘psth\_resolution’ (in seconds).

**Returns**

**conditionwise\_psth** xarray.DataArray

An 3D table that contains the PSTH for every unit/condition, with the following coordinates

- stimulus\_condition\_id
- time\_relative\_to\_stimulus\_onset
- unit\_id

**conditionwise\_statistics**

Create a table of spike statistics, averaged and indexed by every unit\_id, stimulus\_condition\_id pair.

**Returns**

**conditionwise\_statistics: pd.DataFrame** A dataframe indexed by unit\_id and stimulus\_condition containing spike\_count, spike\_mean, spike\_sem, spike\_std and stimulus\_presentation\_count information.

**ecephys\_session**

**empty\_metrics\_table** (*self*)

**get\_intrinsic\_timescale** (*self*, *unit\_ids*)

Calculates the intrinsic timescale for a subset of units

**known\_spontaneous\_keys**

**classmethod known\_stimulus\_keys** ()

Used for discovering the correct stimulus\_name key for a given StimulusAnalysis subclass (when stimulus\_key is not explicitly set). Should return a list of “stimulus\_name” strings.

**metrics**

Returns a pandas DataFrame of the stimulus response metrics for each unit.

**metrics\_dtypes****metrics\_names****name**

Return the stimulus name.

**null\_condition**

**plot\_conditionwise\_raster** (*self*, *unit\_id*)

Plot a matrix of rasters for each condition (orientations x temporal frequencies)

**plot\_raster** (*self*, *condition*, *unit\_id*)

**presentationwise\_spike\_times**

Constructs a table containing all the relevant spike\_times plus the stimulus\_presentation\_id and unit\_id for the given spike.

**Returns**

**presentationwise\_spike\_times** [pd.DataFrame] Indexed by spike\_time, each spike containing the corresponding stimulus\_presentation\_id and unit\_id

**presentationwise\_statistics**

Returns a table of the spike-counts, stimulus-conditions and running speed for every stimulus\_presentation\_id, unit\_id pair.

**Returns**

**presentationwise\_statistics:** pd.DataFrame MultiIndex : unit\_id, stimulus\_presentation\_id Columns : spike\_count, stimulus\_condition\_id, running\_speed

**running\_speed**

Construct a dataframe with the averaged running speed for each stimulus\_presentation\_id

**spikes**

Returns a dictionary of unit\_id -> spike-times.

**stim\_table**

**stim\_table\_spontaneous**

Returns a stimulus table with only 'spontaneous' stimulus selected.

**stimulus\_conditions**

Returns a table of relevant stimulus\_conditions.

**Returns**

**pd.DataFrame :** Index : stimulus\_condition\_id Columns : stimulus parameter types

**total\_presentations**

Total number of presentations / trials

**trial\_duration**

**unit\_count**

Get the number of units.

**unit\_ids**

Returns a list of unit IDs for which to apply the analysis

`allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.calculate_time_delay`

`allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.deg2rad(arr)`

Converts array-like input from degrees to radians

`allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.dsi(ori_vals, tuning)`

Computes the direction selectivity of a cell. See Ringbach 2002, Van Hooser 2014

**Parameters**

**ori\_vals** [complex array of length N] Each value the orientation of the stimulus.

**tuning** [float array of length N] Each value the (averaged) response of the cell at a different orientation.

**Returns**

**osi** [float] An N-dimensional array of the circular variance (scalar value, in radians) of the responses.

`allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.fano_factor(spike_counts)`

Computes the fano factor (var/mean) for the spike-counts across a series of trials.

**Parameters**

**spike\_counts** [array] The spike counts across a series of 2 or more trials

**Returns**

**fano\_factor** [float]

`allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.fit_exp(rsc_time_matrix)`

`allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.get_fr(spikes, num_timestep_second, sweep_length=3.1, filter_width=0.1)`

Uses a gaussian convolution to convert the spike-times into a contiguous firing-rate series.

**Parameters**

**spikes** [array] An array of spike times (shifted to start at 0)

**num\_timestep\_second** [float] The sampling frequency

**sweep\_length** [float] The length of the returned array

**filter\_width: float** The window of the gaussian method

**Returns**

**firing\_rate** [float] A linear-spaced array of length `num_timestep_second*sweep_length` of the smoothed firing rates series.

`allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.lifetime_sparseness`

Computes the lifetime sparseness for one unit. See Olsen & Wilson 2008.

**Parameters**

**responses** [array of floats] An array of a unit's spike-counts over the duration of multiple trials within a given session

**Returns**

**lifetime\_sparseness** [float] The lifetime sparseness for one unit

`allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.osi(ori_vals, tuning)`

Computes the orientation selectivity of a cell. The calculation of the orientation is done using the normalized circular variance (CirVar) as described in Ringbach 2002

**Parameters**

**ori\_vals** [complex array of length N] Each value the orientation of the stimulus.

**tuning** [float array of length N] Each value the (averaged) response of the cell at a different orientation.

**Returns**

**osi** [float] An N-dimensional array of the circular variance (scalar value, in radians) of the responses.

`allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.overall_firing_rate`

Computes the global firing rate of a series of spikes, for only those values within the given start and stop times.

**Parameters**

**start\_times** [array of N floats] A series of stimulus block start times (seconds)

**stop\_times** [array of N floats] Times when the stimulus block ends

**spike\_times** [array of floats] A list of spikes for a given unit

**Returns**

**firing\_rate** [float]

```
allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.reliability(unit_sweep, padding=1, num_times=10, filter_width=1, window_beg=0, window_end=1)
```

Computes the trial-to-trial reliability for a set of sweeps for a given cell

**Parameters**

- **unit\_sweeps** –
- **padding** –

**Returns**

```
allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.running_modulation(spike_counts, running_speeds, speed_threshold=1.0)
```

Given a series of trials that include the spike-counts and (averaged) running-speed, does a statistical comparison to see if there was any difference in spike firing while running and while stationary.

Requires at least 2 trials while the mouse is running and two when the mouse is stationary.

**Parameters**

**spike\_counts** [array of floats of size N.] The spike counts for each trial

**running\_speeds: array floats of size N.** The running velocities (cm/s) of each trial.

**speed\_threshold: float** The minimum threshold for which the animal can be considered running (default 1.0).

**Returns**

**p\_value** [float or Nan] T-test p-value between the running and stationary trials.

**run\_mod** [float or Nan] Relative difference between running and stationary mean firing rates.

**Module contents**

**allensdk.brain\_observatory.ecephys.stimulus\_table package**

**Subpackages**

## allensdk.brain\_observatory.ecephys.stimulus\_table.visualization package

### Submodules

#### allensdk.brain\_observatory.ecephys.stimulus\_table.visualization.view\_blocks module

allensdk.brain\_observatory.ecephys.stimulus\_table.visualization.view\_blocks.**build\_colormap**

allensdk.brain\_observatory.ecephys.stimulus\_table.visualization.view\_blocks.**get\_blocks** (*table*

allensdk.brain\_observatory.ecephys.stimulus\_table.visualization.view\_blocks.**main** (*table\_csv\_pat*

allensdk.brain\_observatory.ecephys.stimulus\_table.visualization.view\_blocks.**plot\_blocks** (*blo*  
col-  
orm

### Module contents

### Submodules



**allensdk.brain\_observatory.ecephys.stimulus\_table.ephys\_pre\_spikes module**

Created on Fri Dec 16 15:11:23 2016

@author: Xiaoxuan Jia

`allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes.apply_display_sequence` (s

Adjust raw sweep frames for a stimulus based on the display sequence for that stimulus.

**Parameters**

**sweep\_frames\_table** [pd.DataFrame] Each row is a sweep. Has two columns, 'start' and 'end', which describe (in frames) when that sweep began and ended.

**frame\_display\_sequence** [np.ndarray] 2D array. Rows are display intervals. The 0th column is the start frame of that interval, the 1st the end frame.

**Returns**

**sweep\_frames\_table** [pd.DataFrame] As above, but start and end frames have been adjusted based on the display sequence.

**Notes**

The frame values in the raw sweep\_frames\_table are given in 0-indexed offsets from the start of display for this stimulus. This domain only takes into account frames which are part of a display interval for that stimulus, so the frame ids need to be adjusted to lie on the global frame sequence.

`allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes.apply_frame_times` (stimulus  
frame\_t  
frames\_  
ex-  
tra\_fram  
map\_co  
'End'))

Converts sweep times from frames to seconds.

**Parameters**

**stimulus\_table** [pd.DataFrame] Rows are sweeps. Columns are stimulus parameters as well as start and end frames for each sweep.

**frame\_times** [numpy.ndarray] Gives the time in seconds at which each frame (indices) began.

**frames\_per\_second** [numeric, optional] If provided, and extra\_frame\_time is True, will be used to calculate the extra\_frame\_time.

**extra\_frame\_time** [float, optional] If provided, an additional frame time will be appended. The time will be incremented by extra\_frame\_time from the previous last frame time, to denote the time at which the last frame ended. If False, no extra time will be appended. If None (default), the increment will be 1.0/fps.

**map\_columns** [tuple of str, optional] Which columns to replace with times. Defaults to 'Start' and 'End'

**Returns**

**stimulus\_table** [pd.DataFrame] As above, but with map\_columns values converted to seconds from frames.

```
allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes.assign_sweep_values(stim_
                                                                    sweep_
                                                                    on='
                                                                    drop:
                                                                    tmp_
```

Left joins a stimulus table to a sweep table in order to associate epochs in time with stimulus characteristics.

**Parameters**

**stim\_table** [pd.DataFrame] Each row is a stimulus epoch, with start and end times and a foreign key onto a particular sweep.

**sweep\_table** [pd.DataFrame] Each row is a sweep. Should have columns in common with the stim\_table - the resulting table will use values from the sweep\_table.

**on** [str, optional] Column on which to join.

**drop** [bool, optional] If True (default), the join column (argument on) will be dropped from the output.

**tmp\_suffix** [str, optional] Will be used to identify overlapping columns. Should not appear in the name of any column in either dataframe.

```
allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes.build_stimuluswise_table
```

Construct a table of sweeps, including their times on the experiment-global clock and the values of each relevant parameter.

**Parameters**

**stimulus** [dict] Describes presentation of a stimulus on a particular experiment. Has a number of fields, of which we are using:

**stim\_path** [str] windows file path to the stimulus data

**sweep\_frames** [list of lists] rows are sweeps, columns are start and end frames of that sweep (in the stimulus-specific frame domain). C-order.

**sweep\_order** [list of int] indices are frames, values are the sweep on that frame

**display\_sequence** [list of list]

rows are intervals in which the stimulus was displayed. Columns are start and end times (s, global) of the display. C-order.

**dimnames** [list of str] Names of parameters for this stimulus (such as “Contrast”)

**sweep\_table** [list of tuple] Each element is a tuple of parameter values (1 per dimname) describing a single sweep.

**seconds\_to\_frames** [function] Converts experiment seconds to frames

**start\_key** [str, optional] key to use for start frame indices. Defaults to ‘Start’

**end\_key** [str, optional] key to use for end frame indices. Defaults to ‘End’

**name\_key** [str, optional] key to use for stimulus name annotations. Defaults to ‘stimulus\_name’

**block\_key** [str, optional] key to use for the 0-index position of this stimulus block

**get\_stimulus\_name** [function | dict -> str, optional] extracts stimulus name from the stimulus dictionary. Default is read\_stimulus\_name\_from\_path

### Returns

**list of pandas.DataFrame** : Each table corresponds to an entry in the display sequence. Rows are sweeps, columns are stimulus parameter values as well as “Start” and “End”.

```
allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes.create_stim_table(stimuli,
stim-
u-
lus_table,
spontaneous_activity_table,
sort_key,
block_key,
in-
dex_key)
```

Build a full stimulus table

### Parameters

**stimuli** [list of dict] Each element is a stimulus dictionary, as provided by the stim.pkl file.

**stimulus\_table** [function] A function which takes a single stimulus dictionary as its argument and returns a stimulus table dataframe.

**spontaneous\_activity\_table** [function] A function which takes a list of stimulus tables as arguments and returns a list of 0 or more tables describing spontaneous activity sweeps.

**sort\_key** [str, optional] Sort the final stimulus table in ascending order by this key. Defaults to ‘Start’.

### Returns

**stim\_table\_full** [pandas.DataFrame] Each row is a sweep. Has columns describing (in frames) the start and end times of each sweep. Other columns describe the values of stimulus parameters on those sweeps.

```
allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes.make_spontaneous_activity_table
```

Fills in frame gaps in a set of stimulus tables. Suitable for use as the `spontaneous_activity_table` in `create_stim_table`.

#### Parameters

**stimulus\_tables** [list of `pd.DataFrame`] Input tables - should have `start_key` and `end_key` columns.

**start\_key** [str, optional] Column name for the start of a sweep. Defaults to 'Start'.

**end\_key** [str, optional] Column name for the end of a sweep. Defaults to 'End'.

**duration\_threshold** [numeric or None] If not None (default is 0), remove spontaneous activity sweeps whose duration is less than this threshold.

#### Returns

**list** : Either empty, or contains a single `pd.DataFrame`. The rows of the dataframe are spontaneous activity sweeps.

```
allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes.read_stimulus_name_from_filename
```

Obtains a human-readable stimulus name by looking at the filename of the 'stim\_path' item.

#### Parameters

**stimulus** [dict] must contain a 'stim\_path' item.

#### Returns

**str** : name of stimulus

```
allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes.split_column(table, column, new_columns, drop_old=True)
```

Divides a dataframe column into multiple columns.

#### Parameters

**table** [`pandas.DataFrame`] Columns will be drawn from and assigned to this dataframe. This dataframe will NOT be modified inplace.

**column** [str] This column will be split.

**new\_columns** [dict, mapping strings to functions] Each key will be the name of a new column, while its value (a function) will be used to build the new column's values. The functions should map from a single value of the original column to a single value of the new column.

**drop\_old** [bool, optional] If True, the original column will be dropped from the table.

#### Returns

**table** [`pd.DataFrame`] The modified table

**allensdk.brain\_observatory.ecephys.stimulus\_table.naming\_utilities module**

`allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities.add_number_to_shuffled_r`

`allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities.collapse_columns` (*table*)  
 merge, where possible, columns that describe the same parameter. This is pretty conservative - it only matches columns by capitalization and it only overrides nans.

`allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities.drop_empty_columns` (*table*)  
 Remove from the stimulus table columns whose values are all nan

`allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities.map_column_names` (*table*,  
*name\_map*,  
*sig*,  
*ignore\_case*)

`allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities.map_stimulus_names` (*table*,  
*name\_map*,  
*stim\_colname*)

Applies a mapping to the stimulus names in a stimulus table

**Parameters**

**table** [pd.DataFrame] the input stimulus table

**name\_map** [dict, optional] rename the stimuli according to this mapping

**stim\_colname: str, optional** look in this column for stimulus names

```
allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities.standardize_movie_numbers
```

Natural movie stimuli in visual coding are numbered using words, like “natural\_movie\_two” rather than “natural\_movie\_2”. This function ensures that all of the natural movie stimuli in an experiment are named by that convention.

#### Parameters

**table** [pd.DataFrame] the incoming stimulus table

**movie\_re** [re.Pattern, optional] regex that matches movie stimulus names

**numeral\_re** [re.Pattern, optional] regex that extracts movie numbers from stimulus names

**digit\_names** [dict, optional] map from numerals to english words

**stim\_colname** [str, optional] the name of the dataframe column that contains stimulus names

#### Returns

**table** [pd.DataFrame] the stimulus table with movie numerals having been mapped to english words

### **allensdk.brain\_observatory.ecephys.stimulus\_table.output\_validation module**

```
allensdk.brain_observatory.ecephys.stimulus_table.output_validation.validate_epoch_duration
```

```
allensdk.brain_observatory.ecephys.stimulus_table.output_validation.validate_epoch_order (table, threshold, error_message)
```

```
allensdk.brain_observatory.ecephys.stimulus_table.output_validation.validate_max_spontaneous
```

### **allensdk.brain\_observatory.ecephys.stimulus\_table.stimulus\_parameter\_extraction module**

```
allensdk.brain_observatory.ecephys.stimulus_table.stimulus_parameter_extraction.extract_com
```

Parameters which are not set as sweep\_params in the stimulus script (usually because they are not varied during the course of the session) are not output in an easily machine-readable format. This function attempts to recover them by parsing the string repr of the stimulus.

#### **Parameters**

**stim\_repr** [str]

The repr of the camstim stimulus object. Served up per-stimulus in the stim pickle.

**repr\_params\_re** [re.Pattern] Extracts attributes as “=”-seperated strings

**array\_re** [re.Pattern] Extracts list reprs from numpy array reprs.

#### **Returns**

**repr\_params** [dict] dictionary of parameter keys and values extracted from the stim repr.  
Where possible, the values are converted to native Python types.

```
allensdk.brain_observatory.ecephys.stimulus_table.stimulus_parameter_extraction.extract_st
```

`allensdk.brain_observatory.ecephys.stimulus_table.stimulus_parameter_extraction.parse_stim`

Read the string representation of a psychopy stimulus and extract stimulus parameters.

**Parameters**

**stim\_repr** [str]

**drop\_params** [tuple]

**repr\_params\_re** [re.Pattern]

**array\_re** [re.Pattern]

**Returns**

**dict** : maps extracted parameter names to values

**Module contents**

`allensdk.brain_observatory.ecephys.visualization` package

**Module contents**

`allensdk.brain_observatory.ecephys.visualization.plot_mean_waveforms` (*mean\_waveforms*,  
*unit\_ids*,  
*peak\_channels*)

Utility for plotting mean waveforms on each unit's peak channel

**Parameters**

**mean\_waveforms** [dictionary] Maps unit ids to channelwise average spike waveforms for those units

**unit\_ids** [array-like] unique integer identifiers for units to be included



```
allensdk.brain_observatory.ecephys.visualization.plot_spike_counts(data_array,
                                                                    time_coords,
                                                                    cbar_label,
                                                                    title, xla-
                                                                    bel='time
                                                                    relative to
                                                                    stimulus
                                                                    onset
                                                                    (s)', yla-
                                                                    bel='unit',
                                                                    xtick_step=20)
```

Utility for making a simple spike counts plot.

#### Parameters

**data\_array** [xarray.DataArray] 2D data array unitwise values per time bin. See EcephysSession.sweepwise\_spike\_counts

```
allensdk.brain_observatory.ecephys.visualization.raster_plot(spike_times,
                                                                figsize=(8,
                                                                8),
                                                                cmap=<matplotlib.colors.ListedColormap
                                                                object at
                                                                0x7fdbfaf1d400>,
                                                                title='spike
                                                                raster', cy-
                                                                cle_colors=False)
```

## allensdk.brain\_observatory.ecephys.write\_nwb package

### Module contents

### Submodules

**allensdk.brain\_observatory.ecephys.ecephys\_project\_cache module**

```
class allensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache (fetch_api:
    al-
    lensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache,
    =
    <al-
    lensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache>,
    fetch_tries:
    int
    =
    2,
    stream_writer:
    Callable[[bytes], bytes]
    =
    <function write_from_manifest_i-
    fest: Union[str, path-
    lib.Path, None]
    =
    None,
    version:
    Optional[str]
    =
    None,
    cache:
    bool
    =
    True)

Bases: allensdk.api.cache.Cache

CHANNELS_KEY = 'channels'
MANIFEST_VERSION = '0.2.1'
NATURAL_MOVIE_DIR_KEY = 'movie_dir'
NATURAL_MOVIE_KEY = 'natural_movie'
NATURAL_SCENE_DIR_KEY = 'natural_scene_dir'
NATURAL_SCENE_KEY = 'natural_scene'
PROBES_KEY = 'probes'
PROBE_LFP_NWB_KEY = 'probe_lfp_nwb'
SESSIONS_KEY = 'sessions'
```

```

SESSION_ANALYSIS_METRICS_KEY = 'session_analysis_metrics'
SESSION_DIR_KEY = 'session_data'
SESSION_NWB_KEY = 'session_nwb'
SUPPRESS_FROM_CHANNELS = ('air_channel_index', 'surface_channel_index', 'name', 'date_
SUPPRESS_FROM_PROBES = ('air_channel_index', 'surface_channel_index', 'date_of_acquisi
SUPPRESS_FROM_SESSION_TABLE = ('has_nwb', 'isi_experiment_id', 'date_of_acquisition')
SUPPRESS_FROM_UNITS = ('air_channel_index', 'surface_channel_index', 'has_nwb', 'lfp_t
TYPEWISE_ANALYSIS_METRICS_KEY = 'typewise_analysis_metrics'
UNITS_KEY = 'units'

```

**add\_manifest\_paths** (*self*, *manifest\_builder*)

Add cache-class specific paths to the manifest. In derived classes, should call super.

**classmethod fixed** (*manifest=None*, *version=None*)

Creates a EcephysProjectCache that refuses to fetch any data - only the existing local cache is accessible. Useful if you want to settle on a fixed dataset for analysis.

#### Parameters

**manifest** [str or Path] full path to existing manifest json

**version** [str] version of manifest file. If this mismatches the version recorded in the file at manifest, an error will be raised.

**classmethod from\_lims** (*lims\_credentials: Union[allensdk.core.authentication.DbCredentials, NoneType] = None*, *scheme: Union[str, NoneType] = None*, *host: Union[str, NoneType] = None*, *asynchronous: bool = True*, *manifest: Union[str, NoneType] = None*, *version: Union[str, NoneType] = None*, *cache: bool = True*, *fetch\_tries: int = 2*)

Create an instance of EcephysProjectCache with an EcephysProjectLimsApi. Retrieves bleeding-edge data stored locally on Allen Institute servers. Only available for use on-site at the Allen Institute or through a vpn. Requires Allen Institute database credentials.

#### Parameters

**lims\_credentials** [DbCredentials] Credentials to access LIMS database. If not provided will attempt to find credentials in environment variables.

**scheme** [str] URI scheme, such as “http”. Defaults to EcephysProjectLimsApi.default value if unspecified. Will not be used unless *host* is also specified.

**host** [str] Web host. Defaults to EcephysProjectLimsApi.default value if unspecified. Will not be used unless *scheme* is also specified.

**asynchronous** [bool] Whether to fetch file asynchronously. Defaults to True.

**manifest** [str or Path] full path at which manifest json will be stored

**version** [str] version of manifest file. If this mismatches the version recorded in the file at manifest, an error will be raised.

**cache: bool** Whether to write to the cache (default=True)

**fetch\_tries** [int] Maximum number of times to attempt a download before giving up and raising an exception. Note that this is total tries, not retries

```
classmethod from_warehouse (scheme: Union[str, NoneType] = None, host: Union[str, NoneType] = None, asynchronous: bool = True, manifest: Union[str, pathlib.Path, NoneType] = None, version: Union[str, NoneType] = None, cache: bool = True, fetch_tries: int = 2)
```

Create an instance of EcephysProjectCache with an EcephysProjectWarehouseApi. Retrieves released data stored in the warehouse.

#### Parameters

**scheme** [str] URI scheme, such as “http”. Defaults to EcephysProjectWarehouse-API.default value if unspecified. Will not be used unless *host* is also specified.

**host** [str] Web host. Defaults to EcephysProjectWarehouseApi.default value if unspecified. Will not be used unless *scheme* is also specified.

**asynchronous** [bool] Whether to fetch file asynchronously. Defaults to True.

**manifest** [str or Path] full path at which manifest json will be stored

**version** [str] version of manifest file. If this mismatches the version recorded in the file at manifest, an error will be raised.

**cache: bool** Whether to write to the cache (default=True)

**fetch\_tries** [int] Maximum number of times to attempt a download before giving up and raising an exception. Note that this is total tries, not retries

```
get_all_ages (self, **session_kwargs)
```

```
get_all_full_genotypes (self, **session_kwargs)
```

```
get_all_session_types (self, **session_kwargs)
```

```
get_all_sexes (self, **session_kwargs)
```

```
get_channels (self, suppress=None)
```

Load (potentially downloading and caching) a table whose rows are individual channels.

```
get_natural_movie_template (self, number)
```

```
get_natural_scene_template (self, number)
```

```
get_probes (self, suppress=None)
```

```
get_session_data (self, session_id: int, filter_by_validity: bool = True, **unit_filter_kwargs)
```

Obtain an EcephysSession object containing detailed data for a single session

```
get_session_table (self, suppress=None) → pandas.core.frame.DataFrame
```

```
get_structure_acronyms (self, **channel_kwargs) → List[str]
```

```
get_unit_analysis_metrics_by_session_type (self, session_type, annotate: bool = True, filter_by_validity: bool = True, **unit_filter_kwargs)
```

Cache and return a table of analysis metrics calculated on each unit from a specified session type. See `get_all_session_types` for a list of session types.

#### Parameters

**session\_type** [str] identifies the session type for which to fetch analysis metrics.

**annotate** [bool, optional] if True, information from the annotated units table will be merged onto the outputs

**filter\_by\_validity** [bool, optional] Filter units used by analysis so that only ‘valid’ units are returned, by default True

**\*\*unit\_filter\_kwargs** : Additional keyword arguments that can be used to filter units (for power users).

#### Returns

**metrics** [pd.DataFrame] Each row corresponds to a single unit, describing a set of analysis metrics calculated on that unit.

```
get_unit_analysis_metrics_for_session(self, session_id, annotate: bool =
                                         True, filter_by_validity: bool = True,
                                         **unit_filter_kwargs)
```

Cache and return a table of analysis metrics calculated on each unit from a specified session. See `get_session_table` for a list of sessions.

#### Parameters

**session\_id** [int] identifies the session from which to fetch analysis metrics.

**annotate** [bool, optional] if True, information from the annotated units table will be merged onto the outputs

**filter\_by\_validity** [bool, optional] Filter units used by analysis so that only ‘valid’ units are returned, by default True

**\*\*unit\_filter\_kwargs** : Additional keyword arguments that can be used to filter units (for power users).

#### Returns

**metrics** [pd.DataFrame] Each row corresponds to a single unit, describing a set of analysis metrics calculated on that unit.

```
get_units(self, suppress: Union[List[str], NoneType] = None, filter_by_validity: bool = True,
           **unit_filter_kwargs) → pandas.core.frame.DataFrame
```

Reports a table consisting of all sorted units across the entire extracellular electrophysiology project.

#### Parameters

**suppress** [Optional[List[str]], optional] A list of dataframe column names to hide, by default None (None will hide dataframe columns specified in: `SUPPRESS_FROM_UNITS`)

**filter\_by\_validity** [bool, optional] Filter units so that only ‘valid’ units are returned, by default True

**\*\*unit\_filter\_kwargs** : Additional keyword arguments that can be used to filter units (for power users).

#### Returns

**pd.DataFrame** A table consisting of sorted units across the entire extracellular electrophysiology project

```
allensdk.brain_observatory.ecephys.ecephys_project_cache.count_owned(this,
                                                                    other,
                                                                    for-
                                                                    eign_key,
                                                                    count_key,
                                                                    in-
                                                                    place=False)
```

```
allensdk.brain_observatory.ecephys.ecephys_project_cache.get_grouped_uniques(this,
                                                                              other,
                                                                              for-
                                                                              eign_key,
                                                                              field_key,
                                                                              unique_key,
                                                                              in-
                                                                              place=False)

allensdk.brain_observatory.ecephys.ecephys_project_cache.read_csv(path)
                                                                → pan-
                                                                das.core.frame.DataFrame

allensdk.brain_observatory.ecephys.ecephys_project_cache.read_metrics_csv(path)

allensdk.brain_observatory.ecephys.ecephys_project_cache.read_movie(path)

allensdk.brain_observatory.ecephys.ecephys_project_cache.read_nwb(path)

allensdk.brain_observatory.ecephys.ecephys_project_cache.read_scene(path)

allensdk.brain_observatory.ecephys.ecephys_project_cache.write_csv(path, df)

allensdk.brain_observatory.ecephys.ecephys_project_cache.write_metrics_csv(path,
                                                                              df)
```

## allensdk.brain\_observatory.ecephys.ecephys\_session module

```
class allensdk.brain_observatory.ecephys.ecephys_session.EcephysSession(api:
                                                                              al-
                                                                              lensdk.brain_observatory.e
                                                                              test:
                                                                              bool
                                                                              =
                                                                              False,
                                                                              **kwargs)
```

Bases: `allensdk.core.lazy_property.lazy_property_mixin.LazyPropertyMixin`

Represents data from a single EcephysSession

### Attributes

**units** [pd.DataFrame] A table whose rows are sorted units (putative neurons) and whose columns are characteristics of those units. Index is:

**unit\_id** [int] Unique integer identifier for this unit.

### Columns are:

**firing\_rate** [float] This unit's firing rate (spikes / s) calculated over the window of that unit's activity (the time from its first detected spike to its last).

**isi\_violations** [float] Estimate of this unit's contamination rate (larger means that more of the spikes assigned to this unit probably originated from other neurons). Calculated as a ratio of the firing rate of the unit over periods where spikes would be isi-violating vs the total firing rate of the unit.

**peak\_channel\_id** [int] Unique integer identifier for this unit's peak channel. A unit's peak channel is the channel on which its peak-to-trough amplitude difference is maximized. This is assessed using the kilosort 2 templates rather than the mean waveforms for a unit.

**snr** [float] Signal to noise ratio for this unit.

**probe\_horizontal\_position** [numeric] The horizontal (short-axis) position of this unit's peak channel in microns.

**probe\_vertical\_position** [numeric] The vertical (long-axis, lower values are closer to the probe base) position of this unit's peak channel in microns.

**probe\_id** [int] Unique integer identifier for this unit's probe.

**probe\_description** [str] Human-readable description carrying miscellaneous information about this unit's probe.

**location** [str] Gross-scale location of this unit's probe.

**spike\_times** [dict] Maps integer unit ids to arrays of spike times (float) for those units.

**running\_speed** [RunningSpeed]

**NamedTuple with two fields**

**timestamps** [numpy.ndarray] Timestamps of running speed data samples

**values** [np.ndarray] Running speed of the experimental subject (in cm / s).

**mean\_waveforms** [dict] Maps integer unit ids to xarray.DataArrays containing mean spike waveforms for that unit.

**stimulus\_presentations** [pd.DataFrame] Table whose rows are stimulus presentations and whose columns are presentation characteristics. A stimulus presentation is the smallest unit of distinct stimulus presentation and lasts for (usually) 1 60hz frame. Since not all parameters are relevant to all stimuli, this table contains many 'null' values. Index is

**stimulus\_presentation\_id** [int] Unique identifier for this stimulus presentation

**Columns are**

**start\_time** [float] Time (s) at which this presentation began

**stop\_time** [float] Time (s) at which this presentation ended

**duration** [float] stop\_time - start\_time (s). Included for convenience.

**stimulus\_name** [str] Identifies the stimulus family (e.g. "drifting\_gratings" or "natural\_movie\_3") used for this presentation. The stimulus family, along with relevant parameter values, provides the information required to reconstruct the stimulus presented during this presentation. The empty string indicates a blank period.

**stimulus\_block** [numeric] A stimulus block is made by sequentially presenting presentations from the same stimulus family. This value is the index of the block which contains this presentation. During a blank period, this is 'null'.

**TF** [float] Temporal frequency, or 'null' when not appropriate.

**SF** [float] Spatial frequency, or 'null' when not appropriate

**Ori** [float] Orientation (in degrees) or 'null' when not appropriate

Contrast : float Pos\_x : float Pos\_y : float Color : numeric Image : numeric Phase : float stimulus\_condition\_id : integer

identifies the session-unique stimulus condition (permutation of parameters) to which this presentation belongs

**stimulus\_conditions** [pd.DataFrame] Each row is a unique permutation (within this session) of stimulus parameters presented during this experiment. Columns are as stimulus presentations, sans start\_time, end\_time, stimulus\_block, and duration.

**inter\_presentation\_intervals** [pd.DataFrame] The elapsed time between each immediately sequential pair of stimulus presentations. This is a dataframe with a two-level multiindex (levels are 'from\_presentation\_id' and 'to\_presentation\_id'). It has a single column, 'interval', which reports the elapsed time between the two presentations in seconds on the experiment's master clock.

**DETAILED\_STIMULUS\_PARAMETERS** = ('colorSpace', 'flipHoriz', 'flipVert', 'depth', 'interage\_in\_days

**channel\_structure\_intervals** (*self*, *channel\_ids*)  
find on a list of channels the intervals of channels inserted into particular structures

#### Parameters

**channel\_ids** [list] A list of channel ids

**structure\_id\_key** [str] use this column for numerically identifying structures

**structure\_label\_key** [str] use this column for human-readable structure identification

#### Returns

**labels** [np.ndarray] for each detected interval, the label associated with that interval

**intervals** [np.ndarray] one element longer than labels. Start and end indices for intervals.

**conditionwise\_spike\_statistics** (*self*, *stimulus\_presentation\_ids=None*, *unit\_ids=None*, *use\_rates=False*)  
Produce summary statistics for each distinct stimulus condition

#### Parameters

**stimulus\_presentation\_ids** [array-like] identifies stimulus presentations from which spikes will be considered

**unit\_ids** [array-like] identifies units whose spikes will be considered

**use\_rates** [bool, optional] If True, use firing rates. If False, use spike counts.

#### Returns

**pd.DataFrame** : Rows are indexed by unit id and stimulus condition id. Values are summary statistics describing spikes emitted by a specific unit across presentations within a specific condition.

**classmethod from\_nwb\_path** (*path*, *nwb\_version=2*, *api\_kwargs=None*, *\*\*kwargs*)

**full\_genotype**

**get\_current\_source\_density** (*self*, *probe\_id*)

Obtain current source density (CSD) of trial-averaged response to a flash stimuli for this probe. See `allensdk.brain_observatory.ecephys.current_source_density` for details of CSD calculation.

CSD is computed with a 1D method (second spatial derivative) without prior spatial smoothing User should apply spatial smoothing of their choice (e.g., Gaussian filter) to the computed CSD

#### Parameters

**probe\_id** [int] identify the probe whose CSD data ought to be loaded

#### Returns



**xr.DataArray** : dimensions are channel (id) and time (seconds, relative to stimulus on-set). Values are current source density assessed on that channel at that time (V/m<sup>2</sup>)

**get\_inter\_presentation\_intervals\_for\_stimulus** (*self*, *stimulus\_names*)

Get a subset of this session's inter-presentation intervals, filtered by stimulus name.

#### Parameters

**stimulus\_names** [array-like of str] The names of stimuli to include in the output.

#### Returns

**pd.DataFrame** : inter-presentation intervals, filtered to the requested stimulus names.

**get\_invalid\_times** (*self*)

Report invalid time intervals with tags describing the scope of invalid data

The tags format: [scope,scope\_id,label]

**scope**: 'EcephysSession': data is invalid across session 'EcephysProbe': data is invalid for a single probe

**label**: 'all\_probes': gain fluctuations on the Neuropixels probe result in missed spikes and LFP saturation events 'stimulus': very long frames (>3x the normal frame length) make any stimulus-locked analysis invalid 'probe#': probe # stopped sending data during this interval (spikes and LFP samples will be missing) 'optotagging': missing optotagging data

#### Returns

**pd.DataFrame** : Rows are invalid intervals, columns are 'start\_time' (s), 'stop\_time' (s), 'tags'

**get\_lfp** (*self*, *probe\_id*, *mask\_invalid\_intervals=True*)

Load an xarray DataArray with LFP data from channels on a single probe

#### Parameters

**probe\_id** [int] identify the probe whose LFP data ought to be loaded

**mask\_invalid\_intervals** [bool] if True (default) will mask data in the invalid intervals with np.nan

#### Returns

**xr.DataArray** : dimensions are channel (id) and time (seconds). Values are sampled LFP data.

### Notes

Unlike many other data access methods on this class. This one does not cache the loaded data in memory due to the large size of the LFP data.

**get\_parameter\_values\_for\_stimulus** (*self*, *stimulus\_name*, *drop\_nulls=True*)

For each stimulus parameter, report the unique values taken on by that parameter while a named stimulus was presented.

#### Parameters

**stimulus\_name** [str] filter to presentations of this stimulus

#### Returns

**dict** : maps parameters (column names) to their unique values.

**get\_pupil\_data** (*self*, *suppress\_pupil\_data*: *bool = True*) → *pandas.core.frame.DataFrame*  
Return a dataframe with eye tracking data

#### Parameters

**suppress\_pupil\_data** [*bool*, optional] Whether or not to suppress eye gaze mapping data in output dataframe, by default *True*.

#### Returns

**pd.DataFrame**

**Contains columns for eye, pupil and cr ellipse fits:** \*\_center\_x \*\_center\_y  
\*\_height \*\_width \*\_phi

May also contain raw/filtered columns for gaze mapping if *suppress\_pupil\_data* is set to *False*:

\*\_eye\_area \*\_pupil\_area \*\_screen\_coordinates\_x\_cm  
\*\_screen\_coordinates\_y\_cm \*\_screen\_coordinates\_spherical\_x\_deg  
\*\_screen\_coorindates\_spherical\_y\_deg

**get\_stimulus\_epochs** (*self*, *duration\_thresholds*=*None*)

Reports continuous periods of time during which a single kind of stimulus was presented flipVert

**duration\_thresholds** [*dict*, optional]

**keys are stimulus names, values are floating point durations in seconds. All epochs with**

- a given stimulus name
- a duration shorter than the associated threshold

will be removed from the results

**get\_stimulus\_parameter\_values** (*self*, *stimulus\_presentation\_ids*=*None*, *drop\_nulls*=*True*)

For each stimulus parameter, report the unique values taken on by that parameter throughout the course of the session.

#### Parameters

**stimulus\_presentation\_ids** [*array-like*, optional] If provided, only parameter values from these stimulus presentations will be considered.

#### Returns

**dict** : maps parameters (column names) to their unique values.

**get\_stimulus\_table** (*self*, *stimulus\_names*=*None*, *include\_detailed\_parameters*=*False*, *include\_unused\_parameters*=*False*)

Get a subset of stimulus presentations by name, with irrelevant parameters filtered off

#### Parameters

**stimulus\_names** [*array-like of str*] The names of stimuli to include in the output.

#### Returns

**pd.DataFrame** : Rows are filtered presentations, columns are the relevant subset of stimulus parameters

**metadata**

**num\_channels**

**num\_probes**

**num\_stimulus\_presentations**

**num\_units**

**presentationwise\_spike\_counts** (*self*, *bin\_edges*, *stimulus\_presentation\_ids*, *unit\_ids*, *binarize=False*, *dtype=None*, *large\_bin\_size\_threshold=0.001*, *time\_domain\_callback=None*)

Build an array of spike counts surrounding stimulus onset per unit and stimulus frame.

**bin\_edges** [numpy.ndarray] Spikes will be counted into the bins defined by these edges. Values are in seconds, relative to stimulus onset.

**stimulus\_presentation\_ids** [array-like] Filter to these stimulus presentations

**unit\_ids** [array-like] Filter to these units

**binarize** [bool, optional] If true, all counts greater than 0 will be treated as 1. This results in lower storage overhead, but is only reasonable if bin sizes are fine ( $\leq 1$  millisecond).

**large\_bin\_size\_threshold** [float, optional] If binarize is True and the largest bin width is greater than this value, a warning will be emitted.

**time\_domain\_callback** [callable, optional] The time domain is a numpy array whose values are trial-aligned bin edges (each row is aligned to a different trial). This optional function will be applied to the time domain before counting spikes.

#### Returns

**xarray.DataArray** : Data array whose dimensions are stimulus presentation, unit, and time bin and whose values are spike counts.

**presentationwise\_spike\_times** (*self*, *stimulus\_presentation\_ids=None*, *unit\_ids=None*)

Produce a table associating spike times with units and stimulus presentations

#### Parameters

**stimulus\_presentation\_ids** [array-like] Filter to these stimulus presentations

**unit\_ids** [array-like] Filter to these units

#### Returns

**pandas.DataFrame** :

**Index is**

**spike\_time** [float] On the session's master clock.

**Columns are**

**stimulus\_presentation\_id** [int] The stimulus presentation on which this spike occurred.

**unit\_id** [int] The unit that emitted this spike.

**rig\_equipment\_name**

**rig\_geometry\_data**

**session\_type**

**sex**

**specimen\_name**

**spike\_times**

**stimulus\_conditions**  
**stimulus\_names**  
**stimulus\_presentations**  
**structure\_acronyms**  
**structurewise\_unit\_counts**  
**units**

`allensdk.brain_observatory.ecephys.ecephys_session.array_intervals(array)`  
find interval bounds (bounding consecutive identical values) in an array

**Parameters**

**array** [np.ndarray]

**Returns**

**np.ndarray** : start and end indices of detected intervals (one longer than the number of intervals)

`allensdk.brain_observatory.ecephys.ecephys_session.build_spike_histogram(time_domain,  
spike_times,  
unit_ids,  
dtype=None,  
bi-  
na-  
rize=False)`  
`allensdk.brain_observatory.ecephys.ecephys_session.build_time_window_domain(bin_edges,  
off-  
sets,  
call-  
back=None)`  
`allensdk.brain_observatory.ecephys.ecephys_session.coerce_scalar(value,  
message,  
warn=False)`  
`allensdk.brain_observatory.ecephys.ecephys_session.is_distinct_from(left,  
right)`  
`allensdk.brain_observatory.ecephys.ecephys_session.nan_intervals(array,  
nan_like=['null'])`  
find interval bounds (bounding consecutive identical values) in an array, which may contain nans

**Parameters**

**array** [np.ndarray]

**Returns**

**np.ndarray** : start and end indices of detected intervals (one longer than the number of intervals)

`allensdk.brain_observatory.ecephys.ecephys_session.removed_unused_stimulus_presentation_co`

**allensdk.brain\_observatory.ecephys.stimulus\_sync module**

```
allensdk.brain_observatory.ecephys.stimulus_sync.allocate_by_vsync(vs_diff,
                                                                    index,
                                                                    starts,
                                                                    ends,
                                                                    frame_duration,
                                                                    irregularity, cycle)

allensdk.brain_observatory.ecephys.stimulus_sync.assign_to_last(index,
                                                                    starts, ends,
                                                                    frame_duration,
                                                                    irregularity,
                                                                    cycle)

allensdk.brain_observatory.ecephys.stimulus_sync.compute_frame_times(photodiode_times,
                                                                    frame_duration,
                                                                    num_frames,
                                                                    cycle,
                                                                    irregular_interval_policy=<function
                                                                    assign_to_last
                                                                    at
                                                                    0x7fdbccc48268>)

allensdk.brain_observatory.ecephys.stimulus_sync.correct_on_off_effects(pd_times)
```

**Notes**

This cannot (without additional info) determine whether an assymmetric offset is odd-long or even-long.

```
allensdk.brain_observatory.ecephys.stimulus_sync.estimate_frame_duration(pd_times,
                                                                    cy-
                                                                    cle=60)

allensdk.brain_observatory.ecephys.stimulus_sync.fix_unexpected_edges(pd_times,
                                                                    ndevs=10,
                                                                    cy-
                                                                    cle=60,
                                                                    max_frame_offset=4)

allensdk.brain_observatory.ecephys.stimulus_sync.flag_unexpected_edges(pd_times,
                                                                    ndevs=10)

allensdk.brain_observatory.ecephys.stimulus_sync.trim_border_pulses(pd_times,
                                                                    vs_times,
                                                                    frame_interval=0.016666666666666666,
                                                                    num_frames=5)

allensdk.brain_observatory.ecephys.stimulus_sync.trimmed_stats(data, pc-
                                                                    tiles=(10, 90))
```

## Module contents

```
allensdk.brain_observatory.ecephys.get_unit_filter_value(key, pop=True, re-  
place_none=True,  
**source)
```

## allensdk.brain\_observatory.extract\_running\_speed package

### Module contents

## allensdk.brain\_observatory.gaze\_mapping package

### Module contents

## allensdk.brain\_observatory.nwb package

### Submodules

## allensdk.brain\_observatory.nwb.metadata module

```
allensdk.brain_observatory.nwb.metadata.create_LabMetaData_extension_from_schemas(schema_list,  
pre-  
fix)
```

```
allensdk.brain_observatory.nwb.metadata.extract_from_schema(schema)
```

```
allensdk.brain_observatory.nwb.metadata.load_LabMetaData_extension(schema,  
prefix)
```

## allensdk.brain\_observatory.nwb.nwb\_api module

```
class allensdk.brain_observatory.nwb.nwb_api.NwbApi(path, **kwargs)  
    Bases: object  
    classmethod from_nwbfile(nwbfile, **kwargs)  
    classmethod from_path(path, **kwargs)  
    get_image(self, name, module, image_api=None) → SimpleITK.SimpleITK.Image  
    get_invalid_times(self) → pandas.core.frame.DataFrame  
    get_running_speed(self) → allensdk.brain_observatory.running_speed.RunningSpeed  
    get_stimulus_presentations(self) → pandas.core.frame.DataFrame  
    nwbfile  
    path
```

**allensdk.brain\_observatory.nwb.schemas module**

```
class allensdk.brain_observatory.nwb.schemas.RunningSpeedPathsSchema (only=None,
                                                                    ex-
                                                                    clude=(),
                                                                    many=False,
                                                                    con-
                                                                    text=None,
                                                                    load_only=(),
                                                                    dump_only=(),
                                                                    par-
                                                                    tial=False,
                                                                    un-
                                                                    known=None)

    Bases: allensdk.brain_observatory.argschema_utilities.RaisingSchema

    opts = <marshmallow.schema.SchemaOpts object>
```

**Module contents**

```
allensdk.brain_observatory.nwb.add_average_image (nwbfile,    average_image,    im-
                                                    age_api=None)

allensdk.brain_observatory.nwb.add_cell_specimen_table (nwbfile,
                                                         cell_specimen_table)

allensdk.brain_observatory.nwb.add_corrected_fluorescence_traces (nwbfile, cor-
                                                                    rected_fluorescence_traces)

allensdk.brain_observatory.nwb.add_dff_traces (nwbfile, dff_traces, ophys_timestamps)

allensdk.brain_observatory.nwb.add_eye_gaze_data_interfaces (pynwb_container:
                                                                pynwb.core.NWBContainer,
                                                                pupil_areas: pan-
                                                                das.core.series.Series,
                                                                eye_areas: pan-
                                                                das.core.series.Series,
                                                                screen_coordinates:
                                                                pan-
                                                                das.core.frame.DataFrame,
                                                                screen_coordinates_spherical:
                                                                pan-
                                                                das.core.frame.DataFrame,
                                                                synced_timestamps:
                                                                pan-
                                                                das.core.series.Series)
                                                                →
                                                                pynwb.core.NWBContainer

allensdk.brain_observatory.nwb.add_eye_gaze_mapping_data_to_nwbfile (nwbfile:
                                                                    pynwb.file.NWBFile,
                                                                    eye_gaze_data:
                                                                    dict) →
                                                                    pynwb.file.NWBFile
```

```
allensdk.brain_observatory.nwb.add_eye_tracking_ellipse_fit_data_to_nwbfile(nwbfile:  
                                                                           pynwb.file.NWBFile,  
                                                                           eye_dlc_tracking_data=  
dict,  
synced_timestamps=  
pandas.core.series.Series  
→  
pynwb.file.NWBFile
```

```
allensdk.brain_observatory.nwb.add_image(nwbfile, image_data, image_name, module_name,  
                                         module_description, image_api=None)
```

```
allensdk.brain_observatory.nwb.add_invalid_times(nwbfile, epochs)
```

Write invalid times to nwbfile if epochs are not empty  
Parameters ———— *nwbfile*: pynwb.NWBFile  
*epochs*: list of dicts

records of invalid epochs

#### Returns

pynwb.NWBFile

```
allensdk.brain_observatory.nwb.add_licks(nwbfile, licks)
```

```
allensdk.brain_observatory.nwb.add_max_projection(nwbfile, max_projection, image_api=None)
```

```
allensdk.brain_observatory.nwb.add_metadata(nwbfile, metadata)
```

```
allensdk.brain_observatory.nwb.add_motion_correction(nwbfile, motion_correction)
```

```
allensdk.brain_observatory.nwb.add_rewards(nwbfile, rewards_df)
```

```
allensdk.brain_observatory.nwb.add_running_data_df_to_nwbfile(nwbfile, running_data_df,  
                                                                unit_dict, index_key='timestamps')
```

Adds running speed data to an NWBFile as timeseries in acquisition and processing

#### Parameters

**nwbfile** [pynwb.NWBFile] File to which running speeds will be written

**running\_speed** [pandas.DataFrame] Contains 'speed' and 'times', 'v\_in', 'vsig', 'dx'

**unit** [str, optional] SI units of running speed values

#### Returns

**nwbfile** [pynwb.NWBFile]

```
allensdk.brain_observatory.nwb.add_running_speed_to_nwbfile(nwbfile, running_speed,  
                                                            name='speed',  
                                                            unit='cm/s')
```

Adds running speed data to an NWBFile as a timeseries in acquisition

#### Parameters

**nwbfile** [pynwb.NWBFile] File to which running speeds will be written

**running\_speed** [RunningSpeed] Contains attributes 'values' and 'timestamps'



**name** [str, optional] used as name of timeseries object

**unit** [str, optional] SI units of running speed values

### Returns

**nwbfile** [pynwb.NWBFile]

```
allensdk.brain_observatory.nwb.add_segmentation_mask_image(nwbfile, segmentation_mask_image, image_api=None)
```

```
allensdk.brain_observatory.nwb.add_stimulus_index(nwbfile, stimulus_index, nwb_template)
```

```
allensdk.brain_observatory.nwb.add_stimulus_presentations(nwbfile, stimulus_table, tag='stimulus_epoch')
```

Adds a stimulus table (defining stimulus characteristics for each time point in a session) to an nwbfile as epochs.

### Parameters

**nwbfile** [pynwb.NWBFile]

**stimulus\_table: pd.DataFrame** Each row corresponds to an epoch of time. Columns define the epoch (start and stop time) and its characteristics. Nans will be replaced with the empty string. Required columns are:

start\_time :: the time at which this epoch started stop\_time :: the time at which this epoch ended

**tag** [str, optional] Each epoch in an nwb file has one or more tags. This string will be applied as a tag to all epochs created here

### Returns

**nwbfile** [pynwb.NWBFile]

```
allensdk.brain_observatory.nwb.add_stimulus_template(nwbfile, image_data, name)
```

```
allensdk.brain_observatory.nwb.add_stimulus_timestamps(nwbfile, stimulus_timestamps, module_name='stimulus')
```

```
allensdk.brain_observatory.nwb.add_task_parameters(nwbfile, task_parameters)
```

```
allensdk.brain_observatory.nwb.add_trials(nwbfile, trials, description_dict={})
```

```
allensdk.brain_observatory.nwb.create_eye_gaze_mapping_dataframe(eye_gaze_data: dict) → pandas.core.frame.DataFrame
```

```
allensdk.brain_observatory.nwb.create_eye_tracking_nwb_processing_module(eye_dlc_tracking_data: dict, synced_timestamps: pandas.core.series.Series) → pynwb.base.ProcessingModule
```

```
allensdk.brain_observatory.nwb.create_gaze_mapping_nwb_processing_modules(eye_gaze_data: dict)
```

```
allensdk.brain_observatory.nwb.eye_tracking_data_is_valid(eye_dlc_tracking_data:  
                                                         dict,  
                                                         synced_timestamps:  
                                                         pan-  
                                                         das.core.series.Series)  
                                                         → bool
```

```
allensdk.brain_observatory.nwb.read_eye_dlc_tracking_ellipses(input_path: path-  
                                                             lib.Path) → dict
```

Reads eye tracking ellipse fit data from an h5 file.

**Args:** *input\_path* (Path): Path to eye tracking ellipse fit h5 file

**Returns:**

**dict:** Loaded h5 data. Each 'params' field contains dataframes with] ellipse fit parameters. Dataframes contain 5 columns each consisting of: "center\_x", "center\_y", "height", "phi", "width"

```
allensdk.brain_observatory.nwb.read_eye_gaze_mappings(input_path: pathlib.Path) →  
                                                         dict
```

Reads eye gaze mapping data from an h5 file.

**Args:**

**input\_path (Path):** Path to eye gaze mapping h5 data file produced by 'allensdk.brain\_observatory.gaze\_mapping' module.

**Returns:**

**dict:** Loaded h5 data. \*\_eye\_areas: Area of eye (in pixels^2) over time \*\_pupil\_areas: Area of pupil (in pixels^2) over time \*\_screen\_coordinates: y, x screen coordinates (in cm) over time \*\_screen\_coordinates\_spherical: y, x screen coordinates (in deg) over time synced\_frame\_timestamps: synced timestamps for video frames (in sec)

```
allensdk.brain_observatory.nwb.setup_table_for_epochs(table, timeseries, tag)
```

```
allensdk.brain_observatory.nwb.setup_table_for_invalid_times(invalid_epochs)
```

Create table with invalid times if invalid\_epochs are present

**Parameters**

**invalid\_epochs:** list of dicts of invalid epoch records

**Returns**

pd.DataFrame of invalid times if epochs are not empty, otherwise return None

**allensdk.brain\_observatory.ophys package**

**Subpackages**

**allensdk.brain\_observatory.ophys.trace\_extraction package**

**Module contents**

**Module contents**

**allensdk.brain\_observatory.receptive\_field\_analysis package**

## Submodules

### allensdk.brain\_observatory.receptive\_field\_analysis.chisquarperf module

`allensdk.brain_observatory.receptive_field_analysis.chisquarperf.NLL_to_pvalue` (*NLLs*, *log\_base=10.0*)

`allensdk.brain_observatory.receptive_field_analysis.chisquarperf.build_trial_matrix` (*LSN\_template*, *num\_trials*, *on\_off\_lum*, *0*)

Construct indicator arrays for on/off pixels across trials.

#### Parameters

**LSN\_template** [np.ndarray] Dimensions are (nTrials, nYPixels, nXPixels). Luminance values per pixel and trial. The size of the first dimension may be larger than the `num_trials` argument (in which case only the first `num_trials` slices will be used) but may not be smaller.

**num\_trials** [int] The number of trials (left-justified) to build indicators for.

**on\_off\_luminance** [array-like, optional] The zeroth element is the luminance value of a pixel when on, the first when off. Defaults are [255, 0].

#### Returns

**trial\_mat** [np.ndarray] Dimensions are (nYPixels, nXPixels, {on, off}, nTrials). Boolean values indicate that a pixel was on/off on a particular trial.

`allensdk.brain_observatory.receptive_field_analysis.chisquarperf.chi_square_binary` (*events*, *LSN\_template*)

`allensdk.brain_observatory.receptive_field_analysis.chisquarperf.chi_square_within_mask` (*exclusion\_mask*, *events\_per\_pixel*, *trials\_per\_pixel*)

Determine if cells respond preferentially to on/off pixels in a mask using a chi2 test.

#### Parameters

**exclusion\_mask** [np.ndarray] Dimensions are (nYPixels, nXPixels, {on, off}). Integer indicator for INCLUSION (!) of a pixel within the testing region.

**events\_per\_pixel** [np.ndarray] Dimensions are (nCells, nYPixels, nXPixels, {on, off}). Integer values are response counts by cell to on/off luminance at each pixel.

**trials\_per\_pixel** [np.ndarray] Dimensions are (nYPixels, nXPixels, {on, off}). Integer values are counts of trials where a pixel is on/off.

#### Returns

**p\_vals** [np.ndarray] One-dimensional, of length nCells. Float values are p-values for the hypothesis that a given cell has a receptive field within the exclusion mask.

**chi** [np.ndarray] Dimensions are (nCells, nYPixels, nXPixels, {on, off}). Values (float) are squared residual event counts divided by expected event counts.

`allensdk.brain_observatory.receptive_field_analysis.chisquarperf.deinterpolate_RF` (*rf\_map*, *x\_pnts*, *y\_pnts*, *deg\_per\_pnt*)

Downsample an image

**Parameters**

**rf\_map** [np.ndarray] Input image

**x\_pnts** [np.ndarray] Count of sample points along the first (column) axis

**y\_pnts** [np.ndarray] Count of sample points along the zeroth (row) axis

**deg\_per\_pnt** [numeric] scale factor

**Returns**

**sampled\_yx** [np.ndarray] Downsampled image

`allensdk.brain_observatory.receptive_field_analysis.chisquarerf.get_disc_masks` (*LSN\_template*,  
*radius=3*,  
*on\_luminance=255*,  
*off\_luminance=0*)

Obtain an indicator mask surrounding each pixel. The mask is a square, excluding pixels which are coactive on any trial with the main pixel.

**Parameters**

**LSN\_template** [np.ndarray] Dimensions are (nTrials, nYPixels, nXPixels). Luminance values per pixel and trial.

**radius** [int] The base mask will be a box whose sides are  $2 * \text{radius} + 1$  in length.

**on\_luminance** [int, optional] The value of the luminance for on trials. Default is 255

**off\_luminance** [int, optional] The value of the luminance for off trials. Default is 0

**Returns**

**masks** [np.ndarray] Dimensions are (nYPixels, nXPixels, nYPixels, nXPixels). The first 2 dimensions describe the pixel from which the mask was computed. The last 2 serve as the dimensions of the mask images themselves. Masks are binary arrays of type float, with 1 indicating inside, 0 outside.

`allensdk.brain_observatory.receptive_field_analysis.chisquarerf.get_events_per_pixel` (*responses*,  
*trial\_matrix*)

Obtain a matrix linking cellular responses to pixel activity.

**Parameters**

**responses\_np** [np.ndarray] Dimensions are (nTrials, nCells). Boolean values indicate presence/absence of a response on a given trial.

**trial\_matrix** [np.ndarray] Dimensions are (nYPixels, nXPixels, {on, off}, nTrials). Boolean values indicate that a pixel was on/off on a particular trial.

**Returns**

**events\_per\_pixel** [np.ndarray] Dimensions are (nCells, nYPixels, nXPixels, {on, off}). Values for each cell, pixel, and on/off state are the sum of events for that cell across all trials where the pixel was in the on/off state.

`allensdk.brain_observatory.receptive_field_analysis.chisquarerf.get_expected_events_by_pixel`

Calculate expected number of events per pixel

**Parameters**

**exclusion\_mask** [np.ndarray] Dimensions are (nYPixels, nXPixels, {on, off}). Integer indicator for INCLUSION (!) of a pixel within the testing region.

**events\_per\_pixel** [np.ndarray] Dimensions are (nCells, nYPixels, nXPixels, {on, off}). Integer values are response counts by cell to on/off luminance at each pixel.

**trials\_per\_pixel** [np.ndarray] Dimensions are (nYPixels, nXPixels, {on, off}). Integer values are counts of trials where a pixel is on/off.

### Returns

**np.ndarray** : Dimensions (nCells, nYPixels, nXPixels, {on, off}). Float values are pixelwise counts of events expected if events are evenly distributed in mask across trials.

```
allensdk.brain_observatory.receptive_field_analysis.chisquarperf.get_peak_significance(chi_sq,
                                                                                   LSN_t,
                                                                                   al-
                                                                                   pha=0)
```

allensdk.brain\_observatory.receptive\_field\_analysis.chisquarperf.interpolate\_RF(*rf\_map*,  
*deg\_per\_pnt*)

Upsample an image

### Parameters

**rf\_map** [np.ndarray] Input image

**deg\_per\_pnt** [numeric] scale factor

### Returns

**interpolated** [np.ndarray] Upsampled image

```
allensdk.brain_observatory.receptive_field_analysis.chisquarperf.locate_median(y,
                                                                                   x)
```

```
allensdk.brain_observatory.receptive_field_analysis.chisquarperf.pvalue_to_NLL(p_values,
                                                                                   max_NLL=10.0)
```

```
allensdk.brain_observatory.receptive_field_analysis.chisquarperf.smooth_STA(STA,
                                                                                   gauss_std=0.75,
                                                                                   to-
                                                                                   tal_degrees=64)
```

Smooth an image by convolution with a gaussian kernel

### Parameters

**STA** [np.ndarray] Input image

**gauss\_std** [numeric, optional] Standard deviation of the gaussian kernel. Will be applied to the upsampled image, so units are visual degrees. Default is 0.75

**total\_degrees** [int, optional] Size in visual degrees of the input image along its zeroth (row) axis. Used to set the scale factor for up/downsampling.

### Returns

**STA\_smoothed** [np.ndarray] Smoothed image

### allensdk.brain\_observatory.receptive\_field\_analysis.eventdetection module

`allensdk.brain_observatory.receptive_field_analysis.eventdetection.detect_events` (*data*,  
*cell\_index*,  
*stimulus*,  
*u-*  
*lus*,  
*de-*  
*bug\_plots=Fa*

### allensdk.brain\_observatory.receptive\_field\_analysis.fit\_parameters module

`allensdk.brain_observatory.receptive_field_analysis.fit_parameters.add_to_fit_parameters_d`  
`allensdk.brain_observatory.receptive_field_analysis.fit_parameters.compute_distance` (*center\_on*,  
*cen-*  
*ter\_off*)  
`allensdk.brain_observatory.receptive_field_analysis.fit_parameters.compute_overlap` (*data\_fitted*,  
*data\_fitted*,  
`allensdk.brain_observatory.receptive_field_analysis.fit_parameters.get_gaussian_fit_single`

### allensdk.brain\_observatory.receptive\_field\_analysis.fitgaussian2D module

**exception** `allensdk.brain_observatory.receptive_field_analysis.fitgaussian2D.GaussianFitError`  
Bases: `RuntimeError`

`allensdk.brain_observatory.receptive_field_analysis.fitgaussian2D.fitgaussian2D` (*data*)  
Fit a 2D gaussian to an image

#### Parameters

**data** [np.ndarray] input image

#### Returns

**p2** [list] height row mean column mean row standard deviation column standard deviation  
rotation

#### Notes

see `gaussian2D` for details about output values

`allensdk.brain_observatory.receptive_field_analysis.fitgaussian2D.gaussian2D` (*height*,  
*cen-*  
*ter\_x*,  
*cen-*  
*ter\_y*,  
*width\_x*,  
*width\_y*,  
*ro-*  
*ta-*  
*tion*)

Build a function which evaluates a scaled 2d gaussian pdf

#### Parameters

**height** [float] scale factor  
**center\_x** [float] first coordinate of mean  
**center\_y** [float] second coordinate of mean  
**width\_x** [float] standard deviation along x axis  
**width\_y** [float] standard deviation along y axis  
**rotation** [float] degrees clockwise by which to rotate the gaussian

#### Returns

**rotgauss: fn** parameters are x and y positions (row/column semantics are set by your inputs to this function). Return value is the scaled gaussian pdf evaluated at the argued point.

`allensdk.brain_observatory.receptive_field_analysis.fitgaussian2D.moments2(data)`  
 Treating input image data as an independent multivariate gaussian, estimate mean and standard deviations

#### Parameters

**data** [np.ndarray] 2d numpy array.

#### Returns

**height** [float] The maximum observed value in the data  
**y** [float] Mean row index  
**x** [float] Mean column index  
**width\_y** [float] The standard deviation along the mean row  
**width\_x** [float] The standard deviation along the mean column  
**None** : This function returns an instance of None.

#### Notes

uses original method from website for finding center

### allensdk.brain\_observatory.receptive\_field\_analysis.postprocessing module

`allensdk.brain_observatory.receptive_field_analysis.postprocessing.get_gaussian_fit(rf)`  
`allensdk.brain_observatory.receptive_field_analysis.postprocessing.run_postprocessing(data, rf)`

### allensdk.brain\_observatory.receptive\_field\_analysis.receptive\_field module

`allensdk.brain_observatory.receptive_field_analysis.receptive_field.compute_receptive_field`

```
allensdk.brain_observatory.receptive_field_analysis.receptive_field.compute_receptive_field
```

```
allensdk.brain_observatory.receptive_field_analysis.receptive_field.events_to_pvalues_no_fmri
```

```
allensdk.brain_observatory.receptive_field_analysis.receptive_field.get_attribute_dict(rf)
```

```
allensdk.brain_observatory.receptive_field_analysis.receptive_field.print_summary(rf)
```

```
allensdk.brain_observatory.receptive_field_analysis.receptive_field.read_h5_group(g)
```

```
allensdk.brain_observatory.receptive_field_analysis.receptive_field.read_receptive_field_fmri
```

```
allensdk.brain_observatory.receptive_field_analysis.receptive_field.write_receptive_field_fmri
```

### **allensdk.brain\_observatory.receptive\_field\_analysis.tools module**

```
allensdk.brain_observatory.receptive_field_analysis.tools.dict_generator(indict,  
                                                                           pre=None)
```

```
allensdk.brain_observatory.receptive_field_analysis.tools.list_of_dicts_to_dict_of_lists(list_of_dicts)
```

```
allensdk.brain_observatory.receptive_field_analysis.tools.read_h5_group(g)
```

### **allensdk.brain\_observatory.receptive\_field\_analysis.utilities module**

```
allensdk.brain_observatory.receptive_field_analysis.utilities.convolve(img,  
                                sigma=4)
```

2D Gaussian convolution

```
allensdk.brain_observatory.receptive_field_analysis.utilities.get_A(data,  
                             stimu-  
                             lus)
```

```
allensdk.brain_observatory.receptive_field_analysis.utilities.get_A_blur(data,  
                                  stimu-  
                                  u-  
                                  lus)
```

```
allensdk.brain_observatory.receptive_field_analysis.utilities.get_attribute_dict(rf)
```

```
allensdk.brain_observatory.receptive_field_analysis.utilities.get_components(receptive_field_data)
```



```
allensdk.brain_observatory.receptive_field_analysis.utilities.get_shuffle_matrix(data,
                                                                              event_vector,
                                                                              A,
                                                                              num-
                                                                              ber_of_shuffle
                                                                              re-
                                                                              sponse_detect
```

```
allensdk.brain_observatory.receptive_field_analysis.utilities.get_sparse_noise_epoch_mask_
```

```
allensdk.brain_observatory.receptive_field_analysis.utilities.smooth(x, win-
                                                                    dow_len=11,
                                                                    win-
                                                                    dow='hanning',
                                                                    mode='valid')
```

smooth the data using a window with requested size.

This method is based on the convolution of a scaled window with the signal. The signal is prepared by introducing reflected copies of the signal (with the window size) in both ends so that transient parts are minimized in the beginning and end part of the output signal.

**input:** x: the input signal window\_len: the dimension of the smoothing window; should be an odd integer  
window: the type of window from 'flat', 'hanning', 'hamming', 'bartlett', 'blackman'

flat window will produce a moving average smoothing.

**output:** the smoothed signal

example:

```
t=linspace(-2,2,0.1) x=sin(t)+randn(len(t))*0.1 y=smooth(x)
```

see also:

numpy.hanning, numpy.hamming, numpy.bartlett, numpy.blackman, numpy.convolve scipy.signal.lfilter

TODO: the window parameter could be the window itself if an array instead of a string NOTE: length(output) != length(input), to correct this: return y[(window\_len/2-1):-(window\_len/2)] instead of just y.

```
allensdk.brain_observatory.receptive_field_analysis.utilities.upsample_image_to_degrees(img
```

## allensdk.brain\_observatory.receptive\_field\_analysis.visualization module

```
allensdk.brain_observatory.receptive_field_analysis.visualization.plot_chi_square_summary(a
a
a
```

```
allensdk.brain_observatory.receptive_field_analysis.visualization.plot_ellipses(gaussian_fit_di
ax=None,
show=True,
close=True,
save_file_name
color='b')
```

Example Usage: oeid, cell\_index, stimulus = 512176430, 12, 'locally\_sparse\_noise' brain\_observatory\_cache = BrainObservatoryCache() data\_set = brain\_observatory\_cache.get\_ophys\_experiment\_data(oeid) lsn = LocallySparseNoise(data\_set, stimulus) result = compute\_receptive\_field\_with\_postprocessing(data\_set,

```
cell_index, stimulus, alpha=.05, number_of_shuffles=5000) plot_ellipses(result['off']['gaussian_fit'],
color='r')

allensdk.brain_observatory.receptive_field_analysis.visualization.plot_fields(on_data,
off_data,
on_axes,
off_axes,
cbar_axes=None,
clim=None,
cmap='magma')

allensdk.brain_observatory.receptive_field_analysis.visualization.plot_gaussian_fit(rf_data,
ax_on,
ax_off,
ax_cbar=None,
cmap='magma')

allensdk.brain_observatory.receptive_field_analysis.visualization.plot_mask(rf_data,
ax_on,
ax_off,
ax_cbar=None,
cmap='magma')

allensdk.brain_observatory.receptive_field_analysis.visualization.plot_msr_summary(lsn,
cell_index,
ax_on,
ax_off,
ax_cbar=None,
cmap=None)

allensdk.brain_observatory.receptive_field_analysis.visualization.plot_p_values(rf_data,
ax_on,
ax_off,
ax_cbar=None,
cmap='magma')

allensdk.brain_observatory.receptive_field_analysis.visualization.plot_receptive_field_data(rf_data,
ax_on,
ax_off,
ax_cbar=None,
cmap='magma')

allensdk.brain_observatory.receptive_field_analysis.visualization.plot_rts_blur_summary(rf_data,
ax_on,
ax_off,
ax_cbar=None,
cmap='magma')

allensdk.brain_observatory.receptive_field_analysis.visualization.plot_rts_summary(rf_data,
ax_on,
ax_off,
ax_cbar=None,
cmap='magma')

allensdk.brain_observatory.receptive_field_analysis.visualization.pvalue_to_NLL(p_values,
max_NLL=10.0)
```

## Module contents

### allensdk.brain\_observatory.sync\_utilities package

## Module contents

```
allensdk.brain_observatory.sync_utilities.get_synchronized_frame_times (session_sync_file:
                                                                    path-
                                                                    lib.Path,
                                                                    sync_line_label_keys:
                                                                    Tu-
                                                                    ple[str,
                                                                    ...])
                                                                    →
                                                                    pan-
                                                                    das.core.series.Series
```

Get experimental frame times from an experiment session sync file.

### Parameters

**session\_sync\_file** [Path] Path to an ephys session sync file. The sync file contains rising/falling edges from a daq system which indicates when certain events occur (so they can be related to each other).

**sync\_line\_label\_keys** [Tuple[str, ...]] Line label keys to get times for. See class attributes of `allensdk.brain_observatory.sync_dataset.Dataset` for a listing of possible keys.

### Returns

**pd.Series** An array of times when frames for the eye tracking camera were acquired.

```
allensdk.brain_observatory.sync_utilities.trim_discontiguous_times (times,
                                                                    thresh-
                                                                    old=100)
```

### allensdk.brain\_observatory.visualization package

## Module contents

```
allensdk.brain_observatory.visualization.plot_running_speed (timestamps,    val-
                                                                    ues, start_index=0,
                                                                    stop_index=None,
                                                                    step=1,          yla-
                                                                    bel='running speed
                                                                    (cm/s)', xlabel='time
                                                                    (s)', title=None)
```

Make a simple plot of a running speed trace

### Parameters

**timestamps** [numpy.ndarray] Times at which running speed samples were collected

**values** [numpy.ndarray] Running speed values (by default: linear cm / s with negative values indicating backwards movement)

## Submodules

### allensdk.brain\_observatory.argschema\_utilities module

**class** allensdk.brain\_observatory.argschema\_utilities.**ArgSchemaParserPlus** (\*args,  
\*\*kwargs)

Bases: argschema.argschema\_parser.ArgSchemaParser

**class** allensdk.brain\_observatory.argschema\_utilities.**InputFile** (default=<marshmallow.missing>,  
attribute=None,  
data\_key=None,  
error=None,  
validate=None,  
required=False,  
allow\_none=None,  
load\_only=False,  
dump\_only=False,  
missing=<marshmallow.missing>,  
error\_messages=None,  
\*\*metadata)

Bases: marshmallow.fields.String

A marshmallow String field subclass which deserializes json str fields that represent a desired input path to pathlib.Path. Also performs read access checking.

**class** allensdk.brain\_observatory.argschema\_utilities.**OutputFile** (default=<marshmallow.missing>,  
attribute=None,  
data\_key=None,  
error=None,  
validate=None,  
required=False,  
allow\_none=None,  
load\_only=False,  
dump\_only=False,  
missing=<marshmallow.missing>,  
error\_messages=None,  
\*\*metadata)

Bases: marshmallow.fields.String

A marshmallow String field subclass which deserializes json str fields that represent a desired output file path to a pathlib.Path. Also performs write access checking.

```

class allensdk.brain_observatory.argschema_utilities.RaisingSchema (only=None,
                                                                    ex-
                                                                    clude=(),
                                                                    many=False,
                                                                    con-
                                                                    text=None,
                                                                    load_only=(),
                                                                    dump_only=(),
                                                                    par-
                                                                    tial=False,
                                                                    un-
                                                                    known=None)

    Bases: argschema.schemas.DefaultSchema

    class Meta
        Bases: object

        unknown = 'raise'

        opts = <marshmallow.schema.SchemaOpts object>

allensdk.brain_observatory.argschema_utilities.check_read_access (path)

allensdk.brain_observatory.argschema_utilities.check_write_access (filepath, al-
                                                                    low_exists=False)

allensdk.brain_observatory.argschema_utilities.check_write_access_dir (dirpath)

allensdk.brain_observatory.argschema_utilities.check_write_access_overwrite (path)

allensdk.brain_observatory.argschema_utilities.optional_lims_inputs (argv, in-
                                                                    put_schema,
                                                                    out-
                                                                    put_schema,
                                                                    lims_input_getter)

allensdk.brain_observatory.argschema_utilities.write_or_print_outputs (data,
                                                                    parser)

```

### allensdk.brain\_observatory.brain\_observatory\_exceptions module

```

exception allensdk.brain_observatory.brain_observatory_exceptions.BrainObservatoryAnalysisException
    Bases: Exception

exception allensdk.brain_observatory.brain_observatory_exceptions.EpochSeparationException
    Bases: Exception

exception allensdk.brain_observatory.brain_observatory_exceptions.MissingStimulusException
    Bases: Exception

exception allensdk.brain_observatory.brain_observatory_exceptions.NoEyeTrackingException
    Bases: Exception

```

### allensdk.brain\_observatory.brain\_observatory\_plotting module

```

allensdk.brain_observatory.brain_observatory_plotting.plot_drifting_grating_traces (dg,
                                                                    save_dir)

    saves figures with a Ori X TF grid of mean responses

```

```
allensdk.brain_observatory.brain_observatory_plotting.plot_lsn_traces (lsn,  
                                                                    save_dir,  
                                                                    suf-  
                                                                    fix="")  
allensdk.brain_observatory.brain_observatory_plotting.plot_ns_traces (nsa,  
                                                                    save_dir)  
allensdk.brain_observatory.brain_observatory_plotting.plot_running_a (dg,  
                                                                    nm1,  
                                                                    nm3,  
                                                                    save_dir)  
allensdk.brain_observatory.brain_observatory_plotting.plot_sg_traces (sg,  
                                                                    save_dir)
```

### **allensdk.brain\_observatory.chisquare\_categorical module**

Created on Wed Jun 5 15:52:22 2019

@author: dan

```
allensdk.brain_observatory.chisquare_categorical.advance_combination (curr_combination,  
                                                                    op-  
                                                                    tions_per_column)  
allensdk.brain_observatory.chisquare_categorical.chisq_from_stim_table (stim_table,  
                                                                    columns,  
                                                                    mean_sweep_events,  
                                                                    num_shuffles=1000,  
                                                                    ver-  
                                                                    bose=False)  
allensdk.brain_observatory.chisquare_categorical.compute_chi (observed,      ex-  
                                                                    pected)  
allensdk.brain_observatory.chisquare_categorical.compute_chi_shuffle (mean_sweep_events,  
                                                                    sweep_categories,  
                                                                    num_shuffles=1000)  
allensdk.brain_observatory.chisquare_categorical.compute_expected (mean_sweep_events,  
                                                                    sweep_conditions)  
allensdk.brain_observatory.chisquare_categorical.compute_observed (mean_sweep_events,  
                                                                    sweep_conditions)  
allensdk.brain_observatory.chisquare_categorical.make_category_dummy (sweep_categories)  
allensdk.brain_observatory.chisquare_categorical.stim_table_to_categories (stim_table,  
                                                                    columns,  
                                                                    ver-  
                                                                    bose=False)
```

### **allensdk.brain\_observatory.circle\_plots module**

```
class allensdk.brain_observatory.circle_plots.CoronaPlotter (angle_start=270,  
                                                                    plot_scale=1.2,  
                                                                    inner_radius=0.3,  
                                                                    *args, **kwargs)
```

Bases: `allensdk.brain_observatory.circle_plots.PolarPlotter`

```

infer_dims (self, category_data)

plot (self, category_data, data=None, clim=None, cmap=<matplotlib.colors.LinearSegmentedColormap
      object at 0x7fdbce546e80>)

set_dims (self, categories)

show_arrow (self, color=None)

show_circle (self, color=None)

class allensdk.brain_observatory.circle_plots.FanPlotter (group_scale=0.9, *args,
                                                         **kwargs)
    Bases: allensdk.brain_observatory.circle_plots.PolarPlotter

    static for_drifting_gratings ()

    static for_static_gratings ()

    infer_dims (self, r_data, angle_data, group_data)

    plot (self, r_data, angle_data, group_data=None, data=None,
          cmap=<matplotlib.colors.LinearSegmentedColormap object at 0x7fdbce546e80>, clim=None,
          rmap=None, rlim=None, axis_color=None, label_color=None)

    set_dims (self, rs, angles, groups)

    show_angle_labels (self, angles=None, labels=None, color=None, offset=0.05, fontdict=None)

    show_axes (self, angles=None, radii=None, closed=False, color=None)

    show_group_labels (self, groups=None, color=None, fontdict=None)

    show_r_labels (self, radii=None, labels=None, color=None, offset=0.1, fontdict=None)

class allensdk.brain_observatory.circle_plots.PolarPlotter (direction=-1, an-
                                                         gle_start=0, cir-
                                                         cle_scale=1.1, in-
                                                         ner_radius=None,
                                                         plot_center=(0.0,
                                                         0.0), plot_scale=0.9)

    Bases: object

    DIR_CCW = 1

    DIR_CW = -1

    finalize (self)

class allensdk.brain_observatory.circle_plots.TrackPlotter (direction=-1, an-
                                                         gle_start=270.0,
                                                         inner_radius=0.45,
                                                         ring_length=None,
                                                         *args, **kwargs)

    Bases: allensdk.brain_observatory.circle_plots.PolarPlotter

    plot (self, data, clim=None, cmap=<matplotlib.colors.LinearSegmentedColormap object at
          0x7fdbce546e80>, mean_cmap=<matplotlib.colors.LinearSegmentedColormap object at
          0x7fdbccce5e48>, norm=None)

    show_arrow (self, color=None)

allensdk.brain_observatory.circle_plots.add_angle_labels (ax, angles, labels, ra-
                                                         dius, color=None, font-
                                                         dict=None, offset=0.05)

```

```
allensdk.brain_observatory.circle_plots.add_arrow(ax, radius, start_angle, end_angle,
                                                    color=None, width=18.0)
allensdk.brain_observatory.circle_plots.angle_lines(angles, inner_radius,
                                                    outer_radius)
allensdk.brain_observatory.circle_plots.build_hex_pack(n)
allensdk.brain_observatory.circle_plots.hex_pack(radius, n)
allensdk.brain_observatory.circle_plots.make_pincushion_plot(data, trials, on,
                                                            nrows, ncols,
                                                            clim=None,
                                                            color_map=None,
                                                            radius=None)
allensdk.brain_observatory.circle_plots.polar_line_circles(radii, theta,
                                                            start_r=0)
allensdk.brain_observatory.circle_plots.polar_linspace(radius, start_angle,
                                                         stop_angle, num, end-
                                                         point=False, degrees=True)
```

Evenly distributed list of x,y coordinates from an input range of angles and a radius in polar coordinates.

```
allensdk.brain_observatory.circle_plots.polar_to_xy(angles, radius)
    Convert an array of angles (in radians) and a radius in polar coordinates to an array of x,y coordinates.
allensdk.brain_observatory.circle_plots.radial_arcs(rs, start_theta, end_theta)
allensdk.brain_observatory.circle_plots.radial_circles(rs)
allensdk.brain_observatory.circle_plots.reset_hex_pack()
allensdk.brain_observatory.circle_plots.rings_in_hex_pack(ct)
allensdk.brain_observatory.circle_plots.spiral_trials(radii, x=0.0, y=0.0)
allensdk.brain_observatory.circle_plots.spiral_trials_polar(r, theta, radii, off-
                                                            set=None)
allensdk.brain_observatory.circle_plots.wedge_ring(N, inner_radius, outer_radius,
                                                    start=0, stop=360)
```

## allensdk.brain\_observatory.demixer module

```
allensdk.brain_observatory.demixer.demix_time_dep_masks(raw_traces, stack, masks)
```

### Parameters

- **raw\_traces** – extracted traces
- **stack** – movie (same length as traces)
- **masks** – binary roi masks

### Returns

demixed traces

```
allensdk.brain_observatory.demixer.find_negative_baselines(trace)
allensdk.brain_observatory.demixer.find_negative_transients_threshold(trace,
                                                                        win-
                                                                        dow=500,
                                                                        length=10,
                                                                        std_devs=3)
allensdk.brain_observatory.demixer.find_zero_baselines(traces)
```



```
allensdk.brain_observatory.demixer.plot_negative_baselines (raw_traces,
                                                            demix_traces,
                                                            mask_array,
                                                            roi_ids_mask,
                                                            plot_dir, ext='png')
allensdk.brain_observatory.demixer.plot_negative_transients (raw_traces,
                                                             demix_traces,
                                                             valid_roi,
                                                             mask_array,
                                                             roi_ids_mask,
                                                             plot_dir, ext='png')
allensdk.brain_observatory.demixer.plot_overlap_masks_lengthOne (roi_ind, masks,
                                                                savefile=None,
                                                                weighted=False)
allensdk.brain_observatory.demixer.plot_traces (raw_trace, demix_trace, roi_id, roi_ind,
                                                save_file)
allensdk.brain_observatory.demixer.plot_transients (roi_ind, t_trans, masks, traces,
                                                    demix_traces, savefile)
allensdk.brain_observatory.demixer.rolling_window (trace, window=500)
```

**Parameters**

- **trace** –
- **window** –

**Returns****allensdk.brain\_observatory.dff module**

```
allensdk.brain_observatory.dff.calculate_dff (traces, dff_computation_cb=None,
                                                save_plot_dir=None)
```

Apply dF/F computation to a set of traces.

The default computation method is `compute_dff_windowed_median()` using default window parameters.

**Parameters**

- traces** [np.ndarray] 2D array of traces to be analyzed.
- dff\_computation\_cb** [function] Function that takes traces as an argument and returns an array of the same shape that is the calculated dF/F.
- save\_plot\_dir** [str] Directory to save dF/F plots to. By default no plots are saved.

**Returns**

- dff** [np.ndarray] 2D array of dF/F traces.

```
allensdk.brain_observatory.dff.compute_dff_windowed_median (traces, median_kernel_long=5401,
                                                            median_kernel_short=101,
                                                            noise_stds=None,
                                                            n_small_baseline_frames=None,
                                                            **kwargs)
```

Compute dF/F of a set of traces with median filter detrending.

The operation is basically:

```
T_long = windowed_median(T) # long timescale kernel
T_dff1 = (T - T_long) / elementwise_max(T_long, noise_std(T))
T_short = windowed_median(T_dff1) # short timescale kernel
T_dff = T_dff1 - elementwise_min(T_short, 2.5*noise_std(T_dff1))
```

#### Parameters

**traces** [np.ndarray] 2D array of traces to be analyzed.

**median\_kernel\_long** [int] Window size to use for long timescale median detrending.

**median\_kernel\_short** [int] Window size to use for short timescale median detrending.

**noise\_stds** [list] List that will contain noise\_std(T\_dff1) for each trace. The value for each trace will be appended to the list if provided.

**n\_small\_baseline\_frames** [list] List that will contain the number of frames for each trace where the long-timescale median window is less than noise\_std(T). The value for each trace will be appended to the list if provided.

**kwargs:** Additional keyword arguments are passed to `noise_std()`.

#### Returns

**dff** [np.ndarray] 2D array of dF/F traces.

```
allensdk.brain_observatory.dff.compute_dff_windowed_mode(traces,
                                                         mode_kernelsize=5400,
                                                         mean_kernelsize=3000)
```

Compute dF/F of a set of traces using a low-pass windowed-mode operator.

The operation is basically:

```
T_mm = windowed_mean(windowed_mode(T))
T_dff = (T - T_mm) / T_mm
```

#### Parameters

**traces** [np.ndarray] 2D array of traces to be analyzed.

**mode\_kernelsize** [int] Window size to use for windowed\_mode.

**mean\_kernelsize** [int] Window size to use for windowed\_mean.

#### Returns

**dff** [np.ndarray] 2D array of dF/F traces.

```
allensdk.brain_observatory.dff.main()
allensdk.brain_observatory.dff.movingaverage(x, kernelsize, y)
```

Compute the windowed average of an array.

#### Parameters

**x** [np.ndarray] Array to be analyzed

**kernelsize** [int] Size of the moving window

**y** [np.ndarray] Output array to store the results

`allensdk.brain_observatory.dff.movingmode_fast(x, kernel_size, y)`

Compute the windowed mode of an array. A running mode is initialized with a histogram of values over the initial `kernel_size/2` values. The mode is then updated as the kernel moves by adding and subtracting values from the histogram.

#### Parameters

**x** [np.ndarray] Array to be analyzed

**kernel\_size** [int] Size of the moving window

**y** [np.ndarray] Output array to store the results

`allensdk.brain_observatory.dff.noise_std(x, noise_kernel_length=31, positive_peak_scale=1.5, outlier_std_scale=2.5)`

Robust estimate of the standard deviation of the trace noise.

`allensdk.brain_observatory.dff.plot_onetrace(dff, fc)`

Debug plotting function

`allensdk.brain_observatory.dff.robust_std(x)`

Robust estimate of standard deviation.

Estimate of the standard deviation using the median absolute deviation of x.

### allensdk.brain\_observatory.drifting\_gratings module

**class** `allensdk.brain_observatory.drifting_gratings.DriftingGratings(data_set, **kwargs)`

Bases: `allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis`

Perform tuning analysis specific to drifting gratings stimulus.

#### Parameters

**data\_set**: BrainObservatoryNwbDataSet object

**static from\_analysis\_file** (data\_set, analysis\_file)

**get\_noise\_correlation** (self, corr='spearman')

**get\_peak** (self)

Computes metrics related to each cell's peak response condition.

#### Returns

Pandas data frame containing the following columns (\_dg suffix is for drifting grating):

- ori\_dg (orientation)
- tf\_dg (temporal frequency)
- reliability\_dg
- osi\_dg (orientation selectivity index)
- dsi\_dg (direction selectivity index)
- peak\_dff\_dg (peak dF/F)
- ptest\_dg
- p\_run\_dg

- `run_modulation_dg`
- `cv_dg` (circular variance)

**`get_representational_similarity`** (*self*, *corr*='spearman')

**`get_response`** (*self*)

Computes the mean response for each cell to each stimulus condition. Return is a (# orientations, # temporal frequencies, # cells, 3) np.ndarray. The final dimension contains the mean response to the condition (index 0), standard error of the mean of the response to the condition (index 1), and the number of trials with a significant response ( $p < 0.05$ ) to that condition (index 2).

#### Returns

Numpy array storing mean responses.

**`get_signal_correlation`** (*self*, *corr*='spearman')

**`number_ori`**

**`number_tf`**

**`open_star_plot`** (*self*, *cell\_specimen\_id*=None, *include\_labels*=False, *cell\_index*=None)

**`orivals`**

**`plot_direction_selectivity`** (*self*, *si\_range*=[0, 1.5], *n\_hist\_bins*=50, *color*='#ccccdd',  
*p\_value\_max*=0.05, *peak\_dff\_min*=3)

**`plot_orientation_selectivity`** (*self*, *si\_range*=[0, 1.5], *n\_hist\_bins*=50, *color*='#ccccdd',  
*p\_value\_max*=0.05, *peak\_dff\_min*=3)

**`plot_preferred_direction`** (*self*, *include\_labels*=False, *si\_range*=[0, 1.5], *color*='#ccccdd',  
*p\_value\_max*=0.05, *peak\_dff\_min*=3)

**`plot_preferred_temporal_frequency`** (*self*, *si\_range*=[0, 1.5], *color*='#ccccdd',  
*p\_value\_max*=0.05, *peak\_dff\_min*=3)

**`populate_stimulus_table`** (*self*)

Implemented by subclasses.

**`reshape_response_array`** (*self*)

**Returns** response array in cells x stim x repetition for noise correlations

**`tfvals`**

## allensdk.brain\_observatory.findlevel module

`allensdk.brain_observatory.findlevel.findlevel` (*inwave*, *threshold*, *direction*='both')

## allensdk.brain\_observatory.locally\_sparse\_noise module

**class** `allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` (*data\_set*,  
*stim-*  
*u-*  
*lus*=None,  
*\*\*kwargs*)

Bases: `allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis`

Perform tuning analysis specific to the locally sparse noise stimulus.

#### Parameters

**data\_set:** BrainObservatoryNwbDataSet object

**stimulus:** string Name of locally sparse noise stimulus. See brain\_observatory.stimulus\_info.

**nrows:** int Number of rows in the stimulus template

**ncol:** int Number of columns in the stimulus template

**LSN**

**LSN\_GREY** = 127

**LSN\_OFF** = 0

**LSN\_OFF\_SCREEN** = 64

**LSN\_ON** = 255

**LSN\_mask**

**cell\_index\_receptive\_field\_analysis\_data**

**extralength**

**static from\_analysis\_file** (*data\_set, analysis\_file, stimulus*)

**get\_mean\_response** (*self*)

**get\_peak** (*self*)

Implemented by subclasses.

**get\_receptive\_field** (*self*)

Calculates receptive fields for each cell

**get\_receptive\_field\_analysis\_data** (*self*)

Calculates receptive fields for each cell

**get\_receptive\_field\_attribute\_df** (*self*)

**interlength**

**mean\_response**

**static merge\_mean\_response** (*rc1, rc2*)

Move out of this class, to session analysis

**open\_pincushion\_plot** (*self, on, cell\_specimen\_id=None, color\_map=None, cell\_index=None*)

**plot\_cell\_receptive\_field** (*self, on, cell\_specimen\_id=None, color\_map=None, clim=None, mask=None, cell\_index=None, scalebar=True*)

**plot\_population\_receptive\_field** (*self, color\_map='RdPu', clim=None, mask=None, scalebar=True*)

**plot\_receptive\_field\_analysis\_data** (*self, cell\_index, \*\*kwargs*)

**populate\_stimulus\_table** (*self*)

Implemented by subclasses.

**static read\_cell\_index\_receptive\_field\_analysis** (*file\_handle, prefix, path=None*)

**receptive\_field**

**static save\_cell\_index\_receptive\_field\_analysis** (*cell\_index\_receptive\_field\_analysis\_data, new\_nwb, prefix*)

**sort\_trials** (*self*)

**sweeplength**

**allensdk.brain\_observatory.natural\_movie module**

```
class allensdk.brain_observatory.natural_movie.NaturalMovie (data_set,  
                                                         movie_name,  
                                                         **kwargs)
```

Bases: *allensdk.brain\_observatory.stimulus\_analysis.StimulusAnalysis*

Perform tuning analysis specific to natural movie stimulus.

**Parameters**

**data\_set:** BrainObservatoryNwbDataSet object

**movie\_name:** string

one of [ *stimulus\_info.NATURAL\_MOVIE\_ONE*, *stimulus\_info.NATURAL\_MOVIE\_TWO*,  
*stimulus\_info.NATURAL\_MOVIE\_THREE* ]

```
static from_analysis_file (data_set, analysis_file, movie_name)
```

```
get_peak (self)
```

Computes properties of the peak response condition for each cell.

**Returns**

Pandas data frame with the below fields. A suffix of “nm1”, “nm2” or “nm3” is appended to the field name

on which of three movie clips was presented.

- *peak\_nm1* (frame with peak response)
- *response\_variability\_nm1*

```
get_sweep_response (self)
```

Returns the dF/F response for each cell

**Returns**

Numpy array

```
open_track_plot (self, cell_specimen_id=None, cell_index=None)
```

```
populate_stimulus_table (self)
```

Implemented by subclasses.

**sweep\_response**

**sweeplength**

**allensdk.brain\_observatory.natural\_scenes module**

```
class allensdk.brain_observatory.natural_scenes.NaturalScenes (data_set,  
                                                         **kwargs)
```

Bases: *allensdk.brain\_observatory.stimulus\_analysis.StimulusAnalysis*

Perform tuning analysis specific to natural scenes stimulus.

**Parameters**

**data\_set:** BrainObservatoryNwbDataSet object

**extralength**

```
static from_analysis_file (data_set, analysis_file)
```

**get\_noise\_correlation** (*self*, *corr*='spearman')

**get\_peak** (*self*)

Computes metrics about peak response condition for each cell.

#### Returns

**Pandas data frame with the following fields ('\_ns' suffix is for natural scene):**

- scene\_ns (scene number)
- reliability\_ns
- peak\_dff\_ns (peak dF/F)
- ptest\_ns
- p\_run\_ns
- run\_modulation\_ns
- time\_to\_peak\_ns

**get\_representational\_similarity** (*self*, *corr*='spearman')

**get\_response** (*self*)

Computes the mean response for each cell to each stimulus condition. Return is a (# scenes, # cells, 3) np.ndarray. The final dimension contains the mean response to the condition (index 0), standard error of the mean of the response to the condition (index 1), and the number of trials with a significant ( $p < 0.05$ ) response to that condition (index 2).

#### Returns

**Numpy array storing mean responses.**

**get\_signal\_correlation** (*self*, *corr*='spearman')

**interlength**

**number\_scenes**

**open\_corona\_plot** (*self*, *cell\_specimen\_id*=None, *cell\_index*=None)

**plot\_time\_to\_peak** (*self*, *p\_value\_max*=0.05, *color\_map*=<matplotlib.colors.LinearSegmentedColormap object at 0x7fdbcce39c88>)

**populate\_stimulus\_table** (*self*)

Implemented by subclasses.

