Allen SDK Documentation

Release dev

Allen Institute for Brain Science

Aug 11, 2021

Contents

1 Install Guide **1**
- 1.1 Quick Start Using Anaconda . . . 1
- 1.2 Quick Start Using Pip . . . 1
- 1.3 Other Distribution Formats . . . 2
- 1.4 Installation with Docker (Optional) . . . 2

2 Data Resources **3**
- 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
- 2.7 Visual Behavior - Optical Physiology . . . 22

3 Models **45**
- 3.1 Generalized LIF Models . . . 45
- 3.2 Biophysical Models . . . 51

4 Examples **61**

5 Authors **63**

6 allensdk package **65**
- 6.1 Subpackages . . . 65
- 6.2 Submodules . . . 382
- 6.3 Module contents . . . 382

7 Allen Brain Observatory **383**

8 Allen Cell Types Database **385**

9 Allen Mouse Brain Connectivity Atlas **387**

10 What's New - 2.12.0 **389**

11 What's New - 2.11.3 **391**

12 What's New - 2.11.2 393

13 What's New - 2.11.1 395

14 What's New - 2.11.0 397

15 What's New - 2.10.3 399

16 What's New - 2.10.2 401

17 What's New - 2.10.1 403

18 What's New - 2.9.0 405

19 What's New - 2.8.0 407

20 What's New - 2.7.0 409

21 What's New - 2.6.0 411

22 What's New - 2.5.0 (January 29, 2021) 413

23 What's New - 2.4.0 (December 21, 2020) 415

24 What's New - 2.3.2 (October 19, 2020) 417

25 What's New - 2.3.1 (October 13, 2020) 419

26 What's New - 2.3.0 (October 9, 2020) 421

27 What's New - 2.2.0 (September 3, 2020) 423

28 What's New - 2.1.0 (July 16, 2020) 425

29 What's New - 2.0.0 (June 11, 2020) 427

30 Previous Release Notes 429

Bibliography 431

Python Module Index 433

Index 439

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.

Attention: As of October 2019, we have dropped Python 2 support. The Allen SDK is developed and tested with Python 3.6 and 3.7. We do not guarantee consistent behavior with other Python versions.

1.1 Quick Start Using Anaconda

1. From the Anaconda downloads page, download the Python 3.7 version for your operating system and run the installer.
2. After the installation is complete, open up a terminal (in Windows open Anaconda3 Command Prompt).
3. Install the AllenSDK using PIP:

```
pip install allensdk
```

4. Download one of our many Jupyter Notebook examples to a new folder.
5. In your terminal, navigate to the directory where you downloaded the Jupyter Notebook example and run the following command:

```
jupyter notebook
```

6. Your browser should open and you should see the Jupyter Notebook example. Enjoy using the Allen SDK!

1.2 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
```

1.3 Other Distribution Formats

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

1.4 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 the image:

   ```
   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 alleninstitute/allensdk /bin/bash
   ```

4. Run the SDK tests:

   ```
   cd allensdk
   make test
   ```

5. Start a Jupyter Notebook:

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

   Using the browser on your host machine, navigate to the path provided by the output from the jupyter notebook command.

CHAPTER 2

Data Resources

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.

_static/container_session_layout.png

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>=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 fluoresence 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 fluorescences 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 discrimiation 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 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:

```python
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 platorms. 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 MouseConenctivityApi 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 projetion 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.core.reference_space_cache.ReferenceSpaceCache* 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.core.reference_space_cache.ReferenceSpaceCache* class. There is an example in the notebook.

2.4.3 ReferenceSpaceCache Class

The *allensdk.core.reference_space_cache.ReferenceSpaceCache* class provides a Python interface for downloading structure trees and annotation volumes. It takes care of knowing if you've already downloaded the files and reads them from disk instead of downloading them again.

The 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 descendents 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(
```

(continues on next page)

(continued from previous page)

```
        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.warehouse_cache.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 to_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 AllenSDK 2.0 and Data Compatability

AllenSDK version 2.0 marks a major update to released Visual Coding Neuropixels datasets. Due to newer versions of pynwb/hdmf having issues reading previously released Visual Coding Neuropixels NWB files and due to the significant reorganization of updated NWB file contents, this release contains breaking changes that necessitate a major version revision. NWB files released prior to 6/11/2020 are not guaranteed to work with the 2.0.0 version of AllenSDK. If you cannot or choose not to re-download the updated NWB files, you can continue using a prior version of AllenSDK (< 2.0.0) to access them. However, no further features or bugfixes for AllenSDK (< 2.0.0) are planned. Data released for other projects (Cell Types, Mouse Connectivity, etc.) are *NOT* affected and will *NOT* need to be re-downloaded.

When using the Visual Coding **EcephysProjectCache** from this updated AllenSDK version, if a **ManifestError** is encountered, this indicates that previously downloaded cached data files need to be removed and re-downloaded. The location these files as well as manifest are user defined and are set when instantiating an **EcephysProjectCache**.

2.6.5 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)

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

Continued on next page

Table 2 – continued from previous page

Stimulus	Metric	Field Name
	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
gabors	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

2.7 Visual Behavior - Optical Physiology

The Visual Behavior 2P project used in vivo 2-photon calcium imaging (also called optical physiology, or "ophys") to measure the activity of populations of genetically identified neurons in the visual cortex of mice performing a visually guided behavioral task (image change detection, described below). We used single- and multi-plane imaging approaches to record the activity of populations of excitatory neurons and two inhibitory classes, Somatostatin (Sst) and Vasoactive Intestinal Peptide (Vip) expressing interneurons, across multiple cortical depths and two visual areas (VSIp and VISl). Each population of neurons was imaged repeatedly over multiple days under different sensory and behavioral contexts, including with familiar and novel stimuli, as well as active behavior and passive viewing conditions. This dataset can be used to evaluate the influence of experience, expectation, and task engagement on neural coding and dynamics.

_static/visual_behavior_2p/datasets.png

While 2-photon imaging data was acquired in well-trained mice, the full behavioral training history of all imaged mice is also provided, allowing investigation into task learning, behavioral strategy, and inter-animal variability.

Overall, the dataset includes neural and behavioral measurements from 82 mice, including 3021 behavior training sessions and 551 in vivo 2-photon imaging sessions, resulting in longitudinal recordings from 34,619 cortical cells.

The table below describes the numbers of mice, sessions, and unique recorded neurons for each transgenic line and imaging platform in the dataset:

_static/visual_behavior_2p/variants_table.png

2.7.1 GETTING STARTED

First, install or update the AllenSDK, our Python based toolkit for accessing and working with Allen Institute datasets.

Data is provided in in NWB format and can be downloaded using the AllenSDK, or accessed directly via an S3 bucket (instructions provided in notebook #1 below). Regardless of which method of file download you choose, we recommend that you load and interact with the data using the tools provided in the AllenSDK, which have been designed to simplify data access and subsequent analysis. No knowledge of the NWB file format is required.

Specific information about how Visual Behavior Optical Physiology data is stored in NWB files and how AllenSDK accesses NWB files can be found here.

To get started, check out these jupyter notebooks to learn how to:

1) Download data using the AllenSDK or directly from our Amazon S3 bucket (download .ipynb) (Open in Colab)
2) Identify experiments of interest using the dataset manifest (download .ipynb) (Open in Colab)
3) Load and visualize data from a 2-photon imaging experiment (download .ipynb) (Open in Colab)
4) Examine the full training history of one mouse (download .ipynb) (Open in Colab)
5) Compare behavior and neural activity across different trial types in the task (download .ipynb) (Open in Colab)

For a description of available AllenSDK methods and attributes for data access, see this further documentation.

For detailed information about the experimental design, data acquisition, and informatics methods, please refer to our technical whitepaper.

If you have questions about the dataset that aren't addressed by the whitepaper or any of our tutorials, please reach out by posting at https://community.brain-map.org/

2.7.2 CHANGE DETECTION TASK

_static/visual_behavior_2p/task.png

We trained mice to perform a go/no-go visual change detection task in which they learned to lick a spout in response to changes in stimulus identity to earn a water reward. Visual stimuli are continuously presented over a 1-hour session, with no explicit cue to indicate the start of a trial. Mice are free to run on a circular disk during the session.

We used a standardized procedure to progress mice through a series of training stages, with transitions between stages determined by specific advancement criteria. First, mice learned to detect changes in the orientation of full field static grating stimuli. Next, a 500ms inter stimulus interval period with mean luminance gray screen was added between the 250ms stimulus presentations, incorporating a short-term memory component to the task. Once mice successfully and consistently performed the orientation change detection with flashed gratings, they moved to the image change detection version of the task. During image change detection, 8 natural scene images were presented during each behavioral session, for a total of 64 possible image transitions. When behavioral performance again reached criterion, mice were transitioned to the 2-photon imaging stage in which they performed the task under a microscope to allow simultaneous measurement of neural activity and behavior.

_static/visual_behavior_2p/automated_training.png

Behavioral training data for mice progressing through these stages of task learning is accessible via the **BehaviorSession** class of the AllenSDK or the `get_behavior_session()` method of the **VisualBehaviorOphysProjectCache**. Each **BehaviorSession** contains the following data streams, event times, and metadata:

- Running speed
- Lick times
- Reward times
- Stimulus presentations
- Behavioral trial information
- Mouse metadata (age, sex, genotype, etc)

_static/visual_behavior_2p/behavior_timeseries.png

2.7.3 2-PHOTON IMAGING DATASET

Once mice are well-trained on the image change detection task, they transition to performing the behavior under a 2-photon microscope. During the imaging phase, mice undergo multiple **session types**, allowing measurement of neural activity across different sensory and behavioral contexts.

_static/visual_behavior_2p/experiment_design.png

Mice initially perform the task under the microscope with the same set of images they observed during training, which have become highly familiar (each image is viewed thousands of times during training). Mice also undergo several sessions with a novel image set that they had not seen prior to the 2-photon imaging portion of the experiment. Interleaved between active behavior sessions, are passive viewing sessions where the mice are given their daily water before the session (and are thus satiated) and view the task stimuli with the lick spout retracted so they are unable to earn water rewards. This allows investigation of the impact of motivation and attention on patterns of neural activity. Finally, stimuli were randomly omitted with a 5% probability, resulting in an extended gray screen period between two presentations of the same stimulus, and disrupting the expected cadence of stimulus presentations. Stimuli were only omitted during the 2-photon imaging sessions (not during training), and change stimuli were never omitted.

We used both single- and multi-plane 2-photon imaging to record the activity of GCaMP6 expressing cells in populations of excitatory (Slc17a7-IRES2-Cre;Camk2a-tTA;Ai93(TITL-GCaMP6)) and inhibitory (Vip-IRES-Cre;Ai148(TIT2L-GC6f-ICL-tTA2) & Sst-IRES-Cre;Ai148(TIT2L-GC6f-ICL-tTA2)) neurons. Imaging took place between 75-400um below the cortical surface.

_static/visual_behavior_2p/cre_lines.png

The data collected in a single continuous recording is defined as a **session**. For single-plane imaging experiments, there is only one imaging plane (referred to as an **experiment**) per session. For multi-plane imaging experiments, there can be up to 8 imaging planes (aka 8 experiments) per session. Due to our strict QC process, described below, not all multi-plane imaging sessions have exactly 8 experiments, as some imaging planes did not meet our data quality criteria.

We aimed to track the activity of single neurons across the session types described above by targeting the same population of neurons over multiple recording sessions, with only one session recorded per day for a given mouse. The collection of imaging sessions for a given population of cells, belonging to a single imaging plane measured across days, is called a **container**. A container can include between 3 and 11 separate sessions for a given imaging plane. Mice imaged with the multi-plane 2-photon microscope can have multiple containers, one for each imaging plane recorded across multiple sessions. The session types available for a given container can vary, due to our selection criteria to ensure data quality (described below).

Thus, each mouse can have one or more **containers**, each representing a unique imaging plane (**experiment**) that has been targeted across multiple recording **sessions**, under different behavioral and sensory conditions (**session types**).

_static/visual_behavior_2p/data_structure.png

The **BehaviorOphysExperiment** class in the AllenSDK (or the `get_behavior_ophys_experiment()` method of the **VisualBehaviorOphysProjectCache**) provides all data for a single imaging plane, recorded in a single session, and contains the following data streams in addition to the behavioral data described above:

- Max intensity image
- Average intensity image
- Segmentation masks
- dF/F traces (baseline corrected, normalized fluorescence traces)
- Corrected fluorescence traces (neuropil subtracted and demixed, but not normalized)
- Events (detected with an L0 event detection algorithm)
- Pupil position
- Pupil area

_static/visual_behavior_2p/behavior_and_ophys_timeseries.png

2.7.4 DATA PROCESSING

Each 2-photon movie is processed through a series of steps to obtain single cell traces of baseline-corrected fluorescence (dF/F) and extracted events, that are packaged into NWB files along with stimulus and behavioral information, as well as other metadata.

_static/visual_behavior_2p/data_processing.png

Detailed descriptions of data processing steps can be found in the technical white paper, as well as our data processing repository.

2.7.5 QUALITY CONTROL

Every 2-photon imaging session was carefully evaluated for a variety of quality control criteria to ensure that the final dataset is of the highest quality possible. Sessions or imaging planes that do not meet our criteria are excluded from the dataset in this release. These are a few of the key aspects of the data that are evaluated:

- intensity drift
- image saturation or bleaching
- z-drift over the course of a session
- accuracy of session-to-session field of view matching
- excessive or uncorrectable motion in the image
- uncorrectable crosstalk between simultaneously recorded multiscope planes
- errors affecting temporal alignment of data streams
- hardware or software failures
- brain health
- animal stress

2.7.6 SUMMARY OF AVAILABLE DATA

Behavior	Physiology	Metadata
Running speed	Max intensity projection image	Mouse genotype, age, sex
Licks	Average projection image	Date of acquisition
Rewards	Segmentation mask image	Imaging parameters
Pupil area	Cell specimen table	Task parameters
Pupil position	Cell ROI masks	Session type
Stimulus presentations table	Corrected fluorescence traces	Stimulus images
Trials table	dF/F activity traces	Performance metrics
Stimulus timestamps	Detected events	
	Ophys timestamps	

2.7.7 DATA FILE CHANGELOG

v1.0.0

New Data

- 107 mice, up from 82
- 4082 behavior training sessions, up from 3021.
- 705 in vivo 2-photon imaging sessions, up from 551.
- 50,489 logitudinal recordings from cortical cells, up from 34,619

Metadata changes

- A new metadata table is present: ophys_cells_table. This table has a project-wide aggregate of cell_specimen_id, cell_roi_id, and ophys_experiment_id.
- Added 'experience_level', 'passive' and 'image_set' columns to ophys_experiments_table

Data Corrections

- 196 BehaviorOphysExperiments had excess invalid ROIs in the dataframe returned by the events field. These have been corrected to remove these invalid ROIs.

v0.3.0

13 sessions were labeled with the wrong session_type in v0.2.0. We have corrected that error. The offending sessions were

behavior_session_id	ophys_session_id	session_type_v0.2.0	session_type_v0.3.0
875020233		OPHYS_3_images_A	OPHYS_2_images_A_passive
902810506		TRAINING_4_images_B_training	TRAINING_3_images_B_10uL_reward
914219174		OPHYS_0_images_B_habituation	TRAINING_5_images_B_handoff_ready
863571063		TRAINING_5_images_A_handoff_ready	TRAINING_1_gratings
974330793		OPHYS_0_images_B_habituation	TRAINING_5_images_B_handoff_ready
863571072		OPHYS_5_images_B_passive	TRAINING_4_images_A_training
1010972317		OPHYS_4_images_A	OPHYS_3_images_B
1011659817		OPHYS_5_images_A_passive	OPHYS_4_images_A
1003302686	1003277121	OPHYS_6_images_A	OPHYS_5_images_A_passive
863571054		OPHYS_7_receptive_field_mapping	TRAINING_5_images_A_epilogue
974282914	974167263	OPHYS_6_images_B	OPHYS_5_images_B_passive
885418521		OPHYS_1_images_A	TRAINING_5_images_A_handoff_lapsed
915739774		OPHYS_1_images_A	OPHYS_0_images_A_habituation

CHAPTER 3

Models

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 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
El	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
voltage_dynamics_method	Update simulation voltage for the next time step.
threshold_dynamics_method	Update simulation threshold for the next time step.
AScurrent_dynamics_method	Update afterspike current coefficients for the next time step.
voltage_reset_method	Reset simulation voltage after a spike occurs.
threshold_reset_method	Reset simulation threshold after a spike occurs.
AScurrent_reset_method	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 voltageto 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 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 be 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
```

(continues on next page)

(continued from previous page)