**reshape\_response\_array** (*self*)

**Returns** response array in cells x stim x repetition for noise correlations

**sweeplength**

## allensdk.brain\_observatory.observatory\_plots module

```
class allensdk.brain_observatory.observatory_plots.DimensionPatchHandler (vals,
                                                                    start_color,
                                                                    end_color,
                                                                    *args,
                                                                    **kwargs)
```

Bases: object

```
dim_color (self, index)

legend_artist (self, legend, orig_handle, fontsize, handlebox)

allensdk.brain_observatory.observatory_plots.figure_in_px (w, h, file_name,
                                                         dpi=96.0, transparent=False)

allensdk.brain_observatory.observatory_plots.finalize_no_axes (pad=0.0)

allensdk.brain_observatory.observatory_plots.finalize_no_labels (pad=0.3, legend=False)

allensdk.brain_observatory.observatory_plots.finalize_with_axes (pad=0.3)

allensdk.brain_observatory.observatory_plots.float_label (n)

allensdk.brain_observatory.observatory_plots.plot_cell_correlation (sig_corrs,
                                                                    labels,
                                                                    colors,
                                                                    scale=15)

allensdk.brain_observatory.observatory_plots.plot_combined_speed (binned_resp_vis,
                                                                    binned_dx_vis,
                                                                    binned_resp_sp,
                                                                    binned_dx_sp,
                                                                    evoked_color,
                                                                    spont_color)

allensdk.brain_observatory.observatory_plots.plot_condition_histogram (vals,
                                                                    bins,
                                                                    color='#ccccdd')

allensdk.brain_observatory.observatory_plots.plot_mask_outline (mask, ax,
                                                                color='k')

allensdk.brain_observatory.observatory_plots.plot_pupil_location (xy_deg, s=1,
                                                                    c=None,
                                                                    cmap=<matplotlib.colors.LinearSegmentedColormap object at 0x7fdbcc9b0ef0>,
                                                                    edge_color='', include_labels=True)

allensdk.brain_observatory.observatory_plots.plot_radial_histogram (angles,
                                                                    counts,
                                                                    all_angles=None,
                                                                    include_labels=False,
                                                                    offset=180.0,
                                                                    direction=-1,
                                                                    closed=False,
                                                                    color='#ccccdd')
```



```

allensdk.brain_observatory.observatory_plots.plot_receptive_field(rf,
                                                                    color_map=None,
                                                                    clim=None,
                                                                    mask=None,
                                                                    out-
                                                                    line_color='#cccccc',
                                                                    scale-
                                                                    bar=True)

allensdk.brain_observatory.observatory_plots.plot_representational_similarity(rs,
                                                                                dims=None,
                                                                                dim_labels=None,
                                                                                col-
                                                                                ors=None,
                                                                                dim_order=None,
                                                                                la-
                                                                                bels=True)

allensdk.brain_observatory.observatory_plots.plot_selectivity_cumulative_histogram(sis,
                                                                                   xla-
                                                                                   bel,
                                                                                   si_range=[
                                                                                   1.5],
                                                                                   n_hist_bins,
                                                                                   color='#cc

allensdk.brain_observatory.observatory_plots.plot_speed(binned_resp,  binned_dx,
                                                         num_bins, color)

allensdk.brain_observatory.observatory_plots.plot_time_to_peak(msrs,      ttps,
                                                                t_start,  t_end,
                                                                stim_start,
                                                                stim_end,
                                                                cmap)

allensdk.brain_observatory.observatory_plots.population_correlation_scatter(sig_corrs,
                                                                              noise_corrs,
                                                                              la-
                                                                              bels,
                                                                              col-
                                                                              ors,
                                                                              scale=15)

```

## allensdk.brain\_observatory.r\_neuropil module

```

class allensdk.brain_observatory.r_neuropil.NeuropilSubtract (lam=0.05, dt=1.0,
                                                             folds=4)
    Bases: object
    TODO: docs

    estimate_error (self, r)
        Estimate error values for a given r for each fold and return the mean.

    fit (self, r_range=[0.0, 2.0], iterations=3, dr=0.1, dr_factor=0.1)
        Estimate error values for a range of r values. Identify a new r range around the minimum error values and
        repeat multiple times. TODO: docs

    fit_block_coordinate_desc (self, r_init=5.0, min_delta_r=1e-08)

```

**set\_F** (*self*, *F\_M*, *F\_N*)

Break the *F\_M* and *F\_N* traces into the number of folds specified in the class constructor and normalize each fold of *F\_M* and *R\_N* relative to *F\_N*.

`allensdk.brain_observatory.r_neuropil.ab_from_T` (*T*, *lam*, *dt*)

`allensdk.brain_observatory.r_neuropil.ab_from_diagonals` (*mat\_dict*)

Constructs value for `scipy.linalg.solve_banded`

#### Parameters

**mat\_dict:** dictionary of diagonals keyed by offsets

#### Returns

**ab:** value for `scipy.linalg.solve_banded`

`allensdk.brain_observatory.r_neuropil.alpha_filter` (*A=1.0*, *alpha=0.05*, *beta=0.25*,  
*T=100*)

`allensdk.brain_observatory.r_neuropil.error_calc` (*F\_M*, *F\_N*, *F\_C*, *r*)

`allensdk.brain_observatory.r_neuropil.error_calc_outlier` (*F\_M*, *F\_N*, *F\_C*, *r*)

`allensdk.brain_observatory.r_neuropil.estimate_contamination_ratios` (*F\_M*,  
*F\_N*,  
*lam=0.05*,  
*folds=4*,  
*iterations=3*,  
*r\_range=[0.0*,  
*2.0]*,  
*dr=0.1*,  
*dr\_factor=0.1*)

Calculates neuropil contamination of ROI

#### Parameters

**F\_M:** ROI trace **F\_N:** Neuropil trace

#### Returns

**dictionary:** key-value pairs

- 'r': the contamination ratio – corrected trace =  $M - r * N$
- 'err': RMS error
- 'min\_error': minimum error
- 'bounds\_error': boolean. True if error or *R* are outside tolerance

`allensdk.brain_observatory.r_neuropil.get_diagonals_from_sparse` (*mat*)

Returns a dictionary of diagonals keyed by offsets

#### Parameters

**mat:** `scipy.sparse` matrix

#### Returns

**dictionary:** diagonals keyed by offsets

`allensdk.brain_observatory.r_neuropil.normalize_F` (*F\_M*, *F\_N*)

`allensdk.brain_observatory.r_neuropil.synthesize_F` (*T*, *af1*, *af2*, *p1=0.05*, *p2=0.1*)

Build a synthetic *F\_C*, *F\_M*, *F\_N*, and *r* of length *T* TODO: docs

`allensdk.brain_observatory.r_neuropil.validate_with_synthetic_F(T, N)`  
 Compute N synthetic traces of length T with known values of r, then estimate r. TODO: docs

## `allensdk.brain_observatory.roi_masks` module

**class** `allensdk.brain_observatory.roi_masks.Mask` (*image\_w*, *image\_h*, *label*,  
*mask\_group*)

Bases: `object`

Abstract class to represent image segmentation mask. Its two main subclasses are `RoiMask` and `NeuropilMask`. The former represents the mask of a region of interest (ROI), such as a cell observed in 2-photon imaging. The latter represents the neuropil around that cell, and is useful when subtracting the neuropil signal from the measured ROI signal.

This class should not be instantiated directly.

### Parameters

**image\_w: integer** Width of image that ROI resides in

**image\_h: integer** Height of image that ROI resides in

**label: text** User-defined text label to identify mask

**mask\_group: integer** User-defined number to help put masks into different categories

**get\_mask\_plane** (*self*)

Returns mask content on full-size image plane

### Returns

numpy 2D array [img\_rows][img\_cols]

**init\_by\_pixels** (*self*, *border*, *pix\_list*)

Initialize mask using a list of mask pixels

### Parameters

**border: float[4]** Coordinates defining useable area of image. See `create_roi_mask()`

**pix\_list: integer[][2]** List of pixel coordinates (x,y) that define the mask

**overlaps\_motion\_border**

**class** `allensdk.brain_observatory.roi_masks.NeuropilMask` (*w*, *h*, *label*, *mask\_group*)

Bases: `allensdk.brain_observatory.roi_masks.Mask`

**init\_by\_mask** (*self*, *border*, *array*)

Initialize mask using spatial mask

### Parameters

**border: float[4]** Border widths on the [right, left, down, up] sides. The resulting neuropil mask will not include pixels falling into a border.

**array: integer[image height][image width]** Image-sized array that describes the mask. Active parts of the mask should have values >0. Background pixels must be zero

**class** `allensdk.brain_observatory.roi_masks.RoiMask` (*image\_w*, *image\_h*, *label*,  
*mask\_group*)

Bases: `allensdk.brain_observatory.roi_masks.Mask`

**init\_by\_mask** (*self*, *border*, *array*)

Initialize mask using spatial mask

### Parameters

**border: float[4]** Coordinates defining useable area of image. See `create_roi_mask()`.

**roi\_mask: integer[image height][image width]** Image-sized array that describes the mask. Active parts of the mask should have values >0. Background pixels must be zero

```
allensdk.brain_observatory.roi_masks.calculate_roi_and_neuropil_traces(movie_h5,
                                                                    roi_mask_list,
                                                                    motion_border)
```

get roi and neuropil masks

```
allensdk.brain_observatory.roi_masks.calculate_traces(stack, mask_list,
                                                       block_size=1000)
```

Calculates the average response of the specified masks in the image stack

### Parameters

**stack: float[image height][image width]** Image stack that masks are applied to

**mask\_list: list<Mask>** List of masks

### Returns

**float[number masks][number frames]** This is the average response for each Mask in each image frame

```
allensdk.brain_observatory.roi_masks.create_neuropil_mask(roi, border, combined_binary_mask,
                                                         label=None)
```

Convenience function to create and initializes a Neuropil mask. Neuropil masks are defined as the region around an ROI, up to 13 pixels out, that does not include other ROIs

### Parameters

**roi: RoiMask object** The ROI that the neuropil masks will be based on

**border: float[4]** Border widths on the [right, left, down, up] sides. The resulting neuropil mask will not include pixels falling into a border.

**combined\_binary\_mask** List of pixel coordinates (x,y) that define the mask

**combined\_binary\_mask: integer[image\_h][image\_w]** Image-sized array that shows the position of all ROIs in the image. ROI masks should have a value of one. Background pixels must be zero. In other words, the combined\_binary\_mask is a bitmap union of all ROI masks

**label: text** User-defined text label to identify the mask

### Returns

**NeuropilMask object**

```
allensdk.brain_observatory.roi_masks.create_roi_mask(image_w, image_h, border,
                                                       pix_list=None,
                                                       roi_mask=None, label=None,
                                                       mask_group=-1)
```

Convenience function to create and initializes an RoiMask

### Parameters

**image\_w: integer** Width of image that ROI resides in

**image\_h: integer** Height of image that ROI resides in

**border: float[4]** Coordinates defining useable area of image. If the entire image is usable, and masks are valid anywhere in the image, this should be [0, 0, 0, 0]. The following constants help describe the array order:

RIGHT\_SHIFT = 0

LEFT\_SHIFT = 1

DOWN\_SHIFT = 2

UP\_SHIFT = 3

When parts of the image are unusable, for example due to motion correction shifting of different image frames, the border array should store the usable image area

**pix\_list: integer[][2]** List of pixel coordinates (x,y) that define the mask

**roi\_mask: integer[image\_h][image\_w]** Image-sized array that describes the mask. Active parts of the mask should have values >0. Background pixels must be zero

**label: text** User-defined text label to identify mask

**mask\_group: integer** User-defined number to help put masks into different categories

#### Returns

**RoiMask object**

`allensdk.brain_observatory.roi_masks.create_roi_mask_array(rois)`  
Create full image mask array from list of RoiMasks.

#### Parameters

**rois: list<RoiMask>** List of roi masks.

#### Returns

**np.ndarray: NxWxH array** Boolean array of of len(rois) image masks.

`allensdk.brain_observatory.roi_masks.validate_mask(mask)`  
Check a given roi or neuropil mask for (a subset of) disqualifying problems.

### allensdk.brain\_observatory.running\_speed module

**class** `allensdk.brain_observatory.running_speed.RunningSpeed`

Bases: tuple

Describes the rate at which an experimental subject ran during a session.

**values** [np.ndarray] running speed (cm/s) at each sample point

**timestamps** [np.ndarray] The time at which each sample was collected (s).

**timestamps**

Alias for field number 0

**values**

Alias for field number 1

**allensdk.brain\_observatory.session\_analysis module**

```
class allensdk.brain_observatory.session_analysis.SessionAnalysis (nwb_path,  
                                                                save_path)
```

Bases: object

Run all of the stimulus-specific analyses associated with a single experiment session.

**Parameters**

**nwb\_path:** string, path to NWB file

**save\_path:** string, path to HDF5 file to store outputs. Recommended NOT to modify the NWB file.

**append\_experiment\_metrics** (*self*, *metrics*)

Extract stimulus-agnostic metrics from an experiment into a dictionary

**append\_metadata** (*self*, *df*)

Append the metadata fields from the NWB file as columns to a pd.DataFrame

**append\_metrics\_drifting\_grating** (*self*, *metrics*, *dg*)

Extract metrics from the DriftingGratings peak response table into a dictionary.

**append\_metrics\_locally\_sparse\_noise** (*self*, *metrics*, *lsn*)

Extract metrics from the LocallySparseNoise peak response table into a dictionary.

**append\_metrics\_natural\_movie\_one** (*self*, *metrics*, *nma*)

Extract metrics from the NaturalMovie(stimulus\_info.NATURAL\_MOVIE\_ONE) peak response table into a dictionary.

**append\_metrics\_natural\_movie\_three** (*self*, *metrics*, *nma*)

Extract metrics from the NaturalMovie(stimulus\_info.NATURAL\_MOVIE\_THREE) peak response table into a dictionary.

**append\_metrics\_natural\_movie\_two** (*self*, *metrics*, *nma*)

Extract metrics from the NaturalMovie(stimulus\_info.NATURAL\_MOVIE\_TWO) peak response table into a dictionary.

**append\_metrics\_natural\_scene** (*self*, *metrics*, *ns*)

Extract metrics from the NaturalScenes peak response table into a dictionary.

**append\_metrics\_static\_grating** (*self*, *metrics*, *sg*)

Extract metrics from the StaticGratings peak response table into a dictionary.

**save\_session\_a** (*self*, *dg*, *nm1*, *nm3*, *peak*)

Save the output of session A analysis to self.save\_path.

**Parameters**

**dg:** DriftingGratings instance

**nm1:** NaturalMovie instance This NaturalMovie instance should have been created with movie\_name = stimulus\_info.NATURAL\_MOVIE\_ONE

**nm3:** NaturalMovie instance This NaturalMovie instance should have been created with movie\_name = stimulus\_info.NATURAL\_MOVIE\_THREE

**peak:** pd.DataFrame The combined peak response property table created in self.session\_a().

**save\_session\_b** (*self*, *sg*, *nm1*, *ns*, *peak*)

Save the output of session B analysis to self.save\_path.

**Parameters****sg: StaticGratings instance****nm1: NaturalMovie instance** This NaturalMovie instance should have been created with `movie_name = stimulus_info.NATURAL_MOVIE_ONE`**ns: NaturalScenes instance****peak: pd.DataFrame** The combined peak response property table created in `self.session_b()`.**save\_session\_c** (*self, lsn, nm1, nm2, peak*)Save the output of session C analysis to `self.save_path`.**Parameters****lsn: LocallySparseNoise instance****nm1: NaturalMovie instance** This NaturalMovie instance should have been created with `movie_name = stimulus_info.NATURAL_MOVIE_ONE`**nm2: NaturalMovie instance** This NaturalMovie instance should have been created with `movie_name = stimulus_info.NATURAL_MOVIE_TWO`**peak: pd.DataFrame** The combined peak response property table created in `self.session_c()`.**save\_session\_c2** (*self, lsn4, lsn8, nm1, nm2, peak*)Save the output of session C2 analysis to `self.save_path`.**Parameters****lsn4: LocallySparseNoise instance** This LocallySparseNoise instance should have been created with `self.stimulus = stimulus_info.LOCALLY_SPARSE_NOISE_4DEG`.**lsn8: LocallySparseNoise instance** This LocallySparseNoise instance should have been created with `self.stimulus = stimulus_info.LOCALLY_SPARSE_NOISE_8DEG`.**nm1: NaturalMovie instance** This NaturalMovie instance should have been created with `movie_name = stimulus_info.NATURAL_MOVIE_ONE`**nm2: NaturalMovie instance** This NaturalMovie instance should have been created with `movie_name = stimulus_info.NATURAL_MOVIE_TWO`**peak: pd.DataFrame** The combined peak response property table created in `self.session_c2()`.**session\_a** (*self, plot\_flag=False, save\_flag=True*)Run stimulus-specific analysis for natural movie one, natural movie three, and drifting gratings. The input NWB be for a `stimulus_info.THREE_SESSION_A` experiment.**Parameters****plot\_flag: bool** Whether to generate brain\_observatory\_plotting work plots after running analysis.**save\_flag: bool** Whether to save the output of analysis to `self.save_path` upon completion.**session\_b** (*self, plot\_flag=False, save\_flag=True*)Run stimulus-specific analysis for natural scenes, static gratings, and natural movie one. The input NWB be for a `stimulus_info.THREE_SESSION_B` experiment.**Parameters**

**plot\_flag: bool** Whether to generate brain\_observatory\_plotting work plots after running analysis.

**save\_flag: bool** Whether to save the output of analysis to self.save\_path upon completion.

**session\_c** (*self*, *plot\_flag=False*, *save\_flag=True*)

Run stimulus-specific analysis for natural movie one, natural movie two, and locally sparse noise. The input NWB be for a stimulus\_info.THREE\_SESSION\_C experiment.

#### Parameters

**plot\_flag: bool** Whether to generate brain\_observatory\_plotting work plots after running analysis.

**save\_flag: bool** Whether to save the output of analysis to self.save\_path upon completion.

**session\_c2** (*self*, *plot\_flag=False*, *save\_flag=True*)

Run stimulus-specific analysis for locally sparse noise (4 deg.), locally sparse noise (8 deg.), natural movie one, and natural movie two. The input NWB be for a stimulus\_info.THREE\_SESSION\_C2 experiment.

#### Parameters

**plot\_flag: bool** Whether to generate brain\_observatory\_plotting work plots after running analysis.

**save\_flag: bool** Whether to save the output of analysis to self.save\_path upon completion.

**verify\_roi\_lists\_equal** (*self*, *roi1*, *roi2*)

TODO: replace this with simpler numpy comparisons

`allensdk.brain_observatory.session_analysis.main()`

`allensdk.brain_observatory.session_analysis.multi_dataframe_merge(dfs)`

merge a number of pd.DataFrames into a single dataframe on their index columns. If any columns are duplicated, prefer the first occurring instance of the column

`allensdk.brain_observatory.session_analysis.run_session_analysis(nwb_path,  
save_path,  
plot_flag=False,  
save_flag=True)`

Inspect an NWB file to determine which experiment session was run and compute all stimulus-specific analyses.

#### Parameters

**nwb\_path: string** Path to NWB file.

**save\_path: string** path to save results. Recommended NOT to use NWB file.

**plot\_flag: bool** Whether to save brain\_observatory\_plotting work plots.

**save\_flag: bool** Whether to save results to save\_path.

### `allensdk.brain_observatory.session_api_utils` module

**class** `allensdk.brain_observatory.session_api_utils.ParamsMixin` (*ignore: set = {'api'}*)

Bases: `object`

This mixin adds parameter management functionality to the class it is mixed into.



This mixin expects that the class it is mixed into will have an `__init__` with type annotated parameters. It also expects for the class to have semi-private attributes of the `__init__` type annotated parameters.

Example:

SomeClassWhereParamManagementIsDesired(ParamsMixin):

```
# Managed params should be typed (with simple types if possible)! def __init__(self,
param_to_ignore, a_param_1: int, a_param_2: float,
    b_param_1: list):
    # Parameters can be ignored by the mixin super().__init__(ignore={'param_to_ignore'})
    # Pay attention to the naming scheme! self._a_param_1 = a_param_1 self._a_param_2
    = a_param_2 self._b_param_1 = b_param_1
    ...
```

After being mixed in, methods like `'get_params'`, `'set_params'`, `'needs_data_refresh'`, and `'clear_updated_params'` will be available.

**clear\_updated\_params** (*self*, *data\_params: set*)

This method clears 'updated params' whose data have been updated

**get\_params** (*self*) → Dict[str, Any]

Get managed params and their values

**needs\_data\_refresh** (*self*, *data\_params: set*) → bool

Check if specific params have been updated via `set_params()`

**set\_params** (*self*, *\*\*params*)

Set managed params

`allensdk.brain_observatory.session_api_utils.is_equal` (*a: Any*, *b: Any*) → bool

Function to deal with checking if two variables of possibly mixed types have the same value.

## allensdk.brain\_observatory.static\_gratings module

**class** `allensdk.brain_observatory.static_gratings.StaticGratings` (*data\_set*, *\*\*kwargs*)

Bases: `allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis`

Perform tuning analysis specific to static gratings stimulus.

### Parameters

**data\_set**: BrainObservatoryNwbDataSet object

**extralength**

**static\_from\_analysis\_file** (*data\_set*, *analysis\_file*)

**get\_noise\_correlation** (*self*, *corr='spearman'*)

**get\_peak** (*self*)

Computes metrics related to each cell's peak response condition.

### Returns

Panda data frame with the following fields (`_sg` suffix is

for static grating):

- `ori_sg` (orientation)

- `sf_sg` (spatial frequency)
- `phase_sg`
- `response_variability_sg`
- `osi_sg` (orientation selectivity index)
- `peak_dff_sg` (peak dF/F)
- `p_test_sg`
- `time_to_peak_sg`

**`get_representational_similarity`** (*self*, *corr*='spearman')

**`get_response`** (*self*)

Computes the mean response for each cell to each stimulus condition. Return is a (# orientations, # spatial frequencies, # phasees, # cells, 3) np.ndarray. The final dimension contains the mean response to the condition (index 0), standard error of the mean of the response to the condition (index 1), and the number of trials with a significant response ( $p < 0.05$ ) to that condition (index 2).

#### Returns

Numpy array storing mean responses.

**`get_signal_correlation`** (*self*, *corr*='spearman')

**`interlength`**

**`number_ori`**

**`number_phase`**

**`number_sf`**

**`open_fan_plot`** (*self*, *cell\_specimen\_id*=None, *include\_labels*=False, *cell\_index*=None)

**`orivals`**

**`phasevals`**

**`plot_orientation_selectivity`** (*self*, *si\_range*=[0, 1.5], *n\_hist\_bins*=50, *color*='#ccccdd',  
*p\_value\_max*=0.05, *peak\_dff\_min*=3)

**`plot_preferred_orientation`** (*self*, *include\_labels*=False, *si\_range*=[0, 1.5], *color*='#ccccdd',  
*p\_value\_max*=0.05, *peak\_dff\_min*=3)

**`plot_preferred_spatial_frequency`** (*self*, *si\_range*=[0, 1.5], *color*='#ccccdd',  
*p\_value\_max*=0.05, *peak\_dff\_min*=3)

**`plot_time_to_peak`** (*self*, *p\_value\_max*=0.05, *color\_map*=<matplotlib.colors.LinearSegmentedColormap  
object at 0x7fdbcce39c88>)

**`populate_stimulus_table`** (*self*)

Implemented by subclasses.

**`reshape_response_array`** (*self*)

**Returns** response array in cells x stim conditions x repetition for noise correlations

this is a re-organization of the mean sweep response table

**`sfvals`**

**`sweeplength`**

**allensdk.brain\_observatory.stimulus\_analysis module**

**class** allensdk.brain\_observatory.stimulus\_analysis.**StimulusAnalysis** (*data\_set*)

Bases: object

Base class for all response analysis code. Subclasses are responsible for computing metrics and traces relevant to a particular stimulus. The base class contains methods for organizing sweep responses row of a stimulus stable (`get_sweep_response`). Subclasses implement the `get_response` method, computes the mean sweep response to all sweeps for a each stimulus condition.

**Parameters**

**data\_set:** BrainObservatoryNwbDataSet instance

**speed\_tuning:** boolean, deprecated Whether or not to compute speed tuning histograms

**acquisition\_rate**

**binned\_cells\_sp**

**binned\_cells\_vis**

**binned\_dx\_sp**

**binned\_dx\_vis**

**cell\_id**

**celltraces**

**dfftraces**

**dxcm**

**dxtime**

**get\_fluorescence** (*self*)

**get\_peak** (*self*)

Implemented by subclasses.

**get\_response** (*self*)

Implemented by subclasses.

**get\_speed\_tuning** (*self*, *binsize*)

Calculates speed tuning, spontaneous versus visually driven. The return is a 5-tuple of speed and dF/F histograms.

**binned\_dx\_sp:** (bins,2) np.ndarray of running speeds binned during spontaneous activity stimulus. The first bin contains all speeds below 1 cm/s. Dimension 0 is mean running speed in the bin. Dimension 1 is the standard error of the mean.

**binned\_cells\_sp:** (bins,2) np.ndarray of fluorescence during spontaneous activity stimulus. First bin contains all data for speeds below 1 cm/s. Dimension 0 is mean fluorescence in the bin. Dimension 1 is the standard error of the mean.

**binned\_dx\_vis:** (bins,2) np.ndarray of running speeds outside of spontaneous activity stimulus. The first bin contains all speeds below 1 cm/s. Dimension 0 is mean running speed in the bin. Dimension 1 is the standard error of the mean.

**binned\_cells\_vis:** np.ndarray of fluorescence outside of spontaneous activity stimulus. First bin contains all data for speeds below 1 cm/s. Dimension 0 is mean fluorescence in the bin. Dimension 1 is the standard error of the mean.

**peak\_run:** pd.DataFrame of speed-related properties of a cell.

**Returns**

**tuple:** `binned_dx_sp`, `binned_cells_sp`, `binned_dx_vis`, `binned_cells_vis`, `peak_run`

**get\_sweep\_response** (*self*)

Calculates the response to each sweep in the stimulus table for each cell and the mean response. The return is a 3-tuple of:

- `sweep_response`: `pd.DataFrame` of response  $dF/F$  traces organized by cell (column) and sweep (row)
- `mean_sweep_response`: mean values of the traces returned in `sweep_response`
- `pval`: p value from 1-way ANOVA comparing response during sweep to response prior to sweep

**Returns**

**3-tuple:** `sweep_response`, `mean_sweep_response`, `pval`

**mean\_sweep\_response**

**numbercells**

**peak**

**peak\_run**

**plot\_representational\_similarity** (*self*, *repsim*, *stimulus=False*)

**plot\_running\_speed\_histogram** (*self*, *xlim=None*, *nbins=None*)

**plot\_speed\_tuning** (*self*, *cell\_specimen\_id=None*, *cell\_index=None*, *evoked\_color='#b30000'*,  
*spontaneous\_color='#0000b3'*)

**populate\_stimulus\_table** (*self*)

Implemented by subclasses.

**pval**

**response**

**roi\_id**

**row\_from\_cell\_id** (*self*, *csid=None*, *idx=None*)

**stim\_table**

**sweep\_response**

**timestamps**

`allensdk.brain_observatory.stimulus_analysis.nonraising_ks_2samp` (*data1*, *data2*,  
*\*\*kwargs*)

`scipy.stats.ks_2samp` now raises a `ValueError` if one of the input arrays is of length 0. Previously it signaled this case by returning nans. This function restores the prior behavior.

**allensdk.brain\_observatory.stimulus\_info module**

**class** `allensdk.brain_observatory.stimulus_info.BinaryIntervalSearchTree` (*search\_list*)

Bases: `object`

**add** (*self*, *input\_list*, *tmp=None*)

**static from\_df** (*input\_df*)

**search** (*self*, *fi*, *tmp=None*)

```

class allensdk.brain_observatory.stimulus_info.BrainObservatoryMonitor (experiment_geometry=None)
    Bases: allensdk.brain_observatory.stimulus_info.Monitor
    http://help.brain-map.org/display/observatory/Documentation?preview=/10616846/10813485/VisualCoding\_VisualStimuli.pdf https://www.cnet.com/products/asus-pa248q/specs/
    grating_to_screen (self, phase, spatial_frequency, orientation, **kwargs)
    lsn_image_to_screen (self, img, **kwargs)
    pixels_to_visual_degrees (self, n, **kwargs)
    visual_degrees_to_pixels (self, vd, **kwargs)
    warp_image (self, img, **kwargs)

class allensdk.brain_observatory.stimulus_info.ExperimentGeometry (distance,
                                                                    mon_height_cm,
                                                                    mon_width_cm,
                                                                    mon_res,
                                                                    eyepoint)
    Bases: object
    generate_warp_coordinates (self)
    warp_coordinates

class allensdk.brain_observatory.stimulus_info.Monitor (n_pixels_r,      n_pixels_c,
                                                         panel_size, spatial_unit)
    Bases: object
    aspect_ratio
    get_mask (self)
    grating_to_screen (self, phase, spatial_frequency, orientation, distance_from_monitor,
                        p2p_amp=256, baseline=127, translation=(0, 0))
    height
    lsn_image_to_screen (self, img, stimulus_type, origin='lower', background_color=127, translation=(0, 0))
    map_stimulus (self, source_stimulus_coordinate, source_stimulus_type, target_stimulus_type)
    mask
    natural_movie_image_to_screen (self, img, origin='lower', translation=(0, 0))
    natural_scene_image_to_screen (self, img, origin='lower', translation=(0, 0))
    panel_size
    pixel_size
    pixels_to_visual_degrees (self, n, distance_from_monitor, small_angle_approximation=True)
    set_spatial_unit (self, new_unit)
    show_image (self, img, ax=None, show=True, mask=False, warp=False, origin='lower')
    spatial_frequency_to_pix_per_cycle (self, spatial_frequency, distance_from_monitor)
    visual_degrees_to_pixels (self, vd, distance_from_monitor,
                             small_angle_approximation=True)
    width

```

```
class allensdk.brain_observatory.stimulus_info.StimulusSearch (nwb_dataset)
    Bases: object
```

```
    search (self, fi)
```

```
allensdk.brain_observatory.stimulus_info.all_stimuli ()
```

```
    Return a list of all stimuli in the data set
```

```
allensdk.brain_observatory.stimulus_info.get_spatial_grating (height=None, aspect_ratio=None,
                                                                ori=None,
                                                                pix_per_cycle=None,
                                                                phase=None,
                                                                p2p_amp=2,
                                                                baseline=0)
```

```
allensdk.brain_observatory.stimulus_info.get_spatio_temporal_grating (t,
                                                                           tempo-
                                                                           ral_frequency=None,
                                                                           **kwargs)
```

```
allensdk.brain_observatory.stimulus_info.lsn_coordinate_to_monitor_coordinate (lsn_coordinate,
                                                                                  mon-
                                                                                  i-
                                                                                  tor_shape,
                                                                                  stim-
                                                                                  u-
                                                                                  lus_type)
```

```
allensdk.brain_observatory.stimulus_info.make_display_mask (display_shape=(1920,
                                                                                1200))
```

```
    Build a display-shaped mask that indicates which pixels are on screen after warping the stimulus.
```

```
allensdk.brain_observatory.stimulus_info.map_monitor_coordinate_to_stimulus_coordinate (moni-
                                                                                          mon-
                                                                                          i-
                                                                                          tor_s-
                                                                                          tim-
                                                                                          u-
                                                                                          lus_t
```

```
allensdk.brain_observatory.stimulus_info.map_monitor_coordinate_to_template_coordinate (moni-
                                                                                          mon-
                                                                                          i-
                                                                                          tor_s-
                                                                                          tem-
                                                                                          plate
```

```
allensdk.brain_observatory.stimulus_info.map_stimulus (source_stimulus_coordinate,
                                                         source_stimulus_type,   tar-
                                                         get_stimulus_type,   moni-
                                                         tor_shape)
```

```
allensdk.brain_observatory.stimulus_info.map_stimulus_coordinate_to_monitor_coordinate (temp-
                                                                                          mon-
                                                                                          i-
                                                                                          tor_s-
                                                                                          stim-
                                                                                          u-
                                                                                          lus_t
```

```
allensdk.brain_observatory.stimulus_info.map_template_coordinate_to_monitor_coordinate(temp-  
mon-  
i-  
tor_s  
tem-  
plate
```

```
allensdk.brain_observatory.stimulus_info.mask_stimulus_template(template_display_coords,  
tem-  
plate_shape,  
dis-  
play_mask=None,  
thresh-  
old=1.0)
```

Build a mask for a stimulus template of a given shape and display coordinates that indicates which part of the template is on screen after warping.

#### Parameters

**template\_display\_coords:** **list** list of (x,y) display coordinates

**template\_shape:** **tuple** (width,height) of the display template

**display\_mask:** **np.ndarray** boolean 2D mask indicating which display coordinates are on screen after warping.

**threshold:** **float** Fraction of pixels associated with a template display coordinate that should remain on screen to count as belonging to the mask.

#### Returns

**tuple:** (template mask, pixel fraction)

```
allensdk.brain_observatory.stimulus_info.monitor_coordinate_to_lsn_coordinate(monitor_coordinate,  
mon-  
i-  
tor_shape,  
stim-  
u-  
lus_type)
```

```
allensdk.brain_observatory.stimulus_info.monitor_coordinate_to_natural_movie_coordinate(mon-  
mon-  
i-  
tor_
```

```
allensdk.brain_observatory.stimulus_info.natural_movie_coordinate_to_monitor_coordinate(natur-  
mon-  
i-  
tor_
```

```
allensdk.brain_observatory.stimulus_info.natural_scene_coordinate_to_monitor_coordinate(natur-  
mon-  
i-  
tor_
```

```
allensdk.brain_observatory.stimulus_info.rotate(X, Y, theta)
```

```
allensdk.brain_observatory.stimulus_info.sessions_with_stimulus(stimulus)
```

Return the names of the sessions that contain a given stimulus.

```
allensdk.brain_observatory.stimulus_info.stimuli_in_session(session, low_unknown=True)
```

Return a list what stimuli are available in a given session.

#### Parameters

**session: string** Must be one of: [stimulus\_info.THREE\_SESSION\_A, stimulus\_info.THREE\_SESSION\_B, stimulus\_info.THREE\_SESSION\_C, stimulus\_info.THREE\_SESSION\_C2]

```
allensdk.brain_observatory.stimulus_info.translate_image_and_fill(img, translation=(0, 0))
```

```
allensdk.brain_observatory.stimulus_info.warp_stimulus_coords(vertices, distance=15.0, mon_height_cm=32.5, mon_width_cm=51.0, mon_res=(1920, 1200), eyepoint=(0.5, 0.5))
```

For a list of screen vertices, provides a corresponding list of texture coordinates.

#### Parameters

**vertices: numpy.ndarray** [[x0,y0], [x1,y1], ...] A set of vertices to convert to texture positions.

**distance: float** distance from the monitor in cm.

**mon\_height\_cm: float** monitor height in cm

**mon\_width\_cm: float** monitor width in cm

**mon\_res: tuple** monitor resolution (x,y)

**eyepoint: tuple**

#### Returns

**np.ndarray** x,y coordinates shaped like the input that describe what pixel coordinates are displayed an the input coordinates after warping the stimulus.

## allensdk.brain\_observatory.sync\_dataset module

dataset.py

**Dataset object for loading and unpacking an HDF5 dataset generated by sync.py**

@author: derrickw

Allen Institute for Brain Science

## Dependencies

numpy <http://www.numpy.org/> h5py <http://www.h5py.org/>

**class** allensdk.brain\_observatory.sync\_dataset.Dataset(path)  
Bases: object

A sync dataset. Contains methods for loading and parsing the binary data.



### Parameters

**path** [str] Path to HDF5 file.

### Examples

```
>>> dset = Dataset('my_h5_file.h5')
>>> logger.info(dset.meta_data)
>>> dset.stats()
>>> dset.close()
```

```
>>> with Dataset('my_h5_file.h5') as d:
...     logger.info(dset.meta_data)
...     dset.stats()
```

**BEHAVIOR\_TRACKING\_KEYS** = ('beh\_frame\_received', 'cam1\_exposure', 'behavior\_monitoring')

**DEPRECATED\_KEYS** = {'behavior\_monitoring', 'cam1\_exposure', 'cam2\_exposure', 'eye\_tracking'}

**EYE\_TRACKING\_KEYS** = ('eye\_frame\_received', 'cam2\_exposure', 'eyetracking', 'eye\_tracking')

**FRAME\_KEYS** = ('frames', 'stim\_vsync')

**OPTOGENETIC\_STIMULATION\_KEYS** = ('LED\_sync', 'opto\_trial')

**PHOTODIODE\_KEYS** = ('photodiode', 'stim\_photodiode')

**analog\_meta\_data**

**close** (*self*)

Closes the dataset.

**duty\_cycle** (*self*, *line*)

Doesn't work right now. Freezes python for some reason.

Returns the duty cycle of a line.

**frequency** (*self*, *line*, *edge*='rising')

Returns the average frequency of a line.

**get\_all\_bits** (*self*)

Returns the data for all bits.

**get\_all\_events** (*self*)

Returns all counter values and their cooresponding IO state.

**get\_all\_times** (*self*, *units*='samples')

Returns all counter values.

### Parameters

**units** [str] Return times in 'samples' or 'seconds'

**get\_analog\_channel** (*self*, *channel*, *start\_time*=0.0, *stop\_time*=None, *downsample*=1)

Returns the data from the specified analog channel between the timepoints.

**Args:** channel (int, str): desired channel index or label start\_time (Optional[float]): start time in seconds  
stop\_time (Optional[float]): stop time in seconds downsample (Optional[int]): downsample factor

**Returns:** ndarray: slice of data for specified channel

**Raises:** KeyError: no analog data present

**get\_analog\_meta** (*self*)

Returns the metadata for the analog data.

**get\_bit** (*self*, *bit*)

Returns the values for a specific bit.

**Parameters**

**bit** [int] Bit to return.

**get\_bit\_changes** (*self*, *bit*)

**Returns the first derivative of a specific bit.** Data points are 1 on rising edges and 255 on falling edges.

**Parameters**

**bit** [int] Bit for which to return changes.

**get\_edges** (*self*, *kind*: str, *keys*: Union[str, Sequence[str]], *units*: str = 'seconds', *permissive*: bool = False) → Union[numpy.ndarray, NoneType]

Utility function for extracting edge times from a line

**Parameters**

**kind** [One of “rising”, “falling”, or “all”. Should this method return] timestamps for rising, falling or both edges on the appropriate line

**keys** [These will be checked in sequence. Timestamps will be returned] for the first which is present in the line labels

**units** [one of “seconds”, “samples”, or “indices”. The returned] “time”stamps will be given in these units.

**raise\_missing** [If True and no matching line is found, a KeyError will] be raised

**Returns**

**An array of edge times. If raise\_missing is False and none of the keys were found,** returns None.

**Raises**

**KeyError** [none of the provided keys were found among this dataset’s] line labels

**get\_events\_by\_bit** (*self*, *bit*, *units*=‘samples’)

**Returns all counter values for transitions (both rising and falling)** for a specific bit.

**Parameters**

**bit** [int] Bit for which to return events.

**get\_events\_by\_line** (*self*, *line*, *units*=‘samples’)

**Returns all counter values for transitions (both rising and falling)** for a specific line.

**Parameters**

**line** [str] Line for which to return events.

**get\_falling\_edges** (*self*, *line*, *units*=‘samples’)

**Returns the counter values for the falling edges for a specific bit** or line.

**Parameters**

**line** [str] Line for which to return edges.

**get\_line** (*self*, *line*)

Returns the values for a specific line.

#### Parameters

**line** [str] Line to return.

**get\_line\_changes** (*self*, *line*)

**Returns the first derivative of a specific line.** Data points are 1 on rising edges and 255 on falling edges.

#### Parameters

**line** [(str)] Line name for which to return changes.

**get\_nearest** (*self*, *source*, *target*, *source\_edge*='rising', *target\_edge*='rising', *direction*='previous', *units*='indices')

**For all values of the source line, finds the nearest edge from the** target line.

By default, returns the indices of the target edges.

**Args:** *source* (str, int): desired source line *target* (str, int): desired target line *source\_edge* [Optional(str)]: “rising” or “falling” source edges *target\_edge* [Optional(str): “rising” or “falling” target edges *direction* (str): “previous” or “next”. Whether to prefer the previous edge or the following edge.  
*units* (str): “indices”

**get\_rising\_edges** (*self*, *line*, *units*='samples')

**Returns the counter values for the rizing edges for a specific bit or** line.

#### Parameters

**line** [str] Line for which to return edges.

**line\_stats** (*self*, *line*, *print\_results*=True)

Quick-and-dirty analysis of a bit.

##TODO: Split this up into smaller functions.

**load** (*self*, *path*)

Loads an hdf5 sync dataset.

#### Parameters

**path** [str] Path to hdf5 file.

**period** (*self*, *line*, *edge*='rising')

Returns a dictionary with avg, min, max, and st of period for a line.

**plot\_all** (*self*, *start\_time*, *stop\_time*, *auto\_show*=True)

Plot all active bits.

Yikes. Come up with a better way to show this.

**plot\_bit** (*self*, *bit*, *start\_time*=0.0, *end\_time*=None, *auto\_show*=True, *axes*=None, *name*="")

Plots a specific bit at a specific time period.

**plot\_bits** (*self*, *bits*, *start\_time*=0.0, *end\_time*=None, *auto\_show*=True)  
Plots a list of bits.

**plot\_line** (*self*, *line*, *start\_time*=0.0, *end\_time*=None, *auto\_show*=True)  
Plots a specific line at a specific time period.

**plot\_lines** (*self*, *lines*, *start\_time*=0.0, *end\_time*=None, *auto\_show*=True)  
Plots specific lines at a specific time period.

**sample\_freq**

**stats** (*self*)

**Quick-and-dirty analysis of all bits. Prints a few things about each** bit where events are found.

`allensdk.brain_observatory.sync_dataset.get_bit (uint_array, bit)`  
Returns a bool array for a specific bit in a uint ndarray.

#### Parameters

**uint\_array** [(numpy.ndarray)] The array to extract bits from.

**bit** [(int)] The bit to extract.

`allensdk.brain_observatory.sync_dataset.unpack_uint32 (uint32_array, endian='L')`  
Unpacks an array of 32-bit unsigned integers into bits.

Default is least significant bit first.

**\*Not currently used by sync dataset because get\_bit is better and does** basically the same thing. I'm just leaving it in because it could potentially account for endianness and possibly have other uses in the future.

## Module contents

**class** `allensdk.brain_observatory.JSONEncoder` (\*, *skipkeys*=False, *ensure\_ascii*=True, *check\_circular*=True, *allow\_nan*=True, *sort\_keys*=False, *indent*=None, *separators*=None, *default*=None)

Bases: `json.encoder.JSONEncoder`

**default** (*self*, *o*)

Implement this method in a subclass such that it returns a serializable object for *o*, or calls the base implementation (to raise a `TypeError`).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

`allensdk.brain_observatory.dict_to_indexed_array (dc, order=None)`

Given a dictionary and an ordered arr, build a concatenation of the dictionary's values and an index describing how that concatenation can be unpacked

`allensdk.brain_observatory.hook (json_dict)`

## 6.1.3 allensdk.config package

### Subpackages

#### allensdk.config.app package

### Submodules

#### allensdk.config.app.application\_config module

```
class allensdk.config.app.application_config.ApplicationConfig(defaults,
                                                                name='app',
                                                                help='Run ap-
                                                                plication.', de-
                                                                fault_log_config=None)
```

Bases: object

Convenience class that handles of application configuration from environment variables, .conf files and the command line using Python standard libraries and formats.

**apply\_configuration\_from\_command\_line** (*self*, *parsed\_args*)

Read application configuration variables from the command line.

Unassigned variables are left unchanged if previously assigned, set to their default values, or None if no default is specified at init time. Assigned variables will overwrite the previous value.

see: <https://docs.python.org/2/howto/argparse.html>

#### Parameters

**parsed\_args** [dict] the arguments as parsed from the command line.

**apply\_configuration\_from\_environment** (*self*)

Read application configuration variables from the environment.

The variable names are upper case and have a prefix defined by the application.

See: <https://docs.python.org/2/library/os.html>

**apply\_configuration\_from\_file** (*self*, *config\_file\_path*)

Read application configuration variables from a .conf file.

Unassigned variables are set to their default values or None if no default is specified at init time. The variables are found in a section named by the application.

#### Parameters

**config\_file\_path** [string] path to to an INI (.conf) or JSON format application config file.

#### Returns

see: <https://docs.python.org/2/library/configparser.html>

**create\_argparser** (*self*)

Initialization for the command-line parsing stage.

An application specific prefix is applied to argument names.

#### Parameters

**prog** [string] Application specific prefix for argument names.

**description** [string] A brief 'help' description of the application.

**Returns**

**argParse.ArgumentParser** The initialized argument parser object.

**Notes**

Defaults are set at the first environment reading. Command line args only override them when present

**from\_json\_file** (*self*, *json\_path*)

Read an application configuration from a JSON format file.

**Parameters**

**json\_path** [string] Path to the JSON file.

**Returns**

**string** An application configuration in INI format

**from\_json\_string** (*self*, *json\_string*)

Read a configuration from a JSON format string.

**Parameters**

**json\_string** [string] A JSON-formatted string containing an application configuration.

**Returns**

**string** An application configuration in INI format

**load** (*self*, *command\_line\_args*, *disable\_existing\_loggers=True*)

Load application configuration options, first from the environment, then from the configuration file, then from the command line.

Each stage of loading can override the previous stage.

**Parameters**

**command\_line\_args** [dict] Parameters passed to the application.

**disable\_existing\_loggers** [boolean] Reset the logging system or not.

**Returns**

**fileConfig** Configuration object with all levels applied

**parse\_command\_line\_args** (*self*, *args*)

Simply call the internal argparser object.

**Parameters**

**args** [array] Parameters passed to the application.

**Returns**

**Namespace** Parsed parameters.

**to\_config\_string** (*self*, *description*)

Create a configuration string from a dict.

**Parameters**

**description** [dict] Configuration options for an application.

**Returns**

**string** Equivalent configuration as an INI format string

## Notes

The Python configparser library natively supports this functionality in Python 3.

## Module contents

`allensdk.config.app` is a package that assists in configuring application software, as opposed to domain-specific configuration.

### `allensdk.config.model` package

#### Subpackages

### `allensdk.config.model.formats` package

#### Submodules

### `allensdk.config.model.formats.hdf5_util` module

```

class allensdk.config.model.formats.hdf5_util.Hdf5Util
    Bases: object
    read(self, file_path)
    write(self, file_path, m)

```

### `allensdk.config.model.formats.json_description_parser` module

```

class allensdk.config.model.formats.json_description_parser.JsonDescriptionParser
    Bases: allensdk.config.model.description_parser.DescriptionParser
    log = <Logger allensdk.config.model.formats.json_description_parser (WARNING)>
    read(self, file_path, description=None, section=None, **kwargs)
        Parse a complete or partial configuration.

```

#### Parameters

- json\_string** [string] Input to parse.
- description** [Description, optional] Where to put the parsed configuration. If None a new one is created.
- section** [string, optional] Where to put the parsed configuration within the description.

#### Returns

- Description** The input description with parsed configuration added.
- Section is only specified for “bare” objects that are to be added to a section array.**

```

read_string(self, json_string, description=None, section=None, **kwargs)
    Parse a complete or partial configuration.

```

#### Parameters

- json\_string** [string] Input to parse.

**description** [Description, optional] Where to put the parsed configuration. If None a new one is created.

**section** [string, optional] Where to put the parsed configuration within the description.

#### Returns

**Description** The input description with parsed configuration added.

**Section is only specified for “bare” objects that are to be added to a section array.**

**write** (*self*, *filename*, *description*)

Write the description to a JSON file.

#### Parameters

**description** [Description] Object to write.

**write\_string** (*self*, *description*)

Write the description to a JSON string.

#### Parameters

**description** [Description] Object to write.

#### Returns

**string** JSON serialization of the input.

### allensdk.config.model.formats.pycfg\_description\_parser module

**class** allensdk.config.model.formats.pycfg\_description\_parser.**PycfgDescriptionParser**  
Bases: *allensdk.config.model.description\_parser.DescriptionParser*

**log** = <Logger allensdk.config.model.formats.pycfg\_description\_parser (WARNING)>

**read** (*self*, *pycfg\_file\_path*, *description=None*, *section=None*, *\*\*kwargs*)

Read a serialized description from a Python (.pycfg) file.

#### Parameters

**filename** [string] Name of the .pycfg file.

#### Returns

**Description** Configuration object.

**read\_string** (*self*, *python\_string*, *description=None*, *section=None*, *\*\*kwargs*)

Read a serialized description from a Python (.pycfg) string.

#### Parameters

**python\_string** [string] Python string with a serialized description.

#### Returns

**Description** Configuration object.

**write** (*self*, *filename*, *description*)

Write the description to a Python (.pycfg) file.

#### Parameters

**filename** [string] Name of the file to write.



**write\_string** (*self*, *description*)

Write the description to a pretty-printed Python string.

**Parameters**

**description** [Description] Configuration object to write.

## Module contents

## Submodules

### allensdk.config.model.description module

**class** allensdk.config.model.description.**Description**

Bases: object

**fix\_unary\_sections** (*self*, *section\_names=None*)

Wrap section contents that don't have the proper array surrounding them in an array.

**Parameters**

**section\_names** [list of strings, optional] Keys of sections that might not be in array form.

**is\_empty** (*self*)

Check if anything is in the object.

**Returns**

**boolean** true if self.data is missing or empty

**unpack** (*self*, *data*, *section=None*)

Read the manifest and other stand-alone configuration structure, or insert a configuration object into a section of an existing configuration.

**Parameters**

**data** [dict] A configuration object including top level sections, or an configuration object to be placed within a section.

**section** [string, optional.] If this is present, place data within an existing section array.

**unpack\_manifest** (*self*, *data*)

Pull the manifest configuration section into a separate place.

**Parameters**

**data** [dict] A configuration structure that still has a manifest section.

**update\_data** (*self*, *data*, *section=None*)

Merge configuration data possibly from multiple files.

**Parameters**

**data** [dict] Configuration structure to add.

**section** [string, optional] What configuration section to read it into if the file does not specify.

## allensdk.config.model.description\_parser module

```
class allensdk.config.model.description_parser.DescriptionParser
```

Bases: object

```
log = <Logger allensdk.config.model.description_parser (WARNING)>
```

```
parser_for_extension (self, filename)
```

Choose a subclass that can read the format.

### Parameters

**filename** [string] For the extension.

### Returns

**DescriptionParser** Appropriate subclass.

```
read (self, file_path, description=None, section=None, **kwargs)
```

Parse data needed for a simulation.

### Parameters

**description** [dict] Configuration from parsing previous files.

**section** [string, optional] What configuration section to read it into if the file does not specify.

```
read_string (self, data_string, description=None, section=None, header=None)
```

Parse data needed for a simulation from a string.

```
write (self, filename, description)
```

Save the configuration.

### Parameters

**filename** [string] Name of the file to write.

## Module contents

### Submodules

## allensdk.config.manifest module

```
class allensdk.config.manifest.Manifest (config=None,      relative_base_dir='.',      ver-
```

```
sion=None)
```

Bases: object

Manages the location of external files referenced in an Allen SDK configuration

```
DIR = 'dir'
```

```
DIRNAME = 'dir_name'
```

```
FILE = 'file'
```

```
VERSION = 'manifest_version'
```

```
add_file (self, file_key, file_name, dir_key=None, path_format=None)
```

Insert a new file entry.

### Parameters

**file\_key** [string] Reference to the entry.

**file\_name** [string] Substitutions of the %s, %d style allowed.

**dir\_key** [string] Reference to the parent directory entry.

**path\_format** [string, optional] File type for further parsing.

**add\_path** (*self*, *key*, *path*, *path\_type*=*'dir'*, *absolute*=*True*, *path\_format*=*None*, *parent\_key*=*None*)  
Insert a new entry.

#### Parameters

**key** [string] Identifier for referencing the entry.

**path** [string] Specification for a path using %s, %d style substitution.

**path\_type** [string enumeration] *'dir'* (default) or *'file'*

**absolute** [boolean] Is the spec relative to the process current directory.

**path\_format** [string, optional] Indicate a known file type for further parsing.

**parent\_key** [string] Refer to another entry.

**add\_paths** (*self*, *path\_info*)  
add information about paths stored in the manifest.

#### Parameters

**path\_info** [dict] Information about the new paths

**as\_dataframe** (*self*)

**check\_dir** (*self*, *path\_key*, *do\_exit*=*False*)  
Verify a directories existence or optionally exit.

#### Parameters

**path\_key** [string] Reference to the entry.

**do\_exit** [boolean] What to do if the directory is not present.

**create\_dir** (*self*, *path\_key*)  
Make a directory for an entry.

#### Parameters

**path\_key** [string] Reference to the entry.

**get\_format** (*self*, *path\_key*)  
Retrieve the type of a path entry.

#### Parameters

**path\_key** [string] reference to the entry

#### Returns

**string** File type.

**get\_path** (*self*, *path\_key*, *\*args*)  
Retrieve an entry with substitutions.

#### Parameters

**path\_key** [string] Refer to the entry to retrieve.

**args** [any types, optional] arguments to be substituted into the path spec for %s, %d, etc.

**Returns**

**string** Path with parent structure and substitutions applied.

**load\_config** (*self*, *config*, *version=None*)

Load paths into the manifest from an Allen SDK config section.

**Parameters**

**config** [Config] Manifest section of an Allen SDK config.

**log** = <Logger allensdk.config.manifest (WARNING)>

**resolve\_paths** (*self*, *description\_dict*, *suffix='\_key'*)

Walk input items and expand those that refer to a manifest entry.

**Parameters**

**description\_dict** [dict] Any entries with key names ending in suffix will be expanded.

**suffix** [string] Indicates the entries to be expanded.

**classmethod safe\_make\_parent\_dirs** (*file\_name*)

Create a parent directories for file.

**Parameters**

**file\_name** [string]

**Returns**

**leftmost** [string] most rootward directory created

**classmethod safe\_mkdir** (*directory*)

Create path if not already there.

**Parameters**

**directory** [string] create it if it doesn't exist

**Returns**

**leftmost** [string] most rootward directory created

**exception** allensdk.config.manifest.**ManifestVersionError** (*message*, *version*,  
*found\_version*)

Bases: Exception

**outdated**

**allensdk.config.manifest\_builder module**

**class** allensdk.config.manifest\_builder.**ManifestBuilder**

Bases: object

**add\_path** (*self*, *key*, *spec*, *typename='dir'*, *parent\_key=None*, *format=None*)

**add\_section** (*self*, *name*, *contents*)

**as\_dataframe** (*self*)

**df\_columns** = ['key', 'parent\_key', 'spec', 'type', 'format']

**from\_dataframe** (*self*, *df*)

**get\_config** (*self*)

```
get_manifest (self)
set_version (self, value)
write_json_file (self, path, overwrite=False)
write_json_string (self)
```

## Module contents

`allensdk.config.enable_console_log (level=None)`  
configure allensdk logging to output to the console.

### Parameters

**level** [int] logging level 0-50 (logging.INFO, logging.DEBUG, etc.)

### Notes

See: [Logging Cookbook](#)

## 6.1.4 allensdk.core package

### Subpackages

`allensdk.core.lazy_property` package

### Submodules

`allensdk.core.lazy_property.lazy_property` module

```
class allensdk.core.lazy_property.lazy_property.LazyProperty (api_method, wrap-
                                                                pers=(),      *args,
                                                                **kwargs)
    Bases: object
    calculate (self)
```

`allensdk.core.lazy_property.lazy_property_mixin` module

```
class allensdk.core.lazy_property.lazy_property_mixin.LazyPropertyMixin
    Bases: object
    LazyProperty
```

## Module contents

### Submodules

`allensdk.core.auth_config` module

**allensdk.core.authentication module**

```
class allensdk.core.authentication.CredentialProvider
```

```
    Bases: abc.ABC
```

```
    METHOD = 'custom'
```

```
    provide (self, credential)
```

```
class allensdk.core.authentication.DbCredentials (dbname, user, host, port, password)
```

```
    Bases: tuple
```

```
    dbname
```

```
        Alias for field number 0
```

```
    host
```

```
        Alias for field number 2
```

```
    password
```

```
        Alias for field number 4
```

```
    port
```

```
        Alias for field number 3
```

```
    user
```

```
        Alias for field number 1
```

```
class allensdk.core.authentication.EnvCredentialProvider (environ:          Optional[Dict[str, Any]] =  
                                                         None)
```

```
    Bases: allensdk.core.authentication.CredentialProvider
```

```
    Provides credentials from environment variables for variables listed in CREDENTIAL_KEYS.
```

```
    METHOD = 'env'
```

```
    provide (self, credential)
```

```
allensdk.core.authentication.credential_injector (credential_map:          Dict[str,  
                                                         Any],                provider:  
                                                         Union[allensdk.core.authentication.CredentialProvider,  
                                                         NoneType] = None)
```

Decorator used to inject credentials from another source if not explicitly provided in the function call. This function will only supply values for keyword arguments. All keys defined in *credential\_map* must correspond to keyword arguments in the function signature.

**Parameters**

**credential\_map:** **Dict[Str: Any]** Dictionary where the keys are the keyword of a credential kwarg passed to the decorated function, and the values are the name of the credential in the credential provider (see CREDENTIAL\_KEYS).

Example of credential\_map for PostgresQueryMixin connecting to LIMS database:

```
{ "dbname": "LIMS_DBNAME", "user": "LIMS_USER", "host":  
  "LIMS_HOST", "password": "LIMS_PASSWORD", "port": "LIMS_PORT"  
}
```

**provider:** **Optional[CredentialProvider]** Subclass of CredentialProvider to provide credentials to the wrapped function. If left unspecified, will default to EnvCredentialProvider, which provides credentials from environment variables.

```
allensdk.core.authentication.get_credential_provider ()
```

`allensdk.core.authentication.set_credential_provider(provider)`

## allensdk.core.brain\_observatory\_cache module

```
class allensdk.core.brain_observatory_cache.BrainObservatoryCache(cache=True,
                                                                    mani-
                                                                    fest_file=None,
                                                                    base_uri=None,
                                                                    api=None)
```

Bases: `allensdk.api.cache.Cache`

Cache class for storing and accessing data from the Brain Observatory. By default, this class will cache any downloaded metadata or files in well known locations defined in a manifest file. This behavior can be disabled.

### Parameters

**cache: boolean** Whether the class should save results of API queries to locations specified in the manifest file. Queries for files (as opposed to metadata) must have a file location. If caching is disabled, those locations must be specified in the function call (e.g. `get_ophys_experiment_data(file_name='file.nwb')`).

**manifest\_file: string** File name of the manifest to be read. Default is "brain\_observatory\_manifest.json".

### Attributes

**api: BrainObservatoryApi instance** The object used for making API queries related to the Brain Observatory.

```
ANALYSIS_DATA_KEY = 'ANALYSIS_DATA'
CELL_SPECIMENS_KEY = 'CELL_SPECIMENS'
EVENTS_DATA_KEY = 'EVENTS_DATA'
EXPERIMENTS_KEY = 'EXPERIMENTS'
EXPERIMENT_CONTAINERS_KEY = 'EXPERIMENT_CONTAINERS'
EXPERIMENT_DATA_KEY = 'EXPERIMENT_DATA'
EYE_GAZE_DATA_KEY = 'EYE_GAZE_DATA'
MANIFEST_VERSION = '1.3'
STIMULUS_MAPPINGS_KEY = 'STIMULUS_MAPPINGS'
```

**build\_manifest(self, file\_name)**

Construct a manifest for this Cache class and save it in a file.

### Parameters

**file\_name: string** File location to save the manifest.

**get\_all\_cre\_lines(self)**

Return a list of all cre driver lines in the data set.

**get\_all\_imaging\_depths(self)**

Return a list of all imaging depths in the data set.

**get\_all\_reporter\_lines(self)**

Return a list of all reporter lines in the data set.

**get\_all\_session\_types** (*self*)

Return a list of all stimulus sessions in the data set.

**get\_all\_stimuli** (*self*)

Return a list of all stimuli in the data set.

**get\_all\_targeted\_structures** (*self*)

Return a list of all targeted structures in the data set.

**get\_cell\_specimens** (*self*, *file\_name=None*, *ids=None*, *experiment\_container\_ids=None*, *include\_failed=False*, *simple=True*, *filters=None*)

Return cell specimens that have certain properties.

#### Parameters

**file\_name: string** File name to save/read the cell specimens. If *file\_name* is *None*, the *file\_name* will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is *None*.

**ids: list** List of cell specimen ids.

**experiment\_container\_ids: list** List of experiment container ids.

**include\_failed: bool** Whether to include cells from failed experiment containers

**simple: boolean** Whether or not to simplify the dictionary properties returned by this method to a more concise subset.

**filters: list of dicts** List of filter dictionaries. The Allen Brain Observatory web site can generate filters in this format to reproduce a filtered set of cells found there. To see what these look like, visit <http://observatory.brain-map.org/visualcoding>, perform a cell search and apply some filters (e.g. find cells in a particular area), then click the “view these cells in the AllenSDK” link on the bottom-left of the search results page. This will take you to a page that contains a code sample you can use to apply those same filters via this argument. For more detail on the filter syntax, see `BrainObservatoryApi.dataframe_query`.

#### Returns

**list of dictionaries**

**get\_experiment\_containers** (*self*, *file\_name=None*, *ids=None*, *targeted\_structures=None*, *imaging\_depths=None*, *cre\_lines=None*, *reporter\_lines=None*, *transgenic\_lines=None*, *include\_failed=False*, *simple=True*)

Get a list of experiment containers matching certain criteria.

#### Parameters

**file\_name: string** File name to save/read the experiment containers. If *file\_name* is *None*, the *file\_name* will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is *None*.

**ids: list** List of experiment container ids.

**targeted\_structures: list** List of structure acronyms. Must be in the list returned by `BrainObservatoryCache.get_all_targeted_structures()`.

**imaging\_depths: list** List of imaging depths. Must be in the list returned by `BrainObservatoryCache.get_all_imaging_depths()`.

**cre\_lines: list** List of cre lines. Must be in the list returned by `BrainObservatoryCache.get_all_cre_lines()`.



**reporter\_lines: list** List of reporter lines. Must be in the list returned by `BrainObservatoryCache.get_all_reporter_lines()`.

**transgenic\_lines: list** List of transgenic lines. Must be in the list returned by `BrainObservatoryCache.get_all_cre_lines()` or `BrainObservatoryCache.get_all_reporter_lines()`.

**include\_failed: boolean** Whether or not to include failed experiment containers.

**simple: boolean** Whether or not to simplify the dictionary properties returned by this method to a more concise subset.

### Returns

list of dictionaries

**get\_nwb\_filepath** (*self*, *ophys\_experiment\_id=None*)

**get\_ophys\_experiment\_analysis** (*self*, *ophys\_experiment\_id*, *stimulus\_type*,  
*file\_name=None*)

Download the h5 analysis file for a stimulus set, for a particular `ophys_experiment` (if it hasn't already been downloaded) and return a data accessor object.

### Parameters

**file\_name: string** File name to save/read the data set. If `file_name` is `None`, the `file_name` will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is `None`.

**ophys\_experiment\_id: int** id of the `ophys_experiment` to retrieve

**stimulus\_name: str** stimulus type; should be an element of `self.list_stimuli()`

### Returns

`BrainObservatoryNwbDataSet`

**get\_ophys\_experiment\_data** (*self*, *ophys\_experiment\_id*, *file\_name=None*)

Download the NWB file for an `ophys_experiment` (if it hasn't already been downloaded) and return a data accessor object.

### Parameters

**file\_name: string** File name to save/read the data set. If `file_name` is `None`, the `file_name` will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is `None`.

**ophys\_experiment\_id: integer** id of the `ophys_experiment` to retrieve

### Returns

`BrainObservatoryNwbDataSet`

**get\_ophys\_experiment\_events** (*self*, *ophys\_experiment\_id*, *file\_name=None*)

Download the npz events file for an `ophys_experiment` if it hasn't already been downloaded and return the events array.

### Parameters

**file\_name: string** File name to save/read the data set. If `file_name` is `None`, the `file_name` will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is `None`.

**ophys\_experiment\_id: int** id of the `ophys_experiment` to retrieve events for

### Returns

\_\_\_\_\_

**events:** `numpy.ndarray` [N\_cells,N\_times] array of events.

**get\_ophys\_experiment\_stimuli** (*self*, *experiment\_id*)

For a single experiment, return the list of stimuli present in that experiment.

**get\_ophys\_experiments** (*self*, *file\_name=None*, *ids=None*, *experiment\_container\_ids=None*, *targeted\_structures=None*, *imaging\_depths=None*, *cre\_lines=None*, *reporter\_lines=None*, *transgenic\_lines=None*, *stimuli=None*, *session\_types=None*, *cell\_specimen\_ids=None*, *include\_failed=False*, *require\_eye\_tracking=False*, *simple=True*)

Get a list of ophys experiments matching certain criteria.

#### Parameters

**file\_name:** `string` File name to save/read the ophys experiments. If *file\_name* is `None`, the *file\_name* will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is `None`.

**ids:** `list` List of ophys experiment ids.

**experiment\_container\_ids:** `list` List of experiment container ids.

**targeted\_structures:** `list` List of structure acronyms. Must be in the list returned by `BrainObservatoryCache.get_all_targeted_structures()`.

**imaging\_depths:** `list` List of imaging depths. Must be in the list returned by `BrainObservatoryCache.get_all_imaging_depths()`.

**cre\_lines:** `list` List of cre lines. Must be in the list returned by `BrainObservatoryCache.get_all_cre_lines()`.

**reporter\_lines:** `list` List of reporter lines. Must be in the list returned by `BrainObservatoryCache.get_all_reporter_lines()`.

**transgenic\_lines:** `list` List of transgenic lines. Must be in the list returned by `BrainObservatoryCache.get_all_cre_lines()` or `BrainObservatoryCache.get_all_reporter_lines()`.

**stimuli:** `list` List of stimulus names. Must be in the list returned by `BrainObservatoryCache.get_all_stimuli()`.

**session\_types:** `list` List of stimulus session type names. Must be in the list returned by `BrainObservatoryCache.get_all_session_types()`.

**cell\_specimen\_ids:** `list` Only include experiments that contain cells with these ids.

**include\_failed:** `boolean` Whether or not to include experiments from failed experiment containers.

**simple:** `boolean` Whether or not to simplify the dictionary properties returned by this method to a more concise subset.

**require\_eye\_tracking:** `boolean` If `True`, only return experiments that have eye tracking results. Default: `False`.

#### Returns

`list of dictionaries`

**get\_ophys\_pupil\_data** (*self*, *ophys\_experiment\_id: int*, *file\_name: str = None*, *suppress\_pupil\_data: bool = True*) → `pandas.core.frame.DataFrame`

Download the h5 eye gaze mapping file for an ophys\_experiment if it hasn't already been downloaded and return it as a `pandas.DataFrame`.

#### Parameters

**file\_name: string** File name to save/read the data set. If file\_name is None, the file\_name will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is None.

**ophys\_experiment\_id: int** id of the ophys\_experiment to retrieve pupil data for.

**suppress\_pupil\_data: bool** Whether or not to suppress pupil data from dataset. Default is True.

#### Returns

**pd.DataFrame**

If 'suppress\_eye\_gaze\_data' is set to 'False':

**Contains raw/filtered columns for gaze mapping:** \*\_eye\_area  
\*\_pupil\_area      \*\_screen\_coordinates\_x\_cm      \*\_screen\_coordinates\_y\_cm  
\*\_screen\_coordinates\_spherical\_x\_deg      \*\_screen\_coorindates\_spherical\_y\_deg

**Otherwise:** An empty pandas DataFrame

### allensdk.core.brain\_observatory\_nwb\_data\_set module

```
class allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet (nwb_file)
    Bases: object
```

```
FILE_METADATA_MAPPING = {'age': 'general/subject/age', 'device_string': 'general/dev
```

```
MOTION_CORRECTION_DATASETS = ['MotionCorrection/2p_image_series/xy_translations', 'Mot
```

```
PIPELINE_DATASET = 'brain_observatory_pipeline'
```

```
STIMULUS_TABLE_TYPES = {'abstract_feature_series': ['drifting_gratings', 'static_grat
```

```
SUPPORTED_PIPELINE_VERSION = '3.0'
```

```
get_cell_specimen_ids (self)
```

Returns an array of cell IDs for all cells in the file

#### Returns

**cell specimen IDs: list**

```
get_cell_specimen_indices (self, cell_specimen_ids)
```

Given a list of cell specimen ids, return their index based on their order in this file.

#### Parameters

**cell\_specimen\_ids: list of cell specimen ids**

```
get_corrected_fluorescence_traces (self, cell_specimen_ids=None)
```

Returns an array of demixed and neuropil-corrected fluorescence traces for all ROIs and the timestamps for each datapoint

#### Parameters

**cell\_specimen\_ids: list or array (optional)** List of cell IDs to return traces for. If this is None (default) then all are returned

#### Returns

**timestamps: 2D numpy array** Timestamp for each fluorescence sample

**traces: 2D numpy array** Corrected fluorescence traces for each cell

**get\_demixed\_traces** (*self*, *cell\_specimen\_ids=None*)

Returns an array of demixed fluorescence traces for all ROIs and the timestamps for each datapoint

**Parameters**

**cell\_specimen\_ids: list or array (optional)** List of cell IDs to return traces for. If this is None (default) then all are returned

**Returns**

**timestamps: 2D numpy array** Timestamp for each fluorescence sample

**traces: 2D numpy array** Demixed fluorescence traces for each cell

**get\_dff\_traces** (*self*, *cell\_specimen\_ids=None*)

Returns an array of dF/F traces for all ROIs and the timestamps for each datapoint

**Parameters**

**cell\_specimen\_ids: list or array (optional)** List of cell IDs to return data for. If this is None (default) then all are returned

**Returns**

**timestamps: 2D numpy array** Timestamp for each fluorescence sample

**dF/F: 2D numpy array** dF/F values for each cell

**get\_fluorescence\_timestamps** (*self*)

Returns an array of timestamps in seconds for the fluorescence traces

**get\_fluorescence\_traces** (*self*, *cell\_specimen\_ids=None*)

Returns an array of fluorescence traces for all ROI and the timestamps for each datapoint

**Parameters**

**cell\_specimen\_ids: list or array (optional)** List of cell IDs to return traces for. If this is None (default) then all are returned

**Returns**

**timestamps: 2D numpy array** Timestamp for each fluorescence sample

**traces: 2D numpy array** Fluorescence traces for each cell

**get\_locally\_sparse\_noise\_stimulus\_template** (*self*, *stimulus*, *mask\_off\_screen=True*)

Return an array of the stimulus template for the specified stimulus.

**Parameters**

**stimulus: string**

**Which locally sparse noise stimulus to retrieve. Must be one of:**

stimulus\_info.LOCALLY\_SPARSE\_NOISE stimu-

lus\_info.LOCALLY\_SPARSE\_NOISE\_4DEG stimu-

lus\_info.LOCALLY\_SPARSE\_NOISE\_8DEG

**mask\_off\_screen: boolean** Set off-screen regions of the stimulus to LocallySparseNoise.LSN\_OFF\_SCREEN.

**Returns**

**tuple: (template, off-screen mask)**

**get\_max\_projection** (*self*)

Returns the maximum projection image for the 2P movie.

**Returns****max projection: np.ndarray****get\_metadata** (*self*)

Returns a dictionary of meta data associated with each experiment, including Cre line, specimen number, visual area imaged, imaging depth

**Returns****metadata: dictionary****get\_motion\_correction** (*self*)

Returns a Panda DataFrame containing the x- and y- translation of each image used for image alignment

**get\_neuropil\_r** (*self*, *cell\_specimen\_ids=None*)

Returns a scalar value of r for neuropil correction of fluorescence traces

**Parameters****cell\_specimen\_ids: list or array (optional)** List of cell IDs to return traces for. If this is None (default) then results for all are returned**Returns****r: 1D numpy array, len(r)=len(cell\_specimen\_ids)** Scalar for neuropil subtraction for each cell**get\_neuropil\_traces** (*self*, *cell\_specimen\_ids=None*)

Returns an array of neuropil fluorescence traces for all ROIs and the timestamps for each datapoint

**Parameters****cell\_specimen\_ids: list or array (optional)** List of cell IDs to return traces for. If this is None (default) then all are returned**Returns****timestamps: 2D numpy array** Timestamp for each fluorescence sample**traces: 2D numpy array** Neuropil fluorescence traces for each cell**get\_pupil\_location** (*self*, *as\_spherical=True*)

Returns the x, y pupil location.

**Parameters****as\_spherical** [bool] Whether to return the location as spherical (default) or not. If true, the result is altitude and azimuth in degrees, otherwise it is x, y in centimeters. (0,0) is the center of the monitor.**Returns****(timestamps, location)** Timestamps is an (Nx1) array of timestamps in seconds. Location is an (Nx2) array of spatial location.**get\_pupil\_size** (*self*)

Returns the pupil area in pixels.

**Returns****(timestamps, areas)** Timestamps is an (Nx1) array of timestamps in seconds. Areas is an (Nx1) array of pupil areas in pixels.**get\_roi\_ids** (*self*)

Returns an array of IDs for all ROIs in the file

**Returns****ROI IDs: list****get\_roi\_mask** (*self*, *cell\_specimen\_ids=None*)

Returns an array of all the ROI masks

**Parameters****cell specimen IDs: list or array (optional)** List of cell IDs to return traces for. If this is None (default) then all are returned**Returns****List of ROI\_Mask objects****get\_roi\_mask\_array** (*self*, *cell\_specimen\_ids=None*)Return a numpy array containing all of the ROI masks for requested cells. If *cell\_specimen\_ids* is omitted, return all masks.**Parameters****cell\_specimen\_ids: list** List of cell specimen ids. Default None.**Returns****np.ndarray: NxWxH array, where N is number of cells****get\_running\_speed** (*self*)

Returns the mouse running speed in cm/s

**get\_session\_type** (*self*)Returns the type of experimental session, presently one of the following: *three\_session\_A*, *three\_session\_B*, *three\_session\_C***Returns****session type: string****get\_stimulus** (*self*, *frame\_ind*)**get\_stimulus\_epoch\_table** (*self*)

Returns a pandas dataframe that summarizes the stimulus epoch duration for each acquisition time index in the experiment

**Parameters****None****Returns****timestamps: 2D numpy array** Timestamp for each fluorescence sample**traces: 2D numpy array** Fluorescence traces for each cell**get\_stimulus\_table** (*self*, *stimulus\_name*)

Return a stimulus table given a stimulus name

**Notes**For more information, see: [http://help.brain-map.org/display/observatory/Documentation?preview=/10616846/10813485/VisualCoding\\_VisualStimuli.pdf](http://help.brain-map.org/display/observatory/Documentation?preview=/10616846/10813485/VisualCoding_VisualStimuli.pdf)**get\_stimulus\_template** (*self*, *stimulus\_name*)

Return an array of the stimulus template for the specified stimulus.

**Parameters**

**stimulus\_name: string** Must be one of the strings returned by `list_stimuli()`.

**Returns**

**stimulus table: pd.DataFrame**

**list\_stimuli** (*self*)

Return a list of the stimuli presented in the experiment.

**Returns**

**stimuli: list of strings**

**number\_of\_cells**

Number of cells in the experiment

**save\_analysis\_arrays** (*self*, \**datasets*)

**save\_analysis\_dataframes** (*self*, \**tables*)

**stimulus\_search**

`allensdk.core.brain_observatory_nwb_data_set.align_running_speed(dxcm, dxtime, timestamps)`

If running speed timestamps differ from fluorescence timestamps, adjust by inserting NaNs to running speed.

**Returns**

**tuple: dxcm, dxtime**

`allensdk.core.brain_observatory_nwb_data_set.get_epoch_mask_list(st, threshold, max_cuts=2)`

Convenience function to cut a stim table into multiple epochs

**Parameters**

- **st** – input stimtable
- **threshold** – threshold on the max duration of a subepoch
- **max\_cuts** – maximum number of allowed epochs to cut into

**Returns** epoch\_mask\_list, a list of indices that define the start and end of sub-epochs

**allensdk.core.cache\_method\_utilities module**

**class** `allensdk.core.cache_method_utilities.CachedInstanceMethodMixin`

Bases: `object`

**cache\_clear** (*self*)

Calls `cache_clear` method on all bound methods in this instance (where valid). Intended to clear calls cached with the `memoize` decorator. Note that this will also clear functions decorated with `lru_cache` and `lru_cache` in this class (or any other function with `cache_clear` attribute).

**allensdk.core.cell\_types\_cache module**

**class** `allensdk.core.cell_types_cache.CellTypesCache(cache=True, manifest_file=None, base_uri=None)`

Bases: `allensdk.api.cache.Cache`

Cache class for storing and accessing data from the Cell Types Database. By default, this class will cache any downloaded metadata or files in well known locations defined in a manifest file. This behavior can be disabled.

#### Parameters

**cache: boolean** Whether the class should save results of API queries to locations specified in the manifest file. Queries for files (as opposed to metadata) must have a file location. If caching is disabled, those locations must be specified in the function call (e.g. `get_ephys_data(file_name='file.nwb')`).

**manifest\_file: string** File name of the manifest to be read. Default is "cell\_types\_manifest.json".

#### Attributes

**api: CellTypesApi instance** The object used for making API queries related to the Cell Types Database

**CELLS\_KEY** = 'CELLS'

**EPHYS\_DATA\_KEY** = 'EPHYS\_DATA'

**EPHYS\_FEATURES\_KEY** = 'EPHYS\_FEATURES'

**EPHYS\_SWEEPS\_KEY** = 'EPHYS\_SWEEPS'

**MANIFEST\_VERSION** = '1.1'

**MARKER\_KEY** = 'MARKER'

**MORPHOLOGY\_FEATURES\_KEY** = 'MORPHOLOGY\_FEATURES'

**RECONSTRUCTION\_KEY** = 'RECONSTRUCTION'

**build\_manifest** (*self*, *file\_name*)

Construct a manifest for this Cache class and save it in a file.

#### Parameters

**file\_name: string** File location to save the manifest.

**get\_all\_features** (*self*, *dataframe=False*, *require\_reconstruction=True*)

Download morphology and electrophysiology features for all cells and merge them into a single table.

#### Parameters

**dataframe: boolean** Return the output as a Pandas DataFrame. If False, return a list of dictionaries.

**require\_reconstruction: boolean** Only return ephys and morphology features for cells that have reconstructions. Default True.

**get\_cells** (*self*, *file\_name=None*, *require\_morphology=False*, *require\_reconstruction=False*, *reporter\_status=None*, *species=None*, *simple=True*)

Download metadata for all cells in the database and optionally return a subset filtered by whether or not they have a morphology or reconstruction.

#### Parameters

**file\_name: string** File name to save/read the cell metadata as JSON. If *file\_name* is None, the *file\_name* will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is None.

**require\_morphology: boolean** Filter out cells that have no morphological images.

**require\_reconstruction: boolean** Filter out cells that have no morphological reconstructions.



**reporter\_status: list** Filter for cells that have one or more cell reporter statuses.

**species: list** Filter for cells that belong to one or more species. If None, return all. Must be one of [ CellTypesApi.MOUSE, CellTypesApi.HUMAN ].

**get\_ephys\_data** (*self, specimen\_id, file\_name=None*)

Download electrophysiology traces for a single cell in the database.

#### Parameters

**specimen\_id: int** The ID of a cell specimen to download.

**file\_name: string** File name to save/read the ephys features metadata as CSV. If file\_name is None, the file\_name will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is None.

#### Returns

**NwbDataSet** A class instance with helper methods for retrieving stimulus and response traces out of an NWB file.

**get\_ephys\_features** (*self, dataframe=False, file\_name=None*)

Download electrophysiology features for all cells in the database.

#### Parameters

**file\_name: string** File name to save/read the ephys features metadata as CSV. If file\_name is None, the file\_name will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is None.

**dataframe: boolean** Return the output as a Pandas DataFrame. If False, return a list of dictionaries.

**get\_ephys\_sweeps** (*self, specimen\_id, file\_name=None*)

Download sweep metadata for a single cell specimen.

#### Parameters

**specimen\_id: int** ID of a cell.

**get\_morphology\_features** (*self, dataframe=False, file\_name=None*)

Download morphology features for all cells with reconstructions in the database.

#### Parameters

**file\_name: string** File name to save/read the ephys features metadata as CSV. If file\_name is None, the file\_name will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is None.

**dataframe: boolean** Return the output as a Pandas DataFrame. If False, return a list of dictionaries.

**get\_reconstruction** (*self, specimen\_id, file\_name=None*)

Download and open a reconstruction for a single cell in the database.

#### Parameters

**specimen\_id: int** The ID of a cell specimen to download.

**file\_name: string** File name to save/read the reconstruction SWC. If file\_name is None, the file\_name will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is None.

#### Returns

**Morphology** A class instance with methods for accessing morphology compartments.

**get\_reconstruction\_markers** (*self*, *specimen\_id*, *file\_name=None*)

Download and open a reconstruction marker file for a single cell in the database.

#### Parameters

**specimen\_id: int** The ID of a cell specimen to download.

**file\_name: string** File name to save/read the reconstruction marker. If *file\_name* is *None*, the *file\_name* will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is *None*.

#### Returns

**Morphology** A class instance with methods for accessing morphology compartments.

**class** allensdk.core.cell\_types\_cache.ReporterStatus

Bases: object

Valid strings for filtering by cell reporter status.

**INDETERMINATE** = *None*

**NA** = *None*

**NEGATIVE** = 'negative'

**POSITIVE** = 'positive'

### allensdk.core.dat\_utilities module

**class** allensdk.core.dat\_utilities.DatUtilities

Bases: object

**classmethod** **save\_voltage** (*output\_path*, *v*, *t*)

Save a single voltage output result into a simple text format.

The output file is one t v pair per line.

#### Parameters

**output\_path** [string] file name for output

**v** [numpy array] voltage

**t** [numpy array] time

### allensdk.core.exceptions module

**exception** allensdk.core.exceptions.DataFrameIndexError (*msg*,  
*caught\_exception=None*)

Bases: LookupError

More verbose method for accessing invalid rows or columns in a dataframe. Should be used when an index error is thrown on a dataframe.

**exception** allensdk.core.exceptions.DataFrameKeyError (*msg*, *caught\_exception=None*)

Bases: LookupError

More verbose method for accessing invalid rows or columns in a dataframe. Should be used when a keyerror is thrown on a dataframe.

**exception** allensdk.core.exceptions.MissingDataError

Bases: ValueError

**allensdk.core.h5\_utilities module**

`allensdk.core.h5_utilities.decode_bytes` (*bytes\_dataset*, *encoding='UTF-8'*)

Convert the elements of a dataset of bytes to str

`allensdk.core.h5_utilities.h5_object_matcher_relname_in` (*relnames*,  
*h5\_object\_name*,  
*h5\_object*)

Asks if an h5 object's relative name (the final section of its absolute name) is contained within a provided array

**Parameters**

**relnames** [array-like] Relative names against which to match

**h5\_object\_name** [str] Full name (path from origin) of h5 object

**h5\_object** [h5py.Group, h5py.Dataset] Check this object's relative name

**Returns**

**bool** : whether the match succeeded

**h5\_object** [h5py.group, h5py.Dataset] the argued object

`allensdk.core.h5_utilities.keyed_locate_h5_objects` (*matcher\_cbs*, *h5\_file*,  
*start\_node=None*)

Traverse an h5 file and build up a dictionary mapping supplied keys to located objects

`allensdk.core.h5_utilities.load_datasets_by_relnames` (*relnames*, *h5\_file*, *start\_node*)

A convenience function for finding and loading into memory one or more datasets from an h5 file

`allensdk.core.h5_utilities.locate_h5_objects` (*matcher\_cb*, *h5\_file*, *start\_node=None*)

Traverse an h5 file and return objects matching supplied criteria

`allensdk.core.h5_utilities.traverse_h5_file` (*callback*, *h5\_file*, *start\_node=None*)

Traverse an h5 file and apply a callback to each node

**allensdk.core.json\_utilities module**

**class** `allensdk.core.json_utilities.JsonComments`

Bases: object

**classmethod** `read_file` (*file\_name*)

**classmethod** `read_string` (*json\_string*)

**classmethod** `remove_comments` (*json\_string*)

Strip single and multiline javascript-style comments.

**Parameters**

**json** [string] Json string with javascript-style comments.

**Returns**

**string** Copy of the input with comments removed.

**Note:** A JSON decoder MAY accept and ignore comments.

**classmethod** `remove_multiline_comments` (*json\_string*)

Rebuild input without substrings matching */.../*.

**Parameters**

**json\_string** [string] may or may not contain multiline comments.

#### Returns

**string** Copy of the input without the comments.

`allensdk.core.json_utilities.json_handler(obj)`

Used by `write_json` convert a few non-standard types to things that the `json` package can handle.

`allensdk.core.json_utilities.read(file_name)`

Shortcut reading JSON from a file.

`allensdk.core.json_utilities.read_url(url, method='POST')`

`allensdk.core.json_utilities.read_url_get(url)`

Transform a JSON contained in a file into an equivalent nested python dict.

#### Parameters

**url** [string] where to get the json.

#### Returns

**dict** Python version of the input

**Note: if the input is a bare array or literal, for example,**

**the output will be of the corresponding type.**

`allensdk.core.json_utilities.read_url_post(url)`

Transform a JSON contained in a file into an equivalent nested python dict.

#### Parameters

**url** [string] where to get the json.

#### Returns

**dict** Python version of the input

**Note: if the input is a bare array or literal, for example,**

**the output will be of the corresponding type.**

`allensdk.core.json_utilities.write(file_name, obj)`

Shortcut for writing JSON to a file. This also takes care of serializing numpy and data types.

`allensdk.core.json_utilities.write_string(obj)`

Shortcut for writing JSON to a string. This also takes care of serializing numpy and data types.

### **allensdk.core.mouse\_connectivity\_cache module**

```
class allensdk.core.mouse_connectivity_cache.MouseConnectivityCache(resolution=None,
                                                                    cache=True,
                                                                    mani-
                                                                    fest_file=None,
                                                                    ccf_version=None,
                                                                    base_uri=None,
                                                                    ver-
                                                                    sion=None)
```

Bases: `allensdk.core.reference_space_cache.ReferenceSpaceCache`

Cache class for storing and accessing data related to the adult mouse Connectivity Atlas. By default, this class will cache any downloaded metadata or files in well known locations defined in a manifest file. This behavior can be disabled.

### Parameters

**resolution: int** Resolution of grid data to be downloaded when accessing projection volume, the annotation volume, and the annotation volume. Must be one of (10, 25, 50, 100). Default is 25.

**ccf\_version: string** Desired version of the Common Coordinate Framework. This affects the annotation volume (`get_annotation_volume`) and structure masks (`get_structure_mask`). Must be one of (`MouseConnectivityApi.CCF_2015`, `MouseConnectivityApi.CCF_2016`). Default: `MouseConnectivityApi.CCF_2016`

**cache: boolean** Whether the class should save results of API queries to locations specified in the manifest file. Queries for files (as opposed to metadata) must have a file location. If caching is disabled, those locations must be specified in the function call (e.g. `get_projection_density(file_name='file.nrrd')`).

**manifest\_file: string** File name of the manifest to be read. Default is "mouse\_connectivity\_manifest.json".

### Attributes

**resolution: int** Resolution of grid data to be downloaded when accessing projection volume, the annotation volume, and the annotation volume. Must be one of (10, 25, 50, 100). Default is 25.

**api: MouseConnectivityApi instance** Used internally to make API queries.

```
ALIGNMENT3D_KEY = 'ALIGNMENT3D'
DATA_MASK_KEY = 'DATA_MASK'
DEFAULT_STRUCTURE_SET_IDS = (167587189,)
DEFORMATION_FIELD_HEADER_KEY = 'DEFORMATION_FIELD_HEADER'
DEFORMATION_FIELD_VOXEL_KEY = 'DEFORMATION_FIELD_VOXELS'
DFMFLD_RESOLUTIONS = (25,)
EXPERIMENTS_KEY = 'EXPERIMENTS'
INJECTION_DENSITY_KEY = 'INJECTION_DENSITY'
INJECTION_FRACTION_KEY = 'INJECTION_FRACTION'
MANIFEST_VERSION = 1.3
PROJECTION_DENSITY_KEY = 'PROJECTION_DENSITY'
STRUCTURE_UNIONIZES_KEY = 'STRUCTURE_UNIONIZES'
SUMMARY_STRUCTURE_SET_ID = 167587189
add_manifest_paths(self, manifest_builder)
    Construct a manifest for this Cache class and save it in a file.
```

### Parameters

**file\_name: string** File location to save the manifest.

**default\_structure\_ids**

**filter\_experiments** (*self, experiments, cre=None, injection\_structure\_ids=None*)

Take a list of experiments and filter them by cre status and injection structure.

#### Parameters

**cre: boolean or list** If True, return only cre-positive experiments. If False, return only cre-negative experiments. If None, return all experiments. If list, return all experiments with cre line names in the supplied list. Default None.

**injection\_structure\_ids: list** Only return experiments that were injected in the structures provided here. If None, return all experiments. Default None.

**filter\_structure\_unionizes** (*self, unionizes, is\_injection=None, structure\_ids=None, include\_descendants=False, hemisphere\_ids=None*)

Take a list of unionizes and return a subset of records filtered by injection status, structure, and hemisphere.

#### Parameters

**is\_injection: boolean** If True, only return unionize records that disregard non-injection pixels. If False, only return unionize records that disregard injection pixels. If None, return all records. Default None.

**structure\_ids: list** Only return unionize records for a set of structures. If None, return all records. Default None.

**include\_descendants: boolean** Include all descendant records for specified structures. Default False.

**hemisphere\_ids: list** Only return unionize records that disregard pixels outside of a hemisphere. or set of hemispheres. Left = 1, Right = 2, Both = 3. If None, include all records [1, 2, 3]. Default None.

**get\_affine\_parameters** (*self, section\_data\_set\_id, direction='trv', file\_name=None*)

Extract the parameters of the 3D affine tranformation mapping this section data set's image-space stack to CCF-space (or vice-versa).

#### Parameters

**section\_data\_set\_id** [int] download the parameters for this data set.

**direction** [str, optional]

#### Valid options are:

**trv** ["transform from reference to volume". Maps CCF points to image space points. If you are ] resampling data into CCF, this is the direction you want.

**tvr** : "transform from volume to reference". Maps image space points to CCF points.

**file\_name** [str] If provided, store the downloaded file here.

#### Returns

**alignment** [numpy.ndarray]

**4 X 3 matrix. In order to transform a point [X\_1, X\_2, X\_3] run** `np.dot([X_1, X_2, X_3, 1], alignment)`. In

**to build a SimpleITK affine transform run:** `transform = sitk.AffineTransform(3)`  
`transform.SetParameters(alignment.flatten())`

**get\_data\_mask** (*self, experiment\_id, file\_name=None*)

Read a data mask volume for a single experiment. Download it first if it doesn't exist. Data mask is a binary mask of voxels that have valid data. Only use valid data in analysis!

**Parameters**

**experiment\_id: int** ID of the experiment to download/read. This corresponds to `section_data_set_id` in the API.

**file\_name: string** File name to store the template volume. If it already exists, it will be read from this file. If `file_name` is `None`, the `file_name` will be pulled out of the manifest. Default is `None`.

**get\_deformation\_field** (*self, section\_data\_set\_id, header\_path=None, voxel\_path=None*)

Extract the local alignment parameters for this dataset. This a 3D vector image (3 components) describing a deformable local mapping from CCF voxels to this section data set's affine-aligned image stack.

**Parameters**

**section\_data\_set\_id** [int]

Download the deformation field for this data set

**header\_path** [str, optional] If supplied, the deformation field header will be downloaded to this path.

**voxel\_path** [str, optional] If supplied, the deformation field voxels will be downloaded to this path.

**Returns**

**numpy.ndarray** : 3D X 3 component vector array (origin 0, 0, 0; 25-micron isotropic resolution) defining a deformable transformation from CCF-space to affine-transformed image space.

**get\_experiment\_structure\_unionizes** (*self, experiment\_id, file\_name=None, is\_injection=None, structure\_ids=None, include\_descendants=False, hemisphere\_ids=None*)

Retrieve the structure unionize data for a specific experiment. Filter by structure, injection status, and hemisphere.

**Parameters**

**experiment\_id: int** ID of the experiment of interest. Corresponds to `section_data_set_id` in the API.

**file\_name: string** File name to save/read the experiments list. If `file_name` is `None`, the `file_name` will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is `None`.

**is\_injection: boolean** If `True`, only return unionize records that disregard non-injection pixels. If `False`, only return unionize records that disregard injection pixels. If `None`, return all records. Default `None`.

**structure\_ids: list** Only return unionize records for a specific set of structures. If `None`, return all records. Default `None`.

**include\_descendants: boolean** Include all descendant records for specified structures. Default `False`.

**hemisphere\_ids: list** Only return unionize records that disregard pixels outside of a hemisphere. or set of hemispheres. Left = 1, Right = 2, Both = 3. If `None`, include all records [1, 2, 3]. Default `None`.

**get\_experiments** (*self*, *dataframe=False*, *file\_name=None*, *cre=None*, *injection\_structure\_ids=None*)

Read a list of experiments that match certain criteria. If caching is enabled, this will save the whole (unfiltered) list of experiments to a file.

#### Parameters

**dataframe: boolean** Return the list of experiments as a Pandas DataFrame. If False, return a list of dictionaries. Default False.

**file\_name: string** File name to save/read the structures table. If file\_name is None, the file\_name will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is None.

**cre: boolean or list** If True, return only cre-positive experiments. If False, return only cre-negative experiments. If None, return all experiments. If list, return all experiments with cre line names in the supplied list. Default None.

**injection\_structure\_ids: list** Only return experiments that were injected in the structures provided here. If None, return all experiments. Default None.

**get\_injection\_density** (*self*, *experiment\_id*, *file\_name=None*)

Read an injection density volume for a single experiment. Download it first if it doesn't exist. Injection density is the proportion of projecting pixels in a grid voxel only including pixels that are part of the injection site in [0,1].

#### Parameters

**experiment\_id: int** ID of the experiment to download/read. This corresponds to section\_data\_set\_id in the API.

**file\_name: string** File name to store the template volume. If it already exists, it will be read from this file. If file\_name is None, the file\_name will be pulled out of the manifest. Default is None.

**get\_injection\_fraction** (*self*, *experiment\_id*, *file\_name=None*)

Read an injection fraction volume for a single experiment. Download it first if it doesn't exist. Injection fraction is the proportion of pixels in the injection site in a grid voxel in [0,1].

#### Parameters

**experiment\_id: int** ID of the experiment to download/read. This corresponds to section\_data\_set\_id in the API.

**file\_name: string** File name to store the template volume. If it already exists, it will be read from this file. If file\_name is None, the file\_name will be pulled out of the manifest. Default is None.

**get\_projection\_density** (*self*, *experiment\_id*, *file\_name=None*)

Read a projection density volume for a single experiment. Download it first if it doesn't exist. Projection density is the proportion of projecting pixels in a grid voxel in [0,1].

#### Parameters

**experiment\_id: int** ID of the experiment to download/read. This corresponds to section\_data\_set\_id in the API.

**file\_name: string** File name to store the template volume. If it already exists, it will be read from this file. If file\_name is None, the file\_name will be pulled out of the manifest. Default is None.

**get\_projection\_matrix** (*self*, *experiment\_ids*, *projection\_structure\_ids=None*, *hemisphere\_ids=None*, *parameter='projection\_volume'*, *dataframe=False*)



**get\_structure\_unionizes** (*self, experiment\_ids, is\_injection=None, structure\_ids=None, include\_descendants=False, hemisphere\_ids=None*)

Get structure unionizes for a set of experiment IDs. Filter the results by injection status, structure, and hemisphere.

#### Parameters

**experiment\_ids: list** List of experiment IDs. Corresponds to section\_data\_set\_id in the API.

**is\_injection: boolean** If True, only return unionize records that disregard non-injection pixels. If False, only return unionize records that disregard injection pixels. If None, return all records. Default None.

**structure\_ids: list** Only return unionize records for a specific set of structures. If None, return all records. Default None.

**include\_descendants: boolean** Include all descendant records for specified structures. Default False.

**hemisphere\_ids: list** Only return unionize records that disregard pixels outside of a hemisphere. or set of hemispheres. Left = 1, Right = 2, Both = 3. If None, include all records [1, 2, 3]. Default None.

**rank\_structures** (*self, experiment\_ids, is\_injection, structure\_ids=None, hemisphere\_ids=None, rank\_on='normalized\_projection\_volume', n=5, threshold=0.01*)

Produces one or more (per experiment) ranked lists of brain structures, using a specified data field.

#### Parameters

**experiment\_ids** [list of int] Obtain injection\_structures for these experiments.

**is\_injection** [boolean] Use data from only injection (or non-injection) unionizes.

**structure\_ids** [list of int, optional] Consider only these structures. It is a good idea to make sure that these structures are not spatially overlapping; otherwise your results will contain redundant information. Defaults to the summary structures - a brain-wide list of nonoverlapping mid-level structures.

**hemisphere\_ids** [list of int, optional] Consider only these hemispheres (1: left, 2: right, 3: both). Like with structures, you might get redundant results if you select overlapping options. Defaults to [1, 2].

**rank\_on** [str, optional] Rank unionize data using this field (descending). Defaults to normalized\_projection\_volume.

**n** [int, optional] Return only the top n structures.

**threshold** [float, optional] Consider only records whose data value - specified by the rank\_on parameter - exceeds this value.

#### Returns

**list :** Each element (1 for each input experiment) is a list of dictionaries. The dictionaries describe the top injection structures in descending order. They are specified by their structure and hemisphere id fields and additionally report the value specified by the rank\_on parameter.

### allensdk.core.nwb\_data\_set module

**class** allensdk.core.nwb\_data\_set.NwbDataSet (*file\_name, spike\_time\_key=None*)

Bases: object

A very simple interface for extracting electrophysiology data from an NWB file.

**DEPRECATED\_SPIKE\_TIMES** = 'aibs\_spike\_times'

**SPIKE\_TIMES** = 'spike\_times'

**fill\_sweep\_responses** (*self*, *fill\_value*=0.0, *sweep\_numbers*=None, *extend\_experiment*=False)

Fill sweep response arrays with a single value.

#### Parameters

**fill\_value: float** Value used to fill sweep response array

**sweep\_numbers: list** List of integer sweep numbers to be filled (default all sweeps)

**extend\_experiment: bool** If True, extend experiment epoch length to the end of the sweep (undo any truncation)

**get\_experiment\_sweep\_numbers** (*self*)

Get all of the sweep numbers for experiment epochs in the file, not including test sweeps.

**get\_pipeline\_version** (*self*)

Returns the AI pipeline version number, stored in the metadata field 'generated\_by'. If that field is missing, version 0.0 is returned.

#### Returns

**int tuple: (major, minor)**

**get\_spike\_times** (*self*, *sweep\_number*, *key*=None)

Return any spike times stored in the NWB file for a sweep.

#### Parameters

**sweep\_number: int** index to access

**key** [string] label where the spike times are stored (default NwbDataSet.SPIKE\_TIMES)

#### Returns

**list** list of spike times in seconds relative to the start of the sweep

**get\_sweep** (*self*, *sweep\_number*)

Retrieve the stimulus, response, index\_range, and sampling rate for a particular sweep. This method hides the NWB file's distinction between a "Sweep" and an "Experiment". An experiment is a subset of a sweep that excludes the initial test pulse. It also excludes any erroneous response data at the end of the sweep (usually for ramp sweeps, where recording was terminated mid-stimulus).

Some sweeps do not have an experiment, so full data arrays are returned. Sweeps that have an experiment return full data arrays (include the test pulse) with any erroneous data trimmed from the back of the sweep.

#### Parameters

**sweep\_number: int**

#### Returns

**dict** A dictionary with 'stimulus', 'response', 'index\_range', and 'sampling\_rate' elements. The index range is a 2-tuple where the first element indicates the end of the test pulse and the second index is the end of valid response data.

**get\_sweep\_metadata** (*self*, *sweep\_number*)

Retrieve the sweep level metadata associated with each sweep. Includes information on stimulus parameters like its name and amplitude as well as recording quality metadata, like access resistance and seal quality.

**Parameters****sweep\_number: int****Returns**

**dict** A dictionary with 'aibs\_stimulus\_amplitude\_pa', 'aibs\_stimulus\_name', 'gain', 'initial\_access\_resistance', 'seal' elements. These specific fields are ones encoded in the original AIBS in vitro .nwb files.

**get\_sweep\_numbers** (*self*)

Get all of the sweep numbers in the file, including test sweeps.

**set\_spike\_times** (*self, sweep\_number, spike\_times, key=None*)

Set or overwrite the spikes times for a sweep.

**Parameters****sweep\_number** [int] index to access**key** [string] where the times are stored (default NwbDataSet.SPIKE\_TIME)**spike\_times: np.array** array of spike times in seconds**set\_sweep** (*self, sweep\_number, stimulus, response*)

Overwrite the stimulus or response of an NWB file. If the supplied arrays are shorter than stored arrays, they are padded with zeros to match the original data size.

**Parameters****sweep\_number: int****stimulus: np.array** Overwrite the stimulus with this array. If None, stimulus is unchanged.**response: np.array** Overwrite the response with this array. If None, response is unchanged.**allensdk.core.obj\_utilities module**`allensdk.core.obj_utilities.parse_obj` (*lines*)

Parse a wavefront obj file into a triplet of vertices, normals, and faces. This parser is specific to obj files generated from our annotation volumes

**Parameters****lines** [list of str] Lines of input obj file**Returns****vertices** [np.ndarray] Dimensions are (nSamples, nCoordinates=3). Locations in the reference space of vertices**vertex\_normals** [np.ndarray] Dimensions are (nSample, nElements=3). Vectors normal to vertices.**face\_vertices** [np.ndarray] Dimensions are (sample, nVertices=3). References are given in indices (0-indexed here, but 1-indexed in the file) of vertices that make up each face.**face\_normals** [np.ndarray] Dimensions are (sample, nNormals=3). References are given in indices (0-indexed here, but 1-indexed in the file) of vertex normals that make up each face.

## Notes

This parser is specialized to the obj files that the Allen Institute for Brain Science generates from our own structure annotations.

```
allensdk.core.obj_utilities.read_obj(path)
```

## allensdk.core.ontology module

```
class allensdk.core.ontology.Ontology(df)
    Bases: object
```

---

**Note:** Deprecated from 0.12.5 *Ontology* has been replaced by *StructureTree*.

---

```
get_child_ids(self, structure_ids)
```

Find the set of ids that are immediate children of one or more structures.

### Parameters

**structure\_ids: iterable** Any iterable type that contains structure ids that can be cast to integers.

### Returns

**set** Set of child structure ids

```
get_children(self, structure_ids)
```

Find the set of structures that are immediate children of one or more structures.

### Parameters

**structure\_ids: iterable** Any iterable type that contains structure ids that can be cast to integers.

### Returns

**pandas.DataFrame** Set of child structures

```
get_descendant_ids(self, structure_ids)
```

Find the set of the ids of structures that are descendants of one or more structures. The returned set will include the input structure ids.

### Parameters

**structure\_ids: iterable** Any iterable type that contains structure ids that can be cast to integers.

### Returns

**set** Set of descendant structure ids.

```
get_descendants(self, structure_ids)
```

Find the set of structures that are descendants of one or more structures. The returned set will include the input structures.

### Parameters

**structure\_ids: iterable** Any iterable type that contains structure ids that can be cast to integers.

### Returns

**pandas.DataFrame** Set of descendant structures.

**structure\_descends\_from** (*self*, *child\_id*, *parent\_id*)

Return whether one structure id is a descendant of another structure id.

## allensdk.core.ophys\_experiment\_session\_id\_mapping module

## allensdk.core.reference\_space module

**class** allensdk.core.reference\_space.**ReferenceSpace** (*structure\_tree*, *annotation*, *resolution*)

Bases: object

**static check\_and\_write** (*base\_dir*, *structure\_id*, *fn*)

A many\_structure\_masks callback that writes the mask to a nrrd file if the file does not already exist.

**check\_coverage** (*self*, *structure\_ids*, *domain\_mask*)

Determines whether a spatial domain is completely covered by structures in a set.

### Parameters

**structure\_ids** [list of int] Specifies the set of structures to check.

**domain\_mask** [numpy ndarray] Same shape as annotation. 1 inside the mask, 0 out. Specifies spatial domain.

### Returns

**numpy ndarray** : 1 where voxels are missing from the candidate, 0 where the candidate exceeds the domain

**direct\_voxel\_counts** (*self*)

Determines the number of voxels directly assigned to one or more structures.

### Returns

**dict** : Keys are structure ids, values are the number of voxels directly assigned to those structures.

**direct\_voxel\_map**

**downsample** (*self*, *target\_resolution*)

Obtain a smaller reference space by downsampling

### Parameters

**target\_resolution** [tuple of numeric] Resolution in microns of the output space.

**interpolator** [string] Method used to interpolate the volume. Currently only 'nearest' is supported

### Returns

**ReferenceSpace** : A new ReferenceSpace with the same structure tree and a downsampled annotation.

**export\_itksnap\_labels** (*self*, *id\_type*=<class 'numpy.uint16'>, *label\_description\_kwargs*=None)

Produces itknap labels, remapping large ids if needed.

### Parameters

**id\_type** [np.integer, optional] Used to determine the type of the output annotation and whether ids need to be remapped to smaller values.

**label\_description\_kwargs** [dict, optional] Keyword arguments passed to StructureTree.export\_label\_description

#### Returns

**np.ndarray** : Annotation volume, remapped if needed

**pd.DataFrame** label\_description dataframe

**get\_slice\_image** (*self*, *axis*, *position*, *cmap=None*)  
Produce a AxBx3 RGB image from a slice in the annotation

#### Parameters

**axis** [int] Along which to slice the annotation volume. 0 is coronal, 1 is horizontal, and 2 is sagittal.

**position** [int] In microns. Take the slice from this far along the specified axis.

**cmap** [dict, optional] Keys are structure ids, values are rgb triplets. Defaults to structure rgb\_triplets.

#### Returns

**np.ndarray** : RGB image array.

### Notes

If you assign a custom colormap, make sure that you take care of the background in addition to the structures.

**make\_structure\_mask** (*self*, *structure\_ids*, *direct\_only=False*)  
Return an indicator array for one or more structures

#### Parameters

**structure\_ids** [list of int] Make a mask that indicates the union of these structures' voxels

**direct\_only** [bool, optional] If True, only include voxels directly assigned to a structure in the mask. Otherwise include voxels assigned to descendants.

#### Returns

**numpy ndarray** : Same shape as annotation. 1 inside mask, 0 outside.

**many\_structure\_masks** (*self*, *structure\_ids*, *output\_cb=None*, *direct\_only=False*)  
Build one or more structure masks and do something with them

#### Parameters

**structure\_ids** [list of int] Specify structures to be masked

**output\_cb** [function, optional] Must have the following signature: `output_cb(structure_id, fn)`. On each requested id, `fn` will be carried to make a mask for that id. Defaults to returning the structure id and mask.

**direct\_only** [bool, optional] If True, only include voxels directly assigned to a structure in the mask. Otherwise include voxels assigned to descendants.

#### Yields

Return values of **output\_cb** called on each **structure\_id**, **structure\_mask** pair.

## Notes

output\_cb is called on every yield, so any side-effects (such as writing to a file) will be carried out regardless of what you do with the return values. You do actually have to iterate through the output, though.

**remove\_unassigned** (*self*, *update\_self=True*)

Obtains a structure tree consisting only of structures that have at least one voxel in the annotation.

### Parameters

**update\_self** [bool, optional] If True, the contained structure tree will be replaced,

### Returns

**list of dict** : elements are filtered structures

**static return\_mask\_cb** (*structure\_id*, *fn*)

A basic callback for many\_structure\_masks

**total\_voxel\_counts** (*self*)

Determines the number of voxels assigned to a structure or its descendants

### Returns

**dict** : Keys are structure ids, values are the number of voxels assigned to structures' descendants.

**total\_voxel\_map**

**validate\_structures** (*self*, *structure\_ids*, *domain\_mask*)

Determines whether a set of structures produces an exact and nonoverlapping tiling of a spatial domain

### Parameters

**structure\_ids** [list of int] Specifies the set of structures to check.

**domain\_mask** [numpy ndarray] Same shape as annotation. 1 inside the mask, 0 out. Specifies spatial domain.

### Returns

**set** : Ids of structures that are the ancestors of other structures in the supplied set.

**numpy ndarray** : Indicator for missing voxels.

**write\_itksnap\_labels** (*self*, *annotation\_path*, *label\_path*, *\*\*kwargs*)

Generate a label file (nrrd) and a label\_description file (csv) for use with ITKSnap

### Parameters

**annotation\_path** [str] write generated label file here

**label\_path** [str] write generated label\_description file here

**\*\*kwargs** : will be passed to self.export\_itksnap\_labels

## allensdk.core.reference\_space\_cache module

```
class allensdk.core.reference_space_cache.ReferenceSpaceCache (resolution, reference_space_key,
                                                                **kwargs)
```

Bases: `allensdk.api.cache.Cache`

**ANNOTATION\_KEY** = 'ANNOTATION'

```
MANIFEST_VERSION = 1.2
REFERENCE_SPACE_VERSION_KEY = 'REFERENCE_SPACE_VERSION'
STRUCTURES_KEY = 'STRUCTURES'
STRUCTURE_MASK_KEY = 'STRUCTURE_MASK'
STRUCTURE_MESH_KEY = 'STRUCTURE_MESH'
STRUCTURE_TREE_KEY = 'STRUCTURE_TREE'
TEMPLATE_KEY = 'TEMPLATE'
```

**add\_manifest\_paths** (*self*, *manifest\_builder*)

Construct a manifest for this Cache class and save it in a file.

#### Parameters

**file\_name: string** File location to save the manifest.

**get\_annotation\_volume** (*self*, *file\_name=None*)

Read the annotation volume. Download it first if it doesn't exist.

#### Parameters

**file\_name: string** File name to store the annotation volume. If it already exists, it will be read from this file. If *file\_name* is None, the *file\_name* will be pulled out of the manifest. Default is None.

**get\_reference\_space** (*self*, *structure\_file\_name=None*, *annotation\_file\_name=None*)

Build a ReferenceSpace from this cache's annotation volume and structure tree. The ReferenceSpace does operations that relate brain structures to spatial domains.

#### Parameters

**structure\_file\_name: string** File name to save/read the structures table. If *file\_name* is None, the *file\_name* will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is None.

**annotation\_file\_name: string** File name to store the annotation volume. If it already exists, it will be read from this file. If *file\_name* is None, the *file\_name* will be pulled out of the manifest. Default is None.

**get\_structure\_mask** (*self*, *structure\_id*, *file\_name=None*, *annotation\_file\_name=None*)

Read a 3D numpy array shaped like the annotation volume that has non-zero values where voxels belong to a particular structure. This will take care of identifying substructures.

#### Parameters

**structure\_id: int** ID of a structure.

**file\_name: string** File name to store the structure mask. If it already exists, it will be read from this file. If *file\_name* is None, the *file\_name* will be pulled out of the manifest. Default is None.

**annotation\_file\_name: string** File name to store the annotation volume. If it already exists, it will be read from this file. If *file\_name* is None, the *file\_name* will be pulled out of the manifest. Default is None.

## Notes

This method downloads structure masks from the Allen Institute. To make your own locally, see `ReferenceSpace.many_structure_masks`.



**get\_structure\_mesh** (*self, structure\_id, file\_name=None*)

Obtain a 3D mesh specifying the surface of an annotated structure.

#### Parameters

**structure\_id: int** ID of a structure.

**file\_name: string** File name to store the structure mesh. If it already exists, it will be read from this file. If `file_name` is `None`, the `file_name` will be pulled out of the manifest. Default is `None`.

#### Returns

**vertices** [np.ndarray] Dimensions are (nSamples, nCoordinates=3). Locations in the reference space of vertices

**vertex\_normals** [np.ndarray] Dimensions are (nSample, nElements=3). Vectors normal to vertices.

**face\_vertices** [np.ndarray] Dimensions are (sample, nVertices=3). References are given in indices (0-indexed here, but 1-indexed in the file) of vertices that make up each face.

**face\_normals** [np.ndarray] Dimensions are (sample, nNormals=3). References are given in indices (0-indexed here, but 1-indexed in the file) of vertex normals that make up each face.

#### Notes

These meshes are meant for 3D visualization and as such have been smoothed. If you are interested in performing quantitative analyses, we recommend that you use the structure masks instead.

**get\_structure\_tree** (*self, file\_name=None, structure\_graph\_id=1*)

Read the list of adult mouse structures and return an `StructureTree` instance.

#### Parameters

**file\_name: string** File name to save/read the structures table. If `file_name` is `None`, the `file_name` will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is `None`.

**structure\_graph\_id: int** Build a tree using structure only from the identified structure graph.

**get\_template\_volume** (*self, file\_name=None*)

Read the template volume. Download it first if it doesn't exist.

#### Parameters

**file\_name: string** File name to store the template volume. If it already exists, it will be read from this file. If `file_name` is `None`, the `file_name` will be pulled out of the manifest. Default is `None`.

**classmethod validate\_structure\_id** (*structure\_id*)

**classmethod validate\_structure\_ids** (*structure\_ids*)

### allensdk.core.simple\_tree module

**class** allensdk.core.simple\_tree.**SimpleTree** (*nodes, node\_id\_cb, parent\_id\_cb*)

Bases: `object`

**ancestor\_ids** (*self*, *node\_ids*)

Obtain the ids of one or more nodes' ancestors

**Parameters**

**node\_ids** [list of hashable] Items are ids of nodes whose ancestors you wish to find.

**Returns**

**list of list of hashable :** Items are lists of input nodes' ancestors' ids.

**Notes**

Given the tree: A -> B -> C

‘-> D

The ancestors of C are [C, B, A]. The ancestors of A are [A]. The ancestors of D are [D, A]

**ancestors** (*self*, *node\_ids*)

Get one or mode nodes' ancestor nodes

**Parameters**

**node\_ids** [list of hashable] Items are ids of nodes whose ancestors will be found.

**Returns**

**list of list of dict :** Items are lists of ancestor nodes corresponding to argued ids.

**child\_ids** (*self*, *node\_ids*)

Obtain the ids of one or more nodes' children

**Parameters**

**node\_ids** [list of hashable] Items are ids of nodes whose children you wish to find.

**Returns**

**list of list of hashable :** Items are lists of input nodes' children's ids.

**children** (*self*, *node\_ids*)

Get one or mode nodes' child nodes

**Parameters**

**node\_ids** [list of hashable] Items are ids of nodes whose children will be found.

**Returns**

**list of list of dict :** Items are lists of child nodes corresponding to argued ids.

**descendant\_ids** (*self*, *node\_ids*)

Obtain the ids of one or more nodes' descendants

**Parameters**

**node\_ids** [list of hashable] Items are ids of nodes whose descendants you wish to find.

**Returns**

**list of list of hashable :** Items are lists of input nodes' descendants' ids.

## Notes

Given the tree: A -> B -> C

‘-> D

The descendants of A are [B, C, D]. The descendants of C are [].

**descendants** (*self*, *node\_ids*)

Get one or more nodes' descendant nodes

### Parameters

**node\_ids** [list of hashable] Items are ids of nodes whose descendants will be found.

### Returns

**list of list of dict** : Items are lists of descendant nodes corresponding to argued ids.

**filter\_nodes** (*self*, *criterion*)

Obtain a list of nodes filtered by some criterion

### Parameters

**criterion** [function | node dict => bool] Only nodes for which criterion returns true will be returned.

### Returns

**list of dict** : Items are node dictionaries that passed the filter.

**node** (*self*, *node\_ids=None*)

**node\_ids** (*self*)

Obtain the node ids of each node in the tree

### Returns

**list** : elements are node ids

**nodes** (*self*, *node\_ids=None*)

Get one or more nodes' full dictionaries from their ids.

### Parameters

**node\_ids** [list of hashable] Items are ids of nodes to be returned. Default is all.

### Returns

**list of dict** : Items are nodes corresponding to argued ids.

**nodes\_by\_property** (*self*, *key*, *values*, *to\_fn=None*)

Get nodes by a specified property

### Parameters

**key** [hashable or function] The property used for lookup. Should be unique. If a function, will be invoked on each node.

**values** [list] Select matching elements from the lookup.

**to\_fn** [function, optional] Defines the outputs, on a per-node basis. Defaults to returning the whole node.

### Returns

**list** : outputs, 1 for each input value.

**parent** (*self*, *node\_ids*)

**parent\_id** (*self*, *node\_ids*)

**parent\_ids** (*self*, *node\_ids*)

Obtain the ids of one or more nodes' parents

**Parameters**

**node\_ids** [list of hashable] Items are ids of nodes whose parents you wish to find.

**Returns**

**list of hashable :** Items are ids of input nodes' parents in order.

**parents** (*self*, *node\_ids*)

Get one or more nodes' parent nodes

**Parameters**

**node\_ids** [list of hashable] Items are ids of nodes whose parents will be found.

**Returns**

**list of dict :** Items are parents of nodes corresponding to argued ids.

**value\_map** (*self*, *from\_fn*, *to\_fn*)

Obtain a look-up table relating a pair of node properties across nodes

**Parameters**

**from\_fn** [function | node dict => hashable value] The keys of the output dictionary will be obtained by calling from\_fn on each node. Should be unique.

**to\_fn** [function | node\_dict => value] The values of the output function will be obtained by calling to\_fn on each node.

**Returns**

**dict :** Maps the node property defined by from\_fn to the node property defined by to\_fn across nodes.

## allensdk.core.sitk\_utilities module

`allensdk.core.sitk_utilities.fix_array_dimensions` (*array*, *ncomponents=1*)

Convenience function that reorders ndarray dimensions for io with SimpleITK

**Parameters**

**array** [np.ndarray] The array to be reordered

**ncomponents** [int, optional] Number of components per pixel, default 1.

**Returns**

**np.ndarray :** Reordered array

`allensdk.core.sitk_utilities.get_sitk_image_information` (*image*)

Extract information about a SimpleITK image

**Parameters**

**image** [sitk.Image] Extract information about this image.

**Returns**

**dict** : Extracted information. Includes spacing, origin, size, direction, and number of components per pixel

```
allensdk.core.sitk_utilities.read_ndarray_with_sitk(path)
```

Read a numpy array from a file using SimpleITK

#### Parameters

**path** [str] Read from this path

#### Returns

**image** [np.ndarray] Obtained array

**information** [dict] Additional information about the array

```
allensdk.core.sitk_utilities.set_sitk_image_information(image, information)
```

Set information on a SimpleITK image

#### Parameters

**image** [sitk.Image] Set information on this image.

**information** [dict] Stores information to be set. Supports spacing, origin, direction. Also checks (but cannot set) size and number of components per pixel

```
allensdk.core.sitk_utilities.write_ndarray_with_sitk(array, path, **information)
```

Write a numpy array to a file using SimpleITK

#### Parameters

**array** [np.ndarray] Array to be written.

**path** [str] Write to here

**\*\*information** [dict] Contains additional information to be stored in the image file. See `set_sitk_image_information` for more information.

## allensdk.core.structure\_tree module

```
class allensdk.core.structure_tree.StructureTree(nodes)
```

Bases: `allensdk.core.simple_tree.SimpleTree`

```
static clean_structures(structures, whitelist=None, data_transforms=None,
                        renames=None)
```

Convert structures\_with\_sets query results into a form that can be used to construct a StructureTree

#### Parameters

**structures** [list of dict] Each element describes a structure. Should have a structure id path field (str values) and a structure\_sets field (list of dict).

**whitelist** [list of str, optional] Only these fields will be included in the final structure record. Default is the output of `StructureTree.whitelist`.

**data\_transforms** [dict, optional] Keys are str field names. Values are functions which will be applied to the data associated with those fields. Default is to map colors from hex to rgb and convert the structure id path to a list of int.

**renames** [dict, optional] Controls the field names that appear in the output structure records. Default is to map 'color\_hex\_triplet' to 'rgb\_triplet'.

#### Returns

**list of dict** : structures, after conversion of structure\_id\_path and structure\_sets

**static collect\_sets** (*structure*)

Structure sets may be specified by full records or id. This method collects all of the structure set records/ids in a structure record and replaces them with a single list of id records.

**static data\_transforms** ()

**export\_label\_description** (*self*, *alphas=None*, *exclude\_label\_vis=None*, *exclude\_mesh\_vis=None*, *label\_key='acronym'*)

Produces an itksnap label\_description table from this structure tree

#### Parameters

**alphas** [dict, optional] Maps structure ids to alpha levels. Optional - will only use provided ids.

**exclude\_label\_vis** [list, optional] The structures denoted by these ids will not be visible in ITKSnap.

**exclude\_mesh\_vis** [list, optional] The structures denoted by these ids will not have visible meshes in ITKSnap.

**label\_key: str, optional** Use this column for display labels.

#### Returns

**pd.DataFrame** : Contains data needed for loading as an ITKSnap label description file.

**get\_ancestor\_id\_map** (*self*)

Get a dictionary mapping structure ids to ancestor ids across all nodes.

#### Returns

**dict** : Keys are structure ids. Values are lists of ancestor ids.

**get\_colormap** (*self*)

Get a dictionary mapping structure ids to colors across all nodes.

#### Returns

**dict** : Keys are structure ids. Values are RGB lists of integers.

**get\_id\_acronym\_map** (*self*)

Get a dictionary mapping structure acronyms to ids across all nodes.

#### Returns

**dict** : Keys are structure acronyms. Values are structure ids.

**get\_name\_map** (*self*)

Get a dictionary mapping structure ids to names across all nodes.

#### Returns

**dict** : Keys are structure ids. Values are structure name strings.

**get\_structure\_sets** (*self*)

Lists all unique structure sets that are assigned to at least one structure in the tree.

#### Returns

**list of int** : Elements are ids of structure sets.

**get\_structures\_by\_acronym** (*self*, *acronyms*)

Obtain a list of brain structures from their acronyms

#### Parameters

**names** [list of str] Get structures corresponding to these acronyms.

**Returns**

**list of dict** : Each item describes a structure.

**get\_structures\_by\_id** (*self*, *structure\_ids*)

Obtain a list of brain structures from their structure ids

**Parameters**

**structure\_ids** [list of int] Get structures corresponding to these ids.

**Returns**

**list of dict** : Each item describes a structure.

**get\_structures\_by\_name** (*self*, *names*)

Obtain a list of brain structures from their names,

**Parameters**

**names** [list of str] Get structures corresponding to these names.

**Returns**

**list of dict** : Each item describes a structure.

**get\_structures\_by\_set\_id** (*self*, *structure\_set\_ids*)

Obtain a list of brain structures from by the sets that contain them.

**Parameters**

**structure\_set\_ids** [list of int] Get structures belonging to these structure sets.

**Returns**

**list of dict** : Each item describes a structure.

**has\_overlaps** (*self*, *structure\_ids*)

Determine if a list of structures contains structures along with their ancestors

**Parameters**

**structure\_ids** [list of int] Check this set of structures for overlaps

**Returns**

**set** : Ids of structures that are the ancestors of other structures in the supplied set.

**static hex\_to\_rgb** (*hex\_color*)

Convert a hexadecimal color string to a uint8 triplet

**Parameters**

**hex\_color** [string] Must be 6 characters long, unless it is 7 long and the first character is #. If hex\_color is a triplet of int, it will be returned unchanged.

**Returns**

**list of int** : 3 characters long - 1 per two characters in the input string.

**static path\_to\_list** (*path*)

Structure id paths are sometimes formatted as "/"-seperated strings. This method converts them to a list of integers, if needed.

**static renames** ()

**structure\_descends\_from** (*self*, *child\_id*, *parent\_id*)

Tests whether one structure descends from another.

**Parameters**

**child\_id** [int] Id of the putative child structure.

**parent\_id** [int] Id of the putative parent structure.

**Returns**

**bool**: True if the structure specified by **child\_id** is a descendant of the one specified by **parent\_id**. Otherwise False.

**static whitelist()**

**allensdk.core.swc module**

**class** allensdk.core.swc.**Compartment** (\*args, \*\*kwargs)

Bases: dict

A dictionary class storing information about a single morphology node

**print\_node** (self)

print out compartment information with field names

**class** allensdk.core.swc.**Marker** (\*args, \*\*kwargs)

Bases: dict

Simple dictionary class for handling reconstruction marker objects.

**CUT\_DENDRITE** = 10

**NO\_RECONSTRUCTION** = 20

**SPACING** = [0.1144, 0.1144, 0.28]

**class** allensdk.core.swc.**Morphology** (compartment\_list=None, compartment\_index=None)

Bases: object

Keep track of the list of compartments in a morphology and provide a few helper methods (soma, tree information, pruning, etc).

**APICAL\_DENDRITE** = 4

**AXON** = 2

**BASAL\_DENDRITE** = 3

**DENDRITE** = 3

**NODE\_TYPES** = [1, 2, 3, 3, 4]

**SOMA** = 1

**append** (self, node\_list)

Add additional nodes to this Morphology. Those nodes must originate from another morphology object.

**Parameters**

**node\_list**: list of Morphology nodes

**apply\_affine** (self, aff, scale=None)

Apply an affine transform to all compartments in this morphology. Node radius is adjusted as well.

Format of the affine matrix is:

[x0 y0 z0] [tx] [x1 y1 z1] [ty] [x2 y2 z2] [tz]



where the left 3x3 the matrix defines the affine rotation and scaling, and the right column is the translation vector.

The matrix must be collapsed and stored in a list as follows:

[x0 y0, z0, x1, y1, z1, x2, y2, z2, tx, ty, tz]

#### Parameters

**aff: 3x4 array of floats (python 2D list, or numpy 2D array)** the transformation matrix

**change\_parent** (*self, child, parent*)

Change the parent of a node. The child node is adjusted to point to the new parent, the child is taken off of the previous parent's child list, and it is added to the new parent's child list.

#### Parameters

**child: integer or Morphology Object** The ID of the child node, or the child node itself

**parent: integer or Morphology Object** The ID of the parent node, or the parent node itself

#### Returns

Nothing

**children\_of** (*self, seg*)

Returns a list of the children of the specified node

#### Parameters

**seg: integer or Morphology Object** The ID of the parent node, or the parent node itself

#### Returns

**A list of the child morphology objects. If the ID of the parent node is invalid, None is returned.**

**compartment\_index**

Return the compartment index. This is a property to ensure that the compartment list and compartment index are in sync.

**compartment\_index\_by\_type** (*self, compartment\_type*)

Return an dictionary of compartments indexed by id that all have a particular compartment type.

#### Parameters

**compartment\_type: int** Desired compartment type

#### Returns

**A dictionary of Morphology Objects, indexed by ID**

**compartment\_list**

Return the compartment list. This is a property to ensure that the compartment list and compartment index are in sync.

**compartment\_list\_by\_type** (*self, compartment\_type*)

Return an list of all compartments having the specified compartment type.

#### Parameters

**compartment\_type: int** Desired compartment type

#### Returns

### A list of of Morphology Objects

**convert\_type** (*self*, *old\_type*, *new\_type*)

Converts all compartments from one type to another. Nodes of the original type are not affected so this procedure can also be used as a merge procedure.

#### Parameters

**old\_type: enum** The compartment type to be changed. Use one of the following constants: SOMA, AXON, DENDRITE, BASAL\_DENDRITE, or APICAL\_DENDRITE

**new\_type: enum** The target compartment type. Use one of the following constants: SOMA, AXON, DENDRITE, BASAL\_DENDRITE, or APICAL\_DENDRITE

**delete\_tree** (*self*, *n*)

Delete tree, and all of its compartments, from the morphology.

#### Parameters

**n: Integer** The tree number to delete

**find** (*self*, *x*, *y*, *z*, *dist*, *node\_type=None*)

Returns a list of Morphology Objects located within 'dist' of coordinate (x,y,z). If node\_type is specified, the search will be constrained to return only nodes of that type.

#### Parameters

**x, y, z: float** The x,y,z coordinates from which to search around

**dist: float** The search radius

**node\_type: enum (optional)** One of the following constants: SOMA, AXON, DENDRITE, BASAL\_DENDRITE or APICAL\_DENDRITE

#### Returns

**A list of all Morphology Objects matching the search criteria**

**node** (*self*, *n*)

Returns the morphology node having the specified ID.

#### Parameters

**n: integer** ID of desired node

#### Returns

**A morphology object having the specified ID, or None if such a node doesn't exist**

**num\_nodes**

Return the number of compartments in the morphology.

**num\_trees**

Return the number of trees in the morphology. A tree is defined as everything following from a single root compartment.

**parent\_of** (*self*, *seg*)

Returns parent of the specified node.

#### Parameters

**seg: integer or Morphology Object** The ID of the child node, or the child node itself

#### Returns

A morphology object, or None if no parent exists or if the specified node ID doesn't exist

**root**

[deprecated] Returns root node of soma, if present. Use 'soma' instead of 'root'

**save** (*self*, *file\_name*)

Write this morphology out to an SWC file

**Parameters**

**file\_name: string** desired name of your SWC file

**soma**

Returns root node of soma, if present

**sparsify** (*self*, *modulo*, *compress\_ids=False*)

Return a new Morphology object that has a given number of non-leaf, non-root nodes removed. IDs can be reassigned so as to be continuous.

**Parameters**

**modulo: int** keep 1 out of every modulo nodes.

**compress\_ids: boolean** Reassign ids so that ids are continuous (no missing id numbers).

**Returns**

**Morphology** A new morphology instance

**strip\_all\_other\_types** (*self*, *node\_type*, *keep\_soma=True*)

Strips everything from the morphology except for the specified type. Parent and child relationships are updated accordingly, creating new roots when necessary.

**Parameters**

**node\_type: enum** The compartment type to keep in the morphology. Use one of the following constants: SOMA, AXON, DENDRITE, BASAL\_DENDRITE, or APICAL\_DENDRITE

**keep\_soma: Boolean (optional)** True (default) if soma nodes should remain in the morphology, and False if the soma should also be stripped

**strip\_type** (*self*, *node\_type*)

Strips all compartments of the specified type from the morphology. Parent and child relationships are updated accordingly, creating new roots when necessary.

**Parameters**

**node\_type: enum** The compartment type to strip from the morphology. Use one of the following constants: SOMA, AXON, DENDRITE, BASAL\_DENDRITE, or APICAL\_DENDRITE

**stumpify\_axon** (*self*, *count=10*)

Remove all axon compartments except the first 'count' nodes, as counted from the connected axon root.

**Parameters**

**count: Integer** The length of the axon 'stump', in number of compartments

**tree** (*self*, *n*)

Returns a list of all Morphology Nodes within the specified tree. A tree is defined as a fully connected graph of nodes. Each tree has exactly one root.

**Parameters**

**n: integer** ID of desired tree

#### Returns

A list of all morphology objects in the specified tree, or None  
if the tree doesn't exist

**write** (*self*, *file\_name*)

`allensdk.core.swc.read_marker_file` (*file\_name*)

read in a marker file and return a list of dictionaries

`allensdk.core.swc.read_swc` (*file\_name*, *columns*='NOT\_USED', *numeric\_columns*='NOT\_USED')

Read in an SWC file and return a Morphology object.

#### Parameters

**file\_name: string** SWC file name.

#### Returns

**Morphology** A Morphology instance.

### allensdk.core.typing module

**class** `allensdk.core.typing.SupportsStr`

Bases: `typing._Protocol`

Classes that support the `__str__` method

### Module contents

## 6.1.5 allensdk.ephys package

### Submodules

#### allensdk.ephys.ephys\_extractor module

**class** `allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor` (*ramps\_ext*,  
*short\_squares\_ext*,  
*long\_squares\_ext*,  
*subthresh\_min\_amp*=-100)

Bases: `object`

**SAG\_TARGET** = -100.0

**SUBTHRESH\_MAX\_AMP** = 0

**as\_dict** (*self*)

Create dict of cell features.

**cell\_features** (*self*)

**long\_squares\_features** (*self*, *option*=None)

**long\_squares\_stim\_amps** (*self*, *option*=None)

**process** (*self*, *keys=None*)

Processes features. Can take a specific key (or set of keys) to do a subset of processing.

**ramps\_features** (*self*, *all=False*)

**short\_squares\_features** (*self*)

```
class allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor (t=None,
                                                                v=None,
                                                                i=None,
                                                                start=None,
                                                                end=None,
                                                                filter=10.0,
                                                                dv_cutoff=20.0,
                                                                max_interval=0.005,
                                                                min_height=2.0,
                                                                min_peak=-
                                                                30.0,
                                                                thresh_frac=0.05,
                                                                base-
                                                                line_interval=0.1,
                                                                base-
                                                                line_detect_thresh=0.3,
                                                                id=None)
```

Bases: object

Feature calculation for a sweep (voltage and/or current time series).

**as\_dict** (*self*)

Create dict of features and spikes.

**burst\_metrics** (*self*)

Find bursts and return max “burstiness” index (normalized max rate in burst vs out).

#### Returns

**max\_burstiness\_index** [max “burstiness” index across detected bursts]

**num\_bursts** [number of bursts detected]

**delay\_metrics** (*self*)

Calculates ratio of latency to dominant time constant of rise before spike

#### Returns

**delay\_ratio** [ratio of latency to tau (higher means more delay)]

**tau** [dominant time constant of rise before spike]

**estimate\_sag** (*self*, *peak\_width=0.005*)

Calculate the sag in a hyperpolarizing voltage response.

#### Parameters

**peak\_width** [window width to get more robust peak estimate in sec (default 0.005)]

#### Returns

**sag** [fraction that membrane potential relaxes back to baseline]

**estimate\_time\_constant** (*self*)

Calculate the membrane time constant by fitting the voltage response with a single exponential.

#### Returns

**tau** [membrane time constant in seconds]

**is\_spike\_feature\_affected\_by\_clipping** (*self*, *key*)

**pause\_metrics** (*self*)

Estimate average number of pauses and average fraction of time spent in a pause

Attempts to detect pauses with a variety of conditions and averages results together.

Pauses that are consistently detected contribute more to estimates.

#### Returns

**avg\_n\_pauses** [average number of pauses detected across conditions]

**avg\_pause\_frac** [average fraction of interval (between start and end) spent in a pause]

**max\_reliability** [max fraction of times most reliable pause was detected given weights tested]

**n\_max\_rel\_pauses** [number of pauses detected with *max\_reliability*]

**process\_new\_spike\_feature** (*self*, *feature\_name*, *feature\_func*, *affected\_by\_clipping=False*)

Add new spike-level feature calculation function

The function should take this sweep extractor as its argument. Its results can be accessed by calling the method `spike_feature(<feature_name>)`.

**process\_new\_sweep\_feature** (*self*, *feature\_name*, *feature\_func*)

Add new sweep-level feature calculation function

The function should take this sweep extractor as its argument. Its results can be accessed by calling the method `sweep_feature(<feature_name>)`.

**process\_spikes** (*self*)

Perform spike-related feature analysis

**set\_stimulus\_amplitude\_calculator** (*self*, *function*)

**spike\_feature** (*self*, *key*, *include\_clipped=False*, *force\_exclude\_clipped=False*)

Get specified feature for every spike.

#### Parameters

**key** [feature name]

**include\_clipped:** return values for every identified spike, even when clipping means they will be incorrect/

#### Returns

**spike\_feature\_values** [ndarray of features for each spike]

**spike\_feature\_keys** (*self*)

Get list of every available spike feature.

**spikes** (*self*)

Get all features for each spike as a list of records.

**stimulus\_amplitude** (*self*)

**sweep\_feature** (*self*, *key*, *allow\_missing=False*)

Get sweep-level feature (*key*).

#### Parameters

**key** [name of sweep-level feature]

**allow\_missing** [return np.nan if key is missing for sweep (default False)]

#### Returns

**sweep\_feature** [sweep-level feature value]

**sweep\_feature\_keys** (*self*)

Get list of every available sweep-level feature.

**voltage\_deflection** (*self*, *deflect\_type=None*)

Measure deflection (min or max, between start and end if specified).

#### Parameters

**deflect\_type** [measure minimal ('min') or maximal ('max') voltage deflection] If not specified, it will check to see if the current (i) is positive or negative between start and end, then choose 'max' or 'min', respectively If the current is not defined, it will default to 'min'.

#### Returns

**deflect\_v** [peak]

**deflect\_index** [index of peak deflection]

```
class allensdk.ephys.ephys_extractor.EphysSweepSetFeatureExtractor (t_set=None,
                                                                    v_set=None,
                                                                    i_set=None,
                                                                    start=None,
                                                                    end=None,
                                                                    filter=10.0,
                                                                    dv_cutoff=20.0,
                                                                    max_interval=0.005,
                                                                    min_height=2.0,
                                                                    min_peak=-30.0,
                                                                    thresh_frac=0.05,
                                                                    base_line_interval=0.1,
                                                                    base_line_detect_thresh=0.3,
                                                                    id_set=None)
```

Bases: object

**classmethod from\_sweeps** (*sweep\_list*)

Initialize EphysSweepSetFeatureExtractor object with a list of pre-existing sweep feature extractor objects.

**process\_spikes** (*self*)

Analyze spike features for all sweeps.

**spike\_feature\_averages** (*self*, *key*)

Get nparray of average spike-level feature (*key*) for all sweeps

**sweep\_features** (*self*, *key*, *allow\_missing=False*)

Get nparray of sweep-level feature (*key*) for all sweeps

#### Parameters

**key** [name of sweep-level feature]

**allow\_missing** [return np.nan if key is missing for sweep (default False)]

**Returns**

**sweep\_feature** [ndarray of sweep-level feature values]

**sweeps** (*self*)

Get list of EphysSweepFeatureExtractor objects.

```
allensdk.ephys.ephys_extractor.cell_extractor_for_nwb(dataset,  
                                                    ramps,  
                                                    short_squares, long_squares,  
                                                    subthresh_min_amp=-100)
```

Initialize EphysCellFeatureExtractor object from NWB data set

**Parameters**

**dataset** [NwbDataSet]

**ramps** [list of sweep numbers of ramp sweeps]

**short\_squares** [list of sweep numbers of short square sweeps]

**long\_squares** [list of sweep numbers of long square sweeps]

```
allensdk.ephys.ephys_extractor.extractor_for_nwb_sweeps(dataset, sweep_numbers,  
                                                       fixed_start=None,  
                                                       fixed_end=None,  
                                                       dv_cutoff=20.0,  
                                                       thresh_frac=0.05)
```

```
allensdk.ephys.ephys_extractor.fit_fit_slope(ext)
```

Fit the rate and stimulus amplitude to a line and return the slope of the fit.

```
allensdk.ephys.ephys_extractor.input_resistance(ext)
```

Estimate input resistance in MOhms, assuming all sweeps in passed extractor are hyperpolarizing responses.

```
allensdk.ephys.ephys_extractor.membrane_time_constant(ext)
```

Average the membrane time constant values estimated from each sweep in passed extractor.

```
allensdk.ephys.ephys_extractor.reset_long_squares_start(when)
```

**allensdk.ephys.ephys\_features module**

**exception** allensdk.ephys.ephys\_features.FeatureError

Bases: Exception

Generic Python-exception-derived object raised by feature detection functions.

```
allensdk.ephys.ephys_features.adaptation_index(isis)
```

Calculate adaptation index of *isis*.

```
allensdk.ephys.ephys_features.analyze_trough_details(v, t, spike_indexes,  
                                                    peak_indexes, clipped=None,  
                                                    end=None, filter=10.0,  
                                                    heavy_filter=1.0,  
                                                    term_frac=0.01,  
                                                    adp_thresh=0.5, tol=0.5,  
                                                    flat_interval=0.002,  
                                                    adp_max_delta_t=0.005,  
                                                    adp_max_delta_v=10.0,  
                                                    dvd_t=None)
```

Analyze trough to determine if an ADP exists and whether the reset is a 'detour' or 'direct'

**Parameters**



**v** [numpy array of voltage time series in mV]  
**t** [numpy array of times in seconds]  
**spike\_indexes** [numpy array of spike indexes]  
**peak\_indexes** [numpy array of spike peak indexes]  
**end** [end of time window (optional)]  
**filter** [cutoff frequency for 4-pole low-pass Bessel filter in kHz (default 1)]  
**heavy\_filter** [lower cutoff frequency for 4-pole low-pass Bessel filter in kHz (default 1)]  
**thresh\_frac** [fraction of average upstroke for threshold calculation (optional, default 0.05)]  
**adp\_thresh: minimum dV/dt in V/s to exceed to be considered to have an ADP (optional, default 1.5)**  
  
**tol** [tolerance for evaluating whether Vm drops appreciably further after end of spike (default 1.0 mV)]  
**flat\_interval: if the trace is flat for this duration, stop looking for an ADP (default 0.002 s)**  
  
**adp\_max\_delta\_t: max possible ADP delta t (default 0.005 s)**  
**adp\_max\_delta\_v: max possible ADP delta v (default 10 mV)**  
**dvdv** [pre-calculated time-derivative of voltage (optional)]

#### Returns

**isi\_types** [numpy array of isi reset types (direct or detour)]  
**fast\_trough\_indexes** [numpy array of indexes at the start of the trough (i.e. end of the spike)]  
**adp\_indexes** [numpy array of adp indexes (np.nan if there was no ADP in that ISI)]  
**slow\_trough\_indexes** [numpy array of indexes at the minimum of the slow phase of the trough] (if there wasn't just a fast phase)

`allensdk.ephys.ephys_features.average_rate(t, spikes, start, end)`  
 Calculate average firing rate during interval between *start* and *end*.

#### Parameters

**t** [numpy array of times in seconds]  
**spikes** [numpy array of spike indexes]  
**start** [start of time window for spike detection]  
**end** [end of time window for spike detection]

#### Returns

**avg\_rate** [average firing rate in spikes/sec]

`allensdk.ephys.ephys_features.average_voltage(v, t, start=None, end=None)`  
 Calculate average voltage between start and end.

#### Parameters

**v** [numpy array of voltage time series in mV]  
**t** [numpy array of times in seconds]  
**start** [start of time window for spike detection (optional, default None)]

**end** [end of time window for spike detection (optional, default None)]

#### Returns

**v\_avg** [average voltage]

`allensdk.ephys.ephys_features.calculate_dvdt` (*v*, *t*, *filter=None*)

Low-pass filters (if requested) and differentiates voltage by time.

#### Parameters

**v** [numpy array of voltage time series in mV]

**t** [numpy array of times in seconds]

**filter** [cutoff frequency for 4-pole low-pass Bessel filter in kHz (optional, default None)]

#### Returns

**dvdt** [numpy array of time-derivative of voltage (V/s = mV/ms)]

`allensdk.ephys.ephys_features.check_thresholds_and_peaks` (*v*, *t*, *spike\_indexes*,  
*peak\_indexes*, *upstroke\_indexes*,  
*end=None*,  
*max\_interval=0.005*,  
*thresh\_frac=0.05*, *filter=10.0*, *dvdt=None*,  
*tol=1.0*)

Validate thresholds and peaks for set of spikes

Check that peaks and thresholds for consecutive spikes do not overlap Spikes with overlapping thresholds and peaks will be merged.

Check that peaks and thresholds for a given spike are not too far apart.

#### Parameters

**v** [numpy array of voltage time series in mV]

**t** [numpy array of times in seconds]

**spike\_indexes** [numpy array of spike indexes]

**peak\_indexes** [numpy array of indexes of spike peaks]

**upstroke\_indexes** [numpy array of indexes of spike upstrokes]

**max\_interval** [maximum allowed time between start of spike and time of peak in sec (default 0.005)]

**thresh\_frac** [fraction of average upstroke for threshold calculation (optional, default 0.05)]

**filter** [cutoff frequency for 4-pole low-pass Bessel filter in kHz (optional, default 10)]

**dvdt** [pre-calculated time-derivative of voltage (optional)]

**tol** [tolerance for returning to threshold in mV (optional, default 1)]

#### Returns

**spike\_indexes** [numpy array of modified spike indexes]

**peak\_indexes** [numpy array of modified spike peak indexes]

**upstroke\_indexes** [numpy array of modified spike upstroke indexes]

**clipped** [numpy array of clipped status of spikes]

`allensdk.ephys.ephys_features.detect_bursts` (*isis, isi\_types, fast\_tr\_v, fast\_tr\_t, slow\_tr\_v, slow\_tr\_t, thr\_v, tol=0.5, pause\_cost=1.0*)

Detect bursts in spike train.

#### Parameters

**isis** [numpy array of n interspike intervals]  
**isi\_types** [numpy array of n interspike interval types]  
**fast\_tr\_v** [numpy array of fast trough voltages for the n + 1 spikes of the train]  
**fast\_tr\_t** [numpy array of fast trough times for the n + 1 spikes of the train]  
**slow\_tr\_v** [numpy array of slow trough voltages for the n + 1 spikes of the train]  
**slow\_tr\_t** [numpy array of slow trough times for the n + 1 spikes of the train]  
**thr\_v** [numpy array of threshold voltages for the n + 1 spikes of the train]  
**tol** [tolerance for the difference in slow trough voltages and thresholds (default 0.5 mV)] Used to identify “delay” interspike intervals that occur within a burst

#### Returns

**bursts** [list of bursts] Each item in list is a tuple of the form (burst\_index, start, end) where *burst\_index* is a comparison index between the highest instantaneous rate within the burst vs the highest instantaneous rate outside the burst. *start* is the index of the first ISI of the burst, and *end* is the ISI index immediately following the burst.

`allensdk.ephys.ephys_features.detect_pauses` (*isis, isi\_types, cost\_weight=1.0*)

Determine which ISIs are “pauses” in ongoing firing.

Pauses are unusually long ISIs with a “detour reset” among “direct resets”.

#### Parameters

**isis** [numpy array of interspike intervals]  
**isi\_types** [numpy array of interspike interval types (‘direct’ or ‘detour’)]  
**cost\_weight** [weight for cost function for calling an ISI a pause] Higher cost weights lead to fewer ISIs identified as pauses. The cost function also depends on the difference between the duration of the “pause” ISIs and the average duration and standard deviation of “non-pause” ISIs.

#### Returns

**pauses** [numpy array of indices corresponding to pauses in *isis*]

`allensdk.ephys.ephys_features.detect_putative_spikes` (*v, t, start=None, end=None, filter=10.0, dv\_cutoff=20.0*)

Perform initial detection of spikes and return their indexes.

#### Parameters

**v** [numpy array of voltage time series in mV]  
**t** [numpy array of times in seconds]  
**start** [start of time window for spike detection (optional)]  
**end** [end of time window for spike detection (optional)]  
**filter** [cutoff frequency for 4-pole low-pass Bessel filter in kHz (optional, default 10)]  
**dv\_cutoff** [minimum dV/dt to qualify as a spike in V/s (optional, default 20)]

**dvd** [pre-calculated time-derivative of voltage (optional)]

#### Returns

**putative\_spikes** [numpy array of preliminary spike indexes]

```
allensdk.ephys.ephys_features.estimate_adjusted_detection_parameters(v_set,  
                                                                    t_set,  
                                                                    inter-  
                                                                    val_start,  
                                                                    inter-  
                                                                    val_end,  
                                                                    fil-  
                                                                    ter=10)
```

Estimate adjusted values for spike detection by analyzing a period when the voltage changes quickly but passively (due to strong current stimulation), which can result in spurious spike detection results.

#### Parameters

**v\_set** [list of numpy arrays of voltage time series in mV]

**t\_set** [list of numpy arrays of times in seconds]

**interval\_start** [start of analysis interval (sec)]

**interval\_end** [end of analysis interval (sec)]

#### Returns

**new\_dv\_cutoff** [adjusted dv/dt cutoff (V/s)]

**new\_thresh\_frac** [adjusted fraction of avg upstroke to find threshold]

```
allensdk.ephys.ephys_features.filter_putative_spikes(v, t, spike_indexes,  
                                                    peak_indexes, min_height=2.0,  
                                                    min_peak=-30.0, filter=10.0,  
                                                    dvd=None)
```

#### Filter out events that are unlikely to be spikes based on:

- Voltage failing to go down between peak and the next spike's threshold
- Height (threshold to peak)
- Absolute peak level

#### Parameters

**v** [numpy array of voltage time series in mV]

**t** [numpy array of times in seconds]

**spike\_indexes** [numpy array of preliminary spike indexes]

**peak\_indexes** [numpy array of indexes of spike peaks]

**min\_height** [minimum acceptable height from threshold to peak in mV (optional, default 2)]

**min\_peak** [minimum acceptable absolute peak level in mV (optional, default -30)]

**filter** [cutoff frequency for 4-pole low-pass Bessel filter in kHz (optional, default 10)]

**dvd** [pre-calculated time-derivative of voltage (optional)]

#### Returns

**spike\_indexes** [numpy array of threshold indexes]

**peak\_indexes** [numpy array of peak indexes]

```
allensdk.ephys.ephys_features.find_downstroke_indexes(v, t, peak_indexes,
                                                         trough_indexes,
                                                         clipped=None, filter=10.0,
                                                         dvdt=None)
```

Find indexes of minimum voltage (troughs) between spikes.

#### Parameters

**v** [numpy array of voltage time series in mV]

**t** [numpy array of times in seconds]

**peak\_indexes** [numpy array of spike peak indexes]

**trough\_indexes** [numpy array of threshold indexes]

**clipped:** boolean array - False if spike not clipped by edge of window

**filter** [cutoff frequency for 4-pole low-pass Bessel filter in kHz (optional, default 10)]

**dvdt** [pre-calculated time-derivative of voltage (optional)]

#### Returns

**downstroke\_indexes** [numpy array of downstroke indexes]

```
allensdk.ephys.ephys_features.find_peak_indexes(v, t, spike_indexes, end=None)
```

Find indexes of spike peaks.

#### Parameters

**v** [numpy array of voltage time series in mV]

**t** [numpy array of times in seconds]

**spike\_indexes** [numpy array of preliminary spike indexes]

**end** [end of time window for spike detection (optional)]

```
allensdk.ephys.ephys_features.find_time_index(t, t_0)
```

Find the index value of a given time (t\_0) in a time series (t).

```
allensdk.ephys.ephys_features.find_trough_indexes(v, t, spike_indexes, peak_indexes,
                                                         clipped=None, end=None)
```

Find indexes of minimum voltage (trough) between spikes.

#### Parameters

**v** [numpy array of voltage time series in mV]

**t** [numpy array of times in seconds]

**spike\_indexes** [numpy array of spike indexes]

**peak\_indexes** [numpy array of spike peak indexes]

**end** [end of time window (optional)]

#### Returns

**trough\_indexes** [numpy array of threshold indexes]

```
allensdk.ephys.ephys_features.find_upstroke_indexes(v, t, spike_indexes, peak_indexes,
                                                         filter=10.0, dvdt=None)
```

Find indexes of maximum upstroke of spike.

**Parameters**

**v** [numpy array of voltage time series in mV]

**t** [numpy array of times in seconds]

**spike\_indexes** [numpy array of preliminary spike indexes]

**peak\_indexes** [numpy array of indexes of spike peaks]

**filter** [cutoff frequency for 4-pole low-pass Bessel filter in kHz (optional, default 10)]

**dvdv** [pre-calculated time-derivative of voltage (optional)]

**Returns**

**upstroke\_indexes** [numpy array of upstroke indexes]

`allensdk.ephys.ephys_features.find_widths(v, t, spike_indexes, peak_indexes, trough_indexes, clipped=None)`

Find widths at half-height for spikes.

Widths are only returned when heights are defined

**Parameters**

**v** [numpy array of voltage time series in mV]

**t** [numpy array of times in seconds]

**spike\_indexes** [numpy array of spike indexes]

**peak\_indexes** [numpy array of spike peak indexes]

**trough\_indexes** [numpy array of trough indexes]

**Returns**

**widths** [numpy array of spike widths in sec]

`allensdk.ephys.ephys_features.fit_membrane_time_constant(v, t, start, end, min_rsme=0.0001)`

Fit an exponential to estimate membrane time constant between start and end

**Parameters**

**v** [numpy array of voltages in mV]

**t** [numpy array of times in seconds]

**start** [start of time window for exponential fit]

**end** [end of time window for exponential fit]

**min\_rsme: minimal acceptable root mean square error (default 1e-4)**

**Returns**

**a, inv\_tau, y0** [Coefficients of equation  $y_0 + a * \exp(-\text{inv\_tau} * x)$ ]

**returns np.nan for values if fit fails**

`allensdk.ephys.ephys_features.fit_prespike_time_constant(v, t, start, spike_time, dv_limit=-0.001, tau_limit=0.3)`

Finds the dominant time constant of the pre-spike rise in voltage

**Parameters**

**v** [numpy array of voltage time series in mV]

**t** [numpy array of times in seconds]  
**start** [start of voltage rise (seconds)]  
**spike\_time** [time of first spike (seconds)]  
**dv\_limit** [dV/dt cutoff (default -0.001)] Shortens fit window if rate of voltage drop exceeds this limit  
**tau\_limit** [upper bound for slow time constant (seconds, default 0.3)] If the slower time constant of a double-exponential fit is twice that of the faster and exceeds this limit, the faster one will be considered the dominant one

#### Returns

**tau** [dominant time constant (seconds)]

`allensdk.ephys.ephys_features.get_isis(t, spikes)`  
 Find interspike intervals in sec between spikes (as indexes).

`allensdk.ephys.ephys_features.has_fixed_dt(t)`  
 Check that all time intervals are identical.

`allensdk.ephys.ephys_features.latency(t, spikes, start)`  
 Calculate time to the first spike.

`allensdk.ephys.ephys_features.norm_diff(a)`  
 Calculate average of  $(a[i] - a[i+1]) / (a[i] + a[i+1])$ .

`allensdk.ephys.ephys_features.norm_sq_diff(a)`  
 Calculate average of  $(a[i] - a[i+1])^2 / (a[i] + a[i+1])^2$ .

`allensdk.ephys.ephys_features.refine_threshold_indexes(v, t, upstroke_indexes, thresh_frac=0.05, filter=10.0, dvdt=None)`  
 Refine threshold detection of previously-found spikes.

#### Parameters

**v** [numpy array of voltage time series in mV]  
**t** [numpy array of times in seconds]  
**upstroke\_indexes** [numpy array of indexes of spike upstrokes (for threshold target calculation)]  
**thresh\_frac** [fraction of average upstroke for threshold calculation (optional, default 0.05)]  
**filter** [cutoff frequency for 4-pole low-pass Bessel filter in kHz (optional, default 10)]  
**dvdt** [pre-calculated time-derivative of voltage (optional)]

#### Returns

**threshold\_indexes** [numpy array of threshold indexes]

### allensdk.ephys.extract\_cell\_features module

`allensdk.ephys.extract_cell_features.extract_cell_features(data_set, ramp_sweep_numbers, short_square_sweep_numbers, long_square_sweep_numbers, sub_thresh_min_amp=None)`

```
allensdk.ephys.extract_cell_features.extract_sweep_features (data_set,  
                                                             sweeps_by_type)  
allensdk.ephys.extract_cell_features.get_ramp_stim_characteristics (i, t)  
    Identify the start time and start index of a ramp sweep.  
allensdk.ephys.extract_cell_features.get_square_stim_characteristics (i, t,  
                                                                    no_test_pulse=False)  
    Identify the start time, duration, amplitude, start index, and end index of a square stimulus. This assumes that  
    there is a test pulse followed by the stimulus square.  
allensdk.ephys.extract_cell_features.get_stim_characteristics (i, t,  
                                                            no_test_pulse=False)  
    Identify the start time, duration, amplitude, start index, and end index of a general stimulus. This assumes that  
    there is a test pulse followed by the stimulus square.  
allensdk.ephys.extract_cell_features.mean_features_spike_zero (sweeps)  
    Compute mean feature values for the first spike in list of extractors
```

## allensdk.ephys.feature\_extractor module

```
class allensdk.ephys.feature_extractor.EphysFeatureExtractor  
    Bases: object  
        adaptation_index (self, spikes, stim_end)  
        calculate_trough (self, spike, v, curr, t, next_idx)  
        isicv (self, spikes)  
        process_instance (self, name, v, curr, t, onset, dur, stim_name)  
        push_summary (self, new_summary)  
        score_feature_set (self, set_num)  
        summarize (self, summary)  
class allensdk.ephys.feature_extractor.EphysFeatures (name)  
    Bases: object  
        clone (self, param_dict)  
        print_out (self)
```

## Module contents

### 6.1.6 allensdk.internal package

#### Subpackages

#### allensdk.internal.api package

#### Subpackages

#### allensdk.internal.api.queries package



## Submodules

### allensdk.internal.api.queries.biophysical\_module\_api module

**class** allensdk.internal.api.queries.biophysical\_module\_api.**BiophysicalModuleApi** (*base\_uri=None*)

Bases: *allensdk.api.queries.rma\_template.RmaTemplate*

**get\_neuronal\_model\_runs** (*self, neuronal\_model\_run\_ids=None*)

List Neuronal Model Rusn available through LIMS with associated info needed to run in NEURON.

#### Parameters

**neuronal\_model\_run\_ids** [integer or list of integers, optional] only select specific neuronal\_model\_runs.

#### Returns

**dict** [neuronal model run metadata]

**get\_neuronal\_models** (*self, neuronal\_model\_ids=None*)

List Neuronal Models available through LIMS with associated info needed to run in NEURON.

#### Parameters

**neuronal\_model\_ids** [integer or list of integers, optional] only select specific neuronal\_models.

#### Returns

**dict** [neuronal model metadata]

**rma\_templates** = {'biophysical\_lims\_queries': [{'name': 'neuronal\_model\_runs\_by\_ids',

### allensdk.internal.api.queries.biophysical\_module\_reader module

**class** allensdk.internal.api.queries.biophysical\_module\_reader.**BiophysicalModuleReader**

Bases: object

**MOD\_FILE\_TYPE\_ID** = 292178729

**MORPHOLOGY\_TYPE\_ID** = 303941301

**STIMULUS\_CONTENT\_TYPE** = None

**fit\_parameters\_file\_entries** (*self*)

read the fit\_parameter file path from the lims result corresponding to the stimulus file :return: well\_known\_file entries :rtype: array of dicts

**fit\_parameters\_path** (*self*)

Get the path to the fit parameters file from the lims result. :return: path to file :rtype: string

**lims\_working\_directory** (*self*)

While this is the same directory as the neuronal\_model\_run directory, it can be mocked out for testing if the other directory is read only.

**mod\_file\_entries** (*self*)

read the NERUON .mod file entries from the lims result corresponding to the NeuronModel :return: well known file entries :rtype: array of dicts

**mod\_file\_paths** (*self*)

Get the paths to the mod files from the lims result. :return: paths to mod files :rtype: array of strings

**model\_type** (*self*)

TODO: comment

**morphology\_file\_entries** (*self*)

read the well known file paths from the lims result corresponding to the morphology

**Returns**

**array of dicts:** well known file entries

**morphology\_path** (*self*)

Get the path to the morphology file from the lims result. :return: path to morphology file :rtype: string

**neuronal\_model\_run\_dir** (*self*)

read the directory path where output goes from the lims optimization config json

**Returns**

**string:** directory path

**read\_json** (*self*, *path*)

**read\_json\_string** (*self*, *json\_string*)

**read\_lims\_file** (*self*, *lims\_path*)

**read\_lims\_message** (*self*, *message*, *lims\_path*)

**set\_workflow\_state** (*self*, *state*)

**stimulus\_file\_entries** (*self*)

read the well known file path from the lims result corresponding to the stimulus file :return: well-known\_file entries :rtype: array of dicts

**stimulus\_path** (*self*)

Get the path to the stimulus file from the lims result. :return: path to stimulus file :rtype: string

**sweep\_entries** (*self*)

read the sweep entries from the lims result corresponding to the stimulus :return: stimulus sweep entries :rtype: array of dicts

**sweep\_numbers** (*self*)

Get the stimulus sweep numbers from the lims result :return: list of sweep numbers :rtype: array of ints

**sweep\_numbers\_by\_type** (*self*)

**to\_manifest** (*self*, *manifest\_path*=None)

**update\_well\_known\_file** (*self*, *path*, *well\_known\_file\_type\_id*=None)

**write\_file** (*self*, *path*)

## allensdk.internal.api.queries.grid\_data\_api\_prerelease module

**class** allensdk.internal.api.queries.grid\_data\_api\_prerelease.**GridDataApiPrerelease** (*storage\_dir*, *res-*

*o-*

*lu-*

*tion*=None

*base\_uri*=None)

Bases: *allensdk.api.queries.grid\_data\_api.GridDataApi*

Client for retrieving prereleased mouse connectivity data from lims.

**Parameters**

**base\_uri** [string, optional] Does not affect pulling from lims.

**file\_name** [string, optional] File name to save/read storage\_directories dict. Passed to GridDataApiPrerelease constructor.

**GRID\_DATA\_DIRECTORY** = 'grid'

**download\_projection\_grid\_data** (*self, path, experiment\_id, file\_name*)

Copy data from path to file\_name.

**Parameters**

**path** [string] path to file in shared directory (copy source)

**experiment\_id** [int] image series id.

**file\_name** [string] path to file destination (copy target)

**classmethod from\_file\_name** (*file\_name, cache=True, \*\*kwargs*)

Alternative constructor using cache path file\_name.

**Parameters**

**file\_name** [string] Path where storage\_directories will be saved.

**\*\*kwargs** Keyword arguments to be supplied to `__init__`

**Returns**

**cls** [instance of GridDataApiPrerelease]

**allensdk.internal.api.queries.mouse\_connectivity\_api\_prerelease module**

**class** allensdk.internal.api.queries.mouse\_connectivity\_api\_prerelease.**MouseConnectivityApi**

Bases: *allensdk.api.queries.mouse\_connectivity\_api.MouseConnectivityApi*

Client for retrieving prereleased mouse connectivity data from lims.

**Parameters**

**base\_uri** [string, optional] Does not affect pulling from lims.

**file\_name** [string, optional] File name to save/read storage\_directories dict. Passed to GridDataApiPrerelease constructor.

**download\_data\_mask** (*self, path, experiment\_id, resolution*)

**download\_injection\_density** (*self, path, experiment\_id, resolution*)

**download\_injection\_fraction** (*self, path, experiment\_id, resolution*)

**download\_projection\_density** (*self, path, experiment\_id, resolution*)

**get\_experiments** (*self*)

Fetch experiment metadata from the Mouse Brain Connectivity Atlas.

**Parameters**

**structure\_ids** [integer or list, optional] injection structure

**Returns**

**url** [string] The constructed URL

```
get_structure_unionizes (self)
```

### **allensdk.internal.api.queries.optimize\_config\_reader module**

```
class allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader
```

```
    Bases: object
```

```
    MOD_FILE_TYPE_ID = 292178729
```

```
    MORPHOLOGY_TYPE_ID = 303941301
```

```
    NEURONAL_MODEL_PARAMETERS = 329230374
```

```
    STIMULUS_CONTENT_TYPE = None
```

```
    build_manifest (self, manifest_path=None)
```

```
    lims_working_directory (self)
```

While this is the same directory as the optimize directory, it can be mocked out for testing if the optimize directory is write only.

```
    mod_file_entries (self)
```

read the NERUON .mod file entries from the lims result corresponding to the NeuronModel :return: well known file entries :rtype: array of dicts

```
    mod_file_paths (self)
```

Get the paths to the mod files from the lims result. :return: paths to mod files :rtype: array of strings

```
    morphology_file_entries (self)
```

read the well known file paths from the lims result corresponding to the morphology

#### **Returns**

**array of dicts:** well known file entries

```
    morphology_path (self)
```

Get the path to the morphology file from the lims result. :return: path to morphology file :rtype: string

```
    neuronal_model_optimize_dir (self)
```

read the directory path where output goes from the lims optimization config json

#### **Returns**

**string:** directory path

```
    output_directory (self)
```

```
    read_json (self, path)
```

```
    read_json_string (self, json_string)
```

```
    read_lims_file (self, lims_path)
```

```
    read_lims_message (self, message, lims_path)
```

```
    stimulus_file_entries (self)
```

read the well known file path from the lims result corresponding to the stimulus file :return: well-known\_file entries :rtype: array of dicts

```
    stimulus_path (self)
```

Get the path to the stimulus file from the lims result. :return: path to stimulus file :rtype: string

**sweep\_entries** (*self*)  
 read the sweep entries from the lims result corresponding to the stimulus :return: stimulus sweep entries  
 :rtype: array of dicts

**sweep\_numbers** (*self*)  
 Get the stimulus sweep numbers from the lims result :return: list of sweep numbers :rtype: array of ints

**to\_manifest** (*self*, *manifest\_path=None*)

**update\_well\_known\_file** (*self*, *path*, *well\_known\_file\_type\_id=None*)

**write\_file** (*self*, *path*)

## allensdk.internal.api.queries.pre\_release module

**class** allensdk.internal.api.queries.pre\_release.**BrainObservatoryApiPreRelease** (*base\_uri=None*,  
*dat-*  
*acube\_uri=None*)

Bases: *allensdk.api.queries.brain\_observatory\_api.BrainObservatoryApi*

**get\_cell\_metrics** (*self*)

Get cell metrics by id

### Parameters

**cell\_metrics\_ids** [integer or list of integers, optional] only select specific cell metric records.

### Returns

**dict** [cell metric metadata]

**get\_experiment\_containers** (*self*)

Get experiment container by id

### Parameters

**experiment\_container\_ids** [integer or list of integers, optional] only select specific experiment containers.

### Returns

**dict** [experiment container metadata]

**get\_ophys\_experiments** (*self*)

Get OPhys Experiments by id

### Parameters

**ophys\_experiment\_ids** [integer or list of integers, optional] only select specific experiments.

### Returns

**dict** [ophys experiment metadata]

## Module contents

## Submodules

**allensdk.internal.api.api\_prerelease module**

**class** allensdk.internal.api.api\_prerelease.**ApiPrerelease** (*api\_base\_url\_string=None*)

Bases: *allensdk.api.api.Api*

Extends allensdk.api.api to copy files ‘locally’ from shared storage.

**retrieve\_file\_from\_storage** (*self, storage\_path, save\_file\_path*)

Copy data from path to file\_name.

**Parameters**

**storage\_path** [string] path to file in shared directory (copy source)

**save\_file\_name** [string] path to file destination (copy target)

**allensdk.internal.api.behavior\_data\_lims\_api module**

**class** allensdk.internal.api.behavior\_data\_lims\_api.**BehaviorDataLimsApi** (*behavior\_session\_id:*  
*int,*  
*lims\_credentials:*  
*Optional[allensdk.core.authentication.Authentication]*  
*=*  
*None,*  
*mtrain\_credentials:*  
*Optional[allensdk.core.authentication.Authentication]*  
*=*  
*None*)

Bases: *allensdk.core.cache\_method\_utilities.CachedInstanceMethodMixin,*  
*allensdk.brain\_observatory.behavior.internal.behavior\_base.BehaviorBase*

**get\_age** (*self*) → str

Returns age code of the subject. :rtype: str

**get\_behavior\_session\_id** (*self*) → int

Getter to be consistent with BehaviorOphysLimsApi.

**get\_behavior\_session\_uuid** (*self*) → Union[int, NoneType]

**get\_behavior\_stimulus\_file** (*self*) → str

Return the path to the StimulusPickle file for a session. :rtype: str

**get\_birth\_date** (*self*) → <method ‘date’ of ‘datetime.datetime’ objects>

Returns the birth date of the animal. :rtype: datetime.date

**get\_driver\_line** (*self*) → List[str]

Returns the genotype name(s) of the driver line(s). :rtype: list

**get\_experiment\_date** (*self*) → datetime.datetime

Return timestamp the behavior stimulus file began recording in UTC :rtype: datetime

**get\_external\_specimen\_name** (*self*) → int

Returns the LabTracks ID :rtype: int

**get\_full\_genotype** (*self*) → str

Return the name of the subject’s genotype :rtype: str

**get\_licks** (*self*) → pandas.core.frame.DataFrame

Get lick data from pkl file. This function assumes that the first sensor in the list of lick\_sensors is the desired lick sensor. If this changes we need to update to get the proper line.

Since licks can occur outside of a trial context, the lick times are extracted from the vsyncs and the frame number in *lick\_events*. Since we don't have a timestamp for when in "experiment time" the vsync stream starts (from self.get\_stimulus\_timestamps), we compute it by fitting a linear regression (frame number x time) for the *start\_trial* and *end\_trial* events in the *trial\_log*, to true up these time streams.

**Returns** pd.DataFrame – A dataframe containing lick timestamps

**get\_metadata** (*self*) → Dict[str, Any]

Return metadata about the session. :rtype: dict

**get\_reporter\_line** (*self*) → List[str]

Returns the genotype name(s) of the reporter line(s). :rtype: list

**get\_rewards** (*self*) → pandas.core.frame.DataFrame

Get reward data from pkl file, based on pkl file timestamps (not sync file).

**Returns** pd.DataFrame – A dataframe containing timestamps of delivered rewards.

**get\_rig\_name** (*self*) → str

Returns the name of the experimental rig. :rtype: str

**get\_running\_data\_df** (*self*) → pandas.core.frame.DataFrame

Get running speed data.

**Returns** pd.DataFrame – dataframe containing various signals used to compute running speed.

**get\_running\_speed** (*self*) → allensdk.brain\_observatory.running\_speed.RunningSpeed

Get running speed using timestamps from self.get\_stimulus\_timestamps.

NOTE: Do not correct for monitor delay.

**Returns** RunningSpeed – a NamedTuple containing the subject's timestamps and running speeds (in cm/s)

**get\_sex** (*self*) → str

Returns sex of the animal (M/F) :rtype: str

**get\_stimulus\_frame\_rate** (*self*) → float

**get\_stimulus\_name** (*self*) → str

Returns the name of the stimulus set used for the session. :rtype: str

**get\_stimulus\_presentations** (*self*) → pandas.core.frame.DataFrame

Get stimulus presentation data.

NOTE: Uses timestamps that do not account for monitor delay.

**Returns** pd.DataFrame – Table whose rows are stimulus presentations (i.e. a given image, for a given duration, typically 250 ms) and whose columns are presentation characteristics.

**get\_stimulus\_templates** (*self*) → Dict[str, numpy.ndarray]

Get stimulus templates (movies, scenes) for behavior session.

**Returns**

**Dict[str, np.ndarray]** A dictionary containing the stimulus images presented during the session. Keys are data set names, and values are 3D numpy arrays.

**get\_stimulus\_timestamps** (*self*) → numpy.ndarray  
Get stimulus timestamps (vsyncs) from pkl file.

NOTE: Located with behavior\_session\_id. Does not use the sync\_file which requires ophys\_session\_id.

**Returns**

**np.ndarray** Timestamps associated with stimulus presentations on the monitor that do not account for monitor delay.

**get\_task\_parameters** (*self*) → dict  
Get task parameters from pkl file.

**Returns**

**dict** A dictionary containing parameters used to define the task runtime behavior.

**get\_trials** (*self*) → pandas.core.frame.DataFrame  
Get trials from pkl file

**Returns**

**pd.DataFrame** A dataframe containing behavioral trial start/stop times, and trial data

## allensdk.internal.api.behavior\_lims\_api module

```
class allensdk.internal.api.behavior_lims_api.BehaviorLimsApi (behavior_experiment_id:
                                                                int,
                                                                lims_credentials:
                                                                Optional[allensdk.core.authentication.DbCredentials] = None)

Bases: object

static behavior_session_id_to_foraging_id (behavior_session_id)
    maps behavior_session_id to foraging_id

static foraging_id_to_behavior_session_id (foraging_id)
    maps foraging_id to behavior_session_id

classmethod from_foraging_id (foraging_id: str, lims_credentials:
                                Union[allensdk.core.authentication.DbCredentials, None-
                                Type] = None)

get_behavior_experiment_id (self)

get_behavior_stimulus_file (self)

get_extended_trials (self)
```

## allensdk.internal.api.behavior\_ophys\_api module

```
class allensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi (ophys_experiment_id:
                                                                int,
                                                                lims_credentials:
                                                                Optional[allensdk.core.authentication.DbCredentials] = None)

Bases: allensdk.internal.api.ophys_lims_api.OphysLimsApi, allensdk.
        brain_observatory.behavior.behavior_ophys_api.BehaviorOphysApiBase
```



```

classmethod get_api_list_by_container_id(container_id)
get_average_projection(self, image_api=None)
get_behavior_session_uuid(self)
get_behavior_stimulus_file(self)
static get_containers_df(only_passed=True)
get_corrected_fluorescence_traces(self)
get_dff_traces(self)
get_experiment_container_id(self)
get_extended_trials(self)
get_eye_tracking(self, z_threshold: float = 3.0, dilation_frames: int = 2)
get_eye_tracking_filepath(self)
get_licks(self)
get_metadata(self)
get_motion_correction(self)
get_nwb_filepath(self)
static get_ophys_experiment_df()
get_ophys_frame_rate(self)
get_ophys_timestamps(self)
get_rewards(self)
get_running_data_df(self)
get_running_speed(self)
get_stimulus_frame_rate(self)
get_stimulus_presentations(self)
get_stimulus_rebase_function(self)
get_stimulus_templates(self)
get_stimulus_timestamps(self)
get_sync_data(self)
get_sync_licks(self)
get_task_parameters(self)
get_trials(self)

```

### allensdk.internal.api.lims\_api module

```

class allensdk.internal.api.lims_api.LimsApi(lims_credentials: Optional
                                              [allensdk.core.authentication.DbCredentials]
                                              = None)
    Bases: object
    get_behavior_tracking_video_filepath_df(self)

```

```
get_experiment_id(self)
get_eye_tracking_video_filepath_df(self)
```

### **allensdk.internal.api.mtrain\_api module**

```
class allensdk.internal.api.mtrain_api.MtrainApi (api_base='http://mtrain:5000')
    Bases: object
    get_behavior_training_df (self, LabTracks_ID=None)
    get_current_stage (self, LabTracks_ID)
    get_df (self, table_name, get_obj=None, **kwargs)
    get_page (self, table_name, get_obj=None, filters=[], **kwargs)
    get_session (self, behavior_session_uuid=None, behavior_session_id=None)
    get_subjects (self)

class allensdk.internal.api.mtrain_api.MtrainSqlApi (dbname=None, user=None,
                                                    host=None, password=None,
                                                    port=None)
    Bases: object
    get_behavior_training_df (self, LabTracks_ID)
    get_subjects (self)
```

### **allensdk.internal.api.ophys\_lims\_api module**

```
class allensdk.internal.api.ophys_lims_api.OphysLimsApi (ophys_experiment_id:
                                                         int, lims_credentials: Op-
                                                         tional[allensdk.core.authentication.DbCredentials]
                                                         = None)
    Bases: allensdk.core.cache_method_utilities.CachedInstanceMethodMixin
    get_age (self)
    get_average_intensity_projection_image_file (self)
    get_cell_roi_ids (self)
    get_cell_specimen_table (self)
    get_demix_file (self)
    get_dff_file (self)
    get_driver_line (self)
    get_equipment_id (self)
    get_experiment_date (self)
    get_external_specimen_name (self, ophys_experiment_id=None)
    get_field_of_view_shape (self)
    get_foraging_id (self)
    get_full_genotype (self)
```

```

get_imaging_depth(self)
get_max_projection(self, image_api=None)
get_max_projection_file(self)
get_metadata(self)
get_motion_corrected_image_stack_file(self)
get_nwb_filepath(self)
get_objectlist_file(self)
get_ophys_cell_segmentation_run_id(self)
get_ophys_experiment_dir(self)
get_ophys_experiment_id(self)
get_raw_cell_specimen_table_dict(self)
get_raw_dff_data(self)
get_reporter_line(self)
get_rig_name(self)
get_rigid_motion_transform_file(self)
get_segmentation_mask_image(self, image_api=None)
get_segmentation_mask_image_file(self)
get_sex(self)
get_stimulus_name(self)
get_surface_2p_pixel_size_um(self)
get_sync_file(self, ophys_experiment_id=None)
get_targeted_structure(self)
get_workflow_state(self)

```

## Module contents

```

exception allensdk.internal.api.OneOrMoreResultExpectedError
    Bases: RuntimeError

class allensdk.internal.api.PostgresQueryMixin(*, dbname, user, host, password, port)
    Bases: object

    fetchall(self, query, strict=True)
    fetchone(self, query, strict=True)
    get_connection(self)
    get_cursor(self)
    select(self, query)
    select_one(self, query)

allensdk.internal.api.psycpg2_select(query, database, host, port, username, password)

```

## allensdk.internal.brain\_observatory package

### Subpackages

## allensdk.internal.brain\_observatory.resources package

### Module contents

### Submodules

## allensdk.internal.brain\_observatory.annotated\_region\_metrics module

Module for calculating annotated region metrics from ISI data

```
allensdk.internal.brain_observatory.annotated_region_metrics.create_region_mask(image_shape,  
                                                                                  x,  
                                                                                  y,  
                                                                                  width,  
                                                                                  height,  
                                                                                  mask)
```

Create mask for region on retinotopic map

#### Parameters

**image\_shape** [tuple] (height, width) of retinotopic map

**x** [int] x offset of region mask within retinotopic map

**y** [int] y offset of region mask within retinotopic map

**width** [int] width of region mask

**height** [int] height of region mask

**mask** [list] region mask as a list of lists

#### Returns

**numpy.ndarray** Region mask

```
allensdk.internal.brain_observatory.annotated_region_metrics.eccentricity(az,  
                                                                            alt,  
                                                                            az_center,  
                                                                            alt_center)
```

Compute eccentricity

#### Parameters

**az** [numpy.ndarray] Azimuth retinotopic map

**alt** [numpy.ndarray] Altitude retinotopic map

**az\_center** [float] Azimuth value to use as center of eccentricity map

**alt\_center** [float] Altitude value to use as center of eccentricity map

#### Returns

**numpy.ndarray** Eccentricity map

```
allensdk.internal.brain_observatory.annotated_region_metrics.get_metrics(altitude_phase,
                                                                    az-
                                                                    imuth_phase,
                                                                    x=None,
                                                                    y=None,
                                                                    width=None,
                                                                    height=None,
                                                                    mask=None,
                                                                    al-
                                                                    ti-
                                                                    tude_scale=0.322,
                                                                    az-
                                                                    imuth_scale=0.383)
```

Calculate annotated region metrics

```
allensdk.internal.brain_observatory.annotated_region_metrics.retinotopy_metric(mask,
                                                                    isi_map)
```

Compute retinotopic metrics for a responding area

#### Parameters

**mask** [numpy.ndarray] Mask representing the area over which to calculate metrics

**isi\_map** [numpy.ndarray] Retinotopic map

#### Returns

**(float, float, float, float) tuple** min, max, range, bias of retinotopic map over masked region

### allensdk.internal.brain\_observatory.demix\_report module

```
allensdk.internal.brain_observatory.demix_report.background_trace(trace,
                                                                    save_dir,
                                                                    data_set=None)

allensdk.internal.brain_observatory.demix_report.compute_correlations(dm,
                                                                    movie_path,
                                                                    movie_dataset)

allensdk.internal.brain_observatory.demix_report.compute_correlations_without_masks(dm)

allensdk.internal.brain_observatory.demix_report.compute_non_overlap_masks(dm)

allensdk.internal.brain_observatory.demix_report.compute_non_overlap_traces(dm,
                                                                    movie_path,
                                                                    movie_dataset)

allensdk.internal.brain_observatory.demix_report.correlation_report(dm,
                                                                    save_dir,
                                                                    with-
                                                                    out_masks=True)
```

**parameters:** dm: [DeMix object] without\_masks: boolean

```
allensdk.internal.brain_observatory.demix_report.plot_masks(dm,
                                                                    save_dir,
                                                                    movie_file,
                                                                    movie_dataset,
                                                                    window=150,
                                                                    add_background=True)
```

**allensdk.internal.brain\_observatory.demixer module**

```
allensdk.internal.brain_observatory.demixer.demix_time_dep_masks(raw_traces,  
                                                                    stack,  
                                                                    masks)
```

**Parameters**

- **raw\_traces** – extracted traces
- **stack** – movie (same length as traces)
- **masks** – binary roi masks

**Returns** demixed traces

```
allensdk.internal.brain_observatory.demixer.find_negative_baselines(trace)
```

```
allensdk.internal.brain_observatory.demixer.find_negative_transients_threshold(trace,  
                                                                                  win-  
                                                                                  dow=500,  
                                                                                  length=10,  
                                                                                  std_devs=3)
```

```
allensdk.internal.brain_observatory.demixer.find_zero_baselines(traces)
```

```
allensdk.internal.brain_observatory.demixer.identify_valid_masks(mask_array)
```

```
allensdk.internal.brain_observatory.demixer.plot_negative_baselines(raw_traces,  
                                                                    demix_traces,  
                                                                    mask_array,  
                                                                    roi_ids_mask,  
                                                                    plot_dir,  
                                                                    ext='png')
```

```
allensdk.internal.brain_observatory.demixer.plot_negative_transients(raw_traces,  
                                                                    demix_traces,  
                                                                    valid_roi,  
                                                                    mask_array,  
                                                                    roi_ids_mask,  
                                                                    plot_dir,  
                                                                    ext='png')
```

```
allensdk.internal.brain_observatory.demixer.plot_overlap_masks_lengthOne(roi_ind,  
                                                                            masks,  
                                                                            save-  
                                                                            file=None,  
                                                                            weighted=False)
```

```
allensdk.internal.brain_observatory.demixer.plot_traces(raw_trace,    demix_trace,  
                                                         roi_id, roi_ind, save_file)
```

```
allensdk.internal.brain_observatory.demixer.plot_transients(roi_ind,    t_trans,  
                                                             masks,    traces,  
                                                             demix_traces, save-  
                                                             file)
```

```
allensdk.internal.brain_observatory.demixer.rolling_window(trace, window=500)
```

**Parameters**

- **trace** –
- **window** –

## Returns

### allensdk.internal.brain\_observatory.eye\_calibration module

```

class allensdk.internal.brain_observatory.eye_calibration.EyeCalibration (monitor_position=array([
    8.62,
    3.16]),
    mon-
    i-
    tor_rotations=array([0.,
    0.,
    0.]),
    led_position=array([25.8
    -
    6.12,
    3.21]),
    cam-
    era_position=array([13.,
    0.,
    0.]),
    cam-
    era_rotations=array([0.,
    0.,
    0.22863813]),
    eye_radius=0.1682,
    cm_per_pixel=0.0010199

```

Bases: object

Class for performing eye-tracking calibration.

Provides methods for estimating the position of the pupil in 3D space and projecting the gaze onto the monitor in both 3D space and monitor space given the experimental geometry.

#### Parameters

**monitor\_position** [numpy.ndarray] [x,y,z] position of monitor in cm.

**monitor\_rotations** [numpy.ndarray] [x,y,z] rotations of monitor in radians.

**led\_position** [numpy.ndarray] [x,y,z] position of LED in cm.

**camera\_position** [numpy.ndarray] [x,y,z] position of camera in cm.

**camera\_rotations** [numpy.ndarray] [x,y,z] rotations for camera in radians. X and Y must be 0.

**eye\_radius** [float] Radius of the eye in cm.

**cm\_per\_pixel** [float] Pixel size of eye-tracking camera.

**compute\_area** (*self*, *pupil\_parameters*)

Compute the area of the pupil.

Assume the pupil is a circle, and that as it moves off-axis with the camera the observed ellipse major axis remains the diameter of the circle.

#### Parameters

**pupil\_parameters** [numpy.ndarray] [nx5] array of pupil parameters.

#### Returns

**numpy.ndarray** [nx1] array of pupil areas in estimated pixels.

**static cr\_position\_in\_mouse\_eye\_coordinates** (*led\_position, eye\_radius*)

Determine the 3D position of the corneal reflection.

The eye is modeled as a spherical mirror, so the reflection appears to be half the radius of the eye from the origin along the eye-LED axis.

#### Parameters

**led\_position** [numpy.ndarray] [x,y,z] position of the LED in eye coordinates.

**eye\_radius** [float] Radius of the eye in centimeters.

#### Returns

**numpy.ndarray** [x,y,z] location of the corneal reflection in eye coordinates.

**pupil\_position\_in\_mouse\_eye\_coordinates** (*self, pupil\_parameters, cr\_parameters*)

Compute the 3D pupil position in mouse eye coordinates.

#### Parameters

**pupil\_parameters** [numpy.ndarray] Array of pupil parameters for each eye tracking frame.

**cr\_parameters** [numpy.ndarray] Array of corneal reflection parameters for each eye tracking frame.

#### Returns

**numpy.ndarray** Pupil position estimates in eye coordinates.

**pupil\_position\_on\_monitor\_in\_cm** (*self, pupil\_parameters, cr\_parameters*)

Compute the pupil position on the monitor in cm.

#### Parameters

**pupil\_parameters** [numpy.ndarray] Array of pupil parameters for each eye tracking frame.

**cr\_parameters** [numpy.ndarray] Array of corneal reflection parameters for each eye tracking frame.

#### Returns

**numpy.ndarray** Pupil position estimates in eye coordinates.

**pupil\_position\_on\_monitor\_in\_degrees** (*self, pupil\_parameters, cr\_parameters*)

Get pupil position on monitor measured in visual degrees.

#### Parameters

**pupil\_parameters** [numpy.ndarray] Array of pupil parameters for each eye tracking frame.

**cr\_parameters** [numpy.ndarray] Array of corneal reflection parameters for each eye tracking frame.

#### Returns

**numpy.ndarray** Pupil position estimate in visual degrees.

**allensdk.internal.brain\_observatory.eye\_calibration.base\_object\_to\_eye\_rotation\_matrix** (*object*)

Rotation matrix to rotate base object frame to eye coordinates.



By convention, any other object's coordinate frame before rotations is set with positive Z pointing from the object's position back to the origin of the eye coordinate system, with X parallel to the eye X-Y plane.

#### Parameters

**object\_position** [np.ndarray] [x, y, z] position of object in eye coordinates.

#### Returns

**numpy.ndarray** [3x3] rotation matrix.

`allensdk.internal.brain_observatory.eye_calibration.object_norm_eye_coordinates(object_position,  
x_rotation,  
y_rotation,  
z_rotation)`

Get the normal vector for the object plane in eye coordinates.

#### Parameters

**object\_position** [numpy.ndarray] [x, y, z] location of the object in eye coordinates.

**x\_rotation** [float] Rotation about the x-axis in radians.

**y\_rotation** [float] Rotation about the y-axis in radians.

**z\_rotation** [float] Rotation about the z-axis in radians.

#### Returns

**numpy.ndarray** Endpoint of the object plane vector in eye coordinates.

`allensdk.internal.brain_observatory.eye_calibration.object_rotation_matrix(x_rotation,  
y_rotation,  
z_rotation)`

Rotation matrix in object coordinate frame.

The rotation matrix for rotating the object coordinate frame from the initial position. This is done by rotating around x, then around y', then around z'.

#### Parameters

**x\_rotation** [float] Rotation about x axis in radians.

**y\_rotation** [float] Rotation about y axis in radians.

**z\_rotation** [float] Rotation about z axis in radians.

#### Returns

**numpy.ndarray** [3x3] rotation matrix.

`allensdk.internal.brain_observatory.eye_calibration.project_to_plane(plane_normal,  
plane_point,  
points)`

Project from the origin through points onto a plane.