```
|-- 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 # perisomatic models
python -m allensdk.model.biophysical.runner manifest.json # legacy all-active models
# new all-active models (axon replaced by a 60 micron long 1 micron diameter stub)
python -m allensdk.model.biophysical.runner manifest.json --axon_type stub
```

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.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	Jupyter notebook (download .ipynb)
Introduction to the Cell Types Database	Jupyter notebook (download .ipynb)
Introduction to the Brain Observatory	Jupyter notebook (download .ipynb)
Brain Observatory Stimulus Manipulation	Jupyter notebook (download .ipynb)
Brain Observatory Tuning Analysis	Jupyter notebook (download .ipynb)
Brain Observatory Receptive Field Analysis	Jupyter notebook (download .ipynb)
Brain Observatory Cell Specimen ID Mapping	Jupyter notebook (download .ipynb)
Brain Observatory Monitor	Jupyter notebook (download .ipynb)
Dynamic Brain Workshop 2015 experiment detail	Jupyter notebook (download .ipynb)
Stimulating a biophysical model with a square pulse	Jupyter notebook (download .ipynb)
Using a Reference Space	Jupyter notebook (download .ipynb)
Downloading Images	Jupyter notebook (download .ipynb)
Visual Coding Neuropixels Quick Start	Jupyter notebook (download .ipynb)
Visual Coding Neuropixels Reference	Jupyter notebook (download .ipynb)

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

Wesley Jones @wesley-jones

Justin Kiggins @neuromusic

Joe Knox @jknox13

Peter Ledochowitsch @ledochowitsch

Nicholas Mei @njmei

Chris Mochizuki @mochic

Ani Nandi @anirban6908

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

Isaak Willett @isaak-willett

Derric Williams @derricw

CHAPTER 6

allensdk package

6.1 Subpackages

6.1.1 allensdk.api package

Subpackages

allensdk.api.cloud_cache package

Submodules

allensdk.api.cloud_cache.cloud_cache module

allensdk.api.cloud_cache.file_attributes module

class allensdk.api.cloud_cache.file_attributes.**CacheFileAttributes**(*url: str, version_id: str, file_hash: str, local_path: pathlib.Path*)

Bases: `object`

This class will contain the attributes of a remotely stored file so that they can easily and consistently be passed around between the methods making up the remote file cache and manifest classes

Parameters

url: str The full URL of the remote file

version_id: str A string specifying the version of the file (probably calculated by S3)

file_hash: str The (hexadecimal) file hash of the file

local_path: pathlib.Path The path to the location where the file's local copy should be stored (probably computed by the Manifest class)

file_hash

local_path

url

version_id

allensdk.api.cloud_cache.manifest module

class allensdk.api.cloud_cache.manifest.**Manifest**(*cache_dir: Union[str, pathlib.Path], json_input, use_static_project_dir: bool = False*)

Bases: object

A class for loading and manipulating the online manifest.json associated with a dataset release

Each Manifest instance should represent the data for 1 and only 1 manifest.json file.

Parameters

cache_dir: str or pathlib.Path The path to the directory where local copies of files will be stored

json_input: A ''.read()''-supporting file-like object containing a JSON document to be deserialized (i.e. same as the first argument to json.load)

use_static_project_dir: bool When determining what the local path of a remote resource (data or metadata file) should be, the Manifest class will typically create a versioned project subdirectory under the user provided *cache_dir* (e.g. f"{cache_dir}/{project_name}-{manifest_version}") to allow the possibility of multiple manifest (and data) versions to be used. In certain cases, like when using a project's s3 bucket directly as the cache_dir, the project directory name needs to be static (e.g. f"{cache_dir}/{project_name}"). When set to True, the Manifest class will use a static project directory to determine local paths for remote resources. Defaults to False.

data_file_attributes(*self, file_id*) → allensdk.api.cloud_cache.file_attributes.CacheFileAttributes

Return the CacheFileAttributes associated with a data file

Parameters

file_id: The identifier of the data file whose attributes are to be returned. Must be a key in self._data['data_files']

Raises

RuntimeError If you try to run this method when self._data is None (meaning you haven't yet loaded a manifest.json file)

ValueError If the file_id is not a valid option

file_id_column

The column in the metadata files used to uniquely identify data files

file_id_values

List of valid file_id values

metadata_file_attributes(*self*, *metadata_file_name: str*) → allensdk.api.cloud_cache.file_attributes.CacheFileAttributes
Return the CacheFileAttributes associated with a metadata file

Parameters

metadata_file_name: str Name of the metadata file. Must be in self.metadata_file_names

Raises

RuntimeError If you try to run this method when self._data is None (meaning you haven't yet loaded a manifest.json)

ValueError If the metadata_file_name is not a valid option

metadata_file_names
List of metadata file names associated with this dataset

project_name
The name of the project whose data and metadata files this manifest tracks.

version
The version of the dataset currently loaded

allensdk.api.cloud_cache.utils module

allensdk.api.cloud_cache.utils.**bucket_name_from_url**(*url: str*) → Union[str, NoneType]
Read in a URL and return the name of the AWS S3 bucket it points towards.

Parameters

URL: str A generic URL, suitable for retrieving an S3 object via an HTTP GET request.

Returns

str An AWS S3 bucket name. Note: if 's3.amazonaws.com' does not occur in the URL, this method will return None and emit a warning.

allensdk.api.cloud_cache.utils.**file_hash_from_path**(*file_path: Union[str, pathlib.Path]*) → str
Return the hexadecimal file hash for a file

Parameters

file_path: Union[str, Path] path to a file

Returns

str: The file hash (Blake2b; hexadecimal) of the file

allensdk.api.cloud_cache.utils.**relative_path_from_url**(*url: str*) → str
Read in a url and return the relative path of the object

Parameters

url: str The url of the object whose path you want

Returns

str: Relative path of the object

Notes

This method returns a str rather than a pathlib.Path because it is used to get the S3 object Key from a URL. If using Pathlib.path on a Windows system, the '/' will get transformed into '', confusing S3.

Module contents

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, 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] '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 repsonse 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 repsonse 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*, *datacube_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 tissuecyte 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 reconstructions.

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 reconstructions.

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*, *reporter_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 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.

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, optiona] 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 metaimage 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 SectionImage, AtlasImage and ProjectionImage Download Services.

See Downloading an Image for more documentation.

COLORMAPS = {'aba': 8, 'aibsmap_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 ImageDownloadApi.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_uppper, 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 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 >= 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=['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=['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

excpt [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

allensdk.api.warehouse_cache package

Submodules

allensdk.api.warehouse_cache.cache module

class allensdk.api.warehouse_cache.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.warehouse_cache.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] df|json->df|json, takes one data argument and returns filtered version, None for pass-through

post [function] df|json->?, 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.warehouse_cache.cache.`**`get_default_manifest_file`**(*cache_name*)

`allensdk.api.warehouse_cache.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.warehouse_cache.caching_utilities module

`allensdk.api.warehouse_cache.caching_utilities.`**`call_caching`**(*fetch: Callable[[], ~Q], write: Callable[[~Q], NoneType], read: Union[Callable[[], ~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]

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.warehouse_cache.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

Submodules

allensdk.api.api module

class allensdk.api.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 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 seperate 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 seperate connect and read timeouts.

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_project_cache package

Subpackages

allensdk.brain_observatory.behavior.behavior_project_cache.external package

Submodules

allensdk.brain_observatory.behavior.behavior_project_cache.external.behavior_project_metadata_writer module

Module contents

allensdk.brain_observatory.behavior.behavior_project_cache.tables package

Subpackages

allensdk.brain_observatory.behavior.behavior_project_cache.tables.util package

Submodules

allensdk.brain_observatory.behavior.behavior_project_cache.tables.util.experiments_table_utils module

allensdk.brain_observatory.behavior.behavior_project_cache.tables.util.experiments_table_ut

adds a column to ophys_experiment_table that contains a string indicating whether a session had exposure level of Familiar, Novel 1, or Novel >1, based on session number and prior_exposure_to_image_set

Parameters

experiments_table: pd.DataFrame

Returns

experiments_table: pd.DataFrame

Notes

Does not change the input DataFrame in-place

allensdk.brain_observatory.behavior.behavior_project_cache.tables.util.experiments_table_ut

Adds a column 'image_set' to the experiment_table, determined based on the image set listed in the session_type column string

Parameters

experiments_table: pd.DataFrame

Returns

experiments_table: pd.DataFrame

Notes

Does not alter the input DataFrame in-place

allensdk.brain_observatory.behavior.behavior_project_cache.tables.util.experiments_table_ut

adds a column to ophys_experiment_table that contains a Boolean indicating whether a session was passive or not based on session number

Parameters

experiments_table: pd.DataFrame

Returns

experiments_table: pd.DataFrame

allensdk.brain_observatory.behavior.behavior_project_cache.tables.util.prior_exposure_processing module

Module contents

Submodules

allensdk.brain_observatory.behavior.behavior_project_cache.tables.experiments_table module

class allensdk.brain_observatory.behavior.behavior_project_cache.tables.experiments_table.**I**

Bases: *allensdk.brain_observatory.behavior.behavior_project_cache.tables.project_table.ProjectTable*, *allensdk.brain_observatory.behavior.behavior_project_cache.tables.ophys_mixin.OphysMixin*

Class for storing and manipulating project-level data at the behavior-ophys experiment level

final_processing(*self*)

postprocess_additional(*self*)
Additional postprocessing should be overridden by subclassess

allensdk.brain_observatory.behavior.behavior_project_cache.tables.ophys_mixin module

class allensdk.brain_observatory.behavior.behavior_project_cache.tables.ophys_mixin.**OphysMi**

Bases: object

A mixin class for ophys project data

allensdk.brain_observatory.behavior.behavior_project_cache.tables.ophys_sessions_table module

class allensdk.brain_observatory.behavior.behavior_project_cache.tables.ophys_sessions_tabl

Bases: *allensdk.brain_observatory.behavior.behavior_project_cache.tables.project_table.ProjectTable*, *allensdk.brain_observatory.behavior.behavior_project_cache.tables.ophys_mixin.OphysMixin*

Class for storing and manipulating project-level data at the behavior-ophys session level

postprocess_additional(*self*)
Additional postprocessing should be overridden by subclassess

allensdk.brain_observatory.behavior.behavior_project_cache.tables.project_table module

class allensdk.brain_observatory.behavior.behavior_project_cache.tables.project_table.**Proje**

Bases: abc.ABC

Class for storing and manipulating project-level data

postprocess(*self*)
Postprocess loop

postprocess_additional(*self*)
Additional postprocessing should be overridden by subclassess

postprocess_base(*self*)
Postprocessing to apply to all project-level data

table

allensdk.brain_observatory.behavior.behavior_project_cache.tables.sessions_table module

Module contents

Submodules

allensdk.brain_observatory.behavior.behavior_project_cache.behavior_project_cache module

Module contents

allensdk.brain_observatory.behavior.metadata package

Submodules

allensdk.brain_observatory.behavior.metadata.behavior_metadata module

class allensdk.brain_observatory.behavior.metadata.behavior_metadata.**BehaviorMetadata**(*extract al- lensdk stim- u- lus_tim numpy be- hav- ior_sti dict*)

Bases: object

Container class for behavior metadata

age_in_days
Converts the age cod into a numeric days representation

behavior_session_id

behavior_session_uuid
Get the universally unique identifier (UUID)

cre_line
Parses cre_line from full_genotype

date_of_acquisition
Return the timestamp for when experiment was started in UTC

NOTE: This method will only get acquisition datetime from extractor (data from LIMS) methods. As a sanity check, it will also read the acquisition datetime from the behavior stimulus (*.pkl) file and raise a warning if the date differs too much from the datetime obtained from the behavior stimulus (*.pkl) file.

Return type datetime

driver_line

equipment_name

full_genotype

get_extractor(*self*)

mouse_id

static parse_age_in_days(*age: str*, *warn=False*) → Union[int, NoneType]
Converts the age code into a numeric days representation

Parameters

age age code, ie P123

warn Whether to output warning if parsing fails

static parse_cre_line (*full_genotype: str*, *warn=False*) → Union[str, NoneType]

Parameters

full_genotype formatted from LIMS, e.g. Vip-IRES-Cre/wt;Ai148(TIT2L-GC6f-ICL-tTA2)/wt

warn Whether to output warning if parsing fails

Returns

cre_line just the Cre line, e.g. Vip-IRES-Cre, or None if not possible to parse

static parse_indicator (*reporter_line: Union[str, NoneType]*, *warn=False*) → Union[str, NoneType]

Parses indicator from reporter

static parse_reporter_line (*reporter_line: Union[List[str], NoneType]*, *warn=False*) → Union[str, NoneType]

There can be multiple reporter lines, so it is returned from LIMS as a list. But there shouldn't be more than 1 for behavior. This tries to convert to str

Parameters

reporter_line List of reporter line

warn Whether to output warnings if parsing fails

Returns

single reporter line, or None if not possible

reporter_line

session_type

sex

stimulus_frame_rate

to_dict (*self*) → dict

Returns dict representation of all properties in class

allensdk.brain_observatory.behavior.metadata.behavior_metadata.**get_expt_description** (*session_t*
str) → str

Determine a behavior ophys session's experiment description based on session type. Matches the regex patterns defined as the keys in description_dict

Parameters

session_type [str] A session description string (e.g. OPHYS_1_images_B)

Returns

str A description of the experiment based on the session_type.

Raises

RuntimeError Behavior ophys sessions should only have 6 different session types. Unknown session types (or malformed session_type strings) will raise an error.

allensdk.brain_observatory.behavior.metadata.behavior_metadata.**get_task_parameters**(*data: Dict*) → Dict

Read task_parameters metadata from the behavior stimulus pickle file.

Parameters

data: dict The nested dict read in from the behavior stimulus pickle file. All of the data expected by this method lives under data['items']['behavior']

Returns

dict A dict containing the task_parameters associated with this session.

allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata module

class allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata.**BehaviorOphysMet**

Bases: *allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata*

Container class for behavior ophys metadata

emission_lambda

excitation_lambda

experiment_container_id

field_of_view_height

field_of_view_width

imaging_depth

imaging_plane_group

imaging_plane_group_count

indicator
Parses indicator from reporter

ophys_experiment_id

ophys_frame_rate

ophys_session_id

project_code

targeted_structure

to_dict (*self*) → dict
Returns dict representation of all properties in class

Module contents

allensdk.brain_observatory.behavior.session_apis package

Subpackages

allensdk.brain_observatory.behavior.session_apis.abcs package

Subpackages

allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base package

Submodules

allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base module

class allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_da
Bases: abc.ABC

Abstract base class implementing required methods for extracting data (from LIMS or from JSON) that will be transformed or passed on to fill behavior session data.

get_age (*self*) → str
Get the age code of the subject (ie P123)

get_behavior_session_id (*self*) → int
Get the ID of the behavior session

get_behavior_stimulus_file (*self*) → str
Get the filepath to the StimulusPickle file for the session

get_date_of_acquisition (*self*) → datetime.datetime
Get the acquisition date of an experiment in UTC

get_driver_line (*self*) → List[str]
Get the (gene) driver line(s) for the subject associated with a behavior or behavior + ophys experiment

get_equipment_name (*self*) → str
Get the name of the experiment rig (ex: CAM2P.3)

get_foraging_id (*self*) → int
Get the foraging ID for the behavior session

get_full_genotype (*self*) → str
Get the full genotype of the subject associated with a behavior or behavior + ophys experiment

get_mouse_id (*self*) → int
Get the mouse id (LabTracks ID) for the subject associated with a behavior experiment

get_reporter_line (*self*) → List[str]
Get the (gene) reporter line(s) for the subject associated with a behavior or behavior + ophys experiment

get_sex (*self*) → str
Get the sex of the subject (ex: 'M', 'F', or 'unknown')

get_stimulus_name (*self*) → str
Get the stimulus set used from the behavior session pkl file :rtype: str

allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_ophys_data_extractor_base module

class allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_op
Bases: *allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDataExtractorBase*

Abstract base class implementing required methods for extracting data (from LIMS or from JSON) that will be transformed or passed on to fill behavior + ophys session data.

get_average_intensity_projection_image_file (*self*) → str
Get the avg intensity project image filepath associated with an ophys experiment

get_demix_file (*self*) → str
Get the filepath of the demixed traces file associated with an ophys experiment

get_dff_file (*self*) → str
Get the filepath of the dff trace file associated with an ophys experiment.

get_event_detection_filepath (*self*) → str
Get the filepath of the .h5 events file associated with an ophys experiment

get_eye_tracking_filepath (*self*) → dict
Get the eye tracking filepath containing ellipse fits

get_eye_tracking_rig_geometry (*self*) → dict
Get the eye tracking rig geometry associated with an ophys experiment

get_field_of_view_shape (*self*) → Dict[str, int]
Get a field of view dictionary for a given ophys experiment. ex: {"width": int, "height": int}

get_imaging_depth (*self*) → int
Get the imaging depth for an ophys experiment (ex: 400, 500, etc.)

get_imaging_plane_group (*self*) → Union[int, NoneType]
Get the imaging plane group number. This is a numeric index that indicates the order that the frames were acquired when there is more than one frame acquired concurrently. Relevant for mesoscope data timestamps, as the laser jumps between plane groups during the scan. Will be None for non-mesoscope data.

get_max_projection_file (*self*) → str
Get the filepath of the max projection image associated with the ophys experiment

get_ophys_cell_segmentation_run_id (*self*) → int
Get the ophys cell segmentation run id associated with an ophys experiment id

get_ophys_container_id (*self*) → int
Get the experiment container id associated with an ophys experiment

get_ophys_experiment_id (*self*) → int
Return the ophys experiment id (experiments are an internal alias for an imaging plane)

get_ophys_session_id (*self*) → int
Return the ophys session id

get_plane_group_count (*self*) → int
Gets the total number of plane groups in the session. This is required for resampling ophys timestamps for mesoscope data. Will be 0 if the scope did not capture multiple concurrent frames. See *get_imaging_plane_group* for more info.

get_project_code (*self*) → str
Get the project code.

get_raw_cell_specimen_table_dict (*self*) → dict
Get the cell_rois table from LIMS in dictionary form

get_rigid_motion_transform_file (*self*) → str
Get the filepath for the motion transform file (.csv) associated with an ophys experiment

get_surface_2p_pixel_size_um (*self*) → float
Get the pixel size for 2-photon movies in micrometers

get_sync_file (*self*) → str
Get the filepath of the sync timing file associated with the ophys experiment

get_targeted_structure (*self*) → str
Get the targeted structure (acronym) for an ophys experiment (ex: "Visp")

Module contents

allensdk.brain_observatory.behavior.session_apis.abcs.session_base package

Submodules

allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_base module

class allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_base.**Beha**
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_behavior_session_id (*self*) → int
Returns the behavior_session_id associated with this experiment, if applicable.

get_licks (*self*) → pandas.core.frame.DataFrame
Get lick data from pkl file.

Returns

pd.Dataframe A dataframe containing lick timestamps.

get_metadata (*self*) → Union[allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata, dict]
Get metadata for Session

Returns

dict if NWB

BehaviorMetadata otherwise

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_acquisition_df(*self*) → pandas.core.frame.DataFrame

Get running speed acquisition data from a behavior pickle file.

Returns

pd.DataFrame

Dataframe with an index of timestamps and the following columns: "speed": computed running speed "dx": angular change, computed during data collection "v_sig": voltage signal from the encoder "v_in": the theoretical maximum voltage that the encoder

will reach prior to "wrapping". This should theoretically be 5V (after crossing 5V goes to 0V, or vice versa). In practice the encoder does not always reach this value before wrapping, which can cause transient spikes in speed at the voltage "wraps".

get_running_speed(*self*) → pandas.core.frame.DataFrame

Get running speed using timestamps from self.get_stimulus_timestamps.

NOTE: Do not correct for monitor delay.

Returns

pd.DataFrame

timestamps [np.ndarray] index consisting of timestamps of running speed data samples

speed [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*) → allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.Stimulus

Get stimulus templates (movies, scenes) for behavior session.

Returns

StimulusTemplate

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 no 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.session_apis.abcs.session_base.behavior_ophys_base module

class allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_bas
Bases: *allensdk.brain_observatory.behavior.session_apis.abcs.session_base.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.

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_events (*self*) → pandas.core.frame.DataFrame
Get event detection data

Returns

pd.DataFrame

index: cell_specimen_id: int

cell_roi_id: int events: np.array filtered_events: np.array lambdas: float64 noise_stds: float64

get_eye_tracking (*self*) → Union[pandas.core.frame.DataFrame, NoneType]
Get eye tracking data from behavior + ophys session.

Returns

pd.DataFrame A refined eye tracking dataframe that contains information about eye tracking ellipse fits, frame times, eye areas, pupil areas, and frames with likely blinks/outliers.

get_eye_tracking_rig_geometry (*self*) → dict
Get eye tracking rig metadata from behavior + ophys session.

Returns

dict Includes geometry of monitor, camera, LED

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*) → Union[allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata.BehaviorOphysMeta dict]
Get behavior+ophys session metadata.

Returns

dict if NWB

BehaviorOphysMetadata otherwise

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_experiment_id (*self*) → Union[int, NoneType]
Returns the ophys_experiment_id for the instantiated BehaviorOphys Session (or BehaviorOphys data fetcher) if applicable.

get_ophys_session_id (*self*) → Union[int, NoneType]
Returns the behavior + ophys_session_id associated with this experiment, if applicable.

get_ophys_timestamps (*self*) → numpy.ndarray
Get optical physiology frame timestamps.

Returns

np.ndarray Timestamps associated with frames captured by the microscope.

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.

Module contents

Module contents

allensdk.brain_observatory.behavior.session_apis.data_io package

Submodules

allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api module

class allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api.**BehaviorJs**

Bases: *allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms.BehaviorDataTransforms*

A data fetching and processing class that serves processed data from a specified raw data source (extractor). Contains all methods needed to fill a BehaviorSession.

class allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api.**BehaviorJs**

Bases: *allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDataExtractorBase*

A class which 'extracts' data from a json file. The extracted data is necessary (but not sufficient) for populating a 'BehaviorSession'.

Most data provided by this extractor needs to be processed by BehaviorDataTransforms methods in order to usable by 'BehaviorSession's.

This class is used by the write_nwb module for behavior sessions.

get_age (*self*) → str
Get the age code of the subject (ie P123)

get_behavior_session_id (*self*) → int
Get the ID of the behavior session

get_behavior_stimulus_file (*self*) → str
Get the filepath to the StimulusPickle file for the session

get_date_of_acquisition (*self*) → datetime.datetime
Get the acquisition date of an experiment (in UTC)

NOTE: LIMS writes to JSON in local time. Needs to be converted to UTC

get_driver_line (*self*) → str
Get the (gene) driver line for the subject associated with an experiment

get_equipment_name (*self*) → str
Get the name of the experiment rig (ex: CAM2P.3)

get_foraging_id (*self*) → int
Get the foraging ID for the behavior session

get_full_genotype (*self*) → str
Get the full genotype of the subject associated with an experiment

get_mouse_id(*self*) → int
Get the external specimen id (LabTracks ID) for the subject associated with a behavior experiment

get_reporter_line(*self*) → str
Get the (gene) reporter line for the subject associated with an experiment

get_sex(*self*) → str
Get the sex of the subject (ex: 'M', 'F', or 'unknown')

allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api module

class allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.**BehaviorL**

Bases: *allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms.BehaviorDataTransforms*, *allensdk.core.cache_method_utilities.CachedInstanceMethodMixin*

A data fetching and processing class that serves processed data from a specified raw data source (extractor). Contains all methods needed to fill a BehaviorSession.

class allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.**BehaviorL**

Bases: *allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDataExtractorBase*

A data fetching class that serves as an API for fetching 'raw' data from LIMS necessary (but not sufficient) for filling a 'BehaviorSession'.

Most 'raw' data provided by this API needs to be processed by BehaviorDataTransforms methods in order to usable by 'BehaviorSession's

classmethod from_foraging_id(*foraging_id: Union[str, uuid.UUID, int], lims_credentials: Union[allensdk.core.authentication.DbCredentials, NoneType] = None*) → 'BehaviorLimsApi'

Create a BehaviorLimsAPI instance from a foraging_id instead of a behavior_session_id.

NOTE: 'foraging_id' in the LIMS behavior_session table should be the same as the 'behavior_session_uuid' in mtrain which should also be the same as the 'session_uuid' field in the .pkl returned by 'get_behavior_stimulus_file()'.

get_age(*self*) → str

Return the age code of the subject (ie P123) :rtype: str

get_behavior_session_id(*self*) → int

Getter to be consistent with BehaviorOphysLimsApi.

get_behavior_stimulus_file(*self*) → str

Return the path to the StimulusPickle file for a session. :rtype: str

get_birth_date(*self*) → datetime.datetime

Returns the birth date of the animal. :rtype: datetime.date

get_date_of_acquisition(*self*) → datetime.datetime

Get the acquisition date of a behavior_session in UTC :rtype: datetime

get_driver_line(*self*) → List[str]

Returns the genotype name(s) of the driver line(s). :rtype: list

get_equipment_name(*self*) → str

Returns the name of the experimental rig. :rtype: str

get_foraging_id(*self*) → int

Get the foraging ID for the behavior session

get_full_genotype(*self*) → str

Return the name of the subject's genotype :rtype: str

get_mouse_id(*self*) → int

Returns the LabTracks ID :rtype: int

get_ophys_container_id(*self*) → Union[int, NoneType]

get_ophys_experiment_ids(*self*) → Union[List[int], NoneType]

get_ophys_session_id(*self*) → Union[int, NoneType]

get_reporter_line(*self*) → List[str]

Returns the genotype name(s) of the reporter line(s). :rtype: list

get_sex(*self*) → str

Returns sex of the animal (M/F) :rtype: str

allensdk.brain_observatory.behavior.session_apis.data_io.behavior_nwb_api module

allensdk.brain_observatory.behavior.session_apis.data_io.behavior_ophys_json_api module

allensdk.brain_observatory.behavior.session_apis.data_io.behavior_ophys_lims_api module

allensdk.brain_observatory.behavior.session_apis.data_io.behavior_ophys_nwb_api module

allensdk.brain_observatory.behavior.session_apis.data_io.ophys_lims_api module

Module contents

allensdk.brain_observatory.behavior.session_apis.data_transforms package

Submodules

allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms module

class allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_trans

Bases: *allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_base.BehaviorBase*

This class provides methods that transform data extracted from LIMS or JSON data sources into final data products necessary for populating a BehaviorSession.

get_behavior_session_id(*self*)
Returns the behavior_session_id associated with this experiment, if applicable.

get_extended_trials(*self*) → pandas.core.frame.DataFrame
Get extended trials from pkl file

Returns

pd.DataFrame A dataframe containing extended behavior trial information.

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 Two columns: "time", which contains the sync time of the licks that occurred in this session and "frame", the frame numbers of licks that occurred in this session

get_metadata(*self*) → allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata
Return metadata about the session. :rtype: BehaviorMetadata

get_monitor_delay(*self*) → float
Return monitor delay for behavior only sessions (in seconds)

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_running_acquisition_df(*self*, *lowpass=True*, *zscore_threshold=10.0*) → pandas.core.frame.DataFrame

Get running speed acquisition data from a behavior pickle file.

NOTE: Rebases timestamps with the self.get_stimulus_timestamps() method which varies between the BehaviorDataTransformer and the BehaviorOphysDataTransformer.

Parameters

lowpass: bool (default=True) Whether to apply a 10Hz low-pass filter to the running speed data.

zscore_threshold: float The threshold to use for removing outlier running speeds which might be noise and not true signal

Returns

pd.DataFrame

Dataframe with an index of timestamps and the following columns: "speed": computed running speed "dx": angular change, computed during data collection "v_sig": voltage signal from the encoder "v_in": the theoretical maximum voltage that the encoder

will reach prior to "wrapping". This should theoretically be 5V (after crossing 5V goes to 0V, or vice versa). In practice the encoder does not always reach this value before wrapping, which can cause transient spikes in speed at the voltage "wraps".

get_running_speed(*self*, *lowpass=True*) → pandas.core.frame.DataFrame

Get running speed using timestamps from self.get_stimulus_timestamps.

NOTE: Do not correct for monitor delay.

Returns pd.DataFrame index: timestamps speed : subject's running speeds (in cm/s)

get_stimulus_frame_rate(*self*) → float

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*) → Union[allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.S NoneType]

Get stimulus templates (movies, scenes) for behavior session.

Returns

StimulusTemplate or None if there are no images for the experiment

get_stimulus_timestamps(*self*) → numpy.ndarray

Get stimulus timestamps (vsyncs) from pkl file. Align to the (frame, time) points in the trial events.

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 no 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.session_apis.data_transforms.behavior_ophys_data_transforms module

class allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_

Bases: *allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms.BehaviorDataTransforms*, *allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base.BehaviorOphysBase*

This class provides methods that transform data extracted from LIMS or JSON data sources into final data products necessary for populating a BehaviorOphysExperiment

get_average_projection (*self*, *image_api=None*)
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_roi_ids (*self*)

get_cell_specimen_table (*self*)
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*)
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*)
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_events (*self*, *filter_scale: float = 2*, *filter_n_time_steps: int = 20*) → pandas.core.frame.DataFrame
Returns events in dataframe format

Parameters

filter_scale: float See filter_events_array for description

filter_n_time_steps: int See filter_events_array for description

See behavior_ophys_experiment.events for return type

get_eye_tracking (*self*, *z_threshold: float = 3.0*, *dilation_frames: int = 2*) → Union[pandas.core.frame.DataFrame, NoneType]
Gets corneal, eye, and pupil ellipse fit data

Parameters

z_threshold [float, optional] The z-threshold when determining which frames likely contain outliers for eye or pupil areas. Influences which frames are considered 'likely blinks'. By default 3.0

dilation_frames [int, optional]

Determines the number of additional adjacent frames to mark as

'likely_blink', by default 2.

Returns

Optional[pd.DataFrame] *_area *_center_x *_center_y *_height *_phi *_width likely_blink

where "*" can be "corneal", "pupil" or "eye"

Will return None if class attr _skip_eye_tracking is True.

get_eye_tracking_rig_geometry (*self*) → Union[dict, NoneType]
Get eye tracking rig metadata from behavior + ophys session.

Returns

dict Includes geometry of monitor, camera, LED

get_max_projection (*self*, *image_api=None*)
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*) → allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata.BehaviorOphysMetadata
Return metadata about the session. :rtype: BehaviorOphysMetadata

get_monitor_delay (*self*)
Return the monitor delay (in seconds)

get_motion_correction (*self*)
Get motion correction trace data.

Returns

pd.DataFrame A dataframe containing trace data used during motion correction computation.

get_ophys_experiment_id(*self*)
Returns the ophys_experiment_id for the instantiated BehaviorOphys Session (or BehaviorOphys data fetcher) if applicable.

get_ophys_session_id(*self*)
Returns the behavior + ophys_session_id associated with this experiment, if applicable.

get_ophys_timestamps(*self*)
Get optical physiology frame timestamps.

Returns

np.ndarray Timestamps associated with frames captured by the microscope.

get_raw_dff_data(*self*)

get_rewards(*self*)
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_roi_masks_by_cell_roi_id(*self, cell_roi_ids: Union[int, Iterable[int], NoneType] = None*)
Obtains boolean masks indicating the location of one or more ROIs in this session.

Parameters

cell_roi_ids [array-like of int, optional] ROI masks for these rois will be returned. The default behavior is to return masks for all rois.

Returns

result [xr.DataArray]

dimensions are:

- roi_id : which 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.

Notes

This method helps Allen Institute scientists to look at sessions that have not yet had cell specimen ids assigned. You probably want to use get_roi_masks instead.

get_stimulus_timestamps(*self*)
Return a numpy array of stimulus timestamps uncorrected for monitor delay (in seconds)

get_sync_data(*self*)

get_sync_licks(*self*)

Module contents

Module contents

allensdk.brain_observatory.behavior.stimulus_processing package

Submodules

allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates module

class allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.**StimulusIm**

Bases: object

Container class for image stimuli

name

class allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.**StimulusIm**

Bases: object

Factory for StimulusImage

static from_processed(*warped: numpy.ndarray*, *unwarped: numpy.ndarray*, *name: str*) → allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.StimulusImage

Creates a StimulusImage from processed input (usually nwb). Image has already been warped and preprocessed

from_unprocessed(*self*, *input_array: numpy.ndarray*, *name: str*) → allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.StimulusImage

Creates a StimulusImage from unprocessed input (usually pkl). Image needs to be warped and preprocessed

class allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.**StimulusTe**

Bases: object

Container class for a collection of image stimuli

image_names

image_set_name

images

items(*self*)

keys(*self*)

to_dataframe(*self*) → pandas.core.frame.DataFrame

values(*self*)

class allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.**StimulusTe**

Bases: object

Factory for StimulusTemplate

static from_processed(*image_set_name: str*, *image_attributes: List[dict]*, *unwarped: List[numpy.ndarray]*, *warped: List[numpy.ndarray]*) → allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.StimulusTemplate

Create StimulusTemplate from nwb or other processed input. Stimulus templates created this way DO NOT need to be processed to acquire unwarped versions of the images presented.

NOTE: The ordering of image_attributes, unwarped, and warped matter!

NOTE: Warped images display what was seen on a monitor by a subject. Unwarped images display a 'diagnostic' version of the stimuli to be presented.

Parameters

image_set_name [str]

The name of the image set. Example: Natural_Images_Lum_Matched_set_TRAINING_2017.07.14

image_attributes [List[dict]] A list of dictionaries containing image metadata. Must at least contain the key:

image_name

But will usually also contain: image_category, orientation, phase, spatial_frequency, image_index

unwarped [List[np.ndarray]] A list of unwarped image arrays

warped [List[np.ndarray]] A list of warped image arrays

Returns

StimulusTemplate A StimulusTemplate object

static from_unprocessed(*image_set_name: str, image_attributes: List[dict], images: List[numpy.ndarray]*) → allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.StimulusTemplate

Create StimulusTemplate from pkl or unprocessed input. Stimulus templates created this way need to be processed to acquire unwarped versions of the images presented.

NOTE: The ordering of image_attributes and images matter!

NOTE: Warped images display what was seen on a monitor by a subject. Unwarped images display a 'diagnostic' version of the stimuli to be presented.

Parameters

image_set_name [str]

The name of the image set. Example: Natural_Images_Lum_Matched_set_TRAINING_2017.07.14

image_attributes [List[dict]] A list of dictionaries containing image metadata. Must at least contain the key:

image_name

But will usually also contain: image_category, orientation, phase, spatial_frequency, image_index

images [List[np.ndarray]] A list of image arrays

Returns

StimulusTemplate A StimulusTemplate object

allensdk.brain_observatory.behavior.stimulus_processing.util module

`allensdk.brain_observatory.behavior.stimulus_processing.util.`**`convert_filepath_caseinsensit`**

`allensdk.brain_observatory.behavior.stimulus_processing.util.`**`get_image_set_name`**(*image_set_path str*) → str

Strips the stem from the image_set filename

Module contents

`allensdk.brain_observatory.behavior.stimulus_processing.`**`get_gratings_metadata`**(*stimuli: Dict, start_idx: int = 0*) → pandas.core.frame.Dat

This function returns the metadata for each unique grating that was presented during the experiment. If no gratings were displayed during this experiment it returns an empty dataframe with the expected columns. Parameters ——— stimuli:

The stimuli field (pkl['items']['behavior']['stimuli']) loaded from the experiment pkl file.

start_idx: The index to start index column

Returns

pd.DataFrame: DataFrame containing the unique stimuli presented during an experiment. The columns contained in this DataFrame are 'image_category', 'image_name', 'image_set', 'phase', 'spatial_frequency', 'orientation', and 'image_index'. This returns empty if no gratings were presented.

`allensdk.brain_observatory.behavior.stimulus_processing.`**`get_images_dict`**(*pkl*) → Dict

Gets the dictionary of images that were presented during an experiment along with image set metadata and the image specific metadata. This function uses the path to the image pkl file to read the images and their metadata from the pkl file and return this dictionary. Parameters ——— pkl: The pkl file containing the data for the stimuli presented during

experiment

Returns

Dict: A dictionary containing keys images, metadata, and image_attributes. These correspond to paths to image arrays presented, metadata on the whole set of images, and metadata on specific images, respectively.

`allensdk.brain_observatory.behavior.stimulus_processing.`**`get_stimulus_metadata`**(*pkl*) → pandas.core.frame.Dat

Gets the stimulus metadata for each type of stimulus presented during the experiment. The metadata is return for gratings, images, and omitted stimuli. Parameters ———- pkl: the pkl file containing the information about what stimuli were

presented during the experiment

Returns

pd.DataFrame: The dataframe containing a row for every stimulus that was presented during the experiment. The row contains the following data, image_category, image_name, image_set, phase, spatial_frequency, orientation, and image index.

allensdk.brain_observatory.behavior.stimulus_processing.**get_stimulus_presentations**(*data*, *stimulus_timestamps*) → pandas.core.frame.DataFrame

This function retrieves the stimulus presentation dataframe and renames the columns, adds a stop_time column, and set's index to stimulus_presentation_id before sorting and returning the dataframe. :param data: stimulus file associated with experiment id :param stimulus_timestamps: timestamps indicating when stimuli switched

during experiment

Returns stimulus_table: dataframe containing the stimuli metadata as well as what stimuli was presented

allensdk.brain_observatory.behavior.stimulus_processing.**get_stimulus_templates**(*pkl: dict*, *grating_images_dict: Union[dict, NoneType] = None*) → Union[allensdk.b… NoneType]

Gets images presented during experiments from the behavior stimulus file (*.pkl)

Parameters

pkl [dict] Loaded pkl dict containing data for the presented stimuli.

grating_images_dict [Optional[dict]] Because behavior pkl files do not contain image versions of grating stimuli, they must be obtained from an external source. The grating_images_dict is a nested dictionary where top level keys correspond to grating image names (e.g. 'gratings_0.0', 'gratings_270.0') as they would appear in table returned by get_gratings_metadata(). Sub-nested dicts are expected to have 'warped' and 'unwarped' keys where values are numpy image arrays of aforementioned warped or unwarped grating stimuli.

Returns

StimulusTemplate: StimulusTemplate object containing images that were presented during the experiment

`allensdk.brain_observatory.behavior.stimulus_processing.`**`get_visual_stimuli_df`**(*data*, *time*) → pandas.core.frame.DataFrame

This function loads the stimuli and the omitted stimuli into a dataframe. These stimuli are loaded from the input data, where the set_log and draw_log contained within are used to calculate the epochs. These epochs are used as start_frame and end_frame and converted to times by input stimulus timestamps. The omitted stimuli do not have a end_frame by design though there duration is always 250ms. :param data: the behavior data file :param time: the stimulus timestamps indicating when each stimuli is

displayed

Returns df: a pandas dataframe containing the stimuli and omitted stimuli that were displayed with their frame, end_frame, start_time, and duration

`allensdk.brain_observatory.behavior.stimulus_processing.`**`is_change_event`**(*stimulus_presentations: pandas.core.frame.DataFrame*) → pandas.core.series.Series

Returns whether a stimulus is a change stimulus A change stimulus is defined as the first presentation of a new image_name Omitted stimuli are ignored The first stimulus in the session is ignored

:param stimulus_presentations The stimulus presentations table

Returns is_change: pd.Series indicating whether a given stimulus is a change stimulus

`allensdk.brain_observatory.behavior.stimulus_processing.`**`load_pickle`**(*pstream*)

`allensdk.brain_observatory.behavior.stimulus_processing.`**`unpack_change_log`**(*change*)

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.**get_behavior_monitoring**(*dataset: allensdk.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: allensdk.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: allensdk.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: allensdk.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: allensdk.brain_observatory.sync_datase*

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: allensdk.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_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: allensdk.brain_observatory.sync_dataset.Dataset*, *permissive: bool = False*) → Union[numpy.ndarray, NoneType]

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 KeyError 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_behavior_nwb package

Module contents

allensdk.brain_observatory.behavior.write_nwb package

Subpackages

allensdk.brain_observatory.behavior.write_nwb.extensions package

Subpackages

allensdk.brain_observatory.behavior.write_nwb.extensions.event_detection package

Submodules

allensdk.brain_observatory.behavior.write_nwb.extensions.event_detection.extension_builder module

allensdk.brain_observatory.behavior.write_nwb.extensions.event_detection.ndx_ophys_events module

Module contents

allensdk.brain_observatory.behavior.write_nwb.extensions.stimulus_template package

Submodules

allensdk.brain_observatory.behavior.write_nwb.extensions.stimulus_template.extension_builder module

allensdk.brain_observatory.behavior.write_nwb.extensions.stimulus_template.ndx_stimulus_template module

Module contents

Module contents

Module contents

Submodules

allensdk.brain_observatory.behavior.behavior_ophys_analysis module

allensdk.brain_observatory.behavior.behavior_ophys_experiment module

allensdk.brain_observatory.behavior.behavior_ophys_session module

allensdk.brain_observatory.behavior.behavior_session module

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 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**(*correc false_a aborte slid- ing_wi*

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.event_detection module

allensdk.brain_observatory.behavior.event_detection.**filter_events_array**(*arr: numpy.ndarray, scale: float = 2, n_time_steps: int = 20*) → numpy.ndarray

Convolve the trace array with a 1d causal half-gaussian filter to smooth it for visualization

Uses a halfnorm distribution as weights to the filter

Modified from initial implementation by Nick Ponvert

Parameters

arr: np.ndarray Trace matrix of dimension n traces x n frames

scale: float std deviation of halfnorm distribution

n_time_steps: int number of time steps to use for the convolution operation

Returns

np.ndarray: Output of the convolution operation

allensdk.brain_observatory.behavior.eye_tracking_processing module

`allensdk.brain_observatory.behavior.eye_tracking_processing.`**`compute_circular_area`**(*df_row: pandas.core.seri* → 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^2.

`allensdk.brain_observatory.behavior.eye_tracking_processing.`**`compute_elliptical_area`**(*df_row: pandas.core.s* → 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^2 allensdk.brain_observatory.behavior.eye_tracking_processing.**determine_likely_blinks**(*eye_areas: pandas.core.s pupil_are pandas.core.s outliers: pandas.core.s dilation_fram int = 2*) → pandas.core.s

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.D z_threshold: float*) → pandas.core.series.Se

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 path-lib.Path*) → pan-das.core.fram

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_da pan-das.co frame_ pan-das.co z_thres float = 3.0*, *di-la-tion_fr int = 2*) → pan-das.co

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.mtrain module

class allensdk.brain_observatory.behavior.mtrain.**ExtendedTrialSchema**(**, only: Union[Sequence[str], Set[str], None] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict[KT, VT]] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = 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: Optional[str] = None, **kwargs*)

Bases: `marshmallow.fields.Date`

class allensdk.brain_observatory.behavior.mtrain.**FriendlyDateTime**(*format: Optional[str] = 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 annotatations:

- 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: Dict*, *timestamps: numpy.ndarray*) → pandas.core.frame.DataFrame

Construct and return a pandas DataFrame containing reward data for this session

data: Dict The dict that results from reading the stimulus pickle file associated with the session

timestamps: np.ndarray[1d] A numpy array of timestamps associated with the stimulus frames in this session. timestamps[ii] is the clock time of the iith frame.

Returns

pd.DataFrame containing the data associated with rewards given in this session

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**(*angular_speed: numpy.ndarray*) → numpy.ndarray

Takes the angular speed (radians/s) at each step in radians, and computes the linear speed in cm/s.

Parameters

angular_speed: np.ndarray (1d) 1d array of angular speed in radians/s

Returns

——-

np.ndarray (1d) Linear speed in cm/s at each time point.

allensdk.brain_observatory.behavior.running_processing.**get_running_df**(*data*, *time: numpy.ndarray*, *lowpass: bool = True*, *zscore_threshold=10.0*)

Given the data from the behavior 'pkl' file object and a 1d array of timestamps, compute the running speed. Returns a dataframe with the raw voltage data as well as the computed speed at each timestamp. By default, the running speed is filtered with a 10 Hz Butterworth lowpass filter to remove artifacts caused by the rotary encoder.

Parameters

data Deserialized 'behavior pkl' file data

time: np.ndarray (1d) Timestamps for running data measurements

lowpass: bool (default=True) Whether to apply a 10Hz low-pass filter to the running speed data.

zscore_threshold: float The threshold to use for removing outlier running speeds which might be noise and not true signal

Returns

pd.DataFrame Dataframe with an index of timestamps and the following columns:

"speed": computed running speed "dx": angular change, computed during data collection "v_sig": voltage signal from the encoder "v_in": the theoretical maximum voltage that the encoder

will reach prior to "wrapping". This should theoretically be 5V (after crossing 5V goes to 0V, or vice versa). In practice the encoder does not always reach this value before wrapping, which can cause transient spikes in speed at the voltage "wraps".

The raw data are provided so that the user may compute their own speed from source, if desired.

Notes

Though the angular change is available in the raw data (key="dx"), this method recomputes the angular change from the voltage signal (key="vsig") due to very specific, low-level artifacts in the data caused by the encoder. See method docstrings for more detailed information. The raw data is included in the final output in case the end user wants to apply their own corrections and compute running speed from the raw source.

allensdk.brain_observatory.behavior.schemas module

class allensdk.brain_observatory.behavior.schemas.**BehaviorMetadataSchema**(*, *only: Union[Sequence[str], Set[str], None] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict[KT, VT]] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None*)

Bases: *allensdk.brain_observatory.behavior.schemas.RaisingSchema*

This schema contains metadata pertaining to behavior.

neurodata_doc = 'Metadata for behavior and behavior + ophys experiments'

neurodata_skip = {'date_of_acquisition'}

neurodata_type = 'BehaviorMetadata'

neurodata_type_inc = 'LabMetaData'

opts = <marshmallow.schema.SchemaOpts object>

class allensdk.brain_observatory.behavior.schemas.**BehaviorTaskParametersSchema**(**, only: Union[Sequence[str], Set[str], None] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict[KT, VT]] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None*)

Bases: *allensdk.brain_observatory.behavior.schemas.RaisingSchema*

This schema encompasses task parameters used for behavior or ophys + behavior.

neurodata_doc = 'Metadata for behavior or behavior + ophys task parameters'

neurodata_type = 'BehaviorTaskParameters'

neurodata_type_inc = 'LabMetaData'

opts = <marshmallow.schema.SchemaOpts object>

class allensdk.brain_observatory.behavior.schemas.**CompleteOphysBehaviorMetadataSchema**(**, only: Union[Set[str], None] = None, exclude: Union[Set[str] = (), many: bool = False, context: Optional[VT]] = None, load_o Union[Set[str] = (), dump_ Union[Set[str] = (), partial: Union[Sequence Set[str] = False, unknown Optional[= None*)

Bases: *allensdk.brain_observatory.behavior.schemas.OphysBehaviorMetadataSchema*, *allensdk.brain_observatory.behavior.schemas.SubjectMetadataSchema*

This schema combines fields from behavior, ophys, and subject schemas. Metadata info is passed by the behavior+ophys session in a combined lump containing all the field types.

opts = <marshmallow.schema.SchemaOpts object>

class allensdk.brain_observatory.behavior.schemas.**EyeTrackingRigGeometry**(**, only: Union[Sequence[str], Set[str], None] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict[KT, VT]] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None*)

Bases: *allensdk.brain_observatory.behavior.schemas.RaisingSchema*

Eye tracking rig geometry

opts = <marshmallow.schema.SchemaOpts object>

class allensdk.brain_observatory.behavior.schemas.**NwbOphysMetadataSchema**(**, only: Union[Sequence[str], Set[str], None] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict[KT, VT]] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None*)

Bases: *allensdk.brain_observatory.behavior.schemas.RaisingSchema*

This schema contains fields that will be stored in pyNWB base classes pertaining to optical physiology.

opts = <marshmallow.schema.SchemaOpts object>

class allensdk.brain_observatory.behavior.schemas.**OphysBehaviorMetadataSchema**(**, only: Union[Sequence[s Set[str], None] = None, exclude: Union[Sequence[s Set[str]] = (), many: bool = False, context: Optional[Dict[KT, VT]] = None, load_only: Union[Sequence[s Set[str]] = (), dump_only: Union[Sequence[s Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None*)

Bases: *allensdk.brain_observatory.behavior.schemas.BehaviorMetadataSchema*, *allensdk.brain_observatory.behavior.schemas.OphysMetadataSchema*

This schema contains fields pertaining to ophys+behavior. It is used as a template for generating our custom NWB behavior + ophys extension.

neurodata_doc = 'Metadata for behavior + ophys experiments'

neurodata_skip = {'date_of_acquisition', 'emission_lambda', 'excitation_lambda', 'indi

neurodata_type = 'OphysBehaviorMetadata'

neurodata_type_inc = 'BehaviorMetadata'

opts = <marshmallow.schema.SchemaOpts object>

class allensdk.brain_observatory.behavior.schemas.**OphysEyeTrackingRigMetadataSchema**(*, *only: Union[Se Set[str], None] = None, exclude: Union[Se Set[str]] = (), many: bool = False, context: Optional[Di VT]] = None, load_only Union[Se Set[str]] = (), dump_on Union[Se Set[str]] = (), partial: Union[bo Sequence[st Set[str]] = False, unknown: Optional[str = None*)

Bases: *allensdk.brain_observatory.behavior.schemas.RaisingSchema*

This schema encompasses metadata for ophys experiment rig

neurodata_doc = 'Metadata for ophys experiment rig'

neurodata_type = 'OphysEyeTrackingRigMetadata'

neurodata_type_inc = 'NWBDataInterface'

opts = <marshmallow.schema.SchemaOpts object>

class allensdk.brain_observatory.behavior.schemas.**OphysMetadataSchema**(**, only: Union[Sequence[str], Set[str], None] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict[KT, VT]] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None*)

Bases: `allensdk.brain_observatory.behavior.schemas.NwbOphysMetadataSchema`

This schema contains metadata pertaining to optical physiology (ophys).

opts = <marshmallow.schema.SchemaOpts object>

class allensdk.brain_observatory.behavior.schemas.**RaisingSchema**(**, only: Union[Sequence[str], Set[str], None] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict[KT, VT]] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None*)

Bases: `marshmallow.schema.Schema`

class Meta

Bases: `object`

unknown = 'raise'

opts = <marshmallow.schema.SchemaOpts object>

class allensdk.brain_observatory.behavior.schemas.**SubjectMetadataSchema**(**, only: Union[Sequence[str], Set[str], None] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict[KT, VT]] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None*)

Bases: *allensdk.brain_observatory.behavior.schemas.RaisingSchema*

This schema contains metadata pertaining to a subject in either a behavior or behavior + ophys experiment.

neurodata_doc = 'Metadata for an AIBS behavior or behavior + ophys subject'

neurodata_skip = {'age_in_days', 'genotype', 'sex', 'subject_id'}

neurodata_type = 'BehaviorSubject'

neurodata_type_inc = 'Subject'

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*, *detect_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.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_response_latency_list**(*trials: pandas.co re- sponse float*) → List

per trial, detemines a response latency

Parameters

trials: pd.DataFrame contains columns "lick_times" and "change_times"

response_window_start: float [seconds] relative to the non-display-lag-compensated presentation of the change-image

Returns

response_latency_list: list len() = trials.shape[0] value is 'inf' if there are no valid licks in the trial

allensdk.brain_observatory.behavior.trials_processing.**calculate_reward_rate**(*response_latency=No starttime=None*, *window=0.75*, *trial_window=25*, *initial_trials=10*)

allensdk.brain_observatory.behavior.trials_processing.**calculate_reward_rate_fix_nans**(*trials: pandas.core re- sponse_ float*) → numpy.r

per trial, detemines the reward rate, replacing infs with nans

Parameters

trials: pd.DataFrame contains columns "lick_times", "change_times", and "start_time"

response_window_start: float [seconds] relative to the non-display-lag-compensated presentation of the change-image

Returns

reward_rate: np.ndarray size = trials.shape[0] value is nan if calculate_reward_rate evaluates to 'inf'

`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.`**`construct_rolling_performance_df`**`(`*trials: pandas.c re- spon ses- sion_* → pandas.c

Return a DataFrame containing trial by trial behavior response performance metrics.

Parameters

trials: pd.DataFrame contains columns "lick_times", "change_times", and "start_time"

response_window_start: float [seconds] relative to the non-display-lag-compensated presentation of the change-image

session_type: str used to check if this was a passive session

Returns

pd.DataFrame

A pandas DataFrame containing:

trials_id [index]: Index of the trial. All trials, including aborted trials, are assigned an index starting at 0 for the first trial.

reward_rate: Rewards earned in the previous 25 trials, normalized by the elapsed time of the same 25 trials. Units are rewards/minute.

hit_rate_raw: Fraction of go trials where the mouse licked in the response window, calculated over the previous 100 non-aborted trials. Without trial count correction applied.

hit_rate: Fraction of go trials where the mouse licked in the response window, calculated over the previous 100 non-aborted trials. With trial count correction applied.

false_alarm_rate_raw: Fraction of catch trials where the mouse licked in the response window, calculated over the previous 100 non-aborted trials. Without trial count correction applied.

false_alarm_rate: Fraction of catch trials where the mouse licked in the response window, calculated over the previous 100 non-aborted trials. Without trial count correction applied.

rolling_dprime: d prime calculated using the rolling hit_rate and rolling false_alarm _rate.

allensdk.brain_observatory.behavior.trials_processing.**create_extended_trials**(*trials=None*, *metadata=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*, *window=3.5*)

allensdk.brain_observatory.behavior.trials_processing.**get_change_time_frame_response_latenc**

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_bounds**(*trial_log: List*) → List

Adjust trial boundaries from a trial_log so that there is no dead time between trials.

Parameters

trial_log: list The trial_log read in from the well known behavior stimulus pickle file

Returns

list Each element in the list is a tuple of the form (start_frame, end_frame) so that the ith element of the list gives the start and end frames of the ith trial. The endframe of the last trial will be -1, indicating that it should map to the last timestamp in the session

allensdk.brain_observatory.behavior.trials_processing.**get_trial_image_names**(*trial*, *stimuli*) → Dict[str, str]

Gets the name of the stimulus presented at the beginning of the trial and what is it changed to at the end of the trial. Parameters ———- trial: A trial in a behavior ophys session stimuli: The stimuli presentation log for the behavior session

Returns

A dictionary indicating the starting_stimulus and what the stimulus is changed to.

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: dict*, *licks: List[float]*, *go: bool*, *catch: bool*, *auto_rewarded: bool*, *hit: bool*, *false_alarm: bool*, *aborted: bool*, *timestamps: numpy.ndarray*, *monitor_delay: float*)

Extract a dictionary of trial timing data. See trial_data_from_log for a description of the trial types.

Parameters

event_dict: dict Dictionary of trial events in the well-known *pkl* file

licks: List[float] list of lick timestamps, from the *get_licks* response for the BehaviorOphysExperiment.api.

go: bool True if "go" trial, False otherwise. Mutually exclusive with *catch*.

catch: bool True if "catch" trial, False otherwise. Mutually exclusive with *go*.

auto_rewarded: bool True if "auto_rewarded" trial, False otherwise.

hit: bool True if "hit" trial, False otherwise

false_alarm: bool True if "false_alarm" trial, False otherwise

aborted: bool True if "aborted" trial, False otherwise

timestamps: np.ndarray[1d] Array of ground truth timestamps for the session (sync times, if available)

monitor_delay: float The monitor delay in seconds associated with the session

Returns

dict

start_time: float The time the trial started (in seconds elapsed from recording start)

stop_time: float The time the trial ended (in seconds elapsed from recording start)

trial_length: float Duration of the trial in seconds

response_time: float The response time, for non-aborted trials. This is equal to the first lick in the trial. For aborted trials or trials without licks, *response_time* is NaN.

change_frame: int The frame number that the stimulus changed

change_time: float The time in seconds that the stimulus changed

response_latency: float or None The time in seconds between the stimulus change and the animal's lick response, if the trial is a "go", "catch", or "auto_rewarded" type. If the animal did not respond, return *float("inf")*. In all other cases, return None.

Notes

The following parameters are mutually exclusive (exactly one can be true):

hit, miss, false_alarm, aborted, auto_rewarded

`allensdk.brain_observatory.behavior.trials_processing.`**`get_trials_from_data_transform`**`(`*input_tr* → pandas.core

Create and return a pandas DataFrame containing data about the trials associated with this session

Parameters

input_transform: An instantiation of a class that inherits from either BehaviorDataTransform or BehaviorOphysDataTransform. This object will be used to get at the data needed by this method to create the trials dataframe.

Returns

pd.DataFrame A dataframe containing data pertaining to the trials that make up this session

Notes

The input_transform object must have the following methods:

input_transform._behavior_stimulus_file Which returns the dict resulting from reading in this session's stimulus_data pickle file

input_transform.get_rewards Which returns a dataframe containing data about rewards given during this session, i.e. the output of allensdk/brain_observatory/behavior/rewards_processing.get_rewards

input_transform.get_licks Which returns a dataframe containing the columns *time* and *frame* denoting the time (in seconds) and frame number at which licks occurred during this session

input_transform.get_stimulus_timestamps Which returns a numpy.ndarray of timestamps (in seconds) associated with the frames presented in this session.

input_transform.get_monitor_delay Which returns the monitory delay (in seconds) associated with the experimental rig

`allensdk.brain_observatory.behavior.trials_processing.`**`get_trials_v0`**(*data*, *time*)

`allensdk.brain_observatory.behavior.trials_processing.`**`local_time`**(*iso_timestamp*, *timezone=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

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 nbi*

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] Minimun 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_barcodes 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_barcodes probe_times probe_barco 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_intervals**

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*, *master_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.`**`BarcodeSyncI`**

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*, *local_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

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, NoneType] = 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*, *ecephys_session_ids: Union[~ArrayLike, NoneType] = None*, *session_types: Union[~ArrayLike, NoneType] = None*) → pandas.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

allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_lims_api module

allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_warehouse_api module

allensdk.brain_observatory.ecephys.ecephys_project_api.http_engine module

allensdk.brain_observatory.ecephys.ecephys_project_api.rma_engine module

allensdk.brain_observatory.ecephys.ecephys_project_api.utilities module

Module contents

allensdk.brain_observatory.ecephys.ecephys_session_api package

Submodules

allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_nwb1_session_api module

allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_nwb_session_api module

allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api module

class allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.**EcephysSes**

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*) → Union[pandas.core.frame.DataFrame, NoneType]

get_rig_metadata (*self*) → Union[dict, NoneType]

get_running_speed (*self*) → allensdk.brain_observatory.running_speed.RunningSpeed

get_screen_gaze_data (*self*, *include_filtered_data=False*) → Union[pandas.core.frame.DataFrame, NoneType]

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*, *timestamps_path*, *total_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_frequen* *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_of* *channel_stride, channel_order, noisy_channels=* *dtype=float64), remove_noisy_cha* *reference_channels=a* *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 refence 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`**(*timestamps, subsampling_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

Submodules

allensdk.brain_observatory.ecephys.nwb.ecephys_nwb_extension_builder module

Module contents

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

allensdk.brain_observatory.ecephys.stimulus_analysis.drifting_gratings module

allensdk.brain_observatory.ecephys.stimulus_analysis.flashes module

allensdk.brain_observatory.ecephys.stimulus_analysis.natural_movies module

allensdk.brain_observatory.ecephys.stimulus_analysis.natural_scenes module

allensdk.brain_observatory.ecephys.stimulus_analysis.receptive_field_mapping module

allensdk.brain_observatory.ecephys.stimulus_analysis.static_gratings module

allensdk.brain_observatory.ecephys.stimulus_analysis.stimulus_analysis module

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*
f
s
e
c
l

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_fran
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.ndarrray] 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 calculcate 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_*
swee
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, stimulus_tabl spontaneous_a sort_key block_k index_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_tabler [function] A function which takes a single stimulus dictionary as its argument and returns a stimulus table dataframe.

spontaneous_activity_tabler [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_activit**

Fills in frame gaps in a set of stimulus tables. Suitable for use as the spontaneous_activity_tabler 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 spontenous activity sweeps.

allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes.**read_stimulus_name_from_**

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_m`**

`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_ma* *ig-nore_case*

`allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities.`**`map_stimulus_names`**(*table*, *name_* *stim_c*

Applies a mappting 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_numbe**

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_duratio**

allensdk.brain_observatory.ecephys.stimulus_table.output_validation.**validate_epoch_order**(*ta*
ti
'E

allensdk.brain_observatory.ecephys.stimulus_table.output_validation.**validate_max_spontaneou**

allensdk.brain_observatory.ecephys.stimulus_table.stimulus_parameter_extraction module

allensdk.brain_observatory.ecephys.stimulus_table.stimulus_parameter_extraction.**extract_cor**

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 paramater 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 averege 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, xlabel='time relative to stimulus onset (s)', ylabel='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 0x7fb14e570ed0>, title='spike raster', cycle_colors=False*)

allensdk.brain_observatory.ecephys.write_nwb package

Module contents

Submodules

allensdk.brain_observatory.ecephys.ecephys_project_cache module

allensdk.brain_observatory.ecephys.ecephys_session module

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 0x7fb11fecf950>*)

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, cycle=60*)

allensdk.brain_observatory.ecephys.stimulus_sync.**fix_unexpected_edges**(*pd_times, ndevs=10, cycle=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=0.016666666666666, num_frames=5*)

allensdk.brain_observatory.ecephys.stimulus_sync.**trimmed_stats**(*data, pctiles=(10, 90)*)

Module contents

allensdk.brain_observatory.ecephys.**get_unit_filter_value**(*key, pop=True, replace_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

Subpackages

allensdk.brain_observatory.nwb.eye_tracking package

Submodules

allensdk.brain_observatory.nwb.eye_tracking.extension_builder module

allensdk.brain_observatory.nwb.eye_tracking.ndx_ellipse_eye_tracking module

Module contents

Submodules

allensdk.brain_observatory.nwb.behavior_ophys_nwb_extension_builder module

allensdk.brain_observatory.nwb.metadata module

allensdk.brain_observatory.nwb.nwb_api module

allensdk.brain_observatory.nwb.nwb_utils module

allensdk.brain_observatory.nwb.schemas module

Module contents

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.chisquarerf module

`allensdk.brain_observatory.receptive_field_analysis.chisquarerf.`**`NLL_to_pvalue`**(*NLLs*, *log_base=10.0*) allensdk.brain_observatory.receptive_field_analysis.chisquarerf.**build_trial_matrix**(*LSN_templ 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.chisquarerf.**chi_square_binary**(*events, LSN_templa*

allensdk.brain_observatory.receptive_field_analysis.chisquarerf.**chi_square_within_mask**(*exclu event tri- als_p*

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.chisquarerf.**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=2*, *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 * 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**(*respons*, *trial_ma*

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_pixe**

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.chisquarerf.`**`get_peak_significance`**(*chi_sq LSN_t al- pha=0*

`allensdk.brain_observatory.receptive_field_analysis.chisquarerf.`**`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.chisquarerf.`**`locate_median`**(*y, x*)

`allensdk.brain_observatory.receptive_field_analysis.chisquarerf.`**`pvalue_to_NLL`**(*p_values, max_NLL=10.0*)

`allensdk.brain_observatory.receptive_field_analysis.chisquarerf.`**`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, stim- 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_di**

allensdk.brain_observatory.receptive_field_analysis.fit_parameters.**compute_distance**(*center_on, center_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.**GaussianFitErro**

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, center_x, center_y, width_x, width_y, rotation*)

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_fiel**

allensdk.brain_observatory.receptive_field_analysis.receptive_field.**compute_receptive_fiel** allensdk.brain_observatory.receptive_field_analysis.receptive_field.**events_to_pvalues_no_fo**

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_fr**

allensdk.brain_observatory.receptive_field_analysis.receptive_field.**write_receptive_field_t**

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**(*lis*

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*, *stimulus*)

allensdk.brain_observatory.receptive_field_analysis.utilities.**get_A_blur**(*data*, *stimulus*)

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*, *number_of_shuffle* *response_detect* allensdk.brain_observatory.receptive_field_analysis.utilities.**get_sparse_noise_epoch_mask_**

allensdk.brain_observatory.receptive_field_analysis.utilities.**smooth**(*x*, *window_len=11*, *window='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 begining 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**(

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=*, *cmap='m*

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=N*, *cmap=Non*

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**

allensdk.brain_observatory.receptive_field_analysis.visualization.**plot_rts_blur_summary**(*rf_d*, *ax_*, *ax_*, *ax_*, *cma*

allensdk.brain_observatory.receptive_field_analysis.visualization.**plot_rts_summary**(*rf_data*, *ax_on*, *ax_off*, *ax_cbar=N*, *cmap='ma*

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: pathlib.Path, sync_line_label_keys: Tuple[str, ...], trim_after_spike: bool = True*) → pandas.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.

trim_after_spike [bool = True] If True, will call trim_discontiguous_times on the frame times before returning them, which will detect any spikes in the data and remove all elements for the list which come after the spike.

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, threshold=100*)

allensdk.brain_observatory.visualization package

Module contents

allensdk.brain_observatory.visualization.**plot_running_speed**(*timestamps, values, start_index=0, stop_index=None, step=1, ylabel='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**(**, load_default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, dump_default: Any = <marshmallow.missing>, default: Any = <marshmallow.missing>, data_key: Optional[str] = None, attribute: Optional[str] = None, validate: Union[Callable[[Any], Any], Iterable[Callable[[Any], Any]], None] = None, required: bool = False, allow_none: Optional[bool] = None, load_only: bool = False, dump_only: bool = False, error_messages: Optional[Dict[str, str]] = None, metadata: Optional[Mapping[str, Any]] = None, **additional_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**(*, *load_default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, dump_default: Any = <marshmallow.missing>, default: Any = <marshmallow.missing>, data_key: Optional[str] = None, attribute: Optional[str] = None, validate: Union[Callable[[Any], Any], Iterable[Callable[[Any], Any]], None] = None, required: bool = False, allow_none: Optional[bool] = None, load_only: bool = False, dump_only: bool = False, error_messages: Optional[Dict[str, str]] = None, metadata: Optional[Mapping[str, Any]] = None, **additional_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: Union[Sequence[str], Set[str], None] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict[KT, VT]] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = 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, allow_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, input_schema, output_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.**BrainObservatoryAnalysisE**

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 resposes

allensdk.brain_observatory.brain_observatory_plotting.**plot_lsn_traces**(*lsn*, *save_dir*, *suffix=''*)

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*, *options_per_column*)

allensdk.brain_observatory.chisquare_categorical.**chisq_from_stim_table**(*stim_table*, *columns*, *mean_sweep_events*, *num_shuffles=1000*, *verbose=False*) allensdk.brain_observatory.chisquare_categorical.**compute_chi**(*observed*, *expected*)

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*, *verbose=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 0x7fb1200d2f10>*)

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 0x7fb1200d2f10>*, *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*, *angle_start=0*, *circle_scale=1.1*, *inner_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*, *angle_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 0x7fb1200d2f10>*, *mean_cmap=<matplotlib.colors.LinearSegmentedColormap object at 0x7fb1202bb1d0>*, *norm=None*)

show_arrow(*self*, *color=None*)

allensdk.brain_observatory.circle_plots.**add_angle_labels**(*ax*, *angles*, *labels*, *radius*, *color=None*, *fontdict=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*, *endpoint=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*, *offset=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: numpy.ndarray*, *stack: numpy.ndarray*, *masks: numpy.ndarray*, *max_block_size: int = 1000*) → Tuple[numpy.ndarray, list]

Demix traces of potentially overlapping masks extraced from a single 2p recording.

Parameters

- **raw_traces** – 2d array of traces for each mask, of dimensions (n, t), where *t* is the number of time points and *n* is the number of masks.
- **stack** – 3d array representing a 1p recording movie, of dimensions (t, H, W) or corresponding hdf5 dataset.
- **masks** – 3d array of binary roi masks, of shape (n, H, W), where *n* is the number of masks, and HW are the dimensions of an individual frame in the movie *stack*.

Max_block_size int representing maximum number of movie frames to read at a time (-1 for full length *t* of *stack*) (the default is 1000)

Returns Tuple of demixed traces and whether each frame was skipped in the demixing calculation.

allensdk.brain_observatory.demixer.**find_negative_baselines**(*trace*)

allensdk.brain_observatory.demixer.**find_negative_transients_threshold**(*trace*, *window=500*, *length=10*, *std_devs=3*)

allensdk.brain_observatory.demixer.**find_zero_baselines**(*traces*)

allensdk.brain_observatory.demixer.**identify_valid_masks**(*mask_array*)

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*, *kernelsize*, *y*)

Compute the windowed mode of an array. A running mode is initialized with a histogram of values over the initial kernelsize/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

kernelsize [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*, *stimulus=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 0x7fb11ff3ff50>*)

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.LinearSegm object at 0x7fb11ff3ff10>, edgecolor='', 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, outline_color='#cccccc', scalebar=True*) allensdk.brain_observatory.observatory_plots.**plot_representational_similarity**(*rs, dims=None, dim_labels=None, colors=None, dim_order=None, labels=True*)

allensdk.brain_observatory.observatory_plots.**plot_selectivity_cumulative_histogram**(*sis, xlabel, 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, labels, colors, 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*)

Conveninece 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, ithe 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*)

Conveninece 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 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 occuring 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.**compare_session_fields** (*x1: Any*, *x2: Any*, *err_msg=''*)
Helper function to compare if two fields (attributes) from a Session object are equal to one another.

Parameters

x1 [Any] The field from the first session to compare

x2 [Any] The corresponding field from the second session to compare

err_msg [str, optional] The error message to display if two compared fields do not equal one another, by default "" (an empty string)

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.session_api_utils.**sessions_are_equal** (*A*, *B*, *reraise=False*) → bool
Check if two Session objects are equal (have same methods and attributes).

Parameters

A [Session A] The first session to compare

B [Session B] The second session to compare

reraise [bool, optional] Whether to reraise when encountering an Assertion or AttributeError, by default False

Returns

bool Whether the two sessions are equal to one another.

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)
- ptest_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 0x7fb11ff3ff50>*)

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 stimulu. 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*, *temporal_frequency=None*, ***kwargs*)

allensdk.brain_observatory.stimulus_info.**lsn_coordinate_to_monitor_coordinate** (*lsn_coordinate*, *monitor_shape*, *stimulus_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 stim- 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, target_stimulus_type, monitor_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, template_shape, display_mask=None, threshold=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_coordinat* *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**(*nat* *mon-i-tor_*

allensdk.brain_observatory.stimulus_info.**natural_scene_coordinate_to_monitor_coordinate**(*nat* *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*, *allow_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: derricw

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()
```

The sync file documentation from MPE can be found at sharepoint > Instrumentation > Shared Documents > Sync_line_labels_discussion_2020-01-27-.xlsx # NOQA E501 Direct link: https://alleninstitute.sharepoint.com/:x:/s/Instrumentation/ES2bi1xJ3E9NupX-zQeXTlYBS2mVVySycfbCQhsD_jPMUw?e=Z9jCwH

BEHAVIOR_TRACKING_KEYS = ('beh_frame_received', 'cam1_exposure', 'behavior_monitoring'

DEPRECATED_KEYS = {}

EYE_TRACKING_KEYS = ('eye_frame_received', 'cam2_exposure', 'eyetracking', 'eye_tracki

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'*, *halp='Run application.'*, *default_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 paramenters.

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='.'*, *version=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] Subtitutions 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: Callable, wrappers: Iterable[T_co] = (), settable: bool = False, *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

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 stimulus_info.LOCALLY_SPARSE_NOISE_4DEG stimulus_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 flourescence 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

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 *lfu_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.warehouse_cache.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*, *manifest_file=None*, *ccf_version=None*, *base_uri=None*, *version=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 experients. 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 unionzes 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, optiona] 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 isometric 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 experients. 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 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 exracting 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 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 itksnap 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 curried 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.warehouse_cache.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 quantative 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 mode 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 mode 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 morpyhology, 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, baseline_interval=0.1, baseline_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/u

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*, *baseline_interval=0.1*, *baseline_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 [nparray 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_fi_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*, *dvdt=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)

dvdt [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)]

dvdt [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*, *interval_start*, *interval_end*, *filter=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*, *dvdt=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)]

dvdt [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)]

dvdt [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 [Coeffients of equation y0 + a * exp(-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*, *subthresh_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

arrary 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_di*
res-
o-
lu-
tion=None
base_uri=l

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.**MouseConnectivityApiP**

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

arrary 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

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.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

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.**db_connection_creator**(*credentials: Union[allensdk.core.authentication.DbCredentials, NoneType] = None, fallback_credentials: Union[dict, NoneType] = None*) → allensdk.internal.api.PostgresQueryMixin

Create a db connection using credentials. If credentials are not provided then use fallback credentials (which attempt to read from shell environment variables).

Note: Must provide one of either 'credentials' or 'fallback_credentials'. If both are provided, 'credentials' will take precedence.

Parameters

credentials [Optional[DbCredentials], optional] User specified credentials, by default None

fallback_credentials [dict] Fallback credentials to use for creating the DB connection in the case that no 'credentials' are provided, by default None.

Fallback credentials will attempt to get db connection info from shell environment variables.

Some examples of environment variables that fallback credentials will try to read from can be found in allensdk.core.auth_config.

Returns

PostgresQueryMixin A DB connection instance which can execute queries to the DB specified by credentials or fallback_credentials.

Raises

RuntimeError If neither 'credentials' nor 'fallback_credentials' were provided.

`allensdk.internal.api.`**`psycopg2_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*, *azimuth_phase*, *x=None*, *y=None*, *width=None*, *height=None*, *mask=None*, *altitude_scale=0.322*, *azimuth_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*, *without_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*, *window=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, savefile=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, savefile*)

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]), monitor_rotations=array([0., 0., 0.]), led_position=array([25.89 - 6.12, 3.21]), camera_position=array([13., 0., 0.]), camera_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_paramaeters [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_paramaeters [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_paramaeters [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**(*objec*
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*, *ffmpeg_bin='ffmpeg'*, *num_frames=None*, *block_size=1*, *cache_frames=False*, *process_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*, *ffmpeg_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 kernal 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_idxs*, *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_idxs*)

intersection(*self*, *mask_idxs*)

intersection_size(*self*, *mask_idxs*)

mask(*self*, *mask_idx*)

mask_is_union_of_set(*self*, *mask_idx*, *set_idxs*, *threshold*)

overlap_fraction(*self*, *idx0*, *idx1*)

size(*self*, *mask_idx*)

union(*self*, *mask_idxs*)

union_size(*self*, *mask_idxs*)

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_file data_hd auxiliary_hd frame_n compression='g compression_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*, *json_meta*, *use_memmap=False*)

Load a frame of a multi-frame raw file.

allensdk.internal.brain_observatory.ophys_session_decomposition.**open_view_on_binary**(*file_like*, *dtype=<c 'numpy.u mode='r' offset=0*, *shape=N order='C'*, *strides=N*

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*, *offset*, *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**(*crit*

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*, *scanner=None*, *dff_file=None*, *stimulus_pkl=None*, *eye_video=None*, *behavior_video=None*, *long_stim_threshold=0.2*)

Bases: object

behavior_video_timestamps

clipped_stim_timestamps

Return the stimulus timestamps with the erroneous initial spike removed (if relevant)

Returns

timestamps: np.ndarray An array of stimulus timestamps in seconds with th emonitor delay added

delta: int Difference between the length of timestamps and the number of frames reported in the stimulus pickle file, i.e. len(timestamps) - len(pkl_file['items']['behavior']['intervalsms']]

corrected_behavior_video_timestamps

corrected_eye_video_timestamps

corrected_ophys_timestamps

corrected_stim_timestamps

The stimulus timestamps corrected for monitor delay

Returns

timestamps: np.ndarray An array of stimulus timestamps in seconds with th emonitor delay added

delta: int Difference between the length of timestamps and the number of frames reported in the stimulus pickle file, i.e. len(timestamps) - len(pkl_file['items']['behavior']['intervalsms']]

delay: float The monitor delay in seconds

dataset

eye_video_timestamps

monitor_delay

The monitor delay (in seconds) associated with the session

ophys_timestamps

Get the timestamps for the ophys data.

stim_timestamps

allensdk.internal.brain_observatory.time_sync.**calculate_monitor_delay**(*sync_dset*, *stim_times*, *photodiode_key*, *transition_frame_interval=60*, *max_monitor_delay=0.07*)

Calculate monitor delay.

allensdk.internal.brain_observatory.time_sync.**corrected_video_timestamps**(*video_name*, *timestamps*, *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: allensdk.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 occured ~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*)

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*, *optional_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*, *content_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*, *strategy_class*, *host='lims2'*, ***kwargs*)

allensdk.internal.core.lims_utilities.**get_well_known_file_by_name**(*wkfs*, *filename*) 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*, *filename*)

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.**MouseConnectivityCachePre**

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 experients. 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, returna 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, decimals=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*, *output_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*, *image_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*, *highlight_sweep_number*, *sweep_numbers*, *highlight_color='#0779BE'*, *background_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*, *image_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*, *working_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*, *image_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*, *image_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*, *image_dir*, *sizes*, *cell_image_files*)

allensdk.internal.ephys.plot_qc_figures3.**plot_sweep_figures**(*nwb_file*, *sweep_data*, *image_dir*, *sizes*) `allensdk.internal.ephys.plot_qc_figures3.`**`plot_sweep_set_summary`**(*nwb_file*, *highlight_sweep_number*, *sweep_numbers*, *highlight_color='#0779BE'*, *background_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*, *image_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_s*

allensdk.internal.model.biophysical.passive_fitting.neuron_utils.**read_neuron_fit_stdout**(*func*

allensdk.internal.model.biophysical.passive_fitting.output_grabber module

class allensdk.internal.model.biophysical.passive_fitting.output_grabber.**OutputGrabber**(*strea*
threa

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_chec*
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_di*

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*, *output_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*, *output_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*, *output_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_s**

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*, *reverse=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 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.