#### Parameters

**plane\_normal** [numpy.ndarray] [x, y, z] normal unit vector to the plane.

**plane\_point** [numpy.ndarray] [x, y, z] point on the plane.

**points** [numpy.ndarray] [nx3] points in space through which to project.

#### Returns

**numpy.ndarray** [nx3] points projected on the plane.

**allensdk.internal.brain\_observatory.fit\_ellipse module**

```
class allensdk.internal.brain_observatory.fit_ellipse.FitEllipse(min_points,  
                                                             max_iter,  
                                                             threshold,  
                                                             num_close)
```

Bases: object

```
choose_inliers (self, candidate_points)
```

```
fit_ellipse (self, inlier_points)
```

```
outlier_cost (self, outlier_points, params)
```

```
ransac_fit (self, candidate_points)
```

```
allensdk.internal.brain_observatory.fit_ellipse.ellipse_angle_of_rotation (a)
```

```
allensdk.internal.brain_observatory.fit_ellipse.ellipse_angle_of_rotation2 (a)
```

```
allensdk.internal.brain_observatory.fit_ellipse.ellipse_axis_length (a)
```

```
allensdk.internal.brain_observatory.fit_ellipse.ellipse_center (a)
```

```
allensdk.internal.brain_observatory.fit_ellipse.fit_ellipse (candidate_points)
```

```
allensdk.internal.brain_observatory.fit_ellipse.rotate_vector (y, x, theta)
```

```
allensdk.internal.brain_observatory.fit_ellipse.test_fit ()
```

**allensdk.internal.brain\_observatory.frame\_stream module**

```
class allensdk.internal.brain_observatory.frame_stream.CvInputStream(movie_path,  
                                                                    num_frames=None,  
                                                                    block_size=1,  
                                                                    cache_frames=False)
```

Bases: object

```
close (self)
```

```
open (self)
```

```
class allensdk.internal.brain_observatory.frame_stream.FfmpegInputStream(movie_path,  
                                                                    frame_shape,  
                                                                    ffm-  
                                                                    peg_bin='ffmpeg',  
                                                                    num_frames=None,  
                                                                    block_size=1,  
                                                                    cache_frames=False,  
                                                                    pro-  
                                                                    cess_frame_cb=None)
```

Bases: `allensdk.internal.brain_observatory.frame_stream.FrameInputStream`

```
close (self)
```

```
create_images (self, output_directory, image_type)
```

```
open (self)
```

```

class allensdk.internal.brain_observatory.frame_stream.FfmpegOutputStream (frame_shape,
                                                                    ffm-
                                                                    peg_bin='ffmpeg',
                                                                    block_size=1)

    Bases: allensdk.internal.brain_observatory.frame_stream.FrameOutputStream

    close (self)

    open (self, movie_path)

class allensdk.internal.brain_observatory.frame_stream.FrameInputStream (movie_path,
                                                                    num_frames=None,
                                                                    block_size=1,
                                                                    cache_frames=False,
                                                                    process_frame_cb=None)

    Bases: object

    close (self)

    create_images (self, output_directory, image_type)

    open (self)

class allensdk.internal.brain_observatory.frame_stream.FrameOutputStream (block_size=1)

    Bases: object

    close (self)

    open (self, movie_path)

    write (self, frame)

class allensdk.internal.brain_observatory.frame_stream.ImageOutputStream (block_size=1)

    Bases: allensdk.internal.brain_observatory.frame_stream.FrameOutputStream

```

## allensdk.internal.brain\_observatory.itracker module

### allensdk.internal.brain\_observatory.itracker\_utils module

```

allensdk.internal.brain_observatory.itracker_utils.default_ray (n)

allensdk.internal.brain_observatory.itracker_utils.eccentricity (a1, a2)

allensdk.internal.brain_observatory.itracker_utils.filter_bad_params (params,
                                                                    frame_width,
                                                                    frame_height)

    Replace positions outside image with nan

allensdk.internal.brain_observatory.itracker_utils.generate_rays (image_array,
                                                                    seed_pixel)

allensdk.internal.brain_observatory.itracker_utils.initial_cr_point (image_array,
                                                                    bbox=None)

    bbox is a tuple of (xmin, xmax, ymin, ymax)

allensdk.internal.brain_observatory.itracker_utils.initial_pupil_point (image_array,
                                                                    bbox=None)

    bbox is a tuple of (xmin, xmax, ymin, ymax)

allensdk.internal.brain_observatory.itracker_utils.medfilt_custom (x,      kernel_size=3)

    This median filter returns 'nan' whenever any value in the kernel width is 'nan' and the median otherwise

```

```
allensdk.internal.brain_observatory.itracker_utils.median_absolute_deviation(a,
```

```
consistency_constant=1.4
```

Calculate the median absolute deviation of a univariate dataset.

#### Parameters

**a** [numpy.ndarray] Sample data.

**consistency\_constant** [float] Constant to make the MAD a consistent estimator of the population standard deviation (1.4826 for a normal distribution).

#### Returns

**float** Median absolute deviation of the data.

```
allensdk.internal.brain_observatory.itracker_utils.post_process_cr(cr_params)
```

This will replace questionable values of the CR x and y position with 'nan'

- 1) threshold ellipse area by 99th percentile area distribution
- 2) median filter using custom median filter
- 3) remove deviations from discontinuous jumps

The 'nan' values likely represent obscured CRs, secondary reflections, merges with the secondary reflection, or visual distortions due to the whisker or deformations of the eye

```
allensdk.internal.brain_observatory.itracker_utils.post_process_pupil(pupil_params)
```

Filter pupil parameters to replace outliers with nan

#### Parameters

**pupil\_params** [numpy.ndarray] (Nx5) array of pupil parameters [x, y, angle, axis1, axis2].

#### Returns

**numpy.ndarray** Pupil parameters with outliers replaced with nan

```
allensdk.internal.brain_observatory.itracker_utils.rotate_ray(ray, theta)
```

```
allensdk.internal.brain_observatory.itracker_utils.sobel_grad(image_array)
```

### allensdk.internal.brain\_observatory.mask\_set module

```
class allensdk.internal.brain_observatory.mask_set.MaskSet(masks)
```

Bases: object

**close** (self, mask\_idx, max\_dist)

**close\_sets** (self, set\_size, max\_dist)

**count**

**detect\_duplicates** (self, overlap\_threshold)

**detect\_unions** (self, set\_size=2, max\_dist=10, threshold=0.7)

**distance** (self, mask\_idx)

**intersection** (self, mask\_idx)

**intersection\_size** (self, mask\_idx)

**mask** (self, mask\_idx)

**mask\_is\_union\_of\_set** (*self*, *mask\_idx*, *set\_idx*, *threshold*)

**overlap\_fraction** (*self*, *idx0*, *idx1*)

**size** (*self*, *mask\_idx*)

**union** (*self*, *mask\_idx*)

**union\_size** (*self*, *mask\_idx*)

`allensdk.internal.brain_observatory.mask_set.bb_dist` (*bbs*)

`allensdk.internal.brain_observatory.mask_set.make_bbs` (*masks*)

## allensdk.internal.brain\_observatory.ophys\_session\_decomposition module

`allensdk.internal.brain_observatory.ophys_session_decomposition.export_frame_to_hdf5` (*raw\_filename*, *data\_hdf5\_filename*, *auxiliary\_hdf5\_filename*, *channel\_description*, *session\_id*, *group\_name*, *compression*, *use\_gzip*, *use\_compression*, *use\_presession\_op*)

Export a frame from raw to hdf5.

Data with the channel\_description *data* is stored in the *data\_hdf5\_filename*, while any other data is stored in the *auxiliary\_hdf5\_filename*

`allensdk.internal.brain_observatory.ophys_session_decomposition.load_frame` (*raw\_filename*, *channel\_description*, *session\_id*, *group\_name*, *compression*, *use\_gzip*, *use\_compression*, *use\_presession\_op*)

Load a frame of a multi-frame raw file.

`allensdk.internal.brain_observatory.ophys_session_decomposition.open_view_on_binary` (*file\_like*, *dtype*, *mode*, *offset*, *set*, *shape*, *order*, *strides*)

Open a view into a memory-mapped binary file.

### Parameters

**file\_like** [{string, file object}] File to open.

**dtype** [numpy.dtype] Numpy dtype to open the memory-mapped array as.

**mode** [string] Mode to open the file in.

**offset** [integer] Offset (in bytes) into the file at which to start the memory map.

**shape** [(tuple, list)] Shape of the array.

**order** [{"C", "F"}] C or Fortran ordering.

**strides** [(tuple, list)] Strides along each axis for reading the array.

#### Returns

**numpy.memmap** Strided view into memory-mapped array.

`allensdk.internal.brain_observatory.ophys_session_decomposition.read_strided(filename, dtype, off-set, shape, strides)`

Load a frame without memory-mapping.

### `allensdk.internal.brain_observatory.roi_filter` module

### `allensdk.internal.brain_observatory.roi_filter_utils` module

`allensdk.internal.brain_observatory.roi_filter_utils.CRITERIA()`

**class** `allensdk.internal.brain_observatory.roi_filter_utils.TrainingLabelClassifier(criteria)`  
Bases: `object`

Very basic threshold\_based classifier.

Has a decision function that is just the number of distinct criteria met by the classifier. Criteria are defined as a list of strings used with `pandas.DataFrame.eval`.

#### Parameters

**criteria** [list] List of evaluation strings.

**decision\_function** (*self*, *X*)

Get the distance from the decision boundary.

#### Parameters

**X** [array-like] Features for each ROI.

#### Returns

**T** [array-like] Distance for each sample from the decision boundary.

**class** `allensdk.internal.brain_observatory.roi_filter_utils.TrainingMultiLabelClassifier(criteria)`  
Bases: `object`

Multilabel classifier using groups of `TrainingLabelClassifiers`.

This was used to generate labeling for training the original SVM for classification.

#### Parameters

**criteria** [dictionary] Label names and criteria for each label.

**get\_eXcluded** (*self*, *X*)

Get the calculated value of the eXcluded column.

This is useful for comparison with the original classifier implementation.

#### Parameters

**X** [pandas.DataFrame] Object features from the object list file.

**Returns**

**numpy.ndarray** Calculated eXcluded score from the classifier.

**label\_data** (*self, X, as\_columns=True*)  
Generate labels for each row in X.

**Parameters**

**X** [pandas.DataFrame] Object features from the object list file.

**Returns**

**numpy.ndarray** Array of label codes representing the combination of labels found for each row.

`allensdk.internal.brain_observatory.roi_filter_utils.calculate_max_border` (*motion\_df, max\_shift*)

Calculate motion boundary from frame offsets.

When the motion correction algorithm fails to find sufficient matches, it generates very large frame offsets. The use of *max\_shift* avoids filtering too many cells due to the large offsets, with the tradeoff that those frames will be noise.

**Parameters**

**motion\_df** [pandas.DataFrame] Dataframe containing the x, y offsets from motion correction.

**max\_shift** [float] Maximum shift to allow when considering motion correction. Any larger shifts are considered outliers.

**Returns**

**list** [right\_shift, left\_shift, down\_shift, up\_shift]

`allensdk.internal.brain_observatory.roi_filter_utils.get_indices_by_distance` (*object\_list\_points, mask\_points*)

Find indices of nearest neighbor matches.

Require a distance of 0 (perfect match) and a unique match between masks and object\_list entries.

`allensdk.internal.brain_observatory.roi_filter_utils.get_rois` (*segmentation\_stack, border=None*)

Extract a list of rois from the segmentation data array.

**Parameters**

**segmentation\_stack** [numpy.ndarray] The array from the maxInt\_masks file showing the object masks.

**border** [list] [right\_shift, left\_shift, down\_shift, up\_shift] bounding box determined from motion correction.

**Returns**

**list** List of RoiMask objects.

`allensdk.internal.brain_observatory.roi_filter_utils.order_rois_by_object_list` (*object\_data, rois*)

Reorder rois by matching bounding boxes to object list.

**Parameters**

**object\_data** [pandas.DataFrame] Object list data.

**rois** [list] List of RoiMasks.

**Returns**

**list** The list of rois reordered to index the same as object\_data.

**allensdk.internal.brain\_observatory.run\_itracker module****allensdk.internal.brain\_observatory.time\_sync module**

```
class allensdk.internal.brain_observatory.time_sync.OphysTimeAligner (sync_file,  
                                                                    scan-  
                                                                    ner=None,  
                                                                    dff_file=None,  
                                                                    stimu-  
                                                                    lus_pkl=None,  
                                                                    eye_video=None,  
                                                                    behav-  
                                                                    ior_video=None,  
                                                                    long_stim_threshold=0.2)
```

Bases: object

**behavior\_video\_timestamps**

**corrected\_behavior\_video\_timestamps**

**corrected\_eye\_video\_timestamps**

**corrected\_ophys\_timestamps**

**corrected\_stim\_timestamps**

**dataset**

**eye\_video\_timestamps**

**ophys\_timestamps**

Get the timestamps for the ophys data.

**stim\_timestamps**

```
allensdk.internal.brain_observatory.time_sync.corrected_video_timestamps (video_name,  
                                                                    times-  
                                                                    tamps,  
                                                                    data_length)
```

```
allensdk.internal.brain_observatory.time_sync.get_alignment_array (ref, other,  
                                                                    int_method=<ufunc  
                                                                    'floor'>)
```

Generate an alignment array

```
allensdk.internal.brain_observatory.time_sync.get_keys (sync_dset:          al-  
                                                                    lensdk.brain_observatory.sync_dataset.Dataset)  
                                                         → dict
```

Gets the correct keys for the sync file by searching the sync file line labels. Removes key from the dictionary if it is not in the sync dataset line labels. Args:

sync\_dset: The sync dataset to search for keys within

**Returns:**

**key\_dict:** dictionary of key value pairs for finding data in the sync file



```
allensdk.internal.brain_observatory.time_sync.get_ophys_data_length(filename)
```

```
allensdk.internal.brain_observatory.time_sync.get_photodiode_events(sync_dset,
                                                                    photodiode_key)
```

Returns the photodiode events with the start/stop indicators and the window init flash stripped off. These transitions occur roughly ~1.0s apart, since the sync square changes state every N frames (where N = 60, and frame rate is 60 Hz). Because there are no markers for when the first transition of this type started, we estimate based on the event intervals. For the first valid event, find the first two events that both meet the following criteria:

The next event occurs ~1.0s later

First the last valid event, find the first two events that both meet the following criteria:

The last valid event occurred ~1.0s before

```
allensdk.internal.brain_observatory.time_sync.get_real_photodiode_events(sync_dset,
                                                                    photodiode_key,
                                                                    anomaly_threshold=0.5)
```

Gets the photodiode events with the anomalies removed.

```
allensdk.internal.brain_observatory.time_sync.get_stim_data_length(filename:
                                                                    str) → int
```

Get stimulus data length from .pkl file.

#### Parameters

**filename** [str] Path of stimulus data .pkl file.

#### Returns

**int** Stimulus data length.

```
allensdk.internal.brain_observatory.time_sync.get_video_length(filename)
```

```
allensdk.internal.brain_observatory.time_sync.monitor_delay(sync_dset,
                                                            stim_times, photodiode_key,
                                                            transition_frame_interval=60,
                                                            max_monitor_delay=0.07)
```

Calculate monitor delay.

## Module contents

### allensdk.internal.core package

#### Submodules

#### allensdk.internal.core.lims\_pipeline\_module module

```
class allensdk.internal.core.lims_pipeline_module.PipelineModule(description="",
                                                                    parser=None)
```

Bases: object

**args**

**input\_data** (self)

```
write_output_data (self, data)
```

```
allensdk.internal.core.lims_pipeline_module.default_argument_parser (description="")
```

```
allensdk.internal.core.lims_pipeline_module.run_module (module,          input_data,
                                                         storage_directory,    op-
                                                         tional_args=None,
                                                         python='/shared/utils.x86_64/python-
                                                         2.7/bin/python',
                                                         sdk_path='/shared/bioapps/infoapps/lims2_modules/
                                                         local=False, pbs=None)
```

### **allensdk.internal.core.lims\_utilities module**

```
allensdk.internal.core.lims_utilities.append_well_known_file (wkfs,          path,
                                                                wkf_type_id=None,
                                                                con-
                                                                tent_type=None)
```

```
allensdk.internal.core.lims_utilities.connect (user='limsreader',      host='limsdb2',
                                                  database='lims2',    password='limsro',
                                                  port=5432)
```

```
allensdk.internal.core.lims_utilities.convert_from_titan_linux (file_name)
```

```
allensdk.internal.core.lims_utilities.get_input_json (object_id, object_class, strat-
                                                         egy_class,          host='lims2',
                                                         **kwargs)
```

```
allensdk.internal.core.lims_utilities.get_well_known_file_by_name (wkfs,  file-
                                                                    name)
```

```
allensdk.internal.core.lims_utilities.get_well_known_file_by_type (wkfs,
                                                                    wkf_type_id)
```

```
allensdk.internal.core.lims_utilities.get_well_known_files_by_name (wkfs, file-
                                                                    name)
```

```
allensdk.internal.core.lims_utilities.get_well_known_files_by_type (wkfs,
                                                                    wkf_type_id)
```

```
allensdk.internal.core.lims_utilities.linux_to_windows (file_name)
```

```
allensdk.internal.core.lims_utilities.query (query, user='limsreader', host='limsdb2',
                                                  database='lims2',    password='limsro',
                                                  port=5432)
```

```
allensdk.internal.core.lims_utilities.safe_system_path (file_name)
```

```
allensdk.internal.core.lims_utilities.select (cursor, query)
```

**allensdk.internal.core.mouse\_connectivity\_cache\_prerelease module**

**class** allensdk.internal.core.mouse\_connectivity\_cache\_prerelease.**MouseConnectivityCachePrerelease**

Bases: *allensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache*

Extends MouseConnectivityCache to use prereleased data from lims.

**Parameters**

**resolution: int** Resolution of grid data to be downloaded when accessing projection volume, the annotation volume, and the annotation volume. Must be one of (10, 25, 50, 100). Default is 25.

**ccf\_version: string** Desired version of the Common Coordinate Framework. This affects the annotation volume (*get\_annotation\_volume*) and structure masks (*get\_structure\_mask*). Must be one of (*MouseConnectivityApi.CCF\_2015*, *MouseConnectivityApi.CCF\_2016*). Default: *MouseConnectivityApi.CCF\_2016*

**cache: boolean** Whether the class should save results of API queries to locations specified in the manifest file. Queries for files (as opposed to metadata) must have a file location. If caching is disabled, those locations must be specified in the function call (e.g. *get\_projection\_density(file\_name='file.nrrd')*).

**manifest\_file: string** File name of the manifest to be read. Default is "mouse\_connectivity\_manifest.json".

**Attributes**

**resolution: int** Resolution of grid data to be downloaded when accessing projection volume, the annotation volume, and the annotation volume. Must be one of (10, 25, 50, 100). Default is 25.

**api: MouseConnectivityApiPrerelease instance** Used internally to make API queries.

**EXPERIMENTS\_PRERELEASE\_KEY = 'EXPERIMENTS\_PRERELEASE'**

**STORAGE\_DIRECTORIES\_PRERELEASE\_KEY = 'STORAGE\_DIRECTORIES\_PRERELEASE'**

**add\_manifest\_paths** (*self, manifest\_builder*)

Construct a manifest for this Cache class and save it in a file.

**Parameters**

**file\_name: string** File location to save the manifest.

**filter\_experiments** (*self, experiments, cre=None, injection\_structure\_ids=None, age=None, gender=None, workflow\_state=None, workflows=None, project\_code=None*)

Take a list of experiments and filter them by cre status and injection structure.

**Parameters**

**cre: boolean or list** If True, return only cre-positive experiments. If False, return only cre-negative experiments. If None, return all experiments. If list, return all experiments with cre line names in the supplied list. Default None.

**injection\_structure\_ids: list** Only return experiments that were injected in the structures provided here. If None, return all experiments. Default None.

**age [list]** Only return experiments with specimens with ages provided here. If None, return all experiments. Default None.

**get\_experiments** (*self*, *dataframe=False*, *file\_name=None*, *cre=None*, *injection\_structure\_ids=None*, *age=None*, *gender=None*, *workflow\_state=None*, *workflows=None*, *project\_code=None*)

Read a list of experiments.

If caching is enabled, this will save the whole (unfiltered) list of experiments to a file.

#### Parameters

**dataframe: boolean** Return the list of experiments as a Pandas DataFrame. If False, return a list of dictionaries. Default False.

**file\_name: string** File name to save/read the structures table. If file\_name is None, the file\_name will be pulled out of the manifest. If caching is disabled, no file will be saved. Default is None.

### allensdk.internal.core.simpletree module

**class** allensdk.internal.core.simpletree.**SimpleTree** (*nodes*, *node\_id\_cb*, *parent\_id\_cb*)

Bases: object

**ancestor\_ids** (*self*, *nid*)

**ancestors** (*self*, *nid*)

**child\_ids** (*self*, *nid*)

**children** (*self*, *nid*)

**descendant\_ids** (*self*, *nid*)

**descendants** (*self*, *nid*)

**node** (*self*, *nid*)

**node\_ids** (*self*)

**nodes** (*self*, *nids=None*)

**parent** (*self*, *nid*)

**parent\_id** (*self*, *nid*)

### allensdk.internal.core.swc module

**class** allensdk.internal.core.swc.**Marker** (*\*args*, *\*\*kwargs*)

Bases: dict

Simple dictionary class for handling reconstruction marker objects.

**CUT\_DENDRITE = 10**

**NO\_RECONSTRUCTION = 20**

```
SPACING = [0.1144, 0.1144, 0.28]
```

```
allensdk.internal.core.swc.read_marker_file(file_name)
```

read in a marker file and return a list of dictionaries

```
allensdk.internal.core.swc.read_swc(file_name)
```

Read in an SWC file and return a Morphology object.

#### Parameters

**file\_name:** string SWC file name.

#### Returns

**Morphology** A Morphology instance.

## Module contents

### allensdk.internal.ephys package

#### Submodules

#### allensdk.internal.ephys.core\_feature\_extract module

```
allensdk.internal.ephys.core_feature_extract.extract_data(data, nwb_file)
```

```
allensdk.internal.ephys.core_feature_extract.filter_sweeps(sweeps, types=None,
                                                           passed_only=True,
                                                           iclamp_only=True)
```

```
allensdk.internal.ephys.core_feature_extract.filtered_sweep_numbers(sweeps,
                                                                     types=None,
                                                                     passed_only=True,
                                                                     iclamp_only=True)
```

```
allensdk.internal.ephys.core_feature_extract.find_coarse_long_square_amp_delta(sweeps,
                                                                                dec-
                                                                                i-
                                                                                mals=0)
```

Find the delta between amplitudes of coarse long square sweeps. Includes failed sweeps.

```
allensdk.internal.ephys.core_feature_extract.find_stim_start(stim, idx0=0)
```

Find the index of the first nonzero positive or negative jump in an array.

#### Parameters

**stim:** np.ndarray Array to be searched

**idx0:** int Start searching with this index (default: 0).

#### Returns

**int**

```
allensdk.internal.ephys.core_feature_extract.find_sweep_stim_start(data_set,
                                                                    sweep_number)
```

```
allensdk.internal.ephys.core_feature_extract.generate_output_cell_features(cell_features,
                                                                            sweep_features,
                                                                            sweep_index)
```

```
allensdk.internal.ephys.core_feature_extract.nan_get (obj, key)  
    Return a value from a dictionary. If it does not exist, return None. If it is NaN, return None  
  
allensdk.internal.ephys.core_feature_extract.save_qc_figures (qc_fig_dir,  
                                                             nwb_file,      out-  
                                                             put_data,  
                                                             plot_cell_figures)  
  
allensdk.internal.ephys.core_feature_extract.update_output_sweep_features (cell_features,  
                                                                           sweep_features,  
                                                                           sweep_index)
```

### **allensdk.internal.ephys.plot\_qc\_figures module**

```
allensdk.internal.ephys.plot_qc_figures.exp_curve (x, a, inv_tau, y0)  
    Function used for tau curve fitting  
  
allensdk.internal.ephys.plot_qc_figures.get_features (sweep_features,  
                                                       sweep_number)  
  
allensdk.internal.ephys.plot_qc_figures.get_spikes (sweep_features, sweep_number)  
  
allensdk.internal.ephys.plot_qc_figures.get_time_string ()  
  
allensdk.internal.ephys.plot_qc_figures.load_experiment (file_name,  
                                                         sweep_number)  
  
allensdk.internal.ephys.plot_qc_figures.main ()  
  
allensdk.internal.ephys.plot_qc_figures.make_cell_html (image_files,  
                                                         ephys_roi_result, file_name,  
                                                         relative_sweep_link)  
  
allensdk.internal.ephys.plot_qc_figures.make_cell_page (nwb_file, ephys_roi_result,  
                                                         working_dir,  
                                                         save_cell_plots=True)  
  
allensdk.internal.ephys.plot_qc_figures.make_sweep_html (sweep_files, file_name)  
  
allensdk.internal.ephys.plot_qc_figures.make_sweep_page (nwb_file, ephys_roi_result,  
                                                         working_dir)  
  
allensdk.internal.ephys.plot_qc_figures.mask_nulls (data)  
  
allensdk.internal.ephys.plot_qc_figures.plot_cell_figures (nwb_file,  
                                                             ephys_roi_result,  
                                                             image_dir, sizes)  
  
allensdk.internal.ephys.plot_qc_figures.plot_fi_curve_figures (nwb_file,  
                                                             cell_features,  
                                                             lims_features,  
                                                             sweep_features,  
                                                             image_dir, sizes,  
                                                             cell_image_files)  
  
allensdk.internal.ephys.plot_qc_figures.plot_hero_figures (nwb_file, cell_features,  
                                                             lims_features,  
                                                             sweep_features,  
                                                             image_dir, sizes,  
                                                             cell_image_files)  
  
allensdk.internal.ephys.plot_qc_figures.plot_images (ephys_roi_result, image_dir,  
                                                         sizes, image_sets)
```

```

allensdk.internal.ephys.plot_qc_figures.plot_instantaneous_threshold_thumbnail (nwb_file,
                                                                    sweep_numbers,
                                                                    cell_features,
                                                                    lims_features,
                                                                    sweep_features,
                                                                    color='red')

allensdk.internal.ephys.plot_qc_figures.plot_long_square_summary (nwb_file,
                                                                    cell_features,
                                                                    lims_features,
                                                                    sweep_features)

allensdk.internal.ephys.plot_qc_figures.plot_ramp_figures (nwb_file,
                                                                    cell_specimen,
                                                                    cell_features,
                                                                    lims_features,
                                                                    sweep_features,
                                                                    image_dir,          sizes,
                                                                    cell_image_files)

allensdk.internal.ephys.plot_qc_figures.plot_rheo_figures (nwb_file, cell_features,
                                                                    lims_features,
                                                                    sweep_features,
                                                                    image_dir,          sizes,
                                                                    cell_image_files)

allensdk.internal.ephys.plot_qc_figures.plot_sag_figures (nwb_file, cell_features,
                                                                    lims_features,
                                                                    sweep_features,
                                                                    image_dir,          sizes,
                                                                    cell_image_files)

allensdk.internal.ephys.plot_qc_figures.plot_short_square_figures (nwb_file,
                                                                    cell_features,
                                                                    lims_features,
                                                                    sweep_features,
                                                                    image_dir,
                                                                    sizes,
                                                                    cell_image_files)

allensdk.internal.ephys.plot_qc_figures.plot_single_ap_values (nwb_file,
                                                                    sweep_numbers,
                                                                    lims_features,
                                                                    sweep_features,
                                                                    cell_features,
                                                                    type_name)

allensdk.internal.ephys.plot_qc_figures.plot_subthreshold_long_square_figures (nwb_file,
                                                                    cell_features,
                                                                    lims_features,
                                                                    sweep_features,
                                                                    im-
                                                                    age_dir,
                                                                    sizes,
                                                                    cell_image_files)

allensdk.internal.ephys.plot_qc_figures.plot_sweep_figures (nwb_file,
                                                                    ephys_roi_result,
                                                                    image_dir, sizes)

```

```
allensdk.internal.ephys.plot_qc_figures.plot_sweep_set_summary(nwb_file, high-  
                                                                light_sweep_number,  
                                                                sweep_numbers,  
                                                                high-  
                                                                light_color='#0779BE',  
                                                                back-  
                                                                ground_color='#dddddd')
```

```
allensdk.internal.ephys.plot_qc_figures.plot_sweep_value_figures(cell_specimen,  
                                                                image_dir,  
                                                                sizes,  
                                                                cell_image_files)
```

```
allensdk.internal.ephys.plot_qc_figures.save_figure(fig, image_name, im-  
                                                    age_set_name, image_dir,  
                                                    sizes, image_sets, scalew=1,  
                                                    scaleh=1, ext='jpg')
```

### allensdk.internal.ephys.plot\_qc\_figures3 module

```
allensdk.internal.ephys.plot_qc_figures3.exp_curve(x, a, inv_tau, y0)  
    Function used for tau curve fitting
```

```
allensdk.internal.ephys.plot_qc_figures3.get_features(sweep_features,  
                                                    sweep_number)
```

```
allensdk.internal.ephys.plot_qc_figures3.get_spikes(sweep_features, sweep_number)
```

```
allensdk.internal.ephys.plot_qc_figures3.get_time_string()
```

```
allensdk.internal.ephys.plot_qc_figures3.load_experiment(file_name,  
                                                         sweep_number)
```

```
allensdk.internal.ephys.plot_qc_figures3.make_cell_html(image_files, file_name,  
                                                         relative_sweep_link,  
                                                         specimen_info, fields)
```

```
allensdk.internal.ephys.plot_qc_figures3.make_cell_page(nwb_file, cell_features,  
                                                         rheo_features,  
                                                         sweep_features,  
                                                         sweep_info,  
                                                         well_known_files,  
                                                         specimen_info, work-  
                                                         ing_dir, fields_to_show,  
                                                         save_cell_plots=True)
```

*nwb\_file*: name of nwb file (string)

*cell\_features*:

*rheo\_features*: dict containing extracted features from rheobase sweep

*sweep\_features*:

*sweep\_info*:

*well\_known\_files*: LIMS-output information containing graphics file names

*working\_dir*:

*save\_cell\_plots*:

```
allensdk.internal.ephys.plot_qc_figures3.make_sweep_html(sweep_files, file_name)
```



---

```

allensdk.internal.ephys.plot_qc_figures3.make_sweep_page(nwb_file,    working_dir,
                                                         sweep_data)

allensdk.internal.ephys.plot_qc_figures3.mask_nulls(data)

allensdk.internal.ephys.plot_qc_figures3.plot_cell_figures(nwb_file,
                                                             cell_features,
                                                             sweep_features,
                                                             rheo_features,    im-
                                                             age_dir,    sweep_info,
                                                             sizes)

allensdk.internal.ephys.plot_qc_figures3.plot_fi_curve_figures(nwb_file,
                                                                  cell_features,
                                                                  rheo_features,
                                                                  sweep_features,
                                                                  image_dir,
                                                                  sizes,
                                                                  cell_image_files)

allensdk.internal.ephys.plot_qc_figures3.plot_hero_figures(nwb_file,
                                                             cell_features,
                                                             rheo_features,
                                                             sweep_features,
                                                             image_dir,    sizes,
                                                             cell_image_files)

allensdk.internal.ephys.plot_qc_figures3.plot_images(well_known_files,    image_dir,
                                                         sizes, image_sets)

allensdk.internal.ephys.plot_qc_figures3.plot_instantaneous_threshold_thumbnail(nwb_file,
                                                                                    sweep_numbers,
                                                                                    cell_features,
                                                                                    rheo_features,
                                                                                    sweep_features,
                                                                                    color='red')

allensdk.internal.ephys.plot_qc_figures3.plot_long_square_summary(nwb_file,
                                                                      cell_features,
                                                                      rheo_features,
                                                                      sweep_features)

allensdk.internal.ephys.plot_qc_figures3.plot_ramp_figures(nwb_file,    sweep_info,
                                                             cell_features,
                                                             rheo_features,
                                                             sweep_features,
                                                             image_dir,    sizes,
                                                             cell_image_files)

allensdk.internal.ephys.plot_qc_figures3.plot_rheo_figures(nwb_file,
                                                             cell_features,
                                                             rheo_features,
                                                             sweep_features,
                                                             image_dir,    sizes,
                                                             cell_image_files)

allensdk.internal.ephys.plot_qc_figures3.plot_sag_figures(nwb_file,    cell_features,
                                                             rheo_features,
                                                             sweep_features,
                                                             image_dir,    sizes,
                                                             cell_image_files)

```

```
allensdk.internal.ephys.plot_qc_figures3.plot_short_square_figures(nwb_file,
                                                                    cell_features,
                                                                    rheo_features,
                                                                    sweep_features,
                                                                    im-
                                                                    age_dir,
                                                                    sizes,
                                                                    cell_image_files)

allensdk.internal.ephys.plot_qc_figures3.plot_single_ap_values(nwb_file,
                                                                sweep_numbers,
                                                                rheo_features,
                                                                sweep_features,
                                                                cell_features,
                                                                type_name)

allensdk.internal.ephys.plot_qc_figures3.plot_subthreshold_long_square_figures(nwb_file,
                                                                                cell_features,
                                                                                rheo_features,
                                                                                sweep_features,
                                                                                im-
                                                                                age_dir,
                                                                                sizes,
                                                                                cell_image_files)

allensdk.internal.ephys.plot_qc_figures3.plot_sweep_figures(nwb_file,
                                                            sweep_data,      im-
                                                            age_dir, sizes)

allensdk.internal.ephys.plot_qc_figures3.plot_sweep_set_summary(nwb_file, high-
                                                                light_sweep_number,
                                                                sweep_numbers,
                                                                high-
                                                                light_color='#0779BE',
                                                                back-
                                                                ground_color='#dddddd')

allensdk.internal.ephys.plot_qc_figures3.plot_sweep_value_figures(sweep_info,
                                                                    image_dir,
                                                                    sizes,
                                                                    cell_image_files)

allensdk.internal.ephys.plot_qc_figures3.save_figure(fig,      image_name,      im-
                                                                age_set_name,      image_dir,
                                                                sizes, image_sets, scalew=1,
                                                                scaleh=1, ext='jpg')
```

## Module contents

**allensdk.internal.model package**

## Subpackages

**allensdk.internal.model.biophysical package**

## Subpackages

**`allensdk.internal.model.biophysical.fits` package**

## Subpackages

**`allensdk.internal.model.biophysical.fits.fit_styles` package**

## Module contents

## Module contents

**`allensdk.internal.model.biophysical.passive_fitting` package**

## Subpackages

**`allensdk.internal.model.biophysical.passive_fitting.passive` package**

## Module contents

## Submodules

**`allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit` module**

```
allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit.arg_parser()  
allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit.main()  
allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit.process_inputs(parser)
```

**`allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit2` module**

```
allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit2.main()
```

**`allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit_elec` module**

```
allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit_elec.main()
```

**`allensdk.internal.model.biophysical.passive_fitting.neuron_utils` module**

```
allensdk.internal.model.biophysical.passive_fitting.neuron_utils.get_h()  
allensdk.internal.model.biophysical.passive_fitting.neuron_utils.load_morphology(filename)  
allensdk.internal.model.biophysical.passive_fitting.neuron_utils.parse_neuron_output(output_)  
allensdk.internal.model.biophysical.passive_fitting.neuron_utils.read_neuron_fit_stdout(fun
```

### allensdk.internal.model.biophysical.passive\_fitting.output\_grabber module

**class** allensdk.internal.model.biophysical.passive\_fitting.output\_grabber.**OutputGrabber** (stream, thread)

Bases: object

Class used to grab standard output or another stream.

**escape\_char** = '\x08'

**readOutput** (*self*)

Read the stream data (one byte at a time) and save the text in *capturedtext*.

**start** (*self*)

Start capturing the stream data.

**stop** (*self*)

Stop capturing the stream data and save the text in *capturedtext*.

### allensdk.internal.model.biophysical.passive\_fitting.preprocess module

allensdk.internal.model.biophysical.passive\_fitting.preprocess.**get\_cap\_check\_indices** (*i*)

allensdk.internal.model.biophysical.passive\_fitting.preprocess.**get\_passive\_fit\_data** (*cap\_checks*, *data\_set*)

allensdk.internal.model.biophysical.passive\_fitting.preprocess.**main** ()

## Module contents

### Submodules

### allensdk.internal.model.biophysical.biophysical\_archiver module

**class** allensdk.internal.model.biophysical.biophysical\_archiver.**BiophysicalArchiver** (*archive\_dir*)

Bases: object

**archive\_cell** (*self*, *ephys\_result\_id*, *specimen\_id*, *template*, *neuronal\_model\_id*)

**get\_cells** (*self*)

**get\_neuronal\_models** (*self*, *specimen\_ids*)

**get\_stimulus\_file** (*self*, *neuronal\_model\_id*)

**get\_template\_names** (*self*)

### allensdk.internal.model.biophysical.check\_fi\_shift module

allensdk.internal.model.biophysical.check\_fi\_shift.**calculate\_fi\_curves** (*data\_set*, *sweeps*)

allensdk.internal.model.biophysical.check\_fi\_shift.**estimate\_fi\_shift** (*data\_set*, *sweeps*)

**allensdk.internal.model.biophysical.deap\_utils module**

**class** allensdk.internal.model.biophysical.deap\_utils.**Utils** (*description*)

Bases: *allensdk.model.biophys\_sim.neuron.hoc\_utils.HocUtils*

**actual\_parameters\_from\_normalized** (*self, params*)

**calculate\_feature\_errors** (*self, t\_ms, v, i*)

**generate\_morphology** (*self, morph\_filename*)

**insert\_iclamp** (*self*)

**load\_cell\_parameters** (*self*)

**normalize\_actual\_parameters** (*self, params*)

**record\_values** (*self*)

**set\_actual\_parameters** (*self, params*)

**set\_iclamp\_params** (*self, amp, delay, dur*)

**set\_normalized\_parameters** (*self, params*)

**allensdk.internal.model.biophysical.ephys\_utils module**

allensdk.internal.model.biophysical.ephys\_utils.**get\_step\_stim\_characteristics** (*i,*  
*t*)

allensdk.internal.model.biophysical.ephys\_utils.**get\_sweep\_v\_i\_t\_from\_set** (*data\_set,*  
*sweep\_number*)

allensdk.internal.model.biophysical.ephys\_utils.**get\_sweeps\_of\_type** (*sweep\_type,*  
*sweeps*)

**allensdk.internal.model.biophysical.fit\_stage\_1 module****allensdk.internal.model.biophysical.fit\_stage\_2 module****allensdk.internal.model.biophysical.make\_deap\_fit\_json module**

**class** allensdk.internal.model.biophysical.make\_deap\_fit\_json.**Report** (*top\_level\_description,*  
*fit\_type*)

Bases: object

**best\_fit\_value** (*self*)

**check\_org\_selections\_for\_noise\_block** (*self*)

**gather\_from\_seeds** (*self*)

**generate\_fit\_file** (*self*)

**make\_fit\_json\_file** (*self*)

**setup\_model** (*self*)

**allensdk.internal.model.biophysical.neuron\_parallel module****allensdk.internal.model.biophysical.optimize module****allensdk.internal.model.biophysical.run\_optimize module**

```
class allensdk.internal.model.biophysical.run_optimize.RunOptimize (input_json,  
                                                                out-  
                                                                put_json)
```

Bases: object

**copy\_local** (*self*)

**generate\_manifest\_lims** (*self, lims\_json\_path, manifest\_path*)

**generate\_manifest\_rma** (*self, neuronal\_model\_id, manifest\_path, api\_url=None*)

**info** (*self, lims\_json\_path*)

return a string that a bash script can use to find the working directory, etc. to clean up.

**load\_manifest** (*self*)

**make\_fit** (*self*)

**nrnivmodl** (*self*)

**start\_specimen** (*self*)

```
allensdk.internal.model.biophysical.run_optimize.main (command, input_json, out-  
                                                         put_json)
```

Entry point for module. :param command: select behavior, nrnivmodl or simulate :type command: string  
:param lims\_strategy\_json: path to json file output from lims. :type lims\_strategy\_json: string :param  
lims\_response\_json: path to json file returned to lims. :type lims\_response\_json: string

**allensdk.internal.model.biophysical.run\_optimize\_workflow module****allensdk.internal.model.biophysical.run\_passive\_fit module**

```
allensdk.internal.model.biophysical.run_passive_fit.main (limit, manifest_path)
```

```
allensdk.internal.model.biophysical.run_passive_fit.run_passive_fit (description)
```

**allensdk.internal.model.biophysical.run\_simulate\_lims module**

```
class allensdk.internal.model.biophysical.run_simulate_lims.RunSimulateLims (input_json,  
                                                                out-  
                                                                put_json)
```

Bases: *allensdk.model.biophysical.run\_simulate.RunSimulate*

**copy\_local** (*self*)

**generate\_manifest\_lims** (*self, lims\_data\_path, manifest\_path*)

**generate\_manifest\_rma** (*self, neuronal\_model\_run\_id, manifest\_path, api\_url=None*)

```
allensdk.internal.model.biophysical.run_simulate_lims.main(command,
                                                         lims_strategy_json,
                                                         lims_response_json)
```

Entry point for module. :param command: select behavior, nrnivmodl or simulate :type command: string  
:param lims\_strategy\_json: path to json file output from lims. :type lims\_strategy\_json: string :param  
lims\_response\_json: path to json file returned to lims. :type lims\_response\_json: string

## allensdk.internal.model.biophysical.run\_simulate\_workflow module

### Module contents

### allensdk.internal.model.glif package

#### Submodules

#### allensdk.internal.model.glif.ASGLM module

#### allensdk.internal.model.glif.MLIN module

```
allensdk.internal.model.glif.MLIN.MLIN(voltage, current, res, cap, dt, MAKE_PLOT=False,
                                         SHOW_PLOT=False, BLOCK=False, PUBLICATION_PLOT=False)
```

voltage, current input:

voltage: numpy array of voltage with test pulse cut out  
current: numpy array of stimulus with test pulse cut out

```
allensdk.internal.model.glif.MLIN.autocorr(x)
allensdk.internal.model.glif.MLIN.exp_decay(time, amp, tau)
allensdk.internal.model.glif.MLIN.expsymm_cdf(v, dv)
allensdk.internal.model.glif.MLIN.expsymm_pdf(v, dv)
allensdk.internal.model.glif.MLIN.find_bin_center(edges)
```

#### allensdk.internal.model.glif.are\_two\_lists\_of\_arrays\_the\_same module

```
allensdk.internal.model.glif.are_two_lists_of_arrays_the_same.are_two_lists_of_arrays_the_same(list1, list2)
```

returns False if to lists of arrays are different. otherwise the function returns True.

#### allensdk.internal.model.glif.configure\_model module

#### allensdk.internal.model.glif.error\_functions module

```
allensdk.internal.model.glif.error_functions.MLIN_list_error(param_guess,
                                                             experiment, input_data)
```

**allensdk.internal.model.glif.find\_spikes module**

```
allensdk.internal.model.glif.find_spikes.align_and_cut_spikes(voltage_list,  
                                                             current_list, dt,  
                                                             spike_window=None)
```

This function aligns the spikes to some criteria and returns a current and voltage trace of of the spike over a time window. Also returns zero crossing, and threshold in reference to the aligned spikes.

```
allensdk.internal.model.glif.find_spikes.find_spikes_list(voltage_list, dt)  
allensdk.internal.model.glif.find_spikes.find_spikes_list_old(voltage_list, dt)  
allensdk.internal.model.glif.find_spikes.find_spikes_old(v, dt)  
allensdk.internal.model.glif.find_spikes.find_spikes_ssq_list(voltage_list,  
                                                             dt, dv_cutoff,  
                                                             thresh_frac)
```

**allensdk.internal.model.glif.find\_sweeps module**

**exception** allensdk.internal.model.glif.find\_sweeps.**MissingSweepException**

Bases: Exception

```
allensdk.internal.model.glif.find_sweeps.find_long_square_sweeps(sweeps)  
allensdk.internal.model.glif.find_sweeps.find_noise_sweeps(sweeps)
```

**Find 1) the noise1 sweeps 2) the noise2 sweeps 4) all noise sweeps**

```
allensdk.internal.model.glif.find_sweeps.find_ramp_sweeps(sweeps)
```

**Find 1) all ramp sweeps**

- 2) all subthreshold ramps
- 3) all superthreshold ramps

```
allensdk.internal.model.glif.find_sweeps.find_ramp_to_rheo_sweeps(sweeps)  
allensdk.internal.model.glif.find_sweeps.find_ranked_sweep(sweep_list, key, re-  
                                                             verse=False)  
allensdk.internal.model.glif.find_sweeps.find_short_square_sweeps(sweeps)
```

**Find 1) all of the subthreshold short square sweeps**

- 2) all of the superthreshold short square sweeps
- 3) the subthresholds short square sweep with maximum stimulus amplitude

```
allensdk.internal.model.glif.find_sweeps.find_sweeps(sweep_list)  
allensdk.internal.model.glif.find_sweeps.get_sweep_numbers(sweep_list)  
allensdk.internal.model.glif.find_sweeps.get_sweeps_by_name(sweeps,  
                                                             sweep_type)  
allensdk.internal.model.glif.find_sweeps.main()  
allensdk.internal.model.glif.find_sweeps.organize_sweeps_by_name(sweeps,  
                                                                    name)  
allensdk.internal.model.glif.find_sweeps.parse_arguments()
```



**allensdk.internal.model.glif.glif\_experiment module**

```
class allensdk.internal.model.glif.glif_experiment.GlifExperiment (neuron, dt,  

stim_list,  

resp_list,  

spike_time_steps,  

grid_spike_times,  

grid_spike_voltages,  

param_fit_names,  

**kwargs)
```

Bases: object

**neuron\_parameter\_count** (*self*)

**run** (*self, param\_guess*)

This code will run the loaded neuron model in reference to the target neuron spikes. inputs:

**self:** is the instance of the neuron model and parameters alone with the values of the target spikes.

NOTE the values in each array of the self.gridSpikeIndexTarge\_list and the self.interpolated\_spike\_times are in reference to the time start of of the stim in each induvidual array (not the universal time)

param\_guess: array of scalars of the values that will be inserted into the mapping function below.

**returns:**

**voltage\_list:** list of array of voltage values. **NOTE: IF THE MODEL NEURON SPIKES BEFORE THE TARG**  
 NOT BE CALCULATED THEREFORE THE RESULTING VECTOR WILL NOT BE AS  
 LONG AS THE TARGET AND ALSO WILL NOT MAKE SENSE WITH THE STIMULUS  
 UNLESS YOU CUT IT AND OUTPUT IT TOO.

grid\_spike\_times\_list: interpolated\_spike\_time\_list: an array of the actual times of the spikes.  
 NOTE: THESE TIMES ARE CALCULATED BY ADDING THE

TIME OF THE INDIVIDUAL SPIKE TO THE TIME OF THE LAST SPIKE.

**gridISIFromLastTargSpike\_list:** list of arrays of spike times of the model in reference to the last target (biologi  
 spike (not in reference to sweep start)

**interpolatedISIFromLastTargSpike\_list:** list of arrays of spike times of the model in reference to the last target  
 spike (not in reference to sweep start)

voltageOfModelAtGridBioSpike\_list: list of arrays of scalars that contain the voltage of the model  
 neuron when the target or bio neuron spikes. theshOfModelAtGridBioSpike\_list: list of arrays of  
 scalars that contain the threshold of the model neuron when the target or bio neuron spikes.

**run\_base\_model** (*self, param\_guess*)

This code will run the loaded neuron model. inputs:

**self:** is the instance of the neuron model and parameters alone with the values of the target spikes.

NOTE the values in each array of the self.gridSpikeIndexTarge\_list and the self.interpolated\_spike\_times are in reference to the time start of of the stim in each induvidual array (not the universal time)

param\_guess: array of scalars of the values that will be inserted into the mapping function below.

**returns:**

**voltage\_list:** list of array of voltage values. **NOTE: IF THE MODEL NEURON SPIKES BEFORE THE TARGET** NOT BE CALCULATED THEREFORE THE RESULTING VECTOR WILL NOT BE AS LONG AS THE TARGET AND ALSO WILL NOT MAKE SENSE WITH THE STIMULUS UNLESS YOU CUT IT AND OUTPUT IT TOO.

**gridTime\_list:** **interpolatedTime\_list:** an array of the actual times of the spikes. **NOTE: THESE TIMES ARE CALCULATED BY ADDING THE**

TIME OF THE INDIVIDUAL SPIKE TO THE TIME OF THE LAST SPIKE.

**grid\_ISI\_list:** list of arrays of spike times of the model in reference to the last target (biological) spike (not in reference to sweep start)

**interpolated\_ISI\_list:** list of arrays of spike times of the model in reference to the last target (biological) spike (not in reference to sweep start)

**grid\_spike\_voltage\_list:** list of arrays of scalars that contain the voltage of the model neuron when the target or bio neuron spikes. **grid\_spike\_threshold\_list:** list of arrays of scalars that contain the threshold of the model neuron when the target or bio neuron spikes.

**set\_neuron\_parameters** (*self*, *param\_guess*)

Maps the parameter guesses to the coefficients of the model. input:

*param\_guess* is vector of values. It is assumed that the length will be

## allensdk.internal.model.glif.glif\_optimizer module

```
class allensdk.internal.model.glif.glif_optimizer.GlifOptimizer(experiment, dt,  
                                                             outer_iterations,  
                                                             inner_  
                                                             iterations,  
                                                             sigma_outer,  
                                                             sigma_inner,  
                                                             param_fit_names,  
                                                             stim,      xtol,  
                                                             ftol,      internal_  
                                                             iterations,  
                                                             bessel,     er-  
                                                             ror_function=None,  
                                                             er-  
                                                             ror_function_data=None,  
                                                             init_params=None)
```

Bases: object

**evaluate** (*self*, *x*, *dt\_multiplier=100*)

**initiate\_unique\_seed** (*self*, *seed=None*)

**randomize\_parameter\_values** (*self*, *values*, *sigma*)

**run\_many** (*self*, *iteration\_finished\_callback=None*, *seed=None*)

**run\_once** (*self*, *param0*)

@param *param0*: a list of the initial guesses for the optimizer @return: tuple including parameters that optimize function and value - see fmin docs

```

run_once_bound (self, low_bound, high_bound)
    @param low_bound: a scalar initial guess for the optimizer @param high_bound: a scalar high bound for
    the optimizer @return: tuple including parameters that optimize function and value - see fmin docs
to_dict (self)

```

## allensdk.internal.model.glif.glif\_optimizer\_neuron module

**exception** allensdk.internal.model.glif.glif\_optimizer\_neuron.GlifBadInitializationException

Bases: Exception

Exception raised when voltage is above threshold at the beginning of a sweep. i.e. probably caused by the optimizer.

**exception** allensdk.internal.model.glif.glif\_optimizer\_neuron.GlifNeuronException (*message*, *data*)

Bases: Exception

Exception for catching simulation errors and reporting intermediate data.

**class** allensdk.internal.model.glif.glif\_optimizer\_neuron.GlifOptimizerNeuron (*\*args*, *\*\*kwargs*)

Bases: *allensdk.model.glif.glif\_neuron.GlifNeuron*

Contains methods for running the neuron model in a “forced-spike” paradigm used during optimization.

**TYPE** = 'GLIF'

**classmethod** **from\_dict** (*d*)

**classmethod** **from\_dict\_legacy** (*d*)

**run\_until\_biological\_spike** (*self*, *voltage\_t0*, *threshold\_t0*, *AScurrents\_t0*, *stimulus*, *response*, *start\_index*, *after\_end\_index*, *bio\_spike\_time\_steps*)

Run the neuron simulation over a segment of a stimulus given initial conditions for use in the “forced spike” optimization paradigm. [Note: the section of stimulus is meant to be between two biological neuron spikes. Thus the stimulus is during the interspike interval (ISI)]. The model is simulated until either the model spikes or the end of the segment is reached. If the model does not spike, a spike time is extrapolated past the end of the simulation segment.

This function also returns the initial conditions for the subsequent stimulus segment. In the forced spike paradigm there are several ways

### Parameters

**voltage\_t0** [float] the current voltage of the neuron

**threshold\_t0** [float] the current spike threshold level of the neuron

**AScurrents\_t0** [np.ndarray] the current state of the afterspike currents in the neuron

**stimulus** [np.ndarray] the full stimulus array (not just the segment of data being simulated)

**response** [np.ndarray] the full response array (not just the segment of data being simulated)

**start\_index** [int] index of global stimulus at which to start simulation

**after\_end\_index** [int] index of global stimulus *after* the last index to be simulated

**bio\_spike\_time\_steps** [list] time steps of input spikes

**Returns****dict**

**a dictionary containing:** 'voltage': simulated voltage value 'threshold': simulated threshold values 'AScurrent\_matrix': afterspike current values during the simulation 'grid\_model\_spike\_time': model spike time (in units of dt) 'interpolated\_model\_spike\_time': model spike time (in units of dt) interpolated between time steps 'voltage\_t0': reset voltage value to be used in subsequent simulation interval 'threshold\_t0': reset threshold value to be used in subsequent simulation interval 'AScurrents\_t0': reset afterspike current value to be used in subsequent simulation interval 'grid\_bio\_spike\_model\_voltage': model voltage at the time of the input spike 'grid\_bio\_spike\_model\_threshold': model threshold at the time of the input spike

**run\_with\_biological\_spikes** (*self, stimulus, response, bio\_spike\_time\_steps*)

Run the neuron simulation over a stimulus, but do not allow the model to spike on its own. Rather, force the simulation to spike and reset at a given set of spike indices. Dynamics rules are applied between spikes regardless of the simulated voltage and threshold values. Reset rules are applied only at input spike times. This is used during optimization to force the model to follow the spikes of biological data. The model is optimized in this way so that history effects due to spiking can be adequately modeled. For example, every time the model spikes a new set of afterspike currents will be initiated. To ensure that afterspike currents can be optimized, we force them to be initiated at the time of the biological spike.

**Parameters****stimulus** [np.ndarray] vector of scalar current values**responses** [np.ndarray] vector of scalar voltage values**bio\_spike\_time\_steps** [list] spike time step indices**Returns****dict**

**a dictionary containing:** 'voltage': simulated voltage values, 'threshold': simulated threshold values, 'AScurrent\_matrix': afterspike currents during the simulation, 'grid\_model\_spike\_times': spike times of the model aligned to the simulation grid (when it would have spiked), 'interpolated\_model\_spike\_times': spike times of the model linearly interpolated between time steps, 'grid\_ISI': interspike interval between grid model spike times, 'interpolated\_ISI': interspike interval between interpolated model spike times, 'grid\_bio\_spike\_model\_voltage': voltage of the model at biological/input spike times, 'grid\_bio\_spike\_model\_threshold': voltage of the model at biological/input spike times interpolated between time steps

**to\_dict** (*self*)

Convert the neuron to a serializable dictionary.

`allensdk.internal.model.glif.glif_optimizer_neuron.extrapolate_model_spike_from_endpoints (`

---

```
allensdk.internal.model.glif.glif_optimizer_neuron.extrapolate_model_spike_from_endpoints_s
```

```
allensdk.internal.model.glif.glif_optimizer_neuron.extrapolate_spike_time(dt,
                                                                    num_time_steps,
                                                                    thresh-
                                                                    old_t0,
                                                                    thresh-
                                                                    old_t1,
                                                                    volt-
                                                                    age_t0,
                                                                    volt-
                                                                    age_t1)
```

Given two voltage and threshold values and an interval between them, extrapolate a spike time by intersecting lines the thresholds and voltages.

```
allensdk.internal.model.glif.glif_optimizer_neuron.extrapolate_spike_voltage(dt,
                                                                    num_time_steps,
                                                                    thresh-
                                                                    old_t0,
                                                                    thresh-
                                                                    old_t1,
                                                                    volt-
                                                                    age_t0,
                                                                    volt-
                                                                    age_t1)
```

Given two voltage and threshold values and an interval between them, extrapolate a spike time by intersecting lines the thresholds and voltages.

```
allensdk.internal.model.glif.glif_optimizer_neuron.find_first_model_spike(voltage,
                                                                    thresh-
                                                                    old,
                                                                    volt-
                                                                    age_t1,
                                                                    thresh-
                                                                    old_t1,
                                                                    dt)
```

```
allensdk.internal.model.glif.glif_optimizer_neuron.interpolate_spike_voltage(dt,
                                                                    time_step,
                                                                    thresh-
                                                                    old_t0,
                                                                    thresh-
                                                                    old_t1,
                                                                    volt-
                                                                    age_t0,
                                                                    volt-
                                                                    age_t1)
```

Given two voltage and threshold values, the dt between them and the initial time step, interpolate a spike time

within the dt interval by intersecting the two lines.

### **allensdk.internal.model.glif.optimize\_neuron module**

### **allensdk.internal.model.glif.plotting module**

Written by Corinne Teeter 3-31-14

```
allensdk.internal.model.glif.plotting.checkPreprocess (originalStim_list,      pro-  
                                                         cessedStim_list,      orig-  
                                                         inalVoltage_list,      pro-  
                                                         cessedVoltage_list,    config,  
                                                         blockME=False)  
  
allensdk.internal.model.glif.plotting.checkSpikeCutting (originalStim_list,    cut-  
                                                         Stim_list,      originalVolt-  
                                                         age_list,    cutVoltage_list,  
                                                         allindOfNonSpiking_list,  
                                                         config, blockME=False)  
  
allensdk.internal.model.glif.plotting.plotLineRegress1 (slope, intercept, r, xlim)  
allensdk.internal.model.glif.plotting.plotLineRegressRed (slope, intercept, r, xlim)  
allensdk.internal.model.glif.plotting.plotSpikes (voltage_list,    spike_ind_list,    dt,  
                                                         blockME=False, method=False)
```

### **allensdk.internal.model.glif.preprocess\_neuron module**

### **allensdk.internal.model.glif.rc module**

```
allensdk.internal.model.glif.rc.least_squares_RCE1_calc_tested (voltage_list,  
                                                                    current_list, dt)
```

Calculate resistance, capacitance and resting potential by performing least squares on current and voltage.

#### **Parameters**

**voltage\_list:** list of arrays voltage responses for several sweep repeats

**current\_list:** list of arrays current injections for several sweep repeats

**dt:** float time step size in voltage and current traces

#### **Returns**

**r\_list:** list of floats each value corresponds to the resistance of a sweep

**c\_list:** list of floats each value corresponds to the capacitance of a sweep

**el\_list:** list of floats each value corresponds to the resting potential of a sweep

## allensdk.internal.model.glif.spike\_cutting module

allensdk.internal.model.glif.spike\_cutting.**calc\_spike\_cut\_and\_v\_reset\_via\_expvar\_residuals**

**This function calculates where the spike should be cut based on explained variance.** The goal is to find a model where the voltage after a spike maximally explains the voltage before a spike. This will also specify the voltage reset rule inputs:

spike\_determination\_method: string specifying the method used to find threshold  
all\_current\_list: list of current (list of current traces injected into neuron) all\_voltage\_list:  
list of voltages (list of voltage trace)

The change is that if the slope is greater than one or intercept is greater than zero it forces it. Regardless of required force the residuals are used.

allensdk.internal.model.glif.spike\_cutting.**plotLineRegress1**(*slope, intercept, r,*  
*xlim*)

allensdk.internal.model.glif.spike\_cutting.**plotLineRegressRed**(*slope, intercept, r,*  
*xlim*)

## allensdk.internal.model.glif.threshold\_adaptation module

allensdk.internal.model.glif.threshold\_adaptation.**calc\_spike\_component\_of\_threshold\_from\_mv**

Calculate the spike components of the threshold by fitting a decaying exponential function to data to threshold versus time since last spike in the multiblip data. The exponential is forced to decay to the local `th_inf` (calculated as the mean all of the threshold values of the first spikes in each individual triblip stimulus). For each multiblip stimulus in a stimulus set if there is more than one spike the difference in voltages from the first and second spike are plotted versus the separation in time. Note that this algorithm should only be implemented on multiblips sweeps where the neuron spike on the first and second blip. Since there is no easy way to do this, this erroneous data should not be provided to this algorithm (i.e is should be visually checked and eliminated the preprocessor should hold back this data manually for now.)

#TODO: check to see if this is still true. Notes: The standard SDK spike detection algorithm does not work with

the multiblip stimulus due to artifacts when the stimulus turns on and off. Please see the `find_multiblip_spikes` module for more information.

Input:

**multi\_SS: dictionary** contains multiblip information such as current and stimulus

**dt: float** time step in seconds

Returns:

**const\_to\_add\_to\_thresh\_for\_reset: float** amplitude of the exponential fit otherwise known as `a_spike`. Note that this is without any spike cutting

**decay\_const: float** decay constant of exponential. Note the function fit is a negative exponential which will mean this value will either have to be negated when it is used or the functions used will have to have to include the negative.

**thresh\_inf: float**

```
allensdk.internal.model.glif.threshold_adaptation.exp_fit_c(t, a1, k1, const)
allensdk.internal.model.glif.threshold_adaptation.exp_force_c(t_const, a1, k1)
allensdk.internal.model.glif.threshold_adaptation.fit_avoltage_bvoltage(x,
                                                                    v_trace_list,
                                                                    El_list,
                                                                    spike_cut_length,
                                                                    all_spikeInd_list,
                                                                    th_inf,
                                                                    dt,
                                                                    a_spike,
                                                                    b_spike,
                                                                    fake=False)
```

This is a version of `fit_avoltage_bvoltage_debug` that does not require the `th_trace`, `v_component_of_thresh_trace`, and `spike_component_of_thresh_trace` needed for debugging. A test should be run to make sure the same output comes out from this and the debug function

This function returns the squared error for the difference between the ‘known’ voltage component of the threshold obtained from the biological neuron and the voltage component of the threshold of the model obtained with the input parameters (so that the minimum can be searched for via `fmin`). The overall threshold is the sum of threshold infinity the spike component of the threshold and the voltage component of the threshold. Therefore threshold infinity and the spike component of the threshold must be subtracted from the threshold of the neuron in order to isolate the voltage component of the threshold. In the evaluation of the model the actual voltage of the neuron is used so that any errors in the other components of the model will not influence the fits here (for example, if a afterspike current was estimated incorrectly)

Notes: \* The spike component of the threshold is subtracted from the

voltage which means that the voltage component of the threshold should only be added to rules.

- **b\_spike was fit using a negative value in the function therefore the negative is placed in the equation.**
- **values in this function are in ‘real’ voltage as opposed to voltage** relative to resting potential.
- **current injection during the spike is not taken into account. This seems reasonable as the** ion channels are open during this time and injected current may not greatly influence the neuron.

**x: numpy array** `x[0]=a_voltage` input, `x[1]` is `b_voltage_input`, `x[2]` is `th_inf`



**v\_trace\_list: list of numpy arrays** voltage traces (v\_trace, El, and th\_inf must be in the same frame of reference)

**El\_list: list of floats** reversal potential (v\_trace, El, and th\_inf must be in the same frame of reference)

**spike\_cut\_length: int** number of indicies removed after initiation of a spike

**all\_spikeInd\_list: list of numpy arrays** indicies of spike trains

**th\_inf: float** threshold infinity (v\_trace, El, and th\_inf must be in the same frame of reference)

**dt: float** size of time step (SI units)

**a\_spike: float** amplitude of spike component of threshold.

**b\_spike: float** decay constant in spike component of the threshold

**fake: Boolean** if True makes uses the voltage value of spike step-1 because there is not a voltage value at the spike step because it is set to nan in the simulator.

```
allensdk.internal.model.glif.threshold_adaptation.fit_avoltage_bvoltage_th(x,
                                                                           v_trace_list,
                                                                           El_list,
                                                                           spike_cut_length,
                                                                           all_spikeInd_list,
                                                                           dt,
                                                                           a_spike,
                                                                           b_spike,
                                                                           fake=False)
```

This is a version of fit\_avoltage\_bvoltage\_th\_debug that does not require the th\_trace, v\_component\_of\_thresh\_trace, and spike\_component\_of\_thresh\_trace needed for debugging. A test should be run to make sure the same output comes out from this and the debug function

This function returns the squared error for the difference between the ‘known’ voltage component of the threshold obtained from the biological neuron and the voltage component of the threshold of the model obtained with the input parameters (so that the minimum can be searched for via fmin). The overall threshold is the sum of threshold infinity the spike component of the threshold and the voltage component of the threshold. Therefore threshold infinity and the spike component of the threshold must be subtracted from the threshold of the neuron in order to isolate the voltage component of the threshold. In the evaluation of the model the actual voltage of the neuron is used so that any errors in the other components of the model will not influence the fits here (for example, if a afterspike current was estimated incorrectly)

Notes: \* The spike component of the threshold is subtracted from the

voltage which means that the voltage component of the threshold should only be added to rules.

- **b\_spike was fit using a negative value in the function therefore the negative is placed in the equation.**
- **values in this function are in ‘real’ voltage as opposed to voltage** relative to resting potential.
- **current injection during the spike is not taken into account. This seems reasonable as the** ion channels are open during this time and injected current may not greatly influence the neuron.

**x: numpy array** x[0]=a\_voltage input, x[1] is b\_voltage\_input, x[2] is th\_inf

**v\_trace\_list: list of numpy arrays** voltage traces (v\_trace, El, and th\_inf must be in the same frame of reference)

**El\_list: list of floats** reversal potential (v\_trace, El, and th\_inf must be in the same frame of reference)

**spike\_cut\_length: int** number of indicies removed after initiation of a spike

**all\_spikeInd\_list:** list of numpy arrays indices of spike trains

**dt:** float size of time step (SI units)

**a\_spike:** float amplitude of spike component of threshold.

**b\_spike:** float decay constant in spike component of the threshold

**fake:** Boolean if True makes uses the voltage value of spike step-1 because there is not a voltage value at the spike step because it is set to nan in the simulator.

`allensdk.internal.model.glif.threshold_adaptation.get_peaks(voltage, above-Value=0)`

This function was written by Corinne Teeter and calculates the action potential peaks of a voltage equation” inputs

voltage: numpy array of voltages aboveValue: scalar voltage value over which voltage is considered a spike.

**outputs:** peakInd: array of indices of peaks

## Module contents

### Submodules

#### `allensdk.internal.model.AIC` module

`allensdk.internal.model.AIC.AIC(RSS, k, n)`

Computes the Akaike Information Criterion.

RSS-residual sum of squares of the fitting errors. k - number of fitted parameters. n - number of observations.

`allensdk.internal.model.AIC.AICc(RSS, k, n)`

Corrected AIC. formula from Wikipedia.

`allensdk.internal.model.AIC.BIC(RSS, k, n)`

Bayesian information criterion or Schwartz information criterion. Formula from wikipedia.

#### `allensdk.internal.model.GLM` module

`allensdk.internal.model.GLM.create_basis_IPSP(neye, ncos, kpeaks, ks, DTsim, t0, I_stim, nkt, flag_exp, npcut)`

`allensdk.internal.model.GLM.ff(x, c, dc)`

`allensdk.internal.model.GLM.invn1(x)`

`allensdk.internal.model.GLM.makeBasis_StimKernel(kbasprs, nkt)`

`allensdk.internal.model.GLM.makeBasis_StimKernel_exp(kbasprs, nkt)`

`allensdk.internal.model.GLM.makeFitStruct_GLM(dtsim, kbasprs, nkt, flag_exp)`

`allensdk.internal.model.GLM.nlin(x)`

`allensdk.internal.model.GLM.normalizecols(A)`

`allensdk.internal.model.GLM.sameconv(A, B)`

## allensdk.internal.model.data\_access module

`allensdk.internal.model.data_access.load_sweep` (*file\_name*, *sweep\_number*, *desired\_dt=None*, *cut=0*, *bessel=False*)

load a data sweep and do specified data processing. Inputs:

**file\_name:** **string** name of .nwb data file

**sweep\_number:** number specifying the sweep to be loaded

**desired\_dt:** the size of the time step the data should be subsampled to

**cut:** indicie of which to start reporting data (i.e. cut off data before this indicie)

**bessel:** **dictionary** contains parameters 'N' and 'Wn' to implement standard python bessel filtering

### Returns:

**dictionary containing** voltage: array current: array dt: time step of the returned data start\_idx: the index at which the first stimulus starts (excluding the test pulse)

`allensdk.internal.model.data_access.load_sweeps` (*file\_name*, *sweep\_numbers*, *dt=None*, *cut=0*, *bessel=False*)

load sweeps and do specified data processing. Inputs:

**file\_name:** **string** name of .nwb data file

**sweep\_numbers:** sweep numbers to be loaded

**desired\_dt:** the size of the time step the data should be subsampled to

**cut:** indicie of which to start reporting data (i.e. cut off data before this indicie)

**bessel:** **dictionary** contains parameters 'N' and 'Wn' to implement standard python bessel filtering

### Returns:

**dictionary containing** voltage: list of voltage trace arrays current: list of current trace arrays dt: list of time step corresponding to each array of the returned data start\_idx: list of the indicies at which the first stimulus starts (excluding the test pulse) in each returned sweep

`allensdk.internal.model.data_access.subsample_data` (*data*, *method*, *present\_time\_step*, *desired\_time\_step*)

## Module contents

### allensdk.internal.morphology package

#### Submodules

### allensdk.internal.morphology.compartment module

**class** `allensdk.internal.morphology.compartment.Compartment` (*node1*, *node2*)  
Bases: object

**allensdk.internal.morphology.morphology module**

**class** allensdk.internal.morphology.morphology.**Morphology** (*node\_list=None*)

Bases: object

Keep track of the list of nodes in a morphology and provide a few helper methods (soma, tree information, pruning, etc).

**APICAL\_DENDRITE = 4**

**AXON = 2**

**BASAL\_DENDRITE = 3**

**NODE\_TYPES = [1, 2, 3, 4]**

**SOMA = 1**

**append** (*self, nodes*)

Add additional nodes to this Morphology. Those nodes must originate from another morphology object.

**Parameters**

**nodes:** list of Morphology nodes

**apply\_affine** (*self, aff, scale=None*)

Apply an affine transform to all nodes in this morphology. Compartment radius is adjusted as well.

Format of the affine matrix is:

[x0 y0 z0] [tx] [x1 y1 z1] [ty] [x2 y2 z2] [tz]

where the left 3x3 the matrix defines the affine rotation and scaling, and the right column is the translation vector.

The matrix must be collapsed and stored in a list as follows:

[x0 y0, z0, x1, y1, z1, x2, y2, z2, tx, ty, tz]

**Parameters**

**aff:** 3x4 array of floats (python 2D list, or numpy 2D array) the transformation matrix

**apply\_affine\_only\_rotation** (*self, aff*)

Apply an affine transform to all nodes in this morphology. Only the rotation element of the transform is performed (i.e., although the entire transformation and translation matrix is supplied, only the rotation element is used). The morphology is translated to the point where the soma root is at 0,0,0.

Format of the affine matrix is:

[x0 y0 z0] [tx] [x1 y1 z1] [ty] [x2 y2 z2] [tz]

where the left 3x3 the matrix defines the affine rotation and scaling, and the right column is the translation vector.

The matrix must be collapsed and stored in a list as follows:

[x0 y0, z0, x1, y1, z1, x2, y2, z2, tx, ty, tz]

**Parameters**

**aff:** 3x4 array of floats (python 2D list, or numpy 2D array) the transformation matrix

**change\_parent** (*self, child, parent*)

Change the parent of a node. The child node is adjusted to point to the new parent, the child is taken off of the previous parent's child list, and it is added to the new parent's child list.

**Parameters**

**child: integer or Morphology Object** The ID of the child node, or the child node itself

**parent: integer or Morphology Object** The ID of the parent node, or the parent node itself

**Returns**

**Nothing**

**children\_of** (*self, seg*)

Returns a list of the children of the specified node

**Parameters**

**seg: integer or Morphology Object** The ID of the parent node, or the parent node itself

**Returns**

**A list of the child morphology objects. If the ID of the parent node is invalid, None is returned.**

**clone** (*self*)

Create a clone (deep copy) of this morphology

**compartment** (*self, n*)

Returns the morphology Compartment having the specified ID.

**Parameters**

**n: integer** ID of desired compartment

**Returns**

**A morphology object having the specified ID, or None if such a node doesn't exist**

**compartment\_list**

**convert\_type** (*self, from\_type, to\_type*)

Convert all nodes in morphology from one type to another

**Parameters**

**from\_type: enum** The node type that will be eliminated and replaced. Use one of the following constants: SOMA, AXON, BASAL\_DENDRITE, or APICAL\_DENDRITE

**to\_type: enum** The new type that will replace it. Use one of the following constants: SOMA, AXON, BASAL\_DENDRITE, or APICAL\_DENDRITE

**delete\_tree** (*self, n*)

Delete tree, and all of its nodes, from the morphology.

**Parameters**

**n: Integer** The tree number to delete

**find** (*self, x, y, z, dist, node\_type=None*)

Returns a list of Morphology Objects located within 'dist' of coordinate (x,y,z). If node\_type is specified, the search will be constrained to return only nodes of that type.

**Parameters**

**x, y, z: float** The x,y,z coordinates from which to search around

**dist: float** The search radius

**node\_type: enum (optional)** One of the following constants: SOMA, AXON, BASAL\_DENDRITE or APICAL\_DENDRITE

**Returns**

**A list of all Morphology Objects matching the search criteria**

**get\_dimensions** (*self*)

Returns tuple of overall width, height and depth of morphology. WARNING: if locations of nodes in morphology are manipulated then this value can become incorrect. It can be reset and recalculated by programmitically setting self.dims to None.

**Returns**

**3 real arrays: [width, height, depth], [min\_x, min\_y, min\_z], [max\_x, max\_y, max\_z]**

**node** (*self, n*)

Returns the morphology node having the specified ID.

**Parameters**

**n: integer** ID of desired node

**Returns**

**A morphology node having the specified ID, or None if such a node doesn't exist**

**node\_list**

Return the node list. This is a property to ensure that the node list and node index are in sync.

**node\_list\_by\_type** (*self, node\_type*)

Return an list of all nodes having the specified node type.

**Parameters**

**node\_type: int** Desired node type

**Returns**

**A list of of Morphology Objects**

**num\_nodes**

Return the number of nodes in the morphology.

**num\_trees**

Return the number of trees in the morphology. A tree is defined as everything following from a single root node.

**parent\_of** (*self, seg*)

Returns parent of the specified node.

**Parameters**

**seg: integer or Morphology Object** The ID of the child node, or the child node itself

**Returns**

**A morphology object, or None if no parent exists or if the**

**specified node ID doesn't exist**

**save** (*self*, *file\_name*)

Write this morphology out to an SWC file

#### Parameters

**file\_name: string** desired name of your SWC file

**soma\_root** (*self*)

Returns root node of soma, if present

**sparsify** (*self*, *modulo*)

Return a new Morphology object that has a given number of non-leaf, non-root nodes removed.

#### Parameters

**modulo: int** keep 1 out of every modulo nodes.

#### Returns

**Morphology** A new morphology instance

**strip\_all\_other\_types** (*self*, *node\_type*, *keep\_soma=True*)

Strips everything from the morphology except for the specified type. Parent and child relationships are updated accordingly, creating new roots when necessary.

#### Parameters

**node\_type: enum** The node type to keep in the morphology. Use one of the following constants: SOMA, AXON, BASAL\_DENDRITE, or APICAL\_DENDRITE

**keep\_soma: Boolean (optional)** True (default) if soma nodes should remain in the morphology, and False if the soma should also be stripped

**strip\_type** (*self*, *node\_type*)

Strips all nodes of the specified type from the morphology. Parent and child relationships are updated accordingly, creating new roots when necessary.

#### Parameters

**node\_type: enum** The node type to strip from the morphology. Use one of the following constants: SOMA, AXON, BASAL\_DENDRITE, or APICAL\_DENDRITE

**stumpify\_axon** (*self*, *count=10*)

Remove all axon nodes except the first 'count' nodes, as counted from the connected axon root.

#### Parameters

**count: Integer** The length of the axon 'stump', in number of nodes

**to\_dict** (*self*)

Returns a dictionary of Node objects. These Nodes are a copy of the Morphology. Modifying them will not modify anything in the Morphology itself.

**tree** (*self*, *n*)

Returns a list of all Morphology nodes within the specified tree. A tree is defined as a fully connected graph of nodes. Each tree has exactly one root.

#### Parameters

**n: integer** ID of desired tree

#### Returns

**A list of all morphology objects in the specified tree, or None**

```
        if the tree doesn't exist
    write (self, file_name)
```

### **allensdk.internal.morphology.morphvis module**

```
class allensdk.internal.morphology.morphvis.MorphologyColors
```

```
    Bases: object
```

```
    set_apical_color (self, r, g, b)
```

```
    set_axon_color (self, r, g, b)
```

```
    set_basal_color (self, r, g, b)
```

```
    set_soma_color (self, r, g, b)
```

```
allensdk.internal.morphology.morphvis.calculate_scale (morph,                pix_width,
                                                         pix_height)
```

Calculates scaling factor and x,y insets required to auto-scale and center morphology into box with specified numbers of pixels

#### **Parameters**

**morph:** AISDK Morphology object

**pix\_width:** int

Number of image pixels on X axis

**pix\_height:** int

Number of image pixels on Y axis

#### **Returns**

real, real, real

First return value is the scaling factor. Second is the number of pixels needed to adjust x-coordinates so that the morphology is horizontally centered. Third is the number of pixels needed to adjust the y-coordinates so that the morphology is vertically centered.

```
allensdk.internal.morphology.morphvis.create_image (w, h, color=None, alpha=False)
```

```
allensdk.internal.morphology.morphvis.draw_density_hist (img, morph, vert_scale,
                                                         inset_left=0, inset_right=0, inset_top=0,
                                                         inset_bottom=0, num_bins=None, colors=None)
```

Draws density histogram onto image When no scaling is applied, and no insets are provided, the coordinates of the morphology are used directly – i.e., 100 in morphology coordinates is equal to 100 pixels.

The scale factor is multiplied to morphology coordinates before being drawn. If scale\_factor=2 then 50 in morphology coordinates is 100 pixels. Left and top insets shift the coordinate axes for drawing. E.g., if left=10 and top=5 then 0,0 in morphology coordinates is 10,5 in pixel space. Bottom and right insets are ignored.



If `scale_to_fit` is set then scale factor is ignored. The morphology is scaled to be the maximum size that fits in the image, taking into account insets. In a 100x100 image, if all insets=10, then the image is scaled to fit into the center 80x80 pixel area, and nothing is drawn in the inset border areas.

Axons are drawn before soma and dendrite compartments.

### Parameters

**img:** PIL image object

**morph:** AISDK Morphology object

**vert\_scale:** real

This is the amount required to multiply to a morphology

y-coordinate to convert it to relative cortical depth (on [0,1]).

This is the inverse of the cortical thickness.

**inset\_\*:** real

This is the number of pixels to use as border on top/bottom/

right/left. If `scale_to_fit` is false then only the top/left

values are used, as the `scale_factor` will determine how

large the morphology is (it can be drawn beyond insets and even

beyond image boundaries)

**num\_bins:** int

The number of bins in the histogram

**colors:** MorphologyColors object

This is the color scheme used to draw the morphology. If

`colors=None` then default coloring is used

### Returns

Histogram arrays: [hist, hist2, hist3, hist4]

where hist is the histogram of all neurites, and hist[234] are

the histograms of SWC types 2,3,4

```
allensdk.internal.morphology.morphvis.draw_morphology(img, morph, inset_left=0,
                                                    inset_right=0, inset_top=0, inset_bottom=0,
                                                    scale_to_fit=False,
                                                    scale_factor=1.0, colors=None)
```

Draws morphology onto image When no scaling is applied, and no insets are provided, the coordinates of the morphology are used directly – i.e., 100 in morphology coordinates is equal to 100 pixels.

The scale factor is multiplied to morphology coordinates before being drawn. If `scale_factor=2` then 50 in morphology coordinates is 100 pixels. Left and top insets shift the coordinate axes for drawing. E.g., if `left=10` and `top=5` then 0,0 in morphology coordinates is 10,5 in pixel space. Bottom and right insets are ignored.

If `scale_to_fit` is set then scale factor is ignored. The morphology is scaled to be the maximum size that fits in the image, taking into account insets. In a 100x100 image, if all insets=10, then the image is scaled to fit into the center 80x80 pixel area, and nothing is drawn in the inset border areas.

Axons are drawn before soma and dendrite compartments.

**Parameters**

**img:** PIL image object

**morph:** AISDK Morphology object

**inset\_\*:** real

This is the number of pixels to use as border on top/bottom/right/left. If `scale_to_fit` is false then only the top/left values are used, as the `scale_factor` will determine how large the morphology is (it can be drawn beyond insets and even beyond image boundaries)

**scale\_to\_fit:** boolean

If true then morphology is scaled to the inset area of the image and `scale_factor` is ignored. Morphology is centered in the image in the sense that the top/bottom and left/right edges of the morphology are equidistant from image borders.

**scale\_factor:** real

A scalar amount that is multiplied to morphology coordinates before drawing

**colors:** MorphologyColors object

This is the color scheme used to draw the morphology. If `colors=None` then default coloring is used

**Returns**

2-dimensional array, the pixel coordinates of the soma root [x,y]

**allensdk.internal.morphology.node module**

```
class allensdk.internal.morphology.node.Node (n, t, x, y, z, r, pn, **kwargs)
```

Bases: object

Represents node in SWC morphology file

```
classmethod from_dict (d)
```

```
short_string (self)
```

create string with node information in succinct, single-line form

```
to_dict (self)
```

Convert the node into a serializable dictionary

```
allensdk.internal.morphology.node.euclidean_distance (node1, node2)
```

```
allensdk.internal.morphology.node.midpoint (node1, node2)
```

## allensdk.internal.morphology.validate\_swc module

**class** allensdk.internal.morphology.validate\_swc.**TestNode** (*n, t, x, y, z, r, pn*)

Bases: object

allensdk.internal.morphology.validate\_swc.**main**()

allensdk.internal.morphology.validate\_swc.**resave\_swc** (*orig\_swc, new\_file*)

Reads SWC file into AllenSDK Morphology object and resaves it. This can fix some problems in an SWC file that may disrupt other software tools reading the file (e.g., NEURON)

### Parameters

**orig\_swc: string** Name of SWC file to read

**new\_file: string** Name of output SWC file

allensdk.internal.morphology.validate\_swc.**validate\_swc** (*swc\_file*)

Tests SWC files for compatibility with AllenSDK

**To be compatible with NEURON, SWC files must have the following properties:**

- 1) a single root node with parent ID '-1'
- 2) sequentially increasing ID numbers
- 3) immediate children of the soma cannot branch

To be compatible with feature analysis, SWC files can only have node types in the range 1-4:

1 = soma 2 = axon 3 = [basal] dendrite 4 = apical dendrite

## Module contents

### allensdk.internal.mouse\_connectivity package

#### Subpackages

#### allensdk.internal.mouse\_connectivity.interval\_unionize package

#### Submodules

#### allensdk.internal.mouse\_connectivity.interval\_unionize.cav\_unionize module

#### allensdk.internal.mouse\_connectivity.interval\_unionize.cav\_unionizer module

#### allensdk.internal.mouse\_connectivity.interval\_unionize.data\_utilities module

allensdk.internal.mouse\_connectivity.interval\_unionize.data\_utilities.**get\_cav\_density** (*cav\_de*

allensdk.internal.mouse\_connectivity.interval\_unionize.data\_utilities.**get\_injection\_data** (*in*

Read nrrd files containing injection signal data

```
allensdk.internal.mouse_connectivity.interval_unionize.data_utilities.get_projection_data (p
```

Read nrrd files containing global signal data

```
allensdk.internal.mouse_connectivity.interval_unionize.data_utilities.get_sum_pixel_intens:
```

```
allensdk.internal.mouse_connectivity.interval_unionize.data_utilities.get_sum_pixels (sum_pix
```

```
allensdk.internal.mouse_connectivity.interval_unionize.data_utilities.load_annotation (annota  
data_m
```

Read data files segmenting the reference space into regions of valid and invalid data, then further among brain structures

```
allensdk.internal.mouse_connectivity.interval_unionize.data_utilities.read (path)
```

### **allensdk.internal.mouse\_connectivity.interval\_unionize.interval\_unionizer module**

**class** allensdk.internal.mouse\_connectivity.interval\_unionize.interval\_unionizer.IntervalUn  
Bases: object

**direct\_unionize** (*self*, *data\_arrays*, *pre\_sorted=False*, *\*\*kwargs*)

Obtain unionize records from directly annotated regions.

#### **Parameters**

**data\_arrays** [dict] Keys identify types of data volume. Values are flattened arrays.

**sorted** [bool, optional] If False, data arrays will be sorted.

**extract\_data** (*self*, *data\_arrays*, *low*, *high*, *\*\*kwargs*)

Given flattened data arrays and a specified interval, generate summary data

#### **Parameters**

**data\_arrays** [dict] Keys identify types of data volume. Values are flattened, sorted arrays.

**low** [int] Index at which interval of interest begins. Inclusive.

**high** [int] Index at which interval of interest ends. Exclusive.

**postprocess\_unionizes** (*self*, *raw\_unionizes*, *\*\*kwargs*)

Carry out additional calculations/formatting derivative of core unionization.

#### **Parameters**

**raw\_unionizes** [list of unionizes] Each entry is a unionize record.

**classmethod propagate\_record** (*child\_record*, *ancestor\_record*, *copy\_all=False*)

Updates one unionize corresponding to a rootward structure with information from a unionize corresponding to a leafward structure

#### **Parameters**

**child\_record** [unionize] Data will be drawn from this record

**ancestor\_record** [unionize] This record will be updated

**classmethod propagate\_to\_bilateral** (*lateral\_unionizes*)

**classmethod propagate\_unionizes** (*direct\_unionizes, ancestor\_id\_map*)

Structures are arranged in a tree, whose leafward-oriented edges indicate physical containment. This method updates rootward unionize records with information from leafward ones.

#### Parameters

**direct\_unionizes** [list of unionizes] Each entry is a unionize record produced from a collection of directly labeled voxels in the segmentation volume.

**ancestor\_id\_map** [dict] Keys are structure ids. Values are ids of all structures rootward in

the tree, including the key node

#### Returns

**output\_unionizes** [list of unionizes] Contains completed unionize records at all depths in the structure tree

**classmethod record\_cb** ()

**setup\_interval\_map** (*self, annotation*)

Build a map from structure ids to intervals in the sorted flattened reference space.

#### Parameters

**annotation** [np.ndarray] Segmentation label array.

**sort\_data\_arrays** (*self, data\_arrays*)

Apply the precomputed sort to flattened data arrays

#### Parameters

**data\_arrays** [dict] Keys identify types of data volume. Values are flattened, unsorted arrays.

#### Returns

**dict** : As input, but values are sorted

**allensdk.internal.mouse\_connectivity.interval\_unionize.run\_tissuecyte\_unionize\_cav** module

**allensdk.internal.mouse\_connectivity.interval\_unionize.run\_tissuecyte\_unionize\_classic** module

**allensdk.internal.mouse\_connectivity.interval\_unionize.run\_tissuecyte\_unionize\_classic.get**

**allensdk.internal.mouse\_connectivity.interval\_unionize.run\_tissuecyte\_unionize\_classic.get**

**allensdk.internal.mouse\_connectivity.interval\_unionize.run\_tissuecyte\_unionize\_classic.run**

**allensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_unionize\_record** module

**class** **allensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_unionize\_record.Ti**

Bases: [\*allensdk.internal.mouse\\_connectivity.interval\\_unionize.unionize\\_record.Unionize\*](#)

**direct\_sum\_projection\_pixels**

**max\_voxel\_density**

**max\_voxel\_index**

**output** (*self*, *output\_spacing\_iso*, *volume\_scale*, *target\_shape*, *sort*)  
Generate derived data for this unionize

**Parameters**

**output\_spacing\_iso** [numeric] Isometric spacing of reference space in microns

**volume\_scale** [numeric] Scale factor mapping pixels to microns<sup>3</sup>

**target\_shape** [array-like of numeric] Shape of reference space

**projection\_density**

**projection\_energy**

**projection\_intensity**

**propagate** (*self*, *ancestor*, *copy\_all=False*)  
Update a rootward unionize with data from this unionize record

**Parameters**

**ancestor** [TissuecyteBaseUnionize] will be updated

**Returns**

**ancestor** [TissuecyteBaseUnionize]

**set\_max\_voxel** (*self*, *density\_array*, *low*)  
Find the voxel of greatest density in this unionizes spatial domain

**Parameters**

**density\_array** [ndarray] Float values are densities per voxel

**low** [int] index in full flattened, sorted array of starting voxel

**sum\_pixel\_intensity**

**sum\_pixels**

**sum\_projection\_pixel\_intensity**

**sum\_projection\_pixels**

**class** allensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_unionize\_record.**Ti**

Bases: *allensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_unionize\_record.TissuecyteBaseUnionize*

**calculate** (*self*, *low*, *high*, *data\_arrays*)

**class** allensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_unionize\_record.**Ti**

Bases: *allensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_unionize\_record.TissuecyteBaseUnionize*

**calculate** (*self*, *low*, *high*, *data\_arrays*, *ij\_record*)

## **allensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_unionizer module**

**class** allensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_unionizer.**Tissuecyt**

Bases: *allensdk.internal.mouse\_connectivity.interval\_unionize.interval\_unionizer.IntervalUnionizer*

A specialization of the IntervalUnionizer set up for unionizing Tissuecyte-derived projection data.

**extract\_data** (*self*, *data\_arrays*, *low*, *high*)

As parent

**postprocess\_unionizes** (*self*, *raw\_unionizes*, *image\_series\_id*, *output\_spacing\_iso*, *volume\_scale*, *target\_shape*, *sort*)

As parent

**classmethod propagate\_record** (*child\_record*, *ancestor\_record*, *copy\_all=False*)

As parent

**classmethod record\_cb** ()

## allensdk.internal.mouse\_connectivity.interval\_unionize.unionize\_record module

**class** allensdk.internal.mouse\_connectivity.interval\_unionize.unionize\_record.**Unionize** (*\*args*, *\*\*kwargs*)

Bases: object

Abstract base class for unionize records.

**calculate** (*self*, *\*args*, *\*\*kwargs*)

**output** (*self*, *\*args*, *\*\*kwargs*)

**propagate** (*self*, *ancestor*, *copy\_all*, *\*args*, *\*\*kwargs*)

**slice\_arrays** (*self*, *low*, *high*, *data\_arrays*)

Extract a slice from several aligned arrays

### Parameters

**low** [int] start of slice, inclusive

**high** [int] end of slice, exclusive

**data\_arrays** [dict] keys are varieties of data. values are sorted, flattened data arrays

## Module contents

### allensdk.internal.mouse\_connectivity.projection\_thumbnail package

#### Submodules

### allensdk.internal.mouse\_connectivity.projection\_thumbnail.generate\_projection\_strip module

allensdk.internal.mouse\_connectivity.projection\_thumbnail.generate\_projection\_strip.**apply**

allensdk.internal.mouse\_connectivity.projection\_thumbnail.generate\_projection\_strip.**blend**

allensdk.internal.mouse\_connectivity.projection\_thumbnail.generate\_projection\_strip.**do\_blur**

```
allensdk.internal.mouse_connectivity.projection_thumbnail.generate_projection_strip.handle
```

```
allensdk.internal.mouse_connectivity.projection_thumbnail.generate_projection_strip.max_cb
```

```
allensdk.internal.mouse_connectivity.projection_thumbnail.generate_projection_strip.run (vol  
imi  
ima  
ro-  
ta-  
tion  
col-  
orm
```

```
allensdk.internal.mouse_connectivity.projection_thumbnail.generate_projection_strip.simple
```

### **allensdk.internal.mouse\_connectivity.projection\_thumbnail.image\_sheet module**

```
class allensdk.internal.mouse_connectivity.projection_thumbnail.image_sheet.ImageSheet  
    Bases: object  
    append (self, new_cell)  
    apply (self, fn, *args, **kwargs)  
    static build_from_image (image, n, axis)  
    copy (self)  
    get_output (self, axis)
```

### **allensdk.internal.mouse\_connectivity.projection\_thumbnail.projection\_functions module**

```
allensdk.internal.mouse_connectivity.projection_thumbnail.projection_functions.convert_axis  
allensdk.internal.mouse_connectivity.projection_thumbnail.projection_functions.max_project
```



```
allensdk.internal.mouse_connectivity.projection_thumbnail.projection_functions.template_pro
```

## allensdk.internal.mouse\_connectivity.projection\_thumbnail.visualization\_utilities module

```
allensdk.internal.mouse_connectivity.projection_thumbnail.visualization_utilities.blend(image_stack, weights)
```

### Parameters

**image\_stack :: list of np.ndarray** The images to be blended. Shapes cannot differ

**weight\_stack :: list of np.ndarray** The weight of each image at each pixel. Will be normalized.

```
allensdk.internal.mouse_connectivity.projection_thumbnail.visualization_utilities.convert_colormap
```

Generates a matplotlib continuous colormap on [0, 1] from a discrete colormap at N evenly spaced points.