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*, *error_function=None*, *error_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.**GlifBadInitializationExceptio**

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

respones [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** (*r* *v* *c* *t* *c* *v* *c* *t* *c* *c*

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*, *threshold_t0*, *threshold_t1*, *voltage_t0*, *voltage_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*, *threshold_t0*, *threshold_t1*, *voltage_t0*, *voltage_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*, *threshold*, *voltage_t1*, *threshold_t1*, *dt*)

allensdk.internal.model.glif.glif_optimizer_neuron.**interpolate_spike_voltage**(*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.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*, *processedStim_list*, *originalVoltage_list*, *processedVoltage_list*, *config*, *blockME=False*)

allensdk.internal.model.glif.plotting.**checkSpikeCutting**(*originalStim_list*, *cutStim_list*, *originalVoltage_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_RCEl_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 specifing 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_mu**

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 indicies 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*, *aboveValue=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 indicies 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.**invnl**(*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 programmitcally 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 morpyhology, 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 amout required to multiply to a moprhology

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 histgram 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*
in
je
tio
in
je
tio

Read nrrd files containing injection signal data

allensdk.internal.mouse_connectivity.interval_unionize.data_utilities.**get_projection_data**(*p*
p
j
t
a

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_n

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.**Tis**

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^3

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.**Tis**

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.**Tis**

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*, ***kwa*

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_c**

allensdk.internal.mouse_connectivity.projection_thumbnail.generate_projection_strip.**blend_w**

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**(*volu*
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_projecti**

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**(*ima wei*

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_c**

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_no**

allensdk.internal.mouse_connectivity.projection_thumbnail.visualization_utilities.**normalize**

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.**VolumePro**

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_diagona**

allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities.**sitk_get_image_p**

allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities.**sitk_get_size_pa**

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_dime tiles, average_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_po**

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*, *stup*, *cb=<built-in function array>*)

`allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.`**`get_blend_component`**(*indic lg*, *axis*, *mesh*

`allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.`**`get_indicator_bound_poi`**

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`**(*indicato meshes*)

`allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.`**`initialize_image`**(*dimensio nchannels*, *dtype*, *order='C'*)

`allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.`**`initialize_images`**(*dimensi nchannels*)

`allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.`**`make_blended_tile`**(*blend*, *tile*, *current_reg*

allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile module

`class` `allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.`**`Tile`**(*index*, *image*, *is_missing*, *bounds*, *channel*, *size*, *margins*, **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_j*
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_ru*
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*,
storage_directory='./',
local=True,
sdk_path='/data/informatics/C
script_path='/data/informatics
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*, *exclude_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*, *neuropil_trace*, *corrected_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*, *local=False*, *OUTPUT_DIR='/data/informatics/CAM/*, *SDK_PATH='/data/informatics/CAM*, *walltime='10:00:00'*, *python='/shared/utils.x86_64/python-2.7/bin/python'*, *queue='braintv'*)

allensdk.internal.pipeline_modules.run_observatory_analysis.**get_experiment_nwb_file**(*experimen*

allensdk.internal.pipeline_modules.run_observatory_analysis.**get_experiment_session**(*experiment*

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*, *prefix*, *aspect*, *configs*, *output_dir*, *axes=None*, *transparent=False*)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_correlation_plots**(*data_s*, *analysis_file*, *configs*, *output_di*

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_drifting_gratings**(*dga*, *configs*, *output_di*

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_experiment_thumbnails**(

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_eye_tracking_plots**(*data_*, *configs*, *output_*

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_locally_sparse_noise**(*ls* allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_natural_movie**(*nma*, *configs*, *output_dir*, *name*)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_natural_scenes**(*nsa*, *configs*, *output_dir*)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_plots**(*prefix*, *aspect*, *configs*, *output_dir*, *axes=None*, *transparent=False*)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_receptive_field**(*lsna*, *configs*, *output_dir*)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_speed_tuning**(*analysis*, *configs*, *output_dir*)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_static_gratings**(*sga*, *configs*, *output_dir*)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**build_type**(*nwb_file*, *data_file*, *configs*, *output_dir*, *type_name*)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**debug**(*experiment_id*, *plots=None*, *local=False*)

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**get_experiment_analysis_file**

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**get_experiment_files**(*experiment*

allensdk.internal.pipeline_modules.run_observatory_thumbnails.**get_experiment_nwb_file**(*experi* 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*, *local=False*)

allensdk.internal.pipeline_modules.run_ophys_eye_calibration.**get_wkf**(*wkf_type*, *experiment_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*, *position_degrees*, *position_cm*, *areas*)

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* *nels* *wid* *heig* *item* *size* *n_p*

allensdk.internal.pipeline_modules.run_ophys_session_decomposition.**debug**(*experiment_id*, *local=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, output_json_path: Optional[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 beore 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) heaviweight 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: allensdk.internal.brain_o*, *experiment_id: int*, *min_stimulus_delay: float*, *max_stimulus_delay: float*) → allensdk.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 defininition 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*, *function_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`**(*args_dict*)

Read configurations.

Parameters

args_dict [dict] Parsed arguments dictionary with following keys.

manifest_file [string] .json file with containing the experiment configuration

axon_type [string] Axon handling for the all-active models

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`**(*args*, *sweeps=None*, *procs=6*)

Main function for simulating sweeps in a biophysical experiment.

Parameters

args [dict] Parsed arguments 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*, *axon_type*)
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 refering 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 uits 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*, *AScurrents_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*, *AScurrents_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*, *threshold_t0*, *voltage_t0*, *AScurrents_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*, *threshold_t0*, *voltage_t0*, *AScurrents_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*, *threshold_t0*, *voltage_t0*, *AScurrents_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*, *voltage_t0*, *AScurrents_t0*, *inj*)

(TODO) Linear voltage dynamics.

allensdk.model.glif.glif_neuron_methods.**dynamics_voltage_linear_forward_euler**(*neuron*, *voltage_t0*, *AScurrents_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_voltag*, *b_voltag*, *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*, *intensity_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*, *polygon_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 posfiltering 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*, *polygon_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, polygon_info, segmentation_paths, intensity_paths, injection_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*, *window_size*)

allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.**block_average**(*volume*, *factor*) allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.**conv**(*image*, *factor*, *window_size*)

allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.**downsample_average**(*volume*, *current_spa* *target_spac*

allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.**extract**(*image*, *factor*, *window_size*, *window_step*, *output_shape*)

allensdk.mouse_connectivity.grid.utilities.downsampling_utilities.**window_average**(*volume*, *factor*)

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**(*dfmfield* *aff_para*

allensdk.mouse_connectivity.grid.utilities.image_utilities.**compute_coarse_parameters**(*in_dims* *in_spaci* *out_spa* *reduce_lev*

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**(*path*)

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*, *dfmfld_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 numpified

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*, *whitespace_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*)

6.3 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

CHAPTER 7

Allen Brain Observatory

The Allen Brain Observatory is a collection of data resources for understanding sensory processing in the mouse visual cortex. These resources systematically measure visual responses in multiple cortical areas and layers using two-photon calcium imaging or high-density extracellular electrophysiology (Neuropixels) probes. Recordings are performed on mice passively viewing visual stimuli or trained to actively perform an image change detection task.

Behavior	Modality	Resource	Initial Release
Passive	Optical physiology	Visual Coding - Optical Physiology	June 2016
Passive	Extracellular electrophysiology	Visual Coding - Neuropixels	October 2019
Active	Optical physiology	Visual Behavior - Optical Physiology	March 2021
Active	Extracellular electrophysiology	Visual Behavior - Neuropixels	Coming soon…

Experiment and stimulus data are provided in Neurodata Without Borders (NWB) files. The AllenSDK provides code to:

- download and organize experiment data according to cortical area, imaging depth, and Cre line
- access experiment metadata and data streams
- transform and analyze data

More information about each study is provided in the linked pages. A web-based entry point to the Visual Coding – Optical physiology data is available at http://observatory.brain-map.org/visualcoding .

_static/ccf_v3_sdk.png

CHAPTER 8

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*

_static/connectivity.png

CHAPTER 9

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 - 2.12.0

- Added ability to specify a static cache directory (use_static_cache=True) to instantiate VisualBehaviorOphysProjectCache.from_local_cache()
- Added 'experience_level', 'passive' and 'image_set' columns to ophys_experiments_table
- Added 'ophys_cells_table' metadata table to track the relationship between ophys_experiment_id and cell_specimen_id

CHAPTER 11

What's New - 2.11.3

- Bugfixes related to NWB creation for BehaviorSessions

CHAPTER 12

What's New - 2.11.2

- Fixed mkdir error for non-existing ecephys upload directory

CHAPTER 13

What's New - 2.11.1

- Refactored the schema for the Ecephys copy utility to avoid raising an error when a previous output file already exists.

CHAPTER 14

What's New - 2.11.0

- python 3.8 compatibility
- CloudCache (the class supporting cloud-based data releases) is now smart enough to construct symlinks between files that are identical across dataset versions (rather than downloading duplicate copies of files).
- VisualBehavioOphysProjectCache supports user-controlled switching between dataset versions.

CHAPTER 15

What's New - 2.10.3

- Adds restriction to require hdmf version to be strictly less than 2.5.0 which accidentally introduced a major version breaking change

CHAPTER 16

What's New - 2.10.2

- This version marks the release of Visual Behavior Optical Physiology data! For more details please visit the: Visual Behavior - Optical Physiology Project Page
- Update documentation to support visual behavior data release
- Fixes a bug with the dictionary returned by BehaviorSession get get_performance_metrics() method
- Adds docstrings to the BehaviorSession get_performance_metrics(), get_rolling_performance_df(), and get_reward_rate() methods

CHAPTER 17

What's New - 2.10.1

- Changes name of BehaviorProjectCache to VisualBehaviorOphysProjectCache
- Changes VisualBehaviorOphysProjectCache method get_session_table() to get_ophys_session_table()
- Changes VisualBehaviorOphysProjectCache method get_experiment_table() to get_ophys_experiment_table()
- VisualBehaviorOphysProjectCache is enabled to instantiate from_s3_cache() and from_local_cache()
- Improvements to BehaviorProjectCache
- Adds project metadata writer

CHAPTER 18

What's New - 2.9.0

- Updates to Session metadata; refactors implementation to use class rather than dict internally
- Fixes a bug that was preventing Allen Institute Windows users from accessing gratings images

CHAPTER 19

What's New - 2.8.0

- Created lookup table to get monitor_delay for cases where calculation from data fails
- If sync timestamp file has more timestamps than eye tracking moving has frame, trim excess timestamps (up to 15)
- Session API returns both warped and unwarped stimulus images, and both are written to NWB

CHAPTER 20

What's New - 2.7.0

- Refactored behavior and ophys session and data APIs to remove a circular inheritance issue
- Fixed segmentation mask and roi_mask misregistration in 'BehaviorOphysSession'
- Replaces BehaviorOphysSession.get_roi_masks() method with roi_masks property
- Fixes bug which prevented the SDK from loading stimuli dataframes for static gratings
- Return event detection data through session API
- Read/write event detection data from/to NWB
- Time stamps for events in trial_log are set to the exact sync timestamp of the corresponding frame.
- For behavior-only sessions, sync-like timestamp of the first frame is set to zero.
- Refactored BehaviorOphysSession to inherit methods and properties from BehaviorSession
- Fixed a test for checking that Behavior and BehaviorOphysSessions contain the same data regardless of which API (LIMS/JSON/NWB) is used. Also fixed resulting failure cases.

CHAPTER 21

What's New - 2.6.0

- Adds ability to write and read behavior only experiments
- Adds eye tracking ellipse fits and metadata as new NWB data stream
- OPhys Behavior data retrieval methods no longer depend on ROIs being ordered identically in different files.

CHAPTER 22

What's New - 2.5.0 (January 29, 2021)

- Adds unfiltered running speed as new data stream
- run_demixing gracefully ignores any ROIs that are not in the input trace file

CHAPTER 23

What's New - 2.4.0 (December 21, 2020)

As of the 2.4.0 release: - When running raster_plot on a spike_times dataframe, the spike times from each unit are plotted twice. (thank you @dgmurx) - improvements and fixes to behavior ophys NWB files. - improvements and fixes to BehaviorProjectCache tables including new column "donor_id" - implemented a timeout to obtaining an ecephys session. (thank you @wesley-jones) - big overhaul of how Behavior and BehaviorOphys classes are structured for the visual behavior project. See https://github.com/AllenInstitute/AllenSDK/pull/1789

CHAPTER 24

What's New - 2.3.2 (October 19, 2020)

As of the 2.3.2 release:

- (Internal) Fixed a running_processing bug for behavior ophys experiments when the input data would have one more encoder entry than timestamp. The behavior of the code now matches what the warning says.

CHAPTER 25

What's New - 2.3.1 (October 13, 2020)

As of the 2.3.1 release:

- (Internal) Fixed a write_nwb bug for behavior ophys experiments involving the BehaviorOphysJsonApi expecting a mesoscope-specific method.

CHAPTER 26

What's New - 2.3.0 (October 9, 2020)

As of the 2.3.0 release:

- Visual behavior running speed is now low-pass filtered at 10Hz. The raw running speed data is still available. The running speed is corrected for encoder threshold croissing artifacts.
- Support for stimulus gratings for visual behavior.
- Fixed an eye-tracking sync problem.
- Updates to some visual behavior pynwb implementations.
- Adds load sync data for individual plane sets to relate accurate event timings to mesoscope data.
- Adds public API method to access the behavior_session_id from an instance of BehaviorOphysSession.

CHAPTER 27

What's New - 2.2.0 (September 3, 2020)

As of the 2.2.0 release:

- AllenSDK HTTP engine streaming requests now include a progress bar
- *ImportError: cannot import name 'MultiContainerInterface' from 'hdmf.container'* errors should now be resolved (by removing explicit version bounds on the *hdmf* package).
- The optical physiology 2-photon trace demixer has been modified to be more memory friendly and should no longer result in out of memory errors when trying to demix very large movie stacks.

CHAPTER 28

What's New - 2.1.0 (July 16, 2020)

As of the 2.1.0 release:

- behavior ophys nwb files can now be written using updated pynwb and hdmf
- A warning has been added if you are using AllenSDK with outdated NWB files
- A new documentation file has been added which will contain Visual Behavior specific terms for quick lookup

CHAPTER 29

What's New - 2.0.0 (June 11, 2020)

As of the 2.0.0 release:

- pynwb and hdmf version requirements have been made less strict
- The organization of data for ecephys neuropixels Neurodata Without Borders (NWB) files has been significantly changed to conform with NWB specifications and best practices
- CCF locations for ecephys neuropixels electrodes are now written to NWB files
- Examples for accessing eye tracking ellipse fit and screen gaze location data have been added to ecephys example notebooks

Important Note: Due to newer versions of pynwb/hdmf having issues reading previously released Visual Coding Neuropixels NWB files and due to the significant reorganization of their NWB file contents, this release contains breaking changes that necessitate a major version revision. NWB files released prior to 6/11/2020 are not guaranteed to work with the 2.0.0 version of AllenSDK. If you cannot or choose not to re-download the updated NWB files, you can continue using a prior version of AllenSDK (< 2.0.0) to access them. However, no further features or bugfixes for AllenSDK (< 2.0.0) are planned. Data released for other projects (Cell Types, Mouse Connectivity, etc.) are *NOT* affected and will *NOT* need to be re-downloaded

CHAPTER 30

Previous Release Notes

- 1.8.0
- 1.7.1
- 1.7.0
- 1.6.0
- 1.5.0
- 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.

Python Module Index

a

allensdk, 382
allensdk.api, 106
allensdk.api.api, 103
allensdk.api.cloud_cache, 68
allensdk.api.cloud_cache.file_attributes, 65
allensdk.api.cloud_cache.manifest, 66
allensdk.api.cloud_cache.utils, 67
allensdk.api.queries, 97
allensdk.api.queries.annotated_section_data_sets_api, 68
allensdk.api.queries.biophysical_api, 69
allensdk.api.queries.brain_observatory_api, 71
allensdk.api.queries.cell_types_api, 74
allensdk.api.queries.connected_services, 76
allensdk.api.queries.glif_api, 76
allensdk.api.queries.grid_data_api, 77
allensdk.api.queries.image_download_api, 79
allensdk.api.queries.mouse_atlas_api, 82
allensdk.api.queries.mouse_connectivity_api, 83
allensdk.api.queries.ontologies_api, 86
allensdk.api.queries.reference_space_api, 87
allensdk.api.queries.rma_api, 89
allensdk.api.queries.rma_pager, 94
allensdk.api.queries.rma_template, 94
allensdk.api.queries.svg_api, 95
allensdk.api.queries.synchronization_api, 95
allensdk.api.queries.tree_search_api, 97
allensdk.api.warehouse_cache, 103
allensdk.api.warehouse_cache.cache, 97
allensdk.api.warehouse_cache.caching_utilities, 101

b

allensdk.brain_observatory, 232
allensdk.brain_observatory.argschema_utilities, 197
allensdk.brain_observatory.behavior, 165
allensdk.brain_observatory.behavior.behavior_projec
109
allensdk.brain_observatory.behavior.behavior_projec
108
allensdk.brain_observatory.behavior.behavior_projec
108
allensdk.brain_observatory.behavior.behavior_projec
109
allensdk.brain_observatory.behavior.behavior_projec
109
allensdk.brain_observatory.behavior.behavior_projec
108
allensdk.brain_observatory.behavior.behavior_projec
107
allensdk.brain_observatory.behavior.criteria, 136
allensdk.brain_observatory.behavior.dprime, 137
allensdk.brain_observatory.behavior.event_detectio
138
allensdk.brain_observatory.behavior.eye_tracking_pr
139
allensdk.brain_observatory.behavior.image_api, 142
allensdk.brain_observatory.behavior.metadata, 113
allensdk.brain_observatory.behavior.metadata.behavi
110
allensdk.brain_observatory.behavior.metadata.behavi
112

allensdk.brain_observatory.behavior.mtrain, 143
allensdk.brain_observatory.behavior.rewards_processing, 145
allensdk.brain_observatory.behavior.running_processing, 145
allensdk.brain_observatory.behavior.schemas, 147
allensdk.brain_observatory.behavior.session_apis, 126
allensdk.brain_observatory.behavior.session_apis.abcs, 119
allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base, 115
allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_ext... 113
allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_ophys_da... 114
allensdk.brain_observatory.behavior.session_apis.abcs.session_base, 119
allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_base, 115
allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base, 117
allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api, 119
allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api, 120
allensdk.brain_observatory.behavior.session_apis.data_transforms, 126
allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms, 122
allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_trans... 124
allensdk.brain_observatory.behavior.session_metrics, 158
allensdk.brain_observatory.behavior.stimulus_processing, 129
allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates, 127
allensdk.brain_observatory.behavior.stimulus_processing.util, 129
allensdk.brain_observatory.behavior.sync, 132
allensdk.brain_observatory.behavior.sync.process_sync, 131
allensdk.brain_observatory.behavior.trial_masks, 158
allensdk.brain_observatory.behavior.trials_processing, 159
allensdk.brain_observatory.behavior.write_behavior_nwb, 135
allensdk.brain_observatory.behavior.write_nwb, 135
allensdk.brain_observatory.behavior.write_nwb.exter... 135
allensdk.brain_observatory.behavior.write_nwb.exter... 135
allensdk.brain_observatory.behavior.write_nwb.exter... 135
allensdk.brain_observatory.brain_observatory_except... 200
allensdk.brain_observatory.brain_observatory_plotti... 200
allensdk.brain_observatory.chisquare_categorical, 200
allensdk.brain_observatory.circle_plots, 201
allensdk.brain_observatory.demixer, 203
allensdk.brain_observatory.dff, 204
allensdk.brain_observatory.drifting_gratings, 206
allensdk.brain_observatory.ecephys, 186
allensdk.brain_observatory.ecephys.align_timestamps... 169
allensdk.brain_observatory.ecephys.align_timestamps... 165
allensdk.brain_observatory.ecephys.align_timestamps... 167
allensdk.brain_observatory.ecephys.align_timestamps... 168
allensdk.brain_observatory.ecephys.copy_utility, 169
allensdk.brain_observatory.ecephys.current_source_... 169
allensdk.brain_observatory.ecephys.ecephys_project_... 170
allensdk.brain_observatory.ecephys.ecephys_session_... 171
allensdk.brain_observatory.ecephys.file_io, 173
allensdk.brain_observatory.ecephys.file_io.continuo... 172
allensdk.brain_observatory.ecephys.file_io.ecephys_... 172
allensdk.brain_observatory.ecephys.file_io.stim_fil... 172
allensdk.brain_observatory.ecephys.lfp_subsampling, 175
allensdk.brain_observatory.ecephys.lfp_subsampling... 173
allensdk.brain_observatory.ecephys.optotagging_tabl... 175
allensdk.brain_observatory.ecephys.stimulus_sync, 185
allensdk.brain_observatory.ecephys.stimulus_table,

184
allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes, 177
allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities, 181
allensdk.brain_observatory.ecephys.stimulus_table.output_validation, 182
allensdk.brain_observatory.ecephys.stimulus_table.stimulus_parameter_extraction, 183
allensdk.brain_observatory.ecephys.stimulus_table.visualization, 177
allensdk.brain_observatory.ecephys.stimulus_table.visualization.view_blocks, 176
allensdk.brain_observatory.ecephys.visualization, 184
allensdk.brain_observatory.ecephys.write_nwb, 185
allensdk.brain_observatory.extract_running_speed, 186
allensdk.brain_observatory.findlevel, 207
allensdk.brain_observatory.gaze_mapping, 186
allensdk.brain_observatory.locally_sparse_noise, 207
allensdk.brain_observatory.natural_movie, 208
allensdk.brain_observatory.natural_scenes, 209
allensdk.brain_observatory.observatory_plots, 210
allensdk.brain_observatory.ophys, 187
allensdk.brain_observatory.ophys.trace_extraction, 187
allensdk.brain_observatory.r_neuropil, 212
allensdk.brain_observatory.receptive_field_analysis, 195
allensdk.brain_observatory.receptive_field_analysis.chisquarerf, 187
allensdk.brain_observatory.receptive_field_analysis.eventdetection, 190
allensdk.brain_observatory.receptive_field_analysis.fit_parameters, 191
allensdk.brain_observatory.receptive_field_analysis.fitgaussian2D, 191
allensdk.brain_observatory.receptive_field_analysis.postprocessing, 192
allensdk.brain_observatory.receptive_field_analysis.receptive_field, 192
allensdk.brain_observatory.receptive_field_analysis.tools, 193
allensdk.brain_observatory.receptive_field_analysis.utilities, 193
allensdk.brain_observatory.receptive_field_analysis, 194
allensdk.brain_observatory.roi_masks, 214
allensdk.brain_observatory.running_speed, 216
allensdk.brain_observatory.session_analysis, 217
allensdk.brain_observatory.session_api_utils, 219
allensdk.brain_observatory.static_gratings, 221
allensdk.brain_observatory.stimulus_analysis, 222
allensdk.brain_observatory.stimulus_info, 224
allensdk.brain_observatory.sync_dataset, 228
allensdk.brain_observatory.sync_utilities, 196
allensdk.brain_observatory.visualization, 196

c

allensdk.config, 240
allensdk.config.app, 234
allensdk.config.app.application_config, 232
allensdk.config.manifest, 238
allensdk.config.manifest_builder, 240
allensdk.config.model, 238
allensdk.config.model.description, 236
allensdk.config.model.description_parser, 237
allensdk.config.model.formats, 236
allensdk.config.model.formats.hdf5_util, 235
allensdk.config.model.formats.json_description_pars 235
allensdk.config.model.formats.pycfg_description_par 236
allensdk.core, 276
allensdk.core.auth_config, 241
allensdk.core.authentication, 241
allensdk.core.brain_observatory_nwb_data_set, 242
allensdk.core.cache_method_utilities, 247
allensdk.core.cell_types_cache, 247
allensdk.core.dat_utilities, 249
allensdk.core.exceptions, 250
allensdk.core.h5_utilities, 250
allensdk.core.json_utilities, 251
allensdk.core.lazy_property, 241

allensdk.core.lazy_property.lazy_property, 241
allensdk.core.lazy_property.lazy_property_mixin, 241
allensdk.core.mouse_connectivity_cache, 252
allensdk.core.nwb_data_set, 257
allensdk.core.obj_utilities, 259
allensdk.core.ontology, 259
allensdk.core.ophys_experiment_session_id_mapping, 260
allensdk.core.reference_space, 260
allensdk.core.reference_space_cache, 263
allensdk.core.simple_tree, 265
allensdk.core.sitk_utilities, 268
allensdk.core.structure_tree, 269
allensdk.core.swc, 271
allensdk.core.typing, 275

d

allensdk.deprecated, 382

e

allensdk.ephys, 288
allensdk.ephys.ephys_extractor, 276
allensdk.ephys.ephys_features, 280
allensdk.ephys.extract_cell_features, 287
allensdk.ephys.feature_extractor, 288

i

allensdk.internal, 359
allensdk.internal.api, 293
allensdk.internal.api.api_prerelease, 293
allensdk.internal.api.lims_api, 293
allensdk.internal.api.queries, 292
allensdk.internal.api.queries.biophysical_module_api, 288
allensdk.internal.api.queries.biophysical_module_reader, 289
allensdk.internal.api.queries.grid_data_api_prerelease, 290
allensdk.internal.api.queries.mouse_connectivity_api_prerelease, 291
allensdk.internal.api.queries.optimize_config_reader, 291
allensdk.internal.brain_observatory, 309
allensdk.internal.brain_observatory.annotated_region_metrics, 294
allensdk.internal.brain_observatory.demix_report, 295
allensdk.internal.brain_observatory.demixer, 296
allensdk.internal.brain_observatory.eye_calibratio 297
allensdk.internal.brain_observatory.fit_ellipse, 300
allensdk.internal.brain_observatory.frame_stream, 301
allensdk.internal.brain_observatory.itracker_utils, 302
allensdk.internal.brain_observatory.mask_set, 303
allensdk.internal.brain_observatory.ophys_session_ 304
allensdk.internal.brain_observatory.resources, 294
allensdk.internal.brain_observatory.roi_filter_uti 305
allensdk.internal.brain_observatory.time_sync, 307
allensdk.internal.core, 312
allensdk.internal.core.lims_pipeline_module, 309
allensdk.internal.core.lims_utilities, 309
allensdk.internal.core.mouse_connectivity_cache_pre 310
allensdk.internal.core.simpletree, 311
allensdk.internal.core.swc, 312
allensdk.internal.ephys, 318
allensdk.internal.ephys.core_feature_extract, 312
allensdk.internal.ephys.plot_qc_figures, 313
allensdk.internal.ephys.plot_qc_figures3, 315
allensdk.internal.model, 335
allensdk.internal.model.AIC, 333
allensdk.internal.model.biophysical, 322
allensdk.internal.model.biophysical.biophysical_ar 320
allensdk.internal.model.biophysical.check_fi_shift, 320
allensdk.internal.model.biophysical.deap_utils, 320
allensdk.internal.model.biophysical.ephys_utils, 320
allensdk.internal.model.biophysical.fits, 318
allensdk.internal.model.biophysical.fits.fit_style 318
allensdk.internal.model.biophysical.make_deap_fit_ 321
allensdk.internal.model.biophysical.passive_fittin 320
allensdk.internal.model.biophysical.passive_fittin

319
allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit2, 319
allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit_elec, 319
allensdk.internal.model.biophysical.passive_fitting.neuron_utils, 319
allensdk.internal.model.biophysical.passive_fitting.output_grabber, 319
allensdk.internal.model.biophysical.passive_fitting.passive, 318
allensdk.internal.model.biophysical.passive_fitting.preprocess, 319
allensdk.internal.model.biophysical.run_optimize, 321
allensdk.internal.model.biophysical.run_optimize_workflow, 322
allensdk.internal.model.biophysical.run_passive_fit, 322
allensdk.internal.model.biophysical.run_simulate_lims, 322
allensdk.internal.model.biophysical.run_simulate_workflow, 322
allensdk.internal.model.data_access, 334
allensdk.internal.model.glif, 333
allensdk.internal.model.glif.are_two_lists_of_arrays_the_same, 323
allensdk.internal.model.glif.error_functions, 323
allensdk.internal.model.glif.find_spikes, 323
allensdk.internal.model.glif.find_sweeps, 323
allensdk.internal.model.glif.glif_experiment, 324
allensdk.internal.model.glif.glif_optimizer, 326
allensdk.internal.model.glif.glif_optimizer_neuron, 326
allensdk.internal.model.glif.MLIN, 322
allensdk.internal.model.glif.plotting, 329
allensdk.internal.model.glif.rc, 330
allensdk.internal.model.glif.spike_cutting, 330
allensdk.internal.model.glif.threshold_adaptation, 331
allensdk.internal.model.GLM, 334
allensdk.internal.morphology, 342
allensdk.internal.morphology.compartment, 335
allensdk.internal.morphology.morphology, 335
allensdk.internal.morphology.morphvis,
339
allensdk.internal.morphology.node, 342
allensdk.internal.morphology.validate_swc, 342
allensdk.internal.mouse_connectivity, 351
allensdk.internal.mouse_connectivity.interval_unio 347
allensdk.internal.mouse_connectivity.interval_unio 343
allensdk.internal.mouse_connectivity.interval_unio 343
allensdk.internal.mouse_connectivity.interval_unio 345
allensdk.internal.mouse_connectivity.interval_unio 345
allensdk.internal.mouse_connectivity.interval_unio 346
allensdk.internal.mouse_connectivity.interval_unio 346
allensdk.internal.mouse_connectivity.projection_thu 349
allensdk.internal.mouse_connectivity.projection_thu 347
allensdk.internal.mouse_connectivity.projection_thu 347
allensdk.internal.mouse_connectivity.projection_thu 348
allensdk.internal.mouse_connectivity.projection_thu 348
allensdk.internal.mouse_connectivity.projection_thu 349
allensdk.internal.mouse_connectivity.projection_thu 349
allensdk.internal.mouse_connectivity.tissuecyte_st 351
allensdk.internal.mouse_connectivity.tissuecyte_st 349
allensdk.internal.mouse_connectivity.tissuecyte_st 350
allensdk.internal.pipeline_modules, 359
allensdk.internal.pipeline_modules.gbm, 352
allensdk.internal.pipeline_modules.gbm.generate_gbm 351
allensdk.internal.pipeline_modules.gbm.generate_gbm 351
allensdk.internal.pipeline_modules.gbm.generate_gbm 352
allensdk.internal.pipeline_modules.run_annotated_re 352
allensdk.internal.pipeline_modules.run_demixing, 352
allensdk.internal.pipeline_modules.run_dff_computat

352

allensdk.internal.pipeline_modules.run_neuropil_correction, 353

allensdk.internal.pipeline_modules.run_observatory_analysis, 353

allensdk.internal.pipeline_modules.run_observatory_thumbnails, 354

allensdk.internal.pipeline_modules.run_ophys_eye_calibration, 356

allensdk.internal.pipeline_modules.run_ophys_session_decomposition, 356

allensdk.internal.pipeline_modules.run_ophys_time_sync, 357

allensdk.internal.pipeline_modules.run_tissuecyte_unionize_classic_from_json, 359

m

allensdk.model, 373

allensdk.model.biophys_sim, 361

allensdk.model.biophys_sim.bps_command, 360

allensdk.model.biophys_sim.config, 360

allensdk.model.biophys_sim.neuron, 360

allensdk.model.biophys_sim.neuron.hoc_utils, 359

allensdk.model.biophys_sim.scripts, 360

allensdk.model.biophysical, 363

allensdk.model.biophysical.run_simulate, 361

allensdk.model.biophysical.runner, 361

allensdk.model.biophysical.utils, 362

allensdk.model.glif, 373

allensdk.model.glif.glif_neuron, 363

allensdk.model.glif.glif_neuron_methods, 367

allensdk.model.glif.simulate_neuron, 372

allensdk.morphology, 373

allensdk.morphology.validate_swc, 373

allensdk.mouse_connectivity, 381

allensdk.mouse_connectivity.grid, 381

allensdk.mouse_connectivity.grid.image_series_gridder, 380

allensdk.mouse_connectivity.grid.subimage, 377

allensdk.mouse_connectivity.grid.subimage.base_subimage, 374

allensdk.mouse_connectivity.grid.subimage.cav_subimage, 375

allensdk.mouse_connectivity.grid.subimage.classic_subimage, 376

allensdk.mouse_connectivity.grid.subimage.count_subimage, 377

allensdk.mouse_connectivity.grid.utilities, 379

allensdk.mouse_connectivity.grid.utilities.downsamp 377

allensdk.mouse_connectivity.grid.utilities.image_ut 378

allensdk.mouse_connectivity.grid.writers, 380

t

allensdk.test_utilities, 382

allensdk.test_utilities.custom_comparators, 381

allensdk.test_utilities.regression_fixture, 381

allensdk.test_utilities.temp_dir, 381

Index

A

ab_from_diagonals() (*in module allensdk.brain_observatory.r_neuropil*), 212
ab_from_T() (*in module allensdk.brain_observatory.r_neuropil*), 212
accumulator_to_numpy() (*allensdk.mouse_connectivity.grid.image_series_gridder.ImageSeriesGridder method*), 380
acquisition_rate (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 222
actual_parameters_from_normalized() (*allensdk.internal.model.biophysical.deap_utils.Utils method*), 320
adaptation_index() (*allensdk.ephys.feature_extractor.EphysFeatureExtractor method*), 288
adaptation_index() (*in module allensdk.ephys.ephys_features*), 280
add() (*allensdk.brain_observatory.stimulus_info.BinaryIntervalSearchTree method*), 224
add_angle_labels() (*in module allensdk.brain_observatory.circle_plots*), 202
add_arrow() (*in module allensdk.brain_observatory.circle_plots*), 202
add_experience_level_to_experiment_table() (*in module allensdk.brain_observatory.behavior.behavior_project_cache.tables.util.experiments_table_utils*), 107
add_file() (*allensdk.config.manifest.Manifest method*), 238
add_image_set_to_experiment_table() (*in module allensdk.brain_observatory.behavior.behavior_project_cache.tables.util.experiments_table_utils*), 107
add_manifest_paths() (*allensdk.api.warehouse_cache.cache.Cache method*), 97
add_manifest_paths() (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method*), 253
add_manifest_paths() (*allensdk.core.reference_space_cache.ReferenceSpaceCache method*), 263
add_manifest_paths() (*allensdk.internal.core.mouse_connectivity_cache_prerelease.Mouse method*), 311
add_number_to_shuffled_movie() (*in module allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities*), 181
add_passive_flag_to_ophys_experiment_table() (*in module allensdk.brain_observatory.behavior.behavior_project_cache.tables*), 107
add_path() (*allensdk.config.manifest.Manifest method*), 238
add_path() (*allensdk.config.manifest_builder.ManifestBuilder method*), 240
add_paths() (*allensdk.config.manifest.Manifest method*), 238
add_section() (*allensdk.config.manifest_builder.ManifestBuilder method*), 240
add_to_fit_parameters_dict_single() (*in module allensdk.brain_observatory.receptive_field_analysis.fit_parameters*), 191
adjust_r_for_negativity() (*in module allensdk.internal.pipeline_modules.run_neuropil_correction*), 353
advance_combination() (*in module allensdk.brain_observatory.chisquare_categorical*), 200
age_in_days (*allensdk.brain_observatory.behavior.metadata.behavior_r attribute*), 110
AIC() (*in module allensdk.internal.model.AIC*), 333
AICc() (*in module allensdk.internal.model.AIC*), 333
align_and_cut_spikes() (*in module al-*

lensdk.internal.model.glif.find_spikes), 323
align_running_speed() (in module allensdk.core.brain_observatory_nwb_data_set), 246
ALIGNMENT3D_KEY (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute), 252
ALL (allensdk.api.queries.rma_api.RmaApi attribute), 89
all_stimuli() (in module allensdk.brain_observatory.stimulus_info), 225
AllActiveUtils (class in allensdk.model.biophysical.utils), 362
allensdk (module), 382
allensdk.api (module), 106
allensdk.api.api (module), 103
allensdk.api.cloud_cache (module), 68
allensdk.api.cloud_cache.file_attributes (module), 65
allensdk.api.cloud_cache.manifest (module), 66
allensdk.api.cloud_cache.utils (module), 67
allensdk.api.queries (module), 97
allensdk.api.queries.annotated_section_data_sets_api (module), 68
allensdk.api.queries.biophysical_api (module), 69
allensdk.api.queries.brain_observatory_api (module), 71
allensdk.api.queries.cell_types_api (module), 74
allensdk.api.queries.connected_services (module), 76
allensdk.api.queries.glif_api (module), 76
allensdk.api.queries.grid_data_api (module), 77
allensdk.api.queries.image_download_api (module), 79
allensdk.api.queries.mouse_atlas_api (module), 82
allensdk.api.queries.mouse_connectivity_api (module), 83
allensdk.api.queries.ontologies_api (module), 86
allensdk.api.queries.reference_space_api (module), 87
allensdk.api.queries.rma_api (module), 89
allensdk.api.queries.rma_pager (module), 94
allensdk.api.queries.rma_template (module), 94
allensdk.api.queries.svg_api (module), 95
allensdk.api.queries.synchronization_api (module), 95
allensdk.api.queries.tree_search_api (module), 97
allensdk.api.warehouse_cache (module), 103
allensdk.api.warehouse_cache.cache (module), 97
allensdk.api.warehouse_cache.caching_utilities (module), 101
allensdk.brain_observatory (module), 232
allensdk.brain_observatory.argschema_utilities (module), 197
allensdk.brain_observatory.behavior (module), 165
allensdk.brain_observatory.behavior.behavior_projec (module), 109
allensdk.brain_observatory.behavior.behavior_projec (module), 108
allensdk.brain_observatory.behavior.behavior_projec (module), 108
allensdk.brain_observatory.behavior.behavior_projec (module), 109
allensdk.brain_observatory.behavior.behavior_projec (module), 109
allensdk.brain_observatory.behavior.behavior_projec (module), 108
allensdk.brain_observatory.behavior.behavior_projec (module), 107
allensdk.brain_observatory.behavior.criteria (module), 136
allensdk.brain_observatory.behavior.dprime (module), 137
allensdk.brain_observatory.behavior.event_detection (module), 138
allensdk.brain_observatory.behavior.eye_tracking_pr (module), 139
allensdk.brain_observatory.behavior.image_api (module), 142
allensdk.brain_observatory.behavior.metadata (module), 113
allensdk.brain_observatory.behavior.metadata.behavi (module), 110
allensdk.brain_observatory.behavior.metadata.behavi (module), 112
allensdk.brain_observatory.behavior.mtrain (module), 143
allensdk.brain_observatory.behavior.rewards_process (module), 145
allensdk.brain_observatory.behavior.running_process (module), 145
allensdk.brain_observatory.behavior.schemas (module), 147
allensdk.brain_observatory.behavior.session_apis (module), 126
allensdk.brain_observatory.behavior.session_apis.ab

(module), 119
allensdk.brain_observatory.behavior.sess
(module), 115
allensdk.brain_observatory.behavior.sess
(module), 113
allensdk.brain_observatory.behavior.sess
(module), 114
allensdk.brain_observatory.behavior.sess
(module), 119
allensdk.brain_observatory.behavior.sess
(module), 115
allensdk.brain_observatory.behavior.sess
(module), 117
allensdk.brain_observatory.behavior.sess
(module), 119
allensdk.brain_observatory.behavior.sess
(module), 120
allensdk.brain_observatory.behavior.sess
(module), 126
allensdk.brain_observatory.behavior.sess
(module), 122
allensdk.brain_observatory.behavior.sess
(module), 124
allensdk.brain_observatory.behavior.sess
(module), 158
allensdk.brain_observatory.behavior.stim
(module), 129
allensdk.brain_observatory.behavior.stim
(module), 127
allensdk.brain_observatory.behavior.stim
(module), 129
allensdk.brain_observatory.behavior.sync
(module), 132
allensdk.brain_observatory.behavior.sync
(module), 131
allensdk.brain_observatory.behavior.trial
(module), 158
allensdk.brain_observatory.behavior.trial
(module), 159
allensdk.brain_observatory.behavior.write
(module), 135
allensdk.brain_observatory.behavior.write
(module), 135
allensdk.brain_observatory.behavior.write
(module), 135
allensdk.brain_observatory.behavior.write
(module), 135
allensdk.brain_observatory.behavior.write
(module), 135
allensdk.brain_observatory.brain_observat
(module), 200
allensdk.brain_observatory.brain_observat
(module), 200
allensdk.brain_observatory.chisquare_categorical
(module), 200
allensdk.brain_observatory.circle_plots
(module), 201
allensdk.brain_observatory.demixer (module), 203
allensdk.brain_observatory.dff (module), 204
allensdk.brain_observatory.drifting_gratings
(module), 206
allensdk.brain_observatory.ecephys (module), 186
allensdk.brain_observatory.ecephys.align_timestamps
(module), 169
allensdk.brain_observatory.ecephys.align_timestamps
(module), 165
allensdk.brain_observatory.ecephys.align_timestamps
(module), 167
allensdk.brain_observatory.ecephys.align_timestamps
(module), 168
allensdk.brain_observatory.ecephys.align_timestamps
(module), 169
allensdk.brain_observatory.ecephys.align_timestamps
(module), 169
allensdk.brain_observatory.ecephys.current_source_
(module), 169
allensdk.brain_observatory.ecephys.ecephys_project_
(module), 170
allensdk.brain_observatory.ecephys.ecephys_session_
(module), 171
allensdk.brain_observatory.ecephys.file_io
(module), 173
allensdk.brain_observatory.ecephys.file_io.continuo
(module), 172
allensdk.brain_observatory.ecephys.file_io.ecephys_
(module), 172
allensdk.brain_observatory.ecephys.file_io.stim_fil
(module), 172
allensdk.brain_observatory.ecephys.lfp_subsampling
(module), 175
allensdk.brain_observatory.ecephys.lfp_subsampling
(module), 173
allensdk.brain_observatory.ecephys.optotagging_tabl
(module), 175
allensdk.brain_observatory.ecephys.stimulus_sync
(module), 185
allensdk.brain_observatory.ecephys.stimulus_table
(module), 184
allensdk.brain_observatory.ecephys.stimulus_table.e
(module), 177
allensdk.brain_observatory.ecephys.stimulus_table.n
(module), 181
allensdk.brain_observatory.ecephys.stimulus_table.o
(module), 182
allensdk.brain_observatory.ecephys.stimulus_table.s

(*module*), 183
allensdk.brain_observatory.ecephys.stimulus_table.[illegible] (*module*), 177
allensdk.brain_observatory.ecephys.stimulus_table.[illegible] (*module*), 176
allensdk.brain_observatory.ecephys.visualization (*module*), 184
allensdk.brain_observatory.ecephys.write_nwb (*module*), 185
allensdk.brain_observatory.extract_running_speed (*module*), 186
allensdk.brain_observatory.findlevel (*module*), 207
allensdk.brain_observatory.gaze_mapping (*module*), 186
allensdk.brain_observatory.locally_sparse_noise (*module*), 207
allensdk.brain_observatory.natural_movie (*module*), 208
allensdk.brain_observatory.natural_scenes (*module*), 209
allensdk.brain_observatory.observatory_plots (*module*), 210
allensdk.brain_observatory.ophys (*module*), 187
allensdk.brain_observatory.ophys.trace_extraction (*module*), 187
allensdk.brain_observatory.r_neuropil (*module*), 212
allensdk.brain_observatory.receptive_field_analysis (*module*), 195
allensdk.brain_observatory.receptive_field_analysis.chisquarerf (*module*), 187
allensdk.brain_observatory.receptive_field_analysis.eventdetection (*module*), 190
allensdk.brain_observatory.receptive_field_analysis.fit_parameters (*module*), 191
allensdk.brain_observatory.receptive_field_analysis.fitgaussian2D (*module*), 191
allensdk.brain_observatory.receptive_field_analysis.postprocessing (*module*), 192
allensdk.brain_observatory.receptive_field_analysis.receptive_field (*module*), 192
allensdk.brain_observatory.receptive_field_analysis.tools (*module*), 193
allensdk.brain_observatory.receptive_field_analysis.utilities (*module*), 193
allensdk.brain_observatory.receptive_field_analysis.visualization (*module*), 194
allensdk.brain_observatory.roi_masks (*module*), 214
allensdk.brain_observatory.running_speed (*module*), 216
allensdk.brain_observatory.session_analysis (*module*), 217
allensdk.brain_observatory.session_api_utils (*module*), 219
allensdk.brain_observatory.static_gratings (*module*), 221
allensdk.brain_observatory.stimulus_analysis (*module*), 222
allensdk.brain_observatory.stimulus_info (*module*), 224
allensdk.brain_observatory.sync_dataset (*module*), 228
allensdk.brain_observatory.sync_utilities (*module*), 196
allensdk.brain_observatory.visualization (*module*), 196
allensdk.config (*module*), 240
allensdk.config.app (*module*), 234
allensdk.config.app.application_config (*module*), 232
allensdk.config.manifest (*module*), 238
allensdk.config.manifest_builder (*module*), 240
allensdk.config.model (*module*), 238
allensdk.config.model.description (*module*), 236
allensdk.config.model.description_parser (*module*), 237
allensdk.config.model.formats (*module*), 236
allensdk.config.model.formats.hdf5_util (*module*), 235
allensdk.config.model.formats.json_description_parser (*module*), 235
allensdk.config.model.formats.pycfg_description_parser (*module*), 236
allensdk.core (*module*), 276
allensdk.core.auth_config (*module*), 241
allensdk.core.authentication (*module*), 241
allensdk.core.brain_observatory_nwb_data_set (*module*), 242
allensdk.core.cache_method_utilities (*module*), 247
allensdk.core.cell_types_cache (*module*), 247
allensdk.core.dat_utilities (*module*), 249
allensdk.core.exceptions (*module*), 250
allensdk.core.h5_utilities (*module*), 250
allensdk.core.json_utilities (*module*), 251
allensdk.core.lazy_property (*module*), 241
allensdk.core.lazy_property.lazy_property (*module*), 241
allensdk.core.lazy_property.lazy_property_mixin (*module*), 241
allensdk.core.mouse_connectivity_cache

(module), 252
allensdk.core.nwb_data_set (module), 257
allensdk.core.obj_utilities (module), 259
allensdk.core.ontology (module), 259
allensdk.core.ophys_experiment_session_id_mapping (module), 260
allensdk.core.reference_space (module), 260
allensdk.core.reference_space_cache (module), 263
allensdk.core.simple_tree (module), 265
allensdk.core.sitk_utilities (module), 268
allensdk.core.structure_tree (module), 269
allensdk.core.swc (module), 271
allensdk.core.typing (module), 275
allensdk.deprecated (module), 382
allensdk.ephys (module), 288
allensdk.ephys.ephys_extractor (module), 276
allensdk.ephys.ephys_features (module), 280
allensdk.ephys.extract_cell_features (module), 287
allensdk.ephys.feature_extractor (module), 288
allensdk.internal (module), 359
allensdk.internal.api (module), 293
allensdk.internal.api.api_prerelease (module), 293
allensdk.internal.api.lims_api (module), 293
allensdk.internal.api.queries (module), 292
allensdk.internal.api.queries.biophysical_module_api (module), 288
allensdk.internal.api.queries.biophysical_module_reader (module), 289
allensdk.internal.api.queries.grid_data_api_prerelease (module), 290
allensdk.internal.api.queries.mouse_connectivity_api_prerelease (module), 291
allensdk.internal.api.queries.optimize_config_reader (module), 291
allensdk.internal.brain_observatory (module), 309
allensdk.internal.brain_observatory.annotated_region_metrics (module), 294
allensdk.internal.brain_observatory.demix_report (module), 295
allensdk.internal.brain_observatory.demixer (module), 296
allensdk.internal.brain_observatory.eye_calibration (module), 297
allensdk.internal.brain_observatory.fit_ellipse (module), 300
allensdk.internal.brain_observatory.frame_stream (module), 301
allensdk.internal.brain_observatory.itracker_utils (module), 302
allensdk.internal.brain_observatory.mask_set (module), 303
allensdk.internal.brain_observatory.ophys_session_d (module), 304
allensdk.internal.brain_observatory.resources (module), 294
allensdk.internal.brain_observatory.roi_filter_util (module), 305
allensdk.internal.brain_observatory.time_sync (module), 307
allensdk.internal.core (module), 312
allensdk.internal.core.lims_pipeline_module (module), 309
allensdk.internal.core.lims_utilities (module), 309
allensdk.internal.core.mouse_connectivity_cache_pre (module), 310
allensdk.internal.core.simpletree (module), 311
allensdk.internal.core.swc (module), 312
allensdk.internal.ephys (module), 318
allensdk.internal.ephys.core_feature_extract (module), 312
allensdk.internal.ephys.plot_qc_figures (module), 313
allensdk.internal.ephys.plot_qc_figures3 (module), 315
allensdk.internal.model (module), 335
allensdk.internal.model.AIC (module), 333
allensdk.internal.model.biophysical (module), 322
allensdk.internal.model.biophysical.biophysical_arc (module), 320
allensdk.internal.model.biophysical.check_fi_shift (module), 320
allensdk.internal.model.biophysical.deap_utils (module), 320
allensdk.internal.model.biophysical.ephys_utils (module), 320
allensdk.internal.model.biophysical.fits (module), 318
allensdk.internal.model.biophysical.fits.fit_styles (module), 318
allensdk.internal.model.biophysical.make_deap_fit_j (module), 321
allensdk.internal.model.biophysical.passive_fitting (module), 320
allensdk.internal.model.biophysical.passive_fitting (module), 319

allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit2 (*module*), 319
allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit_elec (*module*), 319
allensdk.internal.model.biophysical.passive_fitting.neuron_utils (*module*), 319
allensdk.internal.model.biophysical.passive_fitting.output_grabber (*module*), 319
allensdk.internal.model.biophysical.passive_fitting.passive (*module*), 318
allensdk.internal.model.biophysical.passive_fitting.preprocess (*module*), 319
allensdk.internal.model.biophysical.run_optimize (*module*), 321
allensdk.internal.model.biophysical.run_optimize_workflow (*module*), 322
allensdk.internal.model.biophysical.run_passive_fit (*module*), 322
allensdk.internal.model.biophysical.run_simulate_lims (*module*), 322
allensdk.internal.model.biophysical.run_simulate_workflow (*module*), 322
allensdk.internal.model.data_access (*module*), 334
allensdk.internal.model.glif (*module*), 333
allensdk.internal.model.glif.are_two_lists_of_arrays_the_same (*module*), 323
allensdk.internal.model.glif.error_functions (*module*), 323
allensdk.internal.model.glif.find_spikes (*module*), 323
allensdk.internal.model.glif.find_sweeps (*module*), 323
allensdk.internal.model.glif.glif_experiment (*module*), 324
allensdk.internal.model.glif.glif_optimizer (*module*), 326
allensdk.internal.model.glif.glif_optimizer_neuron (*module*), 326
allensdk.internal.model.glif.MLIN (*module*), 322
allensdk.internal.model.glif.plotting (*module*), 329
allensdk.internal.model.glif.rc (*module*), 330
allensdk.internal.model.glif.spike_cutting (*module*), 330
allensdk.internal.model.glif.threshold_adaptation (*module*), 331
allensdk.internal.model.GLM (*module*), 334
allensdk.internal.morphology (*module*), 342
allensdk.internal.morphology.compartment (*module*), 335
allensdk.internal.morphology.morphology (*module*), 335
allensdk.internal.morphology.morphvis (*module*), 339
allensdk.internal.morphology.node (*module*), 342
allensdk.internal.morphology.validate_swc (*module*), 342
allensdk.internal.mouse_connectivity (*module*), 351
allensdk.internal.mouse_connectivity.interval_unio (*module*), 347
allensdk.internal.mouse_connectivity.interval_unio (*module*), 343
allensdk.internal.mouse_connectivity.interval_unio (*module*), 343
allensdk.internal.mouse_connectivity.interval_unio (*module*), 345
allensdk.internal.mouse_connectivity.interval_unio (*module*), 345
allensdk.internal.mouse_connectivity.interval_unio (*module*), 346
allensdk.internal.mouse_connectivity.interval_unio (*module*), 346
allensdk.internal.mouse_connectivity.projection_thu (*module*), 349
allensdk.internal.mouse_connectivity.projection_thu (*module*), 347
allensdk.internal.mouse_connectivity.projection_thu (*module*), 347
allensdk.internal.mouse_connectivity.projection_thu (*module*), 348
allensdk.internal.mouse_connectivity.projection_thu (*module*), 348
allensdk.internal.mouse_connectivity.projection_thu (*module*), 349
allensdk.internal.mouse_connectivity.projection_thu (*module*), 349
allensdk.internal.mouse_connectivity.tissuecyte_sti (*module*), 351
allensdk.internal.mouse_connectivity.tissuecyte_sti (*module*), 349
allensdk.internal.mouse_connectivity.tissuecyte_sti (*module*), 350
allensdk.internal.pipeline_modules (*module*), 359
allensdk.internal.pipeline_modules.gbm (*module*), 352
allensdk.internal.pipeline_modules.gbm.generate_gbm (*module*), 351
allensdk.internal.pipeline_modules.gbm.generate_gbm (*module*), 351
allensdk.internal.pipeline_modules.gbm.generate_gbm (*module*), 352
allensdk.internal.pipeline_modules.run_annotated_re

(*module*), 352
allensdk.internal.pipeline_modules.run_demixing (*module*), 352
allensdk.internal.pipeline_modules.run_dff_computation (*module*), 352
allensdk.internal.pipeline_modules.run_neuropil_correction (*module*), 353
allensdk.internal.pipeline_modules.run_observatory_analysis (*module*), 353
allensdk.internal.pipeline_modules.run_observatory_thumbnails (*module*), 354
allensdk.internal.pipeline_modules.run_ophys_eye_calibration (*module*), 356
allensdk.internal.pipeline_modules.run_ophys_session_decomposition (*module*), 356
allensdk.internal.pipeline_modules.run_ophys_time_sync (*module*), 357
allensdk.internal.pipeline_modules.run_tissuecyte_[illegible]_from_json (*module*), 359
allensdk.model (*module*), 373
allensdk.model.biophys_sim (*module*), 361
allensdk.model.biophys_sim.bps_command (*module*), 360
allensdk.model.biophys_sim.config (*module*), 360
allensdk.model.biophys_sim.neuron (*module*), 360
allensdk.model.biophys_sim.neuron.hoc_utils (*module*), 359
allensdk.model.biophys_sim.scripts (*module*), 360
allensdk.model.biophysical (*module*), 363
allensdk.model.biophysical.run_simulate (*module*), 361
allensdk.model.biophysical.runner (*module*), 361
allensdk.model.biophysical.utils (*module*), 362
allensdk.model.glif (*module*), 373
allensdk.model.glif.glif_neuron (*module*), 363
allensdk.model.glif.glif_neuron_methods (*module*), 367
allensdk.model.glif.simulate_neuron (*module*), 372
allensdk.morphology (*module*), 373
allensdk.morphology.validate_swc (*module*), 373
allensdk.mouse_connectivity (*module*), 381
allensdk.mouse_connectivity.grid (*module*), 381
allensdk.mouse_connectivity.grid.image_series_gridder (*module*), 380
allensdk.mouse_connectivity.grid.subimage (*module*), 377
allensdk.mouse_connectivity.grid.subimage.base_subimage (*module*), 374
allensdk.mouse_connectivity.grid.subimage.cav_subimage (*module*), 375
allensdk.mouse_connectivity.grid.subimage.classic_subimage (*module*), 376
allensdk.mouse_connectivity.grid.subimage.count_subimage (*module*), 377
allensdk.mouse_connectivity.grid.utilities (*module*), 379
allensdk.mouse_connectivity.grid.utilities.downsampling_utilities (*module*), 377
allensdk.mouse_connectivity.grid.utilities.image_utilities (*module*), 378
allensdk.mouse_connectivity.grid.writers (*module*), 380
allensdk.test_utilities (*module*), 382
allensdk.test_utilities.custom_comparators (*module*), 381
allensdk.test_utilities.regression_fixture (*module*), 381
allensdk.test_utilities.temp_dir (*module*), 381
allocate_by_vsync() (*in module allensdk.brain_observatory.ecephys.stimulus_sync*), 185
alpha_filter() (*in module allensdk.brain_observatory.r_neuropil*), 213
analog_meta_data (*allensdk.brain_observatory.sync_dataset.Dataset attribute*), 229
analyze_trough_details() (*in module allensdk.ephys.ephys_features*), 280
ancestor_ids() (*allensdk.core.simple_tree.SimpleTree method*), 265
ancestor_ids() (*allensdk.internal.core.simpletree.SimpleTree method*), 311
ancestors() (*allensdk.core.simple_tree.SimpleTree method*), 265
ancestors() (*allensdk.internal.core.simpletree.SimpleTree method*), 311
angle_lines() (*in module allensdk.brain_observatory.circle_plots*), 202
angular_wheel_rotation (*allensdk.brain_observatory.ecephys.file_io.stim_file.CamStimOnePickleStimFile attribute*), 172
angular_wheel_velocity (*allensdk.brain_observatory.ecephys.file_io.stim_file.CamStimOnePickleStimFile attribute*), 173
annotate_change_detect() (*in module allensdk.brain_observatory.behavior.mtrain*),

144
annotate_trials() (in module allensdk.brain_observatory.behavior.mtrain), 144
AnnotatedSectionDataSetsApi (class in allensdk.api.queries.annotated_section_data_sets_api), 68
ANNOTATION_KEY (allensdk.core.reference_space_cache.ReferenceSpaceCache attribute), 263
Api (class in allensdk.api.api), 103
APICAL_DENDRITE (allensdk.core.swc.Morphology attribute), 272
APICAL_DENDRITE (allensdk.internal.morphology.morphology.Morphology attribute), 335
ApiPrerelease (class in allensdk.internal.api.api_prerelease), 293
append() (allensdk.core.swc.Morphology method), 272
append() (allensdk.internal.morphology.morphology.Morphology method), 335
append() (allensdk.internal.mouse_connectivity.projection_thumbnail.image_sheet.ImageSheet method), 347
append_experiment_metrics() (allensdk.brain_observatory.session_analysis.SessionAnalysis method), 217
append_metadata() (allensdk.brain_observatory.session_analysis.SessionAnalysis method), 217
append_metrics_drifting_grating() (allensdk.brain_observatory.session_analysis.SessionAnalysis method), 217
append_metrics_locally_sparse_noise() (allensdk.brain_observatory.session_analysis.SessionAnalysis method), 217
append_metrics_natural_movie_one() (allensdk.brain_observatory.session_analysis.SessionAnalysis method), 217
append_metrics_natural_movie_three() (allensdk.brain_observatory.session_analysis.SessionAnalysis method), 217
append_metrics_natural_movie_two() (allensdk.brain_observatory.session_analysis.SessionAnalysis method), 217
append_metrics_natural_scene() (allensdk.brain_observatory.session_analysis.SessionAnalysis method), 217
append_metrics_static_grating() (allensdk.brain_observatory.session_analysis.SessionAnalysis method), 217
append_threshold_components() (allensdk.model.glif.glif_neuron.GlifNeuron method), 365
append_well_known_file() (in module allensdk.internal.core.lims_utilities), 309
ApplicationConfig (class in allensdk.config.app.application_config), 232
apply() (allensdk.internal.mouse_connectivity.projection_thumbnail.ima method), 347
apply_affine() (allensdk.core.swc.Morphology method), 272
apply_affine() (allensdk.internal.morphology.morphology.Morphology method), 335
apply_affine_only_rotation() (allensdk.internal.morphology.morphology.Morphology method), 336
apply_average_tile() (allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile method), 350
apply_average_tile_to_self() (allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile method), 350
apply_colormap() (in module allensdk.internal.mouse_connectivity.projection_thumbnail.generat 347
apply_configuration_from_command_line() (allensdk.config.app.application_config.ApplicationConfig method), 232
apply_configuration_from_environment() (allensdk.config.app.application_config.ApplicationConfig method), 233
apply_configuration_from_file() (allensdk.config.app.application_config.ApplicationConfig method), 233
apply_display_sequence() (in module allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spike 177
apply_divisions() (in module allensdk.mouse_connectivity.grid.utilities.downsampling_utilities), 377
apply_frame_times() (in module allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spike 177
apply_mask() (allensdk.mouse_connectivity.grid.subimage.base_subima method), 375
apply_pixel_counter() (allensdk.mouse_connectivity.grid.subimage.base_subimage.SubIma method), 375
ARA_NISSL (allensdk.api.queries.reference_space_api.ReferenceSpaceAp attribute), 87
archive_cell() (allensdk.internal.model.biophysical.biophysical_archiver.Biophysic method), 320
are_two_lists_of_arrays_the_same() (in module allensdk.internal.model.glif.are_two_lists_of_arrays_the_same),