### Parameters

**data** [list of list] Sublists are [r, g, b].

### Returns

**matplotlib.colors.LinearSegmentedColormap** Gamma is 1. Output space is 3 X [0, 1]

```
allensdk.internal.mouse_connectivity.projection_thumbnail.visualization_utilities.minmax_normalize
```

```
allensdk.internal.mouse_connectivity.projection_thumbnail.visualization_utilities.normalize_color
```

```
allensdk.internal.mouse_connectivity.projection_thumbnail.visualization_utilities.sitk_safe
```

## allensdk.internal.mouse\_connectivity.projection\_thumbnail.volume\_projector module

```
class allensdk.internal.mouse_connectivity.projection_thumbnail.volume_projector.VolumeProjector
    Bases: object
```

```
    build_rotation_transform(self, from_axis, to_axis, angle)
```

```
    extract(self, cb, volume=None)
```

```
    classmethod fixed_factory(volume, size)
```

```
    rotate(self, from_axis, to_axis, angle)
```

```
    rotate_and_extract(self, from_axes, to_axes, angles, cb)
```

```
    classmethod safe_factory(volume)
```

## allensdk.internal.mouse\_connectivity.projection\_thumbnail.volume\_utilities module

```
allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities.sitk_get_center  
allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities.sitk_get_diagonal  
allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities.sitk_get_image_p  
allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities.sitk_get_size_p  
allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities.sitk_paste_into
```

## Module contents

### allensdk.internal.mouse\_connectivity.tissuecyte\_stitching package

#### Submodules

### allensdk.internal.mouse\_connectivity.tissuecyte\_stitching.stitcher module

```
class allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.Stitcher(image_dimensions,  
                                         tiles,  
                                         average,  
                                         image_tiles,  
                                         channels)
```

Bases: object

**run** (*self*, *cb=<built-in function array>*)

**stitch** (*self*, *slice\_image*, *stitched\_indicator*, *tile*, *cb=<built-in function array>*)

```
allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.blend_component_from_pos
```

Obtains a normalized component of the blend, which describes depth of overlap along a specified axis in a specified direction

```
allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.get_blend(indicator_region,  
                                         start,  
                                         cb=<built-in function array>)
```

```
allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.get_blend_component(indicator,  
                                         axis,  
                                         mesh)
```

```
allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.get_indicator_bound_pos
```

Finds the index of first change in a binary mask along a specified axis in a specified direction

```

allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.get_overall_blend(indicator, meshes)
allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.initialize_image(dimension, nchan-
nels,
dtype,
or-
der='C')
allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.initialize_images(dimension, nchan-
nels)
allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.make_blended_tile(blend, tile, cur-
rent_reg

```

## allensdk.internal.mouse\_connectivity.tissuecyte\_stitching.tile module

```

class allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile(index, im-
age,
is_missing,
bounds,
chan-
nel,
size,
mar-
gins,
*args,
**kwargs)

```

Bases: object

```

apply_average_tile(self, average_tile)
apply_average_tile_to_self(self, average_tile)
average_tile_is_untrimmed(self, average_tile)
get_image_region(self)
get_missing_path(self)
initialize_image(self)
trim(self, image)
trim_self(self)

```

## Module contents

## Module contents

allensdk.internal.pipeline\_modules package

## Subpackages

### allensdk.internal.pipeline\_modules.gbm package

## Submodules

### allensdk.internal.pipeline\_modules.gbm.generate\_gbm\_analysis\_run\_records module

```
allensdk.internal.pipeline_modules.gbm.generate_gbm_analysis_run_records.main(analysis_records_  
db_host,  
db_port,  
db_name,  
db_user,  
db_passwd)
```

### allensdk.internal.pipeline\_modules.gbm.generate\_gbm\_heatmap module

```
allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap.create_gene_fpkm_table(analysis_r  
Creates a a matrix (“rows x columns = genes x samples”) of fpkm gene expression values for each particular  
(gene, sample) pair. Rows are sorted by entrez_id and columns are by rna_well_id  
allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap.create_genes_for_transcripts(an  
Creates a list that contains the associated gene for each transcript sorted alphabetically  
allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap.create_sample_metadata(sample_me  
Creates a table of sample metadata sorted by rna_well_id  
allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap.create_transcript_fpkm_table(an  
Creates a a matrix (“rows x columns = transcripts x samples”) of fpkm gene expression values for each particular  
(transcript, sample) pair. Rows are sorted by transcript id and columns are by rna_well_id  
allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap.create_transcripts_for_genes(an  
Creates a list that contains the associated transcript for each gene sorted by entrez_id  
allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap.main()
```

### allensdk.internal.pipeline\_modules.gbm.generate\_gbm\_sample\_metadata module

```
allensdk.internal.pipeline_modules.gbm.generate_gbm_sample_metadata.main(sample_metadata_json_lo  
db_host,  
db_port,  
db_name,  
db_user,  
db_passwd)
```

## Module contents

## Submodules

### allensdk.internal.pipeline\_modules.run\_annotated\_region\_metrics module

Run annotated region metrics calculations

```
allensdk.internal.pipeline_modules.run_annotated_region_metrics.debug(region_id,
                                                                    stor-
                                                                    age_directory='./',
                                                                    lo-
                                                                    cal=True,
                                                                    sdk_path='/data/informatics/C
                                                                    script_path='/data/informatic
                                                                    lims_host='lims2')

allensdk.internal.pipeline_modules.run_annotated_region_metrics.load_arrays(h5_file)

allensdk.internal.pipeline_modules.run_annotated_region_metrics.main()
```

### **allensdk.internal.pipeline\_modules.run\_demixing module**

```
allensdk.internal.pipeline_modules.run_demixing.assert_exists(file_name)

allensdk.internal.pipeline_modules.run_demixing.debug(experiment_id, local=False)

allensdk.internal.pipeline_modules.run_demixing.get_path(obj, key, check_exists)

allensdk.internal.pipeline_modules.run_demixing.main()

allensdk.internal.pipeline_modules.run_demixing.parse_input(data,          ex-
                                                            clude_labels)
```

### **allensdk.internal.pipeline\_modules.run\_dff\_computation module**

```
allensdk.internal.pipeline_modules.run_dff_computation.main()

allensdk.internal.pipeline_modules.run_dff_computation.parse_input(data)
```

### **allensdk.internal.pipeline\_modules.run\_eye\_tracking module**

### **allensdk.internal.pipeline\_modules.run\_neuropil\_correction module**

```
allensdk.internal.pipeline_modules.run_neuropil_correction.adjust_r_for_negativity(r,
                                                                                   F_C,
                                                                                   F_M,
                                                                                   F_N)

allensdk.internal.pipeline_modules.run_neuropil_correction.debug(experiment_id,
                                                                    local=False)

allensdk.internal.pipeline_modules.run_neuropil_correction.debug_plot(file_name,
                                                                    roi_trace,
                                                                    neu-
                                                                    ropil_trace,
                                                                    cor-
                                                                    rected_trace,
                                                                    r,
                                                                    r_vals=None,
                                                                    err_vals=None)

allensdk.internal.pipeline_modules.run_neuropil_correction.main()
```

**allensdk.internal.pipeline\_modules.run\_observatory\_analysis module**

```
allensdk.internal.pipeline_modules.run_observatory_analysis.debug(experiment_ids,
                                                                    lo-
                                                                    cal=False,
                                                                    OUT-
                                                                    PUT_DIR='/data/informatics/CAM/
                                                                    SDK_PATH='/data/informatics/CAM/
                                                                    wall-
                                                                    time='10:00:00',
                                                                    python='/shared/utis.x86_64/python
                                                                    2.7/bin/python',
                                                                    queue='braintv')
allensdk.internal.pipeline_modules.run_observatory_analysis.get_experiment_nwb_file(experiment_ids,
                                                                    nwb_file)
allensdk.internal.pipeline_modules.run_observatory_analysis.get_experiment_session(experiment_ids,
                                                                    session_id)
allensdk.internal.pipeline_modules.run_observatory_analysis.main()
```

**allensdk.internal.pipeline\_modules.run\_observatory\_container\_thumbnails module****allensdk.internal.pipeline\_modules.run\_observatory\_thumbnails module**

```
allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_cell_plots(cell_specimen_id,
                                                                    pre-
                                                                    fix,
                                                                    as-
                                                                    pect,
                                                                    con-
                                                                    figs,
                                                                    out-
                                                                    put_dir,
                                                                    axes=None,
                                                                    trans-
                                                                    par-
                                                                    ent=False)
allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_correlation_plots(data_specimen_id,
                                                                    anal-
                                                                    y-
                                                                    sis_file,
                                                                    con-
                                                                    figs,
                                                                    out-
                                                                    put_dir)
allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_drifting_gratings(dga,
                                                                    con-
                                                                    figs,
                                                                    out-
                                                                    put_dir)
```

```

allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_experiment_thumbnails(n
con-
figs,
out-
put_dir)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_eye_tracking_plots(data
con-
figs,
out-
put_dir)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_locally_sparse_noise(lsna,
con-
figs,
out-
put_dir)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_natural_movie(nma,
con-
figs,
out-
put_dir,
name)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_natural_scenes(nsa,
con-
figs,
out-
put_dir)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_plots(prefix,
as-
pect,
con-
figs,
out-
put_dir,
axes=None,
trans-
parent=False)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_receptive_field(lsna,
con-
figs,
out-
put_dir)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_speed_tuning(analysis,
con-
figs,
out-
put_dir)

```

```
allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_static_gratings(sga,
                                          con-
                                          figs,
                                          out-
                                          put_dir)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.build_type(nwb_file,
                                  data_file,
                                  con-
                                  figs,
                                  out-
                                  put_dir,
                                  type_name)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.debug(experiment_id,
                             plots=None,
                             lo-
                             cal=False)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.get_experiment_analysis_file

allensdk.internal.pipeline_modules.run_observatory_thumbnails.get_experiment_files(experiment_id)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.get_experiment_nwb_file(experiment_id)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.get_input_data(experiment_id)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.lsna_check_hvas(data_set,
                                       data_file)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.main()

allensdk.internal.pipeline_modules.run_observatory_thumbnails.parse_input(data)
```

### **allensdk.internal.pipeline\_modules.run\_ophys\_eye\_calibration module**

```
allensdk.internal.pipeline_modules.run_ophys_eye_calibration.debug(experiment_id,
                             lo-
                             cal=False)

allensdk.internal.pipeline_modules.run_ophys_eye_calibration.get_wkf(wkf_type,
                               experi-
                               ment_id)

allensdk.internal.pipeline_modules.run_ophys_eye_calibration.main()

allensdk.internal.pipeline_modules.run_ophys_eye_calibration.parse_input_data(data)

allensdk.internal.pipeline_modules.run_ophys_eye_calibration.write_output(filename,
                                    po-
                                    si-
                                    tion_degrees,
                                    po-
                                    si-
                                    tion_cm,
                                    ar-
                                    eas)
```



**allensdk.internal.pipeline\_modules.run\_ophys\_session\_decomposition module**

```

allensdk.internal.pipeline_modules.run_ophys_session_decomposition.convert_frame (conversion_de
allensdk.internal.pipeline_modules.run_ophys_session_decomposition.create_fake_metadata (exp
raw
cha
nel
wid
heig
item
size
n_p

allensdk.internal.pipeline_modules.run_ophys_session_decomposition.debug (experiment_id,
lo-
cal=False,
raw_path=None)

allensdk.internal.pipeline_modules.run_ophys_session_decomposition.main ()

allensdk.internal.pipeline_modules.run_ophys_session_decomposition.parse_input (data)
Load all input data from the input json.

```

**allensdk.internal.pipeline\_modules.run\_ophys\_time\_sync module**

```

class allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncOutputs
    Bases: tuple
    Schema for synchronization outputs

    behavior_alignment
        Alias for field number 12

    behavior_delta
        Alias for field number 5

    behavior_times
        Alias for field number 9

    experiment_id
        Alias for field number 0

    eye_alignment
        Alias for field number 11

    eye_delta
        Alias for field number 4

    eye_times
        Alias for field number 8

    ophys_delta
        Alias for field number 2

    ophys_times
        Alias for field number 6

    stimulus_alignment
        Alias for field number 10

```

**stimulus\_delay**  
Alias for field number 1

**stimulus\_delta**  
Alias for field number 3

**stimulus\_times**  
Alias for field number 7

```
class allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncWriter (output_h5_path:
                                                                    str,
                                                                    out-
                                                                    put_json_path:
                                                                    Op-
                                                                    tional[str]
                                                                    =
                                                                    None)
```

Bases: object

**validate\_paths** (*self*)  
Determines whether we can actually write to the specified paths, allowing for creation of intermediate directories. It is a good idea to run this before doing any heavy calculations!

**write** (*self*, *outputs: allensdk.internal.pipeline\_modules.run\_ophys\_time\_sync.TimeSyncOutputs*)  
Convenience for writing both an output h5 and (if applicable) an output json.

**Parameters**

**outputs** [the data to be written]

**write\_output\_h5** (*self*, *outputs*)  
Write (mainly) heavyweight data to an h5 file.

**Parameters**

**outputs** [the data to be written]

**write\_output\_json** (*self*, *outputs*)  
Write lightweight data to a json

**Parameters**

**outputs** [the data to be written]

```
allensdk.internal.pipeline_modules.run_ophys_time_sync.check_stimulus_delay (obt_delay:
                                                                    float,
                                                                    min_delay:
                                                                    float,
                                                                    max_delay:
                                                                    float)
```

Raise an exception if the monitor delay is not within specified bounds

**Parameters**

**obt\_delay** [obtained monitor delay (s)]

**min\_delay** [lower threshold (s)]

**max\_delay** [upper threshold (s)]

```
allensdk.internal.pipeline_modules.run_ophys_time_sync.main()
```

```
allensdk.internal.pipeline_modules.run_ophys_time_sync.run_ophys_time_sync(aligner:
al-
lensdk.internal.brain_o
ex-
per-
i-
ment_id:
int,
min_stimulus_delay:
float,
max_stimulus_delay:
float)
→
al-
lensdk.internal.pipeline
```

Carry out synchronization of timestamps across the data streams of an ophys experiment.

#### Parameters

**aligner** [drives alignment. See OphysTimeAligner for details of the] attributes and properties that must be implemented.

**experiment\_id** [unique identifier for the experiment being aligned]

**min\_stimulus\_delay** [reject alignment run (raise a ValueError) if the] calculated monitor delay is below this value (s).

**max\_stimulus\_delay** [reject alignment run (raise a ValueError) if the] calculated monitor delay is above this value (s).

#### Returns

A **TimeSyncOutputs** (see definition for more information) of output parameters and arrays of aligned timestamps.

**allensdk.internal.pipeline\_modules.run\_roi\_filter** module

**allensdk.internal.pipeline\_modules.run\_tissuecyte\_projection\_thumbnail\_from\_json** module

**allensdk.internal.pipeline\_modules.run\_tissuecyte\_stitching\_classic** module

**allensdk.internal.pipeline\_modules.run\_tissuecyte\_unionize\_cav\_from\_json** module

**allensdk.internal.pipeline\_modules.run\_tissuecyte\_unionize\_classic\_counts\_from\_json** module

**allensdk.internal.pipeline\_modules.run\_tissuecyte\_unionize\_classic\_from\_json** module

**allensdk.internal.pipeline\_modules.run\_tissuecyte\_unionize\_classic\_from\_json.main()**

#### Module contents

## Module contents

### 6.1.7 allensdk.model package

#### Subpackages

#### allensdk.model.biophys\_sim package

#### Subpackages

#### allensdk.model.biophys\_sim.neuron package

#### Submodules

#### allensdk.model.biophys\_sim.neuron.hoc\_utils module

**class** allensdk.model.biophys\_sim.neuron.hoc\_utils.**HocUtils** (*description*)

Bases: object

A helper class for containing references to NEUORN.

#### Attributes

**h** [object] The NEURON hoc object.

**nrn** [object] The NEURON python object.

**neuron** [module] The NEURON module.

**h** = None

**initialize\_hoc** (*self*)

Basic setup for NEURON.

**neuron** = None

**nrn** = None

## Module contents

#### allensdk.model.biophys\_sim.scripts package

#### Module contents

#### Submodules

#### allensdk.model.biophys\_sim.bps\_command module

allensdk.model.biophys\_sim.bps\_command.**choose\_bps\_command** (*command*='bps\_simple',  
conf\_file=None)

allensdk.model.biophys\_sim.bps\_command.**run\_module** (*description*, *module\_name*, *func-*  
*tion\_name*)

## allensdk.model.biophys\_sim.config module

**class** allensdk.model.biophys\_sim.config.**Config**

Bases: *allensdk.config.app.application\_config.ApplicationConfig*

**load** (*self*, *config\_path*, *disable\_existing\_logs=False*)

Parse the application configuration then immediately load the model configuration files.

### Parameters

**disable\_existing\_logs** [boolean, optional] If false (default) leave existing logs after configuration.

**read\_model\_description** (*self*)

parse the model\_file field of the application configuration and read the files.

The model\_file field of the application configuration is first split at commas, since it may list more than one file.

The files may be uris of the form `file:filename?section=name`, in which case a bare configuration object is read from filename into the configuration section with key 'name'.

A simple filename without a section option is treated as a standard multi-section configuration file.

### Returns

**description** [Description] Configuration object.

## Module contents

### allensdk.model.biophysical package

#### Submodules

### allensdk.model.biophysical.run\_simulate module

**class** allensdk.model.biophysical.run\_simulate.**RunSimulate** (*input\_json*, *output\_json*)

Bases: object

**load\_manifest** (*self*)

**nrnivmodl** (*self*)

**simulate** (*self*)

allensdk.model.biophysical.run\_simulate.**main** (*command*, *lims\_strategy\_json*, *lims\_response\_json*)

Entry point for module. :param command: select behavior, nrnivmodl or simulate :type command: string  
:param lims\_strategy\_json: path to json file output from lims. :type lims\_strategy\_json: string :param  
lims\_response\_json: path to json file returned to lims. :type lims\_response\_json: string

### allensdk.model.biophysical.runner module

allensdk.model.biophysical.runner.**load\_description** (*manifest\_json\_path*)

Read configuration file.

### Parameters

**manifest\_json\_path** [string] File containing the experiment configuration.

#### Returns

**Config** Object with all information needed to run the experiment.

`allensdk.model.biophysical.runner.prepare_nwb_output` (*nwb\_stimulus\_path*,  
*nwb\_result\_path*)

Copy the stimulus file, zero out the recorded voltages and spike times.

#### Parameters

**nwb\_stimulus\_path** [string] NWB file name

**nwb\_result\_path** [string] NWB file name

`allensdk.model.biophysical.runner.run` (*description*, *sweeps=None*, *procs=6*)

Main function for simulating sweeps in a biophysical experiment.

#### Parameters

**description** [Config] All information needed to run the experiment.

**procs** [int] number of sweeps to simulate simultaneously.

**sweeps** [list] list of experiment sweep numbers to simulate. If None, simulate all sweeps.

`allensdk.model.biophysical.runner.run_sync` (*description*, *sweeps=None*)

Single-process main function for simulating sweeps in a biophysical experiment.

#### Parameters

**description** [Config] All information needed to run the experiment.

**sweeps** [list] list of experiment sweep numbers to simulate. If None, simulate all sweeps.

`allensdk.model.biophysical.runner.save_nwb` (*output\_path*, *v*, *sweep*, *sweeps\_by\_type*)

Save a single voltage output result into an existing sweep in a NWB file. This is intended to overwrite a recorded trace with a simulated voltage.

#### Parameters

**output\_path** [string] file name of a pre-existing NWB file.

**v** [numpy array] voltage

**sweep** [integer] which entry to overwrite in the file.

## allensdk.model.biophysical.utils module

**class** `allensdk.model.biophysical.utils.AllActiveUtils` (*description*)

Bases: `allensdk.model.biophysical.utils.Utils`

**generate\_morphology** (*self*, *morph\_filename*)

Load a neurolucida or swc-format cell morphology file.

#### Parameters

**morph\_filename** [string] Path to morphology.

**load\_cell\_parameters** (*self*)

Configure a neuron after the cell morphology has been loaded.

**class** allensdk.model.biophysical.utils.**Utils** (*description*)

Bases: *allensdk.model.biophys\_sim.neuron.hoc\_utils.HocUtils*

A helper class for NEURON functionality needed for biophysical simulations.

#### Attributes

**h** [object] The NEURON hoc object.

**nrn** [object] The NEURON python object.

**neuron** [module] The NEURON module.

**generate\_morphology** (*self, morph\_filename*)

Load a swc-format cell morphology file.

#### Parameters

**morph\_filename** [string] Path to swc.

**get\_recorded\_data** (*self, vec*)

Extract recorded voltages and timestamps given the recorded Vector instance. If self.stimulus\_sampling\_rate is smaller than self.simulation\_sampling\_rate, resample to self.stimulus\_sampling\_rate.

#### Parameters

**vec** [neuron.Vector] constructed by self.record\_values

#### Returns

dict with two keys: 'v' = numpy.ndarray with voltages, 't' = numpy.ndarray with timestamps

**load\_cell\_parameters** (*self*)

Configure a neuron after the cell morphology has been loaded.

**static nearest\_neuron\_sampling\_rate** (*hz, target\_hz=40000*)

**read\_stimulus** (*self, stimulus\_path, sweep=0*)

Load current values for a specific experiment sweep and setup simulation and stimulus sampling rates.

NOTE: NEURON only allows simulation timestamps of multiples of 40KHz. To avoid aliasing, we set the simulation sampling rate to the least common multiple of the stimulus sampling rate and 40KHz.

#### Parameters

**stimulus path** [string] NWB file name

**sweep** [integer, optional] sweep index

**record\_values** (*self*)

Set up output voltage recording.

**setup\_iclamp** (*self, stimulus\_path, sweep=0*)

Assign a current waveform as input stimulus.

#### Parameters

**stimulus\_path** [string] NWB file name

**update\_default\_cell\_hoc** (*self, description, default\_cell\_hoc='cell.hoc'*)

replace the default 'cell.hoc' path in the manifest with 'cell.hoc' packaged within AllenSDK if it does not exist

allensdk.model.biophysical.utils.**create\_utils** (*description, model\_type=None*)

Factory method to create a Utils subclass.

**Parameters**

**description** [Config instance] used to initialize Utils subclass

**model\_type** [string] Must be one of [PERISOMATIC\_TYPE, ALL\_ACTIVE\_TYPE]. If none, defaults to PERISOMATIC\_TYPE

**Returns**

Utils instance

**Module contents****allensdk.model.glif package****Submodules****allensdk.model.glif.glif\_neuron module**

**exception** allensdk.model.glif.glif\_neuron.**GlifBadResetException** (*message*, *dv*)

Bases: Exception

Exception raised when voltage is still above threshold after a reset rule is applied.

**class** allensdk.model.glif.glif\_neuron.**GlifNeuron** (*El*, *dt*, *asc\_tau\_array*, *R\_input*, *C*, *asc\_amp\_array*, *spike\_cut\_length*, *th\_inf*, *th\_adapt*, *coeffs*, *AScurrent\_dynamics\_method*, *voltage\_dynamics\_method*, *threshold\_dynamics\_method*, *AScurrent\_reset\_method*, *voltage\_reset\_method*, *threshold\_reset\_method*, *init\_voltage*, *init\_threshold*, *init\_AScurrents*, *\*\*kwargs*)

Bases: object

Implements the current-based Mihalas Neiber GLIF neuron. Simulations model the voltage, threshold, and afterspike currents of a neuron given an input stimulus. A set of modular dynamics rules are applied until voltage crosses threshold, at which point a set of modular reset rules are applied. See `glif_neuron_methods.py` for a list of what options there are for voltage, threshold, and afterspike current dynamics and reset rules.

**Parameters**

**El** [float]

resting potential

**dt** [float] duration between time steps

**asc\_tau\_array**: `np.ndarray` TODO

**R\_input** [float] input resistance

**C** [float] capacitance

**asc\_amp\_arrap** [`np.ndarray`] afterspike current vector. one element per element of `asc_tau_array`.

**spike\_cut\_length** [int] how many time steps to replace with NaNs when a spike occurs.



**th\_inf** [float] instantaneous threshold

**coeffs** [dict] dictionary coefficients premultiplied to neuron properties during simulation. used for optimization.

**AScurrent\_dynamics\_method** [dict] dictionary containing the ‘name’ of the afterspike current dynamics method to use and a ‘params’ dictionary parameters to pass to that function.

**voltage\_dynamics\_method** [dict] dictionary containing the ‘name’ of the voltage dynamics method to use and a ‘params’ dictionary parameters to pass to that function.

**threshold\_dynamics\_method** [dict] dictionary containing the ‘name’ of the threshold dynamics method to use and a ‘params’ dictionary parameters to pass to that function.

**AScurrent\_reset\_method** [dict] dictionary containing the ‘name’ of the afterspike current dynamics method to use and a ‘params’ dictionary parameters to pass to that function.

**voltage\_reset\_method** [dict] dictionary containing the ‘name’ of the voltage dynamics method to use and a ‘params’ dictionary parameters to pass to that function.

**threshold\_reset\_method** [dict] dictionary containing the ‘name’ of the threshold dynamics method to use and a ‘params’ dictionary parameters to pass to that function.

**init\_voltage** [float] initial voltage value

**init\_threshold** [float] initial spike threshold value

**init\_AScurrents** [np.ndarray] initial afterspike current vector. one element per element of `asc_tau_array`.

**TYPE** = 'GLIF'

**append\_threshold\_components** (*self, spike, voltage*)

**static configure\_library\_method** (*method\_type, params*)

Create a GlifNeuronMethod instance out of a library of functions organized by type name. This refers to the `METHOD_LIBRARY` in `glif_neuron_methods.py`, which lays out the available functions that can be used for dynamics and reset rules.

#### Parameters

**method\_type** [string] the name of a function category (e.g. ‘AScurrent\_dynamics\_method’ for the afterspike current dynamics methods)

**params** [dict] a dictionary with two members. ‘name’: the string name of function you want, and ‘params’: parameters you want to pass to that function

#### Returns

**GlifNeuronMethod** a GlifNeuronMethod instance

**static configure\_method** (*method\_name, method, method\_params*)

Create a GlifNeuronMethod instance given a name, a function, and function parameters. This is just a shortcut to the GlifNeuronMethod constructor.

#### Parameters

**method\_name** [string] name for referring to this method later

**method** [function] a python function

**method\_parameters** [dict] function arguments whose values should be fixed

**Returns**

**GlifNeuronMethod** a GlifNeuronMethod instance

**dynamics** (*self*, *voltage\_t0*, *threshold\_t0*, *AScurrents\_t0*, *inj*, *time\_step*, *spike\_time\_steps*)

Update the voltage, threshold, and afterspike currents of the neuron for a single time step.

**Parameters**

**voltage\_t0** [float] the current voltage of the neuron

**threshold\_t0** [float] the current spike threshold level of the neuron

**AScurrents\_t0** [np.ndarray] the current state of the afterspike currents in the neuron

**inj** [float] the current value of the current injection into the neuron

**time\_step** [int] the current time step of the neuron simulation

**spike\_time\_steps** [list] a list of all of the time steps of spikes in the neuron

**Returns**

**tuple** voltage\_t1 (voltage at next time step), threshold\_t1 (threshold at next time step),  
AScurrents\_t1 (afterspike currents at next time step)

**classmethod from\_dict** (*d*)

**reset** (*self*, *voltage\_t0*, *threshold\_t0*, *AScurrents\_t0*)

Apply reset rules to the neuron's voltage, threshold, and afterspike currents assuming a spike has occurred (voltage is above threshold).

**Parameters**

**voltage\_t0** [float] the current voltage of the neuron

**threshold\_t0** [float] the current spike threshold level of the neuron

**AScurrents\_t0** [np.ndarray] the current state of the afterspike currents in the neuron

**Returns**

**tuple** voltage\_t1 (voltage at next time step), threshold\_t1 (threshold at next time step),  
AScurrents\_t1 (afterspike currents at next time step)

**run** (*self*, *stim*)

Run neuron simulation over a given stimulus. This steps through the stimulus applying dynamics equations. After each step it checks if voltage is above threshold. If so, `self.spike_cut_length` NaNs are inserted into the output voltages, reset rules are applied to the voltage, threshold, and afterspike currents, and the simulation resumes.

**Parameters**

**stim** [np.ndarray] vector of scalar current values

**Returns**

**dict**

**a dictionary containing:** 'voltage': simulated voltage values, 'threshold': threshold values during the simulation, 'AScurrents': afterspike current values during the simulation, 'grid\_spike\_times': spike times (in units of `self.dt`) aligned to simulation time steps, 'interpolated\_spike\_times': spike times (in units of `self.dt`) linearly interpolated between time steps, 'spike\_time\_steps': the indices of grid spike times, 'interpolated\_spike\_voltage': voltage of the simulation at interpolated spike times,

`'interpolated_spike_threshold'`: threshold of the simulation at interpolated spike times

**tau\_m**

**to\_dict** (*self*)

Convert the neuron to a serializable dictionary.

`allensdk.model.glif.glif_neuron.interpolate_spike_time` (*dt*, *time\_step*, *threshold\_t0*, *threshold\_t1*, *voltage\_t0*, *voltage\_t1*)

Given two voltage and threshold values, the *dt* between them and the initial time step, interpolate a spike time within the *dt* interval by intersecting the two lines.

`allensdk.model.glif.glif_neuron.interpolate_spike_value` (*dt*, *interpolated\_spike\_time\_offset*, *v0*, *v1*)

Take a value at two adjacent time steps and linearly interpolate what the value would be at an offset between the two time steps.

`allensdk.model.glif.glif_neuron.line_crossing_x` (*dx*, *a0*, *a1*, *b0*, *b1*)

Find the *x* value of the intersection of two lines.

`allensdk.model.glif.glif_neuron.line_crossing_y` (*dx*, *a0*, *a1*, *b0*, *b1*)

Find the *y* value of the intersection of two lines.

## allensdk.model.glif.glif\_neuron\_methods module

The methods in this module are used for configuring dynamics and reset rules for the GlifNeuron. For more details on how to use these methods, see [Generalized LIF Models](#).

**class** `allensdk.model.glif.glif_neuron_methods.GlifNeuronMethod` (*method\_name*, *method*, *method\_params*)

Bases: `object`

A simple class to keep track of the name and parameters associated with a neuron method. This class is initialized with a name, function, and parameters to pass to the function. The function then has those passed parameters fixed to a partial function using `functools.partial`. This class then mimics a function itself using the `__call__` convention. Parameters that are not fixed in this way are assumed to be passed into the method when it is called. If the passed parameters contain an argument that is not part of the function signature, an exception will be raised.

### Parameters

**method\_name** [string] A shorthand name that will be used to reference this method in the *GlifNeuron*.

**method** [function] A python function to be called when this instance is called.

**method\_params** [dict] A dictionary mapping function arguments to values for values that should be fixed.

**modify\_parameter** (*self*, *param*, *operator*)

Modify a function parameter needs to be modified after initialization.

### Parameters

**param** [string] the name of the parameter to modify

**operator** [callable] a function or lambda that returns the desired modified value

**Returns**

**type** the new value of the variable that was just modified.

**to\_dict** (*self*)

```
allensdk.model.glif.glif_neuron_methods.dynamics_AScurrent_exp(neuron, AS-  
currents_t0,  
time_step,  
spike_time_steps)
```

Exponential afterspike current dynamics method takes a current at t0 and returns the current at a time step later.

```
allensdk.model.glif.glif_neuron_methods.dynamics_AScurrent_none(neuron, AS-  
currents_t0,  
time_step,  
spike_time_steps)
```

This method always returns zeros for the afterspike currents, regardless of input.

```
allensdk.model.glif.glif_neuron_methods.dynamics_threshold_inf(neuron, thresh-  
old_t0, volt-  
age_t0, AS-  
currents_t0,  
inj)
```

Set threshold to the neuron's instantaneous threshold.

**Parameters**

**neuron** [class]

**threshold\_t0** [not used here]

**voltage\_t0** [not used here]

**AScurrents\_t0** [not used here]

**inj** [not used here]

**AScurrents\_t0** [not used here]

**inj** [not used here]

```
allensdk.model.glif.glif_neuron_methods.dynamics_threshold_spike_component(neuron,  
thresh-  
old_t0,  
volt-  
age_t0,  
AS-  
cur-  
rents_t0,  
inj,  
a_spike,  
b_spike,  
a_voltage,  
b_voltage)
```

Analytical solution for spike component of threshold. The threshold will adapt via a component initiated by a spike which decays as an exponential. The component is in reference to threshold infinity and are recorded in the neuron's threshold components. The voltage component of the threshold is set to zero in the threshold components because it is zero here The third component refers to th\_inf which is added separately as opposed to being included in the voltage component of the threshold as is done in equation 2.1 of Mihalas and Nieber 2009. Threshold infinity is removed for simple optimization.

**Parameters**

**neuron** [class]  
**threshold\_t0** [float] threshold input to function  
**voltage\_t0** [float] voltage input to function  
**AScurrents\_t0** [vector] values of after spike currents  
**inj** [float] current injected into the neuron

```
allensdk.model.glif.glif_neuron_methods.dynamics_threshold_three_components_exact(neuron,
                                                                                   thresh-
                                                                                   old_t0,
                                                                                   volt-
                                                                                   age_t0,
                                                                                   AS-
                                                                                   cur-
                                                                                   rents_t0,
                                                                                   inj,
                                                                                   a_spike,
                                                                                   b_spike,
                                                                                   a_voltage,
                                                                                   b_voltage)
```

Analytical solution for threshold dynamics. The threshold will adapt via two mechanisms: 1. a voltage dependent adaptation. 2. a component initiated by a spike which decays as an exponential. These two component are in reference to threshold infinity and are recorded in the neuron's threshold components. The third component refers to `th_inf` which is added separately as opposed to being included in the voltage component of the threshold as is done in equation 2.1 of Mihalas and Nieber 2009. Threshold infinity is removed for simple optimization.

#### Parameters

**neuron** [class]  
**threshold\_t0** [float] threshold input to function  
**voltage\_t0** [float] voltage input to function  
**AScurrents\_t0** [vector] values of after spike currents  
**inj** [float] current injected into the neuron

```
allensdk.model.glif.glif_neuron_methods.dynamics_voltage_linear_exact(neuron,
                                                                        volt-
                                                                        age_t0,
                                                                        AS-
                                                                        cur-
                                                                        rents_t0,
                                                                        inj)
```

(TODO) Linear voltage dynamics.

```
allensdk.model.glif.glif_neuron_methods.dynamics_voltage_linear_forward_euler(neuron,
                                                                                volt-
                                                                                age_t0,
                                                                                AS-
                                                                                cur-
                                                                                rents_t0,
                                                                                inj)
```

(TODO) Linear voltage dynamics.

```
allensdk.model.glif.glif_neuron_methods.max_of_line_and_const(x, b, c, d)
```

Find the maximum of a value and a position on a line

**Parameters**

**x: float** x position on line 1  
**c: float** slope of line 1  
**d: float** y-intercept of line 1  
**b: float** y-intercept of line 2

**Returns**

**float** the max of a line value and a constant

`allensdk.model.glif.glif_neuron_methods.min_of_line_and_zero(x, c, d)`

Find the minimum of a value and a position on a line

**Parameters**

**x: float** x position on line 1  
**c: float** slope of line 1  
**d: float** y-intercept of line 1  
**b: float** y-intercept of line 2

**Returns**

**float** the max of a line value and a constant

`allensdk.model.glif.glif_neuron_methods.reset_AScurrent_none(neuron, AScurrents_t0)`

Reset afterspike currents to zero.

`allensdk.model.glif.glif_neuron_methods.reset_AScurrent_sum(neuron, AScurrents_t0, r)`

Reset afterspike currents by adding summed exponentials. Left over currents from last spikes as well as newly initiated currents from current spike. Currents amplitudes in `neuron.asc_amp_array` need to be the amplitudes advanced though the spike cutting. I.e. In the preprocessor if the after spike currents are calculated via the GLM from spike initiation the amplitude at the time after the spike cutting needs to be calculated and `neuron.asc_amp_array` needs to be set to this value.

**Parameters**

**r** [np.ndarray] a coefficient vector applied to the afterspike currents

`allensdk.model.glif.glif_neuron_methods.reset_threshold_inf(neuron, threshold_t0, voltage_v1)`

Reset the threshold to instantaneous threshold.

`allensdk.model.glif.glif_neuron_methods.reset_threshold_three_components(neuron, threshold_t0, voltage_v1, a_spike, b_spike)`

This method calculates the two components of the threshold: a spike (fast) component and a voltage (slow) component. The `threshold_components` vectors are then updated so that the traces match the voltage, current, and total threshold traces. The spike component of the threshold decays via an exponential fit specified by the amplitude `a_spike` and the time constant `b_spike` fit via the multiblip data. The voltage component does not change during the duration of the spike. The spike component are threshold component are summed along with threshold infinity to return the total threshold. Note that in the current implementation `a_spike` is added to the

last value of the `threshold_components` which means that `a_spike` is the amplitude after spike cutting (if there is any).

#### Inputs:

**neuron: class** contains attributes of the neuron

**threshold\_t0, voltage\_t0: float** are not used but are here for consistency with other methods

**a\_spike: float** amplitude of the exponential decay of spike component of threshold after spike cutting has been implemented.

**b\_spike: float** amplitude of the exponential decay of spike component of threshold

#### Outputs:

**Returns: float** the total threshold which is the sum of the spike component of threshold, the voltage component of threshold and threshold infinity (with it's corresponding coefficient)

**neuron.threshold\_components: dictionary containing**

**a\_spike: list** vector of spiking component of threshold that corresponds to the voltage, current, and total threshold traces

**b\_spike: list**

**vector of voltage component of threshold that corresponds to the voltage, current, and total threshold traces.**

Note that this function can be changed to use `a_spike` at the time of the spike and then have the the spike component plus the residual decay thought the spike. There are benefits and drawbacks to this. This potential change would be beneficial as it perhaps makes more biological sense for the threshold to go up at the time of spike if the traces are ever used. Also this would mean that `a_spike` would not have to be adjusted thought the spike cutting after the multiblip fit. However the current implementation makes sense in that it is similar to how afterspike currents are implemented.

```
allensdk.model.glif.glif_neuron_methods.reset_voltage_v_before(neuron, voltage_t0, a, b)
```

Reset voltage to the previous value with a scale and offset applied.

#### Parameters

**a** [float] voltage scale constant

**b** [float] voltage offset constant

```
allensdk.model.glif.glif_neuron_methods.reset_voltage_zero(neuron, voltage_t0)
```

Reset voltage to zero.

```
allensdk.model.glif.glif_neuron_methods.spike_component_of_threshold_exact(th0, b_spike, t)
```

Spike component of threshold modeled as an exponential decay. Implemented here as exact analytical solution.

#### Parameters

**th0** [float] threshold input to function

**b\_spike** [float] decay constant of exponential

**t** [float or array] time step if used in an Euler setup time if used analytically

```
allensdk.model.glif.glif_neuron_methods.spike_component_of_threshold_forward_euler(th_t0, b_spike, dt)
```

Spike component of threshold modeled as an exponential decay. Implemented here for forward Euler

**Parameters**

**th\_t0** [float] threshold input to function  
**b\_spike** [float] decay constant of exponential  
**dt** [float] time step

```
allensdk.model.glif.glif_neuron_methods.voltage_component_of_threshold_exact(th0,
                                                                              v0,
                                                                              I,
                                                                              t,
                                                                              a_voltage,
                                                                              b_voltage,
                                                                              C,
                                                                              g,
                                                                              El)
```

Note this function is the exact formulation; however, dt is used because t0 is the initial time and dt is the time the function is exactly evaluated at. Note: that here, this equation is in reference to th\_inf. Therefore th0 is the total threshold-thr\_inf (threshold\_inf replaced with 0 in the equation to be verbose). This is done so that th\_inf can be optimized without affecting this function.

**Parameters**

**th0** [float] threshold input to function  
**v0** [float] voltage input to function  
**I** [float] total current entering neuron (note if there are after spike currents these must be included in this value)  
**t** [float or array] time step if used in an Euler setup time if used analytically  
**a\_voltage** [float] constant a  
**b\_voltage** [float] constant b  
**C** [float] capacitance  
**g** [float] conductance (1/resistance)  
**El** [float] reversal potential

```
allensdk.model.glif.glif_neuron_methods.voltage_component_of_threshold_forward_euler(th_t0,
                                                                                       v_t0,
                                                                                       dt,
                                                                                       a_voltage,
                                                                                       b_voltage,
                                                                                       El)
```

Equation 2.1 of Mihalas and Nieber, 2009 implemented for use in forward Euler. Note here all variables are in reference to threshold infinity. Therefore thr\_inf is zero here (replaced threshold\_inf with 0 in the equation to be verbose). This is done so that th\_inf can be optimized without affecting this function.

**Parameters**

**th\_t0** [float] threshold input to function  
**v\_t0** [float] voltage input to function  
**dt** [float] time step  
**a\_voltage** [float] constant a  
**b\_voltage** [float] constant b



El [float] reversal potential

## allensdk.model.glif.simulate\_neuron module

```
allensdk.model.glif.simulate_neuron.load_sweep(file_name, sweep_number)
    Load the stimulus for a sweep from file.

allensdk.model.glif.simulate_neuron.main()
allensdk.model.glif.simulate_neuron.parse_arguments()
    Use argparse to get required arguments from the command line

allensdk.model.glif.simulate_neuron.simulate_neuron(neuron, sweep_numbers,
                                                    input_file_name, output_file_name, spike_cut_value)

allensdk.model.glif.simulate_neuron.simulate_sweep(neuron, stimulus, spike_cut_value)
    Simulate a neuron given a stimulus and initial conditions.

allensdk.model.glif.simulate_neuron.simulate_sweep_from_file(neuron, sweep_number,
                                                            input_file_name, output_file_name, spike_cut_value)
    Load a sweep stimulus, simulate the response, and write it out.

allensdk.model.glif.simulate_neuron.write_sweep_response(file_name, sweep_number, response, spike_times)
    Overwrite the response in a file.
```

## Module contents

A Generalized Linear Integrate and Fire (GLIF) neuron modeling package. Use this code to run the GLIF models available in the Allen Cell Types Atlas. See [Generalized LIF Models](#) for more details.

## Module contents

### 6.1.8 allensdk.morphology package

#### Submodules

#### allensdk.morphology.validate\_swc module

```
allensdk.morphology.validate_swc.main()
allensdk.morphology.validate_swc.validate_swc(swc_file)
```

To be compatible with NEURON, SWC files must have the following properties:

- 1) a single root node with parent ID '-1'
- 2) sequentially increasing ID numbers
- 3) immediate children of the soma cannot branch

## Module contents

### 6.1.9 allensdk.mouse\_connectivity package

#### Subpackages

allensdk.mouse\_connectivity.grid package

#### Subpackages

allensdk.mouse\_connectivity.grid.subimage package

#### Submodules

allensdk.mouse\_connectivity.grid.subimage.base\_subimage module

```
class allensdk.mouse_connectivity.grid.subimage.base_subimage.IntensitySubImage(reduce_level,  
                                                                           in_dims,  
                                                                           in_spacing,  
                                                                           coarse_spacing,  
                                                                           in-  
                                                                           ten-  
                                                                           sity_paths,  
                                                                           *args,  
                                                                           **kwargs)
```

Bases: *allensdk.mouse\_connectivity.grid.subimage.base\_subimage.SubImage*

```
get_intensity(self)
```

```
required_intensities = []
```

```
setup_images(self)
```

```
class allensdk.mouse_connectivity.grid.subimage.base_subimage.PolygonSubImage(reduce_level,  
                                                                           in_dims,  
                                                                           in_spacing,  
                                                                           coarse_spacing,  
                                                                           poly-  
                                                                           gon_info,  
                                                                           *args,  
                                                                           **kwargs)
```

Bases: *allensdk.mouse\_connectivity.grid.subimage.base\_subimage.SubImage*

```
get_polygons(self)
```

```
optional_polys = []
```

```
required_polys = []
```

```
setup_images(self)
```

```
class allensdk.mouse_connectivity.grid.subimage.base_subimage.SegmentationSubImage (reduce_level,
in_dims,
in_spacing,
coarse_spacing,
segmentation_paths,
*args,
**kwargs)
```

Bases: `allensdk.mouse_connectivity.grid.subimage.base_subimage.SubImage`

```
extract_injection_from_segmentation (self, segmentation_name='segmentation', injection_name='injection')
```

### Notes

Currently, the segmentation uses a series of codes to map 8-bit values onto meaningful classifications. The code for signal pixels is a 1 in at least one of of the 5 rightmost bits.

```
extract_signal_from_segmentation (self, segmentation_name='segmentation', signal_name='signal')
```

### Notes

Currently, the segmentation uses a series of codes to map 8-bit values onto meaningful classifications. The code for signal pixels is a 1 in the leftmost bit.

In some cases, bit 5 indicates that the pixel was not removed in a postfiltering process. Optionally, this postfilter can be applied in gridding.

```
get_segmentation (self)
```

```
process_segmentation (self)
```

```
read_segmentation_image (self, segmentation_name='segmentation')
```

### Notes

We downsample in memory rather than using the jp2 pyramid because the segmentation is a label image.

```
required_segmentations = []
```

```
setup_images (self)
```

```
class allensdk.mouse_connectivity.grid.subimage.base_subimage.SubImage (reduce_level,
in_dims,
in_spacing,
coarse_spacing,
*args,
**kwargs)
```

Bases: `object`

```
apply_mask (self, image_name, mask_name, positive=True)
```

```
apply_pixel_counter (self, accumulator_name, image)
```

```
binarize (self, image_name)
```

```
compute_coarse_planes (self)
```

```
make_pixel_counter (self)
```

```
pixel_counter
```

```
setup_images (self)
```

```
allensdk.mouse_connectivity.grid.subimage.base_subimage.run_subimage (input_data)
```

### **allensdk.mouse\_connectivity.grid.subimage.cav\_subimage module**

```
class allensdk.mouse_connectivity.grid.subimage.cav_subimage.CavSubImage (reduce_level,  
                                                                           in_dims,  
                                                                           in_spacing,  
                                                                           coarse_spacing,  
                                                                           poly-  
                                                                           gon_info,  
                                                                           *args,  
                                                                           **kwargs)
```

```
Bases:          allensdk.mouse_connectivity.grid.subimage.base_subimage.  
              PolygonSubImage
```

```
compute_coarse_planes (self)
```

```
required_polys = ['missing_tile', 'cav_tracer']
```

### **allensdk.mouse\_connectivity.grid.subimage.classic\_subimage module**

```
class allensdk.mouse_connectivity.grid.subimage.classic_subimage.ClassicSubImage (reduce_level,  
                                                                           in_dims,  
                                                                           in_spacing,  
                                                                           coarse_spacing,  
                                                                           poly-  
                                                                           gon_info,  
                                                                           seg-  
                                                                           men-  
                                                                           ta-  
                                                                           tion_paths,  
                                                                           in-  
                                                                           ten-  
                                                                           sity_paths,  
                                                                           in-  
                                                                           jec-  
                                                                           tion_polygon_  
                                                                           *args,  
                                                                           **kwargs)
```

```
Bases:          allensdk.mouse_connectivity.grid.subimage.base_subimage.  
              IntensitySubImage, allensdk.mouse_connectivity.grid.subimage.base_subimage.  
              SegmentationSubImage,          allensdk.mouse_connectivity.grid.subimage.  
              base_subimage.PolygonSubImage
```

```
compute_coarse_planes (self)
```

```
compute_injection (self)
```

```
compute_intensity (self)
```

```

compute_projection(self)
compute_sum_pixels(self)
optional_polys = ['aav_tracer']
process_segmentation(self)
required_intensities = ['green']
required_polys = ['missing_tile', 'no_signal', 'aav_exclusion']
required_segmentations = ['segmentation']

```

## allensdk.mouse\_connectivity.grid.subimage.count\_subimage module

```

class allensdk.mouse_connectivity.grid.subimage.count_subimage.CountSubImage(reduce_level,
in_dims,
in_spacing,
coarse_spacing,
polygon_info,
segmentation_paths,
injection_polygon_key=
*args,
**kwargs)

```

Bases: `allensdk.mouse_connectivity.grid.subimage.base_subimage.SegmentationSubImage`, `allensdk.mouse_connectivity.grid.subimage.base_subimage.PolygonSubImage`

```

compute_coarse_planes(self)
compute_injection(self)
compute_projection(self)
compute_sum_pixels(self)
process_segmentation(self)
required_polys = ['missing_tile', 'no_signal', 'aav_exclusion']
required_segmentations = ['segmentation']

```

## Module contents

`allensdk.mouse_connectivity.grid.subimage.run_subimage(input_data)`

## allensdk.mouse\_connectivity.grid.utilities package

## Submodules

**allensdk.mouse\_connectivity.grid.utilities.downsampling\_utilities module**

```
allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.apply_divisions(image,
                                          win-
                                          dow_size)

allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.block_average(volume,
                                          fac-
                                          tor)

allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.conv(image,
                               fac-
                               tor,
                               win-
                               dow_size)

allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.downsample_average(volume,
                                              cur-
                                              rent_spa-
                                              tar-
                                              get_spa-
                                              ce)

allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.extract(image,
                                   fac-
                                   tor,
                                   win-
                                   dow_size,
                                   win-
                                   dow_step,
                                   out-
                                   put_shape)

allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.window_average(volume,
                                          fac-
                                          tor)
```

**allensdk.mouse\_connectivity.grid.utilities.image\_utilities module**

```
allensdk.mouse_connectivity.grid.utilities.image_utilities.block_apply(in_image,
                                out_shape,
                                dtype,
                                blocks,
                                fn)

allensdk.mouse_connectivity.grid.utilities.image_utilities.build_affine_transform(aff_params)

allensdk.mouse_connectivity.grid.utilities.image_utilities.build_composite_transform(dfnfield,
                                             aff_params)

allensdk.mouse_connectivity.grid.utilities.image_utilities.compute_coarse_parameters(in_dims,
                                              in_spacing,
                                              out_spacing,
                                              re-
                                              duce_level)

allensdk.mouse_connectivity.grid.utilities.image_utilities.grid_image_blocks(in_shape,
                                       in_spacing,
                                       out_spacing)
```

```

allensdk.mouse_connectivity.grid.utilities.image_utilities.image_from_array(array,
                                     spacing,
                                     origin=True)

allensdk.mouse_connectivity.grid.utilities.image_utilities.new_image(dims,
                             spacing,
                             dtype,
                             origin=True)

allensdk.mouse_connectivity.grid.utilities.image_utilities.np_sitk_convert(np_type)

allensdk.mouse_connectivity.grid.utilities.image_utilities.rasterize_polygons(shape,
                                     scale,
                                     polys)

allensdk.mouse_connectivity.grid.utilities.image_utilities.read_intensity_image()

allensdk.mouse_connectivity.grid.utilities.image_utilities.read_segmentation_image(path)

allensdk.mouse_connectivity.grid.utilities.image_utilities.resample_into_volume(image,
                                     transform,
                                     z,
                                     vol,
                                     dtype=8)

allensdk.mouse_connectivity.grid.utilities.image_utilities.resample_volume(volume,
                                     dims,
                                     spacing,
                                     interpolator=None,
                                     transform=None)

allensdk.mouse_connectivity.grid.utilities.image_utilities.set_image_spacing(image,
                                     spacing,
                                     origin=True)

allensdk.mouse_connectivity.grid.utilities.image_utilities.sitk_np_convert(sitk_type)

allensdk.mouse_connectivity.grid.utilities.image_utilities.write_volume(volume,
                                     name,
                                     prefix=None,
                                     specify_resolution=None,
                                     extension='nrrd',
                                     paths=None)

```

## Module contents

### allensdk.mouse\_connectivity.grid.writers package

## Module contents

```
allensdk.mouse_connectivity.grid.writers.cav_writer(gridder, grid_prefix, accumulator_prefix, **kwargs)
allensdk.mouse_connectivity.grid.writers.classic_writer(gridder, grid_prefix, accumulator_prefix, target_spacings, **kwargs)
allensdk.mouse_connectivity.grid.writers.count_writer(gridder, grid_prefix, accumulator_prefix, target_spacings, **kwargs)
allensdk.mouse_connectivity.grid.writers.handle_pyramid(isg, key, target_spacings, prefix, paths)
allensdk.mouse_connectivity.grid.writers.ratio_and_pyramid(isg, num, den, out, accumulator_prefix, grid_prefix, target_spacings, paths)
```

## Submodules

### allensdk.mouse\_connectivity.grid.image\_series\_gridder module

```
class allensdk.mouse_connectivity.grid.image_series_gridder.ImageSeriesGridder(in_dims, in_spacing, out_dims, out_spacing, reduce_level, subimages, subimage_kwargs, nprocesses, affine_params, dfm fld_path)

Bases: object

accumulator_to_numpy(self, key, cb)
build_coarse_grids(self)
consume_volume(self, key, cb)
initialize_coarse_volume(self, key, dtype)
make_ratio_volume(self, num_key, den_key, ratio_key)
    assume parents numified
paste_slice(self, key, index, slice_array)
```



```

paste_subimage (self, index, output)
    Inserts planar accumulators into coarse grid volumes

resample_volume (self, key)

set_coarse_grid_parameters (self)

setup_subimages (self)

transform

```

## Module contents

## Module contents

### 6.1.10 allensdk.test\_utilities package

#### Submodules

#### allensdk.test\_utilities.custom\_comparators module

```

class allensdk.test_utilities.custom_comparators.WhitespaceStrippedString (string:
                                                                    str,
                                                                    whites-
                                                                    pace_chars:
                                                                    str
                                                                    =
                                                                    '\s',
                                                                    ASCII:
                                                                    bool
                                                                    =
                                                                    False)

```

Bases: object

**Comparator class to compare strings that have been stripped of** whitespace. By default removes any unicode whitespace character that matches the regex `s`, (which includes `[` `]`, and other unicode whitespace characters).

#### allensdk.test\_utilities.regression\_fixture module

```
allensdk.test_utilities.regression_fixture.get_list_of_path_dict()
```

#### allensdk.test\_utilities.temp\_dir module

```
allensdk.test_utilities.temp_dir.temp_dir(request)
```

## Module contents

## 6.2 Submodules

### 6.2.1 allensdk.deprecated module

`allensdk.deprecated.class_deprecated(message=None)`

`allensdk.deprecated.deprecated(message=None)`

`allensdk.deprecated.legacy(message=None)`

### 6.2.2 allensdk.tmp module

## 6.3 Module contents


**exception** `allensdk.OneResultExpectedError`

Bases: `RuntimeError`

`allensdk.one(x)`

The Allen Software Development Kit houses source code for reading and processing Allen Brain Atlas data. The Allen SDK focuses on the Allen Brain Observatory, Cell Types Database, and Mouse Brain Connectivity Atlas.

**Attention:** As of October 2019, we have dropped Python 2 support and any files with a py2 dependency (for example analysis files) have been updated.



`_static/sdk_cam.png`

---

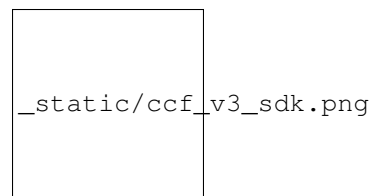
### Allen Brain Observatory

---

The [Allen Brain Observatory](#) is a data resource for understanding sensory processing in the mouse visual cortex. This study systematically measures visual responses in multiple cortical areas and layers using two-photon calcium imaging of GCaMP6-labeled neurons targeted using Cre driver lines. Response characterizations include orientation tuning, spatial and temporal frequency tuning, temporal dynamics, and spatial receptive field structure.

The mean fluorescence traces for all segmented cells are available in the Neurodata Without Borders file format ([NWB files](#)). These files contain standardized descriptions of visual stimuli to support stimulus-specific tuning analysis. The Allen SDK provides code to:

- download and organize experiment data according to cortical area, imaging depth, and Cre line
- remove the contribution of neuropil signal from fluorescence traces
- access (or compute) dF/F traces based on the neuropil-corrected traces
- perform stimulus-specific tuning analysis (e.g. drifting grating direction tuning)





---

### Allen Cell Types Database

---

The [Allen Cell Types Database](#) contains electrophysiological and morphological characterizations of individual neurons in the mouse primary visual cortex. The Allen SDK provides Python code for accessing electrophysiology measurements ([NWB files](#)) for all neurons and morphological reconstructions ([SWC files](#)) for a subset of neurons.

The Database also contains two classes of models fit to this data set: biophysical models produced using the NEURON simulator and generalized leaky integrate and fire models (GLIFs) produced using custom Python code provided with this toolkit.

The Allen SDK provides sample code demonstrating how to download neuronal model parameters from the Allen Brain Atlas API and run your own simulations using stimuli from the Allen Cell Types Database or custom current injections:

- *Biophysical Models*
- *Generalized LIF Models*





---

### Allen Mouse Brain Connectivity Atlas

---

The [Allen Mouse Brain Connectivity Atlas](#) is a high-resolution map of neural connections in the mouse brain. Built on an array of transgenic mice genetically engineered to target specific cell types, the Atlas comprises a unique compendium of projections from selected neuronal populations throughout the brain. The primary data of the Atlas consists of high-resolution images of axonal projections targeting different anatomic regions or various cell types using Cre-dependent specimens. Each data set is processed through an informatics data analysis pipeline to obtain spatially mapped quantified projection information.

The Allen SDK provides Python code for accessing experimental metadata along with projection signal volumes registered to a common coordinate framework. This framework has structural annotations, which allows users to compute structure-level signal statistics.

See the [mouse connectivity section](#) for more details.





## CHAPTER 10

---

### What's New - 1.7.0 (April 29, 2020)

---

As of the 1.7.0 release:

- Added functionality so internal users can now access *eye\_tracking* ellipse fit data from behavior + ophys Session objects
- Added a new mixin for managing processing parameters for Session objects
- Update the monitor delay calculation to better handle edge cases; no longer provide a default delay value if encounter an error
- Added support for additional sync file line labels
- Fixed bug with loading line labels from sync files



---

## What's New - 1.6.0 (March 23, 2020)

---

As of the 1.6.0 release:

- added `get_receptive_field alias()` for `_get_rf()` in `allensdk/brain_observatory/ecephys/stimulus_analysis/receptive_field_mapping.py`
- Added required version to namespace and caches spec in ecephy nwb outputs in `allensdk/brain_observatory/ecephys/nwb/AIBS_ecephys_namespace.yaml`
- Added version for ophys behavior nwb output to `allensdk/brain_observatory/nwb/AIBS_ophys_behavior_namespace.yaml`
- Behavior and ECEphys project caches no longer accept arbitrary keywords to prevent confusion when user supplies incorrect kwargs to constructor.
- New ecephys notebook for optotagging tutorial.



---

### What's New - 1.5.0 (February 10, 2020)

---

As of the 1.5.0 release:

- users have an option to provide credentials for accessing the database either explicitly via public API or by setting up the environment variables
- allow users to modify BehaviorDataSession and BehaviorOphysSession data
- invalid extracellular electrophysiology spikes no longer show up as spikes at time -1
- morphology.apply\_affine correctly rescales radii



## CHAPTER 13

---

### Previous Release Notes

---

- 1.4.0
- 1.3.0
- 1.2.0
- 1.1.1
- 1.1.0
- 1.0.2
- 0.16.3
- 0.16.2
- 0.16.1
- 0.16.0
- 0.14.5
- 0.14.4
- 0.14.3
- 0.14.2
- 0.13.2
- 0.13.1
- 0.13.0
- 0.12.4





---

## Bibliography

---

[1] Allen Brain Atlas Data Portal: [Downloading a WellKnownFile](#).



### a

allensdk, 374  
allensdk.api, 80  
allensdk.api.api, 72  
allensdk.api.cache, 76  
allensdk.api.caching\_utilities, 79  
allensdk.api.queries, 72  
allensdk.api.queries.annotated\_section\_data\_sets\_api, 43  
allensdk.api.queries.biophysical\_api, 44  
allensdk.api.queries.brain\_observatory\_api, 46  
allensdk.api.queries.cell\_types\_api, 49  
allensdk.api.queries.connected\_services, 51  
allensdk.api.queries.glif\_api, 52  
allensdk.api.queries.grid\_data\_api, 52  
allensdk.api.queries.image\_download\_api, 54  
allensdk.api.queries.mouse\_atlas\_api, 57  
allensdk.api.queries.mouse\_connectivity\_api, 58  
allensdk.api.queries.ontologies\_api, 61  
allensdk.api.queries.reference\_space\_api, 63  
allensdk.api.queries.rma\_api, 65  
allensdk.api.queries.rma\_pager, 70  
allensdk.api.queries.rma\_template, 70  
allensdk.api.queries.svg\_api, 70  
allensdk.api.queries.synchronization\_api, 70  
allensdk.api.queries.tree\_search\_api, 72  
  
allensdk.brain\_observatory, 216  
allensdk.brain\_observatory.argschema\_utilities, 184  
allensdk.brain\_observatory.behavior, 112  
allensdk.brain\_observatory.behavior.behavior\_data\_s, 89  
allensdk.brain\_observatory.behavior.behavior\_ophys, 91  
allensdk.brain\_observatory.behavior.behavior\_ophys, 81  
allensdk.brain\_observatory.behavior.behavior\_ophys, 80  
allensdk.brain\_observatory.behavior.behavior\_ophys, 92  
allensdk.brain\_observatory.behavior.behavior\_proje, 95  
allensdk.brain\_observatory.behavior.behavior\_proje, 97  
allensdk.brain\_observatory.behavior.criteria, 98  
allensdk.brain\_observatory.behavior.dprime, 100  
allensdk.brain\_observatory.behavior.eye\_tracking\_p, 101  
allensdk.brain\_observatory.behavior.image\_api, 104  
allensdk.brain\_observatory.behavior.internal, 85  
allensdk.brain\_observatory.behavior.internal.behav, 82  
allensdk.brain\_observatory.behavior.internal.behav, 83  
allensdk.brain\_observatory.behavior.internal.behav, 85  
allensdk.brain\_observatory.behavior.metadata\_proces, 104  
allensdk.brain\_observatory.behavior.mtrain, 104  
allensdk.brain\_observatory.behavior.rewards\_proces, 106  
allensdk.brain\_observatory.behavior.running\_proces, 106  
allensdk.brain\_observatory.behavior.schemas,

### b

106 allensdk.brain\_observatory.ecephys.ecephys\_project.  
 allensdk.brain\_observatory.behavior.session\_metadata, 122  
 107 allensdk.brain\_observatory.ecephys.ecephys\_project.  
 allensdk.brain\_observatory.behavior.stimulus\_processing, 124  
 108 allensdk.brain\_observatory.ecephys.ecephys\_project.  
 allensdk.brain\_observatory.behavior.sync, 125  
 86 allensdk.brain\_observatory.ecephys.ecephys\_project.  
 allensdk.brain\_observatory.behavior.sync.processing, 126  
 85 allensdk.brain\_observatory.ecephys.ecephys\_project.  
 allensdk.brain\_observatory.behavior.trial\_masks, 158  
 108 allensdk.brain\_observatory.ecephys.ecephys\_session.  
 allensdk.brain\_observatory.behavior.trials\_processing, 162  
 109 allensdk.brain\_observatory.ecephys.ecephys\_session.  
 allensdk.brain\_observatory.behavior.validation, 128  
 111 allensdk.brain\_observatory.ecephys.ecephys\_session.  
 allensdk.brain\_observatory.behavior.write\_nwb, 125  
 89 allensdk.brain\_observatory.ecephys.ecephys\_session.  
 allensdk.brain\_observatory.brain\_observatory\_exceptions, 126  
 185 allensdk.brain\_observatory.ecephys.ecephys\_session.  
 allensdk.brain\_observatory.brain\_observatory\_plotting, 127  
 185 allensdk.brain\_observatory.ecephys.file\_io,  
 allensdk.brain\_observatory.chisquare\_categorical, 129  
 186 allensdk.brain\_observatory.ecephys.file\_io.continuous,  
 allensdk.brain\_observatory.circle\_plots, 128  
 186 allensdk.brain\_observatory.ecephys.file\_io.ecephys.  
 allensdk.brain\_observatory.demixer, 188 128  
 allensdk.brain\_observatory.dff, 189 allensdk.brain\_observatory.ecephys.file\_io.stim\_files,  
 allensdk.brain\_observatory.drifting\_gratings, 129  
 191 allensdk.brain\_observatory.ecephys.lfp\_subsampling,  
 allensdk.brain\_observatory.ecephys, 170 132  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 130  
 116 allensdk.brain\_observatory.ecephys.lfp\_subsampling,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 130  
 112 allensdk.brain\_observatory.ecephys.nwb,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 132  
 114 allensdk.brain\_observatory.ecephys.optotagging\_tables,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 133  
 115 allensdk.brain\_observatory.ecephys.stimulus\_analysis,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 147  
 115 allensdk.brain\_observatory.ecephys.stimulus\_analysis,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 133  
 116 allensdk.brain\_observatory.ecephys.stimulus\_analysis,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 134  
 116 allensdk.brain\_observatory.ecephys.stimulus\_analysis,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 136  
 125 allensdk.brain\_observatory.ecephys.stimulus\_analysis,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 136  
 117 allensdk.brain\_observatory.ecephys.stimulus\_analysis,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 137  
 117 allensdk.brain\_observatory.ecephys.stimulus\_analysis,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 138  
 118 allensdk.brain\_observatory.ecephys.stimulus\_analysis,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 142  
 121 allensdk.brain\_observatory.ecephys.stimulus\_analysis,  
 allensdk.brain\_observatory.ecephys.alignment\_masks, 143

[allensdk.brain\\_observatory.ecephys.stimulus\\_analysis](#), 169  
[allensdk.brain\\_observatory.ecephys.stimulus\\_analysis](#), 156  
[allensdk.brain\\_observatory.ecephys.stimulus\\_analysis](#), 149  
[allensdk.brain\\_observatory.ecephys.stimulus\\_analysis](#), 153  
[allensdk.brain\\_observatory.ecephys.stimulus\\_analysis](#), 154  
[allensdk.brain\\_observatory.ecephys.stimulus\\_analysis](#), 155  
[allensdk.brain\\_observatory.ecephys.stimulus\\_analysis](#), 148  
[allensdk.brain\\_observatory.ecephys.stimulus\\_analysis](#), 148  
[allensdk.brain\\_observatory.ecephys.visualize](#), 156  
[allensdk.brain\\_observatory.ecephys.write\\_nwb](#), 157  
[allensdk.brain\\_observatory.extract\\_running\\_speed](#), 170  
[allensdk.brain\\_observatory.findlevel](#), 192  
[allensdk.brain\\_observatory.gaze\\_mapping](#), 170  
[allensdk.brain\\_observatory.locally\\_sparse\\_embeddings](#), 192  
[allensdk.brain\\_observatory.natural\\_movie\\_embeddings](#), 194  
[allensdk.brain\\_observatory.natural\\_scenes\\_embeddings](#), 194  
[allensdk.brain\\_observatory.nwb](#), 171  
[allensdk.brain\\_observatory.nwb.metadata](#), 170  
[allensdk.brain\\_observatory.nwb.nwb\\_api](#), 170  
[allensdk.brain\\_observatory.nwb.schemas](#), 171  
[allensdk.brain\\_observatory.observatory\\_plots](#), 195  
[allensdk.brain\\_observatory.oophys](#), 174  
[allensdk.brain\\_observatory.oophys.trace\\_extraction](#), 174  
[allensdk.brain\\_observatory.r\\_neuropil](#), 197  
[allensdk.brain\\_observatory.receptive\\_field\\_analysis](#), 183  
[allensdk.brain\\_observatory.receptive\\_field\\_analysis](#), 175  
[allensdk.brain\\_observatory.receptive\\_field\\_analysis](#), 178  
[allensdk.brain\\_observatory.receptive\\_field\\_analysis](#), 178  
[allensdk.brain\\_observatory.receptive\\_field\\_analysis](#), 178

**C**  
[allensdk.config](#), 225  
[allensdk.config.app](#), 219  
[allensdk.config.app.application\\_config](#), 217  
[allensdk.config.manifest](#), 222  
[allensdk.config.manifest\\_builder](#), 224  
[allensdk.config.model](#), 222  
[allensdk.config.model.description](#), 221  
[allensdk.config.model.description\\_parser](#), 222  
[allensdk.config.model.formats](#), 221  
[allensdk.config.model.formats.hdf5\\_util](#), 219  
[allensdk.config.model.formats.json\\_description\\_parser](#), 219  
[allensdk.config.model.formats.chisquarperf](#), 219  
[allensdk.config.model.formats.pycfg\\_description\\_parser](#), 220  
[allensdk.config.model.formats.eventdetection](#), 220  
[allensdk.core](#), 264  
[allensdk.core.fit\\_parameters](#), 225  
[allensdk.core.authentication](#), 226

[allensdk.core.brain\\_observatory\\_cache](#), 227  
[allensdk.core.brain\\_observatory\\_nwb\\_data\\_set](#), 231  
[allensdk.core.cache\\_method\\_utilities](#), 235  
[allensdk.core.cell\\_types\\_cache](#), 235  
[allensdk.core.dat\\_utilities](#), 238  
[allensdk.core.exceptions](#), 238  
[allensdk.core.h5\\_utilities](#), 239  
[allensdk.core.json\\_utilities](#), 239  
[allensdk.core.lazy\\_property](#), 225  
[allensdk.core.lazy\\_property.lazy\\_property](#), 225  
[allensdk.core.lazy\\_property.lazy\\_property](#), 225  
[allensdk.core.mouse\\_connectivity\\_cache](#), 240  
[allensdk.core.nwb\\_data\\_set](#), 245  
[allensdk.core.obj\\_utilities](#), 247  
[allensdk.core.ontology](#), 248  
[allensdk.core.ophys\\_experiment\\_session\\_id\\_mapping](#), 249  
[allensdk.core.reference\\_space](#), 249  
[allensdk.core.reference\\_space\\_cache](#), 251  
[allensdk.core.simple\\_tree](#), 253  
[allensdk.core.sitk\\_utilities](#), 256  
[allensdk.core.structure\\_tree](#), 257  
[allensdk.core.swc](#), 260  
[allensdk.core.typing](#), 264  
**d**  
[allensdk.deprecated](#), 374  
**e**  
[allensdk.ephys](#), 276  
[allensdk.ephys.ephys\\_extractor](#), 264  
[allensdk.ephys.ephys\\_features](#), 268  
[allensdk.ephys.extract\\_cell\\_features](#), 275  
[allensdk.ephys.feature\\_extractor](#), 276  
**i**  
[allensdk.internal](#), 352  
[allensdk.internal.api](#), 287  
[allensdk.internal.api.api\\_prerelease](#), 282  
[allensdk.internal.api.behavior\\_data\\_lims\\_api](#), 282  
[allensdk.internal.api.behavior\\_lims\\_api](#), 284  
[allensdk.internal.api.behavior\\_ophys\\_api](#), 284  
[allensdk.internal.api.lims\\_api](#), 285  
[allensdk.internal.api.mtrain\\_api](#), 286  
[allensdk.internal.api.ophys\\_lims\\_api](#), 286  
[allensdk.internal.api.queries](#), 281  
[allensdk.internal.api.queries.biophysical\\_module\\_api](#), 277  
[allensdk.internal.api.queries.biophysical\\_module\\_report](#), 277  
[allensdk.internal.api.queries.grid\\_data\\_api\\_prerelease](#), 278  
[allensdk.internal.api.queries.mouse\\_connectivity\\_api](#), 279  
[allensdk.internal.api.queries.optimize\\_config\\_read](#), 280  
[allensdk.internal.api.queries.pre\\_release](#), 281  
[allensdk.internal.brain\\_observatory](#), 301  
[allensdk.internal.brain\\_observatory.annotated\\_regions](#), 288  
[allensdk.internal.brain\\_observatory.demix\\_report](#), 289  
[allensdk.internal.brain\\_observatory.demixer](#), 290  
[allensdk.internal.brain\\_observatory.eye\\_calibration](#), 291  
[allensdk.internal.brain\\_observatory.fit\\_ellipse](#), 294  
[allensdk.internal.brain\\_observatory.frame\\_stream](#), 294  
[allensdk.internal.brain\\_observatory.itracker\\_utils](#), 295  
[allensdk.internal.brain\\_observatory.mask\\_set](#), 296  
[allensdk.internal.brain\\_observatory.ophys\\_session\\_id\\_mapping](#), 297  
[allensdk.internal.brain\\_observatory.resources](#), 288  
[allensdk.internal.brain\\_observatory.roi\\_filter\\_utilities](#), 298  
[allensdk.internal.brain\\_observatory.time\\_sync](#), 300  
[allensdk.internal.core](#), 305  
[allensdk.internal.core.lims\\_pipeline\\_module](#), 301  
[allensdk.internal.core.lims\\_utilities](#), 302  
[allensdk.internal.core.mouse\\_connectivity\\_cache\\_prerelease](#), 303  
[allensdk.internal.core.simpletree](#), 304  
[allensdk.internal.core.swc](#), 304  
[allensdk.internal.ephys](#), 310  
[allensdk.internal.ephys.core\\_feature\\_extract](#), 305  
[allensdk.internal.ephys.plot\\_qc\\_figures](#), 305

---

**Python Module Index**
**395**

[341](#)  
[allensdk.internal.mouse\\_connectivity.projects.tissuecyte\\_stitching,](#)  
[342](#)  
[allensdk.internal.mouse\\_connectivity.projects.tissuecyte\\_stitching,](#)  
[343](#)  
[allensdk.internal.mouse\\_connectivity.projects.tissuecyte\\_stitching,](#)  
[342](#)  
[allensdk.internal.mouse\\_connectivity.projects.tissuecyte\\_stitching,](#)  
[343](#)  
[allensdk.internal.pipeline\\_modules,](#)[351](#)  
[allensdk.internal.pipeline\\_modules.gbm,](#)  
[344](#)  
[allensdk.internal.pipeline\\_modules.gbm.generate\\_gbm\\_analysis\\_run\\_records,](#)  
[344](#)  
[allensdk.internal.pipeline\\_modules.gbm.generate\\_gbm\\_heatmap,](#)  
[344](#)  
[allensdk.internal.pipeline\\_modules.gbm.generate\\_gbm\\_sample\\_metadata,](#)  
[344](#)  
[allensdk.internal.pipeline\\_modules.run\\_annotated\\_region\\_metrics,](#)  
[344](#)  
[allensdk.internal.pipeline\\_modules.run\\_demixing,](#)[369](#)  
[345](#)  
[allensdk.internal.pipeline\\_modules.run\\_dff\\_computation,](#)  
[345](#)  
[allensdk.internal.pipeline\\_modules.run\\_neuropil\\_correction,](#)  
[345](#)  
[allensdk.internal.pipeline\\_modules.run\\_observatory\\_analysis,](#)  
[346](#)  
[allensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails,](#)  
[346](#)  
[allensdk.internal.pipeline\\_modules.runophys\\_eye\\_calibration,](#)  
[348](#)  
[allensdk.internal.pipeline\\_modules.runophys\\_eye\\_calibration,](#)  
[349](#)  
[allensdk.internal.pipeline\\_modules.runophys\\_eye\\_calibration,](#)  
[349](#)  
[allensdk.internal.pipeline\\_modules.run\\_tissuecyte\\_uniutils.classifier,](#)  
[351](#)

## m

[allensdk.model,](#)[365](#)  
[allensdk.model.biophys\\_sim,](#)[353](#)  
[allensdk.model.biophys\\_sim.bps\\_command,](#)  
[352](#)  
[allensdk.model.biophys\\_sim.config,](#)[353](#)  
[allensdk.model.biophys\\_sim.neuron,](#)[352](#)  
[allensdk.model.biophys\\_sim.neuron.hoc\\_utils,](#)  
[352](#)  
[allensdk.model.biophys\\_sim.scripts,](#)[352](#)  
[allensdk.model.biophysical,](#)[356](#)  
[allensdk.model.biophysical.run\\_simulate,](#)  
[353](#)  
[allensdk.model.biophysical.runner,](#)[353](#)  
[allensdk.model.biophysical.utils,](#)[354](#)



## A

`ab_from_diagonals()` (in module `allensdk.brain_observatory.r_neuropil`), 198  
`ab_from_T()` (in module `allensdk.brain_observatory.r_neuropil`), 198  
`accumulator_to_numpy()` (in module `allensdk.mouse_connectivity.grid.image_series_gridder.ImageSeriesGridder`), 372  
`acquisition_rate` (in module `allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis`), 207  
`actual_parameters_from_normalized()` (in module `allensdk.internal.model.biophysical.deap_utils.Utils`), 313  
`adaptation_index()` (in module `allensdk.ephys.feature_extractor.EphysFeatureExtractor`), 276  
`adaptation_index()` (in module `allensdk.ephys.ephys_features`), 268  
`add()` (`allensdk.brain_observatory.stimulus_info.BinaryIntervalSearchTree`), 208  
`add_angle_labels()` (in module `allensdk.brain_observatory.circle_plots`), 187  
`add_arrow()` (in module `allensdk.brain_observatory.circle_plots`), 187  
`add_average_image()` (in module `allensdk.brain_observatory.nwb`), 171  
`add_cell_specimen_table()` (in module `allensdk.brain_observatory.nwb`), 171  
`add_corrected_fluorescence_traces()` (in module `allensdk.brain_observatory.nwb`), 171  
`add_dff_traces()` (in module `allensdk.brain_observatory.nwb`), 171  
`add_eye_gaze_data_interfaces()` (in module `allensdk.brain_observatory.nwb`), 171  
`add_eye_gaze_mapping_data_to_nwbfile()` (in module `allensdk.brain_observatory.nwb`), 171  
`add_eye_tracking_ellipse_fit_data_to_nwbfile()` (in module `allensdk.brain_observatory.nwb`), 171  
`add_file()` (`allensdk.config.manifest.Manifest`), 222  
`add_image()` (in module `allensdk.brain_observatory.nwb`), 172  
`add_invalid_times()` (in module `allensdk.brain_observatory.nwb`), 172  
`add_licks()` (in module `allensdk.brain_observatory.nwb`), 172  
`add_manifest_paths()` (`allensdk.api.cache.Cache`), 76  
`add_manifest_paths()` (`allensdk.brain_observatory.behavior.behavior_project_cache.BehaviorProjectCache`), 95  
`add_manifest_paths()` (`allensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache`), 159  
`add_manifest_paths()` (`allensdk.core.mouse_connectivity_cache.MouseConnectivityCache`), 241  
`add_manifest_paths()` (`allensdk.core.reference_space_cache.ReferenceSpaceCache`), 252  
`add_manifest_paths()` (`allensdk.internal.core.mouse_connectivity_cache_prerelease.MouseConnectivityCachePrerelease`), 303  
`add_max_projection()` (in module `allensdk.brain_observatory.nwb`), 172  
`add_metadata()` (in module `allensdk.brain_observatory.nwb`), 172  
`add_motion_correction()` (in module `allensdk.brain_observatory.nwb`), 172  
`add_number_to_shuffled_movie()` (in module `allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities`), 153  
`add_page_params()` (`allensdk.brain_observatory.ecephys.ecephys_project_api.rma_engine.RmaEngine`), 124

`add_path()` (*allensdk.config.manifest.Manifest* method), 223  
`add_path()` (*allensdk.config.manifest\_builder.ManifestBuilder* method), 224  
`add_paths()` (*allensdk.config.manifest.Manifest* method), 223  
`add_rewards()` (in module *allensdk.brain\_observatory.nwb*), 172  
`add_running_data_df_to_nwbfile()` (in module *allensdk.brain\_observatory.nwb*), 172  
`add_running_speed_to_nwbfile()` (in module *allensdk.brain\_observatory.nwb*), 172  
`add_section()` (*allensdk.config.manifest\_builder.ManifestBuilder* method), 224  
`add_segmentation_mask_image()` (in module *allensdk.brain\_observatory.nwb*), 173  
`add_stimulus_index()` (in module *allensdk.brain\_observatory.nwb*), 173  
`add_stimulus_presentations()` (in module *allensdk.brain\_observatory.nwb*), 173  
`add_stimulus_template()` (in module *allensdk.brain\_observatory.nwb*), 173  
`add_stimulus_timestamps()` (in module *allensdk.brain\_observatory.nwb*), 173  
`add_task_parameters()` (in module *allensdk.brain\_observatory.nwb*), 173  
`add_to_fit_parameters_dict_single()` (in module *allensdk.brain\_observatory.receptive\_field\_analysis\_fit\_parameters*), 178  
`add_trials()` (in module *allensdk.brain\_observatory.nwb*), 173  
`adjust_r_for_negativity()` (in module *allensdk.internal.pipeline\_modules.run\_neuropil\_correction*), 345  
`advance_combination()` (in module *allensdk.brain\_observatory.chisquare\_categorical*), 186  
`age_in_days` (*allensdk.brain\_observatory.ecephys.ecephys\_sessionule.EcephysSession* attribute), 164  
`age_in_days` (*allensdk.brain\_observatory.ecephys.nwb.EcephysLabMetaData* attribute), 132  
`AIC()` (in module *allensdk.internal.model.AIC*), 326  
`AICc()` (in module *allensdk.internal.model.AIC*), 326  
`align_and_cut_spikes()` (in module *allensdk.internal.model.glif.find\_spikes*), 316  
`align_running_speed()` (in module *allensdk.core.brain\_observatory\_nwb\_data\_set*), 235  
`ALIGNMENT3D_KEY` (*allensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache* attribute), 241  
`ALL` (*allensdk.api.queries.rma\_api.RmaApi* attribute), 65  
`all_stimuli()` (in module *allensdk.brain\_observatory.stimulus\_info*), 210  
`AllActiveUtils` (class in *allensdk.model.biophysical.utils*), 354  
`allensdk` (module), 374  
`allensdk.api` (module), 80  
`allensdk.api.api` (module), 72  
`allensdk.api.cache` (module), 76  
`allensdk.api.caching_utilities` (module), 79  
`allensdk.api.queries` (module), 72  
`allensdk.api.queries.annotated_section_data_sets_api` (module), 43  
`allensdk.api.queries.biophysical_api` (module), 44  
`allensdk.api.queries.brain_observatory_api` (module), 46  
`allensdk.api.queries.cell_types_api` (module), 49  
`allensdk.api.queries.connected_services` (module), 51  
`allensdk.api.queries.glif_api` (module), 52  
`allensdk.api.queries.grid_data_api` (module), 52  
`allensdk.api.queries.image_download_api` (module), 54  
`allensdk.api.queries.mouse_atlas_api` (module), 57  
`allensdk.api.queries.mouse_connectivity_api` (module), 58  
`allensdk.api.queries.ontologies_api` (module), 61  
`allensdk.api.queries.reference_space_api` (module), 63  
`allensdk.api.queries.rma_api` (module), 65  
`allensdk.api.queries.rma_pager` (module), 70  
`allensdk.api.queries.rma_template` (module), 70  
`allensdk.api.queries.svg_api` (module), 70  
`allensdk.api.queries.synchronization_api` (module), 70  
`allensdk.api.queries.tree_search_api` (module), 72  
`allensdk.brain_observatory` (module), 216  
`allensdk.brain_observatory.argschema_utilities` (module), 184  
`allensdk.brain_observatory.behavior` (module), 112  
`allensdk.brain_observatory.behavior.behavior_data_set` (module), 89  
`allensdk.brain_observatory.behavior.behavior_ophys` (module), 91

allensdk.brain\_observatory.behavior.behavioral\_hybrary.brain\_observatory.brain\_observatory\_plots  
(module), 81 (module), 185

allensdk.brain\_observatory.behavior.behavioral\_hybrary.behavioral\_hybrary\_categorical  
(module), 80 (module), 186

allensdk.brain\_observatory.behavior.behavioral\_hybrary.brain\_observatory.circle\_plots  
(module), 92 (module), 186

allensdk.brain\_observatory.behavior.behavioral\_hybrary.brain\_observatory.demixer (mod-  
(module), 95 ular), 188

allensdk.brain\_observatory.behavior.behavioral\_hybrary.brain\_observatory.dff (module),  
(module), 97 189

allensdk.brain\_observatory.behavior.criteria.allensdk.brain\_observatory.drifting\_gratings  
(module), 98 (module), 191

allensdk.brain\_observatory.behavior.dprime.allensdk.brain\_observatory.ecephys (mod-  
(module), 100 ular), 170

allensdk.brain\_observatory.behavior.eye\_tracking.allensdk.brain\_observatory.ecephys.align\_timestamps  
(module), 101 (module), 116

allensdk.brain\_observatory.behavior.image\_analysis.allensdk.brain\_observatory.ecephys.align\_timestamps  
(module), 104 (module), 112

allensdk.brain\_observatory.behavior.interaction.allensdk.brain\_observatory.ecephys.align\_timestamps  
(module), 85 (module), 114

allensdk.brain\_observatory.behavior.interaction.allensdk.brain\_observatory.ecephys.align\_timestamps  
(module), 82 (module), 115

allensdk.brain\_observatory.behavior.interaction.allensdk.brain\_observatory.ecephys.align\_timestamps  
(module), 83 (module), 115

allensdk.brain\_observatory.behavior.interaction.allensdk.brain\_observatory.ecephys.copy\_utility  
(module), 85 (module), 116

allensdk.brain\_observatory.behavior.metadata.allensdk.brain\_observatory.ecephys.current\_source\_  
(module), 104 (module), 116

allensdk.brain\_observatory.behavior.mtrails.allensdk.brain\_observatory.ecephys.ecephys\_project\_  
(module), 104 (module), 125

allensdk.brain\_observatory.behavior.reward\_allensdk.brain\_observatory.ecephys.ecephys\_project\_  
(module), 106 (module), 117

allensdk.brain\_observatory.behavior.running\_allensdk.brain\_observatory.ecephys.ecephys\_project\_  
(module), 106 (module), 117

allensdk.brain\_observatory.behavior.schemas.allensdk.brain\_observatory.ecephys.ecephys\_project\_  
(module), 106 (module), 118

allensdk.brain\_observatory.behavior.sessions.allensdk.brain\_observatory.ecephys.ecephys\_project\_  
(module), 107 (module), 121

allensdk.brain\_observatory.behavior.stimulus.allensdk.brain\_observatory.ecephys.ecephys\_project\_  
(module), 108 (module), 122

allensdk.brain\_observatory.behavior.sync\_allensdk.brain\_observatory.ecephys.ecephys\_project\_  
(module), 86 (module), 124

allensdk.brain\_observatory.behavior.sync\_allensdk.brain\_observatory.ecephys.ecephys\_project\_  
(module), 85 (module), 125

allensdk.brain\_observatory.behavior.trials\_allensdk.brain\_observatory.ecephys.ecephys\_project\_  
(module), 108 (module), 116

allensdk.brain\_observatory.behavior.trials\_allensdk.brain\_observatory.ecephys.ecephys\_project\_  
(module), 109 (module), 158

allensdk.brain\_observatory.behavior.validation.allensdk.brain\_observatory.ecephys.ecephys\_session\_  
(module), 111 (module), 162

allensdk.brain\_observatory.behavior.write\_allensdk.brain\_observatory.ecephys.ecephys\_session\_  
(module), 89 (module), 128

allensdk.brain\_observatory.brain\_observatory.allensdk.brain\_observatory.ecephys.ecephys\_session\_  
(module), 185 (module), 125



allensdk.brain\_observatory.running\_speed (module), 201  
 allensdk.brain\_observatory.session\_analysis (module), 202  
 allensdk.brain\_observatory.session\_api\_utils (module), 204  
 allensdk.brain\_observatory.static\_gratings (module), 205  
 allensdk.brain\_observatory.stimulus\_analysis (module), 207  
 allensdk.brain\_observatory.stimulus\_info (module), 208  
 allensdk.brain\_observatory.sync\_dataset (module), 212  
 allensdk.brain\_observatory.sync\_utilities (module), 183  
 allensdk.brain\_observatory.visualization (module), 183  
 allensdk.config (module), 225  
 allensdk.config.app (module), 219  
 allensdk.config.app.application\_config (module), 217  
 allensdk.config.manifest (module), 222  
 allensdk.config.manifest\_builder (module), 224  
 allensdk.config.model (module), 222  
 allensdk.config.model.description (module), 221  
 allensdk.config.model.description\_parser (module), 222  
 allensdk.config.model.formats (module), 221  
 allensdk.config.model.formats.hdf5\_utils (module), 219  
 allensdk.config.model.formats.json\_description\_parser (module), 219  
 allensdk.config.model.formats.pycfg\_description\_parser (module), 220  
 allensdk.core (module), 264  
 allensdk.core.auth\_config (module), 225  
 allensdk.core.authentication (module), 226  
 allensdk.core.brain\_observatory\_cache (module), 227  
 allensdk.core.brain\_observatory\_nwb\_data\_set (module), 231  
 allensdk.core.cache\_method\_utilities (module), 235  
 allensdk.core.cell\_types\_cache (module), 235  
 allensdk.core.dat\_utilities (module), 238  
 allensdk.core.exceptions (module), 238  
 allensdk.core.h5\_utilities (module), 239  
 allensdk.core.json\_utilities (module), 239  
 allensdk.core.lazy\_property (module), 225  
 allensdk.core.lazy\_property.lazy\_property (module), 225  
 allensdk.core.lazy\_property.lazy\_property\_mixin (module), 225  
 allensdk.core.mouse\_connectivity\_cache (module), 240  
 allensdk.core.nwb\_data\_set (module), 245  
 allensdk.core.obj\_utilities (module), 247  
 allensdk.core.ontology (module), 248  
 allensdk.core.ophys\_experiment\_session\_id\_mapping (module), 249  
 allensdk.core.reference\_space (module), 249  
 allensdk.core.reference\_space\_cache (module), 251  
 allensdk.core.simple\_tree (module), 253  
 allensdk.core.sitk\_utilities (module), 256  
 allensdk.core.structure\_tree (module), 257  
 allensdk.core.swc (module), 260  
 allensdk.core.typing (module), 264  
 allensdk.deprecated (module), 374  
 allensdk.ephys (module), 276  
 allensdk.ephys.ephys\_extractor (module), 264  
 allensdk.ephys.ephys\_features (module), 268  
 allensdk.ephys.extract\_cell\_features (module), 275  
 allensdk.ephys.feature\_extractor (module), 276  
 allensdk.internal (module), 352  
 allensdk.internal.api (module), 287  
 allensdk.internal.api.api\_prerelease (module), 282  
 allensdk.internal.api.behavior\_data\_lims\_api (module), 282  
 allensdk.internal.api.behavior\_lims\_api (module), 284  
 allensdk.internal.api.behavior\_ophys\_api (module), 284  
 allensdk.internal.api.lims\_api (module), 285  
 allensdk.internal.api.mtrain\_api (module), 286  
 allensdk.internal.api.ophys\_lims\_api (module), 286  
 allensdk.internal.api.queries (module), 281  
 allensdk.internal.api.queries.biophysical\_module\_api (module), 277  
 allensdk.internal.api.queries.biophysical\_module\_release (module), 277  
 allensdk.internal.api.queries.grid\_data\_api\_prerelease (module), 278



```

allensdk.internal.api.queries.mouse_connectivity_pre_release
    (module), 279
allensdk.internal.api.queries.optimize_config_report
    (module), 280
allensdk.internal.api.queries.pre_release
    (module), 281
allensdk.internal.brain_observatory
    (module), 301
allensdk.internal.brain_observatory.annotated_reports_matrices
    (module), 288
allensdk.internal.brain_observatory.demix_report
    (module), 289
allensdk.internal.brain_observatory.demixer
    (module), 290
allensdk.internal.brain_observatory.eye_calibration
    (module), 291
allensdk.internal.brain_observatory.fit_ellipse
    (module), 294
allensdk.internal.brain_observatory.frame_stream
    (module), 294
allensdk.internal.brain_observatory.itracker_utils
    (module), 295
allensdk.internal.brain_observatory.mask_set
    (module), 296
allensdk.internal.brain_observatory.ophys_session_composition
    (module), 297
allensdk.internal.brain_observatory.resources
    (module), 288
allensdk.internal.brain_observatory.roi_filter
    (module), 298
allensdk.internal.brain_observatory.time_sync
    (module), 300
allensdk.internal.core (module), 305
allensdk.internal.core.lims_pipeline_model
    (module), 301
allensdk.internal.core.lims_utilities
    (module), 302
allensdk.internal.core.mouse_connectivity_analysis_pre_release
    (module), 303
allensdk.internal.core.simpletree (module), 304
allensdk.internal.core.swc (module), 304
allensdk.internal.ephys (module), 310
allensdk.internal.ephys.core_feature_extract
    (module), 305
allensdk.internal.ephys.plot_qc_figures
    (module), 306
allensdk.internal.ephys.plot_qc_figures3
    (module), 308
allensdk.internal.model (module), 327
allensdk.internal.model.AIC (module), 326
allensdk.internal.model.biophysical
    (module), 315
allensdk.internal.model.biophysical.biophysical
    (module), 318
allensdk.internal.model.biophysical.check_fit_shift
    (module), 312
allensdk.internal.model.biophysical.deap_utils
    (module), 313
allensdk.internal.model.biophysical.ephys_utils
    (module), 313
allensdk.internal.model.biophysical.fits
    (module), 311
allensdk.internal.model.biophysical.fits.fit_styles
    (module), 311
allensdk.internal.model.biophysical.make_deap_fit
    (module), 313
allensdk.internal.model.biophysical.passive_fitting
    (module), 312
allensdk.internal.model.biophysical.passive_fitting
    (module), 311
allensdk.internal.model.biophysical.passive_fitting
    (module), 311
allensdk.internal.model.biophysical.passive_fitting
    (module), 311
allensdk.internal.model.biophysical.passive_fitting
    (module), 311
allensdk.internal.model.biophysical.passive_fitting
    (module), 311
allensdk.internal.model.biophysical.passive_fitting
    (module), 311
allensdk.internal.model.biophysical.run_optimize
    (module), 314
allensdk.internal.model.biophysical.run_optimize_w
    (module), 314
allensdk.internal.model.biophysical.run_passive_fit
    (module), 314
allensdk.internal.model.biophysical.run_simulate_l
    (module), 314
allensdk.internal.model.biophysical.run_simulate_w
    (module), 315
allensdk.internal.model.data_access
    (module), 327
allensdk.internal.model.glif (module), 326
allensdk.internal.model.glif.are_two_lists_of_array
    (module), 315
allensdk.internal.model.glif.error_functions
    (module), 315
allensdk.internal.model.glif.find_spikes
    (module), 316
allensdk.internal.model.glif.find_sweeps
    (module), 316
allensdk.internal.model.glif.glif_experiment
    (module), 317
allensdk.internal.model.glif.glif_optimizer
    (module), 318