323
arg_parser() (*in module allensdk.internal.model.biophysical.passive_fitting.neuron_passive_fit*), 319
args (*allensdk.internal.core.lims_pipeline_module.PipelineModule attribute*), 309
ArgSchemaParserPlus (*class in allensdk.brain_observatory.argschema_utilities*), 197
ARRAY (*allensdk.api.queries.connected_services.ConnectedServices attribute*), 76
as_dataframe() (*allensdk.config.manifest.Manifest method*), 239
as_dataframe() (*allensdk.config.manifest_builder.ManifestBuilder method*), 240
as_dict() (*allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor method*), 276
as_dict() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 276
aspect_ratio (*allensdk.brain_observatory.stimulus_info.Monitor attribute*), 224
assert_exists() (*in module allensdk.internal.pipeline_modules.run_demixing*), 352
assign_session_id() (*in module allensdk.brain_observatory.behavior.mtrain*), 144
assign_sweep_values() (*in module allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes*), 178
assign_to_last() (*in module allensdk.brain_observatory.ecephys.stimulus_sync*), 185
atlas_image_query() (*allensdk.api.queries.image_download_api.ImageDownloadApi method*), 79
autocorr() (*in module allensdk.internal.model.glif.MLIN*), 322
average_rate() (*in module allensdk.ephys.ephys_features*), 281
AVERAGE_TEMPLATE (*allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute*), 87
average_tile_is_untrimmed() (*allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile method*), 350
average_voltage() (*in module allensdk.ephys.ephys_features*), 281
AXON (*allensdk.core.swc.Morphology attribute*), 272
AXON (*allensdk.internal.morphology.morphology.Morphology attribute*), 335

B

background_trace() (*in module allensdk.internal.brain_observatory.demix_report*), 295
barcode_line (*allensdk.brain_observatory.ecephys.align_timestamps.b attribute*), 167
BarcodeSyncDataset (*class in allensdk.brain_observatory.ecephys.align_timestamps.barcode_syn*
167
BASAL_DENDRITE (*allensdk.core.swc.Morphology attribute*), 272
BASAL_DENDRITE (*allensdk.internal.morphology.morphology.Morphology attribute*), 335
base_object_to_eye_rotation_matrix() (*in module allensdk.internal.brain_observatory.eye_calibration*), 299
bb_dist() (*in module allensdk.internal.brain_observatory.mask_set*), 303
behavior_alignment (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSync attribute*), 357
behavior_delta (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSync attribute*), 357
behavior_session_id (*allensdk.brain_observatory.behavior.metadata.behavior_metadata. attribute*), 110
behavior_session_uuid (*allensdk.brain_observatory.behavior.metadata.behavior_metadata. attribute*), 110
behavior_times (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSync attribute*), 357
BEHAVIOR_TRACKING_KEYS (*allensdk.brain_observatory.sync_dataset.Dataset attribute*), 228
behavior_video_timestamps (*allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute*), 307
BehaviorBase (*class in allensdk.brain_observatory.behavior.session_apis.abcs.session_bas*
115
BehaviorDataExtractorBase (*class in allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac*
113
BehaviorDataTransforms (*class in allensdk.brain_observatory.behavior.session_apis.data_transforms.*
122
BehaviorJsonApi (*class in allensdk.brain_observatory.behavior.session_apis.data_io.behavior*
119

BehaviorJsonExtractor (*class in allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api*), 119
BehaviorLimsApi (*class in allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api*), 120
BehaviorLimsExtractor (*class in allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api*), 120
BehaviorMetadata (*class in allensdk.brain_observatory.behavior.metadata.behavior_metadata*), 110
BehaviorMetadataSchema (*class in allensdk.brain_observatory.behavior.schemas*), 147
BehaviorOphysBase (*class in allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base*), 117
BehaviorOphysDataExtractorBase (*class in allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_ophys_data_extractor_base*), 114
BehaviorOphysDataTransforms (*class in allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms*), 124
BehaviorOphysMetadata (*class in allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata*), 112
BehaviorOphysSessionsTable (*class in allensdk.brain_observatory.behavior.behavior_project_cache.tables.ophys_sessions_table*), 109
BehaviorTaskParametersSchema (*class in allensdk.brain_observatory.behavior.schemas*), 148
best_fit_value() (*allensdk.internal.model.biophysical.make_deap_fit_json.Report method*), 321
BIC() (*in module allensdk.internal.model.AIC*), 334
binarize() (*allensdk.mouse_connectivity.grid.subimage.base_subimage.SubImage method*), 375
BinaryIntervalSearchTree (*class in allensdk.brain_observatory.stimulus_info*), 224
binned_cells_sp (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 222
binned_cells_vis (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 222
binned_dx_sp (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 222
binned_dx_vis (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 222
BIOPHYSICAL_MODEL_TYPE_IDS (*allensdk.api.queries.biophysical_api.BiophysicalApi attribute*), 69
BiophysicalApi (*class in allensdk.api.queries.biophysical_api*), 69
BiophysicalArchiver (*class in allensdk.internal.model.biophysical.biophysical_archiver*), 320
BiophysicalModuleApi (*class in allensdk.internal.api.queries.biophysical_module_api*), 288
BiophysicalModuleReader (*class in allensdk.internal.api.queries.biophysical_module_reader*), 289
blend() (*in module allensdk.internal.mouse_connectivity.projection_thumbnail.visualiz*), 348
blend_component_from_point() (*in module allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher*), 349
blend_with_background() (*in module allensdk.internal.mouse_connectivity.projection_thumbnail.generat*), 347
block_apply() (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 378
block_average() (*in module allensdk.mouse_connectivity.grid.utilities.downsampling_utilities*), 377
BOOLEAN (*allensdk.api.queries.connected_services.ConnectedServices attribute*), 76
BrainObservatoryAnalysisException, 200
BrainObservatoryApi (*class in allensdk.api.queries.brain_observatory_api*), 71
BrainObservatoryMonitor (*class in allensdk.brain_observatory.stimulus_info*), 224
BrainObservatoryNwbDataSet (*class in allensdk.core.brain_observatory_nwb_data_set*), 242
bucket_name_from_url() (*in module allensdk.api.cloud_cache.utils*), 67
build_affine_transform() (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 378
build_cell_plots() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 354
build_grids() (*allensdk.mouse_connectivity.grid.image_series_gridder.ImageSerie method*), 380
build_colormap() (*in module allensdk.brain_observatory.ecephys.stimulus_table.visualization.vie*), 176

build_composite_transform() (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 378
build_correlation_plots() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 354
build_drifting_gratings() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 354
build_experiment_thumbnails() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 354
build_eye_tracking_plots() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 354
build_from_image() (*allensdk.internal.mouse_connectivity.projection_thumbnail.image_sheet.ImageSheet static method*), 347
build_hex_pack() (*in module allensdk.brain_observatory.circle_plots*), 202
build_locally_sparse_noise() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 354
build_manifest() (*allensdk.api.warehouse_cache.cache.Cache method*), 97
build_manifest() (*allensdk.core.cell_types_cache.CellTypesCache method*), 247
build_manifest() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 291
build_natural_movie() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 354
build_natural_scenes() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 355
build_plots() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 355
build_query() (*allensdk.api.queries.svg_api.SvgApi method*), 95
build_query_url() (*allensdk.api.queries.rma_api.RmaApi method*), 90
build_receptive_field() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 355
build_reference_aligned_image_channel_volumes_url() (*allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method*), 83
build_rma() (*allensdk.api.queries.biophysical_api.BiophysicalApi method*), 69
build_rotation_transform() (*allensdk.internal.mouse_connectivity.projection_thumbnail.volume_ method*), 349
build_schema_query() (*allensdk.api.queries.rma_api.RmaApi method*), 90
build_speed_tuning() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 355
build_static_gratings() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 355
build_stimuluswise_table() (*in module allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spike*), 178
build_trial_matrix() (*in module allensdk.brain_observatory.receptive_field_analysis.chisquarerf*), 187
build_type() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 355
build_url() (*allensdk.api.queries.connected_services.ConnectedServic method*), 76
build_volumetric_data_download_url() (*allensdk.api.queries.reference_space_api.ReferenceSpaceApi method*), 88
burst_metrics() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 276

C

Cache (*class in allensdk.api.warehouse_cache.cache*), 97
cache_clear() (*allensdk.core.cache_method_utilities.CachedInstanceMethodMixin method*), 247
cache_csv() (*allensdk.api.warehouse_cache.cache.Cache static method*), 97
cache_csv_dataframe() (*allensdk.api.warehouse_cache.cache.Cache static method*), 97
cache_csv_json() (*allensdk.api.warehouse_cache.cache.Cache static method*), 97
cache_data() (*allensdk.api.queries.biophysical_api.BiophysicalApi method*), 69
cache_json() (*allensdk.api.warehouse_cache.cache.Cache static method*), 97
cache_json_dataframe() (*allensdk.api.warehouse_cache.cache.Cache static method*), 97
cache_stimulus_file() (*allensdk.api.queries.glif_api.GlifApi method*),

76
cacheable() (in module allensdk.api.warehouse_cache.cache), 100
CachedInstanceMethodMixin (class in allensdk.core.cache_method_utilities), 247
CacheFileAttributes (class in allensdk.api.cloud_cache.file_attributes), 65
cacher() (allensdk.api.warehouse_cache.cache.Cache static method), 97
calc_deriv() (in module allensdk.brain_observatory.behavior.running_processing), 145
calc_spike_component_of_threshold_from_multiple() (in module allensdk.internal.model.glif.threshold_adaptation), 331
calc_spike_cut_and_v_reset_via_expvar_residuals() (in module allensdk.internal.model.glif.spike_cutting), 330
calculate() (allensdk.core.lazy_property.lazy_property.LazyProperty method), 241
calculate() (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record.TissuecyteInjectionUnionize method), 346
calculate() (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record.TissuecyteProjectionUnionize method), 346
calculate() (allensdk.internal.mouse_connectivity.interval_unionize.unionize_record.Unionize method), 346
calculate_delay() (in module allensdk.brain_observatory.behavior.sync.process_sync), 131
calculate_dff() (in module allensdk.brain_observatory.dff), 204
calculate_dvdt() (in module allensdk.ephys.ephys_features), 281
calculate_feature_errors() (allensdk.internal.model.biophysical.deap_utils.Utils method), 320
calculate_fi_curves() (in module allensdk.internal.model.biophysical.check_fi_shift), 320
calculate_injection_centroid() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 83
calculate_max_border() (in module allensdk.internal.brain_observatory.roi_filter_utils), 306
calculate_monitor_delay() (in module allensdk.internal.brain_observatory.time_sync), 307
calculate_response_latency_list() (in module allensdk.brain_observatory.behavior.trials_processing), 159
calculate_reward_rate() (in module allensdk.brain_observatory.behavior.trials_processing), 159
calculate_reward_rate_fix_nans() (in module allensdk.brain_observatory.behavior.trials_processing), 159
calculate_roi_and_neuropil_traces() (in module allensdk.brain_observatory.roi_masks), 215
calculate_scale() (in module allensdk.internal.morphology.morphvis), 339
calculate_traces() (in module allensdk.brain_observatory.roi_masks), 215
calculate_trough() (allensdk.ephys.feature_extractor.EphysFeatureExtractor method), 288
call_caching() (in module allensdk.api.warehouse_cache.caching_utilities), 101
CamStimOnePickleStimFile (class in allensdk.brain_observatory.ecephys.file_io.stim_file), 172
categorize_one_trial() (in module allensdk.brain_observatory.behavior.trials_processing), 160
cav_writer() (in module allensdk.mouse_connectivity.grid.writers), 380
CavSubImage (class in allensdk.mouse_connectivity.grid.subimage.cav_subimage), 375
CCF_2015 (allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute), 87
CCF_2016 (allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute), 87
CCF_2017 (allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute), 87
CCF_VERSION_DEFAULT (allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute), 87
cell_extractor_for_nwb() (in module allensdk.ephys.ephys_extractor), 279
cell_features() (allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor method), 276
cell_id (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute), 222
cell_index_receptive_field_analysis_data (allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise attribute), 208
CELL_MAPPING_ID (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi attribute), 71

CELLS_KEY (*allensdk.core.cell_types_cache.CellTypesCache attribute*), 247
celltraces (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 222
CellTypesApi (*class in allensdk.api.queries.cell_types_api*), 74
CellTypesCache (*class in allensdk.core.cell_types_cache*), 247
change_parent() (*allensdk.core.swc.Morphology method*), 272
change_parent() (*allensdk.internal.morphology.morphology.Morphology method*), 336
check_and_write() (*allensdk.core.reference_space.ReferenceSpace static method*), 260
check_coverage() (*allensdk.core.reference_space.ReferenceSpace method*), 260
check_dir() (*allensdk.config.manifest.Manifest method*), 239
check_org_selections_for_noise_block() (*allensdk.internal.model.biophysical.make_deap_fit_json.Report method*), 321
check_read_access() (*in module allensdk.brain_observatory.argschema_utilities*), 199
check_stimulus_delay() (*in module allensdk.internal.pipeline_modules.run_ophys_time_sync*), 358
check_thresholds_and_peaks() (*in module allensdk.ephys.ephys_features*), 281
check_write_access() (*in module allensdk.brain_observatory.argschema_utilities*), 199
check_write_access_dir() (*in module allensdk.brain_observatory.argschema_utilities*), 199
check_write_access_overwrite() (*in module allensdk.brain_observatory.argschema_utilities*), 199
checkPreprocess() (*in module allensdk.internal.model.glif.plotting*), 329
checkSpikeCutting() (*in module allensdk.internal.model.glif.plotting*), 329
chi_square_binary() (*in module allensdk.brain_observatory.receptive_field_analysis.chisquarerf*), 188
chi_square_within_mask() (*in module allensdk.brain_observatory.receptive_field_analysis.chisquarerf*), 188
child_ids() (*allensdk.core.simple_tree.SimpleTree method*), 265
child_ids() (*allensdk.internal.core.simpletree.SimpleTree method*), 311
children() (*allensdk.core.simple_tree.SimpleTree method*), 266
children() (*allensdk.internal.core.simpletree.SimpleTree method*), 311
children_of() (*allensdk.core.swc.Morphology method*), 272
children_of() (*allensdk.internal.morphology.morphology.Morphology method*), 336
chisq_from_stim_table() (*in module allensdk.brain_observatory.chisquare_categorical*), 200
choose_bps_command() (*in module allensdk.model.biophys_sim.bps_command*), 360
choose_inliers() (*allensdk.internal.brain_observatory.fit_ellipse.FitEllipse method*), 300
class_deprecated() (*in module allensdk.deprecated*), 382
clean_structures() (*allensdk.core.structure_tree.StructureTree static method*), 269
cleanup_truncated_file() (*allensdk.api.api.Api method*), 103
clear_updated_params() (*allensdk.brain_observatory.session_api_utils.ParamsMixin method*), 220
clipped_stim_timestamps (*allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute*), 307
clone() (*allensdk.ephys.feature_extractor.EphysFeatures method*), 288
clone() (*allensdk.internal.morphology.morphology.Morphology method*), 336
close() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 229
close() (*allensdk.internal.brain_observatory.frame_stream.CvInputStrea method*), 301
close() (*allensdk.internal.brain_observatory.frame_stream.FfmpegInput method*), 301
close() (*allensdk.internal.brain_observatory.frame_stream.FfmpegOutpu method*), 301
close() (*allensdk.internal.brain_observatory.frame_stream.FrameInputS method*), 301
close() (*allensdk.internal.brain_observatory.frame_stream.FrameOutpu

method), 301

`close()` (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303

`close_sets()` (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303

`collapse_columns()` (*in module allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities*), 181

`collect_sets()` (*allensdk.core.structure_tree.StructureTree static method*), 269

`colormap()` (*in module allensdk.brain_observatory.behavior.trials_processing*), 160

`COLORMAPS` (*allensdk.api.queries.image_download_api.ImageDownloadApi attribute*), 79

`compare_session_fields()` (*in module allensdk.brain_observatory.session_api_utils*), 220

`Compartment` (*class in allensdk.core.swc*), 271

`Compartment` (*class in allensdk.internal.morphology.compartment*), 335

`compartment()` (*allensdk.internal.morphology.morphology.Morphology method*), 336

`compartment_index` (*allensdk.core.swc.Morphology attribute*), 273

`compartment_index_by_type()` (*allensdk.core.swc.Morphology method*), 273

`compartment_list` (*allensdk.core.swc.Morphology attribute*), 273

`compartment_list` (*allensdk.internal.morphology.morphology.Morphology attribute*), 337

`compartment_list_by_type()` (*allensdk.core.swc.Morphology method*), 273

`CompleteOphysBehaviorMetadataSchema` (*class in allensdk.brain_observatory.behavior.schemas*), 149

`compute()` (*allensdk.brain_observatory.ecephys.align_timestamps.probe_synchronizer.ProbeSynchronizer class method*), 169

`compute_area()` (*allensdk.internal.brain_observatory.eye_calibration.EyeCalibration method*), 298

`compute_chi()` (*in module allensdk.brain_observatory.chisquare_categorical*), 200

`compute_chi_shuffle()` (*in module allensdk.brain_observatory.chisquare_categorical*), 201

`compute_circular_area()` (*in module allensdk.brain_observatory.behavior.eye_tracking_processing*), 139

`compute_coarse_parameters()` (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 378

`compute_coarse_planes()` (*allensdk.mouse_connectivity.grid.subimage.base_subimage.SubIma method*), 375

`compute_coarse_planes()` (*allensdk.mouse_connectivity.grid.subimage.cav_subimage.CavSubI method*), 375

`compute_coarse_planes()` (*allensdk.mouse_connectivity.grid.subimage.classic_subimage.Class method*), 376

`compute_coarse_planes()` (*allensdk.mouse_connectivity.grid.subimage.count_subimage.Count method*), 377

`compute_correlations()` (*in module allensdk.internal.brain_observatory.demix_report*), 295

`compute_correlations_without_masks()` (*in module allensdk.internal.brain_observatory.demix_report*), 295

`compute_dff_windowed_median()` (*in module allensdk.brain_observatory.dff*), 204

`compute_dff_windowed_mode()` (*in module allensdk.brain_observatory.dff*), 205

`compute_distance()` (*in module allensdk.brain_observatory.receptive_field_analysis.fit_parameters*), 191

`compute_elliptical_area()` (*in module allensdk.brain_observatory.behavior.eye_tracking_processing*), 139

`compute_expected()` (*in module allensdk.brain_observatory.chisquare_categorical*), 201

`compute_frame_times()` (*in module allensdk.brain_observatory.ecephys.stimulus_sync*), 185

`compute_injection()` (*allensdk.mouse_connectivity.grid.subimage.classic_subimage.Class method*), 376

`compute_injection()` (*allensdk.mouse_connectivity.grid.subimage.count_subimage.Count method*), 377

`compute_intensity()` (*allensdk.mouse_connectivity.grid.subimage.classic_subimage.Class method*), 376

`compute_non_overlap_masks()` (*in module allensdk.internal.brain_observatory.demix_report*), 296

`compute_non_overlap_traces()` (*in module allensdk.internal.brain_observatory.demix_report*), 296

compute_observed() (in module allensdk.brain_observatory.chisquare_categorical), 201
compute_overlap() (in module allensdk.brain_observatory.receptive_field_analysis.fit_parameters), 191
compute_projection() (allensdk.mouse_connectivity.grid.subimage.classic_subimage.ClassicSubImage method), 376
compute_projection() (allensdk.mouse_connectivity.grid.subimage.count_subimage.CountSubImage method), 377
compute_receptive_field() (in module allensdk.brain_observatory.receptive_field_analysis.receptive_field), 192
compute_receptive_field_with_postprocessing() (in module allensdk.brain_observatory.receptive_field_analysis.receptive_field), 192
compute_sum_pixels() (allensdk.mouse_connectivity.grid.subimage.classic_subimage.ClassicSubImage method), 376
compute_sum_pixels() (allensdk.mouse_connectivity.grid.subimage.count_subimage.CountSubImage method), 377
Config (class in allensdk.model.biophys_sim.config), 360
configure_library_method() (allensdk.model.glif.glif_neuron.GlifNeuron static method), 365
configure_method() (allensdk.model.glif.glif_neuron.GlifNeuron static method), 365
connect() (in module allensdk.internal.core.lims_utilities), 309
ConnectedServices (class in allensdk.api.queries.connected_services), 76
consistency_is_key() (in module allensdk.brain_observatory.behavior.criteria), 136
consistent_behavior_within_session() (in module allensdk.brain_observatory.behavior.criteria), 136
construct_rolling_performance_df() (in module allensdk.brain_observatory.behavior.trials_processing), 160
construct_well_known_file_download_url() (allensdk.api.api.Api method), 103
consume_volume() (allensdk.mouse_connectivity.grid.image_series_gridder.ImageSeriesGridder method), 380
contingent_trials() (in module allensdk.brain_observatory.behavior.trial_masks), 158
ContinuousFile (class in allensdk.brain_observatory.ecephys.file_io.continuous_file), 172
conv() (in module allensdk.mouse_connectivity.grid.utilities.downsampling_utilities), 377
convert_axis() (in module allensdk.internal.mouse_connectivity.projection_thumbnail.projecti 348
convert_discrete_colormap() (in module allensdk.internal.mouse_connectivity.projection_thumbnail.visualiz 348
convert_filepath_caseinsensitive() (in module allensdk.brain_observatory.behavior.stimulus_processing.util), 129
convert_frame() (in module allensdk.internal.pipeline_modules.run_ophys_session_decomposit 356
convert_from_titan_linux() (in module allensdk.internal.core.lims_utilities), 309
convert_type() (allensdk.core.swc.Morphology method), 273
convert_type() (allensdk.internal.morphology.morphology.Morphology method), 337
convolve() (in module allensdk.brain_observatory.receptive_field_analysis.utilities), 193
copy() (allensdk.internal.mouse_connectivity.projection_thumbnail.image method), 348
copy_local() (allensdk.internal.model.biophysical.run_optimize.RunO method), 321
copy_local() (allensdk.internal.model.biophysical.run_simulate_lims.I method), 322
CoronaPlotter (class in allensdk.brain_observatory.circle_plots), 201
correct_on_off_effects() (in module allensdk.brain_observatory.ecephys.stimulus_sync), 186
corrected_behavior_video_timestamps (allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute), 307
corrected_eye_video_timestamps (allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute), 307
corrected_ophys_timestamps (allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute), 307
corrected_stim_timestamps (allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute), 307

corrected_video_timestamps() (*in module allensdk.internal.brain_observatory.time_sync*), 308
correlation_report() (*in module allensdk.internal.brain_observatory.demix_report*), 296
COUNT (*allensdk.api.queries.rma_api.RmaApi attribute*), 89
count (*allensdk.internal.brain_observatory.mask_set.MaskSet attribute*), 303
count_writer() (*in module allensdk.mouse_connectivity.grid.writers*), 380
CountSubImage (*class in allensdk.mouse_connectivity.grid.subimage.count_subimage*), 377
cr_position_in_mouse_eye_coordinates() (*allensdk.internal.brain_observatory.eye_calibration.EyeCalibration static method*), 298
cre_line (*allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata attribute*), 110
create_argparser() (*allensdk.config.app.application_config.ApplicationConfig method*), 233
create_basis_IPSP() (*in module allensdk.internal.model.GLM*), 334
create_dir() (*allensdk.config.manifest.Manifest method*), 239
create_extended_trials() (*in module allensdk.brain_observatory.behavior.trials_processing*), 160
create_fake_metadata() (*in module allensdk.internal.pipeline_modules.run_ophys_session_decomposition*), 356
create_gene_fpkm_table() (*in module allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap*), 351
create_genes_for_transcripts() (*in module allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap*), 351
create_image() (*in module allensdk.internal.morphology.morphvis*), 340
create_images() (*allensdk.internal.brain_observatory.frame_stream.FfmpegInputStream method*), 301
create_images() (*allensdk.internal.brain_observatory.frame_stream.FrameInputStream method*), 301
create_manifest() (*allensdk.api.queries.biophysical_api.BiophysicalApi method*), 69
create_neuropil_mask() (*in module allensdk.brain_observatory.roi_masks*), 215
create_region_mask() (*in module allensdk.internal.brain_observatory.annotated_region_metrics*), 294
create_roi_mask() (*in module allensdk.brain_observatory.roi_masks*), 215
create_roi_mask_array() (*in module allensdk.brain_observatory.roi_masks*), 216
create_sample_metadata() (*in module allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap*), 351
create_stim_table() (*in module allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spik*), 179
create_transcript_fpkm_table() (*in module allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap*), 351
create_transcripts_for_genes() (*in module allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap*), 351
create_utils() (*in module allensdk.model.biophysical.utils*), 363
credential_injector() (*in module allensdk.core.authentication*), 242
CredentialProvider (*class in allensdk.core.authentication*), 241
CRITERIA (*allensdk.api.queries.rma_api.RmaApi attribute*), 89
CRITERIA() (*in module allensdk.internal.brain_observatory.roi_filter_utils*), 305
csv_converter() (*allensdk.api.warehouse_cache.cache.Cache static method*), 98
CUT_DENDRITE (*allensdk.core.swc.Marker attribute*), 271
CUT_DENDRITE (*allensdk.internal.core.swc.Marker attribute*), 312
CvInputStream (*class in allensdk.internal.brain_observatory.frame_stream*), 301

D

data (*allensdk.brain_observatory.behavior.image_api.Image attribute*), 142
data_file_attributes() (*allensdk.api.cloud_cache.manifest.Manifest method*), 66
DATA_MASK (*allensdk.api.queries.grid_data_api.GridDataApi attribute*), 77
DATA_MASK_KEY (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute*), 252
data_to_licks() (*in module al-*

lensdk.brain_observatory.behavior.trials_processing), 161
data_to_metadata() (in module allensdk.brain_observatory.behavior.trials_processing), 161
data_transforms() (allensdk.core.structure_tree.StructureTree static method), 269
dataframe_query() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 71
dataframe_query_string() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 71
DataFrameIndexError, 250
DataFrameKeyError, 250
dataset (allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute), 307
Dataset (class in allensdk.brain_observatory.sync_dataset), 228
date_of_acquisition (allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata attribute), 110
DatUtilities (class in allensdk.core.dat_utilities), 249
db_connection_creator() (in module allensdk.internal.api), 293
DbCredentials (class in allensdk.core.authentication), 241
dbname (allensdk.core.authentication.DbCredentials attribute), 241
DEBUG (allensdk.api.queries.rma_api.RmaApi attribute), 89
debug() (in module allensdk.internal.pipeline_modules.run_annotated_region_metrics), 352
debug() (in module allensdk.internal.pipeline_modules.run_demixing), 352
debug() (in module allensdk.internal.pipeline_modules.run_neuropil_correction), 353
debug() (in module allensdk.internal.pipeline_modules.run_observatory_analysis), 353
debug() (in module allensdk.internal.pipeline_modules.run_observatory_thumbnails), 355
debug() (in module allensdk.internal.pipeline_modules.run_ophys_eye_calibration), 356
debug() (in module allensdk.internal.pipeline_modules.run_ophys_session_decomposition), 356
debug_clause() (allensdk.api.queries.rma_api.RmaApi method), 90
debug_plot() (in module allensdk.internal.pipeline_modules.run_neuropil_correction), 353
decision_function() (allensdk.internal.brain_observatory.roi_filter_utils.TrainingLabelC method), 305
decode_bytes() (in module allensdk.core.h5_utilities), 250
default() (allensdk.brain_observatory.JSONEncoder method), 232
default_api_url (allensdk.api.api.Api attribute), 103
default_argument_parser() (in module allensdk.internal.core.lims_pipeline_module), 309
default_ray() (in module allensdk.internal.brain_observatory.itracker_utils), 302
DEFAULT_STRUCTURE_SET_IDS (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute), 253
DEFAULT_STRUCTURE_SET_IDS (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute), 252
DEFORMATION_FIELD_HEADER_KEY (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute), 252
DEFORMATION_FIELD_VOXEL_KEY (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute), 252
deg_to_dist() (in module allensdk.brain_observatory.behavior.running_processing), 145
deinterpolate_RF() (in module allensdk.brain_observatory.receptive_field_analysis.chisquarerf), 188
delay_metrics() (allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method), 277
delete_tree() (allensdk.core.swc.Morphology method), 273
delete_tree() (allensdk.internal.morphology.morphology.Morphology method), 337
demix_time_dep_masks() (in module allensdk.brain_observatory.demixer), 203
demix_time_dep_masks() (in module allensdk.internal.brain_observatory.demixer), 296
DENDRITE (allensdk.core.swc.Morphology attribute),