```

<b>Index</b>	<b>403</b>
--------------	------------

[ule](#)), 354  
[allensdk.model.glif \(module\)](#), 365  
[allensdk.model.glif.glif\\_neuron \(module\)](#), 356  
[allensdk.model.glif.glif\\_neuron\\_methods \(module\)](#), 359  
[allensdk.model.glif.simulate\\_neuron \(module\)](#), 365  
[allensdk.morphology \(module\)](#), 366  
[allensdk.morphology.validate\\_swc \(module\)](#), 365  
[allensdk.mouse\\_connectivity \(module\)](#), 373  
[allensdk.mouse\\_connectivity.grid \(module\)](#), 373  
[allensdk.mouse\\_connectivity.grid.image\\_series\\_generator \(module\)](#), 372  
[allensdk.mouse\\_connectivity.grid.subimage \(module\)](#), 369  
[allensdk.mouse\\_connectivity.grid.subimage\\_base\\_subimage \(module\)](#), 366  
[allensdk.mouse\\_connectivity.grid.subimage.cav\\_subimage \(module\)](#), 368  
[allensdk.mouse\\_connectivity.grid.subimage.class\\_subimage \(module\)](#), 368  
[allensdk.mouse\\_connectivity.grid.subimage.annotated\\_subimage \(module\)](#), 369  
[allensdk.mouse\\_connectivity.grid.utilities \(module\)](#), 372  
[allensdk.mouse\\_connectivity.grid.utilities.download\\_reference\\_space\\_cache \(module\)](#), 370  
[allensdk.mouse\\_connectivity.grid.utilities.Api \(class in allensdk.api\)](#), 370  
[allensdk.mouse\\_connectivity.grid.writers \(module\)](#), 372  
[allensdk.test\\_utilities \(module\)](#), 374  
[allensdk.test\\_utilities.custom\\_comparators \(module\)](#), 373  
[allensdk.test\\_utilities.regression\\_fixture \(module\)](#), 373  
[allensdk.test\\_utilities.temp\\_dir \(module\)](#), 373  
[allocate\\_by\\_vsync\(\) \(in module allensdk.brain\\_observatory.ecephys.stimulus\\_sync\)](#), 169  
[alpha\\_filter\(\) \(in module allensdk.brain\\_observatory.r\\_neuropil\)](#), 198  
[analog\\_meta\\_data \(allensdk.brain\\_observatory.sync\\_dataset.Dataset attribute\)](#), 213  
[ANALYSIS\\_DATA\\_KEY \(allensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache attribute\)](#), 227  
[analyze\\_trough\\_details\(\) \(in module allensdk.ephys.ephys\\_features\)](#), 268  
[ancestor\\_ids\(\) \(allensdk.core.simple\\_tree.SimpleTree method\)](#), 253  
[ancestor\\_ids\(\) \(allensdk.internal.core.simpletree.SimpleTree method\)](#), 304  
[ancestors\(\) \(allensdk.core.simple\\_tree.SimpleTree method\)](#), 254  
[ancestors\(\) \(allensdk.internal.core.simpletree.SimpleTree method\)](#), 304  
[angle\\_lines\(\) \(in module allensdk.brain\\_observatory.circle\\_plots\)](#), 188  
[angular\\_wheel\\_rotation \(allensdk.brain\\_observatory.ecephys.file\\_io.stim\\_file.CamStimOneP attribute\)](#), 129  
[angular\\_wheel\\_velocity \(allensdk.brain\\_observatory.ecephys.file\\_io.stim\\_file.CamStimOneP attribute\)](#), 129  
[an\\_image\\_base\\_subimage\\_detect\(\) \(in module allensdk.brain\\_observatory.behavior.mtrain\)](#), 105  
[annotate\\_trials\(\) \(in module allensdk.brain\\_observatory.behavior.mtrain\)](#), 105  
[AnnotatedSectionDataSetsApi \(class in allensdk.api.queries.annotated\\_section\\_data\\_sets\\_api\)](#), 43  
[ANNOTATION\\_KEY \(allensdk.core.reference\\_space\\_cache.ReferenceSpaceCache attribute\)](#), 251  
[Api \(class in allensdk.api\)](#), 72  
[APICAL\\_DENDRITE \(allensdk.core.swc.Morphology attribute\)](#), 260  
[APICAL\\_DENDRITE \(allensdk.internal.morphology.morphology.Morphology attribute\)](#), 328  
[ApiPrerelease \(class in allensdk.internal.api.api\\_prerelease\)](#), 282  
[append\(\) \(allensdk.core.swc.Morphology method\)](#), 260  
[append\(\) \(allensdk.internal.morphology.morphology.Morphology method\)](#), 328  
[append\(\) \(allensdk.internal.mouse\\_connectivity.projection\\_thumbnail.image\\_projection\\_thumbnail method\)](#), 340  
[append\\_experiment\\_metrics\(\) \(allensdk.brain\\_observatory.session\\_analysis.SessionAnalysis method\)](#), 202  
[append\\_metadata\(\) \(allensdk.brain\\_observatory.session\\_analysis.SessionAnalysis method\)](#), 202  
[append\\_metrics\\_drifting\\_grating\(\) \(allensdk.brain\\_observatory.session\\_analysis.SessionAnalysis method\)](#), 202  
[append\\_metrics\\_locally\\_sparse\\_noise\(\)](#)



(allensdk.brain\_observatory.session\_analysis.SessionAnalysis), 149  
 method), 202      apply\_divisions() (in module al-  
 lensdk.brain\_observatory.session\_analysis.SessionAnalysis), 370  
 method), 202      apply\_frame\_times() (in module al-  
 lensdk.brain\_observatory.ecephys.stimulus\_table.ephys\_pre\_spikes), 150  
 (allensdk.brain\_observatory.session\_analysis.SessionAnalysis), 149  
 method), 202      apply\_mask() (allensdk.mouse\_connectivity.grid.subimage.base\_subimage.SubImage), 367  
 lensdk.brain\_observatory.session\_analysis.SessionAnalysis), 367  
 method), 202      apply\_pixel\_counter() (al-  
 lensdk.mouse\_connectivity.grid.subimage.base\_subimage.SubImage), 367  
 method), 202      apply\_ssl(allensdk.api.queries.reference\_space\_api.ReferenceSpaceApi), 63  
 attribute), 63  
 method), 202      archive\_cell() (al-  
 lensdk.brain\_observatory.session\_analysis.SessionAnalysis), 312  
 method), 202      are\_two\_lists\_of\_arrays\_the\_same() (in module al-  
 lensdk.internal.model.glif.glif\_neuron.GlifNeuron), 357  
 method), 357      are\_two\_lists\_of\_arrays\_the\_same(), 315  
 append\_well\_known\_file() (in module al-  
 lensdk.internal.core.lims\_utilities), 302      arg\_parser() (in module al-  
 lensdk.internal.model.biophysical.passive\_fitting.neuron\_passive\_fitting), 311  
 ApplicationConfig (class in al-  
 lensdk.config.app.application\_config), 217  
 apply() (allensdk.internal.mouse\_connectivity.projection\_thresholding\_utils.algebraic\_pipeline\_module.PipelineModule), 340  
 method), 340      attribute), 301  
 apply\_affine() (allensdk.core.swc.Morphology), 260  
 method), 260      ArgSchemaParserPlus (class in al-  
 lensdk.brain\_observatory.argschema\_utilities), 184  
 apply\_affine() (al-  
 lensdk.internal.morphology.morphology.Morphology), 328  
 method), 328      ARRAY (allensdk.api.queries.connected\_services.ConnectedServices), 51  
 attribute), 51  
 apply\_affine\_only\_rotation() (al-  
 lensdk.internal.morphology.morphology.Morphology), 328  
 method), 328      array\_intervals() (in module al-  
 lensdk.brain\_observatory.ecephys.ecephys\_session), 168  
 apply\_average\_tile() (al-  
 lensdk.internal.mouse\_connectivity.tissuecyte\_stitching.tile.Tile), 223  
 method), 223      as\_dataframe() (allensdk.config.manifest.Manifest), 223  
 method), 343      as\_dataframe() (al-  
 lensdk.config.manifest\_builder.ManifestBuilder), 224  
 method), 224      as\_dict() (allensdk.ephys.ephys\_extractor.EphysCellFeatureExtractor), 264  
 method), 343      as\_dict() (allensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor), 265  
 apply\_colormap() (in module al-  
 lensdk.internal.mouse\_connectivity.projection\_thresholding\_utils.algebraic\_pipeline\_module.PipelineModule), 339  
 method), 265  
 apply\_configuration\_from\_command\_line() aspect\_ratio(allensdk.brain\_observatory.stimulus\_info.Monitor), 209  
 (allensdk.config.app.application\_config.ApplicationConfig), 217  
 method), 217      assert\_exists() (in module al-  
 lensdk.internal.pipeline\_modules.run\_demixing), 345  
 method), 217      assign\_session\_id() (in module al-  
 lensdk.brain\_observatory.behavior.mtrain), 105  
 lensdk.config.app.application\_config.ApplicationConfig), 105  
 method), 217      assign\_sweep\_values() (in module al-  
 lensdk.brain\_observatory.ecephys.stimulus\_table.ephys\_pre\_spikes), 150  
 apply\_display\_sequence() (in module al-  
 lensdk.brain\_observatory.ecephys.stimulus\_table.ephys\_pre\_spikes), 150

`assign_to_last()` (in module `al- behavior_alignment` (al-  
`lensdk.brain_observatory.ecephys.stimulus_sync`), `lensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSync`  
169 attribute), 349

`AsyncHttpEngine` (class in `al- BEHAVIOR_ANALYSIS_LOG_KEY` (al-  
`lensdk.brain_observatory.ecephys.ecephys_project_api.http_lensdk.brain_observatory.behavior.behavior_project_cache.Behav`  
122 attribute), 95

`AsyncRmaEngine` (class in `al- behavior_delta` (al-  
`lensdk.brain_observatory.ecephys.ecephys_project_api.rma_lensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSync`  
124 attribute), 349

`atlas_image_query()` (al- `behavior_session_id` (al-  
`lensdk.api.queries.image_download_api.ImageDownloadApi`, `lensdk.brain_observatory.behavior.behavior_data_session.Behav`  
method), 54 attribute), 89

`autocorr()` (in module `al- behavior_session_id_to_foraging_id()`  
`lensdk.internal.model.glif.MLIN`), 315 (allensdk.internal.api.behavior\_lims\_api.BehaviorLimsApi

`average_projection` (al- static method), 284  
`lensdk.brain_observatory.behavior.behavior_ophys_session_id_to_ophys_session` (al-  
attribute), 92 `lensdk.brain_observatory.behavior.behavior_project_cache.Behav`

`average_rate()` (in module `al- attribute`), 95  
`lensdk.ephys.ephys_features`), 269

`AVERAGE_TEMPLATE` (al- `lensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSync`  
`lensdk.api.queries.reference_space_api.ReferenceSpaceApi` attribute), 349  
attribute), 63

`average_tile_is_untrimmed()` (al- `lensdk.brain_observatory.sync_dataset.Dataset`  
`lensdk.internal.mouse_connectivity.tissuecyte_stitching.tile_tile` attribute), 213  
method), 343

`average_voltage()` (in module `al- lensdk.internal.brain_observatory.time_sync.OphysTimeAligner`  
`lensdk.ephys.ephys_features`), 269 attribute), 300

`AXON` (allensdk.core.swc.Morphology attribute), 260 `BehaviorBase` (class in al-  
`AXON` (allensdk.internal.morphology.morphology.Morphology `lensdk.brain_observatory.behavior.internal.behavior_base`),  
attribute), 328 82

`azimuths` (allensdk.brain\_observatory.ecephys.stimulus\_sync\_behavioral\_fields\_mapping.adaptive\_field\_mapping  
attribute), 139 `lensdk.internal.api.behavior_data_lims_api`),  
282

**B** `BehaviorDataSession` (class in al-  
`lensdk.brain_observatory.behavior.behavior_data_session`),  
background\_trace() (in module `al- 89`  
`lensdk.internal.brain_observatory.demix_report`), 289

`barcode_line` (allensdk.brain\_observatory.ecephys.align\_timestamps\_barcode\_sync\_dataset\_barcode\_sync\_dataset  
attribute), 114 `BehaviorLimsApi` (class in al-  
284

`BarcodeSyncDataset` (class in al- `lensdk.brain_observatory.behavior.behavior_ophys_analysis`),  
`lensdk.brain_observatory.ecephys.align_timestamps.barcode_sync_dataset`,  
114 `BehaviorOphysAnalysis` (class in al-  
91

`BASAL_DENDRITE` (allensdk.core.swc.Morphology at- `lensdk.brain_observatory.behavior.behavior_ophys_api`),  
tribute), 260 81

`BASAL_DENDRITE` (al- `BehaviorOphysBase` (class in al-  
`lensdk.internal.morphology.morphology.Morphology` `lensdk.brain_observatory.behavior.internal.behavior_ophys_base`  
attribute), 328 83

`base_object_to_eye_rotation_matrix()` `BehaviorOphysLimsApi` (class in al-  
(in module `al- lensdk.internal.api.behavior_ophys_api`),  
`lensdk.internal.brain_observatory.eye_calibration`), 284  
292

`bb_dist()` (in module `al- lensdk.brain_observatory.behavior.behavior_ophys_api.behavior`  
`lensdk.internal.brain_observatory.mask_set`), 80  
297 `BehaviorOphysSession` (class in al-

[lensdk.brain\\_observatory.behavior.behavior\\_ophys\\_session.apply\(\)](#) (in module [al-lensdk.mouse\\_connectivity.grid.utilities.image\\_utilities](#)),  
 92  
[BehaviorProjectBase](#) (class in [al-lensdk.brain\\_observatory.behavior.internal.behavior\\_project\\_base](#)),  
 85  
[BehaviorProjectCache](#) (class in [al-lensdk.brain\\_observatory.behavior.behavior\\_project\\_cache](#)),  
 95  
[BehaviorProjectLimsApi](#) (class in [al-BrainObservatoryAnalysisException](#),  
 97  
[lensdk.brain\\_observatory.behavior.behavior\\_project\\_lims\\_api](#)),  
[best\\_fit\\_value\(\)](#) (al-[lensdk.brain\\_observatory.stimulus\\_analysis.stimulus\\_analysis](#)),  
 46  
[lensdk.internal.model.biophysical.make\\_deep\\_fit\\_from\\_report](#) (class in [al-lensdk.internal.api.queries.pre\\_release](#)),  
 281  
[BIC\(\)](#) (in module [allensdk.internal.model.AIC](#)), 326  
[BrainObservatoryCache](#) (class in [al-lensdk.brain\\_observatory.cache](#)), 227  
[binarize\(\)](#) ([allensdk.mouse\\_connectivity.grid.subimage.base\\_subimage.subimage](#)), 367  
[BrainObservatoryMonitor](#) (class in [al-lensdk.brain\\_observatory.stimulus\\_info](#)),  
 208  
[BinaryIntervalSearchTree](#) (class in [al-lensdk.brain\\_observatory.stimulus\\_info](#)),  
 208  
[BrainObservatoryNwbDataSet](#) (class in [al-lensdk.core.brain\\_observatory\\_nwb\\_data\\_set](#)),  
 231  
[binned\\_cells\\_sp](#) (al-[lensdk.brain\\_observatory.stimulus\\_analysis.stimulus\\_analysis](#)),  
 207  
[binned\\_cells\\_vis](#) (al-[lensdk.brain\\_observatory.stimulus\\_analysis.stimulus\\_analysis](#)),  
 207  
[binned\\_dx\\_sp](#) ([allensdk.brain\\_observatory.stimulus\\_analysis.stimulus\\_analysis](#)),  
 207  
[binned\\_dx\\_vis](#) (al-[lensdk.brain\\_observatory.stimulus\\_analysis.stimulus\\_analysis](#)),  
 207  
[BIOPHYSICAL\\_MODEL\\_TYPE\\_IDS](#) (al-[lensdk.api.queries.biophysical\\_api.BiophysicalApi](#)),  
 44  
[BiophysicalApi](#) (class in [al-lensdk.api.queries.biophysical\\_api](#)), 44  
[BiophysicalArchiver](#) (class in [al-lensdk.internal.model.biophysical.biophysical\\_archiver](#)),  
 312  
[BiophysicalModuleApi](#) (class in [al-lensdk.internal.api.queries.biophysical\\_module\\_api](#)),  
 277  
[BiophysicalModuleReader](#) (class in [al-lensdk.internal.api.queries.biophysical\\_module\\_reader](#)),  
 277  
[blend\(\)](#) (in module [al-lensdk.internal.mouse\\_connectivity.projection\\_thumbnail\\_visualization\\_utilities](#)),  
 341  
[blend\\_component\\_from\\_point\(\)](#) (in module [al-lensdk.internal.mouse\\_connectivity.tissuecyte\\_stitching\\_stitcher](#)),  
 342  
[blend\\_with\\_background\(\)](#) (in module [al-lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
 339  
[build\\_affine\\_transform\(\)](#) (in module [al-lensdk.mouse\\_connectivity.grid.utilities.image\\_utilities](#)),  
 370  
[build\\_and\\_execute\(\)](#) (in module [al-lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_api.utilities](#)),  
 125  
[build\\_cell\\_plots\(\)](#) (in module [al-lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
 346  
[build\\_coarse\\_grids\(\)](#) (al-[lensdk.mouse\\_connectivity.grid.image\\_series\\_gridder.ImageSeriesGridder](#)),  
 372  
[build\\_colormap\(\)](#) (in module [al-lensdk.brain\\_observatory.ecephys.stimulus\\_table.visualization.visualization\\_utilities](#)),  
 148  
[build\\_composite\\_transform\(\)](#) (in module [al-lensdk.mouse\\_connectivity.grid.utilities.image\\_utilities](#)),  
 370  
[build\\_correlation\\_plots\(\)](#) (in module [al-lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
 346  
[build\\_drifting\\_gratings\(\)](#) (in module [al-lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
 346  
[blend\\_visualization\\_utilities](#) (in module [al-lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_api.utilities](#)),  
 125  
[build\\_experiment\\_thumbnails\(\)](#) (in module [al-lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
 346  
[generate\\_projection\\_strip\(\)](#) (in module [al-lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
 339  
[build\\_eye\\_tracking\\_plots\(\)](#) (in module [al-lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
 339

[lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnail](#),  
[347](#) [build\\_static\\_gratings\(\)](#) (in module [al-](#)  
[lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
[lensdk.internal.mouse\\_connectivity.projection\\_thumbnail.image\\_sheet.ImageSheet](#)  
[static method](#)), [340](#) [build\\_stimuluswise\\_table\(\)](#) (in module [al-](#)  
[lensdk.brain\\_observatory.ecephys.stimulus\\_table.ecephys\\_pre\\_spike](#)  
[lensdk.brain\\_observatory.circle\\_plots](#)), [188](#) [150](#)  
[build\\_locally\\_sparse\\_noise\(\)](#) (in module [al-](#) [build\\_time\\_window\\_domain\(\)](#) (in module [al-](#)  
[lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)), [lensdk.brain\\_observatory.ecephys.ecephys\\_session](#)),  
[347](#) [168](#)  
[build\\_manifest\(\)](#) ([allensdk.api.cache.Cache](#) [build\\_trial\\_matrix\(\)](#) (in module [al-](#)  
[method](#)), [76](#) [lensdk.brain\\_observatory.receptive\\_field\\_analysis.chisquarperf](#)),  
[build\\_manifest\(\)](#) (al- [175](#)  
[lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#) [build\\_volumetric\\_data\\_download\\_url\(\)](#) (in module [al-](#)  
[method](#)), [227](#) [lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
[build\\_manifest\(\)](#) (al- [348](#)  
[lensdk.core.cell\\_types\\_cache.CellTypesCache](#) [build\\_url\(\)](#) ([allensdk.api.queries.connected\\_services.ConnectedService](#)  
[method](#)), [236](#) [method](#)), [51](#)  
[build\\_manifest\(\)](#) (al- [build\\_volumetric\\_data\\_download\\_url\(\)](#)  
[lensdk.internal.api.queries.optimize\\_config\\_reader.OptimizeConfigReader](#) [allensdk.api.queries.reference\\_space\\_api.ReferenceSpaceApi](#)  
[method](#)), [280](#) [method](#)), [63](#)  
[build\\_natural\\_movie\(\)](#) (in module [al-](#) [burst\\_metrics\(\)](#) (al-  
[lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)), [lensdk.ecephys.ecephys\\_extractor.EcephysSweepFeatureExtractor](#)  
[347](#) [method](#)), [265](#)  
[build\\_natural\\_scenes\(\)](#) (in module [al-](#)  
[lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
[347](#) [c50\(\)](#) (in module [al-](#)  
[lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)),  
[347](#) [lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.drifting\\_gratings](#)  
[Cache](#) (class in [allensdk.api.cache](#)), [76](#)  
[build\\_query\(\)](#) ([allensdk.api.queries.svg\\_api.SvgApi](#) [cache\\_clear\(\)](#) (al-  
[method](#)), [70](#) [lensdk.brain\\_observatory.behavior.behavior\\_data\\_session.BehaviorDataSession](#)  
[build\\_query\\_url\(\)](#) (al- [method](#)), [89](#)  
[lensdk.api.queries.rma\\_api.RmaApi](#) [method](#)), [cache\\_clear\(\)](#) (al-  
[65](#) [lensdk.brain\\_observatory.behavior.behavior\\_ophys\\_session.BehaviorOphysSession](#)  
[build\\_receptive\\_field\(\)](#) (in module [al-](#) [method](#)), [92](#)  
[lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)), [cache\\_clear\(\)](#) (al-  
[347](#) [lensdk.core.cache\\_method\\_utilities.CachedInstanceMethodMixin](#)  
[build\\_reference\\_aligned\\_image\\_channel\\_volumes\\_url\(\)](#) [method](#)), [235](#)  
[\(allensdk.api.queries.mouse\\_connectivity\\_api.MouseConnectivityApi](#) (allensdk.api.cache.Cache [static](#)  
[method](#)), [58](#) [method](#)), [76](#)  
[build\\_rma\(\)](#) ([allensdk.api.queries.biophysical\\_api.BiophysicalApi](#) [cache\\_csv\\_dataframe\(\)](#) (al-  
[method](#)), [44](#) [lensdk.api.cache.Cache](#) [static method](#)), [76](#)  
[build\\_rotation\\_transform\(\)](#) (al- [cache\\_csv\\_json\(\)](#) ([allensdk.api.cache.Cache](#) [static](#)  
[lensdk.internal.mouse\\_connectivity.projection\\_thumbnail.volume\\_projector.VolumeProjector](#)  
[method](#)), [341](#) [method](#)), [76](#)  
[build\\_schema\\_query\(\)](#) (al- [cache\\_data\(\)](#) ([allensdk.api.queries.biophysical\\_api.BiophysicalApi](#)  
[lensdk.api.queries.rma\\_api.RmaApi](#) [method](#)), [method](#)), [45](#)  
[66](#) [cache\\_json\(\)](#) ([allensdk.api.cache.Cache](#) [static](#)  
[method](#)), [76](#)  
[build\\_speed\\_tuning\(\)](#) (in module [al-](#) [cache\\_json\\_dataframe\(\)](#) (al-  
[lensdk.internal.pipeline\\_modules.run\\_observatory\\_thumbnails](#)), [lensdk.api.cache.Cache](#) [static method](#)), [76](#)  
[347](#) [cache\\_stimulus\\_file\(\)](#) (al-  
[build\\_spike\\_histogram\(\)](#) (in module [al-](#) [lensdk.api.queries.glif\\_api.GlifApi](#) [method](#)),  
[lensdk.brain\\_observatory.ecephys.ecephys\\_session](#)), [52](#)



`cacheable()` (in module `allensdk.api.cache`), 78  
`CachedInstanceMethodMixin` (class in `allensdk.core.cache_method_utilities`), 235  
`cached()` (`allensdk.api.cache.Cache` static method), 76  
`calc_deriv()` (in module `allensdk.brain_observatory.behavior.running_processing`), 106  
`calc_spike_component_of_threshold_from_masking()` (in module `allensdk.internal.model.glif.threshold_adaptation`), 323  
`calc_spike_cut_and_v_reset_via_expvar_residuals()` (in module `allensdk.internal.model.glif.spike_cutting`), 323  
`calculate()` (`allensdk.core.lazy_property.lazy_property.LazyProperty` method), 225  
`calculate()` (`allensdk.internal.mouse_connectivity.interval_union_intersection_utilities.reference_space_api.ReferenceSpaceApi` method), 338  
`calculate()` (`allensdk.internal.mouse_connectivity.interval_union_intersection_utilities.reference_space_api.ReferenceSpaceApi` method), 338  
`calculate()` (`allensdk.internal.mouse_connectivity.interval_union_intersection_utilities.reference_space_api.ReferenceSpaceApi` method), 339  
`calculate_delay()` (in module `allensdk.brain_observatory.behavior.sync.process_sync`), 85  
`calculate_dff()` (in module `allensdk.brain_observatory.dff`), 189  
`calculate_dvdt()` (in module `allensdk.ephys.ephys_features`), 270  
`calculate_feature_errors()` (`allensdk.internal.model.biophysical.deap_utils.Utils` method), 313  
`calculate_fi_curves()` (in module `allensdk.internal.model.biophysical.check_fi_shift`), 312  
`calculate_injection_centroid()` (`allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi` method), 58  
`calculate_max_border()` (in module `allensdk.internal.brain_observatory.roi_filter_utils`), 299  
`calculate_reward_rate()` (in module `allensdk.brain_observatory.behavior.trials_processing`), 109  
`calculate_roi_and_neuropil_traces()` (in module `allensdk.brain_observatory.roi_masks`), 200  
`calculate_scale()` (in module `allensdk.internal.morphology.morphvis`), 332  
`calculate_time_delayed_correlation()` (in module `allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis_utilities`), 145  
`calculate_traces()` (in module `allensdk.brain_observatory.roi_masks`), 200  
`calculate_trough()` (`allensdk.ephys.feature_extractor.EphysFeatureExtractor` method), 276  
`caching()` (in module `allensdk.api.caching_utilities`), 79  
`CanslimOnePickleStimFile` (class in `allensdk.brain_observatory.ecephys.file_io.stim_file`), 129  
`categorize_one_trial()` (in module `allensdk.brain_observatory.behavior.trials_processing`), 109  
`cav_writer()` (in module `allensdk.mouse_connectivity.grid.writers`), 373  
`CavSubImage` (class in `allensdk.mouse_connectivity.grid.subimage_projection_utilities`), 368  
`CCF_2017` (`allensdk.api.queries.reference_space_api.ReferenceSpaceApi` attribute), 63  
`CCF_VERSION_DEFAULT` (`allensdk.api.queries.reference_space_api.ReferenceSpaceApi` attribute), 63  
`cell_extractor_for_nwb()` (in module `allensdk.ephys.ephys_extractor`), 268  
`cell_features()` (`allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor` method), 264  
`cell_id` (`allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis` attribute), 207  
`cell_index_receptive_field_analysis_data` (`allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoiseApi` attribute), 193  
`CELL_MAPPING_ID` (`allensdk.api.queries.brain_observatory_api.BrainObservatoryApi` attribute), 46  
`cell_specimen_table` (`allensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession` attribute), 92  
`CELL_SPECIMENS_KEY` (`allensdk.core.brain_observatory_cache.BrainObservatoryCache` attribute), 227  
`CELLS_KEY` (`allensdk.core.cell_types_cache.CellTypesCache` attribute), 236  
`celltraces` (`allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis` attribute), 207  
`CellTypesApi` (class in `allensdk.api.queries.cell_types_api`), 49  
`CellTypesCache` (class in `allensdk.core.cell_types_cache`), 236

`lensdk.core.cell_types_cache)`, 235  
`change_parent()` (`allensdk.core.swc.Morphology` method), 261  
`change_parent()` (`allensdk.internal.morphology.morphology.Morphology` method), 328  
`channel_structure_intervals()` (`allensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` method), 164  
`CHANNELS_KEY` (`allensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` attribute), 158  
`check_and_write()` (`allensdk.core.reference_space.ReferenceSpace` static method), 249  
`check_coverage()` (`allensdk.core.reference_space.ReferenceSpace` method), 249  
`check_dir()` (`allensdk.config.manifest.Manifest` method), 223  
`check_org_selections_for_noise_block()` (`allensdk.internal.model.biophysical.make_deep_fit_report` method), 313  
`check_read_access()` (in module `allensdk.brain_observatory.argschema_utilities`), 185  
`check_stimulus_delay()` (in module `allensdk.internal.pipeline_modules.run_ophys_time_sync`), 350  
`check_thresholds_and_peaks()` (in module `allensdk.ephys.ephys_features`), 270  
`check_write_access()` (in module `allensdk.brain_observatory.argschema_utilities`), 185  
`check_write_access_dir()` (in module `allensdk.brain_observatory.argschema_utilities`), 185  
`check_write_access_overwrite()` (in module `allensdk.brain_observatory.argschema_utilities`), 185  
`checkPreprocess()` (in module `allensdk.internal.model.glif.plotting`), 322  
`checkSpikeCutting()` (in module `allensdk.internal.model.glif.plotting`), 322  
`chi_square_binary()` (in module `allensdk.brain_observatory.receptive_field_analysis.chisquare`), 175  
`chi_square_within_mask()` (in module `allensdk.brain_observatory.receptive_field_analysis.chisquare`), 175  
`child_ids()` (`allensdk.core.simple_tree.SimpleTree` method), 254  
`child_ids()` (`allensdk.internal.core.simpletree.SimpleTree` method), 304  
`children()` (`allensdk.core.simple_tree.SimpleTree` method), 254  
`children()` (`allensdk.internal.core.simpletree.SimpleTree` method), 304  
`children_of()` (`allensdk.core.swc.Morphology` method), 261  
`children_of()` (`allensdk.internal.morphology.morphology.Morphology` method), 329  
`CHISQ_CACHE` (`allensdk.brain_observatory.chisquare_categorical` module attribute), 186  
`choose_bps_command()` (in module `allensdk.model.biophys_sim.bps_command`), 352  
`choose_inliers()` (`allensdk.internal.brain_observatory.fit_ellipse.FitEllipse` method), 294  
`class_deprecated()` (in module `allensdk.deprecated`), 374  
`close_iter()` (in module `allensdk.mouse_connectivity.grid.writers`), 372  
`ClassicSubImage` (class in `allensdk.mouse_connectivity.grid.subimage.classic_subimage`), 368  
`clone_structures()` (`allensdk.core.structure_tree.StructureTree` static method), 257  
`cleanup_truncated_file()` (`allensdk.api.api.Api` method), 72  
`clear_updated_params()` (`allensdk.brain_observatory.session_api_utils.ParamsMixin` method), 205  
`clobbering_merge()` (in module `allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api`), 127  
`clone()` (`allensdk.ephys.feature_extractor.EphysFeatures` method), 276  
`clone()` (`allensdk.internal.morphology.morphology.Morphology` method), 329  
`close()` (`allensdk.brain_observatory.sync_dataset.Dataset` method), 213  
`close()` (`allensdk.internal.brain_observatory.frame_stream.CvInputStream` method), 294  
`close()` (`allensdk.internal.brain_observatory.frame_stream.FfmpegInputStream` method), 294  
`close()` (`allensdk.internal.brain_observatory.frame_stream.FfmpegOutputStream` method), 295  
`close()` (`allensdk.internal.brain_observatory.frame_stream.FrameInputStream` method), 295  
`close()` (`allensdk.internal.brain_observatory.frame_stream.FrameOutputStream` method), 295  
`close()` (`allensdk.internal.brain_observatory.mask_set.MaskSet` method), 295

`method`), 296  
`close_sets()` (`allensdk.internal.brain_observatory.mask_set.MaskSet` `method`), 296  
`coerce_scalar()` (in module `allensdk.brain_observatory.ecephys.ecephys_session`), 168  
`collapse_columns()` (in module `allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities`), 153  
`collect_sets()` (`allensdk.core.structure_tree.StructureTree` `static method`), 258  
`colormap()` (in module `allensdk.brain_observatory.behavior.trials_processing`), 109  
`COLORMAPS` (`allensdk.api.queries.image_download_api.ImageDownloadApi` `attribute`), 54  
`colors` (`allensdk.brain_observatory.ecephys.stimulus_analysis.flash_flashes` `attribute`), 136  
`compare_fields()` (in module `allensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_api`), 81  
`Compartment` (`class` in `allensdk.core.swc`), 260  
`Compartment` (`class` in `allensdk.internal.morphology.compartment`), 327  
`compartment()` (`allensdk.internal.morphology.morphology.Morphology` `method`), 329  
`compartment_index` (`allensdk.core.swc.Morphology` `attribute`), 261  
`compartment_index_by_type()` (`allensdk.core.swc.Morphology` `method`), 261  
`compartment_list` (`allensdk.core.swc.Morphology` `attribute`), 261  
`compartment_list` (`allensdk.internal.morphology.morphology.Morphology` `attribute`), 329  
`compartment_list_by_type()` (`allensdk.core.swc.Morphology` `method`), 261  
`compute()` (`allensdk.brain_observatory.ecephys.align_timestamps.probe_synchronizer.ProbeSynchronizer` `class method`), 115  
`compute_area()` (`allensdk.internal.brain_observatory.eye_calibration.EyeCalibration` `method`), 291  
`compute_chi()` (in module `allensdk.brain_observatory.chisquare_categorical`), 186  
`compute_chi_shuffle()` (in module `allensdk.brain_observatory.chisquare_categorical`), 186  
`compute_circular_area()` (in module `allensdk.brain_observatory.behavior.eye_tracking_processing`), 101  
`compute_coarse_parameters()` (in module `allensdk.mouse_connectivity.grid.utilities.image_utilities`), 370  
`compute_coarse_planes()` (`allensdk.mouse_connectivity.grid.subimage.base_subimage.SubImage` `method`), 367  
`compute_coarse_planes()` (`allensdk.mouse_connectivity.grid.subimage.cav_subimage.CavSubImage` `method`), 368  
`compute_correlations()` (in module `allensdk.brain_observatory.demix_report`), 289  
`compute_correlations_without_masks()` (in module `allensdk.brain_observatory.demix_report`), 289  
`compute_dff_windowed_median()` (in module `allensdk.brain_observatory.dff`), 189  
`compute_dff_windowed_mode()` (in module `allensdk.brain_observatory.dff`), 190  
`compute_distance()` (in module `allensdk.brain_observatory.receptive_field_analysis.fit_parameters`), 178  
`compute_elliptical_area()` (in module `allensdk.brain_observatory.behavior.eye_tracking_processing`), 101  
`compute_expected()` (in module `allensdk.brain_observatory.chisquare_categorical`), 186  
`compute_frame_times()` (in module `allensdk.brain_observatory.ecephys.stimulus_sync`), 169  
`compute_injection()` (`allensdk.mouse_connectivity.grid.subimage.classic_subimage.ClassicSubImage` `method`), 369  
`compute_intensity()` (`allensdk.mouse_connectivity.grid.subimage.classic_subimage.ClassicSubImage` `method`), 368  
`compute_non_overlap_masks()` (in module `allensdk.internal.brain_observatory.demix_report`), 289  
`compute_non_overlap_traces()` (in module `allensdk.internal.brain_observatory.demix_report`), 289  
`compute_observed()` (in module `allensdk.brain_observatory.demix_report`), 289

`lensdk.brain_observatory.chisquare_categorical()`, 98  
`186` `construct_well_known_file_download_url()`  
`compute_overlap()` (in module `al-` `(allensdk.api.api.Api method)`, 73  
`lensdk.brain_observatory.receptive_field_analysis.parameters`, `name()` (al-  
`178` `lensdk.mouse_connectivity.grid.image_series_gridder.ImageSeries`  
`compute_projection()` (al- `method)`, 372  
`lensdk.mouse_connectivity.grid.subimage.classic_subimage` (class in module `al-`  
`method)`, 368 `lensdk.brain_observatory.behavior.trial_masks`,  
`compute_projection()` (al- `108`  
`lensdk.mouse_connectivity.grid.subimage.count_subimage` (class in `al-`  
`method)`, 369 `lensdk.brain_observatory.ecephys.file_io.continuous_file`,  
`compute_receptive_field()` (in module `al-` `128`  
`lensdk.brain_observatory.receptive_field_analysis.receptive_field` (allensdk.brain\_observatory.ecephys.stimulus\_analysis.d  
`179` `attribute)`, 134  
`compute_receptive_field_with_postprocessing()` (in module `al-`  
`(in module` `al-` `lensdk.mouse_connectivity.grid.utilities.downsampling_utilities)`,  
`lensdk.brain_observatory.receptive_field_analysis.receptive_field`,  
`179` `convert_axis()` (in module `al-`  
`compute_sum_pixels()` (al- `lensdk.internal.mouse_connectivity.projection_thumbnail.projecti`  
`lensdk.mouse_connectivity.grid.subimage.classic_subimage` `ClassicSubImage`  
`method)`, 369 `convert_azimuth_to_degrees()` (in module `al-`  
`compute_sum_pixels()` (al- `lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_fie`  
`lensdk.mouse_connectivity.grid.subimage.count_subimage` `CountSubImage`  
`method)`, 369 `convert_discrete_colormap()` (in module `al-`  
`conditionwise_psth` (al- `lensdk.internal.mouse_connectivity.projection_thumbnail.visualiz`  
`lensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis` `StimulusAnalysis`  
`attribute)`, 143 `convert_elevation_to_degrees()`  
`conditionwise_spike_statistics()` (al- `(in module` `al-`  
`lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` `lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_fie`  
`method)`, 164 `139`  
`conditionwise_statistics` (al- `convert_filepath_caseinsensitive()`  
`lensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis` `StimulusAnalysis` `al-`  
`attribute)`, 144 `lensdk.brain_observatory.behavior.stimulus_processing`,  
`conditionwise_statistics_contrast` (al- `108`  
`lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings` `(DriftingGratings module` `al-`  
`attribute)`, 134 `lensdk.internal.pipeline_modules.run_ophys_session_decomposit`  
`Config` (class in `allensdk.model.biophys_sim.config`), `349`  
`353` `convert_from_titan_linux()` (in module `al-`  
`configure_library_method()` (al- `lensdk.internal.core.lims_utilities)`, 302  
`lensdk.model.glif.glif_neuron.GlifNeuron` `convert_pixel_area_to_degrees()`  
`static method)`, 357 (in module `al-`  
`configure_method()` (al- `lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_fie`  
`lensdk.model.glif.glif_neuron.GlifNeuron` `140`  
`static method)`, 357 `convert_pixels_to_degrees()` (in module `al-`  
`connect()` (in module `al-` `lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_fie`  
`lensdk.internal.core.lims_utilities)`, 302 `140`  
`ConnectedServices` (class in `al-` `convert_type()` (`allensdk.core.swc.Morphology`  
`lensdk.api.queries.connected_services`), 51 `method)`, 262  
`consistency_is_key()` (in module `al-` `convert_type()` (al-  
`lensdk.brain_observatory.behavior.criteria`), `lensdk.internal.morphology.morphology.Morphology`  
`98` `method)`, 329  
`consistent_behavior_within_session()` `convolve()` (in module `al-`  
`(in module` `al-` `lensdk.brain_observatory.receptive_field_analysis.utilities)`,  
`lensdk.brain_observatory.behavior.criteria`, `180`



`copy()` (`allensdk.internal.mouse_connectivity.projection_thumbnail_image_sheet` (class in `allensdk.internal.mouse_connectivity.projection_thumbnail_image_sheet`), 340)  
`copy_local()` (`allensdk.internal.model.biophysical.run_optimize_biophysical_simulation` (method), 314)  
`copy_local()` (`allensdk.internal.model.biophysical.run_simulation` (method), 314)  
`CoronaPlotter` (class in `allensdk.brain_observatory.circle_plots`), 186  
`create_extended_trials()` (in module `allensdk.brain_observatory.behavior.trials_processing`), 109  
`create_eye_tracking_nwb_processing_module()` (in module `allensdk.brain_observatory.nwb`), 173  
`create_fake_metadata()` (in module `allensdk.internal.pipeline_modules.run_ophys_session_decomposition`), 349  
`create_image()` (in module `allensdk.internal.morphology.morphvis`), 332  
`create_images()` (in module `allensdk.brain_observatory.frame_stream.FfmpegInputStream`), 294  
`create_images()` (in module `allensdk.brain_observatory.frame_stream.FrameInputStream`), 295  
`create_manifest()` (in module `allensdk.api.queries.biophysical_api.BiophysicalApi`), 45  
`create_neuropil_mask()` (in module `allensdk.brain_observatory.roi_masks`), 200  
`create_region_mask()` (in module `allensdk.internal.brain_observatory.annotated_region_metrics`), 288  
`create_roi_mask()` (in module `allensdk.brain_observatory.roi_masks`), 200  
`create_roi_mask_array()` (in module `allensdk.brain_observatory.roi_masks`), 201  
`create_stim_table()` (in module `allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spike`), 151  
`create_transcript_fpkms_table()` (in module `allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spike`), 151  
`create_dir()` (`allensdk.config.manifest.Manifest` (class in `allensdk.config.manifest`), 217)

`lensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap()`, (class in `al-`  
344 `lensdk.brain_observatory.sync_dataset`),  
`create_transcripts_for_genes()` 212  
(in module `al-` `DatUtilities` (class in `allensdk.core.dat_utilities`),  
`lensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap`), 238  
344 `DbCredentials` (class in `al-`  
`create_utils()` (in module `al-` `lensdk.core.authentication`), 226  
`lensdk.model.biophysical.utils`), 355  
`credential_injector()` (in module `al-` `dbname` (`allensdk.core.authentication.DbCredentials` at-  
`lensdk.core.authentication`), 226 `tribute`), 226  
`CredentialProvider` (class in `al-` `DEBUG` (`allensdk.api.queries.rma_api.RmaApi` attribute),  
`lensdk.core.authentication`), 226 65  
`CRITERIA` (`allensdk.api.queries.rma_api.RmaApi` at- `debug()` (in module `al-`  
`tribute`), 65 `lensdk.internal.pipeline_modules.run_annotated_region_metrics`),  
`CRITERIA()` (in module `al-` `debug()` (in module `al-`  
`lensdk.internal.brain_observatory.roi_filter_utils`), `lensdk.internal.pipeline_modules.run_demixing`),  
298 345  
`csv_writer()` (`allensdk.api.cache.Cache` static `debug()` (in module `al-`  
`method`), 76 `lensdk.internal.pipeline_modules.run_neuropil_correction`),  
`CUT_DENDRITE` (`allensdk.core.swc.Marker` attribute), 345  
260 `debug()` (in module `al-`  
`CUT_DENDRITE` (`allensdk.internal.core.swc.Marker` at- `lensdk.internal.pipeline_modules.run_observatory_analysis`),  
`tribute`), 304 346  
`CvInputStream` (class in `al-` `debug()` (in module `al-`  
`lensdk.internal.brain_observatory.frame_stream`), `lensdk.internal.pipeline_modules.run_observatory_thumbnails`),  
294 348  
  
D `debug()` (in module `al-`  
`data` (`allensdk.brain_observatory.behavior.image_api.Image` `lensdk.internal.pipeline_modules.run_ophys_eye_calibration`),  
`attribute`), 104 348  
`DATA_MASK` (`allensdk.api.queries.grid_data_api.GridDataApi` `debug()` (in module `al-`  
`attribute`), 52 `lensdk.internal.pipeline_modules.run_ophys_session_decomposition`),  
349  
`DATA_MASK_KEY` (`al-` `debug_clause()` (`al-`  
`lensdk.core.mouse_connectivity_cache.MouseConnectivityCache` `lensdk.api.queries.rma_api.RmaApi` method),  
`attribute`), 241 66  
`data_to_ticks()` (in module `al-` `debug_plot()` (in module `al-`  
`lensdk.brain_observatory.behavior.trials_processing`), `lensdk.internal.pipeline_modules.run_neuropil_correction`),  
109 345  
`data_to_metadata()` (in module `al-` `decision_function()` (`al-`  
`lensdk.brain_observatory.behavior.trials_processing`), `lensdk.internal.brain_observatory.roi_filter_utils.TrainingLabelC`  
109 `method`), 298  
`data_transforms()` (`al-` `decode_bytes()` (in module `al-`  
`lensdk.core.structure_tree.StructureTree` static `lensdk.core.h5_utilities`), 239  
`method`), 258 `default()` (`allensdk.brain_observatory.behavior.behavior_project_lims`  
`dataframe_query()` (`al-` `class method`), 97  
`lensdk.api.queries.brain_observatory_api.BrainObservatoryApi` (`allensdk.brain_observatory.ecephys.ecephys_project_api.ece`  
`method`), 46 `class method`), 118  
`dataframe_query_string()` (`al-` `default()` (`allensdk.brain_observatory.ecephys.ecephys_project_api.ece`  
`lensdk.api.queries.brain_observatory_api.BrainObservatoryApi` `method`), 121  
`method`), 46 `default()` (`allensdk.brain_observatory.JSONEncoder`  
`DataFrameIndexError`, 238 `method`), 216  
`DataFrameKeyError`, 238 `default_api_url` (`allensdk.api.api.Api` attribute), 73  
`dataset` (`allensdk.internal.brain_observatory.time_sync.OphysTimeAlignment` `parse_argument_parser()` (in module `al-`  
`attribute`), 300 `lensdk.internal.core.lims_pipeline_module`),

302  
 default\_ray() (in module allensdk.internal.brain\_observatory.itracker\_utils), 295  
 default\_structure\_ids (allensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache attribute), 241  
 DEFAULT\_STRUCTURE\_SET\_IDS (allensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache attribute), 241  
 DEFORMATION\_FIELD\_HEADER\_KEY (allensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache attribute), 241  
 DEFORMATION\_FIELD\_VOXEL\_KEY (allensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache attribute), 241  
 deg2rad() (in module allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis), 145  
 deg\_to\_dist() (in module allensdk.brain\_observatory.behavior.running\_processing), 106  
 deinterpolate\_RF() (in module allensdk.brain\_observatory.receptive\_field\_analysis.receptive\_field\_analysis), 175  
 delay\_metrics() (allensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor method), 265  
 delete\_tree() (allensdk.core.swc.Morphology method), 262  
 delete\_tree() (allensdk.internal.morphology.morphology.Morphology method), 329  
 demix\_time\_dep\_masks() (in module allensdk.brain\_observatory.demixer), 188  
 demix\_time\_dep\_masks() (in module allensdk.internal.brain\_observatory.demixer), 290  
 DENDRITE (allensdk.core.swc.Morphology attribute), 260  
 DENSITY (allensdk.api.queries.grid\_data\_api.GridDataApi attribute), 53  
 deprecated() (in module allensdk.deprecated), 374  
 DEPRECATED\_KEYS (allensdk.brain\_observatory.sync\_dataset.Dataset attribute), 213  
 DEPRECATED\_SPIKE\_TIMES (allensdk.core.nwb\_data\_set.NwbDataSet attribute), 246  
 descendant\_ids() (allensdk.core.simple\_tree.SimpleTree method), 254  
 descendant\_ids() (allensdk.internal.core.simpletree.SimpleTree method), 304  
 descendants() (allensdk.core.simple\_tree.SimpleTree method), 255  
 descendants() (allensdk.internal.core.simpletree.SimpleTree method), 304  
 Description (allensdk.config.model.description), 221  
 DescriptionParser (allensdk.config.model.description\_parser), 221  
 deserialize() (allensdk.brain\_observatory.behavior.image\_api.ImageApi static method), 104  
 deserialize\_image() (allensdk.brain\_observatory.behavior.behavior\_ophys\_session.BehaviorOphysSession method), 112  
 DETAILED\_STIMULUS\_PARAMETERS (allensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession attribute), 164  
 detect\_bursts() (in module allensdk.ephys.ephys\_features), 270  
 detect\_duplicates() (allensdk.internal.brain\_observatory.mask\_set.MaskSet method), 296  
 detect\_events() (in module allensdk.brain\_observatory.receptive\_field\_analysis.eventdetection), 178  
 detect\_pauses() (in module allensdk.ephys.ephys\_features), 271  
 detect\_putative\_spikes() (in module allensdk.ephys.ephys\_features), 271  
 detect\_unions() (allensdk.internal.brain\_observatory.mask\_set.MaskSet method), 296  
 determine\_likely\_blinks() (in module allensdk.brain\_observatory.behavior.eye\_tracking\_processing), 101  
 determine\_outliers() (in module allensdk.brain\_observatory.behavior.eye\_tracking\_processing), 102  
 DEVMOUSE\_2012 (allensdk.api.queries.reference\_space\_api.ReferenceSpaceApi attribute), 63  
 DEVMOUSE\_ATLAS\_PRODUCTS (allensdk.api.queries.mouse\_atlas\_api.MouseAtlasApi attribute), 57  
 df\_columns (allensdk.config.manifest\_builder.ManifestBuilder attribute), 224  
 dff\_traces (allensdk.brain\_observatory.behavior.behavior\_ophys\_session.BehaviorOphysSession attribute), 92  
 dfftraces (allensdk.brain\_observatory.stimulus\_analysis.StimulusAnalysis attribute), 207

DFMFLD\_RESOLUTIONS (al- [lensdk.api.queries.mouse\\_connectivity\\_api.MouseConnectivityApi](#) method), 58

[lensdk.core.mouse\\_connectivity\\_cache.MouseConnectivityCache](#) (class in [lensdk.core.mouse\\_connectivity\\_cache](#)), 241

[lensdk.brain\\_observatory.receptive\\_field\\_analysis.tools](#) (module in [lensdk.brain\\_observatory](#)), 180

[dict\\_generator\(\)](#) (in module [lensdk.brain\\_observatory.receptive\\_field\\_analysis.tools](#)), 279

[dict\\_to\\_indexed\\_array\(\)](#) (in module [lensdk.brain\\_observatory](#)), 216

[dim\\_color\(\)](#) ([allensdk.brain\\_observatory.observatory\\_plots.DimensionPatchHandler](#) method), 195

[DimensionPatchHandler](#) (class in [lensdk.brain\\_observatory.observatory\\_plots](#)), 195

[DIR](#) ([allensdk.config.manifest.Manifest](#) attribute), 222

[DIR\\_CCW](#) ([allensdk.brain\\_observatory.circle\\_plots.PolarPlotter](#) attribute), 187

[DIR\\_CW](#) ([allensdk.brain\\_observatory.circle\\_plots.PolarPlotter](#) attribute), 187

[direct\\_sum\\_projection\\_pixels](#) (al- [lensdk.internal.mouse\\_connectivity.interval\\_unionize.tissuecyte\\_base\\_unionize.record.TissuecyteBaseUnionize](#) attribute), 337

[direct\\_unionize\(\)](#) (al- [lensdk.internal.mouse\\_connectivity.interval\\_unionize.interval\\_unionize.IntervalUnionizer](#) method), 336

[direct\\_voxel\\_counts\(\)](#) (al- [lensdk.core.reference\\_space.ReferenceSpace](#) method), 249

[direct\\_voxel\\_map](#) (al- [lensdk.core.reference\\_space.ReferenceSpace](#) attribute), 249

[directions](#) ([allensdk.brain\\_observatory.ecephys.stimulus\\_analysis.interval\\_unionize.record.TissuecyteBaseUnionize](#) attribute), 133

[DIRNAME](#) ([allensdk.config.manifest.Manifest](#) attribute), 222

[distance\(\)](#) ([allensdk.internal.brain\\_observatory.mask\\_set.MaskSet](#) method), 296

[do\\_blur\(\)](#) (in module [lensdk.internal.mouse\\_connectivity.projection\\_thumbnail\\_generation.projection\\_strip](#)), 339

[do\\_query\(\)](#) ([allensdk.api.api.Api](#) method), 73

[do\\_rma\\_query\(\)](#) ([allensdk.api.api.Api](#) method), 73

[DotMotion](#) (class in [lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.dot\\_motion.dot\\_motion.DotMotion](#)), 133

[download\\_alignment3d\(\)](#) (al- [lensdk.api.queries.grid\\_data\\_api.GridDataApi](#) method), 53

[download\\_annotation\\_volume\(\)](#) (al- [lensdk.api.queries.reference\\_space\\_api.ReferenceSpaceApi](#) method), 63

[download\\_atlas\\_image\(\)](#) (al- [lensdk.api.queries.image\\_download\\_api.ImageDownloadApi](#) method), 55

[download\\_data\\_mask\(\)](#) (al- [lensdk.api.queries.mouse\\_connectivity\\_api.MouseConnectivityApi](#) method), 58

[download\\_data\\_mask\(\)](#) (al- [lensdk.internal.api.queries.mouse\\_connectivity\\_api\\_prerelease.MouseConnectivityApiPrerelease](#) method), 279

[download\\_deformation\\_field\(\)](#) (al- [lensdk.api.queries.grid\\_data\\_api.GridDataApi](#) method), 53

[download\\_density\(\)](#) (al- [lensdk.api.queries.mouse\\_atlas\\_api.MouseAtlasApi](#) method), 57

[download\\_expression\\_energy\(\)](#) (al- [lensdk.api.queries.mouse\\_atlas\\_api.MouseAtlasApi](#) method), 57

[download\\_expression\\_grid\\_data\(\)](#) (al- [lensdk.api.queries.grid\\_data\\_api.GridDataApi](#) method), 53

[download\\_expression\\_intensity\(\)](#) (al- [lensdk.api.queries.mouse\\_atlas\\_api.MouseAtlasApi](#) method), 55

[download\\_gene\\_expression\\_grid\\_data\(\)](#) (al- [lensdk.api.queries.grid\\_data\\_api.GridDataApi](#) method), 53

[download\\_image\(\)](#) (al- [lensdk.api.queries.image\\_download\\_api.ImageDownloadApi](#) method), 55

[download\\_injection\\_density\(\)](#) (al- [lensdk.api.queries.mouse\\_connectivity\\_api.MouseConnectivityApi](#) method), 58

[download\\_injection\\_density\(\)](#) (al- [lensdk.internal.api.queries.mouse\\_connectivity\\_api\\_prerelease.MouseConnectivityApiPrerelease](#) method), 279

[download\\_injection\\_fraction\(\)](#) (al- [lensdk.api.queries.mouse\\_connectivity\\_api.MouseConnectivityApi](#) method), 58

[download\\_injection\\_fraction\(\)](#) (al- [lensdk.internal.api.queries.mouse\\_connectivity\\_api\\_prerelease.MouseConnectivityApiPrerelease](#) method), 279

[download\\_mouse\\_atlas\\_volume\(\)](#) (al- [lensdk.api.queries.reference\\_space\\_api.ReferenceSpaceApi](#) method), 64

[download\\_projection\\_density\(\)](#) (al- [lensdk.api.queries.mouse\\_connectivity\\_api.MouseConnectivityApi](#) method), 59

[download\\_projection\\_density\(\)](#) (al- [lensdk.internal.api.queries.mouse\\_connectivity\\_api\\_prerelease.MouseConnectivityApiPrerelease](#) method), 279

[download\\_projection\\_grid\\_data\(\)](#) (al- [lensdk.api.queries.grid\\_data\\_api.GridDataApi](#) method), 54

[download\\_projection\\_grid\\_data\(\)](#) (al- [lensdk.internal.api.queries.grid\\_data\\_api\\_prerelease.GridDataApiPrerelease](#) method), 279

[download\\_projection\\_image\(\)](#) (al- [lensdk.api.queries.grid\\_data\\_api.GridDataApi](#) method), 54





118  
 EcephysProjectWarehouseApi (class in al-  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_warehouse\_api), 65  
 121  
 EcephysSession (class in al-  
 lensdk.brain\_observatory.ecephys.ecephys\_session), 81  
 162  
 EcephysSessionApi (class in al-  
 lensdk.brain\_observatory.ecephys.ecephys\_session\_api), 198  
 127  
 EcephysSyncDataset (class in al-  
 lensdk.brain\_observatory.ecephys.file\_io.ecephys\_sync\_dataset), 312  
 128  
 elevations (allensdk.brain\_observatory.ecephys.stimulus\_analysis.receptive\_field\_mapping  
 attribute), 139  
 ellipse\_angle\_of\_rotation() (in module al-  
 lensdk.internal.brain\_observatory.fit\_ellipse),  
 294  
 ellipse\_angle\_of\_rotation2() (in module al-  
 lensdk.internal.brain\_observatory.fit\_ellipse),  
 294  
 ellipse\_axis\_length() (in module al-  
 lensdk.internal.brain\_observatory.fit\_ellipse),  
 294  
 ellipse\_center() (in module al-  
 lensdk.internal.brain\_observatory.fit\_ellipse),  
 294  
 empty\_metrics\_table() (al-  
 lensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis.  
 method), 144  
 enable\_console\_log() (in module al-  
 lensdk.config), 225  
 ENERGY (allensdk.api.queries.grid\_data\_api.GridDataApi  
 attribute), 53  
 EnvCredentialProvider (class in al-  
 lensdk.core.authentication), 226  
 EPHYS\_DATA\_KEY (al-  
 lensdk.core.cell\_types\_cache.CellTypesCache  
 attribute), 236  
 EPHYS\_FEATURES\_KEY (al-  
 lensdk.core.cell\_types\_cache.CellTypesCache  
 attribute), 236  
 EPHYS\_SWEEPS\_KEY (al-  
 lensdk.core.cell\_types\_cache.CellTypesCache  
 attribute), 236  
 EphysCellFeatureExtractor (class in al-  
 lensdk.ephys.ephys\_extractor), 264  
 EphysFeatureExtractor (class in al-  
 lensdk.ephys.feature\_extractor), 276  
 EphysFeatures (class in al-  
 lensdk.ephys.feature\_extractor), 276  
 EphysSweepFeatureExtractor (class in al-  
 lensdk.ephys.ephys\_extractor), 265  
 EphysSweepSetFeatureExtractor (class in al-  
 lensdk.ephys.ephys\_extractor), 267  
 EpochSeparationException, 185  
 equals() (in module al-  
 lensdk.brain\_observatory.behavior.behavior\_ophys\_api.behavior.  
 81  
 error\_calc() (in module al-  
 lensdk.brain\_observatory.r\_neuropil), 198  
 escape\_char (allensdk.internal.model.biophysical.passive\_fitting.output  
 attribute), 312  
 estimate\_adjusted\_detection\_parameters()  
 (in module al-  
 lensdk.brain\_observatory.r\_neuropil), 198  
 estimate\_contamination\_ratios() (in mod-  
 ule allensdk.brain\_observatory.r\_neuropil),  
 198  
 estimate\_error() (al-  
 lensdk.brain\_observatory.r\_neuropil.NeuropilSubtract  
 method), 197  
 estimate\_fi\_shift() (in module al-  
 lensdk.internal.model.biophysical.check\_fi\_shift),  
 312  
 estimate\_frame\_duration() (in module al-  
 lensdk.brain\_observatory.ecephys.stimulus\_sync),  
 169  
 estimate\_sag() (al-  
 lensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor  
 method), 265  
 estimate\_time\_constant() (al-  
 lensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor  
 method), 265  
 euclidean\_distance() (in module al-  
 lensdk.internal.morphology.node), 334  
 evaluate() (allensdk.internal.model.glif.glif\_optimizer.GlifOptimizer  
 method), 318  
 EVENTS\_DATA\_KEY (al-  
 lensdk.core.brain\_observatory\_cache.BrainObservatoryCache  
 attribute), 227  
 events\_to\_pvalues\_no\_fdr\_correction()  
 (in module al-  
 lensdk.brain\_observatory.receptive\_field\_analysis.receptive\_field  
 180  
 EXCEPT (allensdk.api.queries.rma\_api.RmaApi at-  
 tribute), 65  
 EXCPT (allensdk.api.queries.rma\_api.RmaApi attribute),  
 65  
 execute\_templated() (in module al-  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_api.utilities),  
 125  
 exp\_curve() (in module al-  
 lensdk.internal.ephys.plot\_qc\_figures), 306  
 exp\_curve() (in module al-  
 lensdk.internal.ephys.plot\_qc\_figures3),

308  
 exp\_decay() (in module al-  
 lensdk.internal.model.glif.MLIN), 315  
 exp\_fit\_c() (in module al-  
 lensdk.internal.model.glif.threshold\_adaptation),  
 324  
 exp\_force\_c() (in module al-  
 lensdk.internal.model.glif.threshold\_adaptation),  
 324  
 exp\_function() (in module al-  
 lensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_graphs,  
 143  
 EXPERIMENT\_CONTAINERS\_KEY (al-  
 lensdk.core.brain\_observatory\_cache.BrainObservatoryCache  
 attribute), 227  
 experiment\_correlation\_search() (al-  
 lensdk.api.queries.mouse\_connectivity\_api.MouseConnectivityApi  
 method), 59  
 EXPERIMENT\_DATA\_KEY (al-  
 lensdk.core.brain\_observatory\_cache.BrainObservatoryCache  
 attribute), 227  
 experiment\_id (al-  
 lensdk.internal.pipeline\_modules.run\_ophys\_time\_sync.TimeSyncOutputs  
 attribute), 349  
 experiment\_injection\_coordinate\_search() (al-  
 lensdk.brain\_observatory.ecephys.align\_timestamps.barcode\_syn-  
 (allensdk.api.queries.mouse\_connectivity\_api.MouseConnectivityApi  
 method), 59  
 experiment\_source\_search() (al-  
 lensdk.api.queries.mouse\_connectivity\_api.MouseConnectivityApi  
 method), 60  
 experiment\_spatial\_search() (al-  
 lensdk.api.queries.mouse\_connectivity\_api.MouseConnectivityApi  
 method), 60  
 ExperimentGeometry (class in al-  
 lensdk.brain\_observatory.stimulus\_info),  
 209  
 EXPERIMENTS\_KEY (al-  
 lensdk.core.brain\_observatory\_cache.BrainObservatoryCache  
 attribute), 227  
 EXPERIMENTS\_KEY (al-  
 lensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache  
 attribute), 241  
 EXPERIMENTS\_PRERELEASE\_KEY (al-  
 lensdk.internal.core.mouse\_connectivity\_cache.prerelease.MouseConnectivityCachePrerelease  
 attribute), 303  
 explode\_response\_window() (in module al-  
 lensdk.brain\_observatory.behavior.mtrain),  
 105  
 export\_frame\_to\_hdf5() (in module al-  
 lensdk.internal.brain\_observatory.ophys\_session\_decomposition.math,  
 297  
 export\_itksnap\_labels() (al-  
 lensdk.core.reference\_space.ReferenceSpace  
 method), 249  
 export\_label\_description() (al-  
 lensdk.core.structure\_tree.StructureTree  
 method), 258  
 expsymm\_cdf() (in module al-  
 lensdk.internal.model.glif.MLIN), 315  
 expsymm\_pdf() (in module al-  
 lensdk.internal.model.glif.MLIN), 315  
 ExtendedTrialSchema (class in al-  
 lensdk.brain\_observatory.behavior.mtrain),  
 104  
 extract() (in module al-  
 lensdk.mouse\_connectivity.grid.utilities.downsampling\_utilities),  
 370  
 extract\_barcodes() (al-  
 lensdk.brain\_observatory.ecephys.align\_timestamps.barcode\_syn-  
 method), 114  
 extract\_barcodes\_from\_states() (al-  
 lensdk.brain\_observatory.ecephys.align\_timestamps.channel\_stat-  
 module al-  
 lensdk.brain\_observatory.ecephys.align\_timestamps.channel\_stat-  
 115  
 extract\_outputs\_from\_times() (in module al-  
 lensdk.brain\_observatory.ecephys.align\_timestamps.barcode\_syn-  
 method), 114  
 extract\_cell\_features() (in module al-  
 lensdk.ephys.extract\_cell\_features), 275  
 extract\_params\_from\_stim\_repr() (in module al-  
 lensdk.brain\_observatory.ecephys.stimulus\_table.stimulus\_param-  
 method), 155  
 extract\_data() (al-  
 lensdk.internal.mouse\_connectivity.interval\_unionize.interval\_un-  
 method), 336  
 extract\_data() (al-  
 lensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_u-  
 method), 339  
 extract\_data() (in module al-  
 lensdk.internal.ephys.core\_feature\_extract),  
 306  
 extract\_frame\_times() (al-  
 lensdk.brain\_observatory.ecephys.file\_io.ecephys\_sync\_dataset.E-  
 method), 128  
 extract\_frame\_times\_from\_photodiode() (al-  
 lensdk.brain\_observatory.ecephys.file\_io.ecephys\_sync\_dataset.E-  
 method), 128  
 extract\_frame\_times\_from\_vsyncs() (al-  
 lensdk.brain\_observatory.ecephys.file\_io.ecephys\_sync\_dataset.E-  
 method), 128  
 extract\_from\_schema() (in module al-  
 lensdk.brain\_observatory.nwb.metadata),  
 170  
 extract\_injection\_from\_segmentation()

(`allensdk.mouse_connectivity.grid.subimage.base_subimage.SegmentationsSubImage` attribute), 213

`extract_led_times()` (`al- eye_video_timestamps` (al-  
`lensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset.EcephysSyncDataset` attribute), 300  
method), 129

`extract_signal_from_segmentation()` (`al- EyeCalibration` (class in `al-  
lensdk.mouse_connectivity.grid.subimage.base_subimage.SegmentationsSubImage` `lensdk.brain_observatory.eye_calibration`),  
method), 367 291

`extract_splits_from_states()` (in module `al-  
lensdk.brain_observatory.ecephys.align_timestamps.channel_states`),  
115 `f1_f0()` (in module `al-  
lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings` attribute), 135

`extract_stim_class_from_repr()` (in module `al-  
lensdk.brain_observatory.ecephys.stimulus_table.stimulus_parameters` attribute), 155  
(`lensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset.EcephysSyncDataset` class method), 129

`extract_sweep_features()` (in module `al- factory()` (`allensdk.brain_observatory.ecephys.file_io.stim_file.CamStim`  
`lensdk.ephys.extract_cell_features`), 276 class method), 129

`extractor_for_nwb_sweeps()` (in module `al- FALSE` (`allensdk.api.queries.rma_api.RmaApi` attribute),  
`lensdk.ephys.ephys_extractor`), 268 65

`extralength` (`allensdk.brain_observatory.locally_sparse_noise.locally_sparse_noise` (in module `al-  
lensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis` attribute), 193

`extralength` (`allensdk.brain_observatory.natural_scenes.NaturalScenes` attribute), 194 `FanPlotter` (class in `al-  
lensdk.brain_observatory.circle_plots`), 187

`extralength` (`allensdk.brain_observatory.static_gratings.StaticGratings` attribute), 205 `FeatureError`, 268

`extrapolate_model_spike_from_endpoints()` `fetchall()` (`allensdk.internal.api.PostgresQueryMixin`  
(in module `al- method`), 287  
`lensdk.internal.model.glif.glif_optimizer_neuron`), `fetchone()` (`allensdk.internal.api.PostgresQueryMixin`  
320 method), 287

`extrapolate_model_spike_from_endpoints_single_input()` (in module `allensdk.internal.model.GLM`), 326  
(in module `al- FfmpegInputStream` (class in `al-  
lensdk.internal.model.glif.glif_optimizer_neuron`), `lensdk.internal.brain_observatory.frame_stream`),  
320 294

`extrapolate_spike_time()` (in module `al- FfmpegOutputStream` (class in `al-  
lensdk.internal.model.glif.glif_optimizer_neuron`), `lensdk.internal.brain_observatory.frame_stream`),  
321 294

`extrapolate_spike_voltage()` (in module `al- figure_in_px()` (in module `al-  
lensdk.internal.model.glif.glif_optimizer_neuron`), `lensdk.brain_observatory.observatory_plots`),  
321 196

`eye_alignment` (`al- FILE` (`allensdk.config.manifest.Manifest` attribute), 222  
`lensdk.internal.pipeline_modules.run_ophys_time_sync_outputs` (al-  
attribute), 349 `lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet` attribute), 222

`eye_delta` (`allensdk.internal.pipeline_modules.run_ophys_time_sync_outputs` (al-  
attribute), 349 `fill_sweep_responses()` (al-  
`lensdk.core.nwb_data_set.NwbDataSet` attribute), 246

`EYE_GAZE_DATA_KEY` (`al- lensdk.core.brain_observatory_cache.BrainObservatoryCache` attribute), 227  
`lensdk.core.brain_observatory_cache.BrainObservatoryCache` method), 246

`eye_times` (`allensdk.internal.pipeline_modules.run_ophys_time_sync_outputs` (al-  
attribute), 349 `filter()` (`allensdk.api.queries.rma_api.RmaApi` attribute), 222

`eye_tracking` (`allensdk.brain_observatory.behavior.behavior_optimizer.BehaviorOptimizer` attribute), 92  
`lensdk.brain_observatory.behavior_optimizer.BehaviorOptimizer` attribute), 92

`eye_tracking_data_is_valid()` (in module `al- filter_cell_specimens()` (al-  
`lensdk.brain_observatory.nwb`), 173 `lensdk.api.queries.brain_observatory_api.BrainObservatoryApi` attribute), 222

`EYE_TRACKING_KEYS` (al- method), 46



<code>filter_cells()</code> (al- lensdk.api.queries.cell_types_api.CellTypesApi method), 49	<code>find_bin_center()</code> (in module al- lensdk.internal.model.glif.MLIN), 315
<code>filter_cells_api()</code> (al- lensdk.api.queries.cell_types_api.CellTypesApi method), 49	<code>find_coarse_long_square_amp_delta()</code> (in module al- lensdk.internal.ephys.core_feature_extract), 305
<code>filter_digital()</code> (in module al- lensdk.brain_observatory.behavior.sync.process_sync), 85	<code>find_container_tags()</code> (in module al- lensdk.api.queries.brain_observatory_api), 49
<code>filter_experiment_containers()</code> (al- lensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 47	<code>find_downstroke_indexes()</code> (in module al- lensdk.ephys.ephys_features), 273
<code>filter_experiments()</code> (al- lensdk.core.mouse_connectivity_cache.MouseConnectivityCache method), 241	<code>find_experiment_acquisition_age()</code> (in module al- lensdk.api.queries.brain_observatory_api), 49
<code>filter_experiments()</code> (al- lensdk.internal.core.mouse_connectivity_cache_prerelease.MouseConnectivityCachePrerelease method), 303	<code>find_first_model_spike()</code> (in module al- lensdk.internal.core.mouse_connectivity_cache_prerelease.MouseConnectivityCachePrerelease), 321
<code>filter_experiments_and_containers()</code> (al- lensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 47	<code>find_licks()</code> (in module al- lensdk.brain_observatory.behavior.trials_processing), 109
<code>filter_nodes()</code> (al- lensdk.core.simple_tree.SimpleTree method), 255	<code>find_long_square_sweeps()</code> (in module al- lensdk.internal.model.glif.find_sweeps), 316
<code>filter_ophys_experiments()</code> (al- lensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 47	<code>find_matching_index()</code> (in module al- lensdk.brain_observatory.ecephys.align_timestamps.barcode), 421
<code>filter_putative_spikes()</code> (in module al- lensdk.ephys.ephys_features), 272	<code>find_negative_baselines()</code> (in module al- lensdk.brain_observatory.demixer), 188
<code>filter_structure_unionizes()</code> (al- lensdk.core.mouse_connectivity_cache.MouseConnectivityCache method), 242	<code>find_negative_baselines()</code> (in module al- lensdk.internal.brain_observatory.demixer), 200
<code>filter_sweeps()</code> (in module al- lensdk.internal.ephys.core_feature_extract), 305	<code>find_negative_transients_threshold()</code> (in module allensdk.brain_observatory.demixer), 188
<code>filtered_sweep_numbers()</code> (in module al- lensdk.internal.ephys.core_feature_extract), 305	<code>find_negative_transients_threshold()</code> (in module al- lensdk.internal.brain_observatory.demixer), 290
<code>filters()</code> (allensdk.api.queries.rma_api.RmaApi method), 66	<code>find_noise_sweeps()</code> (in module al- lensdk.internal.model.glif.find_sweeps), 316
<code>finalize()</code> (allensdk.brain_observatory.circle_plots.PolarPlotter method), 187	<code>find_peak_indexes()</code> (in module al- lensdk.ephys.ephys_features), 273
<code>finalize_no_axes()</code> (in module al- lensdk.brain_observatory.observatory_plots), 196	<code>find_ramp_sweeps()</code> (in module al- lensdk.internal.model.glif.find_sweeps), 316
<code>finalize_no_labels()</code> (in module al- lensdk.brain_observatory.observatory_plots), 196	<code>find_ramp_to_rheo_sweeps()</code> (in module al- lensdk.internal.model.glif.find_sweeps), 316
<code>finalize_with_axes()</code> (in module al- lensdk.brain_observatory.observatory_plots), 196	<code>find_ranked_sweep()</code> (in module al- lensdk.internal.model.glif.find_sweeps), 316
<code>find()</code> (allensdk.core.swc.Morphology method), 262	<code>find_short_square_sweeps()</code> (in module al- lensdk.internal.model.glif.find_sweeps), 316
<code>find()</code> (allensdk.internal.morphology.morphology.Morphology method), 329	<code>find_specimen_cre_line()</code> (in module al- lensdk.api.queries.brain_observatory_api), 49
	<code>find_specimen_reporter_line()</code> (in module

`allensdk.api.queries.brain_observatory_api`),  
49

`find_specimen_transgenic_lines()` (in module `allensdk.api.queries.brain_observatory_api`),  
49

`find_spikes_list()` (in module `allensdk.internal.model.glif.find_spikes`), 316

`find_spikes_list_old()` (in module `allensdk.internal.model.glif.find_spikes`), 316

`find_spikes_old()` (in module `allensdk.internal.model.glif.find_spikes`), 316

`find_spikes_ssq_list()` (in module `allensdk.internal.model.glif.find_spikes`), 316

`find_stim_start()` (in module `allensdk.internal.ephys.core.feature_extract`),  
305

`find_sweep_stim_start()` (in module `allensdk.internal.ephys.core.feature_extract`),  
305

`find_sweeps()` (in module `allensdk.internal.model.glif.find_sweeps`), 316

`find_time_index()` (in module `allensdk.ephys.ephys_features`), 273

`find_trough_indexes()` (in module `allensdk.ephys.ephys_features`), 273

`find_upstroke_indexes()` (in module `allensdk.ephys.ephys_features`), 273

`find_widths()` (in module `allensdk.ephys.ephys_features`), 274

`find_zero_baselines()` (in module `allensdk.brain_observatory.demixer`), 188

`find_zero_baselines()` (in module `allensdk.internal.brain_observatory.demixer`),  
290

`findlevel()` (in module `allensdk.brain_observatory.findlevel`), 192

`fit()` (`allensdk.brain_observatory.r_neuropil.NeuropilSubtract` method), 197

`fit_2d_gaussian()` (in module `allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping`),  
140

`fit_avoltage_bvoltage()` (in module `allensdk.internal.model.glif.threshold_adaptation`),  
324

`fit_avoltage_bvoltage_th()` (in module `allensdk.internal.model.glif.threshold_adaptation`),  
325

`fit_block_coordinate_desc()` (`allensdk.brain_observatory.r_neuropil.NeuropilSubtract` method), 197

`fit_ellipse()` (`allensdk.internal.brain_observatory.fit_ellipse.FitEllipse` method), 294

`fit_ellipse()` (in module `allensdk.internal.brain_observatory.fit_ellipse`),  
294

`fit_exp()` (in module `allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis`),  
146

`fit_fit_slope()` (in module `allensdk.ephys.ephys_extractor`), 268

`fit_membrane_time_constant()` (in module `allensdk.ephys.ephys_features`), 274

`fit_parameters_file_entries()` (`allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader` method), 277

`fit_parameters_path()` (`allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader` method), 277

`fit_prespike_time_constant()` (in module `allensdk.ephys.ephys_features`), 274

`fit_sf_tuning()` (in module `allensdk.brain_observatory.ecephys.stimulus_analysis.static_grating_tuning`),  
143

`FitEllipse` (class in `allensdk.internal.brain_observatory.fit_ellipse`),  
294

`fitgaussian2D()` (in module `allensdk.brain_observatory.receptive_field_analysis.fitgaussian2D`),  
178

`fix_array_dimensions()` (in module `allensdk.core.sitk_utilities`), 256

`fix_change_time()` (in module `allensdk.brain_observatory.behavior.mtrain`),  
106

`fix_unary_sections()` (`allensdk.config.model.description.Description` method), 221

`fix_unexpected_edges()` (in module `allensdk.brain_observatory.ecephys.stimulus_sync`),  
169

`fixed()` (`allensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache` class method), 159

`fix_receptive_field_mapping()` (`allensdk.internal.mouse_connectivity.projection_thumbnail.volume_projection_thumbnail` class method), 341

`flag_unexpected_edges()` (in module `allensdk.brain_observatory.ecephys.stimulus_sync`),  
169

`Flashes` (class in `allensdk.brain_observatory.ecephys.stimulus_analysis.flashes`),  
136

`Fluorophore` (`allensdk.api.queries.connected_services.ConnectedServices` attribute), 51

`float_label()` (in module `allensdk.brain_observatory.observatory_plots`),  
196

`for_drifting_gratings()` (alias `lensdk.brain_observatory.circle_plots.FanPlotter` static method), 187  
`for_drifting_gratings()` (alias `lensdk.brain_observatory.circle_plots.FanPlotter` static method), 187  
`for_static_gratings()` (alias `lensdk.brain_observatory.circle_plots.FanPlotter` static method), 187  
`foraging_id_to_behavior_session_id()` (alias `lensdk.internal.api.behavior_lims_api.BehaviorLimsApi` static method), 284  
`format_query_string` (alias `lensdk.brain_observatory.ecephys.ecephys_project_api.rma_client.RmaClient` attribute), 124  
`FRAME_KEYS` (alias `lensdk.brain_observatory.sync_dataset.Dataset` attribute), 213  
`frame_time_offset()` (in module `lensdk.brain_observatory.behavior.sync`), 86  
`FrameInputStream` (class in `lensdk.internal.brain_observatory.frame_stream`), 295  
`FrameOutputStream` (class in `lensdk.internal.brain_observatory.frame_stream`), 295  
`frames` (alias `lensdk.brain_observatory.ecephys.stimulus_analysis_utils.to_neural_scenes` attribute), 137  
`frames_per_second` (alias `lensdk.brain_observatory.ecephys.file_io.stim_file.CamStimFile` attribute), 129  
`frequency()` (alias `lensdk.brain_observatory.sync_dataset.Dataset` method), 213  
`FriendlyDate` (class in `lensdk.brain_observatory.behavior.mtrain`), 105  
`FriendlyDateTime` (class in `lensdk.brain_observatory.behavior.mtrain`), 105  
`from_analysis_file()` (alias `lensdk.brain_observatory.drifting_gratings.DriftingGratings` static method), 191  
`from_analysis_file()` (alias `lensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` static method), 193  
`from_analysis_file()` (alias `lensdk.brain_observatory.natural_movie.NaturalMovie` static method), 194  
`from_analysis_file()` (alias `lensdk.brain_observatory.natural_scenes.NaturalScenes` static method), 194  
`from_analysis_file()` (alias `lensdk.brain_observatory.static_gratings.StaticGratings` static method), 205  
`from_dataframe()` (alias `lensdk.config.manifest_builder.ManifestBuilder` method), 224  
`from_df()` (alias `lensdk.brain_observatory.stimulus_info.BinaryIntervalSequence` static method), 208  
`from_dict()` (alias `lensdk.internal.model.glif.glif_optimizer_neuron.GlifOptimizerNeuron` class method), 319  
`from_dict()` (alias `lensdk.internal.morphology.node.Node` class method), 334  
`from_dict()` (alias `lensdk.model.glif.glif_neuron.GlifNeuron` class method), 358  
`from_dict_legacy()` (alias `lensdk.internal.model.glif.glif_optimizer_neuron.GlifOptimizerNeuron` class method), 358  
`from_file_name()` (alias `lensdk.internal.api.queries.grid_data_api_prerelease.GridDataApi` class method), 279  
`from_foraging_id()` (alias `lensdk.internal.api.behavior_lims_api.BehaviorLimsApi` class method), 284  
`from_json_file()` (alias `lensdk.config.app.application_config.ApplicationConfig` method), 218  
`from_json_string()` (alias `lensdk.config.app.application_config.ApplicationConfig` method), 218  
`from_lims()` (alias `lensdk.brain_observatory.behavior.behavior_data_session.BehaviorDataSession` class method), 89  
`from_lims()` (alias `lensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession` class method), 89  
`from_lims()` (alias `lensdk.brain_observatory.behavior.behavior_project_cache.BehaviorProjectCache` class method), 159  
`from_nwb_path()` (alias `lensdk.brain_observatory.behavior.behavior_data_session.BehaviorDataSession` class method), 89  
`from_nwb_path()` (alias `lensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession` class method), 89  
`from_nwb_path()` (alias `lensdk.brain_observatory.behavior.behavior_project_cache.BehaviorProjectCache` class method), 159  
`from_path()` (alias `lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` class method), 164  
`from_path()` (alias `lensdk.brain_observatory.nwb.nwb_api.NwbApi` class method), 170  
`from_path()` (alias `lensdk.brain_observatory.ecephys.ecephys_session_api.EcephysSessionApi` class method), 126  
`from_path()` (alias `lensdk.brain_observatory.nwb.nwb_api.NwbApi` class method), 170  
`from_sweeps()` (alias `lensdk.ephys.ephys_extractor.EphysSweepSetFeatureExtractor` class method), 267  
`from_warehouse()` (alias `lensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache` class method), 159  
`full_genotype` (alias `lensdk.config.manifest_builder.ManifestBuilder` method), 224

[lensdk.brain\\_observatory.ecephys.ecephys\\_session.EcephysSession](#) (in module [lensdk.brain\\_observatory.ecephys.ecephys\\_session](#)), 164  
[lensdk.brain\\_observatory.ecephys.nwb.EcephysLaboratoryMetaDatum](#) (in module [lensdk.brain\\_observatory.ecephys.nwb](#)), 132  
**G**  
[gather\\_from\\_seeds\(\)](#) (in module [lensdk.core.mouse\\_connectivity\\_cache.MouseConnectivityCache](#)), 180  
[lensdk.internal.model.biophysical.make\\_deep\\_fit\\_json.Reporter](#) (method), 242  
[get\\_age\(\)](#) ([allensdk.internal.api.behavior\\_data\\_lims\\_api.BehaviorDataLimsApi](#)), 180  
[lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.get\\_age\(\)](#) (in module [lensdk.brain\\_observatory.ecephys.stimulus\\_analysis](#)), 143  
[get\\_affine\\_parameters\(\)](#) (in module [lensdk.core.mouse\\_connectivity\\_cache.MouseConnectivityCache](#)), 180  
[get\\_alignment\\_array\(\)](#) (in module [lensdk.internal.brain\\_observatory.time\\_sync](#)), 300  
[gaussian2D\(\)](#) (in module [lensdk.brain\\_observatory.receptive\\_field\\_analysis.fitgaussian2d](#)), 178  
[gaussian\\_moments\\_2d\(\)](#) (in module [lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.receptive\\_field\\_analysis](#)), 140  
[GaussianFitError](#), 178  
[generate\\_fit\\_file\(\)](#) (in module [lensdk.brain\\_observatory.sync\\_dataset.Dataset](#)), 213  
[lensdk.internal.model.biophysical.make\\_deep\\_fit\\_json.Reporter](#) (method), 242  
[generate\\_manifest\\_lims\(\)](#) (in module [lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)), 227  
[lensdk.internal.model.biophysical.run\\_optimize.RunOptimizer](#) (method), 314  
[generate\\_manifest\\_lims\(\)](#) (in module [lensdk.brain\\_observatory.sync\\_dataset.Dataset](#)), 213  
[lensdk.internal.model.biophysical.run\\_simulate\\_lims.RunSimulateLims](#) (method), 314  
[generate\\_manifest\\_rma\(\)](#) (in module [lensdk.core.cell\\_types\\_cache.CellTypesCache](#)), 236  
[lensdk.internal.model.biophysical.run\\_optimize.RunOptimizer](#) (method), 314  
[generate\\_manifest\\_rma\(\)](#) (in module [lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache.EcephysProjectCache](#)), 160  
[lensdk.internal.model.biophysical.run\\_simulate\\_lims.RunSimulateLims](#) (method), 314  
[generate\\_morphology\(\)](#) (in module [lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)), 227  
[lensdk.internal.model.biophysical.deap\\_utils.Utils](#) (method), 313  
[generate\\_morphology\(\)](#) (in module [lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)), 227  
[lensdk.model.biophysical.utils.AllActiveUtils](#) (method), 354  
[generate\\_morphology\(\)](#) (in module [lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache.EcephysProjectCache](#)), 160  
[lensdk.model.biophysical.utils.Utils](#) (method), 355  
[generate\\_output\\_cell\\_features\(\)](#) (in module [lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)), 227  
[lensdk.internal.ephys.core\\_feature\\_extract](#) (in module [lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache.EcephysProjectCache](#)), 305  
[generate\\_rays\(\)](#) (in module [lensdk.internal.brain\\_observatory.itracker\\_utils](#)), 295  
[generate\\_warp\\_coordinates\(\)](#) (in module [lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)), 228  
[lensdk.brain\\_observatory.stimulus\\_info.ExperimentGeometry](#) (method), 209  
[get\\_all\\_ages\(\)](#) (in module [lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache.EcephysProjectCache](#)), 160  
[get\\_all\\_bits\(\)](#) (in module [lensdk.brain\\_observatory.sync\\_dataset.Dataset](#)), 213  
[get\\_all\\_cre\\_lines\(\)](#) (in module [lensdk.brain\\_observatory.sync\\_dataset.Dataset](#)), 213  
[get\\_all\\_events\(\)](#) (in module [lensdk.brain\\_observatory.sync\\_dataset.Dataset](#)), 213  
[get\\_all\\_features\(\)](#) (in module [lensdk.core.cell\\_types\\_cache.CellTypesCache](#)), 236  
[get\\_all\\_full\\_genotypes\(\)](#) (in module [lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache.EcephysProjectCache](#)), 160  
[get\\_all\\_imaging\\_depths\(\)](#) (in module [lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)), 227  
[get\\_all\\_reporter\\_lines\(\)](#) (in module [lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)), 227  
[get\\_all\\_session\\_types\(\)](#) (in module [lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache.EcephysProjectCache](#)), 160  
[get\\_all\\_session\\_types\(\)](#) (in module [lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)), 227  
[get\\_all\\_sexes\(\)](#) (in module [lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache.EcephysProjectCache](#)), 160  
[get\\_all\\_stimuli\(\)](#) (in module [lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)), 228  
[get\\_all\\_targeted\\_structures\(\)](#) (in module [lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)), 228

method), 228

get\_all\_times() (al-  
lensdk.brain\_observatory.sync\_dataset.Dataset  
method), 213

get\_analog\_channel() (al-  
lensdk.brain\_observatory.sync\_dataset.Dataset  
method), 213

get\_analog\_meta() (al-  
lensdk.brain\_observatory.sync\_dataset.Dataset  
method), 213

get\_ancestor\_id\_map() (al-  
lensdk.core.structure\_tree.StructureTree  
method), 258

get\_ancestor\_id\_map() (in module al-  
lensdk.internal.mouse\_connectivity.interval\_unionize.run\_tibensdk.brain\_observatory.sync\_dataset.Dataset  
method), 337

get\_annotated\_section\_data\_sets() (al-  
lensdk.api.queries.annotated\_section\_data\_sets\_api.AnnotatedSectionDataSetsApi  
method), 43

get\_annotated\_section\_data\_sets\_via\_rma() (al-  
lensdk.api.queries.annotated\_section\_data\_sets\_api.AnnotatedSectionDataSetsApi  
method), 44

get\_annotation\_volume() (al-  
lensdk.core.reference\_space\_cache.ReferenceSpaceCache  
method), 252

get\_api\_list\_by\_container\_id() (al-  
lensdk.internal.api.behavior\_ophys\_api.BehaviorOphysLimsApi  
class method), 284

get\_atlases() (al-  
lensdk.api.queries.ontologies\_api.OntologiesApi  
method), 61

get\_atlases\_table() (al-  
lensdk.api.queries.ontologies\_api.OntologiesApi  
method), 61

get\_attribute\_dict() (in module al-  
lensdk.brain\_observatory.receptive\_field\_analysis.receptive\_field\_utilities, 180

get\_attribute\_dict() (in module al-  
lensdk.brain\_observatory.receptive\_field\_analysis.utilities), 180

get\_average\_intensity\_projection\_image\_file() (al-  
lensdk.internal.api.ophys\_lims\_api.OphysLimsApi  
method), 286

get\_average\_projection() (al-  
lensdk.brain\_observatory.behavior.behavior\_ophys\_api.BehaviorOphysLimsApi  
method), 80

get\_average\_projection() (al-  
lensdk.brain\_observatory.behavior.behavior\_ophys\_api.BehaviorOphysLimsApi  
method), 81

get\_average\_projection() (al-  
lensdk.brain\_observatory.behavior.behavior\_ophys\_session.BehaviorOphysSession, 93

get\_average\_projection() (al-  
lensdk.brain\_observatory.behavior.internal.behavior\_ophys\_session.BehaviorOphysSession, 93

method), 83

get\_average\_projection() (al-  
lensdk.internal.api.behavior\_ophys\_api.BehaviorOphysLimsApi  
method), 285

get\_barcode\_table() (al-  
lensdk.brain\_observatory.ecephys.align\_timestamps.barcode\_sync, 114

get\_behavior\_experiment\_id() (al-  
lensdk.internal.api.behavior\_lims\_api.BehaviorLimsApi  
method), 284

get\_behavior\_monitoring() (in module al-  
lensdk.brain\_observatory.behavior.sync, 86

get\_behavior\_only\_session\_data() (al-  
lensdk.brain\_observatory.behavior.behavior\_project\_lims\_api.BehaviorProjectLimsApi  
method), 97

get\_behavior\_only\_session\_data() (al-  
lensdk.brain\_observatory.behavior.internal.behavior\_project\_base.BehaviorProjectBase, 85

get\_behavior\_only\_session\_table() (al-  
lensdk.brain\_observatory.behavior.behavior\_project\_lims\_api.BehaviorProjectLimsApi  
method), 97

get\_behavior\_only\_session\_table() (al-  
lensdk.brain\_observatory.behavior.internal.behavior\_project\_base.BehaviorProjectBase, 85

get\_behavior\_session\_data() (al-  
lensdk.brain\_observatory.behavior.behavior\_project\_cache.BehaviorProjectCache, 96

get\_behavior\_session\_id() (al-  
lensdk.internal.api.behavior\_data\_lims\_api.BehaviorDataLimsApi  
method), 282

get\_behavior\_session\_table() (al-  
lensdk.brain\_observatory.behavior.behavior\_project\_cache.BehaviorProjectCache, 96

get\_behavior\_session\_uuid() (al-  
lensdk.internal.api.behavior\_data\_lims\_api.BehaviorDataLimsApi  
method), 282

get\_behavior\_session\_uuid() (al-  
lensdk.internal.api.behavior\_ophys\_api.BehaviorOphysLimsApi  
method), 285

get\_behavior\_stimulus\_file() (al-  
lensdk.internal.api.behavior\_data\_lims\_api.BehaviorDataLimsApi  
method), 282

get\_behavior\_stimulus\_file() (al-  
lensdk.internal.api.behavior\_ophys\_api.BehaviorOphysLimsApi  
method), 284

get\_behavior\_stimulus\_file() (al-  
lensdk.internal.api.behavior\_ophys\_api.BehaviorOphysLimsApi  
method), 285

get\_behavior\_tracking\_video\_filepath\_df() (al-  
lensdk.internal.api.behavior\_data\_lims\_api.BehaviorDataLimsApi  
method), 285

get\_behavior\_training\_df() (al-  
lensdk.brain\_observatory.behavior.oplms\_base.OplmsBase, 93



<code>get_behavior_training_df()</code> ( <i>allensdk.internal.api.mtrain_api.MtrainSqlApi</i> method), 286	<code>get_cell_specimen_indices()</code> ( <i>allensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession</i> method), 93
<code>get_birth_date()</code> ( <i>allensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi</i> method), 282	<code>get_cell_specimen_indices()</code> ( <i>allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet</i> method), 231
<code>get_bit()</code> ( <i>allensdk.brain_observatory.sync_dataset.Dataset</i> method), 214	<code>get_cell_specimen_table()</code> ( <i>allensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_session.BehaviorOphysSession</i> method), 80
<code>get_bit()</code> (in <i>module</i> <i>allensdk.brain_observatory.sync_dataset</i> ), 216	<code>get_cell_specimen_table()</code> ( <i>allensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysSession</i> method), 81
<code>get_bit_changes()</code> ( <i>allensdk.brain_observatory.sync_dataset.Dataset</i> method), 214	<code>get_cell_specimen_table()</code> ( <i>allensdk.brain_observatory.behavior.internal.behavior_ophys_base.BehaviorOphysBase</i> method), 83
<code>get_blend()</code> (in <i>module</i> <i>allensdk.internal.mouse_connectivity.tissuecyte_stitching_stitcher</i> ), 342	<code>get_cell_specimen_table()</code> ( <i>allensdk.internal.api.ophys_lims_api.OphysLimsApi</i> method), 286
<code>get_blend_component()</code> (in <i>module</i> <i>allensdk.internal.mouse_connectivity.tissuecyte_stitching_stitcher</i> ), 342	<code>get_cell_specimens()</code> ( <i>allensdk.core.brain_observatory_cache.BrainObservatoryCache</i> method), 228
<code>get_blocks()</code> (in <i>module</i> <i>allensdk.brain_observatory.ecephys.stimulus_table.visualization.view(allensdk)</i> ), 148	<code>get_cell_types_cache()</code> ( <i>allensdk.core.cell_types_cache.CellTypesCache</i> method), 236
<code>get_cache_path()</code> ( <i>allensdk.api.cache.Cache</i> method), 76	<code>get_cells()</code> ( <i>allensdk.internal.model.biophysical.biophysical_archiver.BiophysicalArchiver</i> method), 312
<code>get_cap_check_indices()</code> (in <i>module</i> <i>allensdk.internal.model.biophysical.passive_fitting.preprocessor</i> ), 312	<code>get_change_time_frame_response_latency()</code> (in <i>module</i> <i>allensdk.brain_observatory.behavior.trials_processing</i> ), 109
<code>get_catch_responses()</code> (in <i>module</i> <i>allensdk.brain_observatory.behavior.dprime</i> ), 100	<code>get_channels()</code> ( <i>allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api</i> method), 117
<code>get_cav_density()</code> (in <i>module</i> <i>allensdk.internal.mouse_connectivity.interval_unionize_data_utilities</i> ), 335	<code>get_channels()</code> ( <i>allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api</i> method), 117
<code>get_cell()</code> ( <i>allensdk.api.queries.cell_types_api.CellTypesApi</i> method), 49	<code>get_channels()</code> ( <i>allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api</i> method), 118
<code>get_cell_metrics()</code> ( <i>allensdk.api.queries.brain_observatory_api.BrainObservatoryApi</i> method), 47	<code>get_channels()</code> ( <i>allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api</i> method), 118
<code>get_cell_metrics()</code> ( <i>allensdk.internal.api.queries.pre_release.BrainObservatoryApiPreRelease</i> method), 281	<code>get_channels()</code> ( <i>allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api</i> method), 118
<code>get_cell_roi_ids()</code> ( <i>allensdk.internal.api.ophys_lims_api.OphysLimsApi</i> method), 286	<code>get_channels()</code> ( <i>allensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache</i> method), 160
<code>get_cell_specimen_id_mapping()</code> ( <i>allensdk.api.queries.brain_observatory_api.BrainObservatoryApi</i> method), 47	<code>get_channels()</code> ( <i>allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api</i> method), 126
<code>get_cell_specimen_ids()</code> ( <i>allensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession</i> method), 93	<code>get_channels()</code> ( <i>allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api</i> method), 126
<code>get_cell_specimen_ids()</code> ( <i>allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet</i> method), 231	<code>get_channels()</code> ( <i>allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api</i> method), 126