272
DENSITY (*allensdk.api.queries.grid_data_api.GridDataApi attribute*), 77
deprecated() (*in module allensdk.deprecated*), 382
DEPRECATED_KEYS (*allensdk.brain_observatory.sync_dataset.Dataset attribute*), 228
DEPRECATED_SPIKE_TIMES (*allensdk.core.nwb_data_set.NwbDataSet attribute*), 257
descendant_ids() (*allensdk.core.simple_tree.SimpleTree method*), 266
descendant_ids() (*allensdk.internal.core.simpletree.SimpleTree method*), 311
descendants() (*allensdk.core.simple_tree.SimpleTree method*), 266
descendants() (*allensdk.internal.core.simpletree.SimpleTree method*), 311
Description (*class in allensdk.config.model.description*), 236
DescriptionParser (*class in allensdk.config.model.description_parser*), 237
deserialize() (*allensdk.brain_observatory.behavior.image_api.ImageApi static method*), 142
detect_bursts() (*in module allensdk.ephys.ephys_features*), 282
detect_duplicates() (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303
detect_events() (*in module allensdk.brain_observatory.receptive_field_analysis.eventdetection*), 190
detect_pauses() (*in module allensdk.ephys.ephys_features*), 283
detect_putative_spikes() (*in module allensdk.ephys.ephys_features*), 283
detect_unions() (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303
determine_likely_blinks() (*in module allensdk.brain_observatory.behavior.eye_tracking_processing*), 139
determine_outliers() (*in module allensdk.brain_observatory.behavior.eye_tracking_processing*), 140
DEVMOUSE_2012 (*allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute*), 87
DEVMOUSE_ATLAS_PRODUCTS (*allensdk.api.queries.mouse_atlas_api.MouseAtlasApi attribute*), 82
df_columns (*allensdk.config.manifest_builder.ManifestBuilder attribute*), 240
dfftraces (*allensdk.brain_observatory.stimulus_analysis.StimulusAnaly attribute*), 222
DFMFLD_RESOLUTIONS (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute*), 252
dict_generator() (*in module allensdk.brain_observatory.receptive_field_analysis.tools*), 193
dict_to_indexed_array() (*in module allensdk.brain_observatory*), 232
dim_color() (*allensdk.brain_observatory.observatory_plots.Dimension method*), 210
DimensionPatchHandler (*class in allensdk.brain_observatory.observatory_plots*), 210
DIR (*allensdk.config.manifest.Manifest attribute*), 238
DIR_CCW (*allensdk.brain_observatory.circle_plots.PolarPlotter attribute*), 202
DIR_CW (*allensdk.brain_observatory.circle_plots.PolarPlotter attribute*), 202
direct_sum_projection_pixels (*allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_u attribute*), 345
direct_unionize() (*allensdk.internal.mouse_connectivity.interval_unionize.interval_un method*), 343
direct_voxel_counts() (*allensdk.core.reference_space.ReferenceSpace method*), 260
direct_voxel_map (*allensdk.core.reference_space.ReferenceSpace attribute*), 261
DIRNAME (*allensdk.config.manifest.Manifest attribute*), 238
distance() (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303
do_blur() (*in module allensdk.internal.mouse_connectivity.projection_thumbnail.generat*), 347
do_query() (*allensdk.api.api.Api method*), 103
do_rma_query() (*allensdk.api.api.Api method*), 103
download_alignment3d() (*allensdk.api.queries.grid_data_api.GridDataApi method*), 77
download_annotation_volume() (*allensdk.api.queries.reference_space_api.ReferenceSpaceApi method*), 88
download_atlas_image() (*allensdk.api.queries.image_download_api.ImageDownloadApi*

method), 79
download_data_mask() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 83
download_data_mask() (allensdk.internal.api.queries.mouse_connectivity_api_prerelease.MouseConnectivityApiPrerelease method), 291
download_deformation_field() (allensdk.api.queries.grid_data_api.GridDataApi method), 78
download_expression_density() (allensdk.api.queries.mouse_atlas_api.MouseAtlasApi method), 82
download_expression_energy() (allensdk.api.queries.mouse_atlas_api.MouseAtlasApi method), 82
download_expression_grid_data() (allensdk.api.queries.grid_data_api.GridDataApi method), 78
download_expression_intensity() (allensdk.api.queries.mouse_atlas_api.MouseAtlasApi method), 82
download_gene_expression_grid_data() (allensdk.api.queries.grid_data_api.GridDataApi method), 78
download_image() (allensdk.api.queries.image_download_api.ImageDownloadApi method), 79
download_injection_density() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 83
download_injection_density() (allensdk.internal.api.queries.mouse_connectivity_api_prerelease.MouseConnectivityApiPrerelease method), 291
download_injection_fraction() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 83
download_injection_fraction() (allensdk.internal.api.queries.mouse_connectivity_api_prerelease.MouseConnectivityApiPrerelease method), 291
download_mouse_atlas_volume() (allensdk.api.queries.reference_space_api.ReferenceSpaceApi method), 88
download_projection_density() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 83
download_projection_density() (allensdk.internal.api.queries.mouse_connectivity_api_prerelease.MouseConnectivityApiPrerelease method), 291
download_projection_grid_data() (allensdk.api.queries.grid_data_api.GridDataApi method), 78
download_projection_grid_data() (allensdk.internal.api.queries.grid_data_api_prerelease.GridDataApiPrerelease method), 290
download_projection_image() (allensdk.api.queries.image_download_api.ImageDownloadApi method), 80
download_reference_aligned_image_channel_volumes() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivity method), 83
download_section_image() (allensdk.api.queries.image_download_api.ImageDownloadApi method), 80
download_structure_mask() (allensdk.api.queries.reference_space_api.ReferenceSpaceApi method), 88
download_structure_mesh() (allensdk.api.queries.reference_space_api.ReferenceSpaceApi method), 89
download_svg() (allensdk.api.queries.svg_api.SvgApi method), 95
download_template_volume() (allensdk.api.queries.reference_space_api.ReferenceSpaceApi method), 89
download_url (allensdk.api.api.Api attribute), 104
download_volumetric_data() (allensdk.api.queries.reference_space_api.ReferenceSpaceApi method), 89
downsample() (allensdk.core.reference_space.ReferenceSpace method), 261
downsample_average() (in module allensdk.mouse_connectivity.grid.utilities.downsampling_utilities), 378
draw_density_hist() (in module allensdk.internal.morphology.morphvis), 340
draw_morphology() (in module allensdk.internal.morphology.morphvis), 341
DriftingGratings (class in allensdk.brain_observatory.drifting_gratings), 206
driver_line (allensdk.brain_observatory.behavior.metadata.behavior_ attribute), 110
drop_empty_columns() (in module allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities 181
duty_cycle() (allensdk.brain_observatory.sync_dataset.Dataset method), 229
dxcm (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute), 222
dxcm_path (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute), 222
dynamics() (allensdk.model.glif.glif_neuron.GlifNeuron method), 365
dynamics_AScurrent_exp() (in module allensdk.model.glif.glif_neuron_methods), 367
dynamics_AScurrent_none() (in module al-

lensdk.model.glif.glif_neuron_methods), 367

dynamics_threshold_inf() (*in module allensdk.model.glif.glif_neuron_methods*), 367

dynamics_threshold_spike_component() (*in module allensdk.model.glif.glif_neuron_methods*), 368

dynamics_threshold_three_components_exact() (*in module allensdk.model.glif.glif_neuron_methods*), 368

dynamics_voltage_linear_exact() (*in module allensdk.model.glif.glif_neuron_methods*), 369

dynamics_voltage_linear_forward_euler() (*in module allensdk.model.glif.glif_neuron_methods*), 369

E

eccentricity() (*in module allensdk.internal.brain_observatory.annotated_region_metrics*), 295

eccentricity() (*in module allensdk.internal.brain_observatory.itracker_utils*), 302

EcephysProjectApi (*class in allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api*), 170

EcephysSessionApi (*class in allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api*), 171

EcephysSyncDataset (*class in allensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset*), 172

ellipse_angle_of_rotation() (*in module allensdk.internal.brain_observatory.fit_ellipse*), 300

ellipse_angle_of_rotation2() (*in module allensdk.internal.brain_observatory.fit_ellipse*), 300

ellipse_axis_length() (*in module allensdk.internal.brain_observatory.fit_ellipse*), 300

ellipse_center() (*in module allensdk.internal.brain_observatory.fit_ellipse*), 300

emission_lambda (*allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata.BehaviorOphysMetadata attribute*), 112

enable_console_log() (*in module allensdk.config*), 240

ENERGY (*allensdk.api.queries.grid_data_api.GridDataApi attribute*), 77

EnvCredentialProvider (*class in allensdk.core.authentication*), 242

EPHYS_DATA_KEY (*allensdk.core.cell_types_cache.CellTypesCache attribute*), 247

EPHYS_FEATURES_KEY (*allensdk.core.cell_types_cache.CellTypesCache attribute*), 247

EPHYS_SWEEPS_KEY (*allensdk.core.cell_types_cache.CellTypesCache attribute*), 247

EphysCellFeatureExtractor (*class in allensdk.ephys.ephys_extractor*), 276

EphysFeatureExtractor (*class in allensdk.ephys.feature_extractor*), 288

EphysFeatures (*class in allensdk.ephys.feature_extractor*), 288

EphysSweepFeatureExtractor (*class in allensdk.ephys.ephys_extractor*), 276

EphysSweepSetFeatureExtractor (*class in allensdk.ephys.ephys_extractor*), 278

EpochSeparationException, 200

EQ (*allensdk.api.queries.rma_api.RmaApi attribute*), 89

equipment_name (*allensdk.brain_observatory.behavior.metadata.behavior_metadata.B attribute*), 110

error_calc() (*in module allensdk.brain_observatory.r_neuropil*), 213

error_calc_outlier() (*in module allensdk.brain_observatory.r_neuropil*), 213

escape_char (*allensdk.internal.model.biophysical.passive_fitting.output attribute*), 319

estimate_adjusted_detection_parameters() (*in module allensdk.ephys.ephys_features*), 283

estimate_contamination_ratios() (*in module allensdk.brain_observatory.r_neuropil*), 213

estimate_error() (*allensdk.brain_observatory.r_neuropil.NeuropilSubtract method*), 212

estimate_fi_shift() (*in module allensdk.internal.model.biophysical.check_fi_shift*), 320

estimate_frame_duration() (*in module allensdk.brain_observatory.ecephys.stimulus_sync*), 186

estimate_sag() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 277

estimate_time_constant() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 277

euclidean_distance() (*in module allensdk.internal.morphology.node*), 342

evaluate() (allensdk.internal.model.glif.glif_optimizer.GlifOptimizer method), 326
events_to_pvalues_no_fdr_correction() (in module allensdk.brain_observatory.receptive_field_analysis.receptive_field), 192
EXCEPT (allensdk.api.queries.rma_api.RmaApi attribute), 89
excitation_lambda (allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata.BehaviorOphysMetadata attribute), 112
EXCPT (allensdk.api.queries.rma_api.RmaApi attribute), 89
exp_curve() (in module allensdk.internal.ephys.plot_qc_figures), 313
exp_curve() (in module allensdk.internal.ephys.plot_qc_figures3), 315
exp_decay() (in module allensdk.internal.model.glif.MLIN), 322
exp_fit_c() (in module allensdk.internal.model.glif.threshold_adaptation), 331
exp_force_c() (in module allensdk.internal.model.glif.threshold_adaptation), 331
experiment_container_id (allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata.BehaviorOphysMetadata attribute), 112
experiment_correlation_search() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 83
experiment_id (allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncOutputs attribute), 357
experiment_injection_coordinate_search() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 84
experiment_source_search() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 84
experiment_spatial_search() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 85
ExperimentGeometry (class in allensdk.brain_observatory.stimulus_info), 224
EXPERIMENTS_KEY (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute), 252
EXPERIMENTS_PRERELEASE_KEY (allensdk.internal.core.mouse_connectivity_cache_prerelease.MouseConnectivityCachePrerelease attribute), 311
ExperimentsTable (class in allensdk.brain_observatory.behavior.behavior_project_cache.tables
108
explode_response_window() (in module allensdk.brain_observatory.behavior.mtrain), 144
export_frame_to_hdf5() (in module allensdk.internal.brain_observatory.ophys_session_decomposition)
304
export_itksnap_labels() (allensdk.core.reference_space.ReferenceSpace method), 261
export_label_description() (allensdk.core.structure_tree.StructureTree method), 269
expsymm_cdf() (in module allensdk.internal.model.glif.MLIN), 322
expsymm_pdf() (in module allensdk.internal.model.glif.MLIN), 322
ExtendedTrialSchema (class in allensdk.brain_observatory.behavior.mtrain), 143
extract() (allensdk.internal.mouse_connectivity.projection_thumbnail.v
method), 349
extract() (in module allensdk.mouse_connectivity.grid.utilities.downsampling_utilities), 378
extract_barcodes() (allensdk.brain_observatory.ecephys.align_timestamps.barcode_syn
method), 167
extract_barcodes_from_states() (in module allensdk.brain_observatory.ecephys.align_timestamps.channel_stat
168
extract_barcodes_from_times() (in module allensdk.brain_observatory.ecephys.align_timestamps.barcode), 165
extract_cell_features() (in module allensdk.ephys.extract_cell_features), 287
extract_const_params_from_stim_repr() (in module allensdk.brain_observatory.ecephys.stimulus_table.stimulus_param
183
extract_data() (allensdk.internal.mouse_connectivity.interval_unionize.interval_un
method), 343
extract_data() (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_u
method), 346
extract_data() (in module allensdk.internal.ephys.core_feature_extract), 302
extract_frame_times() (allensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset.E

method), 172
extract_frame_times_from_photodiode() (allensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset.EcephysSyncDataset method), 172
extract_frame_times_from_vsyncs() (allensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset.EcephysSyncDataset method), 172
extract_injection_from_segmentation() (allensdk.mouse_connectivity.grid.subimage.base_subimage.SegmentationSubImage method), 374
extract_led_times() (allensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset.EcephysSyncDataset method), 172
extract_signal_from_segmentation() (allensdk.mouse_connectivity.grid.subimage.base_subimage.SegmentationSubImage method), 374
extract_splits_from_states() (in module allensdk.brain_observatory.ecephys.align_timestamps.channel_states), 168
extract_stim_class_from_repr() (in module allensdk.brain_observatory.ecephys.stimulus_table.stimulus_parameter_extraction), 183
extract_sweep_features() (in module allensdk.ephys.extract_cell_features), 287
extractor_for_nwb_sweeps() (in module allensdk.ephys.ephys_extractor), 279
extralength (allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise attribute), 208
extralength (allensdk.brain_observatory.natural_scenes.NaturalScenes attribute), 209
extralength (allensdk.brain_observatory.static_gratings.StaticGratings attribute), 221
extrapolate_model_spike_from_endpoints() (in module allensdk.internal.model.glif.glif_optimizer_neuron), 328
extrapolate_model_spike_from_endpoints_single_tau() (in module allensdk.internal.model.glif.glif_optimizer_neuron), 328
extrapolate_spike_time() (in module allensdk.internal.model.glif.glif_optimizer_neuron), 328
extrapolate_spike_voltage() (in module allensdk.internal.model.glif.glif_optimizer_neuron), 328
eye_alignment (allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncOutputs attribute), 357
eye_delta (allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncOutputs attribute), 357
eye_times (allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncOutputs attribute), 357
EYE_TRACKING_KEYS (allensdk.brain_observatory.sync_dataset.Dataset attribute), 228
eye_video_timestamps (allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute), 307
EyeCalibration (class in allensdk.internal.brain_observatory.eye_calibration), 297
EyeTrackingRigGeometry (class in allensdk.brain_observatory.behavior.schemas), 150

F

factory() (allensdk.brain_observatory.ecephys.file_io.ecephys_sync_da class method), 172
factory() (allensdk.brain_observatory.ecephys.file_io.stim_file.CamStin class method), 173
FALSE (allensdk.api.queries.rma_api.RmaApi attribute), 89
FanPlotter (class in allensdk.brain_observatory.circle_plots), 201
FeatureError, 280
fetchall() (allensdk.internal.api.PostgresQueryMixin method), 293
fetchone() (allensdk.internal.api.PostgresQueryMixin method), 293
ff() (in module allensdk.internal.model.GLM), 334
FfmpegInputStream (class in allensdk.internal.brain_observatory.frame_stream), 301
FfmpegOutputStream (class in allensdk.internal.brain_observatory.frame_stream), 301
field_of_view_height (allensdk.brain_observatory.behavior.metadata.behavior_ophys_me attribute), 112
field_of_view_width (allensdk.brain_observatory.behavior.metadata.behavior_ophys_me attribute), 112
figure_in_px() (in module allensdk.brain_observatory.observatory_plots), 210
FILE (allensdk.config.manifest.Manifest attribute), 238
file_hash (allensdk.api.cloud_cache.file_attributes.CacheFileAttributes attribute), 66
file_hash_from_path() (in module allensdk.api.cloud_cache.utils), 67
file_id_column (allensdk.api.cloud_cache.manifest.Manifest attribute), 66
file_id_values (allensdk.api.cloud_cache.manifest.Manifest attribute), 66

FILE_METADATA_MAPPING (allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet attribute), 242
fill_sweep_responses() (allensdk.core.nwb_data_set.NwbDataSet method), 257
filter() (allensdk.api.queries.rma_api.RmaApi method), 91
filter_bad_params() (in module allensdk.internal.brain_observatory.itracker_utils), 302
filter_cell_specimens() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 71
filter_cells() (allensdk.api.queries.cell_types_api.CellTypesApi method), 74
filter_cells_api() (allensdk.api.queries.cell_types_api.CellTypesApi method), 74
filter_digital() (in module allensdk.brain_observatory.behavior.sync.process_sync), 131
filter_events_array() (in module allensdk.brain_observatory.behavior.event_detection), 138
filter_experiment_containers() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 71
filter_experiments() (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method), 253
filter_experiments() (allensdk.internal.core.mouse_connectivity_cache_prerelease.MouseConnectivityCachePrerelease method), 311
filter_experiments_and_containers() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 71
filter_nodes() (allensdk.core.simple_tree.SimpleTree method), 266
filter_ophys_experiments() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 71
filter_putative_spikes() (in module allensdk.ephys.ephys_features), 284
filter_structure_unionizes() (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method), 253
filter_sweeps() (in module allensdk.internal.ephys.core_feature_extract), 312
filtered_sweep_numbers() (in module allensdk.internal.ephys.core_feature_extract), 312
FilterOperators() (allensdk.api.queries.rma_api.RmaApi method), 91
final_processing() (allensdk.brain_observatory.behavior.behavior_project_cache.tables method), 108
finalize() (allensdk.brain_observatory.circle_plots.PolarPlotter method), 202
finalize_no_axes() (in module allensdk.brain_observatory.observatory_plots), 210
finalize_no_labels() (in module allensdk.brain_observatory.observatory_plots), 210
finalize_with_axes() (in module allensdk.brain_observatory.observatory_plots), 210
find() (allensdk.core.swc.Morphology method), 273
find() (allensdk.internal.morphology.morphology.Morphology method), 337
find_bin_center() (in module allensdk.internal.model.glif.MLIN), 322
find_coarse_long_square_amp_delta() (in module allensdk.internal.ephys.core_feature_extract), 312
find_container_tags() (in module allensdk.api.queries.brain_observatory_api), 73
find_downstroke_indexes() (in module allensdk.ephys.ephys_features), 284
find_experiment_acquisition_age() (in module allensdk.api.queries.brain_observatory_api), 74
find_first_model_spike() (in module allensdk.internal.model.glif.glif_optimizer_neuron), 329
find_licks() (in module allensdk.brain_observatory.behavior.trials_processing), 161
find_long_square_sweeps() (in module allensdk.internal.model.glif.find_sweeps), 323
find_matching_index() (in module allensdk.brain_observatory.ecephys.align_timestamps.barcode), 166
find_negative_baselines() (in module allensdk.brain_observatory.demixer), 203
find_negative_baselines() (in module allensdk.internal.brain_observatory.demixer), 296
find_negative_transients_threshold() (in module allensdk.brain_observatory.demixer), 203

find_negative_transients_threshold() (*in module allensdk.internal.brain_observatory.demixer*), 296
find_noise_sweeps() (*in module allensdk.internal.model.glif.find_sweeps*), 323
find_peak_indexes() (*in module allensdk.ephys.ephys_features*), 285
find_ramp_sweeps() (*in module allensdk.internal.model.glif.find_sweeps*), 323
find_ramp_to_rheo_sweeps() (*in module allensdk.internal.model.glif.find_sweeps*), 323
find_ranked_sweep() (*in module allensdk.internal.model.glif.find_sweeps*), 323
find_short_square_sweeps() (*in module allensdk.internal.model.glif.find_sweeps*), 323
find_specimen_cre_line() (*in module allensdk.api.queries.brain_observatory_api*), 74
find_specimen_reporter_line() (*in module allensdk.api.queries.brain_observatory_api*), 74
find_specimen_transgenic_lines() (*in module allensdk.api.queries.brain_observatory_api*), 74
find_spikes_list() (*in module allensdk.internal.model.glif.find_spikes*), 323
find_spikes_list_old() (*in module allensdk.internal.model.glif.find_spikes*), 323
find_spikes_old() (*in module allensdk.internal.model.glif.find_spikes*), 323
find_spikes_ssq_list() (*in module allensdk.internal.model.glif.find_spikes*), 323
find_stim_start() (*in module allensdk.internal.ephys.core_feature_extract*), 312
find_sweep_stim_start() (*in module allensdk.internal.ephys.core_feature_extract*), 313
find_sweeps() (*in module allensdk.internal.model.glif.find_sweeps*), 324
find_time_index() (*in module allensdk.ephys.ephys_features*), 285
find_trough_indexes() (*in module allensdk.ephys.ephys_features*), 285
find_upstroke_indexes() (*in module allensdk.ephys.ephys_features*), 285
find_widths() (*in module allensdk.ephys.ephys_features*), 285
find_zero_baselines() (*in module allensdk.brain_observatory.demixer*), 203
find_zero_baselines() (*in module allensdk.internal.brain_observatory.demixer*), 296
findlevel() (*in module allensdk.brain_observatory.findlevel*), 207
fit() (*allensdk.brain_observatory.r_neuropil.NeuropilSubtract method*), 212
fit_avoltage_bvoltage() (*in module allensdk.internal.model.glif.threshold_adaptation*), 331
fit_avoltage_bvoltage_th() (*in module allensdk.internal.model.glif.threshold_adaptation*), 332
fit_block_coordinate_desc() (*allensdk.brain_observatory.r_neuropil.NeuropilSubtract method*), 212
fit_ellipse() (*allensdk.internal.brain_observatory.fit_ellipse.FitEllipse method*), 300
fit_ellipse() (*in module allensdk.internal.brain_observatory.fit_ellipse*), 300
fit_fi_slope() (*in module allensdk.ephys.ephys_extractor*), 280
fit_membrane_time_constant() (*in module allensdk.ephys.ephys_features*), 286
fit_parameters_file_entries() (*allensdk.internal.api.queries.biophysical_module_reader.Biophysic method*), 289
fit_parameters_path() (*allensdk.internal.api.queries.biophysical_module_reader.Biophysic method*), 289
fit_prespike_time_constant() (*in module allensdk.ephys.ephys_features*), 286
FitEllipse (*class in allensdk.internal.brain_observatory.fit_ellipse*), 300
fitgaussian2D() (*in module allensdk.brain_observatory.receptive_field_analysis.fitgaussian2D*) 191
fix_array_dimensions() (*in module allensdk.core.sitk_utilities*), 268
fix_change_time() (*in module allensdk.brain_observatory.behavior.mtrain*), 145
fix_unary_sections() (*allensdk.config.model.description.Description method*), 236
fix_unexpected_edges() (*in module allensdk.brain_observatory.ecephys.stimulus_sync*), 186
fixed_factory() (*allensdk.internal.mouse_connectivity.projection_thumbnail.volume_ class method*), 349
flag_unexpected_edges() (*in module allensdk.brain_observatory.ecephys.stimulus_sync*),

186
FLOAT (*allensdk.api.queries.connected_services.ConnectedServices attribute*), 76
float_label() (*in module allensdk.brain_observatory.observatory_plots*), 211
for_drifting_gratings() (*allensdk.brain_observatory.circle_plots.FanPlotter static method*), 201
for_static_gratings() (*allensdk.brain_observatory.circle_plots.FanPlotter static method*), 201
FRAME_KEYS (*allensdk.brain_observatory.sync_dataset.Dataset attribute*), 228
FrameInputStream (*class in allensdk.internal.brain_observatory.frame_stream*), 301
FrameOutputStream (*class in allensdk.internal.brain_observatory.frame_stream*), 301
frames_per_second (*allensdk.brain_observatory.ecephys.file_io.stim_file.CamStimOnePickleStimFile attribute*), 173
frequency() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 229
FriendlyDate (*class in allensdk.brain_observatory.behavior.mtrain*), 143
FriendlyDateTime (*class in allensdk.brain_observatory.behavior.mtrain*), 143
from_analysis_file() (*allensdk.brain_observatory.drifting_gratings.DriftingGratings static method*), 206
from_analysis_file() (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise static method*), 208
from_analysis_file() (*allensdk.brain_observatory.natural_movie.NaturalMovie static method*), 209
from_analysis_file() (*allensdk.brain_observatory.natural_scenes.NaturalScenes static method*), 209
from_analysis_file() (*allensdk.brain_observatory.static_gratings.StaticGratings static method*), 221
from_dataframe() (*allensdk.config.manifest_builder.ManifestBuilder method*), 240
from_df() (*allensdk.brain_observatory.stimulus_info.BinaryIntervalSearchTree static method*), 224
from_dict() (*allensdk.internal.model.glif.glif_optimizer_neuron.GlifOptimizerNeuron class method*), 326
from_dict() (*allensdk.internal.morphology.node.Node class method*), 342
from_dict() (*allensdk.model.glif.glif_neuron.GlifNeuron class method*), 365
from_dict_legacy() (*allensdk.internal.model.glif.glif_optimizer_neuron.GlifOptimizerNe class method*), 327
from_file_name() (*allensdk.internal.api.queries.grid_data_api_prerelease.GridDataA class method*), 290
from_foraging_id() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavio class method*), 121
from_json_file() (*allensdk.config.app.application_config.ApplicationConfig method*), 233
from_json_string() (*allensdk.config.app.application_config.ApplicationConfig method*), 233
from_processed() (*allensdk.brain_observatory.behavior.stimulus_processing.stimulus_ static method*), 127
from_processed() (*allensdk.brain_observatory.behavior.stimulus_processing.stimulus_ static method*), 127
from_sweeps() (*allensdk.ephys.ephys_extractor.EphysSweepSetFeatureExtractor class method*), 279
from_unprocessed() (*allensdk.brain_observatory.behavior.stimulus_processing.stimulus_ method*), 127
from_unprocessed() (*allensdk.brain_observatory.behavior.stimulus_processing.stimulus_ static method*), 128
full_genotype (*allensdk.brain_observatory.behavior.metadata.behavior_metadata. attribute*), 110

G

gather_from_seeds() (*allensdk.internal.model.biophysical.make_deap_fit_json.Report method*), 321
gaussian2D() (*in module allensdk.brain_observatory.receptive_field_analysis.fitgaussian2D*), 191
GaussianFitError, 191
generate_fit_file() (*allensdk.internal.model.biophysical.make_deap_fit_json.Report method*), 321
generate_manifest_lims() (*allensdk.internal.model.biophysical.run_optimize.RunOptimize method*), 321
generate_manifest_lims() (*allensdk.internal.model.biophysical.run_simulate_lims.RunSimulat method*), 322

generate_manifest_rma() (*allensdk.internal.model.biophysical.run_optimize.RunOptimizer method*), 321
generate_manifest_rma() (*allensdk.internal.model.biophysical.run_simulate_lims.RunSimulateLims method*), 322
generate_morphology() (*allensdk.internal.model.biophysical.deap_utils.Utils method*), 320
generate_morphology() (*allensdk.model.biophysical.utils.AllActiveUtils method*), 362
generate_morphology() (*allensdk.model.biophysical.utils.Utils method*), 362
generate_output_cell_features() (*in module allensdk.internal.ephys.core_feature_extract*), 313
generate_rays() (*in module allensdk.internal.brain_observatory.itracker_utils*), 302
generate_warp_coordinates() (*allensdk.brain_observatory.stimulus_info.ExperimentGeometry method*), 224
get_A() (*in module allensdk.brain_observatory.receptive_field_analysis.utilities*), 193
get_A_blur() (*in module allensdk.brain_observatory.receptive_field_analysis.utilities*), 193
get_affine_parameters() (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method*), 253
get_age() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDa method*), 113
get_age() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api.BehaviorJsonExtractor method*), 119
get_age() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method*), 121
get_alignment_array() (*in module allensdk.internal.brain_observatory.time_sync*), 308
get_all_bits() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 229
get_all_events() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 229
get_all_features() (*allensdk.core.cell_types_cache.CellTypesCache method*), 247
get_all_times() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 229
get_analog_channel() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 229
get_analog_meta() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 229
get_ancestor_id_map() (*allensdk.core.structure_tree.StructureTree method*), 269
get_ancestor_id_map() (*in module allensdk.internal.mouse_connectivity.interval_unionize.run_tissuec* 345
get_annotated_section_data_sets() (*allensdk.api.queries.annotated_section_data_sets_api.AnnotatedSe method*), 68
get_annotated_section_data_sets_via_rma() (*allensdk.api.queries.annotated_section_data_sets_api.Annotated method*), 68
get_annotation_volume() (*allensdk.core.reference_space_cache.ReferenceSpaceCache method*), 263
get_atlases() (*allensdk.api.queries.ontologies_api.OntologiesApi method*), 86
get_atlases_table() (*allensdk.api.queries.ontologies_api.OntologiesApi method*), 86
get_attribute_dict() (*in module allensdk.brain_observatory.receptive_field_analysis.receptive_field* 193
get_attribute_dict() (*in module allensdk.brain_observatory.receptive_field_analysis.utilities*), 193
get_average_projection() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extr method*), 114
get_average_projection() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method*), 117
get_average_projection() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 124
get_barcode_table() (*allensdk.brain_observatory.ecephys.align_timestamps.barcode_syn method*), 168
get_behavior_monitoring() (*in module allensdk.brain_observatory.behavior.sync*), 132
get_behavior_session_id() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 113
get_behavior_session_id() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas*

method), 115
get_behavior_session_id() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api.BehaviorJsonExtractor method), 119
get_behavior_session_id() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method), 121
get_behavior_session_id() (allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms.BehaviorDataTransforms method), 122
get_behavior_stimulus_file() (allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDataExtractorBase method), 113
get_behavior_stimulus_file() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api.BehaviorJsonExtractor method), 119
get_behavior_stimulus_file() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method), 121
get_behavior_tracking_video_filepath_df() (allensdk.internal.api.lims_api.LimsApi method), 293
get_birth_date() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method), 121
get_bit() (allensdk.brain_observatory.sync_dataset.Dataset method), 229
get_bit() (in module allensdk.brain_observatory.sync_dataset), 231
get_bit_changes() (allensdk.brain_observatory.sync_dataset.Dataset method), 229
get_blend() (in module allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher), 349
get_blend_component() (in module allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher), 350
get_blocks() (in module allensdk.brain_observatory.ecephys.stimulus_table.visualization_blocks), 176
get_cache_path() (allensdk.api.warehouse_cache.cache.Cache method), 98
get_cap_check_indices() (in module allensdk.internal.model.biophysical.passive_fitting.preprocess), 319
get_catch_responses() (in module allensdk.brain_observatory.behavior.dprime), 137
get_cav_density() (in module allensdk.internal.mouse_connectivity.interval_unionize.data_utilities), 343
get_cell() (allensdk.api.queries.cell_types_api.CellTypesApi method), 74
get_cell_metrics() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 72
get_cell_roi_ids() (allensdk.brain_observatory.behavior.session_apis.data_transforms. method), 124
get_cell_specimen_id_mapping() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 72
get_cell_specimen_ids() (allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method), 242
get_cell_specimen_indices() (allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method), 243
get_cell_specimen_table() (allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method), 117
get_cell_specimen_table() (allensdk.brain_observatory.behavior.session_apis.data_transforms. method), 124
get_cells() (allensdk.core.cell_types_cache.CellTypesCache method), 248
get_cells() (allensdk.internal.model.biophysical.biophysical_archiver. method), 320
get_change_time_frame_response_latency() (in module allensdk.brain_observatory.behavior.trials_processing), 161
get_channels() (allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_ method), 170
get_channels() (allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_ method), 171
get_child_ids() (allensdk.core.ontology.Ontology method), 259
get_children() (allensdk.core.ontology.Ontology method), 259
get_colormap() (allensdk.core.structure_tree.StructureTree method), 269
get_column_definitions() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 72
get_components() (in module allensdk.brain_observatory.receptive_field_analysis.utilities), 193
get_compound_annotated_section_data_sets() (allensdk.api.queries.annotated_section_data_sets_api.Annotated method), 69
get_config() (allensdk.config.manifest_builder.ManifestBuilder