`get_child_ids()` (*allensdk.core.ontology.Ontology* *method*), 242  
`get_children()` (*allensdk.core.ontology.Ontology* *method*), 248  
`get_colormap()` (*allensdk.core.structure\_tree.StructureTree* *method*), 258  
`get_column_definitions()` (*allensdk.api.queries.brain\_observatory\_api.BrainObservatoryApi* *method*), 47  
`get_components()` (*in module allensdk.brain\_observatory.receptive\_field\_analysis.utilities*), *method*), 231  
`get_compound_annotated_section_data_sets()` (*allensdk.api.queries.annotated\_section\_data\_sets\_api.AnnotatedSectionDataSetsApi* *method*), 44  
`get_config()` (*allensdk.config.manifest\_builder.ManifestBuilder* *method*), 224  
`get_connection()` (*allensdk.internal.api.PostgresQueryMixin* *method*), 287  
`get_containers_df()` (*allensdk.internal.api.behavior\_ophys\_api.BehaviorOphysLimsApi* *static method*), 285  
`get_corrected_fluorescence_traces()` (*allensdk.brain\_observatory.behavior.behavior\_ophys\_api.behavior\_ophys\_nwb\_api.BehaviorOphysNwbApi* *method*), 80  
`get_corrected_fluorescence_traces()` (*allensdk.brain\_observatory.behavior.behavior\_ophys\_api.BehaviorOphysBase* *method*), 81  
`get_corrected_fluorescence_traces()` (*allensdk.brain\_observatory.behavior.internal.behavior\_ophys\_base.BehaviorOphysBase* *method*), 83  
`get_corrected_fluorescence_traces()` (*allensdk.core.brain\_observatory\_nwb\_data\_set.BrainObservatoryNwbDataSet* *method*), 231  
`get_corrected_fluorescence_traces()` (*allensdk.internal.api.behavior\_ophys\_api.BehaviorOphysLimsApi* *method*), 285  
`get_credential_provider()` (*in module allensdk.core.authentication*), 226  
`get_current_source_density()` (*allensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession* *method*), 164  
`get_current_source_density()` (*allensdk.brain\_observatory.ecephys.ecephys\_session\_api.ecephys\_nwb\_session\_api.EcephysNwbSessionApi* *method*), 126  
`get_current_stage()` (*allensdk.internal.api.mtrain\_api.MtrainApi* *method*), 286  
`get_cursor()` (*allensdk.internal.api.PostgresQueryMixin* *method*), 287  
`get_data_mask()` (*allensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache* *method*), 243  
`get_default_manifest_file()` (*in module allensdk.api.cache*), 79  
`get_deformation_field()` (*allensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache* *method*), 243  
`get_demix_file()` (*allensdk.internal.api.ophys\_lims\_api.OphysLimsApi* *method*), 286  
`get_demixed_traces()` (*allensdk.core.brain\_observatory\_nwb\_data\_set.BrainObservatoryNwbDataSet* *method*), 231  
`get_descendant_ids()` (*allensdk.core.ontology.Ontology* *method*), 248  
`get_descendants()` (*allensdk.core.ontology.Ontology* *method*), 248  
`get_df()` (*allensdk.internal.api.mtrain\_api.MtrainApi* *method*), 286  
`get_dff_file()` (*allensdk.internal.api.ophys\_lims\_api.OphysLimsApi* *method*), 286  
`get_dff_traces()` (*allensdk.brain\_observatory.behavior.behavior\_ophys\_api.behavior\_ophys\_nwb\_api.BehaviorOphysNwbApi* *method*), 80  
`get_dff_traces()` (*allensdk.brain\_observatory.behavior.behavior\_ophys\_api.BehaviorOphysBase* *method*), 81  
`get_dff_traces()` (*allensdk.brain\_observatory.behavior.internal.behavior\_ophys\_base.BehaviorOphysBase* *method*), 83  
`get_dff_traces()` (*allensdk.core.brain\_observatory\_nwb\_data\_set.BrainObservatoryNwbDataSet* *method*), 231  
`get_dff_traces()` (*allensdk.internal.api.behavior\_ophys\_api.BehaviorOphysLimsApi* *method*), 285  
`get_dff_traces()` (*allensdk.internal.api.behavior\_ophys\_api.BehaviorOphysLimsApi* *method*), 285  
`get_diagonals_from_sparse()` (*in module allensdk.brain\_observatory.r\_neuropil*), 198  
`get_dimensions()` (*allensdk.internal.morphology.morphology.Morphology* *method*), 386  
`get_disc_masks()` (*in module allensdk.brain\_observatory.receptive\_field\_analysis.chisquarperf*), 176  
`get_dprime()` (*in module allensdk.brain\_observatory.behavior.dprime*), 100  
`get_driver_line()` (*allensdk.internal.api.behavior\_data\_lims\_api.BehaviorDataLimsApi* *method*), 286

<code>method)</code> , 282	<code>get_expected_events_by_pixel()</code>
<code>get_driver_line()</code> (al- lensdk.internal.api.ophys_lims_api.OphysLimsApi method), 286	(in module al- lensdk.brain_observatory.receptive_field_analysis.chisquarperf), 176
<code>get_ecephys_session_id()</code> (al- lensdk.brain_observatory.ecephys.ecephys_session_api.EcephysSessionApi method), 126	<code>get_experiment_analysis_file()</code>
<code>get_ecephys_session_id()</code> (al- lensdk.brain_observatory.ecephys.ecephys_session_api.EcephysSessionApi method), 126	(in module al- lensdk.internal.pipeline_modules.run_observatory_thumbnails), 348
<code>get_ecephys_session_id()</code> (al- lensdk.brain_observatory.ecephys.ecephys_session_api.EcephysSessionApi method), 126	<code>get_ecephys_nwb_session_api.EcephysNwbSessionApi</code> (al- lensdk.internal.pipeline_modules.run_observatory_thumbnails), 348
<code>get_ecephys_session_id()</code> (al- lensdk.brain_observatory.ecephys.ecephys_session_api.EcephysSessionApi method), 127	<code>get_ecephys_nwb_session_api.EcephysNwbSessionApi</code> (al- lensdk.internal.pipeline_modules.run_observatory_thumbnails), 348
<code>get_edges()</code> (allensdk.brain_observatory.sync_dataset.Dataset method), 214	<code>get_experiment_containers()</code> (al- lensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 48
<code>get_ephys_data()</code> (al- lensdk.core.cell_types_cache.CellTypesCache method), 237	<code>get_experiment_containers()</code> (al- lensdk.core.brain_observatory_cache.BrainObservatoryCache method), 228
<code>get_ephys_features()</code> (al- lensdk.api.queries.cell_types_api.CellTypesApi method), 50	<code>get_experiment_containers()</code> (al- lensdk.internal.api.queries.pre_release.BrainObservatoryApiPreRelease method), 281
<code>get_ephys_features()</code> (al- lensdk.core.cell_types_cache.CellTypesCache method), 237	<code>get_experiment_date()</code> (al- lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi method), 282
<code>get_ephys_sweeps()</code> (al- lensdk.api.queries.cell_types_api.CellTypesApi method), 50	<code>get_experiment_date()</code> (al- lensdk.internal.api.ophys_lims_api.OphysLimsApi method), 286
<code>get_ephys_sweeps()</code> (al- lensdk.api.queries.glif_api.GlifApi method), 52	<code>get_experiment_detail()</code> (al- lensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 61
<code>get_ephys_sweeps()</code> (al- lensdk.core.cell_types_cache.CellTypesCache method), 237	<code>get_experiment_files()</code> (in module al- lensdk.internal.pipeline_modules.run_observatory_thumbnails), 348
<code>get_epoch_mask_list()</code> (in module al- lensdk.core.brain_observatory_nwb_data_set), 235	<code>get_experiment_id()</code> (al- lensdk.internal.api.lims_api.LimsApi method), 285
<code>get_equipment_id()</code> (al- lensdk.internal.api.ophys_lims_api.OphysLimsApi method), 286	<code>get_experiment_nwb_file()</code> (in module al- lensdk.internal.pipeline_modules.run_observatory_analysis), 346
<code>get_even_sampling()</code> (in module al- lensdk.brain_observatory.behavior.trials_processing), 109	<code>get_experiment_nwb_file()</code> (in module al- lensdk.internal.pipeline_modules.run_observatory_thumbnails), 348
<code>get_events_by_bit()</code> (al- lensdk.brain_observatory.sync_dataset.Dataset method), 214	<code>get_experiment_session()</code> (in module al- lensdk.internal.pipeline_modules.run_observatory_analysis), 346
<code>get_events_by_line()</code> (al- lensdk.brain_observatory.sync_dataset.Dataset method), 214	<code>get_experiment_structure_unionizes()</code> (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method), 243
<code>get_events_per_pixel()</code> (in module al- lensdk.brain_observatory.receptive_field_analysis.chisquarperf), 176	<code>get_experiment_sweep_numbers()</code> (al- lensdk.core.nwb_data_set.NwbDataSet method), 298
<code>get_excluded()</code> (al- lensdk.internal.brain_observatory.roi_filter_utils.TrainingModelClassifier method), 298	<code>get_experiment_table()</code> (al-



```

lensdk.brain_observatory.behavior.behavior_project_cache.BehaviorProjectCache), 96
lensdk.brain_observatory.behavior.behavior_project_cache.BehaviorProjectCache), 100
get_experiment_table() (al- get_features() (in module al-
lensdk.brain_observatory.behavior.behavior_project_lims_api.BehaviorProjectLimsApi), 306
method), 97 get_features() (in module al-
get_experiments() (al- lensdk.internal.ephys.plot_qc_figures3),
lensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi
method), 61 get_field_of_view_shape() (al-
get_experiments() (al- lensdk.internal.api.ophys_lims_api.OphysLimsApi
lensdk.core.mouse_connectivity_cache.MouseConnectivityCache), 286
method), 243 get_fluorescence() (al-
get_experiments() (al- lensdk.brain_observatory.stimulus_analysis.StimulusAnalysis
lensdk.internal.api.queries.mouse_connectivity_api_prerelease_api.MouseConnectivityApiPrerelease
method), 279 get_fluorescence_timestamps() (al-
get_experiments() (al- lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet
lensdk.internal.core.mouse_connectivity_cache_prerelease_api.MouseConnectivityCachePrerelease
method), 304 get_fluorescence_traces() (al-
get_experiments_api() (al- lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet
lensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi), 232
method), 61 get_foraging_id() (al-
get_extended_trials() (al- lensdk.internal.api.ophys_lims_api.OphysLimsApi
lensdk.internal.api.behavior_lims_api.BehaviorLimsApi method), 286
method), 284 get_format() (allensdk.config.manifest.Manifest
get_extended_trials() (al- method), 223
lensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi (in module al-
method), 285 lensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis
get_extended_trials() (in module al- 146
lensdk.brain_observatory.behavior.trials_processing), 109 get_full_genotype() (al-
lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi
method), 282 get_genotype() (al-
lensdk.internal.api.ophys_lims_api.OphysLimsApi
method), 286 get_gaussian_fit() (in module al-
lensdk.internal.api.ophys_lims_api.OphysLimsApi lensdk.brain_observatory.receptive_field_analysis.postprocessing
method), 286 179
get_eye_tracking() (al- lensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi
lensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi an_fit_single_channel()
method), 285 (in module al-
get_eye_tracking() (in module al- lensdk.brain_observatory.receptive_field_analysis.fit_parameters
lensdk.brain_observatory.behavior.sync), 178
86 get_genes() (allensdk.api.queries.mouse_atlas_api.MouseAtlasApi
method), 57
get_eye_tracking_data() (al- lensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysApiBase (in module al-
lensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysApiBase (in module al-
method), 81 lensdk.brain_observatory.behavior.dprime),
get_eye_tracking_filepath() (al- 100
lensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi
method), 285 get_prepared_uniques() (in module al-
lensdk.brain_observatory.ecephys.ecephys_project_cache),
get_eye_tracking_video_filepath_df() 161
(allensdk.internal.api.lims_api.LimsApi get_h() (in module al-
method), 286 lensdk.internal.model.biophysical.passive_fitting.neuron_utils),
get_falling_edges() (al- 311
lensdk.brain_observatory.sync_dataset.Dataset get_hit_rate() (in module al-
method), 214 lensdk.brain_observatory.behavior.dprime),
get_false_alarm_rate() (in module al- 100

```

`get_id()` (`allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.IDCreator` method), 126  
`get_id_acronym_map()` (`allensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` method), 165  
`get_image()` (`allensdk.brain_observatory.nwb.nwb_api.NwbApi` method), 144  
`get_image_info_from_trial()` (`in module allensdk.brain_observatory.behavior.trials_processing`), 109  
`get_image_region()` (`allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile` method), 127  
`get_image_to_atlas()` (`allensdk.api.queries.synchronization_api.SynchronizationApi` method), 70  
`get_image_to_image()` (`allensdk.api.queries.synchronization_api.SynchronizationApi` method), 71  
`get_image_to_image_2d()` (`allensdk.api.queries.synchronization_api.SynchronizationApi` method), 71  
`get_image_to_reference()` (`allensdk.api.queries.synchronization_api.SynchronizationApi` method), 71  
`get_images_dict()` (`in module allensdk.brain_observatory.behavior.stimulus_processing`), 108  
`get_imaging_depth()` (`allensdk.internal.api.ophys_lims_api.OphysLimsApi` method), 286  
`get_indicator_bound_point()` (`in module allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitching`), 342  
`get_indices_by_distance()` (`in module allensdk.internal.brain_observatory.roi_filter_utils`), 299  
`get_injection_data()` (`in module allensdk.internal.mouse_connectivity.interval_union_utils`), 335  
`get_injection_density()` (`allensdk.core.mouse_connectivity_cache.MouseConnectivityCache` method), 244  
`get_injection_fraction()` (`allensdk.core.mouse_connectivity_cache.MouseConnectivityCache` method), 244  
`get_input_data()` (`in module allensdk.internal.pipeline_modules.run_observatory_thumbnails`), 348  
`get_input_json()` (`in module allensdk.internal.core.lims_utilities`), 302  
`get_intensity()` (`allensdk.mouse_connectivity.grid.subimage.base_subimage.InteractiveSubImage` method), 366  
`get_intrinsic_timescale()` (`allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis` method), 144  
`get_invalid_times()` (`allensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` method), 165  
`get_isi_experiments()` (`allensdk.api.queries.brain_observatory_api.BrainObservatoryApi` method), 48  
`get_isi_experiments()` (`allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api` method), 117  
`get_isi_experiments()` (`allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api` method), 117  
`get_isis()` (`in module allensdk.ephys.ephys_features`), 275  
`get_keys()` (`in module allensdk.internal.brain_observatory.time_sync`), 300  
`get_lfp()` (`allensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` method), 165  
`get_lfp()` (`allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api` method), 126  
`get_lfp()` (`allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api` method), 127  
`get_lfp_channel_order()` (`allensdk.brain_observatory.ecephys.file_io.continuous_file.ContinuousFile` method), 128  
`get_lfp_channels()` (`in module allensdk.brain_observatory.behavior.sync`), 87  
`get_lfp_channels()` (`allensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_api` method), 80  
`get_licks()` (`allensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_api` method), 81  
`get_licks()` (`allensdk.brain_observatory.behavior.internal.behavior_internal` method), 82  
`get_licks()` (`allensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi` method), 282  
`get_licks()` (`allensdk.internal.api.behavior_ophys_api.BehaviorOphysApi` method), 285  
`get_line()` (`allensdk.brain_observatory.sync_dataset.Dataset` method), 366  
`get_line_changes()` (`allensdk.brain_observatory.sync_dataset.Dataset` method), 366

`lensdk.brain_observatory.sync_dataset.Dataset` `get_metadata()` (al-  
 method), 215 `lensdk.brain_observatory.behavior.behavior_ophys_api.Behavior`  
`get_list_of_path_dict()` (in module al- `method)`, 81  
`lensdk.test_utilities.regression_fixture)`, 373 `get_metadata()` (al-  
`get_locally_sparse_noise_stimulus_template()` `lensdk.brain_observatory.behavior.internal.behavior_ophys_base`  
 (allensdk.core.brain\_observatory\_nwb\_data\_set.BrainObservatoryNWBDataSet  
 method), 232 `get_metadata()` (al-  
`get_manifest()` (al- `lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_`  
`lensdk.config.manifest_builder.ManifestBuilder` `method)`, 126  
 method), 224 `get_metadata()` (al-  
`get_manual_injection_summary()` (al- `lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_`  
`lensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi`, 127  
 method), 61 `get_metadata()` (al-  
`get_mask()` (allensdk.brain\_observatory.stimulus\_info.Monitor `lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN`  
 method), 209 `method)`, 233  
`get_mask_plane()` (al- `get_metadata()` (al-  
`lensdk.brain_observatory.roi_masks.Mask` `lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi`  
 method), 199 `method)`, 283  
`get_max_projection()` (al- `get_metadata()` (al-  
`lensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_api.behavior_ophys_api.behavior_ophys_lims_api`  
 method), 80 `method)`, 285  
`get_max_projection()` (al- `get_metadata()` (al-  
`lensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysApi` `lensdk.brain_observatory.behavior_ophys_lims_api.OphysLimsApi`  
 method), 81 `method)`, 287  
`get_max_projection()` (al- `get_metrics()` (in module al-  
`lensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession` `lensdk.brain_observatory.annotated_region_metrics)`,  
 method), 93 288  
`get_max_projection()` (al- `get_missing_path()` (al-  
`lensdk.brain_observatory.behavior.internal.behavior_ophys_lims_api.BehaviorOphysLimsApi` `lensdk.brain_observatory.behavior_ophys_session.BehaviorOphysSession`  
 method), 84 `method)`, 343  
`get_max_projection()` (al- `get_morphology_features()` (al-  
`lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNWBDataSet` `lensdk.core.cell_types_api.CellTypesApi`  
 method), 232 `method)`, 50  
`get_max_projection()` (al- `get_morphology_features()` (al-  
`lensdk.internal.api.ophys_lims_api.OphysLimsApi` `lensdk.core.cell_types_cache.CellTypesCache`  
 method), 287 `method)`, 237  
`get_max_projection_file()` (al- `get_motion_corrected_image_stack_file()`  
`lensdk.internal.api.ophys_lims_api.OphysLimsApi` (allensdk.internal.api.ophys\_lims\_api.OphysLimsApi  
 method), 287 `method)`, 287  
`get_mean_response()` (al- `get_motion_correction()` (al-  
`lensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` `lensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_api`  
 method), 193 `method)`, 81  
`get_mean_waveforms()` (al- `get_motion_correction()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.ecephys_session_api.ecephys_session_api`  
 method), 126 `method)`, 81  
`get_mean_waveforms()` (al- `get_motion_correction()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.ecephys_session_api.ecephys_session_api`  
 method), 126 `method)`, 84  
`get_mean_waveforms()` (al- `get_motion_correction()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.ecephys_session_api.ecephys_session_api`  
 method), 127 `method)`, 233  
`get_metadata()` (al- `get_motion_correction()` (al-  
`lensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_api.behavior_ophys_api.behavior_ophys_lims_api`  
 method), 81 `method)`, 285

`get_mouse_id()` (in module `al- lensdk.brain_observatory.behavior.trials_processing`), `lensdk.api.queries.glif_api.GlifApi` method), 109 52

`get_name_map()` (al- `lensdk.core.structure_tree.StructureTree` method), 258 (al- `lensdk.api.queries.glif_api.GlifApi` method), 52

`get_natural_movie_template()` (al- `lensdk.brain_observatory.behavior.behavior_project_lims_data_set.BrainObservatoryNwbDataSet` method), 98 (al- `lensdk.brain_observatory.behavior.project_lims_data_set.BrainObservatoryNwbDataSet` method), 277

`get_natural_movie_template()` (al- `lensdk.brain_observatory.behavior.internal.behavior_project_lims_data_set.BrainObservatoryNwbDataSet` method), 85 (al- `lensdk.api.queries.glif_api.GlifApi` method), 52

`get_natural_movie_template()` (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.BiophysicalModuleApi` method), 117 (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.BiophysicalModuleApi` method), 45

`get_natural_movie_template()` (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.BiophysicalModuleApi` method), 117 (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.BiophysicalModuleApi` method), 52

`get_natural_movie_template()` (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.BiophysicalModuleApi` method), 119 (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.BiophysicalModuleApi` method), 277

`get_natural_movie_template()` (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.BiophysicalModuleApi` method), 121 (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.BiophysicalModuleApi` method), 312

`get_natural_movie_template()` (al- `lensdk.brain_observatory.ecephys.ecephys_project_cache.ecephys_project_cache.BiophysicalModuleApi` method), 160 (al- `lensdk.brain_observatory.ecephys.ecephys_project_cache.ecephys_project_cache.BiophysicalModuleApi` method), 52

`get_natural_scene_template()` (al- `lensdk.brain_observatory.behavior.behavior_project_lims_data_set.BrainObservatoryNwbDataSet` method), 98 (al- `lensdk.brain_observatory.behavior.project_lims_data_set.BrainObservatoryNwbDataSet` method), 233

`get_natural_scene_template()` (al- `lensdk.brain_observatory.behavior.internal.behavior_project_lims_data_set.BrainObservatoryNwbDataSet` method), 85 (al- `lensdk.brain_observatory.behavior.project_lims_data_set.BrainObservatoryNwbDataSet` method), 233

`get_natural_scene_template()` (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.DriftingGratings` method), 117 (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.DriftingGratings` method), 191

`get_natural_scene_template()` (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.DriftingGratings` method), 117 (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.DriftingGratings` method), 194

`get_natural_scene_template()` (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.DriftingGratings` method), 119 (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.DriftingGratings` method), 205

`get_natural_scene_template()` (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.DriftingGratings` method), 121 (al- `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.DriftingGratings` method), 229

`get_natural_scene_template()` (al- `lensdk.brain_observatory.ecephys.ecephys_project_cache.ecephys_project_cache.BrainObservatoryNwbDataSet` method), 160 (al- `lensdk.brain_observatory.ecephys.ecephys_project_cache.ecephys_project_cache.BrainObservatoryNwbDataSet` method), 285

`get_nearest()` (al- `lensdk.brain_observatory.sync_dataset.Dataset` method), 215 (al- `lensdk.internal.api.ophys_lims_api.OphysLimsApi` method), 287

`get_neuron_config()` (al- `lensdk.api.queries.glif_api.GlifApi` method), 52 (al- `lensdk.internal.api.ophys_lims_api.OphysLimsApi` method), 287

Index 433



`lensdk.internal.model.glif.threshold_adaptation)`, `get_projection_matrix()` (al-  
 326 `lensdk.core.mouse_connectivity_cache.MouseConnectivityCache`  
`get_performance_metrics()` (al- `method`), 244  
`lensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession` (al-  
`method`), 93 `lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession`  
`get_photodiode_events()` (in module al- `method`), 166  
`lensdk.internal.brain_observatory.time_sync)`, `get_pupil_data()` (al-  
 301 `lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_`  
`get_pipeline_version()` (al- `method`), 127  
`lensdk.core.nwb_data_set.NwbDataSet` `get_pupil_data()` (al-  
`method`), 246 `lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_`  
`get_polygons()` (al- `method`), 127  
`lensdk.mouse_connectivity.grid.subimage.base_subimage.PolygonSubImage` (al-  
`method`), 366 `lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet`  
`get_probe_lfp_data()` (al- `method`), 233  
`lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi` (al-  
`method`), 117 `lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet`  
`get_probe_lfp_data()` (al- `method`), 233  
`lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_fixed_api.EcephysProjectFixedApi` (al-  
`method`), 117 `lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi`  
`get_probe_lfp_data()` (al- `get_raw_cell_specimen_table_dict()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi` `lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi`  
`method`), 119 `method`), 287  
`get_probe_lfp_data()` (al- `get_raw_dff_data()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi` `lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi`  
`method`), 121 `method`), 287  
`get_probe_time_offset()` (in module al- `get_raw_ophys_file_shape()` (in module al-  
`lensdk.brain_observatory.ecephys.align_timestamps.barcode)`, `lensdk.brain_observatory.behavior.validation`),  
 113 111  
`get_probes()` (`allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi`  
`method`), 117 `lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi`  
`get_probes()` (`allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_fixed_api.EcephysProjectFixedApi`  
`method`), 117 `get_raw_stimulus_frames()` (in module al-  
`lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi`  
`method`), 119 87  
`get_probes()` (`allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi`  
`method`), 121 `lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi`  
`get_probes()` (`allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi`  
`method`), 160 `lensdk.brain_observatory.behavior.internal.behavior_ophys_base`  
`get_real_photodiode_events()` (in module al-  
`lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi`  
`method`), 126 301  
`get_probes()` (`allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi`  
`method`), 127 `lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_fie`  
`get_projection_data()` (in module al- `lensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise`  
`lensdk.internal.mouse_connectivity.interval_unionize.data_utilities`), 193  
 336 `get_receptive_field_analysis_data()` (al-  
`lensdk.core.mouse_connectivity_cache.MouseConnectivityCache` `method`), 193  
`method`), 244 `get_receptive_field_attribute_df()` (al-  
`lensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise`  
`get_projection_image_info()` (al- `lensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise`  
`lensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi` `method`), 193  
`method`), 61 `get_reconstruction()` (al-

<code>lensdk.core.cell_types_cache.CellTypesCache</code> <code>method</code> ), 237	<code>lensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_api</code> <code>method</code> ), 81
<code>get_reconstruction_markers()</code> (alias of <code>lensdk.core.cell_types_cache.CellTypesCache</code> <code>method</code> ), 237	<code>get_rewards()</code> (alias of <code>lensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysSession</code> <code>method</code> ), 81
<code>get_recorded_data()</code> (alias of <code>lensdk.model.biophysical.utils.Utils</code> <code>method</code> ), 355	<code>get_rewards()</code> (alias of <code>lensdk.brain_observatory.behavior.internal.behavior_base.BehaviorBase</code> <code>method</code> ), 82
<code>get_reference_aligned_image_channel_volumes_new()</code> (alias of <code>lensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi</code> <code>method</code> ), 61	<code>get_rewards()</code> (alias of <code>lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi</code> <code>method</code> ), 283
<code>get_reference_space()</code> (alias of <code>lensdk.core.reference_space_cache.ReferenceSpaceCache</code> <code>method</code> ), 252	<code>get_rewards()</code> (alias of <code>lensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi</code> <code>method</code> ), 285
<code>get_reference_to_image()</code> (alias of <code>lensdk.api.queries.synchronization_api.SynchronizationApi</code> <code>method</code> ), 71	<code>get_rewards()</code> (in module <code>lensdk.brain_observatory.behavior.rewards_processing</code> ), 106
<code>get_reporter_line()</code> (alias of <code>lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi</code> <code>method</code> ), 283	<code>get_rig_metadata()</code> (alias of <code>lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api</code> <code>method</code> ), 127
<code>get_reporter_line()</code> (alias of <code>lensdk.internal.api.ophys_lims_api.OphysLimsApi</code> <code>method</code> ), 287	<code>get_rig_metadata()</code> (alias of <code>lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api</code> <code>method</code> ), 127
<code>get_representational_similarity()</code> (alias of <code>lensdk.brain_observatory.drifting_gratings.DriftingGratings</code> <code>method</code> ), 192	<code>get_rig_name()</code> (alias of <code>lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi</code> <code>method</code> ), 283
<code>get_representational_similarity()</code> (alias of <code>lensdk.brain_observatory.natural_scenes.NaturalScenes</code> <code>method</code> ), 195	<code>get_rig_name()</code> (alias of <code>lensdk.internal.api.ophys_lims_api.OphysLimsApi</code> <code>method</code> ), 287
<code>get_representational_similarity()</code> (alias of <code>lensdk.brain_observatory.static_gratings.StaticGratings</code> <code>method</code> ), 206	<code>get_rigid_motion_transform_file()</code> (alias of <code>lensdk.internal.api.ophys_lims_api.OphysLimsApi</code> <code>method</code> ), 287
<code>get_response()</code> (alias of <code>lensdk.brain_observatory.drifting_gratings.DriftingGratings</code> <code>method</code> ), 192	<code>get_rising_edges()</code> (alias of <code>lensdk.brain_observatory.sync_dataset.Dataset</code> <code>method</code> ), 215
<code>get_response()</code> (alias of <code>lensdk.brain_observatory.natural_scenes.NaturalScenes</code> <code>method</code> ), 195	<code>get_rma()</code> (alias of <code>lensdk.brain_observatory.ecephys.ecephys_project_api.rma_engine</code> <code>method</code> ), 124
<code>get_response()</code> (alias of <code>lensdk.brain_observatory.static_gratings.StaticGratings</code> <code>method</code> ), 206	<code>get_rma_list()</code> (alias of <code>lensdk.brain_observatory.ecephys.ecephys_project_api.rma_engine</code> <code>method</code> ), 124
<code>get_response()</code> (alias of <code>lensdk.brain_observatory.stimulus_analysis.StimulusAnalysis</code> <code>method</code> ), 207	<code>get_rma_tabular()</code> (alias of <code>lensdk.brain_observatory.ecephys.ecephys_project_api.rma_engine</code> <code>method</code> ), 124
<code>get_response_latency()</code> (in module <code>lensdk.brain_observatory.behavior.trials_processing</code> ), 109	<code>get_roi_ids()</code> (alias of <code>lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet</code> <code>method</code> ), 233
<code>get_response_type()</code> (in module <code>lensdk.brain_observatory.behavior.trials_processing</code> ), 110	<code>get_roi_mask()</code> (alias of <code>lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet</code> <code>method</code> ), 234
<code>get_reward_rate()</code> (alias of <code>lensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession</code> <code>method</code> ), 93	<code>get_roi_mask_array()</code> (alias of <code>lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet</code> <code>method</code> ), 234
<code>get_rewards()</code> (alias of <code>lensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession</code> <code>method</code> ), 93	<code>get_roi_masks()</code> (alias of <code>lensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession</code> <code>method</code> ), 93





`get_session_data()` (al- `method`), 206  
`lensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache.get_session_data()` (in module `al-`  
`method`), 160  
`lensdk.core.sitk_utilities`), 256  
`get_session_start_time()` (al- `get_slice_image()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi.get_session_start_time()` (in module `al-`  
`method`), 127  
`method`), 250  
`get_session_start_time()` (al- `get_sparse_noise_epoch_mask_list()`  
`lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi.get_session_start_time()` (in module `al-`  
`method`), 127  
`lensdk.brain_observatory.receptive_field_analysis.utilities`),  
`get_session_table()` (al- 181  
`lensdk.brain_observatory.behavior.behavior_project_cache.BehaviorProjectCache.get_session_table()` (in module `al-`  
`method`), 96  
`lensdk.brain_observatory.stimulus_info`),  
`get_session_table()` (al- 210  
`lensdk.brain_observatory.behavior.behavior_project_data_lims_api.BehaviorProjectDataLimsApi.get_session_table()` (in module `al-`  
`method`), 98  
`allensdk.brain_observatory.stimulus_info`), 210  
`get_session_table()` (al- `get_speed_tuning()` (al-  
`lensdk.brain_observatory.behavior.internal.behavior_project_data_lims_api.BehaviorProjectDataLimsApi.get_session_table()` (in module `al-`  
`method`), 85  
`lensdk.brain_observatory.stimulus_analysis.StimulusAnalysis`  
`method`), 207  
`get_session_table()` (al- `get_spike_amplitudes()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache.get_session_table()` (in module `al-`  
`method`), 160  
`method`), 127  
`get_session_type()` (al- `get_spike_amplitudes()` (al-  
`lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet.get_session_type()` (in module `al-`  
`method`), 234  
`method`), 127  
`get_sessions()` (al- `get_spike_times()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi.get_sessions()` (in module `al-`  
`method`), 117  
`method`), 126  
`get_sessions()` (al- `get_spike_times()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi.get_sessions()` (in module `al-`  
`method`), 118  
`method`), 127  
`get_sessions()` (al- `get_spike_times()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi.get_sessions()` (in module `al-`  
`method`), 120  
`method`), 128  
`get_sessions()` (al- `get_spike_times()` (al-  
`lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi.get_sessions()` (in module `al-`  
`method`), 121  
`lensdk.brain_observatory.nwb_data_set.NwbDataSet.get_sessions()` (in module `al-`  
`method`), 246  
`get_sex()` (allensdk.internal.api.behavior\_data\_lims\_api.BehaviorDataLimsApi.get\_sex() (in module `al-`  
`method`), 283  
`lensdk.internal.ephys.plot_qc_figures`), 306  
`get_sex()` (allensdk.internal.api.ophys\_lims\_api.OphysLimsApi.get\_sex() (in module `al-`  
`method`), 287  
`lensdk.internal.ephys.plot_qc_figures3`),  
`get_sfdi()` (in module `al-` 308  
`lensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings.get_sfdi()` (in module `al-`  
143  
`module allensdk.ephys.extract_cell_features`),  
`get_shuffle_matrix()` (in module `al-` 276  
`lensdk.brain_observatory.receptive_field_analysis.utilities`),  
180  
`get_stim_characteristics()` (in module `al-`  
`get_signal_correlation()` (al- `lensdk.internal.model.biophysical.ephys_utils`),  
`lensdk.brain_observatory.drifting_gratings.DriftingGratings.get_signal_correlation()` (in module `al-`  
`method`), 192  
`method`), 13  
`get_stim_characteristics()` (in module `al-`  
`get_signal_correlation()` (al- `lensdk.ephys.extract_cell_features`), 276  
`lensdk.brain_observatory.natural_scenes.NaturalScenes.get_signal_correlation()` (in module `al-`  
`method`), 195  
`lensdk.internal.brain_observatory.time_sync`),  
`get_signal_correlation()` (al- 301  
`lensdk.brain_observatory.static_gratings.StaticGratings.get_signal_correlation()` (in module `al-`  
`method`), 195  
`lensdk.brain_observatory.static_gratings.StaticGratings.get_stim_photodiode()` (in module `al-`  
`method`), 195  
`lensdk.brain_observatory.static_gratings.StaticGratings.get_stim_photodiode()` (in module `al-`  
`method`), 195

<code>lensdk.brain_observatory.behavior.sync</code> ), 88	<code>lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api</code> method), 128
<code>get_stimulus()</code> (al- <code>lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet</code> method), 234	<code>get_stimulus_presentations()</code> (al- <code>lensdk.brain_observatory.nwb.nwb_api.NwbApi</code> method), 170
<code>get_stimulus_attr_changes()</code> (in module al- <code>lensdk.brain_observatory.behavior.trials_processing</code> ), 110	<code>get_stimulus_presentations()</code> (al- <code>lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi</code> method), 283
<code>get_stimulus_epoch_table()</code> (al- <code>lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet</code> method), 234	<code>get_stimulus_presentations()</code> (al- <code>lensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi</code> method), 285
<code>get_stimulus_epochs()</code> (al- <code>lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession</code> method), 166	<code>get_stimulus_presentations()</code> (in module al- <code>lensdk.brain_observatory.behavior.stimulus_processing</code> ), 108
<code>get_stimulus_file()</code> (al- <code>lensdk.internal.model.biophysical.biophysical_archiver.BiophysicalArchiver</code> method), 312	<code>get_stimulus_rebase_function()</code> (al- <code>lensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi</code> method), 285
<code>get_stimulus_frame_rate()</code> (al- <code>lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi</code> method), 283	<code>get_stimulus_rebase_function()</code> (in module <code>lensdk.brain_observatory.behavior.sync</code> ), 88
<code>get_stimulus_frame_rate()</code> (al- <code>lensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi</code> method), 285	<code>get_stimulus_table()</code> (al- <code>lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession</code> method), 166
<code>get_stimulus_mappings()</code> (al- <code>lensdk.api.queries.brain_observatory_api.BrainObservatoryApi</code> method), 48	<code>get_stimulus_table()</code> (al- <code>lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet</code> method), 234
<code>get_stimulus_metadata()</code> (in module al- <code>lensdk.brain_observatory.behavior.stimulus_processing</code> ), 108	<code>get_stimulus_templates()</code> (al- <code>lensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_api</code> method), 81
<code>get_stimulus_name()</code> (al- <code>lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi</code> method), 283	<code>get_stimulus_templates()</code> (al- <code>lensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysLimsApi</code> method), 81
<code>get_stimulus_name()</code> (al- <code>lensdk.internal.api.ophys_lims_api.OphysLimsApi</code> method), 287	<code>get_stimulus_templates()</code> (al- <code>lensdk.brain_observatory.behavior.internal.behavior_base.BehaviorOphysBase</code> method), 82
<code>get_stimulus_parameter_values()</code> (al- <code>lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession</code> method), 166	<code>get_stimulus_templates()</code> (al- <code>lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi</code> method), 285
<code>get_stimulus_presentations()</code> (al- <code>lensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysLimsApi</code> method), 81	<code>get_stimulus_templates()</code> (al- <code>lensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysLimsApi</code> method), 285
<code>get_stimulus_presentations()</code> (al- <code>lensdk.brain_observatory.behavior.internal.behavior_base.BehaviorOphysBase</code> method), 82	<code>get_stimulus_templates()</code> (in module al- <code>lensdk.brain_observatory.behavior.stimulus_processing</code> ), 108
<code>get_stimulus_presentations()</code> (al- <code>lensdk.brain_observatory.behavior.internal.behavior_ophys_lims_api.BehaviorOphysLimsApi</code> method), 84	<code>get_stimulus_timestamps()</code> (al- <code>lensdk.brain_observatory.behavior.behavior_ophys_api.behavior_ophys_api</code> method), 126
<code>get_stimulus_presentations()</code> (al- <code>lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api</code> method), 126	<code>get_stimulus_timestamps()</code> (al- <code>lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api</code> method), 127
<code>get_stimulus_presentations()</code> (al- <code>lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api</code> method), 127	<code>get_stimulus_timestamps()</code> (al- <code>lensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysLimsApi</code> method), 127
<code>get_stimulus_presentations()</code> (al- <code>lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api</code> method), 127	<code>get_stimulus_timestamps()</code> (al- <code>lensdk.brain_observatory.behavior.internal.behavior_base.BehaviorOphysBase</code> method), 127



get_sync_file()	(al- lensdk.internal.api.ophys_lims_api.OphysLimsApi method), 287	lensdk.brain_observatory.behavior.dprime(), 101
get_sync_licks()	(al- lensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi method), 285	get_trial_image_names() (in module al- lensdk.brain_observatory.behavior.trials_processing), 110
get_synchronized_frame_times() (in module allensdk.brain_observatory.sync_utilities), 183	get_trial_lick_times() (in module al- lensdk.brain_observatory.behavior.trials_processing), 110	get_trial_reward_time() (in module al- lensdk.brain_observatory.behavior.trials_processing), 110
get_targeted_regions()	(al- lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.FixedApi method), 118	get_trial_timing() (in module al- lensdk.brain_observatory.behavior.trials_processing), 110
get_targeted_structure()	(al- lensdk.internal.api.ophys_lims_api.OphysLimsApi method), 287	get_trials() (allensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysNwbApi method), 81
get_task_parameters()	(al- lensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysNwbApi method), 81	get_trials() (allensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysNwbApi method), 82
get_task_parameters()	(al- lensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysNwbApi method), 82	get_trials() (allensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysNwbApi method), 83
get_task_parameters()	(al- lensdk.brain_observatory.behavior.internal.behavior_base.BehaviorBase method), 83	get_trials() (allensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi method), 284
get_task_parameters()	(al- lensdk.internal.api.behavior_data_lims_api.BehaviorDataLimsApi method), 284	get_trials() (allensdk.brain_observatory.behavior.trials_processing), 110
get_task_parameters()	(al- lensdk.internal.api.behavior_ophys_api.BehaviorOphysLimsApi_v0() method), 285	get_trials() (allensdk.brain_observatory.behavior.trials_processing), 110
get_task_parameters() (in module al- lensdk.brain_observatory.behavior.metadata_processing), 104	get_trigger() (in module al- lensdk.brain_observatory.behavior.sync), 88	get_trigger() (in module al- lensdk.brain_observatory.behavior.sync), 88
get_template_names()	(al- lensdk.internal.model.biophysical.biophysical_archiver.BiophysicalArchiver method), 312	get_unit_analysis_metrics() (al- lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.FixedApi method), 117
get_template_volume()	(al- lensdk.core.reference_space_cache.ReferenceSpaceCache method), 253	get_unit_analysis_metrics() (al- lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.FixedApi method), 120
get_time() (in module al- lensdk.brain_observatory.behavior.trials_processing), 110	get_unit_analysis_metrics() (al- lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.FixedApi method), 121	get_unit_analysis_metrics_by_session_type() (allensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache method), 160
get_time_string() (in module al- lensdk.internal.ephys.plot_qc_figures), 306	get_unit_analysis_metrics_for_session() (allensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache method), 161	get_unit_filter_value() (in module al- lensdk.brain_observatory.ecephys), 170
get_time_string() (in module al- lensdk.internal.ephys.plot_qc_figures3), 308	get_units() (allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.FixedApi method), 117	get_units() (allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.FixedApi method), 118
get_tree() (allensdk.api.queries.tree_search_api.TreeSearchApi method), 72		
get_trial_count_corrected_false_alarm_rate() (in module al- lensdk.brain_observatory.behavior.dprime), 100		
get_trial_count_corrected_hit_rate() (in module al- lensdk.brain_observatory.behavior.dprime), 100		

`get_units()` (`allensdk.brain_observatory.ecephys.ecephys_project_lims_api.EcephysProjectLimsApi` method), 120  
`get_units()` (`allensdk.brain_observatory.ecephys.ecephys_project_lims_api.EcephysProjectLimsApi` method), 121  
`get_units()` (`allensdk.brain_observatory.ecephys.ecephys_project_cache.EcephysProjectCache` (al-  
`lensdk.brain_observatory.stimulus_info.BrainObservatoryMonitor` method), 161  
`get_units()` (`allensdk.brain_observatory.ecephys.ecephys_session_api.EcephysSessionApi` method), 126  
`get_units()` (`allensdk.brain_observatory.ecephys.ecephys_session_api.EcephysSessionApi` method), 127  
`get_units()` (`allensdk.brain_observatory.ecephys.ecephys_session_api.EcephysSessionApi` method), 128  
`get_video_length()` (in module `allensdk.internal.brain_observatory.time_sync`), 301  
`get_visual_stimuli_df()` (in module `allensdk.brain_observatory.behavior.stimulus_processing.single_data_api` (class in al-  
`lensdk.api.queries.grid_data_api`), 52  
`get_volume_scale()` (in module `allensdk.internal.mouse_connectivity.interval_unionize.run_tibssusdyintunialize_classes.grid_data_api_prerelease`), 337  
`get_well_known_file_by_name()` (in module `allensdk.internal.core.lims_utilities`), 302  
`get_well_known_file_by_type()` (in module `allensdk.internal.core.lims_utilities`), 302  
`get_well_known_file_ids()` (al-  
`lensdk.api.queries.biophysical_api.BiophysicalApi` method), 45  
`get_well_known_files_by_name()` (in module `allensdk.internal.core.lims_utilities`), 302  
`get_well_known_files_by_type()` (in module `allensdk.internal.core.lims_utilities`), 302  
`get_wkf()` (in module `allensdk.internal.pipeline_modules.run_ophys_eye_calibration`), 348  
`get_workflow_state()` (al-  
`lensdk.internal.api.ophys_lims_api.OphysLimsApi` method), 287  
`GLIF_TYPES` (`allensdk.api.queries.glif_api.GlifApi` attribute), 52  
`GlifApi` (class in `allensdk.api.queries.glif_api`), 52  
`GlifBadInitializationException`, 319  
`GlifBadResetException`, 356  
`GlifExperiment` (class in al-  
`lensdk.internal.model.glif.glif_experiment`), 317  
`GlifNeuron` (class in `allensdk.model.glif.glif_neuron`), 356  
`GlifNeuronException`, 319  
`GlifNeuronMethod` (class in al-  
`lensdk.model.glif.glif_neuron_methods`), 359  
`GlifOptimizer` (class in al-  
`lensdk.internal.model.glif.glif_optimizer`), 218  
`GlifOptimizerNeuron` (class in al-  
`lensdk.internal.model.glif.glif_optimizer_neuron`), 219  
`GlifProjectWarehouse` (class in al-  
`lensdk.brain_observatory.ecephys.ecephys_project_warehouse`), 219  
`GlifSessionApiPrerelease` (class in al-  
`lensdk.internal.api.queries.grid_data_api_prerelease`), 278  
`GridDataApiPrerelease` (class in al-  
`lensdk.internal.mouse_connectivity.interval_unionize.run_tibssusdyintunialize_classes.grid_data_api_prerelease`), 278  
`grid_image_blocks()` (in module `allensdk.mouse_connectivity.grid.utilities.image_utilities`), 370  
`grating_to_screen()` (al-  
`lensdk.brain_observatory.stimulus_info.BrainObservatoryMonitor` method), 209  
`h` (`allensdk.model.biophys_sim.neuron.hoc_utils.HocUtils` attribute), 352  
`h5_object_matcher_relname_in()` (in module `allensdk.core.h5_utilities`), 239  
`handle_output_image()` (in module `allensdk.internal.mouse_connectivity.projection_thumbnail.generator`), 339  
`handle_pyramid()` (in module `allensdk.mouse_connectivity.grid.writers`), 372  
`has_calibration_dt()` (in module `allensdk.ephys.ephys_features`), 275  
`has_lfp_data` (`allensdk.brain_observatory.ecephys.nwb.EcephysProbe` attribute), 132  
`has_overlaps()` (al-  
`lensdk.core.structure_tree.StructureTree` method), 259  
`Hdf5Util` (class in al-  
`lensdk.config.model.formats.hdf5_util`), 219  
`height` (`allensdk.brain_observatory.stimulus_info.Monitor` attribute), 209  
`hex_pack()` (in module `allensdk.brain_observatory.circle_plots`), 188  
`hex_to_rgb()` (`allensdk.core.structure_tree.StructureTree` static method), 259  
`HocUtils` (class in al-  
`lensdk.model.biophys_sim.neuron.hoc_utils`), 352  
`hook()` (in module `allensdk.brain_observatory`), 216  
`host` (`allensdk.core.authentication.DbCredentials` attribute), 226



[HttpEngine](#) (class in [allensdk.brain\\_observatory.ecephys.ecephys\\_project\\_api.HttpEngine](#)), 187  
[HUMAN](#) ([allensdk.api.queries.cell\\_types\\_api.CellTypesApi](#) attribute), 49  
[HUMAN\\_ORGANISM](#) ([allensdk.api.queries.mouse\\_atlas\\_api.MouseAtlasApi](#) attribute), 57  
[IDCreator](#) (class in [allensdk.brain\\_observatory.ecephys.ecephys\\_session\\_api.IDCreator](#)), 126  
[identify\\_valid\\_masks\(\)](#) (in module [allensdk.internal.brain\\_observatory.demixer](#)), 290  
[Image](#) (class in [allensdk.brain\\_observatory.behavior.image\\_api](#)), 104  
[image\\_from\\_array\(\)](#) (in module [allensdk.mouse\\_connectivity.grid.utilities.image\\_utilities](#)), 370  
[image\\_selectivity\(\)](#) (in module [allensdk.brain\\_observatory.ecephys.stimulus\\_analysis.natural\\_scenes](#)), 138  
[ImageApi](#) (class in [allensdk.brain\\_observatory.behavior.image\\_api](#)), 104  
[ImageDownloadApi](#) (class in [allensdk.api.queries.image\\_download\\_api](#)), 54  
[ImageOutputStream](#) (class in [allensdk.internal.brain\\_observatory.frame\\_stream](#)), 295  
[images](#) ([allensdk.brain\\_observatory.ecephys.stimulus\\_analysis.natural\\_scenes.NaturalScenes](#) attribute), 137  
[images\\_nonblank](#) ([allensdk.brain\\_observatory.ecephys.stimulus\\_analysis.natural\\_scenes.NaturalScenes](#) attribute), 138  
[ImageSeriesGridder](#) (class in [allensdk.mouse\\_connectivity.grid.image\\_series\\_gridder](#)), 372  
[ImageSheet](#) (class in [allensdk.internal.mouse\\_connectivity.projection\\_thumbnail\\_image\\_sheet](#)), 340  
[INCLUDE](#) ([allensdk.api.queries.rma\\_api.RmaApi](#) attribute), 65  
[INDETERMINATE](#) ([allensdk.core.cell\\_types\\_cache.ReporterStatus](#) attribute), 238  
[infer\\_column\\_types\(\)](#) (in module [allensdk.brain\\_observatory.ecephys.ecephys\\_project\\_api.rma\\_engine](#)), 124  
[infer\\_dims\(\)](#) ([allensdk.brain\\_observatory.circle\\_plots.CirclePlots](#) method), 186  
[infer\\_dims\(\)](#) ([allensdk.brain\\_observatory.circle\\_plots.FanPlots](#) method), 187  
[info\(\)](#) ([allensdk.internal.model.biophysical.run\\_optimize.RunOptimize](#) method), 314  
[init\\_by\\_mask\(\)](#) ([allensdk.brain\\_observatory.roi\\_masks.NeuropilMask](#) method), 199  
[init\\_by\\_mask\(\)](#) ([allensdk.brain\\_observatory.roi\\_masks.RoiMask](#) method), 199  
[init\\_by\\_pixels\(\)](#) ([allensdk.brain\\_observatory.roi\\_masks.Mask](#) method), 199  
[initial\\_cr\\_point\(\)](#) (in module [allensdk.internal.brain\\_observatory.itracker\\_utils](#)), 295  
[initial\\_pupil\\_point\(\)](#) (in module [allensdk.internal.brain\\_observatory.itracker\\_utils](#)), 295  
[initialize\\_coarse\\_volume\(\)](#) ([allensdk.mouse\\_connectivity.grid.image\\_series\\_gridder.ImageSeriesGridder](#) method), 372  
[initialize\\_hoc\(\)](#) ([allensdk.model.biophys\\_sim.neuron.hoc\\_utils.HocUtils](#) method), 352  
[initialize\\_image\(\)](#) ([allensdk.internal.mouse\\_connectivity.tissuecyte\\_stitching.tile.Tile](#) method), 343  
[initialize\\_image\(\)](#) (in module [allensdk.internal.mouse\\_connectivity.tissuecyte\\_stitching.stitcher](#)), 343  
[initialize\\_images\(\)](#) (in module [allensdk.internal.mouse\\_connectivity.tissuecyte\\_stitching.stitcher](#)), 343  
[initiate\\_unique\\_seed\(\)](#) ([allensdk.internal.model.glif.glif\\_optimizer.GlifOptimizer](#) method), 318  
[INJECTION\\_DENSITY](#) ([allensdk.api.queries.grid\\_data\\_api.GridDataApi](#) attribute), 53  
[INJECTION\\_DENSITY\\_KEY](#) ([allensdk.core.mouse\\_connectivity\\_cache.MouseConnectivityCache](#) attribute), 241  
[INJECTION\\_ENERGY](#) ([allensdk.api.queries.grid\\_data\\_api.GridDataApi](#) attribute), 53  
[INJECTION\\_FRACTION](#) ([allensdk.api.queries.grid\\_data\\_api.GridDataApi](#) attribute), 53  
[INJECTION\\_FRACTION\\_KEY](#) ([allensdk.core.mouse\\_connectivity\\_cache.MouseConnectivityCache](#) attribute), 241  
[init\\_data\(\)](#) ([allensdk.internal.core.lims\\_pipeline\\_module.PipelineModule](#) method), 301

allensdk.ephys.ephys\_extractor), 268  
 InputFile (class in *allensdk.brain\_observatory.argschema\_utilities*), 184  
 insert\_iclamp() (*allensdk.internal.model.biophysical.deap\_utils.Utils* method), 313  
 INTEGER (*allensdk.api.queries.connected\_services.ConnectedService* attribute), 51  
 INTENSITY (*allensdk.api.queries.grid\_data\_api.GridDataApi* attribute), 53  
 IntensitySubImage (class in *allensdk.mouse\_connectivity.grid.subimage.base\_subimage*), 366  
 interlength (*allensdk.brain\_observatory.locally\_sparse\_noise.LocallySparseNoise* attribute), 193  
 interlength (*allensdk.brain\_observatory.natural\_scenes.NaturalScenes* attribute), 195  
 interlength (*allensdk.brain\_observatory.static\_gratings.StaticGratings* attribute), 206  
 interpolate\_RF() (in module *allensdk.brain\_observatory.receptive\_field\_analysis.chisquare*), 177  
 interpolate\_spike\_time() (in module *allensdk.model.glif.glif\_neuron*), 359  
 interpolate\_spike\_value() (in module *allensdk.model.glif.glif\_neuron*), 359  
 interpolate\_spike\_voltage() (in module *allensdk.internal.model.glif.glif\_optimizer\_neuron*), 321  
 intersection() (*allensdk.internal.brain\_observatory.mask\_set.MaskSet* method), 296  
 intersection\_size() (*allensdk.internal.brain\_observatory.mask\_set.MaskSet* method), 296  
 IntervalUnionizer (class in *allensdk.internal.mouse\_connectivity.interval\_unionize.interval\_unionizer*), 336  
 invert\_rf() (in module *allensdk.brain\_observatory.ecephys.stimulus\_analysis.receptive\_field\_mapping*), 141  
 invnl() (in module *allensdk.internal.model.GLM*), 326  
 IS (*allensdk.api.queries.rma\_api.RmaApi* attribute), 65  
 is\_distinct\_from() (in module *allensdk.brain\_observatory.ecephys.ecephys\_session*), 168  
 is\_empty() (*allensdk.config.model.description.Description* method), 221  
 is\_equal() (in module *allensdk.brain\_observatory.session\_api\_utils*), 205  
 is\_rf\_inverted() (in module *allensdk.brain\_observatory.ecephys.stimulus\_analysis.receptive\_field\_analysis*), 141  
 is\_spike\_feature\_affected\_by\_clipping() (*allensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor* method), 266  
 is\_well\_known\_file\_type() (*allensdk.api.queries.biophysical\_api.BiophysicalApi* method), 45  
 IService (*allensdk.ephys.feature\_extractor.EphysFeatureExtractor* method), 276  
 J  
 json\_handler() (in module *allensdk.core.json\_utilities*), 240  
 json\_msg\_query() (*allensdk.api.api.Api* method), 240  
 json\_remove\_keys() (*allensdk.api.cache.Cache* method), 77  
 json\_rename\_columns() (*allensdk.api.cache.Cache* static method), 77  
 JsonComments (class in *allensdk.core.json\_utilities*), 239  
 json\_description\_parser (class in *allensdk.config.model.formats.json\_description\_parser*), 219  
 JSONEncoder (class in *allensdk.brain\_observatory*), 216  
 K  
 keyed\_locate\_h5\_objects() (in module *allensdk.core.h5\_utilities*), 239  
 known\_spontaneous\_keys (*allensdk.brain\_observatory.ecephys.stimulus\_analysis.dot\_motion.L* attribute), 133  
 known\_spontaneous\_keys (*allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_and* attribute), 144  
 known\_stimulus\_keys() (*allensdk.brain\_observatory.ecephys.stimulus\_analysis.dot\_motion.L* class method), 133  
 known\_stimulus\_keys() (*allensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_grating* class method), 134  
 known\_stimulus\_keys() (*allensdk.brain\_observatory.ecephys.stimulus\_analysis.flashes.Flash* class method), 136  
 known\_stimulus\_keys() (*allensdk.brain\_observatory.ecephys.stimulus\_analysis.natural\_mov* class method), 137  
 known\_stimulus\_keys() (*allensdk.brain\_observatory.ecephys.stimulus\_analysis.natural\_scen* class method), 138  
 known\_stimulus\_keys() (*allensdk.brain\_observatory.ecephys.stimulus\_analysis.receptive\_fie*

class method), 139  
 known\_stimulus\_keys() (al-  
   lensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_gratings.StaticGratings  
   class method), 142  
 known\_stimulus\_keys() (al-  
   lensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_gratings.StaticGratings  
   class method), 144  
**L**  
 label\_data() (allensdk.internal.brain\_observatory.roi\_filter\_utils.TrainingMultiLabelClassifier  
   method), 299  
 latency() (in module allensdk.ephys.ephys\_features),  
   275  
 LazyProperty (allensdk.core.lazy\_property.lazy\_property\_mixin.LazyProperty  
   attribute), 225  
 LazyProperty (class in al-  
   lensdk.core.lazy\_property.lazy\_property),  
   225  
 LazyPropertyMixin (class in al-  
   lensdk.core.lazy\_property.lazy\_property\_mixin),  
   225  
 least\_squares\_RCEl\_calc\_tested() (in mod-  
   ule allensdk.internal.model.glif.rc), 322  
 legacy() (in module allensdk.deprecated), 374  
 legend\_artist() (al-  
   lensdk.brain\_observatory.observatory\_plots.DimensionPatcher  
   method), 196  
 lfp\_sampling\_rate (al-  
   lensdk.brain\_observatory.ecephys.nwb.EcephysProbe  
   attribute), 132  
 licks (allensdk.brain\_observatory.behavior.behavior\_data\_session.BehaviorDataSession  
   attribute), 89  
 licks (allensdk.brain\_observatory.behavior.behavior\_ophys\_session.BehaviorOphysSession  
   attribute), 94  
 lifetime\_sparseness() (in module al-  
   lensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis),  
   146  
 lims\_working\_directory() (al-  
   lensdk.internal.api.queries.biophysical\_module\_reader.BiophysicalModuleReader  
   method), 277  
 lims\_working\_directory() (al-  
   lensdk.internal.api.queries.optimize\_config\_reader.OptimizeConfigReader  
   method), 280  
 LimsApi (class in allensdk.internal.api.lims\_api), 285  
 line\_crossing\_x() (in module al-  
   lensdk.model.glif.glif\_neuron), 359  
 line\_crossing\_y() (in module al-  
   lensdk.model.glif.glif\_neuron), 359  
 line\_stats() (allensdk.brain\_observatory.sync\_dataset.Dataset  
   method), 215  
 linear\_transform\_from\_intervals() (in  
   module al-  
   lensdk.brain\_observatory.ecephys.align\_timestamps.barcode),  
   113  
 linux\_to\_windows() (in module al-  
   lensdk.internal.core.lims\_utilities), 302  
 list\_cells\_api() (al-  
   lensdk.api.queries.cell\_types\_api.CellTypesApi  
   method), 50  
 list\_cells\_api() (al-  
   lensdk.api.queries.cell\_types\_api.CellTypesApi  
   method), 50  
 list\_column\_definition\_class\_names() (allensdk.api.queries.brain\_observatory\_api.BrainObservatoryApi  
   method), 48  
 list\_injectable\_properties() (al-  
   lensdk.api.queries.brain\_observatory\_api.BrainObservatoryApi  
   method), 48  
 list\_neuronal\_models() (al-  
   lensdk.api.queries.glif\_api.GlifApi  
   method),  
   52  
 list\_of\_dicts\_to\_dict\_of\_lists() (in  
   module al-  
   lensdk.brain\_observatory.receptive\_field\_analysis.tools),  
   180  
 list\_stimuli() (al-  
   lensdk.core.brain\_observatory\_nwb\_data\_set.BrainObservatoryNwbDataSet  
   method), 235  
 load() (allensdk.brain\_observatory.ecephys.file\_io.continuous\_file.ContinuousFile  
   method), 128  
 load() (allensdk.brain\_observatory.sync\_dataset.Dataset  
   method), 215  
 load() (allensdk.config.application\_config.ApplicationConfig  
   method), 218  
 load() (allensdk.model.biophys\_sim.config.Config  
   method), 353  
 load\_annotation() (in module al-  
   lensdk.internal.mouse\_connectivity.interval\_unionize.data\_utilities),  
   336  
 load\_api\_schema() (allensdk.api.api.Api  
   method), 74  
 load\_arrays() (in module al-  
   lensdk.internal.pipeline\_modules.run\_annotated\_region\_metrics),  
   345  
 load\_cell\_parameters() (al-  
   lensdk.internal.model.biophysical.deap\_utils.Utils  
   method), 313  
 load\_cell\_parameters() (al-  
   lensdk.model.biophysical.utils.AllActiveUtils  
   method), 354  
 load\_cell\_parameters() (al-  
   lensdk.model.biophysical.utils.Utils  
   method),  
   355  
 load\_config() (allensdk.config.manifest.Manifest  
   method), 224  
 load\_csv() (allensdk.api.cache.Cache  
   method), 77



`load_datasets_by_relnames()` (in module `allensdk.core.h5_utilities`), 239  
`load_description()` (in module `allensdk.model.biophysical.runner`), 353  
`load_experiment()` (in module `allensdk.internal.ephys.plot_qc_figures`), 306  
`load_experiment()` (in module `allensdk.internal.ephys.plot_qc_figures3`), 308  
`load_eye_tracking_hdf()` (in module `allensdk.brain_observatory.behavior.eye_tracking_processing`), 102  
`load_frame()` (in module `allensdk.internal.brain_observatory.ophys_session_decomposition`), 297  
`load_json()` (`allensdk.api.cache.Cache` method), 77  
`load_LabMetaData_extension()` (in module `allensdk.brain_observatory.nwb.metadata`), 170  
`load_manifest()` (`allensdk.api.cache.Cache` method), 77  
`load_manifest()` (`allensdk.internal.model.biophysical.run_optimize.RunOptimize` method), 314  
`load_manifest()` (`allensdk.model.biophysical.run_simulate.RunSimulate` method), 353  
`load_morphology()` (in module `allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit`), 311  
`load_pickle()` (in module `allensdk.brain_observatory.behavior.stimulus_processing`), 108  
`load_structure_assignments()` (in module `allensdk.brain_observatory.ecephys.ecephys_project_api.warehouse_utilities`), 116  
`load_sweep()` (in module `allensdk.internal.model.data_access`), 327  
`load_sweep()` (in module `allensdk.model.glif.simulate_neuron`), 365  
`load_sweeps()` (in module `allensdk.internal.model.data_access`), 327  
`local_time()` (in module `allensdk.brain_observatory.behavior.trials_processing`), 111  
`LocallySparseNoise` (class in `allensdk.brain_observatory.locally_sparse_noise`), 192  
`locate_h5_objects()` (in module `allensdk.core.h5_utilities`), 239  
`locate_median()` (in module `allensdk.brain_observatory.receptive_field_analysis.chisquare_tests`), 177  
`log(allensdk.config.manifest.Manifest` attribute), 224  
`log(allensdk.config.model.description_parser.DescriptionParser` attribute), 222  
`log(allensdk.config.model.formats.json_description_parser.JsonDescriptionParser` attribute), 219  
`log(allensdk.config.model.formats.pycfg_description_parser.PycfgDescriptionParser` attribute), 220  
`long_squares_features()` (`allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor` method), 264  
`long_squares_stim_amps()` (`allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor` method), 264  
`LSN(allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` attribute), 193  
`LSN_GREY(allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` attribute), 193  
`LSN_MASK(allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` attribute), 193  
`LSN_OFF_SCREEN(allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` attribute), 193  
`LSN_ON(allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` attribute), 193  
`LSN_ON_FULL(allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` attribute), 193  
`lsna_check_hvas()` (in module `allensdk.internal.pipeline_modules.run_observatory_thumbnails`), 348

## M

`macros()` (in module `allensdk.brain_observatory.ecephys.ecephys_project_api.utilities`), 125  
`main()` (in module `allensdk.brain_observatory.dff`), 190  
`main()` (in module `allensdk.brain_observatory.ecephys.stimulus_table.visualization.view`), 148  
`main()` (in module `allensdk.brain_observatory.session_analysis`), 204  
`main()` (in module `allensdk.internal.ephys.plot_qc_figures`), 306  
`main()` (in module `allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit`), 311

```

main()          (in          module          al-          lensdk.internal.pipeline_modules.run_ophys_session_decomposit
lensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit2),
311
main()          (in          module          al-          lensdk.internal.pipeline_modules.run_ophys_time_sync),
lensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit_elec),
311
main()          (in          module          al-          lensdk.internal.pipeline_modules.run_tissuecyte_unionize_classic
lensdk.internal.model.biophysical.passive_fitting.preprocess),
312
main()          (in          module          al-          lensdk.model.biophysical.run_simulate),
lensdk.internal.model.biophysical.run_optimize),
314
main()          (in          module          al-          lensdk.model.glif.simulate_neuron),
lensdk.internal.model.biophysical.run_passive_fit),
314
main()          (in          module          al-          make_bbs()          (in          module          al-
lensdk.internal.model.biophysical.run_simulate_lims),          lensdk.internal.brain_observatory.mask_set),
314
main()          (in          module          al-          make_blended_tile()          (in          module          al-
lensdk.internal.model.glif.find_sweeps),
316
main()          (in          module          al-          lensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher),
lensdk.internal.morphology.validate_swc),
335
main()          (in          module          al-          make_category_dummy()          (in          module          al-
lensdk.internal.pipeline_modules.gbm.generate_gbm_analysis_run_macrods),          lensdk.brain_observatory.chisquare_categorical),
344
main()          (in          module          al-          make_cell_html()          (in          module          al-
lensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap),          lensdk.internal.ephys.plot_qc_figures),
344
main()          (in          module          al-          make_cell_page()          (in          module          al-
lensdk.internal.pipeline_modules.gbm.generate_gbm_sample_cell_html),          lensdk.internal.ephys.plot_qc_figures3),
344
main()          (in          module          al-          make_cell_page()          (in          module          al-
lensdk.internal.pipeline_modules.run_annotated_region_masks),          lensdk.internal.ephys.plot_qc_figures3),
345
main()          (in          module          al-          make_display_mask()          (in          module          al-
lensdk.internal.pipeline_modules.run_demixing),          lensdk.brain_observatory.stimulus_info),
345
main()          (in          module          al-          make_fan_plot()          (al-
lensdk.internal.pipeline_modules.run_dff_computation),          lensdk.brain_observatory.ecephys.stimulus_analysis.static_gratin
345
main()          (in          module          al-          make_fit()          (allensdk.internal.model.biophysical.run_optimize.RunOptim
method),
314
lensdk.internal.pipeline_modules.run_neuropil_correction),          fit_json_file()          (al-
345
main()          (in          module          al-          lensdk.internal.model.biophysical.make_deap_fit_json.Report
method),
313
lensdk.internal.pipeline_modules.run_observatory_analysis),          cushion_plot()          (in          module          al-
346
main()          (in          module          al-          lensdk.brain_observatory.circle_plots),
188
lensdk.internal.pipeline_modules.run_observatory_thumbnail),          make_pixel_counter()          (al-
348
main()          (in          module          al-          lensdk.mouse_connectivity.grid.subimage.base_subimage.SubIma
method),
368
lensdk.internal.pipeline_modules.run_ophys_eye_calibration),          make_ratio_volume()          (al-
348
main()          (in          module          al-          lensdk.mouse_connectivity.grid.image_series_gridder.ImageSerie
method),
372
main()          (in          module          al-          make_spontaneous_activity_tables()

```

(in module al- method), 250  
 lensdk.brain\_observatory.ecephys.stimulus\_table.naming\_utilities (in module al-  
 151 lensdk.brain\_observatory.ecephys.stimulus\_table.naming\_utilities)  
 make\_star\_plot() (al- 153  
 lensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_gratings.DriftingGratings\_stimulus\_coordinate()  
 method), 134 (in module al-  
 make\_structure\_mask() (al- lensdk.brain\_observatory.stimulus\_info),  
 lensdk.core.reference\_space.ReferenceSpace 210  
 method), 250 map\_monitor\_coordinate\_to\_template\_coordinate()  
 make\_sweep\_html() (in module al- (in module al-  
 lensdk.internal.ephys.plot\_qc\_figures), 306 lensdk.brain\_observatory.stimulus\_info),  
 make\_sweep\_html() (in module al- 210  
 lensdk.internal.ephys.plot\_qc\_figures3), map\_stimulus() (al-  
 308 lensdk.brain\_observatory.stimulus\_info.Monitor  
 make\_sweep\_page() (in module al- method), 209  
 lensdk.internal.ephys.plot\_qc\_figures), 306 map\_stimulus() (in module al-  
 make\_sweep\_page() (in module al- lensdk.brain\_observatory.stimulus\_info),  
 lensdk.internal.ephys.plot\_qc\_figures3), 210  
 308 map\_stimulus\_coordinate\_to\_monitor\_coordinate()  
 makeBasis\_StimKernel() (in module al- (in module al-  
 lensdk.internal.model.GLM), 326 lensdk.brain\_observatory.stimulus\_info),  
 makeBasis\_StimKernel\_exp() (in module al- 210  
 lensdk.internal.model.GLM), 326 map\_stimulus\_names() (in module al-  
 makeFitStruct\_GLM() (in module al- lensdk.brain\_observatory.ecephys.stimulus\_table.naming\_utilities  
 lensdk.internal.model.GLM), 326 153  
 Manifest (class in allensdk.config.manifest), 222 map\_template\_coordinate\_to\_monitor\_coordinate()  
 MANIFEST\_CONFIG (al- (in module al-  
 lensdk.brain\_observatory.behavior.behavior\_project\_cache.BehaviorProjectCache.stimulus\_info),  
 attribute), 95 210  
 manifest\_dataframe() (allensdk.api.cache.Cache Marker (class in allensdk.core.swc), 260  
 method), 77 Marker (class in allensdk.internal.core.swc), 304  
 MANIFEST\_VERSION (al- MARKER\_FILE\_TYPE (al-  
 lensdk.brain\_observatory.behavior.behavior\_project\_cache.BehaviorProjectCache.stimulus\_info),  
 attribute), 95 49  
 MANIFEST\_VERSION (al- MARKER\_KEY (allensdk.core.cell\_types\_cache.CellTypesCache  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache  
 attribute), 158 236  
 MANIFEST\_VERSION (al- mask (allensdk.brain\_observatory.stimulus\_info.Monitor  
 lensdk.core.brain\_observatory\_cache.BrainObservatoryCache (class in allensdk.brain\_observatory.roi\_masks),  
 attribute), 227 199  
 MANIFEST\_VERSION (al- mask() (allensdk.internal.brain\_observatory.mask\_set.MaskSet  
 lensdk.core.cell\_types\_cache.CellTypesCache method), 296  
 attribute), 236 mask\_is\_union\_of\_set() (al-  
 MANIFEST\_VERSION (al- lensdk.internal.brain\_observatory.mask\_set.MaskSet  
 lensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache method), 296  
 attribute), 241 mask\_nulls() (in module al-  
 MANIFEST\_VERSION (al- lensdk.internal.ephys.plot\_qc\_figures), 306  
 lensdk.core.reference\_space\_cache.ReferenceSpaceCache mask\_nulls() (in module al-  
 attribute), 251 lensdk.internal.ephys.plot\_qc\_figures3),  
 ManifestBuilder (class in al- 309  
 lensdk.config.manifest\_builder), 224 mask\_stimulus\_template() (in module al-  
 ManifestVersionError, 224 lensdk.brain\_observatory.stimulus\_info), 211  
 many\_structure\_masks() (al- MaskSet (class in al-  
 lensdk.core.reference\_space.ReferenceSpace lensdk.internal.brain\_observatory.mask\_set),

296  
 match\_barcodes() (in module al- metrics (allensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_ attribute), 133  
 lensdk.brain\_observatory.ecephys.align\_timestamps.barcode attribute), 134  
 114 metrics (allensdk.brain\_observatory.ecephys.stimulus\_analysis.flashes.F  
 max\_cb() (in module al- attribute), 136  
 lensdk.internal.mouse\_connectivity.projection\_thumbnail\_generation\_pipeline attribute), 137  
 340 metrics (allensdk.brain\_observatory.ecephys.stimulus\_analysis.natural\_  
 max\_of\_line\_and\_const() (in module al- metrics (allensdk.brain\_observatory.ecephys.stimulus\_analysis.natural\_ attribute), 138  
 lensdk.model.glif.glif\_neuron\_methods), 361  
 max\_projection (al- metrics (allensdk.brain\_observatory.ecephys.stimulus\_analysis.receptive attribute), 139  
 lensdk.brain\_observatory.behavior.behavior\_ophys\_session.BehaviorOphysSession  
 attribute), 94 metrics (allensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_grati  
 max\_projection() (in module al- attribute), 142  
 lensdk.internal.mouse\_connectivity.projection\_thumbnail\_generation\_pipeline attribute), 144  
 340 metrics (allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_  
 max\_voxel\_density (al- METRICS\_COLUMNS (al-  
 lensdk.internal.mouse\_connectivity.interval\_unionize.tissuesdk.brain\_observatory.ecephys.stimulus\_analysis.dot\_motion.I  
 attribute), 337 attribute), 133  
 max\_voxel\_index (al- METRICS\_COLUMNS (al-  
 lensdk.internal.mouse\_connectivity.interval\_unionize.tissuesdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_grati  
 attribute), 338 attribute), 134  
 mean\_features\_spike\_zero() (in module al- METRICS\_COLUMNS (al-  
 lensdk.ephys.extract\_cell\_features), 276 lensdk.brain\_observatory.ecephys.stimulus\_analysis.flashes.Flash  
 mean\_response (al- attribute), 136  
 lensdk.brain\_observatory.locally\_sparse\_noise.LocallySparseNoise attribute), 193  
 mean\_sweep\_response (al- attribute), 137  
 lensdk.brain\_observatory.stimulus\_analysis.StimulusAnalysis attribute), 208  
 295 lensdk.brain\_observatory.ecephys.stimulus\_analysis.natural\_scen  
 medfilt\_custom() (in module al- attribute), 137  
 lensdk.internal.brain\_observatory.itracker\_utils), METRICS\_COLUMNS (al-  
 295 lensdk.brain\_observatory.ecephys.stimulus\_analysis.receptive\_fie  
 median\_absolute\_deviation() (in module al- attribute), 139  
 lensdk.internal.brain\_observatory.itracker\_utils), METRICS\_COLUMNS (al-  
 296 lensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_gratin  
 meets\_engagement\_criteria() (in module al- attribute), 142  
 lensdk.brain\_observatory.behavior.criteria), 99 METRICS\_COLUMNS (al-  
 membrane\_time\_constant() (in module al- lensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_and  
 lensdk.ephys.ephys\_extractor), 268 attribute), 143  
 memoize() (in module allensdk.api.cache), 79 metrics\_dtypes (al-  
 merge\_mean\_response() (al- lensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_and  
 lensdk.brain\_observatory.locally\_sparse\_noise.LocallySparseNoise attribute), 144  
 static method), 193 metrics\_names (al-  
 metadata (allensdk.brain\_observatory.behavior.behavior\_data\_sessions.BehaviorDataSession ecephys.stimulus\_analysis.stimulus\_and  
 attribute), 90 attribute), 144  
 metadata (allensdk.brain\_observatory.behavior.behavior\_ophys\_session.BehaviorOphysSession module al-  
 attribute), 94 lensdk.internal.morphology.node), 334  
 metadata (allensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession ecephys\_session\_zero() (in module al-  
 attribute), 166 lensdk.model.glif.glif\_neuron\_methods),  
 METHOD (allensdk.core.authentication.CredentialProvider 362  
 attribute), 226 minmax\_norm() (in module al-  
 METHOD (allensdk.core.authentication.EnvCredentialProvider lensdk.internal.mouse\_connectivity.projection\_thumbnail.visualiz  
 attribute), 226 341  
 metrics (allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis\_id.DriftingGratingMotion, 238

MissingStimulusException, 185  
 MissingSweepException, 316  
 MLIN() (in module `allensdk.internal.model.glif.MLIN`), 315  
 MLIN\_list\_error() (in module `allensdk.internal.model.glif.error_functions`), 315  
 mod\_file\_entries() (in module `allensdk.internal.api.queries.biophysical_module_reader`), 277  
 mod\_file\_entries() (in module `allensdk.internal.api.queries.optimize_config_reader`), 280  
 mod\_file\_paths() (in module `allensdk.internal.api.queries.biophysical_module_reader`), 277  
 mod\_file\_paths() (in module `allensdk.internal.api.queries.optimize_config_reader`), 280  
 MOD\_FILE\_TYPE\_ID (in module `allensdk.internal.api.queries.biophysical_module_reader`), 277  
 MOD\_FILE\_TYPE\_ID (in module `allensdk.internal.api.queries.optimize_config_reader`), 280  
 MODEL (in module `allensdk.api.queries.rma_api.RmaApi` attribute), 65  
 model\_query() (in module `allensdk.api.queries.rma_api.RmaApi` method), 67  
 model\_stage() (in module `allensdk.api.queries.rma_api.RmaApi` method), 67  
 model\_type() (in module `allensdk.internal.api.queries.biophysical_module_reader`), 277  
 modify\_parameter() (in module `allensdk.model.glif.glif_neuron_methods.GlifNeuronMethod`), 359  
 modulation\_index() (in module `allensdk.brain_observatory.ecephys.stimulus_analysis.drifting_neurons`), 135  
 moments2() (in module `allensdk.brain_observatory.receptive_field_analysis.moments2D`), 179  
 Monitor (class in module `allensdk.brain_observatory.stimulus_info`), 209  
 monitor\_coordinate\_to\_lsn\_coordinate() (in module `allensdk.brain_observatory.stimulus_info`), 211  
 monitor\_coordinate\_to\_natural\_movie\_coordinate() (in module `allensdk.brain_observatory.stimulus_info`), 211  
 monitor\_delay() (in module `allensdk.internal.brain_observatory.time_sync`), 301  
 Morphology (class in module `allensdk.core.swc`), 260  
 Morphology (class in module `allensdk.internal.morphology.morphology`), 328  
 MORPHOLOGY\_MODULE\_KEY (in module `allensdk.core.cell_types_cache.CellTypesCache` attribute), 236  
 MorphologyModuleReader (class in module `allensdk.internal.api.queries.biophysical_module_reader`), 278  
 MorphologyModuleReader (class in module `allensdk.internal.api.queries.optimize_config_reader`), 280  
 MORPHOLOGY\_TYPE\_ID (in module `allensdk.internal.api.queries.optimize_config_reader`), 280  
 MORPHOLOGY\_TYPE\_ID (in module `allensdk.internal.api.queries.biophysical_module_reader`), 277  
 MORPHOLOGY\_TYPE\_ID (in module `allensdk.internal.api.queries.optimize_config_reader`), 280  
 MorphologyColors (class in module `allensdk.internal.morphology.morphvis`), 332  
 mostly\_useful() (in module `allensdk.brain_observatory.behavior.criteria`), 94  
 motion\_correction (in module `allensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession` attribute), 94  
 MOTION\_CORRECTION\_DATASETS (in module `allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNWBDataSet` attribute), 231  
 MOUSE (in module `allensdk.api.queries.cell_types_api.CellTypesApi` attribute), 49  
 MOUSE\_2D (in module `allensdk.api.queries.reference_space_api.ReferenceSpaceApi` attribute), 63  
 MOUSE\_ATLAS\_PRODUCTS (in module `allensdk.api.queries.mouse_atlas_api.MouseAtlasApi` attribute), 57  
 MOUSE\_ORGANISM (in module `allensdk.api.queries.mouse_atlas_api.MouseAtlasApi` attribute), 57  
 MouseAtlasApi (class in module `allensdk.api.queries.mouse_atlas_api`), 57  
 MouseConnectivityApi (class in module `allensdk.api.queries.mouse_connectivity_api`), 57



58  
 MouseConnectivityApiPrerelease (class in al-  
 lensdk.internal.api.queries.mouse\_connectivity\_api\_prerelease),  
 279  
 MouseConnectivityCache (class in al-  
 lensdk.core.mouse\_connectivity\_cache),  
 240  
 MouseConnectivityCachePrerelease (class in al-  
 lensdk.internal.core.mouse\_connectivity\_cache\_prerelease),  
 303  
 movie\_re (allensdk.brain\_observatory.ecephys.ecephys\_project\_api\_utils.EcephysProjectWarehouseApi  
 attribute), 121  
 movingaverage() (in module al-  
 lensdk.brain\_observatory.dff), 190  
 movingmode\_fast() (in module al-  
 lensdk.brain\_observatory.dff), 190  
 MtrainApi (class in allensdk.internal.api.mtrain\_api),  
 286  
 MtrainSqlApi (class in al-  
 lensdk.internal.api.mtrain\_api), 286  
 multi\_dataframe\_merge() (in module al-  
 lensdk.brain\_observatory.session\_analysis),  
 204  
 nan\_intervals() (in module al-  
 lensdk.brain\_observatory.ecephys.ecephys\_session),  
 168  
 natural\_movie\_coordinate\_to\_monitor\_coordinate() (in module al-  
 lensdk.brain\_observatory.stimulus\_info),  
 211  
 NATURAL\_MOVIE\_DIR\_KEY (al-  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache attribute), 158  
 natural\_movie\_image\_to\_screen() (al-  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache method), 209  
 NATURAL\_MOVIE\_KEY (al-  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache attribute), 158  
 natural\_scene\_coordinate\_to\_monitor\_coordinate() (in module al-  
 lensdk.brain\_observatory.stimulus\_info),  
 211  
 NATURAL\_SCENE\_DIR\_KEY (al-  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache attribute), 158  
 natural\_scene\_image\_to\_screen() (al-  
 lensdk.brain\_observatory.stimulus\_info.Monitor method), 209  
 NATURAL\_SCENE\_KEY (al-  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache attribute), 158  
 n\_complete() (in module al-  
 lensdk.brain\_observatory.behavior.criteria),  
 99  
 NA (allensdk.core.cell\_types\_cache.ReporterStatus attribute), 238  
 name (allensdk.brain\_observatory.ecephys.stimulus\_analysis.dot\_motion.DotMotion attribute), 133  
 name (allensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_gratings.DriftingGratings attribute), 134  
 name (allensdk.brain\_observatory.ecephys.stimulus\_analysis.flashes.Flashes attribute), 136  
 name (allensdk.brain\_observatory.ecephys.stimulus\_analysis.natural\_movies.NaturalMovies attribute), 137  
 name (allensdk.brain\_observatory.ecephys.stimulus\_analysis.natural\_scenes.NaturalScenes attribute), 138  
 name (allensdk.brain\_observatory.ecephys.stimulus\_analysis.receptive\_field\_mapping.ReceptiveFieldMapping attribute), 139  
 name (allensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_gratings.StaticGratings attribute), 142  
 name (allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis\_utils.ParamsMixin attribute), 144  
 namespace (allensdk.brain\_observatory.ecephys.nwb.EcephysNWBDataInterface attribute), 132  
 namespace (allensdk.brain\_observatory.ecephys.nwb.EcephysProbeData attribute), 132  
 nan\_get() (in module al-  
 lensdk.internal.ephys.core\_feature\_extract),  
 305  
 needs\_data\_refresh() (al-  
 lensdk.brain\_observatory.behavior.schemas.OphysBehaviorMetadata attribute), 106  
 neurodata\_type (al-  
 lensdk.brain\_observatory.behavior.schemas.OphysBehaviorTaskFeature attribute), 106

attribute), 107  
 neurodata\_type (al- method), 304  
 lensdk.brain\_observatory.ecephys.nwb.EcephysLabMetaDataSet (allensdk.internal.morphology.morphology.Morphology attribute), 132  
 neurodata\_type (al- node\_list\_by\_type() (al- method), 330  
 lensdk.brain\_observatory.ecephys.nwb.EcephysProbe lensdk.internal.morphology.morphology.Morphology attribute), 132  
 neuron (allensdk.model.biophys\_sim.neuron.hoc\_utils.HocUtils.NODE\_TYPES (allensdk.core.swc.Morphology attribute), attribute), 352  
 neuron\_parameter\_count() (al- NODE\_TYPES (allensdk.internal.morphology.morphology.Morphology attribute), 328  
 lensdk.internal.model.glif.glif\_experiment.GlifExperiment method), 317  
 nodes() (allensdk.core.simple\_tree.SimpleTree method), 255  
 neuronal\_model\_optimize\_dir() (al- method), 255  
 lensdk.internal.api.queries.optimize\_config\_reader.OptimizeConfigReader (allensdk.internal.core.simpletree.SimpleTree method), 280  
 NEURONAL\_MODEL\_PARAMETERS (al- nodes\_by\_property() (al- method), 255  
 lensdk.internal.api.queries.optimize\_config\_reader.OptimizeConfigReader (allensdk.internal.core.simpletree.SimpleTree attribute), 280  
 neuronal\_model\_run\_dir() (al- NoEyeTrackingException, 185  
 lensdk.internal.api.queries.biophysical\_module\_reader.BiophysicalModuleReader module al- method), 278  
 lensdk.brain\_observatory.dff), 191  
 NeuropilMask (class in al- nonraising\_ks\_2samp() (in module al- lensdk.brain\_observatory.roi\_masks), 199  
 lensdk.brain\_observatory.stimulus\_analysis), 208  
 NeuropilSubtract (class in al- norm\_diff() (in module al- lensdk.brain\_observatory.r\_neuropil), 197  
 new\_image() (in module al- lensdk.ephys.ephys\_features), 275  
 lensdk.mouse\_connectivity.grid.utilities.image\_utilities), sq\_diff() (in module al- 371  
 lensdk.ephys.ephys\_features), 275  
 nlin() (in module allensdk.internal.model.GLM), 326  
 normalize\_actual\_parameters() (al- lensdk.internal.model.biophysical.deap\_utils.Utilis method), 313  
 NLL\_to\_pvalue() (in module al- 175  
 normalize\_F() (in module al- lensdk.brain\_observatory.r\_neuropil), 198  
 NO\_RECONSTRUCTION (allensdk.core.swc.Marker attribute), 260  
 normalize\_intensity() (in module al- lensdk.internal.mouse\_connectivity.projection\_thumbnail.visualiz 341  
 NO\_RECONSTRUCTION (al- lensdk.internal.core.swc.Marker attribute), 304  
 normalizecols() (in module al- lensdk.internal.model.GLM), 326  
 no\_response\_bias() (in module al- np\_sitk\_convert() (in module al- lensdk.brain\_observatory.behavior.criteria), 99  
 lensdk.mouse\_connectivity.grid.utilities.image\_utilities), 371  
 nocache\_dataframe() (allensdk.api.cache.Cache static method), 77  
 nrn (allensdk.model.biophys\_sim.neuron.hoc\_utils.HocUtils attribute), 352  
 nocache\_json() (allensdk.api.cache.Cache static method), 77  
 nrnivmodl() (allensdk.internal.model.biophysical.run\_optimize.RunOptimize method), 314  
 Node (class in allensdk.internal.morphology.node), 334  
 nrnivmodl() (allensdk.model.biophysical.run\_simulate.RunSimulate method), 353  
 node() (allensdk.core.simple\_tree.SimpleTree method), 255  
 null\_condition (al- lensdk.brain\_observatory.ecephys.stimulus\_analysis.dot\_motion.I attribute), 133  
 node() (allensdk.core.swc.Morphology method), 262  
 null\_condition (al- lensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_gra attribute), 134  
 node() (allensdk.internal.core.simpletree.SimpleTree method), 304  
 null\_condition (al- 330  
 node() (allensdk.internal.morphology.morphology.Morphology method), 330  
 node\_ids() (allensdk.core.simple\_tree.SimpleTree method), 255

`lensdk.brain_observatory.ecephys.stimulus_analysis.flashes.Flashes` (al-  
 attribute), 136 `lensdk.brain_observatory.ecephys.stimulus_analysis.natural_scenes.NaturalScenes` (al-  
 null\_condition (al- attribute), 138 `lensdk.brain_observatory.ecephys.stimulus_analysis.natural_movies.NaturalMovies` (al-  
 attribute), 137 `lensdk.brain_observatory.ecephys.stimulus_analysis.natural_scenes.NaturalScenes` (al-  
 null\_condition (al- attribute), 138 `lensdk.brain_observatory.ecephys.stimulus_analysis.natural_scenes.NaturalScenes` (al-  
 attribute), 138 `lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet` (class in `lensdk.core.brain_observatory_nwb_data_set`), 235  
`lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.ReceptiveFieldMapping` (al-  
 attribute), 139 `lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.DriftingGratings` (al-  
 null\_condition (al- number\_ori (allensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_gratings.DriftingGratings), 192  
 attribute), 139 `lensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings` (al-  
 attribute), 142 number\_ori (allensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_gratings.StaticGratings), 142  
 null\_condition (al- attribute), 142 `lensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings` (al-  
 attribute), 144 `lensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings` (al-  
 attribute), 206 `lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` (class in `lensdk.brain_observatory.ecephys`), 166  
 num\_channels (allensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession), 142 `lensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings` (al-  
 attribute), 166 `lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` (class in `lensdk.brain_observatory.ecephys`), 166  
 num\_contingent\_trials() (in module al- number\_phase (allensdk.brain\_observatory.static\_gratings.StaticGratings), 206  
`lensdk.brain_observatory.behavior.session_metrics`), attribute), 206  
 107 number\_scenes (al- attribute), 206  
 num\_nodes (allensdk.core.swc.Morphology attribute), `lensdk.brain_observatory.natural_scenes.NaturalScenes` (al-  
 262 attribute), 195  
 num\_nodes (allensdk.internal.morphology.morphology.Morphology), sf (allensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_gratings.StaticGratings), 142  
 attribute), 330 `lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` (class in `lensdk.brain_observatory.ecephys`), 166  
 num\_probes (allensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession), 206 `lensdk.brain_observatory.static_gratings.StaticGratings` (al-  
 attribute), 166 `lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` (class in `lensdk.brain_observatory.ecephys`), 166  
 NUM\_ROWS (allensdk.api.queries.rma\_api.RmaApi attribute), number\_speeds (al- attribute), 65  
 num\_stimulus\_presentations (al- attribute), 133 `lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` (class in `lensdk.brain_observatory.ecephys`), 166  
 attribute), 166 `lensdk.brain_observatory.drifting_gratings.DriftingGratings` (al- attribute), 192  
 num\_trees (allensdk.core.swc.Morphology attribute), number\_tf (allensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_gratings.DriftingGratings), 134  
 262 attribute), 134  
 num\_trees (allensdk.internal.morphology.morphology.Morphology), cells (allensdk.brain\_observatory.stimulus\_analysis.StimulusAnalysis), 208  
 attribute), 330 `lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` (class in `lensdk.brain_observatory.ecephys`), 166  
 num\_units (allensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession), 167 `lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` (class in `lensdk.brain_observatory.ecephys`), 167  
 attribute), 167 `lensdk.api.queries.brain_observatory_api.BrainObservatoryApi` (class in `lensdk.api.queries`), 46  
 number\_azimuths (al- attribute), 46 `lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.ReceptiveFieldMapping` (al-  
 attribute), 139 `lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.ReceptiveFieldMapping` (al-  
 attribute), 139 `lensdk.api.queries.cell_types_api.CellTypesApi` (class in `lensdk.api.queries`), 49  
 number\_colors (al- attribute), 49 `lensdk.brain_observatory.ecephys.stimulus_analysis.flashes.Flashes` (al-  
 attribute), 136 `lensdk.brain_observatory.ecephys.stimulus_analysis.flashes.Flashes` (al-  
 attribute), 52 `lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.DriftingGratings` (al-  
 number\_contrast (al- NwbApi (class in `lensdk.brain_observatory.nwb.nwb_api`), 170  
 attribute), 134 `lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.DriftingGratings` (al-  
 attribute), 134 `lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.DriftingGratings` (al-  
 number\_directions (al- NwbDataSet (class in `allensdk.core.nwb_data_set`), 170  
 attribute), 133 `lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.DriftingGratings` (al-  
 attribute), 133 `lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.DriftingGratings` (al-  
 number\_elevations (al- nwbfile (allensdk.brain\_observatory.nwb.nwb\_api.NwbApi), 170  
 attribute), 133 `lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.DriftingGratings` (al-  
 attribute), 133 `lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.ReceptiveFieldMapping` (al-  
 attribute), 139 `lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.ReceptiveFieldMapping` (al-  
 attribute), 139



## O

`object_norm_eye_coordinates()` (in module `al-lensdk.internal.brain_observatory.eye_calibration`), 293  
`object_rotation_matrix()` (in module `al-lensdk.internal.brain_observatory.eye_calibration`), 293  
`one()` (in module `allensdk`), 374  
`one_file_call_caching()` (in module `al-lensdk.api.caching_utilities`), 80  
`OneOrMoreResultExpectedError`, 287  
`OneResultExpectedError`, 374  
`ONLY` (`allensdk.api.queries.rma_api.RmaApi` attribute), 65  
`only_except_tabular_clause()` (`al-lensdk.api.queries.rma_api.RmaApi` method), 68  
`OntologiesApi` (class in `al-lensdk.api.queries.ontologies_api`), 61  
`Ontology` (class in `allensdk.core.ontology`), 248  
`open()` (`allensdk.internal.brain_observatory.frame_stream.CvInputStream` method), 294  
`open()` (`allensdk.internal.brain_observatory.frame_stream.FfmpegInputStream` method), 294  
`open()` (`allensdk.internal.brain_observatory.frame_stream.FfmpegOutputStream` method), 295  
`open()` (`allensdk.internal.brain_observatory.frame_stream.FrameInputStream` method), 295  
`open()` (`allensdk.internal.brain_observatory.frame_stream.FrameOutputStream` method), 295  
`open_corona_plot()` (`al-lensdk.brain_observatory.natural_scenes.NaturalScenes` method), 195  
`open_fan_plot()` (`al-lensdk.brain_observatory.static_gratings.StaticGratings` method), 206  
`open_pincushion_plot()` (`al-lensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` method), 193  
`open_star_plot()` (`al-lensdk.brain_observatory.drifting_gratings.DriftingGratings` method), 192  
`open_track_plot()` (`al-lensdk.brain_observatory.natural_movie.NaturalMovie` method), 194  
`open_view_on_binary()` (in module `al-lensdk.internal.brain_observatory.ophys_session_decomposition`), 297  
`OPHYS_ANALYSIS_FILE_TYPE` (`al-lensdk.api.queries.brain_observatory_api.BrainObservatoryApi` attribute), 46  
`OPHYS_ANALYSIS_LOG_KEY` (`al-lensdk.brain_observatory.behavior.behavior_project_cache.BehaviorProjectCache` attribute), 95  
`ophys_delta` (`allensdk.internal.pipeline_modules.run_ophys_time_sync` attribute), 349  
`OPHYS_EVENTS_FILE_TYPE` (`al-lensdk.api.queries.brain_observatory_api.BrainObservatoryApi` attribute), 46  
`ophys_experiment_id` (`al-lensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession` attribute), 94  
`ophys_experiment_ids` (`al-lensdk.brain_observatory.behavior.behavior_data_session.BehaviorDataSession` attribute), 90  
`OPHYS_EXPERIMENTS_KEY` (`al-lensdk.brain_observatory.behavior.behavior_project_cache.BehaviorProjectCache` attribute), 95  
`ophys_session_id` (`al-lensdk.brain_observatory.behavior.behavior_data_session.BehaviorDataSession` attribute), 90  
`OPHYS_SESSIONS_KEY` (`al-lensdk.brain_observatory.behavior.behavior_project_cache.BehaviorProjectCache` attribute), 95  
`ophys_times` (`allensdk.internal.pipeline_modules.run_ophys_time_sync` attribute), 349  
`ophys_timestamps` (`al-lensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession` attribute), 94  
`ophys_timestamps` (`al-lensdk.internal.brain_observatory.time_sync.OphysTimeAligner` attribute), 300  
`OphysBehaviorMetadataSchema` (class in `al-lensdk.brain_observatory.behavior.schemas`), 106  
`OphysBehaviorTaskParametersSchema` (class in `al-lensdk.brain_observatory.behavior.schemas`), 106  
`OphysLimsApi` (class in `al-lensdk.internal.api.ophys_lims_api`), 286  
`OphysTimeAligner` (class in `al-lensdk.internal.brain_observatory.time_sync`), 300  
`OptimizeConfigReader` (class in `al-lensdk.internal.api.queries.optimize_config_reader`), 280  
`optional_lims_inputs()` (in module `al-lensdk.brain_observatory.argschema_utilities`), 185  
`optional_polys` (`al-lensdk.mouse_connectivity.grid.subimage.base_subimage.PolygonSubimage` attribute), 366  
`optional_polys` (`al-lensdk.mouse_connectivity.grid.subimage.classic_subimage.ClassicSubimage` attribute), 369  
`OPTIONS` (`allensdk.api.queries.rma_api.RmaApi` attribute), 65

tribute), 65

options\_clause() (allensdk.api.queries.rma\_api.RmaApi method), 68

OPTOGENETIC\_STIMULATION\_KEYS (allensdk.brain\_observatory.sync\_dataset.Dataset attribute), 213

opts (allensdk.brain\_observatory.argschema\_utilities.RaisingSchema attribute), 185

opts (allensdk.brain\_observatory.behavior.mtrain.ExtendedTrialSchema attribute), 104

opts (allensdk.brain\_observatory.behavior.schemas.OphysBehaviorMaskSet attribute), 106

opts (allensdk.brain\_observatory.behavior.schemas.OphysBehaviorTaskParametersSchema attribute), 107

opts (allensdk.brain\_observatory.behavior.schemas.RaisingSchema attribute), 107

opts (allensdk.brain\_observatory.nwb.schemas.RunningSpeedPathSchema attribute), 171

ORDER (allensdk.api.queries.rma\_api.RmaApi attribute), 65

order\_clause() (allensdk.api.queries.rma\_api.RmaApi method), 68

order\_rois\_by\_object\_list() (in module allensdk.internal.brain\_observatory.roi\_filter\_utils), 299

organize\_sweeps\_by\_name() (in module allensdk.internal.model.glif.find\_sweeps), 316

orivals (allensdk.brain\_observatory.drifting\_gratings.DriftingGratings attribute), 192

orivals (allensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_gratings.DriftingGratings attribute), 134

orivals (allensdk.brain\_observatory.ecephys.stimulus\_analysis.static\_gratings.StaticGratings attribute), 142

orivals (allensdk.brain\_observatory.static\_gratings.StaticGratings attribute), 206

osi() (in module allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis), 146

outdated (allensdk.config.manifest.ManifestVersionError attribute), 224

outlier\_cost() (allensdk.internal.brain\_observatory.fit\_ellipse.FitEllipse method), 294

output() (allensdk.internal.mouse\_connectivity.interval\_unionize\_tissue\_to\_unionize\_record.TissueToUnionizeRecord method), 338

output() (allensdk.internal.mouse\_connectivity.interval\_unionize\_unionize\_record.Unionize module allensdk.internal.pipeline\_modules.run\_demixing), 339

output\_directory() (allensdk.internal.api.queries.optimize\_config\_reader.OptimizeConfigReader method), 280

OutputFile (class in allensdk.brain\_observatory.argschema\_utilities), 184

OutputGrabber (class in allensdk.internal.model.biophysical.passive\_fitting.output\_grabber), 312

overall\_firing\_rate() (in module allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis), 146

overlaps\_fraction() (allensdk.internal.brain\_observatory.mask\_set.MaskSet method), 297

overlaps\_motion\_border (allensdk.brain\_observatory.roi\_masks.Mask attribute), 199

**P**

page\_size() (in module allensdk.api.queries.rma\_pager), 70

pager (allensdk.api.queries.rma\_pager.RmaPager static method), 70

panel\_size (allensdk.brain\_observatory.stimulus\_info.Monitor attribute), 209

ParamsMixin (class in allensdk.brain\_observatory.session\_api\_utils), 204

parent() (allensdk.core.simple\_tree.SimpleTree method), 255

parent() (allensdk.internal.core.simpletree.SimpleTree method), 304

parent\_id() (allensdk.core.simple\_tree.SimpleTree method), 256

parent\_id() (allensdk.internal.core.simpletree.SimpleTree method), 304

parent\_ids() (allensdk.core.simple\_tree.SimpleTree method), 256

parent\_of() (allensdk.core.swc.Morphology method), 262

parent\_of() (allensdk.internal.morphology.morphology.Morphology method), 330

parse\_arguments() (in module allensdk.internal.model.glif.find\_sweeps), 316

parse\_arguments() (in module allensdk.model.glif.simulate\_neuron), 365

parse\_command\_line\_args() (allensdk.config.application\_config.ApplicationConfig method), 218

unionize (module allensdk.internal.pipeline\_modules.run\_demixing), 345

unionize\_record (in module allensdk.internal.pipeline\_modules.run\_dff\_computation), 345

[parse\\_input\(\)](#) (in module `allensdk.internal.pipeline_modules.run_observatory_pipeline.py`), 69  
[parse\\_input\(\)](#) (in module `allensdk.internal.pipeline_modules.run_ophys_session.py`), 348  
[parse\\_input\(\)](#) (in module `allensdk.internal.pipeline_modules.run_ophys_session.py`), 349  
[parse\\_input\\_data\(\)](#) (in module `allensdk.internal.pipeline_modules.run_ophys_eye_calibration.py`), 348  
[parse\\_neuron\\_output\(\)](#) (in module `allensdk.internal.model.biophysical.passive_fitting.py`), 311  
[parse\\_obj\(\)](#) (in module `allensdk.core.obj_utilities`), 247  
[parse\\_stim\\_repr\(\)](#) (in module `allensdk.brain_observatory.ecephys.stimulus_table.py`), 155  
[parser\\_for\\_extension\(\)](#) (in module `allensdk.config.model.description_parser.py`), 222  
[password\(\)](#) (in module `allensdk.core.authentication.DbCredentials`), 226  
[paste\\_slice\(\)](#) (in module `allensdk.mouse_connectivity.grid.image_series_gridder.py`), 372  
[paste\\_subimage\(\)](#) (in module `allensdk.mouse_connectivity.grid.image_series_gridder.py`), 373  
[path\(\)](#) (in module `allensdk.brain_observatory.nwb.nwb_api.NwbApi`), 170  
[path\\_to\\_list\(\)](#) (in module `allensdk.core.structure_tree.StructureTree`), 259  
[pathfinder\(\)](#) (in module `allensdk.api.cache.Cache`), 77  
[pause\\_metrics\(\)](#) (in module `allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor`), 266  
[peak\(\)](#) (in module `allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis`), 208  
[peak\\_run\(\)](#) (in module `allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis`), 208  
[period\(\)](#) (in module `allensdk.brain_observatory.sync_dataset.Dataset`), 215  
[phasevals\(\)](#) (in module `allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings`), 142  
[phasevals\(\)](#) (in module `allensdk.brain_observatory.static_gratings.StaticGratings`), 206  
[PHOTODIODE\\_KEYS](#) (in module `allensdk.brain_observatory.sync_dataset.Dataset`), 213  
[PIPE](#) (in module `allensdk.api.queries.rma_api.RmaApi`), 65  
[pipe\\_stage\(\)](#) (in module `allensdk.api.queries.rma_api.RmaApi`), 65  
[pixels\\_to\\_visual\\_degrees\(\)](#) (in module `allensdk.brain_observatory.stimulus_info.BrainObservatoryMonitor`), 209  
[plot\(\)](#) (in module `allensdk.brain_observatory.circle_plots.FanPlotter`), 187  
[plot\(\)](#) (in module `allensdk.brain_observatory.circle_plots.TrackPlotter`), 187  
[plot\\_all\(\)](#) (in module `allensdk.brain_observatory.sync_dataset.Dataset`), 215  
[plot\\_bits\(\)](#) (in module `allensdk.brain_observatory.sync_dataset.Dataset`), 215  
[plot\\_blocks\(\)](#) (in module `allensdk.brain_observatory.ecephys.stimulus_table.visualization.visualize`), 148  
[plot\\_cell\\_correlation\(\)](#) (in module `allensdk.brain_observatory.observatory_plots`), 196  
[plot\\_cell\\_figures\(\)](#) (in module `allensdk.internal.ephys.plot_qc_figures`), 306  
[plot\\_cell\\_figures3\(\)](#) (in module `allensdk.internal.ephys.plot_qc_figures3`), 306  
[plot\\_cell\\_receptive\\_field\(\)](#) (in module `allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise`), 193  
[plot\\_combined\\_speed\(\)](#) (in module `allensdk.brain_observatory.observatory_plots`), 196  
[plot\\_condition\\_histogram\(\)](#) (in module `allensdk.brain_observatory.observatory_plots`), 196  
[plot\\_conditionwise\\_raster\(\)](#) (in module `allensdk.brain_observatory.observatory_plots`), 196

`lensdk.brain_observatory.ecephys.stimulus_analysis.StimulusAnalysis` module  
`method`), 144  
`plot_direction_selectivity()` (al- `lensdk.brain_observatory.brain_observatory_plotting`), 185  
`plot_drifting_grating_traces()` (in module `lensdk.brain_observatory.receptive_field_analysis.visualization`), 182  
`plot_ellipses()` (in module `al- plot_mask_outline()` (in module `al- lensdk.brain_observatory.brain_observatory_plotting`), `lensdk.brain_observatory.observatory_plots`), 185  
`plot_example_traces_and_behavior()` (al- `plot_masks()` (in module `al- lensdk.brain_observatory.receptive_field_analysis.visualization`), 181  
`plot_example_traces_and_behavior()` (al- `plot_mean_waveforms()` (in module `al- lensdk.brain_observatory.behavior.behavior_ophys_analysis.BehaviorOphysAnalysis`), `lensdk.brain_observatory.ecephys.visualization`), `method`), 91  
`plot_example_traces_and_behavior()` (in module `al- plot_msr_summary()` (in module `al- lensdk.brain_observatory.receptive_field_analysis.visualization`), `lensdk.brain_observatory.behavior.behavior_ophys_analysis`), 91  
`plot_fi_curve_figures()` (in module `al- plot_negative_baselines()` (in module `al- lensdk.internal.ephys.plot_qc_figures`), 306  
`plot_fi_curve_figures()` (in module `al- plot_negative_baselines()` (in module `al- lensdk.internal.ephys.plot_qc_figures3`), 309  
`plot_fields()` (in module `al- plot_negative_transients()` (in module `al- lensdk.brain_observatory.receptive_field_analysis.visualization`), 182  
`plot_gaussian_fit()` (in module `al- plot_negative_transients()` (in module `al- lensdk.internal.brain_observatory.demixer`), `lensdk.brain_observatory.receptive_field_analysis.visualization`), 182  
`plot_hero_figures()` (in module `al- plot_ns_traces()` (in module `al- lensdk.internal.ephys.plot_qc_figures`), 306  
`plot_hero_figures()` (in module `al- plot_onetrace()` (in module `al- lensdk.internal.ephys.plot_qc_figures3`), 309  
`plot_images()` (in module `al- plot_orientation_selectivity()` (al- `lensdk.internal.ephys.plot_qc_figures`), 306  
`plot_images()` (in module `al- plot_orientation_selectivity()` (al- `lensdk.internal.ephys.plot_qc_figures3`), 309  
`plot_instantaneous_threshold_thumbnail()` (in module `al- plot_overlap_masks_lengthOne()` (in module `al- lensdk.internal.ephys.plot_qc_figures`), 307  
`plot_instantaneous_threshold_thumbnail()` (in module `al- plot_overlap_masks_lengthOne()` (in module `al- lensdk.internal.ephys.plot_qc_figures3`), 309  
`plot_line()` (`allensdk.brain_observatory.sync_dataset.Dataset` `method`), 216  
`plot_lines()` (`allensdk.brain_observatory.sync_dataset.Dataset` `method`), 216  
`plot_long_square_summary()` (in module `al- plot_p_values()` (in module `al- lensdk.internal.ephys.plot_qc_figures`), 307  
`plot_long_square_summary()` (in module `al- plot_population_receptive_field()` (al- `lensdk.internal.ephys.plot_qc_figures3`), 309  
`plot_mask_outline()` (in module `al- plot_preferred_direction()` (al- `lensdk.brain_observatory.brain_observatory_plotting`), 185  
`plot_masks()` (in module `al- plot_preferred_orientation()` (al- `lensdk.brain_observatory.receptive_field_analysis.visualization`), 181  
`plot_mean_waveforms()` (in module `al- plot_preferred_orientation()` (al- `lensdk.brain_observatory.ecephys.visualization`), 156  
`plot_msr_summary()` (in module `al- plot_preferred_orientation()` (al- `lensdk.brain_observatory.receptive_field_analysis.visualization`), 182  
`plot_negative_baselines()` (in module `al- plot_preferred_orientation()` (al- `lensdk.brain_observatory.demixer`), 189  
`plot_negative_baselines()` (in module `al- plot_preferred_orientation()` (al- `lensdk.internal.brain_observatory.demixer`), 290  
`plot_negative_transients()` (in module `al- plot_preferred_orientation()` (al- `lensdk.brain_observatory.demixer`), 189  
`plot_ns_traces()` (in module `al- plot_preferred_orientation()` (al- `lensdk.brain_observatory.brain_observatory_plotting`), 186  
`plot_onetrace()` (in module `al- plot_preferred_orientation()` (al- `lensdk.brain_observatory.dff`), 191  
`plot_orientation_selectivity()` (al- `plot_preferred_orientation()` (al- `lensdk.brain_observatory.drifting_gratings.DriftingGratings` `method`), 192  
`plot_orientation_selectivity()` (al- `plot_preferred_orientation()` (al- `lensdk.brain_observatory.static_gratings.StaticGratings` `method`), 206  
`plot_overlap_masks_lengthOne()` (in module `al- plot_preferred_orientation()` (al- `allensdk.brain_observatory.demixer`), 189  
`plot_overlap_masks_lengthOne()` (in module `al- plot_preferred_orientation()` (al- `allensdk.internal.brain_observatory.demixer`), 290  
`plot_p_values()` (in module `al- plot_preferred_orientation()` (al- `lensdk.brain_observatory.receptive_field_analysis.visualization`), 182  
`plot_population_receptive_field()` (al- `plot_preferred_orientation()` (al- `lensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` `method`), 193  
`plot_preferred_direction()` (al- `plot_preferred_orientation()` (al- `lensdk.brain_observatory.drifting_gratings.DriftingGratings` `method`), 192  
`plot_preferred_orientation()` (al- `plot_preferred_orientation()` (al- `lensdk.brain_observatory.drifting_gratings.DriftingGratings` `method`), 192



`lensdk.brain_observatory.static_gratings.StaticGratings`  
`method)`, 206

`plot_preferred_spatial_frequency()` (al-  
`lensdk.brain_observatory.static_gratings.StaticGratings`  
`method)`, 206

`plot_preferred_temporal_frequency()` (al-  
`lensdk.brain_observatory.drifting_gratings.DriftingGratings`  
`method)`, 192

`plot_pupil_location()` (in module al-  
`lensdk.brain_observatory.observatory_plots`),  
196

`plot_radial_histogram()` (in module al-  
`lensdk.brain_observatory.observatory_plots`),  
196

`plot_ramp_figures()` (in module al-  
`lensdk.internal.ephys.plot_qc_figures`), 307

`plot_ramp_figures()` (in module al-  
`lensdk.internal.ephys.plot_qc_figures3`),  
309

`plot_raster()` (al-  
`lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.DriftingGratings`  
`method)`, 134

`plot_raster()` (al-  
`lensdk.brain_observatory.ecephys.stimulus_analysis.flashes.Flashes`  
`method)`, 136

`plot_raster()` (al-  
`lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.ReceptiveFieldMapping`  
`method)`, 139

`plot_raster()` (al-  
`lensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings` in module al-  
`lensdk.internal.ephys.plot_qc_figures3`),  
309

`plot_raster()` (al-  
`lensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.StimulusAnalysis`  
`method)`, 144

`plot_receptive_field()` (in module al-  
`lensdk.brain_observatory.observatory_plots`),  
196

`plot_receptive_field_analysis_data()`  
(al-  
`lensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise`  
`method)`, 193

`plot_receptive_field_data()` (in module al-  
`lensdk.brain_observatory.receptive_field_analysis.visualization`),  
182

`plot_representational_similarity()` (al-  
`lensdk.brain_observatory.stimulus_analysis.StimulusAnalysis`  
`method)`, 208

`plot_representational_similarity()`  
(in module al-  
`lensdk.brain_observatory.observatory_plots`),  
197

`plot_response()` (al-  
`lensdk.brain_observatory.ecephys.stimulus_analysis.flashes.Flashes`  
`method)`, 136

`plot_response_summary()` (al-  
`lensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings.DriftingGratings`  
`method)`, 135

`plot_response_summary()` (al-  
`lensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings`  
`method)`, 142

`plot_rf()` (al-  
`lensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping.ReceptiveFieldMapping`  
`method)`, 139

`plot_rheo_figures()` (in module al-  
`lensdk.internal.ephys.plot_qc_figures`), 307

`plot_rheo_figures()` (in module al-  
`lensdk.internal.ephys.plot_qc_figures3`),  
309

`plot_rts_blur_summary()` (in module al-  
`lensdk.brain_observatory.receptive_field_analysis.visualization`),  
182

`plot_rts_summary()` (in module al-  
`lensdk.brain_observatory.receptive_field_analysis.visualization`),  
182

`plot_running_a()` (in module al-  
`lensdk.brain_observatory.brain_observatory_plotting`),  
186

`plot_running_speed()` (in module al-  
`lensdk.brain_observatory.visualization`),  
182

`plot_running_speed_histogram()` (al-  
`lensdk.brain_observatory.stimulus_analysis.StimulusAnalysis`  
`method)`, 208

`plot_sag_figures()` (in module al-  
`lensdk.internal.ephys.plot_qc_figures`), 307

`plot_sg_figures()` (in module al-  
`lensdk.internal.ephys.plot_qc_figures3`), 310

`plot_sg_traces()` (in module al-  
`lensdk.brain_observatory.brain_observatory_plotting`),  
186

`plot_short_square_figures()` (in module al-  
`lensdk.internal.ephys.plot_qc_figures`), 307

`plot_square_figures()` (in module al-  
`lensdk.internal.ephys.plot_qc_figures3`), 310

`plot_single_ap_values()` (in module al-  
`lensdk.internal.ephys.plot_qc_figures`), 307

`plot_single_ap_values()` (in module al-  
`lensdk.internal.ephys.plot_qc_figures3`), 310

`plot_speed()` (in module al-  
`lensdk.brain_observatory.observatory_plots`),  
197

`plot_speed_tuning()` (al-  
`lensdk.brain_observatory.stimulus_analysis.StimulusAnalysis`  
`method)`, 208

`plot_spike_counts()` (in module al-

<code>lensdk.brain_observatory.ecephys.visualization)</code> , 156	<code>plotSpikes()</code> (in module <code>lensdk.internal.model.glif.plotting</code> ), 322
<code>plot_subthreshold_long_square_figures()</code> (in module <code>lensdk.internal.ephys.plot_qc_figures</code> ), 307	<code>polar_line_circles()</code> (in module <code>lensdk.brain_observatory.circle_plots</code> ), 188
<code>plot_subthreshold_long_square_figures3()</code> (in module <code>lensdk.internal.ephys.plot_qc_figures3</code> ), 310	<code>polar_linspace()</code> (in module <code>lensdk.brain_observatory.circle_plots</code> ), 188
<code>plot_sweep_figures()</code> (in module <code>lensdk.internal.ephys.plot_qc_figures</code> ), 307	<code>polar_to_xy()</code> (in module <code>lensdk.brain_observatory.circle_plots</code> ), 188
<code>plot_sweep_figures3()</code> (in module <code>lensdk.internal.ephys.plot_qc_figures3</code> ), 310	<code>PolarPlotter</code> (class in <code>lensdk.brain_observatory.circle_plots</code> ), 187
<code>plot_sweep_set_summary()</code> (in module <code>lensdk.internal.ephys.plot_qc_figures</code> ), 307	<code>PolygonSubImage</code> (class in <code>lensdk.mouse_connectivity.grid.subimage.base_subimage</code> ), 366
<code>plot_sweep_set_summary3()</code> (in module <code>lensdk.internal.ephys.plot_qc_figures3</code> ), 310	<code>populate_stimulus_table()</code> (in <code>lensdk.brain_observatory.drifting_gratings.DriftingGratings</code> method), 192
<code>plot_sweep_value_figures()</code> (in module <code>lensdk.internal.ephys.plot_qc_figures</code> ), 308	<code>populate_stimulus_table()</code> (in <code>lensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise</code> method), 193
<code>plot_sweep_value_figures3()</code> (in module <code>lensdk.internal.ephys.plot_qc_figures3</code> ), 310	<code>populate_stimulus_table()</code> (in <code>lensdk.brain_observatory.natural_movie.NaturalMovie</code> method), 194
<code>plot_time_to_peak()</code> (in <code>lensdk.brain_observatory.natural_scenes.NaturalScenes</code> method), 195	<code>populate_stimulus_table()</code> (in <code>lensdk.brain_observatory.natural_scenes.NaturalScenes</code> method), 195
<code>plot_time_to_peak3()</code> (in <code>lensdk.brain_observatory.static_gratings.StaticGratings</code> method), 206	<code>populate_stimulus_table()</code> (in <code>lensdk.brain_observatory.static_gratings.StaticGratings</code> method), 206
<code>plot_time_to_peak4()</code> (in module <code>lensdk.brain_observatory.observatory_plots</code> ), 197	<code>populate_stimulus_table()</code> (in <code>lensdk.brain_observatory.stimulus_analysis.StimulusAnalysis</code> method), 208
<code>plot_trace()</code> (in module <code>lensdk.brain_observatory.behavior.behavior_ophys_analysis</code> ), 91	<code>population_correlation_scatter()</code> (in module <code>lensdk.brain_observatory.observatory_plots</code> ), 197
<code>plot_traces()</code> (in module <code>lensdk.brain_observatory.demixer</code> ), 189	<code>port</code> ( <code>allensdk.core.authentication.DbCredentials</code> attribute), 226
<code>plot_traces3()</code> (in module <code>lensdk.internal.brain_observatory.demixer</code> ), 290	<code>POSITIVE</code> ( <code>allensdk.core.cell_types_cache.ReporterStatus</code> attribute), 238
<code>plot_transients()</code> (in module <code>lensdk.brain_observatory.demixer</code> ), 189	<code>post_process_cr()</code> (in module <code>lensdk.internal.brain_observatory.itracker_utils</code> ), 296
<code>plot_transients3()</code> (in module <code>lensdk.internal.brain_observatory.demixer</code> ), 290	<code>post_process_pupil()</code> (in module <code>lensdk.internal.brain_observatory.itracker_utils</code> ), 296
<code>plotLineRegress1()</code> (in module <code>lensdk.internal.model.glif.plotting</code> ), 322	<code>postgres_macros()</code> (in module <code>lensdk.brain_observatory.ecephys.ecephys_project_api.utilities</code> ), 125
<code>plotLineRegress13()</code> (in module <code>lensdk.internal.model.glif.spike_cutting</code> ), 323	<code>PostgresQueryMixin</code> (class in <code>lensdk.internal.api</code> ), 287
<code>plotLineRegressRed()</code> (in module <code>lensdk.internal.model.glif.plotting</code> ), 322	<code>postprocess_unionizes()</code> (in <code>lensdk.internal.mouse_connectivity.interval_unionize.interval_unionize</code> method), 336
<code>plotLineRegressRed3()</code> (in module <code>lensdk.internal.model.glif.spike_cutting</code> ), 323	<code>postprocess_unionizes2()</code> (in <code>lensdk.internal.mouse_connectivity.interval_unionize.interval_unionize</code> method), 336

lensdk.internal.mouse\_connectivity.interval\_unionize.tissucyte\_u  
method), 339  
pre\_blank\_sec (al- process\_segmentation() (al-  
lensdk.brain\_observatory.ecephys.file\_io.stim\_file.CamStimOmsPkltoStimFile  
attribute), 129  
prepare\_nwb\_output() (in module al- process\_segmentation() (al-  
lensdk.model.biophysical.runner), 354  
presentationwise\_spike\_counts() (al-  
lensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession  
method), 167  
presentationwise\_spike\_times (al-  
lensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis.StimulusAnalysis (al-  
attribute), 144  
presentationwise\_spike\_times() (al-  
lensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession  
method), 167  
presentationwise\_statistics (al-  
lensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis.StimulusAnalysis  
attribute), 145  
PREVIEW (allensdk.api.queries.rma\_api.RmaApi at- project\_to\_plane() (in module al-  
tribute), 65  
print\_node() (allensdk.core.swc.Compartment PROJECTION\_DENSITY (al-  
method), 260  
print\_out() (allensdk.ephys.feature\_extractor.EphysFeatures lensdk.api.queries.grid\_data\_api.GridDataApi  
method), 276  
print\_summary() (in module al- projection\_density (al-  
lensdk.brain\_observatory.receptive\_field\_analysis.receptive\_field\_analysis.ReceptiveFieldAnalysis  
180  
PROBE\_LFP\_NWB\_KEY (al- PROJECTION\_DENSITY\_KEY (al-  
lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache  
attribute), 158  
PROBES\_KEY (allensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache (al-  
attribute), 158  
ProbeSynchronizer (class in al- attribute), 53  
lensdk.brain\_observatory.ecephys.align\_timestamps.probe\_synchronizer (al-  
115  
process() (allensdk.ephys.ephys\_extractor.EphysCellFeatureExtractor attribute), 338  
method), 264  
process\_eye\_tracking\_data() (in module al- projection\_intensity (al-  
lensdk.brain\_observatory.behavior.eye\_tracking\_processing.attribute), 338  
103  
process\_inputs() (in module al- propagate() (allensdk.internal.mouse\_connectivity.interval\_unionize.tissucyte\_u  
lensdk.internal.model.biophysical.passive\_fitting.neuron\_passive\_fit) allensdk.internal.mouse\_connectivity.interval\_unionize.ur  
311  
process\_instance() (al- propagate\_record() (al-  
lensdk.ephys.feature\_extractor.EphysFeatureExtractor lensdk.internal.mouse\_connectivity.interval\_unionize.interval\_un  
method), 276  
process\_new\_spike\_feature() (al- propagate\_record() (al-  
lensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor lensdk.internal.mouse\_connectivity.interval\_unionize.tissucyte\_u  
method), 266  
process\_new\_sweep\_feature() (al- propagate\_to\_bilateral() (al-  
lensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor lensdk.internal.mouse\_connectivity.interval\_unionize.interval\_un  
method), 266  
process\_segmentation() (al- propagate\_unionizes() (al-

lensdk.internal.mouse\_connectivity.interval\_unionize.IntervalUnionizer  
 class method), 337

provide() (allensdk.core.authentication.CredentialProvider  
 method), 226

provide() (allensdk.core.authentication.EnvCredentialProvider  
 method), 226

psycpg2\_select() (in module al-  
 lensdk.internal.api), 287

published\_at (allensdk.brain\_observatory.ecephys.ecephys\_project\_api.ecephys\_project\_cache.Extractor  
 attribute), 121

published\_at\_not\_null (al-  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_api.ecephys\_project\_cache.Extractor  
 attribute), 121

pupil\_position\_in\_mouse\_eye\_coordinates (rank\_structures() (al-  
 (allensdk.internal.brain\_observatory.eye\_calibration.EyeCalibration  
 method), 292

pupil\_position\_on\_monitor\_in\_cm() (al-  
 lensdk.internal.brain\_observatory.eye\_calibration.EyeCalibration  
 method), 292

pupil\_position\_on\_monitor\_in\_degrees() (allensdk.internal.brain\_observatory.eye\_calibration.EyeCalibration  
 method), 292

push\_summary() (al-  
 lensdk.ephys.feature\_extractor.EphysFeatureExtractor  
 method), 276

pval (allensdk.brain\_observatory.stimulus\_analysis.StimulusAnalysis  
 attribute), 208

pvalue\_to\_NLL() (in module al-  
 lensdk.brain\_observatory.receptive\_field\_analysis.chisquare\_util  
 method), 177

pvalue\_to\_NLL() (in module al-  
 lensdk.brain\_observatory.receptive\_field\_analysis.result\_util  
 method), 182

PycfgDescriptionParser (class in al-  
 lensdk.config.model.formats.pycfg\_description\_parser), 220

**Q**

query() (in module al-  
 lensdk.internal.core.lims\_utilities), 302

quote\_string() (al-  
 lensdk.api.queries.rma\_api.RmaApi  
 method), 69

**R**

radial\_arcs() (in module al-  
 lensdk.brain\_observatory.circle\_plots), 188

radial\_circles() (in module al-  
 lensdk.brain\_observatory.circle\_plots), 188

RaisingSchema (class in al-  
 lensdk.brain\_observatory.argschema\_utilities), 184

RaisingSchema (class in al-  
 lensdk.brain\_observatory.behavior.schemas),

RaisingSchema.Meta (class in al-  
 lensdk.brain\_observatory.argschema\_utilities), 185

RaisingSchema.Meta (class in al-  
 lensdk.brain\_observatory.behavior.schemas), 107

ramps\_features() (al-  
 lensdk.ephys.feature\_extractor.EphysFeatureExtractor  
 method), 265

randomize\_parameter\_values() (al-  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_api.ecephys\_project\_cache.Extractor  
 method), 318

rank\_structures() (al-  
 lensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache  
 method), 245

ransac\_fit() (allensdk.internal.brain\_observatory.fit\_ellipse.FitEllipse  
 method), 294

raster\_plot() (in module al-  
 lensdk.brain\_observatory.ecephys.visualization), 174

rasterize\_polygons() (in module al-  
 lensdk.mouse\_connectivity.grid.utilities.image\_utilities), 371

ratio\_and\_pyramid() (in module al-  
 lensdk.mouse\_connectivity.grid.writers), 372

read() (allensdk.config.model.description\_parser.DescriptionParser  
 method), 222

read() (allensdk.config.model.formats.hdf5\_util.Hdf5Util  
 method), 219

read() (allensdk.config.model.formats.json\_description\_parser.JsonDescriptionParser  
 method), 219

read() (allensdk.config.model.formats.pycfg\_description\_parser.PycfgDescriptionParser  
 method), 220

read() (in module allensdk.core.json\_utilities), 240

read() (in module al-  
 lensdk.internal.mouse\_connectivity.interval\_unionize.data\_utilities), 336

read\_cell\_index\_receptive\_field\_analysis() (allensdk.brain\_observatory.locally\_sparse\_noise.LocallySparseNoise  
 static method), 193

read\_csv() (in module al-  
 lensdk.brain\_observatory.ecephys.ecephys\_project\_cache), 162

read\_data() (allensdk.api.api.Api method), 74

read\_eye\_dlc\_tracking\_ellipses() (in mod-  
 ule allensdk.brain\_observatory.nwb), 174

read\_eye\_gaze\_mappings() (in module al-  
 lensdk.brain\_observatory.nwb), 174

read\_file() (allensdk.core.json\_utilities.JsonComments  
 class method), 239

read\_h5\_group() (in module al-  
 lensdk.brain\_observatory.receptive\_field\_analysis.receptive\_field\_analysis),



[180](#)  
[read\\_h5\\_group\(\)](#) (in module [al-lensdk.brain\\_observatory.receptive\\_field\\_analysis.tools](#)), [lensdk.brain\\_observatory.receptive\\_field\\_analysis.receptive\\_field](#)  
[180](#)  
[read\\_intensity\\_image\(\)](#) (in module [al-lensdk.mouse\\_connectivity.grid.utilities.image\\_utilities](#)), [lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache](#),  
[371](#) [162](#)  
[read\\_json\(\)](#) ([allensdk.api.queries.biophysical\\_api.BiophysicalAPI](#) [segmentation\\_image\(\)](#) (al-  
[method](#)), [46](#) [lensdk.mouse\\_connectivity.grid.subimage.base\\_subimage.Segment](#)  
[read\\_json\(\)](#) ([allensdk.internal.api.queries.biophysical\\_module\\_reader.BiophysicalModuleReader](#)  
[method](#)), [278](#) [read\\_segmentation\\_image\(\)](#) (in module [al-](#)  
[read\\_json\(\)](#) ([allensdk.internal.api.queries.optimize\\_config\\_reader.OptimizeConfigReader](#) [lensdk.mouse\\_connectivity.grid.utilities.image\\_utilities](#)),  
[method](#)), [280](#) [371](#)  
[read\\_json\\_string\(\)](#) (al- [read\\_stimulus\(\)](#) (al-  
[lensdk.internal.api.queries.biophysical\\_module\\_reader.BiophysicalModuleReader](#) [lensdk.model.biophys\\_utils.Utils](#) [method](#)),  
[method](#)), [278](#) [355](#)  
[read\\_json\\_string\(\)](#) (al- [read\\_stimulus\\_name\\_from\\_path\(\)](#)  
[lensdk.internal.api.queries.optimize\\_config\\_reader.OptimizeConfigReader](#) [module](#) [al-](#)  
[method](#)), [280](#) [lensdk.brain\\_observatory.ecephys.stimulus\\_table.ephys\\_pre\\_spike](#)  
[read\\_lims\\_file\(\)](#) (al- [152](#)  
[lensdk.internal.api.queries.biophysical\\_module\\_reader.BiophysicalModuleReader](#) [module](#) [al-](#)  
[method](#)), [278](#) [lensdk.internal.brain\\_observatory.ophys\\_session\\_decomposition](#)  
[read\\_lims\\_file\(\)](#) (al- [298](#)  
[lensdk.internal.api.queries.optimize\\_config\\_reader.OptimizeConfigReader](#) (al-  
[method](#)), [280](#) [lensdk.config.model.description\\_parser.DescriptionParser](#)  
[read\\_lims\\_message\(\)](#) (al- [method](#)), [222](#)  
[lensdk.internal.api.queries.biophysical\\_module\\_reader.BiophysicalModuleReader](#) (al-  
[method](#)), [278](#) [lensdk.config.model.formats.json\\_description\\_parser.JsonDescrip](#)  
[read\\_lims\\_message\(\)](#) (al- [method](#)), [219](#)  
[lensdk.internal.api.queries.optimize\\_config\\_reader.OptimizeConfigReader](#) (al-  
[method](#)), [280](#) [lensdk.config.model.formats.pycfg\\_description\\_parser.PycfgDesc](#)  
[read\\_marker\\_file\(\)](#) (in module [al-lensdk.core.swc](#)), [264](#) [read\\_string\(\)](#) (al-  
[lensdk.internal.core.swc](#)), [305](#) [lensdk.core.json\\_utilities.JsonComments](#)  
[read\\_metrics\\_csv\(\)](#) (in module [al-lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache](#)), [lensdk.core.json\\_utilities.JsonComments](#)  
[162](#) [305](#) [class method](#)), [239](#)  
[read\\_model\\_description\(\)](#) (al- [read\\_url\(\)](#) (in module [allensdk.core.json\\_utilities](#)),  
[lensdk.model.biophys\\_sim.config.Config](#) [240](#)  
[method](#)), [353](#) [read\\_url\\_get\(\)](#) (in module [al-](#)  
[read\\_movie\(\)](#) (in module [al-lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache](#)), [lensdk.core.json\\_utilities](#), [240](#)  
[162](#) [read\\_url\\_post\(\)](#) (in module [al-](#)  
[lensdk.core.json\\_utilities](#)), [240](#)  
[read\\_ndarray\\_with\\_sitk\(\)](#) (in module [al-lensdk.core.sitk\\_utilities](#)), [257](#) [readOutput\(\)](#) ([allensdk.internal.model.biophysical.passive\\_fitting.outp](#)  
[lensdk.core.sitk\\_utilities](#)), [257](#) [method](#)), [312](#)  
[read\\_neuron\\_fit\\_stdout\(\)](#) (in module [al-lensdk.internal.model.biophysical.passive\\_fitting.neuron\\_utils](#)), [lensdk.brain\\_observatory.locally\\_sparse\\_noise.LocallySparseNoi](#)  
[311](#) [attribute](#)), [193](#)  
[read\\_nwb\(\)](#) (in module [al-lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_cache](#)), [lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.receptive\\_fie](#)  
[162](#) [attribute](#)), [139](#)  
[read\\_obj\(\)](#) (in module [allensdk.core.obj\\_utilities](#)), [ReceptiveFieldMapping](#) (class in [al-](#)  
[248](#) [lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.receptive\\_fie](#)

138  
RECONSTRUCTION\_KEY (al-  
lensdk.core.cell\_types\_cache.CellTypesCache  
attribute), 236

record\_cb() (allensdk.internal.mouse\_connectivity.interval\_unionize.interval\_unionize\_interval(class method), 337  
record\_cb() (allensdk.internal.mouse\_connectivity.interval\_unionize.interval\_unionize\_interval(class method), 339  
record\_values() (al-  
lensdk.internal.model.biophysical.deap\_utils.Utils  
method), 313  
record\_values() (al-  
lensdk.model.biophysical.utils.Utils  
method), 355

REFERENCE\_SPACE\_VERSION\_KEY (al-  
lensdk.core.reference\_space\_cache.ReferenceSpaceCache  
attribute), 252

ReferenceSpace (class in al-  
lensdk.core.reference\_space), 249  
ReferenceSpaceApi (class in al-  
lensdk.api.queries.reference\_space\_api),  
63  
ReferenceSpaceCache (class in al-  
lensdk.core.reference\_space\_cache), 251  
refine\_threshold\_indexes() (in module al-  
lensdk.ephys.ephys\_features), 275  
reliability() (in module al-  
lensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis),  
147  
remove\_comments() (al-  
lensdk.core.json\_utilities.JsonComments  
class method), 239  
remove\_keys() (allensdk.api.cache.Cache static  
method), 78  
remove\_lfp\_noise() (in module al-  
lensdk.brain\_observatory.ecephys.lfp\_subsampling.subsampling),  
130  
remove\_lfp\_offset() (in module al-  
lensdk.brain\_observatory.ecephys.lfp\_subsampling.subsampling),  
130  
remove\_multiline\_comments() (al-  
lensdk.core.json\_utilities.JsonComments  
class method), 239  
remove\_unassigned() (al-  
lensdk.core.reference\_space.ReferenceSpace  
method), 251  
removed\_unused\_stimulus\_presentation\_columns() (in module al-  
lensdk.brain\_observatory.ecephys.ecephys\_session),  
168  
rename\_columns() (allensdk.api.cache.Cache static  
method), 78  
renames() (allensdk.core.structure\_tree.StructureTree  
static method), 259

replace\_bad\_structure\_assignments()  
(in module al-  
lensdk.brain\_observatory.ecephys.ecephys\_project\_api.warehouse),  
116  
ReporterStatus (class in al-  
lensdk.core.cell\_types\_cache), 238  
required\_intensities (al-  
lensdk.mouse\_connectivity.grid.subimage.base\_subimage.Intensity  
attribute), 366  
required\_intensities (al-  
lensdk.mouse\_connectivity.grid.subimage.classic\_subimage.ClassicSubimage  
attribute), 369  
required\_polys (al-  
lensdk.mouse\_connectivity.grid.subimage.base\_subimage.Polygon  
attribute), 366  
required\_polys (al-  
lensdk.mouse\_connectivity.grid.subimage.cav\_subimage.CavSubimage  
attribute), 368  
required\_polys (al-  
lensdk.mouse\_connectivity.grid.subimage.classic\_subimage.ClassicSubimage  
attribute), 369  
required\_polys (al-  
lensdk.mouse\_connectivity.grid.subimage.count\_subimage.CountSubimage  
attribute), 369  
required\_segmentations (al-  
lensdk.mouse\_connectivity.grid.subimage.base\_subimage.Segmentations  
attribute), 367  
required\_segmentations (al-  
lensdk.mouse\_connectivity.grid.subimage.classic\_subimage.ClassicSubimage  
attribute), 369  
required\_segmentations (al-  
lensdk.mouse\_connectivity.grid.subimage.count\_subimage.CountSubimage  
attribute), 369  
resample\_into\_volume() (in module al-  
lensdk.mouse\_connectivity.grid.utilities.image\_utilities),  
371  
resample\_volume() (al-  
lensdk.mouse\_connectivity.grid.image\_series\_gridder.ImageSeriesGridder  
method), 373  
resample\_volume() (in module al-  
lensdk.mouse\_connectivity.grid.utilities.image\_utilities),  
371  
resave\_swc() (in module al-  
lensdk.internal.morphology.validate\_swc),  
335  
reset() (allensdk.model.glif.glif\_neuron.GlifNeuron  
method), 358  
reset\_ACurrent\_none() (in module al-  
lensdk.model.glif.glif\_neuron\_methods),  
362  
reset\_ACurrent\_sum() (in module al-

[lensdk.model.glif.glif\\_neuron\\_methods\(\)](#), 362  
[reset\\_hex\\_pack\(\)](#) (in module [allensdk.brain\\_observatory.circle\\_plots](#)), 188  
[reset\\_long\\_squares\\_start\(\)](#) (in module [allensdk.ephys.ephys\\_extractor](#)), 268  
[reset\\_threshold\\_inf\(\)](#) (in module [allensdk.model.glif.glif\\_neuron\\_methods](#)), 362  
[reset\\_threshold\\_three\\_components\(\)](#) (in module [allensdk.model.glif.glif\\_neuron\\_methods](#)), 362  
[reset\\_voltage\\_v\\_before\(\)](#) (in module [allensdk.model.glif.glif\\_neuron\\_methods](#)), 363  
[reset\\_voltage\\_zero\(\)](#) (in module [allensdk.model.glif.glif\\_neuron\\_methods](#)), 363  
[reshape\\_response\\_array\(\)](#) ([allensdk.brain\\_observatory.drifting\\_gratings.DriftingGratings](#) method), 192  
[reshape\\_response\\_array\(\)](#) ([allensdk.brain\\_observatory.natural\\_scenes.NaturalScenes](#) method), 195  
[reshape\\_response\\_array\(\)](#) ([allensdk.brain\\_observatory.static\\_gratings.StaticGratings](#) method), 206  
[resolve\\_initial\\_image\(\)](#) (in module [allensdk.brain\\_observatory.behavior.trials\\_processing](#)), 111  
[resolve\\_paths\(\)](#) ([allensdk.config.manifest.Manifest](#) method), 224  
[response\(\)](#) ([allensdk.brain\\_observatory.stimulus\\_analysis.StimulusAnalysis](#) attribute), 208  
[response\\_bias\(\)](#) (in module [allensdk.brain\\_observatory.behavior.session\\_metrics](#)), 107  
[retinotopy\\_metric\(\)](#) (in module [allensdk.internal.brain\\_observatory.annotated\\_region\\_metrics](#)), 289  
[retrieve\\_file\\_from\\_storage\(\)](#) ([allensdk.internal.api.api\\_prerelease.ApiPrerelease](#) method), 282  
[retrieve\\_file\\_over\\_http\(\)](#) ([allensdk.api.api.Api](#) method), 74  
[retrieve\\_parsed\\_json\\_over\\_http\(\)](#) ([allensdk.api.api.Api](#) method), 75  
[retrieve\\_xml\\_over\\_http\(\)](#) ([allensdk.api.api.Api](#) method), 75  
[return\\_mask\\_cb\(\)](#) ([allensdk.core.reference\\_space.ReferenceSpace](#) static method), 251  
[reward\\_rate\(\)](#) (in module [allensdk.brain\\_observatory.behavior.trial\\_masks](#)), 108  
[rewards\(\)](#) ([allensdk.brain\\_observatory.behavior.behavior\\_data\\_session.BehaviorDataSession](#) attribute), 90  
[rewards\(\)](#) ([allensdk.brain\\_observatory.behavior.behavior\\_ophys\\_session.BehaviorOphysSession](#) attribute), 94  
[rf\\_on\\_screen\(\)](#) (in module [allensdk.brain\\_observatory.ecephys.stimulus\\_analysis.receptive\\_field\\_analysis.ReceptiveFieldAnalysis](#)), 141  
[rig\\_equipment\\_name](#) ([allensdk.brain\\_observatory.ecephys.ecephys\\_session.EcephysSession](#) attribute), 167  
[rig\\_geometry\\_data](#) ([allensdk.brain\\_observatory.ecephys.ecephys\\_session.EcephysSession](#) attribute), 167  
[rings\\_in\\_hex\\_pack\(\)](#) (in module [allensdk.brain\\_observatory.circle\\_plots](#)), 188  
[rma\\_macros\(\)](#) (in module [allensdk.brain\\_observatory.ecephys.ecephys\\_project\\_api.utilities](#)), 125  
[rma\\_templates](#) ([allensdk.api.queries.biophysical\\_api.BiophysicalApi](#) attribute), 46  
[rma\\_templates](#) ([allensdk.api.queries.brain\\_observatory\\_api.BrainObservatoryApi](#) attribute), 49  
[rma\\_templates](#) ([allensdk.api.queries.glif\\_api.GlifApi](#) attribute), 52  
[rma\\_templates](#) ([allensdk.api.queries.image\\_download\\_api.ImageDownloadApi](#) attribute), 57  
[rma\\_templates](#) ([allensdk.api.queries.ontologies\\_api.OntologiesApi](#) attribute), 63  
[rma\\_templates](#) ([allensdk.internal.api.queries.biophysical\\_module\\_api.BiophysicalModuleApi](#) attribute), 277  
[RmaApi](#) (class in [allensdk.api.queries.rma\\_api](#)), 65  
[RmaEngine](#) (class in [allensdk.brain\\_observatory.ecephys.ecephys\\_project\\_api.rma\\_engine](#)), 124  
[RmaPager](#) (class in [allensdk.api.queries.rma\\_pager](#)), 70  
[RmaRequestError](#), 124  
[RmaTemplate](#) (class in [allensdk.api.queries.rma\\_template](#)), 70  
[robust\\_std\(\)](#) (in module [allensdk.brain\\_observatory.dff](#)), 191  
[roi\\_id](#) ([allensdk.brain\\_observatory.stimulus\\_analysis.StimulusAnalysis](#) attribute), 208  
[RoiMask](#) (class in [allensdk.brain\\_observatory.roi\\_masks](#)), 199  
[rolling\\_window\(\)](#) (in module [allensdk.brain\\_observatory.demixer](#)), 189  
[rolling\\_window\(\)](#) (in module [allensdk.brain\\_observatory.dff](#)), 191

[illegible]

[RunSimulateLims](#) (class in `allensdk.model.biophysical.runner`), 354  
[lensdk.internal.model.biophysical.run\\_simulate\\_lims](#), 314  
[save\\_nwb\(\)](#) (in module `lensdk.model.biophysical.runner`), 354  
[save\\_ophys\\_experiment\\_analysis\\_data\(\)](#) (in module `allensdk.api.queries.brain_observatory_api.BrainObservatoryApi`), 49  
[save\\_ophys\\_experiment\\_data\(\)](#) (in module `allensdk.api.queries.brain_observatory_api.BrainObservatoryApi`), 49  
[safe\\_factory\(\)](#) (in module `lensdk.internal.mouse_connectivity.projection_thumbnail.volume_projector.VolumeProjector`), 49  
[safe\\_make\\_parent\\_dirs\(\)](#) (in module `lensdk.config.manifest.Manifest` class method), 224  
[safe\\_mkdir\(\)](#) (in module `allensdk.config.manifest.Manifest` class method), 224  
[safe\\_system\\_path\(\)](#) (in module `lensdk.internal.core.lims_utilities`), 302  
[SAG\\_TARGET](#) (in module `allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor` attribute), 264  
[sameconv\(\)](#) (in module `lensdk.internal.model.GLM`), 326  
[sample\\_freq](#) (in module `allensdk.brain_observatory.sync_dataset.BrainSyncDataset` attribute), 216  
[sample\\_frequency](#) (in module `lensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset.EcephysSyncDataset` attribute), 129  
[sampling\\_rate](#) (in module `lensdk.brain_observatory.ecephys.nwb.EcephysProbe` attribute), 132  
[sampling\\_rate\\_scale](#) (in module `lensdk.brain_observatory.ecephys.align_timestamps.probe_synchronizer.ProbeSynchronizer` attribute), 116  
[save\(\)](#) (in module `allensdk.brain_observatory.behavior.behavior_ophys_api.BehaviorOphysNwbApi` method), 81  
[save\(\)](#) (in module `allensdk.core.swc.Morphology` method), 263  
[save\(\)](#) (in module `allensdk.internal.morphology.morphology.Morphology` method), 331  
[save\\_analysis\\_arrays\(\)](#) (in module `lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet` method), 235  
[save\\_analysis\\_dataframes\(\)](#) (in module `lensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet` method), 235  
[save\\_cell\\_index\\_receptive\\_field\\_analysis\(\)](#) (in module `allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` static method), 193  
[save\\_ephys\\_data\(\)](#) (in module `lensdk.api.queries.cell_types_api.CellTypesApi` method), 50  
[save\\_figure\(\)](#) (in module `lensdk.internal.ephys.plot_qc_figures`), 308  
[save\\_figure\(\)](#) (in module `lensdk.internal.ephys.plot_qc_figures3`), 310  
[save\\_qc\\_figures\(\)](#) (in module `lensdk.internal.ephys.core_feature_extract`), 206  
[save\\_reconstruction\(\)](#) (in module `lensdk.api.queries.cell_types_api.CellTypesApi` method), 50  
[save\\_reconstruction\\_markers\(\)](#) (in module `lensdk.api.queries.cell_types_api.CellTypesApi` method), 51  
[save\\_session\\_b\(\)](#) (in module `lensdk.brain_observatory.session_analysis.SessionAnalysis` method), 202  
[save\\_session\\_c2\(\)](#) (in module `lensdk.brain_observatory.session_analysis.SessionAnalysis` method), 203  
[save\\_voltage\(\)](#) (in module `lensdk.core.dat_utilities.DatUtilities` class method), 238  
[scene\\_re](#) (in module `allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_queries.connected_services.ConnectedServices` class method), 51  
[search\(\)](#) (in module `allensdk.brain_observatory.stimulus_info.BinaryIntervalSearch` method), 208  
[search\(\)](#) (in module `allensdk.brain_observatory.stimulus_info.StimulusSearch` method), 210  
[section\\_image\\_query\(\)](#) (in module `lensdk.api.queries.image_download_api.ImageDownloadApi` method), 57  
[segmentation\\_mask\\_image](#) (in module `lensdk.brain_observatory.behavior.behavior_ophys_session.BehaviorOphysSession` attribute), 132



attribute), 94  
 SegmentationSubImage (class in al-  
   lensdk.mouse\_connectivity.grid.subimage.base\_subimage), 366  
 select() (allensdk.internal.api.PostgresQueryMixin  
   method), 287  
 select() (in module al-  
   lensdk.internal.core.lims\_utilities), 302  
 select\_channels() (in module al-  
   lensdk.brain\_observatory.ecephys.lfp\_subsampling.subsampling  
   130  
 select\_one() (allensdk.internal.api.PostgresQueryMixin  
   method), 287  
 serialize() (allensdk.brain\_observatory.behavior.image\_api\_image\_api  
   static method), 104  
 SERVICE (allensdk.api.queries.rma\_api.RmaApi at-  
   tribute), 65  
 service\_query() (al-  
   lensdk.api.queries.rma\_api.RmaApi method),  
   69  
 service\_stage() (al-  
   lensdk.api.queries.rma\_api.RmaApi method),  
   69  
 session\_a() (allensdk.brain\_observatory.session\_analysis.SessionAnalysis  
   method), 203  
 SESSION\_ANALYSIS\_METRICS\_KEY (al-  
   lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache  
   attribute), 158  
 session\_b() (allensdk.brain\_observatory.session\_analysis.SessionAnalysis  
   method), 203  
 session\_c() (allensdk.brain\_observatory.session\_analysis.SessionAnalysis  
   method), 204  
 session\_c2() (allensdk.brain\_observatory.session\_analysis.SessionAnalysis  
   method), 204  
 SESSION\_DIR\_KEY (al-  
   lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache  
   attribute), 159  
 session\_na (allensdk.brain\_observatory.ecephys.ecephys\_session\_api.EcephysSessionApi  
   attribute), 128  
 SESSION\_NWB\_KEY (al-  
   lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache  
   attribute), 159  
 session\_type (allensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession  
   attribute), 167  
 SessionAnalysis (class in al-  
   lensdk.brain\_observatory.session\_analysis),  
   202  
 SESSIONS\_KEY (allensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache  
   attribute), 158  
 sessions\_with\_stimulus() (in module al-  
   lensdk.brain\_observatory.stimulus\_info), 211  
 set\_actual\_parameters() (al-  
   lensdk.internal.model.biophysical.deap\_utils.Utils  
   method), 313  
 set\_api\_urls() (allensdk.api.api.Api method), 75  
 set\_apical\_color() (al-  
   lensdk.internal.morphology.morphvis.MorphologyColors  
   method), 332  
 set\_axon\_color() (al-  
   lensdk.internal.morphology.morphvis.MorphologyColors  
   method), 332  
 set\_basal\_color() (al-  
   lensdk.internal.morphology.morphvis.MorphologyColors  
   method), 332  
 set\_coarse\_grid\_parameters() (al-  
   lensdk.mouse\_connectivity.grid.image\_series\_gridder.ImageSeriesGridder  
   method), 373  
 set\_image\_api\_provider() (in module al-  
   lensdk.core.authentication), 226  
 set\_default\_working\_directory() (al-  
   lensdk.api.api.Api method), 75  
 set\_dims() (allensdk.brain\_observatory.circle\_plots.CoronaPlotter  
   method), 187  
 set\_dims() (allensdk.brain\_observatory.circle\_plots.FanPlotter  
   method), 187  
 set\_F() (allensdk.brain\_observatory.r\_neuropil.NeuropilSubtract  
   method), 197  
 set\_session\_parameters() (al-  
   lensdk.internal.model.biophysical.deap\_utils.Utils  
   method), 313  
 set\_session\_cache (in module al-  
   lensdk.mouse\_connectivity.grid.utilities.image\_utilities),  
   214  
 set\_max\_voxel() (al-  
   lensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_utils  
   method), 338  
 set\_session\_parameters() (al-  
   lensdk.internal.model.glif.glif\_experiment.GlifExperiment  
   method), 318  
 set\_session\_parameters() (al-  
   lensdk.internal.model.biophysical.deap\_utils.Utils  
   method), 313  
 set\_session\_api (allensdk.brain\_observatory.session\_api.EcephysSessionApi  
   method), 205  
 set\_session\_information() (in module al-  
   lensdk.core.sitk\_utilities), 257  
 set\_session\_type (allensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession  
   method), 332  
 set\_spatial\_unit() (al-  
   lensdk.brain\_observatory.stimulus\_info.Monitor  
   method), 189  
 set\_spike\_times() (al-  
   lensdk.core.nwb\_data\_set.NwbDataSet  
   method), 247  
 set\_stimulus\_amplitude\_calculator() (al-  
   lensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor  
   method), 266

`set_sweep()` (`allensdk.core.nwb_data_set.NwbDataSet` method), 187  
`set_version()` (`allensdk.config.manifest_builder.ManifestBuilder` method), 225  
`set_workflow_state()` (`allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader` method), 278  
`setup_iclamp()` (`allensdk.model.biophysical.utils.Utils` method), 355  
`setup_images()` (`allensdk.mouse_connectivity.grid.subimage.base_subimage.InteractivitySubImage` method), 366  
`setup_images()` (`allensdk.mouse_connectivity.grid.subimage.base_subimage.PolyhedronSubImage` method), 366  
`setup_images()` (`allensdk.mouse_connectivity.grid.subimage.base_subimage.SegmentationSubImage` method), 367  
`setup_images()` (`allensdk.mouse_connectivity.grid.subimage.base_subimage.SubImage` method), 368  
`setup_interval_map()` (`allensdk.internal.mouse_connectivity.interval_unionize.interval_unionize.IntervalUnionizer` method), 337  
`setup_model()` (`allensdk.internal.model.biophysical.make_deap_fit_json.Reporter` method), 313  
`setup_subimages()` (`allensdk.mouse_connectivity.grid.image_series_gridder.ImageSeriesGridder` method), 373  
`setup_table_for_epochs()` (in module `allensdk.brain_observatory.nwb`), 174  
`setup_table_for_invalid_times()` (in module `allensdk.brain_observatory.nwb`), 174  
`sex` (`allensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` attribute), 167  
`sex` (`allensdk.brain_observatory.ecephys.nwb.EcephysLabMetaData` attribute), 132  
`sfvals` (`allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings` attribute), 143  
`sfvals` (`allensdk.brain_observatory.static_gratings.StaticGratings` attribute), 206  
`short_squares_features()` (`allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor` method), 265  
`short_string()` (`allensdk.internal.morphology.node.Node` method), 334  
`show_angle_labels()` (`allensdk.brain_observatory.circle_plots.FanPlotter` method), 187  
`show_arrow()` (`allensdk.brain_observatory.circle_plots.CoronaPlotter` method), 187  
`show_arrow()` (`allensdk.brain_observatory.circle_plots.TrackPlotter` method), 187  
`show_axes()` (`allensdk.brain_observatory.circle_plots.FanPlotter` method), 187  
`show_circle()` (`allensdk.brain_observatory.circle_plots.CoronaPlotter` method), 187  
`show_group_labels()` (`allensdk.brain_observatory.circle_plots.FanPlotter` method), 187  
`show_image()` (`allensdk.brain_observatory.stimulus_info.Monitor` method), 187  
`show_r_labels()` (`allensdk.brain_observatory.circle_plots.FanPlotter` method), 187  
`simple_rotation()` (in module `allensdk.internal.mouse_connectivity.projection_thumbnail.generator.SimpleTree` (class in `allensdk.core.simple_tree`), 253  
`SimpleTree` (class in `allensdk.internal.core.simpletree`), 304  
`simplify_cells_api()` (`allensdk.api.queries.cell_types_api.CellTypesApi` method), 49  
`simplify_experiment_containers()` (`allensdk.api.queries.brain_observatory_api.BrainObservatoryApi` method), 49  
`simplify_ophys_experiments()` (`allensdk.api.queries.brain_observatory_api.BrainObservatoryApi` method), 49  
`simulate()` (`allensdk.model.biophysical.run_simulate.RunSimulate` method), 353  
`simulate_neuron()` (in module `allensdk.model.glif.simulate_neuron`), 365  
`simulate_sweep()` (in module `allensdk.model.glif.simulate_neuron`), 365  
`simulate_sweep_from_file()` (in module `allensdk.model.glif.simulate_neuron`), 365  
`sitk_get_center()` (in module `allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings`), 342  
`sitk_get_diagonal_length()` (in module `allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings`), 342  
`sitk_get_image_parameters()` (in module `allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings`), 342  
`sitk_get_size_parity()` (in module `allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings.StaticGratings`), 342  
`sitk_np_convert()` (in module `allensdk.mouse_connectivity.grid.utilities.image_utilities`), 342

`sitk_paste_into_center()` (in module `al-` `lensdk.model.glif.glif_neuron_methods`),  
`lensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities`),  
342 `spike_component_of_threshold_forward_euler()`  
`sitk_safe_ln()` (in module `al-` (in module `al-`  
`lensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities`),  
341 `lensdk.model.glif.glif_neuron_methods`),  
363  
`size()` (`allensdk.internal.brain_observatory.mask_set.MaskSet` feature()) (al-  
method), 297 `lensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor`  
`slice_arrays()` (al- method), 266  
`lensdk.internal.mouse_connectivity.interval_unionsize_interval_unionize_arrays()` (al-  
method), 339 `lensdk.ephys.ephys_extractor.EphysSweepSetFeatureExtractor`  
`smooth()` (in module `al-` method), 267  
`lensdk.brain_observatory.receptive_field_analysis_utilities` feature\_keys() (al-  
181 `lensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor`  
`smooth_STA()` (in module `al-` method), 266  
`lensdk.brain_observatory.receptive_field_analysis_utilities` spikes (allensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession  
177 attribute), 167  
`sobel_grad()` (in module `al-` `SPIKE_TIMES` (allensdk.core.nwb\_data\_set.NwbDataSet  
`lensdk.internal.brain_observatory.itracker_utils`), attribute), 246  
296 `spikes` (allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis\_stimulus\_table.StimulusAnalysisStimulusTable  
attribute), 145  
`SOMA` (allensdk.core.swc.Morphology attribute), 260 `spikes()` (allensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor  
method), 266  
`soma` (allensdk.core.swc.Morphology attribute), 263 `spiral_trials()` (in module `al-`  
`SOMA` (allensdk.internal.morphology.morphology.Morphology attribute), 328 `lensdk.brain_observatory.circle_plots`), 188  
`soma_root()` (allensdk.internal.morphology.morphology.Morphology method), 331 `spiral_trials_polar()` (in module `al-`  
`lensdk.brain_observatory.circle_plots`), 188  
`sort_data_arrays()` (al- `size_interval_unionize_arrays()` (in module `al-`  
`lensdk.internal.mouse_connectivity.interval_unionsize_interval_unionize_arrays` method), 337 `lensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes`  
`sort_trials()` (al- 152  
`lensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise` (class in `al-`  
method), 193 `lensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api`  
`spacing` (allensdk.brain\_observatory.behavior.image\_api.Image attribute), 104 `standardize_movie_numbers()` (in module `al-`  
`lensdk.brain_observatory.ecephys.stimulus_table.naming_utilities`), 153  
`SPACING` (allensdk.core.swc.Marker attribute), 260 `start()` (allensdk.internal.model.biophysical.passive\_fitting.output\_grabber.OutputGrabber  
method), 312  
`SPACING` (allensdk.internal.core.swc.Marker attribute), 305 `START_ROW` (allensdk.api.queries.rma\_api.RmaApi attribute), 65  
`sparsify()` (allensdk.core.swc.Morphology method), 263 `start_specimen()` (al-  
`sparsify()` (allensdk.internal.morphology.morphology.Morphology method), 331 `lensdk.internal.model.biophysical.run_optimize.RunOptimize`  
`spatial_frequency_to_pix_per_cycle()` (allensdk.brain\_observatory.stimulus\_info.Monitor method), 209  
`specimen_name` (al- `StaticGratings` (class in `al-`  
`lensdk.brain_observatory.ecephys.ecephys_session.EcephysSession` attribute), 167 `StaticGratings` (class in `al-`  
`lensdk.brain_observatory.static_gratings`), 205  
`specimen_name` (al- `stats()` (allensdk.brain\_observatory.sync\_dataset.Dataset  
`lensdk.brain_observatory.ecephys.nwb.EcephysLabMetaDataSet` attribute), 132  
`speeds` (allensdk.brain\_observatory.ecephys.stimulus\_analysis.dot\_motion\_dot\_motion attribute), 133  
`stim_table` (allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis\_stimulus\_table.StimulusAnalysisStimulusTable  
attribute), 145  
`spike_component_of_threshold_exact()` (in module `al-` `stim_table` (allensdk.brain\_observatory.stimulus\_analysis.StimulusAnalysisStimulusTable  
attribute), 145



[attribute](#)), 208  
[stim\\_table\\_contrast](#) (al- [lensdk.brain\\_observatory.ecephys.nwb.EcephysLabMetaData](#)  
[lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.drifting\\_gratings.DriftingGratings](#)  
[attribute](#)), 135  
[stim\\_table\\_spontaneous](#) (al- [lensdk.brain\\_observatory.ecephys.ecephys\\_session.EcephysSession](#)  
[lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.stimulus\\_table.StimulusAnalysis](#)  
[attribute](#)), 145  
[stim\\_table\\_to\\_categories\(\)](#) (in module al- [lensdk.internal.api.queries.biophysical\\_module\\_reader.Biophysical](#)  
[lensdk.brain\\_observatory.chisquare\\_categorical](#)), [method](#)), 278  
186  
[stim\\_timestamps](#) (al- [lensdk.internal.api.queries.optimize\\_config\\_reader.OptimizeConfig](#)  
[lensdk.internal.brain\\_observatory.time\\_sync.OphysTimeAlignment](#)  
[attribute](#)), 300  
[stimuli](#) ([allensdk.brain\\_observatory.ecephys.file\\_io.stim\\_file.CamStimulusFile](#)  
[attribute](#)), 129  
[stimuli\\_in\\_session\(\)](#) (in module al- [stimulus\\_presentations](#) (al-  
[lensdk.brain\\_observatory.stimulus\\_info](#)), [lensdk.brain\\_observatory.behavior.behavior\\_data\\_session.Behavior](#)  
[211](#) [attribute](#)), 94  
[stimulus\\_alignment](#) (al- [stimulus\\_presentations](#) (al-  
[lensdk.internal.pipeline\\_modules.run\\_ophys\\_time\\_sync.TimeSyncOutput](#)  
[attribute](#)), 349 [attribute](#)), 168  
[stimulus\\_amplitude\(\)](#) (al- [stimulus\\_search](#) (al-  
[lensdk.ephys.ephys\\_extractor.EphysSweepFeatureExtractor](#) [lensdk.core.brain\\_observatory\\_nwb\\_data\\_set.BrainObservatoryNWB](#)  
[method](#)), 266 [attribute](#)), 235  
[stimulus\\_conditions](#) (al- [STIMULUS\\_TABLE\\_TYPES](#) (al-  
[lensdk.brain\\_observatory.ecephys.ecephys\\_session.EcephysSession](#)  
[attribute](#)), 167 [lensdk.core.brain\\_observatory\\_nwb\\_data\\_set.BrainObservatoryNWB](#)  
[attribute](#)), 231  
[stimulus\\_conditions](#) (al- [STIMULUS\\_TEMPLATE\\_NAMESPACE](#) (al-  
[lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.stimulus\\_templates.StimulusTemplates](#)  
[attribute](#)), 145 [lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_api.ecephys\\_project\\_api](#)  
[attribute](#)), 118  
[stimulus\\_conditions\\_contrast](#) (al- [stimulus\\_templates](#) (al-  
[lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.drifting\\_gratings.DriftingGratings](#)  
[attribute](#)), 135 [attribute](#)), 90  
[STIMULUS\\_CONTENT\\_TYPE](#) (al- [stimulus\\_templates](#) (al-  
[lensdk.internal.api.queries.biophysical\\_module\\_reader.BiophysicalModuleReader](#)  
[attribute](#)), 277 [lensdk.brain\\_observatory.behavior.behavior\\_ophys\\_session.BehaviorOphysSession](#)  
[attribute](#)), 94  
[STIMULUS\\_CONTENT\\_TYPE](#) (al- [stimulus\\_templates](#) (al-  
[lensdk.internal.api.queries.optimize\\_config\\_reader.OptimizeConfigReader](#)  
[attribute](#)), 280 [lensdk.brain\\_observatory.ecephys.ecephys\\_project\\_api.ecephys\\_project\\_api](#)  
[attribute](#)), 121  
[stimulus\\_delay](#) (al- [stimulus\\_times](#) (al-  
[lensdk.internal.pipeline\\_modules.run\\_ophys\\_time\\_sync.TimeSyncOutput](#)  
[attribute](#)), 349 [attribute](#)), 350  
[stimulus\\_delta](#) (al- [stimulus\\_timestamps](#) (al-  
[lensdk.internal.pipeline\\_modules.run\\_ophys\\_time\\_sync.TimeSyncOutput](#)  
[attribute](#)), 350 [attribute](#)), 90  
[stimulus\\_file\\_entries\(\)](#) (al- [stimulus\\_timestamps](#) (al-  
[lensdk.internal.api.queries.biophysical\\_module\\_reader.BiophysicalModuleReader](#)  
[method](#)), 278 [lensdk.brain\\_observatory.behavior.behavior\\_ophys\\_session.BehaviorOphysSession](#)  
[attribute](#)), 94  
[stimulus\\_file\\_entries\(\)](#) (al- [StimulusAnalysis](#) (class in al-  
[lensdk.internal.api.queries.optimize\\_config\\_reader.OptimizeConfigReader](#)  
[method](#)), 280 [lensdk.brain\\_observatory.ecephys.stimulus\\_analysis.stimulus\\_analysis](#)  
[143](#)  
[STIMULUS\\_MAPPINGS\\_KEY](#) (al- [StimulusAnalysis](#) (class in al-  
[lensdk.core.brain\\_observatory\\_cache.BrainObservatoryCache](#)  
[attribute](#)), 227 [lensdk.brain\\_observatory.stimulus\\_analysis](#)),  
[207](#)

StimulusSearch (class in al-  
lensdk.brain\_observatory.stimulus\_info),  
209  
STRUCTURE\_UNIONIZES\_KEY (al-  
lensdk.brain\_observatory.stimulus\_info),  
209  
stitch() (allensdk.internal.mouse\_connectivity.tissuecyte\_stitching.  
method), 342  
Stitcher (class in al-  
lensdk.internal.mouse\_connectivity.tissuecyte\_stitching.stitcher),  
342  
stop() (allensdk.internal.model.biophysical.passive\_fitting.output\_grabber.  
method), 312  
stop() (allensdk.internal.model.biophysical.passive\_fitting.output\_grabber.  
method), 312  
STORAGE\_DIRECTORIES\_PRERELEASE\_KEY (al-  
lensdk.internal.core.mouse\_connectivity\_cache\_prerelease.  
attribute), 303  
strain (allensdk.brain\_observatory.ecephys.nwb.EcephysSubMetaData.  
attribute), 132  
stream() (allensdk.brain\_observatory.ecephys.ecephys\_project\_api.http\_engine.  
method), 122  
stream() (allensdk.brain\_observatory.ecephys.ecephys\_project\_api.http\_engine.  
method), 123  
stream\_file\_over\_http() (in module al-  
lensdk.api.api), 75  
stream\_zip\_directory\_over\_http() (in mod-  
ule allensdk.api.api), 75  
STRING (allensdk.api.queries.connected\_services.ConnectedServices.  
attribute), 51  
strip\_all\_other\_types() (al-  
lensdk.core.swc.Morphology method), 263  
strip\_all\_other\_types() (al-  
lensdk.internal.morphology.morphology.Morphology  
method), 331  
strip\_type() (allensdk.core.swc.Morphology  
method), 263  
strip\_type() (allensdk.internal.morphology.morphology.Morphology  
method), 331  
structure\_acronyms (al-  
lensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession.  
attribute), 168  
structure\_assignment\_fname\_map() (in  
module allensdk.brain\_observatory.ecephys.ecephys\_project\_api.warehouse.  
attribute), 117  
structure\_descends\_from() (al-  
lensdk.core.ontology.Ontology method),  
249  
structure\_descends\_from() (al-  
lensdk.core.structure\_tree.StructureTree  
method), 259  
STRUCTURE\_MASK\_KEY (al-  
lensdk.core.reference\_space\_cache.ReferenceSpaceCache.  
attribute), 252  
STRUCTURE\_MESH\_KEY (al-  
lensdk.core.reference\_space\_cache.ReferenceSpaceCache.  
attribute), 252  
STRUCTURE\_TREE\_KEY (al-  
lensdk.core.reference\_space\_cache.ReferenceSpaceCache.  
attribute), 252  
STRUCTURE\_UNIONIZES\_KEY (al-  
lensdk.brain\_observatory.stimulus\_info),  
209  
STRUCTURES\_KEY (al-  
lensdk.core.reference\_space\_cache.ReferenceSpaceCache.  
attribute), 252  
subclass (class in al-  
lensdk.core.structure\_tree), 257  
subimage (class in al-  
lensdk.mouse\_connectivity.grid.subimage.base\_subimage),  
367  
subsample\_data() (in module al-  
lensdk.internal.model.data\_access), 327  
subsample\_lfp() (in module al-  
lensdk.brain\_observatory.ecephys.lfp\_subsampling.subsampling),  
131  
subsample\_timestamps() (in module al-  
lensdk.brain\_observatory.ecephys.lfp\_subsampling.subsampling),  
131  
SUBTHRESH\_MAX\_AMP (al-  
lensdk.ephys.ephys\_extractor.EphysCellFeatureExtractor  
attribute), 264  
sum\_projection\_pixel\_intensity (al-  
lensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_u  
nionize.tissuecyte\_u), 338  
sum\_projection\_pixels (al-  
lensdk.internal.mouse\_connectivity.interval\_unionize.tissuecyte\_u  
nionize.tissuecyte\_u), 338  
summarize() (allensdk.ephys.feature\_extractor.EphysFeatureExtractor  
method), 276  
SUMMARY\_STRUCTURE\_SET\_ID (al-  
lensdk.core.mouse\_connectivity\_cache.MouseConnectivityCache.  
attribute), 241  
superclass (class in al-  
lensdk.brain\_observatory.behavior.criteria),  
99  
SUPPORTED\_PIPELINE\_VERSION (al-  
lensdk.core.brain\_observatory\_nwb\_data\_set.BrainObservatoryN  
wbDataSet.  
attribute), 231

SupportsStr (class in allensdk.core.typing), 264  
SUPPRESS\_FROM\_CHANNELS (al- sweeps () (allensdk.ephys.ephys\_extractor.EphysSweepSetFeatureExtractor attribute), 206  
lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache  
attribute), 159 SynchronizationApi (class in al-  
SUPPRESS\_FROM\_PROBES (al- lensdk.api.queries.synchronization\_api),  
lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache  
attribute), 159 synthesize\_F () (in module al-  
SUPPRESS\_FROM\_SESSION\_TABLE (al- lensdk.brain\_observatory.r\_neuropil), 198  
lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache  
attribute), 159 T  
SUPPRESS\_FROM\_UNITS (al- TABULAR (allensdk.api.queries.rma\_api.RmaApi at-  
lensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache  
attribute), 159 tiling\_utilities (class in al-  
SvgApi (class in allensdk.api.queries.svg\_api), 70 task\_parameters (al-  
SWC\_FILE\_TYPE (al- attribute), 91  
lensdk.api.queries.cell\_types\_api.CellTypesApi task\_parameters (al-  
attribute), 49 lensdk.brain\_observatory.behavior.behavior\_data\_session.BehaviorDataSession  
sweep\_entries () (al- attribute), 94  
lensdk.internal.api.queries.biophysical\_module\_reader.BiophysicalModuleReader  
method), 278 tiling\_utilities (class in al-  
sweep\_entries () (al- temp\_dir () (in module al-  
lensdk.internal.api.queries.optimize\_config\_reader.OptimizeConfigReader  
method), 280 tiling\_utilities.temp\_dir), 373  
sweep\_feature () (al- TEMPLATE\_KEY (allensdk.core.reference\_space\_cache.ReferenceSpaceCache  
attribute), 252  
lensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor template\_projection () (in module al-  
method), 266 lensdk.internal.mouse\_connectivity.projection\_thumbnail.projection\_thumbnail  
sweep\_feature\_keys () (al- 340  
lensdk.ephys.ephys\_extractor.EphysSweepFeatureExtractor template\_query () (al-  
method), 267 lensdk.api.queries.rma\_template.RmaTemplate  
sweep\_features () (al- method), 70  
lensdk.ephys.ephys\_extractor.EphysSweepSetFeatureExtractor (allensdk.brain\_observatory.ecephys.ecephys\_session\_api.ecephys\_session\_api  
method), 267 method), 127  
sweep\_numbers () (al- test () (allensdk.brain\_observatory.ecephys.ecephys\_session\_api.ecephys\_session\_api  
lensdk.internal.api.queries.biophysical\_module\_reader.BiophysicalModuleReader  
method), 278 method), 127  
sweep\_numbers () (al- test\_fit () (in module al-  
lensdk.internal.api.queries.optimize\_config\_reader.OptimizeConfigReader  
method), 281 TestNode (class in al-  
sweep\_numbers\_by\_type () (al- lensdk.internal.morphology.validate\_swc),  
lensdk.internal.api.queries.biophysical\_module\_reader.BiophysicalModuleReader  
method), 278 tfvals (allensdk.brain\_observatory.drifting\_gratings.DriftingGratings  
sweep\_response (al- attribute), 192  
lensdk.brain\_observatory.natural\_movie.NaturalMovie tfvals (allensdk.brain\_observatory.ecephys.stimulus\_analysis.drifting\_gratings\_analysis  
attribute), 194 attribute), 135  
sweep\_response (al- threshold\_rf () (in module al-  
lensdk.brain\_observatory.stimulus\_analysis.StimulusAnalysis lensdk.brain\_observatory.ecephys.stimulus\_analysis.receptive\_field\_analysis  
attribute), 208 141  
sweeplength (allensdk.brain\_observatory.locally\_sparse\_noise.LocallySparseNoise  
attribute), 193 tiling\_utilities (class in allensdk.internal.mouse\_connectivity.tissuecyte\_stitching.tissuecyte\_stitching\_utilities  
attribute), 343  
sweeplength (allensdk.brain\_observatory.natural\_movie.NaturalMovie timestamps (allensdk.brain\_observatory.running\_speed.RunningSpeed  
attribute), 194 attribute), 201  
sweeplength (allensdk.brain\_observatory.natural\_scenes.NaturalScenes timestamps (allensdk.brain\_observatory.stimulus\_analysis.StimulusAnalysis  
attribute), 195 attribute), 208  
sweeplength (allensdk.brain\_observatory.static\_gratings.StaticGratings

`TimeSyncOutputs` (class in `al-` `lensdk.brain_observatory.circle_plots`), 187  
`lensdk.internal.pipeline_modules.run_ophys_time_syncingLabelClassifier` (class in `al-`  
349 `lensdk.internal.brain_observatory.roi_filter_utils`),  
`TimeSyncWriter` (class in `al-` 298  
350 `lensdk.internal.pipeline_modules.run_ophys_time_syncingMultiLabelClassifier` (class in `al-`  
`TissuecyteBaseUnionize` (class in `al-` 298  
337 `lensdk.internal.mouse_connectivity.interval_unionize_tissuecyte` (`allensdk.mouse`); `connectivity.grid.image_series_gridded.ImageSeriesGridding` attribute), 373  
`TissuecyteInjectionUnionize` (class in `al-` `translate_image_and_fill()` (in module `al-`  
338 `lensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_injection_unionize`; `lensdk.brain_observatory.stimulus_info`), 212  
`TissuecyteProjectionUnionize` (class in `al-` `traverse_h5_file()` (in module `al-`  
338 `lensdk.internal.mouse_connectivity.interval_unionize_tissuecyte_projection_unionize_recoMorphology` method), 263  
`TissuecyteUnionizer` (class in `al-` `tree()` (`allensdk.internal.morphology.morphology.Morphology`  
338 `lensdk.internal.mouse_connectivity.interval_unionize_tissuecyte_unionizer`), (class in `al-`  
`to_config_string()` (`al-` `trial_data_from_log()` (in module `al-`  
`lensdk.config.app.application_config.ApplicationConfig` `lensdk.brain_observatory.behavior.trials_processing`),  
method), 218 111  
`to_dict()` (`allensdk.brain_observatory.ecephys.nwb.EcephysRawMetaData`) (in module `al-`  
method), 132 `lensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis_stimulus_analysis` attribute), 145  
`to_dict()` (`allensdk.internal.model.glif.glif_optimizer.GlifOptimizer` method), 319  
`to_dict()` (`allensdk.internal.model.glif.glif_optimizer_neuron.GlifOptimizerNeuron` method), 320  
`to_dict()` (`allensdk.internal.morphology.morphology.MorphologyTypes`) (in module `al-`  
method), 331 `lensdk.brain_observatory.behavior.trial_masks`),  
`to_dict()` (`allensdk.internal.morphology.node.Node` method), 334  
`to_dict()` (`allensdk.model.glif.glif_neuron.GlifNeuron` method), 359  
`to_dict()` (`allensdk.model.glif.glif_neuron_methods.GlifNeuronMethods` method), 360  
`to_filter_rhs()` (`al-` `trim()` (`allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile`  
`lensdk.api.queries.rma_template.RmaTemplate` method), 70 343  
`to_manifest()` (`al-` `trim_border_pulses()` (in module `al-`  
`lensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader` method), 278 `lensdk.brain_observatory.ecephys.stimulus_sync`),  
`to_manifest()` (`al-` `trim_self()` (`allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile`  
`lensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader` method), 281 69  
`total_presentations` (`al-` `trimmed_stats()` (in module `al-`  
`lensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis.StimulusAnalysis` attribute), 145  
`total_voxel_counts()` (`al-` TRUE (`allensdk.api.queries.rma_api.RmaApi` attribute),  
`lensdk.core.reference_space.ReferenceSpace` method), 251 65  
`total_voxel_map` (`al-` `tuple_filters()` (`al-`  
`lensdk.core.reference_space.ReferenceSpace` attribute), 251 `lensdk.api.queries.rma_api.RmaApi` method),  
`TrackPlotter` (class in `al-` 69  
TYPE (`allensdk.internal.model.glif.glif_optimizer_neuron.GlifOptimizerNeuron`



[attribute](#)), 319  
 TYPE (allensdk.model.glif.glif\_neuron.GlifNeuron [attribute](#)), 357  
 TYPEWISE\_ANALYSIS\_METRICS\_KEY (allensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache [attribute](#)), 159  
**U**  
 union() (allensdk.internal.brain\_observatory.mask\_set.MaskSet [method](#)), 297  
 union\_size() (allensdk.internal.brain\_observatory.mask\_set.MaskSet [method](#)), 297  
 Unionize (class in allensdk.internal.mouse\_connectivity.interval\_unionize.unionize\_record), 339  
 unit (allensdk.brain\_observatory.behavior.image\_api.Image [attribute](#)), 104  
 unit\_count (allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis.StimulusAnalysis [attribute](#)), 145  
 unit\_ids (allensdk.brain\_observatory.ecephys.stimulus\_analysis.stimulus\_analysis.StimulusAnalysis [attribute](#)), 145  
 units (allensdk.brain\_observatory.ecephys.ecephys\_session.EcephysSession [attribute](#)), 168  
 UNITS\_KEY (allensdk.brain\_observatory.ecephys.ecephys\_project\_cache.EcephysProjectCache [attribute](#)), 159  
 unknown (allensdk.brain\_observatory.argschema\_utilities.RaisingSchemaError [attribute](#)), 185  
 unknown (allensdk.brain\_observatory.behavior.schemas.RaisingSchemaError [attribute](#)), 107  
 unpack() (allensdk.config.model.description.Description [method](#)), 221  
 unpack\_change\_log() (in module allensdk.brain\_observatory.behavior.stimulus\_processing), 108  
 unpack\_manifest() (allensdk.config.model.description.Description [method](#)), 221  
 unpack\_structure\_set\_ancestors() (allensdk.api.queries.ontologies\_api.OntologiesApi [method](#)), 63  
 unpack\_uint32() (in module allensdk.brain\_observatory.sync\_dataset), 216  
 update\_data() (allensdk.config.model.description.Description [method](#)), 221  
 update\_default\_cell\_hoc() (allensdk.model.biophysical.utils.Utils [method](#)), 355  
 update\_output\_sweep\_features() (in module allensdk.internal.ephys.core\_feature\_extract), 306  
 update\_well\_known\_file() (allensdk.internal.api.queries.biophysical\_module\_reader.BiophysicalModuleReader [method](#)), 278  
 update\_well\_known\_file() (allensdk.internal.api.queries.optimize\_config\_reader.OptimizeConfigReader [method](#)), 281  
 units\_degrees() (in module allensdk.brain\_observatory.receptive\_field\_analysis.utilities), 181  
 user (allensdk.core.authentication.DbCredentials [attribute](#)), 226  
 Utils (class in allensdk.internal.model.biophysical.deap\_utils), 353  
 Utils (class in allensdk.model.biophysical.utils), 354  
 validate\_epoch\_durations() (in module allensdk.brain\_observatory.ecephys.stimulus\_table.output\_validation), 154  
 validate\_epoch\_durations() (in module allensdk.brain\_observatory.ecephys.stimulus\_table.output\_validation), 154  
 validate\_last\_trial\_ends\_adjacent\_to\_flash() (in module allensdk.brain\_observatory.behavior.validation), 111  
 validate\_mask() (in module allensdk.brain\_observatory.roi\_masks), 201  
 validate\_max\_spontaneous\_epoch\_duration() (in module allensdk.brain\_observatory.ecephys.stimulus\_table.output\_validation), 154  
 validate\_ophys\_dff\_length() (in module allensdk.brain\_observatory.behavior.validation), 111  
 validate\_ophys\_timestamps() (in module allensdk.brain\_observatory.behavior.validation), 111  
 validate\_paths() (allensdk.internal.pipeline\_modules.run\_ophys\_time\_sync.TimeSync [method](#)), 350  
 validate\_structure\_id() (allensdk.core.reference\_space\_cache.ReferenceSpaceCache [class method](#)), 253  
 validate\_structure\_ids() (allensdk.core.reference\_space\_cache.ReferenceSpaceCache [class method](#)), 253  
 validate\_structures() (allensdk.core.reference\_space.ReferenceSpace [method](#)), 251  
 validate\_swc() (in module allensdk.internal.morphology.validate\_swc), 335  
 validate\_swc() (in module allensdk.morphology.validate\_swc), 365  
 validate\_non\_exclusivity()

(in module `allensdk.brain_observatory.behavior.trials_processing`), 111

`validate_with_synthetic_F()` (in module `allensdk.brain_observatory.r_neuropil`), 198

`ValidationError`, 111

`value_map()` (`allensdk.core.simple_tree.SimpleTree` method), 256

`values` (`allensdk.brain_observatory.running_speed.RunningSpeed` attribute), 201

`verify_roi_lists_equal()` (`allensdk.brain_observatory.session_analysis.SessionAnalysis` method), 204

`VERSION` (`allensdk.config.manifest.Manifest` attribute), 222

`vin` (`allensdk.brain_observatory.ecephys.file_io.stim_file.CamStimOneRickleStimFile` attribute), 129

`visual_degrees_to_pixels()` (`allensdk.brain_observatory.stimulus_info.BrainObservatoryMonitor` method), 209

`visual_degrees_to_pixels()` (`allensdk.brain_observatory.stimulus_info.Monitor` method), 209

`voltage_component_of_threshold_exact()` (in module `allensdk.model.glif.glif_neuron_methods`), 364

`voltage_component_of_threshold_forward_exact()` (in module `allensdk.model.glif.glif_neuron_methods`), 364

`voltage_deflection()` (`allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor` method), 267

`VolumeProjector` (class in `allensdk.internal.mouse_connectivity.projection_thumbnail.volume_projector`), 341

`VOXEL_RESOLUTION_100_MICRONS` (`allensdk.api.queries.reference_space_api.ReferenceSpaceApi` attribute), 63

`VOXEL_RESOLUTION_10_MICRONS` (`allensdk.api.queries.reference_space_api.ReferenceSpaceApi` attribute), 63

`VOXEL_RESOLUTION_25_MICRONS` (`allensdk.api.queries.reference_space_api.ReferenceSpaceApi` attribute), 63

`VOXEL_RESOLUTION_50_MICRONS` (`allensdk.api.queries.reference_space_api.ReferenceSpaceApi` attribute), 63

`vsig` (`allensdk.brain_observatory.ecephys.file_io.stim_file.CamStimOneRickleStimFile` attribute), 129

**W**

`warp_coordinates` (in module `allensdk.brain_observatory.stimulus_info.ExperimentGeometry` attribute), 209

`warp_image()` (`allensdk.brain_observatory.stimulus_info.BrainObservatoryMonitor` method), 209

`warp_stimulus_coords()` (in module `allensdk.brain_observatory.stimulus_info`), 212

`wedge_ring()` (in module `allensdk.brain_observatory.circle_plots`), 188

`whitelist()` (`allensdk.core.structure_tree.StructureTree` static method), 260

`WhitespaceStrippedString` (class in `allensdk.test_utilities.custom_comparators`), 373

`whole_lotta_trials()` (in module `allensdk.brain_observatory.behavior.criteria`), 99

`width` (`allensdk.brain_observatory.stimulus_info.Monitor` attribute), 209

`window_average()` (in module `allensdk.mouse_connectivity.grid.utilities.downsampling_utilities`), 370

`wrap()` (`allensdk.api.cache.Cache` method), 78

`write()` (`allensdk.config.model.description_parser.DescriptionParser` method), 222

`write()` (`allensdk.config.model.formats.hdf5_util.Hdf5Util` method), 219

`write()` (`allensdk.config.model.formats.json_description_parser.JsonDescriptionParser` method), 220

`write()` (`allensdk.config.model.formats.pycfg_description_parser.PycfgDescriptionParser` method), 220

`write()` (`allensdk.core.swc.Morphology` method), 264

`write()` (`allensdk.internal.brain_observatory.frame_stream.FrameOutput` method), 295

`write()` (`allensdk.internal.morphology.morphology.Morphology` method), 320

`write()` (`allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSync` method), 350

`write_bytes_from_coroutine()` (in module `allensdk.brain_observatory.ecephys.ecephys_project_api.http_engine`), 123

`write_csv()` (in module `allensdk.brain_observatory.ecephys.ecephys_project_cache`), 162

`write_file()` (`allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader` method), 278

`write_file()` (`allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader` method), 281

`write_itksnap_labels()` (in module `allensdk.brain_observatory.ecephys.ecephys_project_api.http_engine`), 123

`write_itksnap_labels()` (in module `allensdk.core.reference_space.ReferenceSpace`), 281

`method`), 251  
`write_json_file()` (al-  
`lensdk.config.manifest_builder.ManifestBuilder`  
`method`), 225  
`write_json_string()` (al-  
`lensdk.config.manifest_builder.ManifestBuilder`  
`method`), 225  
`write_metrics_csv()` (in module al-  
`lensdk.brain_observatory.ecephys.ecephys_project_cache`),  
162  
`write_ndarray_with_sitk()` (in module al-  
`lensdk.core.sitk_utilities`), 257  
`write_or_print_outputs()` (in module al-  
`lensdk.brain_observatory.argschema_utilities`),  
185  
`write_output()` (in module al-  
`lensdk.internal.pipeline_modules.run_ophys_eye_calibration`),  
348  
`write_output_data()` (al-  
`lensdk.internal.core.lims_pipeline_module.PipelineModule`  
`method`), 301  
`write_output_h5()` (al-  
`lensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncWriter`  
`method`), 350  
`write_output_json()` (al-  
`lensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncWriter`  
`method`), 350  
`write_receptive_field_to_h5()`  
(in module al-  
`lensdk.brain_observatory.receptive_field_analysis.receptive_field`),  
180  
`write_string()` (al-  
`lensdk.config.model.formats.json_description_parser.JsonDescriptionParser`  
`method`), 220  
`write_string()` (al-  
`lensdk.config.model.formats.pycfg_description_parser.PycfgDescriptionParser`  
`method`), 220  
`write_string()` (in module al-  
`lensdk.core.json_utilities`), 240  
`write_sweep_response()` (in module al-  
`lensdk.model.glif.simulate_neuron`), 365  
`write_volume()` (in module al-  
`lensdk.mouse_connectivity.grid.utilities.image_utilities`),  
371

## Y

`yesterday_was_good()` (in module al-  
`lensdk.brain_observatory.behavior.criteria`),  
100