method), 240
get_connection() (*allensdk.internal.api.PostgresQueryMixin method*), 293
get_corrected_fluorescence_traces() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base.BehaviorOphysBase method*), 117
get_corrected_fluorescence_traces() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphysDataTrans method*), 124
get_corrected_fluorescence_traces() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method*), 243
get_credential_provider() (*in module allensdk.core.authentication*), 242
get_cursor() (*allensdk.internal.api.PostgresQueryMixin method*), 293
get_data_mask() (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method*), 254
get_date_of_acquisition() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDataExtra method*), 113
get_date_of_acquisition() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api.BehaviorJsonExtractor method*), 119
get_date_of_acquisition() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method*), 121
get_default_manifest_file() (*in module allensdk.api.warehouse_cache.cache*), 100
get_deformation_field() (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method*), 254
get_demix_file() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_ophys_data_extractor_base.BehaviorOp method*), 114
get_demixed_traces() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method*), 243
get_descendant_ids() (*allensdk.core.ontology.Ontology method*), 260
get_descendants() (*allensdk.core.ontology.Ontology method*), 260
get_dff_file() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_ophys_data_extractor_base.BehaviorOp method*), 114
get_dff_traces() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base.BehaviorOphysBase method*), 117
get_dff_traces() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphysDataTrans method*), 124
get_dff_traces() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method*), 243
get_diagonals_from_sparse() (*in module allensdk.brain_observatory.r_neuropil*), 213
get_dimensions() (*allensdk.internal.morphology.morphology.Morphology method*), 337
get_disc_masks() (*in module allensdk.brain_observatory.receptive_field_analysis.chisquarerf*), 189
get_dprime() (*in module allensdk.brain_observatory.behavior.dprime*), 137
get_driver_line() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 113
get_driver_line() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior method*), 119
get_driver_line() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior method*), 121
get_ecephys_session_id() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_ method*), 171
get_edges() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 229
get_ephys_data() (*allensdk.core.cell_types_cache.CellTypesCache method*), 248
get_ephys_features() (*allensdk.api.queries.cell_types_api.CellTypesApi method*), 74
get_ephys_features() (*allensdk.core.cell_types_cache.CellTypesCache method*), 248
get_ephys_sweeps() (*allensdk.api.queries.cell_types_api.CellTypesApi method*), 74
get_ephys_sweeps() (*allensdk.api.queries.glif_api.GlifApi method*), 76
get_ephys_sweeps() (*allensdk.core.cell_types_cache.CellTypesCache method*), 248
get_epoch_list() (*in module allensdk.core.brain_observatory_nwb_data_set*), 246
get_external_specimen_name() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 113
get_external_specimen_name() (*al-

lensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api.BehaviorJsonExtractor method), 119

get_equipment_name() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method*), 121

get_even_sampling() (*in module allensdk.brain_observatory.behavior.trials_processing*), 161

get_event_detection_filepath() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_ophys_data_extractor_base.BehaviorOp method*), 114

get_events() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base.BehaviorOphysBase method*), 117

get_events() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphy method*), 124

get_events_by_bit() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 230

get_events_by_line() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 230

get_events_per_pixel() (*in module allensdk.brain_observatory.receptive_field_analysis.chisquare*), 189

get_eXcluded() (*allensdk.internal.brain_observatory.roi_filter_utils.TrainingMultiLabelClassifier method*), 305

get_expected_events_by_pixel() (*in module allensdk.brain_observatory.receptive_field_analysis.chisquare*), 189

get_experiment_analysis_file() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 355

get_experiment_container_metrics() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method*), 72

get_experiment_containers() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method*), 72

get_experiment_detail() (*allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method*), 85

get_experiment_files() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 355

get_experiment_id() (*allensdk.internal.api.lims_api.LimsApi method*), 293

get_experiment_nwb_file() (*in module allensdk.internal.pipeline_modules.run_observatory_analysis*), 353

get_experiment_nwb_file() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 355

get_experiment_session() (*in module allensdk.internal.pipeline_modules.run_observatory_analysis*), 353

get_experiment_structure_unionizes() (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCac method*), 254

get_experiment_sweep_numbers() (*allensdk.core.nwb_data_set.NwbDataSet method*), 257

get_experiments() (*allensdk.api.queries.mouse_connectivity_api.MouseConnectivityAp method*), 85

get_experiments() (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method*), 255

get_experiments() (*allensdk.internal.api.queries.mouse_connectivity_api_prerelease.M method*), 291

get_experiments() (*allensdk.internal.core.mouse_connectivity_cache_prerelease.Mouse method*), 311

get_experiments_api() (*allensdk.api.queries.mouse_connectivity_api.MouseConnectivityAp method*), 85

get_expt_description() (*in module allensdk.brain_observatory.behavior.metadata.behavior_metadata*), 111

get_extended_trials() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 122

get_extended_trials() (*in module allensdk.brain_observatory.behavior.trials_processing*), 161

get_extractor() (*allensdk.brain_observatory.behavior.metadata.behavior_metadata. method*), 110

get_eye_tracking() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method*), 117

get_eye_tracking() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 125

get_eye_tracking() (*in module allensdk.brain_observatory.behavior.sync*), 132

get_eye_tracking_filepath() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 114

get_eye_tracking_rig_geometry() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 114

get_eye_tracking_rig_geometry() (*al-

allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base.BehaviorOphysBase method), 118
get_eye_tracking_rig_geometry() (allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphysDataTran method), 125
get_eye_tracking_video_filepath_df() (allensdk.internal.api.lims_api.LimsApi method), 293
get_falling_edges() (allensdk.brain_observatory.sync_dataset.Dataset method), 230
get_false_alarm_rate() (in module allensdk.brain_observatory.behavior.dprime), 138
get_features() (in module allensdk.internal.ephys.plot_qc_figures), 313
get_features() (in module allensdk.internal.ephys.plot_qc_figures3), 315
get_field_of_view_shape() (allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_ophys_data_extractor_base.BehaviorOp method), 114
get_fluorescence() (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis method), 222
get_fluorescence_timestamps() (allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method), 243
get_fluorescence_traces() (allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method), 243
get_foraging_id() (allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDataExtra method), 113
get_foraging_id() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api.BehaviorJsonExtractor method), 119
get_foraging_id() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method), 121
get_format() (allensdk.config.manifest.Manifest method), 239
get_full_genotype() (allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDataExtra method), 113
get_full_genotype() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api.BehaviorJsonExtractor method), 119
get_full_genotype() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method), 121
get_gaussian_fit() (in module allensdk.brain_observatory.receptive_field_analysis.postprocessing), 192
get_gaussian_fit() (in module allensdk.brain_observatory.receptive_field_analysis.fit_parameters), 191
get_genes() (allensdk.api.queries.mouse_atlas_api.MouseAtlasApi method), 82
get_go_responses() (in module allensdk.brain_observatory.behavior.dprime), 138
get_gratings_metadata() (in module allensdk.brain_observatory.behavior.stimulus_processing), 129
get_h() (in module allensdk.internal.model.biophysical.passive_fitting.neuron_utils), 319
get_hit_rate() (in module allensdk.brain_observatory.behavior.dprime), 138
get_id_acronym_map() (allensdk.core.structure_tree.StructureTree method), 170
get_image_info_from_trial() (in module allensdk.brain_observatory.behavior.trials_processing), 161
get_image_region() (allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile method), 350
get_image_set_name() (in module allensdk.brain_observatory.behavior.stimulus_processing.util), 129
get_image_to_atlas() (allensdk.api.queries.synchronization_api.SynchronizationApi method), 95
get_image_to_image() (allensdk.api.queries.synchronization_api.SynchronizationApi method), 95
get_image_to_image_2d() (allensdk.api.queries.synchronization_api.SynchronizationApi method), 96
get_image_to_reference() (allensdk.api.queries.synchronization_api.SynchronizationApi method), 96
get_images_dict() (in module allensdk.brain_observatory.behavior.stimulus_processing), 129
get_imaging_depth() (allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method), 114
get_imaging_plane_group() (allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method), 114
get_indicator_bound_point() (in module allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher), 350

get_indices_by_distance() (*in module allensdk.internal.brain_observatory.roi_filter_utils*), 306
get_injection_data() (*in module allensdk.internal.mouse_connectivity.interval_unionize.data_utilities*), 343
get_injection_density() (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method*), 255
get_injection_fraction() (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method*), 255
get_input_data() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 356
get_input_json() (*in module allensdk.internal.core.lims_utilities*), 309
get_intensity() (*allensdk.mouse_connectivity.grid.subimage.base_subimage.IntensitySubImage method*), 374
get_invalid_times() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi method*), 171
get_isi_experiments() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method*), 73
get_isi_experiments() (*allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi method*), 170
get_isis() (*in module allensdk.ephys.ephys_features*), 286
get_keys() (*in module allensdk.internal.brain_observatory.time_sync*), 308
get_lfp() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi method*), 171
get_lfp_channel_order() (*allensdk.brain_observatory.ecephys.file_io.continuous_file.ContinuousFile method*), 172
get_lick_times() (*in module allensdk.brain_observatory.behavior.sync*), 132
get_licks() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_base.BehaviorBase method*), 115
get_licks() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms.BehaviorDataTransfo method*), 122
get_line() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 230
get_line_changes() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 230
get_list_of_path_dict() (*in module allensdk.test_utilities.regression_fixture*), 381
get_locally_sparse_noise_stimulus_template() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservator method*), 244
get_manifest() (*allensdk.config.manifest_builder.ManifestBuilder method*), 240
get_manual_injection_summary() (*allensdk.api.queries.mouse_connectivity_api.MouseConnectivityAp method*), 85
get_mask() (*allensdk.brain_observatory.stimulus_info.Monitor method*), 224
get_mask_plane() (*allensdk.brain_observatory.roi_masks.Mask method*), 214
get_max_projection() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method*), 118
get_max_projection() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 125
get_max_projection() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method*), 244
get_max_projection_file() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 114
get_mean_response() (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoi method*), 208
get_mean_waveforms() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_ method*), 171
get_metadata() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method*), 115
get_metadata() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method*), 118
get_metadata() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 122
get_metadata() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 125
get_metadata() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_ method*), 171
get_metadata() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method*), 244
get_metrics() (*in module allensdk.internal.brain_observatory.annotated_region_metrics*), 295
get_missing_path() (*allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile*

method), 351
get_monitor_delay() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms.BehaviorDataTransforms method*), 122
get_monitor_delay() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphysDataTrans method*), 125
get_morphology_features() (*allensdk.api.queries.cell_types_api.CellTypesApi method*), 75
get_morphology_features() (*allensdk.core.cell_types_cache.CellTypesCache method*), 249
get_motion_correction() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base.BehaviorOphysBase method*), 118
get_motion_correction() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphysDataTrans method*), 125
get_motion_correction() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method*), 244
get_mouse_id() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDataExt method*), 113
get_mouse_id() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior_json_api.BehaviorJsonExtractor method*), 119
get_mouse_id() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method*), 121
get_mouse_id() (*in module allensdk.brain_observatory.behavior.trials_processing*), 161
get_name_map() (*allensdk.core.structure_tree.StructureTree method*), 270
get_natural_movie_template() (*allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi method*), 170
get_natural_scene_template() (*allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi method*), 170
get_nearest() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 230
get_neuron_config() (*allensdk.api.queries.glif_api.GlifApi method*), 76
get_neuron_configs() (*allensdk.api.queries.glif_api.GlifApi method*), 77
get_neuronal_model() (*allensdk.api.queries.glif_api.GlifApi method*), 77
get_neuronal_model_runs() (*allensdk.internal.api.queries.biophysical_module_api.BiophysicalM method*), 288
get_neuronal_model_templates() (*allensdk.api.queries.glif_api.GlifApi method*), 77
get_neuronal_models() (*allensdk.api.queries.biophysical_api.BiophysicalApi method*), 70
get_neuronal_models() (*allensdk.api.queries.glif_api.GlifApi method*), 77
get_neuronal_models() (*allensdk.internal.api.queries.biophysical_module_api.BiophysicalM method*), 288
get_neuronal_models() (*allensdk.internal.model.biophysical.biophysical_archiver.Biophysical method*), 320
get_neuronal_models_by_id() (*allensdk.api.queries.glif_api.GlifApi method*), 77
get_neuropil_r() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method*), 244
get_neuropil_traces() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method*), 244
get_noise_correlation() (*allensdk.brain_observatory.drifting_gratings.DriftingGratings method*), 206
get_noise_correlation() (*allensdk.brain_observatory.natural_scenes.NaturalScenes method*), 209
get_noise_correlation() (*allensdk.brain_observatory.static_gratings.StaticGratings method*), 221
get_ophys_cell_segmentation_run_id() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extr method*), 114
get_ophys_container_id() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 114
get_ophys_container_id() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior method*), 121
get_ophys_data_length() (*in module allensdk.internal.brain_observatory.time_sync*), 308
get_ophys_experiment_id() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 114
get_ophys_experiment_id() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas*

method), 118

get_ophys_experiment_id() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphysDataTrans method*), 125

get_ophys_experiment_ids() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method*), 121

get_ophys_experiments() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method*), 73

get_ophys_frames() (*in module allensdk.brain_observatory.behavior.sync*), 133

get_ophys_session_id() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_ophys_data_extractor_base.BehaviorOp method*), 114

get_ophys_session_id() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base.BehaviorOphysBase method*), 118

get_ophys_session_id() (*allensdk.brain_observatory.behavior.session_apis.data_io.behavior_lims_api.BehaviorLimsExtractor method*), 121

get_ophys_session_id() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphysDataTrans method*), 126

get_ophys_timestamps() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base.BehaviorOphysBase method*), 118

get_ophys_timestamps() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphysDataTrans method*), 126

get_optogenetic_stimulation() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi method*), 171

get_ori_info_from_trial() (*in module allensdk.brain_observatory.behavior.trials_processing*), 161

get_output() (*allensdk.internal.mouse_connectivity.projection_thumbnail.image_sheet.ImageSheet method*), 348

get_overall_blend() (*in module allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher*), 350

get_params() (*allensdk.brain_observatory.session_api_utils.ParamsMixin method*), 220

get_params() (*in module allensdk.brain_observatory.behavior.trials_processing*), 161

get_passive_fit_data() (*in module allensdk.internal.model.biophysical.passive_fitting.preprocess*), 319

get_path() (*allensdk.config.manifest.Manifest method*), 239

get_path() (*in module allensdk.internal.pipeline_modules.run_demixing*), 352

get_peak() (*allensdk.brain_observatory.drifting_gratings.DriftingGrati method*), 206

get_peak() (*allensdk.brain_observatory.locally_sparse_noise.LocallySp method*), 208

get_peak() (*allensdk.brain_observatory.natural_movie.NaturalMovie method*), 209

get_peak() (*allensdk.brain_observatory.natural_scenes.NaturalScenes method*), 209

get_peak() (*allensdk.brain_observatory.static_gratings.StaticGratings method*), 221

get_peak() (*allensdk.brain_observatory.stimulus_analysis.StimulusAna method*), 223

get_peak_significance() (*in module allensdk.brain_observatory.receptive_field_analysis.chisquarerf*), 190

get_peaks() (*in module allensdk.internal.model.glif.threshold_adaptation*), 333

get_photodiode_events() (*in module allensdk.internal.brain_observatory.time_sync*), 308

get_pipeline_version() (*allensdk.core.nwb_data_set.NwbDataSet method*), 257

get_plane_group_count() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 114

get_polygons() (*allensdk.mouse_connectivity.grid.image_series.PolygonImageSeries method*), 374

get_probe_lfp_data() (*allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_p method*), 170

get_probe_time_offset() (*in module allensdk.brain_observatory.ecephys.align_timestamps.barcode*), 166

get_probes() (*allensdk.brain_observatory.ecephys.ecephys_project_ap method*), 170

get_probes() (*allensdk.brain_observatory.ecephys.ecephys_session_ap method*), 171

get_project_code() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 115

get_projection_data() (*in module allensdk.internal.mouse_connectivity.interval_unionize.data_utilitie*), 343

get_projection_density() (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method*), 256

get_projection_image_info() (*allensdk.api.queries.mouse_connectivity_api.MouseConnectivityAp method*), 85

get_projection_matrix() (*al-

lensdk.core.mouse_connectivity_cache.MouseConnectivityCache method), 256
get_pupil_data() (allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi method), 171
get_pupil_location() (allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method), 245
get_pupil_size() (allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method), 245
get_ramp_stim_characteristics() (in module allensdk.ephys.extract_cell_features), 287
get_raw_cell_specimen_table_dict() (allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_ophys_data_extractor_base.BehaviorOp method), 115
get_raw_dff_data() (allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphysDataTran method), 126
get_raw_stimulus_frames() (in module allensdk.brain_observatory.behavior.sync), 133
get_real_photodiode_events() (in module allensdk.internal.brain_observatory.time_sync), 308
get_receptive_field() (allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise method), 208
get_receptive_field_analysis_data() (allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise method), 208
get_receptive_field_attribute_df() (allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise method), 208
get_reconstruction() (allensdk.core.cell_types_cache.CellTypesCache method), 249
get_reconstruction_markers() (allensdk.core.cell_types_cache.CellTypesCache method), 249
get_recorded_data() (allensdk.model.biophysical.utils.Utils method), 362
get_reference_aligned_image_channel_volumes_url() (allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method), 86
get_reference_space() (allensdk.core.reference_space_cache.ReferenceSpaceCache method), 263
get_reference_to_image() (allensdk.api.queries.synchronization_api.SynchronizationApi method), 96
get_reporter_line() (allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDataExtra method), 113
get_reporter_line() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior method), 120
get_reporter_line() (allensdk.brain_observatory.behavior.session_apis.data_io.behavior method), 121
get_representational_similarity() (allensdk.brain_observatory.drifting_gratings.DriftingGratings method), 206
get_representational_similarity() (allensdk.brain_observatory.natural_scenes.NaturalScenes method), 210
get_representational_similarity() (allensdk.brain_observatory.static_gratings.StaticGratings method), 221
get_response() (allensdk.brain_observatory.drifting_gratings.DriftingGratings method), 206
get_response() (allensdk.brain_observatory.natural_scenes.NaturalScenes method), 210
get_response() (allensdk.brain_observatory.static_gratings.StaticGratings method), 221
get_response() (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis method), 223
get_response_latency() (in module allensdk.brain_observatory.behavior.trials_processing), 161
get_response_type() (in module allensdk.brain_observatory.behavior.trials_processing), 161
get_rewards() (allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method), 115
get_rewards() (allensdk.brain_observatory.behavior.session_apis.data_transforms. method), 122
get_rewards() (allensdk.brain_observatory.behavior.session_apis.data_transforms. method), 126
get_rewards() (in module allensdk.brain_observatory.behavior.rewards_processing), 145
get_rig_metadata() (allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_s method), 171
get_rigid_motion_transform_file() (allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method), 115
get_rising_edges() (allensdk.brain_observatory.sync_dataset.Dataset

method), 231
get_roi_ids() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method*), 245
get_roi_mask() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method*), 245
get_roi_mask_array() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method*), 245
get_roi_masks_by_cell_roi_id() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_ophys_data_transforms.BehaviorOphysDataTran*... *method*), 126
get_rois() (*in module allensdk.internal.brain_observatory.roi_filter_utils*), 306
get_rolling_dprime() (*in module allensdk.brain_observatory.behavior.dprime*), 138
get_running_acquisition_df() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_base.BehaviorBase method*), 116
get_running_acquisition_df() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms.BehaviorDataTransforms method*), 122
get_running_df() (*in module allensdk.brain_observatory.behavior.running_processing*), 145
get_running_speed() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_base.BehaviorBase method*), 116
get_running_speed() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms.BehaviorDataTransforms method*), 123
get_running_speed() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi method*), 171
get_running_speed() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method*), 245
get_schema() (*allensdk.api.queries.rma_api.RmaApi method*), 91
get_screen_gaze_data() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi method*), 171
get_section_data_sets() (*allensdk.api.queries.mouse_atlas_api.MouseAtlasApi method*), 82
get_section_data_sets_by_product() (*allensdk.api.queries.image_download_api.ImageDownloadApi method*), 80
get_section_image_ranges() (*allensdk.api.queries.image_download_api.ImageDownloadApi method*), 81
get_segmentation() (*allensdk.mouse_connectivity.grid.subimage.base_subimage.Segmen*... *method*), 375
get_session_data() (*allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_p*... *method*), 170
get_session_start_time() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_s*... *method*), 171
get_session_type() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN*... *method*), 245
get_sessions() (*allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_p*... *method*), 170
get_sex() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_*... *method*), 113
get_sex() (*allensdk.brain_observatory.behavior.session_apis.data_io.be*... *method*), 120
get_sex() (*allensdk.brain_observatory.behavior.session_apis.data_io.be*... *method*), 121
get_shuffle_matrix() (*in module allensdk.brain_observatory.receptive_field_analysis.utilities*), 193
get_signal_correlation() (*allensdk.brain_observatory.drifting_gratings.DriftingGratings method*), 207
get_signal_correlation() (*allensdk.brain_observatory.natural_scenes.NaturalScenes method*), 210
get_signal_correlation() (*allensdk.brain_observatory.static_gratings.StaticGratings method*), 221
get_sitk_image_information() (*in module allensdk.core.sitk_utilities*), 268
get_slice_image() (*allensdk.core.reference_space.ReferenceSpace method*), 261
get_sparse_noise_epoch_mask_list() (*in module allensdk.brain_observatory.receptive_field_analysis.utilities*), 193
get_spatial_grating() (*in module allensdk.brain_observatory.stimulus_info*), 225
get_spatio_temporal_grating() (*in module allensdk.brain_observatory.stimulus_info*), 225
get_speed_tuning() (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis method*), 223
get_spike_amplitudes() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_s*... *method*), 171
get_spike_times() (*al-*

lensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi method), 171
get_spike_times() (*allensdk.core.nwb_data_set.NwbDataSet method*), 257
get_spikes() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
get_spikes() (*in module allensdk.internal.ephys.plot_qc_figures3*), 315
get_square_stim_characteristics() (*in module allensdk.ephys.extract_cell_features*), 287
get_step_stim_characteristics() (*in module allensdk.internal.model.biophysical.ephys_utils*), 320
get_stim_characteristics() (*in module allensdk.ephys.extract_cell_features*), 287
get_stim_data_length() (*in module allensdk.internal.brain_observatory.time_sync*), 308
get_stim_photodiode() (*in module allensdk.brain_observatory.behavior.sync*), 133
get_stimulus() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method*), 245
get_stimulus_attr_changes() (*in module allensdk.brain_observatory.behavior.trials_processing*), 161
get_stimulus_epoch_table() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet method*), 245
get_stimulus_file() (*allensdk.internal.model.biophysical.biophysical_archiver.BiophysicalArchiver method*), 320
get_stimulus_frame_rate() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms.BehaviorDataTransforms method*), 123
get_stimulus_mappings() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method*), 73
get_stimulus_metadata() (*in module allensdk.brain_observatory.behavior.stimulus_processing*), 129
get_stimulus_name() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_data_extractor_base.BehaviorDataExtractorBase method*), 114
get_stimulus_presentations() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_base.BehaviorBase method*), 116
get_stimulus_presentations() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_ophys_base.BehaviorOphysBase method*), 118
get_stimulus_presentations() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 123
get_stimulus_presentations() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_ method*), 171
get_stimulus_presentations() (*in module allensdk.brain_observatory.behavior.stimulus_processing*), 130
get_stimulus_table() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method*), 246
get_stimulus_template() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method*), 246
get_stimulus_templates() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method*), 116
get_stimulus_templates() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 123
get_stimulus_templates() (*in module allensdk.brain_observatory.behavior.stimulus_processing*), 130
get_stimulus_timestamps() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method*), 116
get_stimulus_timestamps() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method*), 118
get_stimulus_timestamps() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 123
get_stimulus_timestamps() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 126
get_structure_graphs() (*allensdk.api.queries.ontologies_api.OntologiesApi method*), 86
get_structure_mask() (*allensdk.core.reference_space_cache.ReferenceSpaceCache method*), 264
get_structure_mesh() (*allensdk.core.reference_space_cache.ReferenceSpaceCache method*), 264
get_structure_sets() (*allensdk.api.queries.ontologies_api.OntologiesApi method*), 86
get_structure_sets() (*allensdk.core.structure_tree.StructureTree method*), 270
get_structure_to_image() (*allensdk.api.queries.synchronization_api.SynchronizationApi*

method), 96
get_structure_tree() (*allensdk.core.reference_space_cache.ReferenceSpaceCache method*), 264
get_structure_unionizes() (*allensdk.api.queries.mouse_connectivity_api.MouseConnectivityApi method*), 86
get_structure_unionizes() (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method*), 256
get_structure_unionizes() (*allensdk.internal.api.queries.mouse_connectivity_api_prerelease.MouseConnectivityApiPrerelease method*), 291
get_structures() (*allensdk.api.queries.ontologies_api.OntologiesApi method*), 86
get_structures_by_acronym() (*allensdk.core.structure_tree.StructureTree method*), 270
get_structures_by_id() (*allensdk.core.structure_tree.StructureTree method*), 270
get_structures_by_name() (*allensdk.core.structure_tree.StructureTree method*), 270
get_structures_by_set_id() (*allensdk.core.structure_tree.StructureTree method*), 270
get_structures_with_sets() (*allensdk.api.queries.ontologies_api.OntologiesApi method*), 87
get_sum_pixel_intensities() (*in module allensdk.internal.mouse_connectivity.interval_unionize.data_utilities*), 343
get_sum_pixels() (*in module allensdk.internal.mouse_connectivity.interval_unionize.data_utilities*), 343
get_surface_2p_pixel_size_um() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extractor_base.behavior_ophys_data_extractor_base.BehaviorOp method*), 115
get_svg() (*allensdk.api.queries.svg_api.SvgApi method*), 95
get_sweep() (*allensdk.core.nwb_data_set.NwbDataSet method*), 257
get_sweep_metadata() (*allensdk.core.nwb_data_set.NwbDataSet method*), 258
get_sweep_numbers() (*allensdk.core.nwb_data_set.NwbDataSet method*), 258
get_sweep_numbers() (*in module allensdk.internal.model.glif.find_sweeps*), 324
get_sweep_response() (*allensdk.brain_observatory.natural_movie.NaturalMovie method*), 209
get_sweep_response() (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis method*), 223
get_sweep_v_i_t_from_set() (*in module allensdk.internal.model.biophysical.ephys_utils*), 320
get_sweeps_by_name() (*in module allensdk.internal.model.glif.find_sweeps*), 324
get_sweeps_of_type() (*in module allensdk.internal.model.biophysical.ephys_utils*), 320
get_sync_data() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 126
get_sync_data() (*in module allensdk.brain_observatory.behavior.sync*), 134
get_sync_file() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 115
get_sync_licks() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 126
get_synchronized_frame_times() (*in module allensdk.brain_observatory.sync_utilities*), 196
get_targeted_structure() (*allensdk.brain_observatory.behavior.session_apis.abcs.data_extrac method*), 115
get_task_parameters() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_bas method*), 116
get_task_parameters() (*allensdk.brain_observatory.behavior.session_apis.data_transforms. method*), 123
get_task_parameters() (*in module allensdk.brain_observatory.behavior.metadata.behavior_metadata*), 111
get_template_parameters() (*allensdk.internal.model.biophysical.biophysical_archiver.Biophysic method*), 320
get_template_volume() (*allensdk.core.reference_space_cache.ReferenceSpaceCache method*), 265
get_time() (*in module allensdk.brain_observatory.behavior.trials_processing*), 161
get_time_string() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
get_time_string() (*in module allensdk.internal.ephys.plot_qc_figures3*), 315
get_tree() (*allensdk.api.queries.tree_search_api.TreeSearchApi method*), 97

get_trial_bounds() (*in module allensdk.brain_observatory.behavior.trials_processing*), 161

get_trial_count_corrected_false_alarm_rate() (*in module allensdk.brain_observatory.behavior.dprime*), 138

get_trial_count_corrected_hit_rate() (*in module allensdk.brain_observatory.behavior.dprime*), 138

get_trial_image_names() (*in module allensdk.brain_observatory.behavior.trials_processing*), 162

get_trial_reward_time() (*in module allensdk.brain_observatory.behavior.trials_processing*), 162

get_trial_timing() (*in module allensdk.brain_observatory.behavior.trials_processing*), 162

get_trials() (*allensdk.brain_observatory.behavior.session_apis.abcs.session_base.behavior_base.BehaviorBase method*), 116

get_trials() (*allensdk.brain_observatory.behavior.session_apis.data_transforms.behavior_data_transforms.BehaviorDataTransforms method*), 124

get_trials_from_data_transform() (*in module allensdk.brain_observatory.behavior.trials_processing*), 163

get_trials_v0() (*in module allensdk.brain_observatory.behavior.trials_processing*), 164

get_trigger() (*in module allensdk.brain_observatory.behavior.sync*), 134

get_unit_analysis_metrics() (*allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi method*), 170

get_unit_filter_value() (*in module allensdk.brain_observatory.ecephys*), 186

get_units() (*allensdk.brain_observatory.ecephys.ecephys_project_api.ecephys_project_api.EcephysProjectApi method*), 170

get_units() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi method*), 171

get_video_length() (*in module allensdk.internal.brain_observatory.time_sync*), 309

get_visual_stimuli_df() (*in module allensdk.brain_observatory.behavior.stimulus_processing*), 131

get_volume_scale() (*in module allensdk.internal.mouse_connectivity.interval_unionize.run_tissuecyte_unionize_classic*), 345

get_well_known_file_by_name() (*in module allensdk.internal.core.lims_utilities*), 309

get_well_known_file_by_type() (*in module allensdk.internal.core.lims_utilities*), 309

get_well_known_file_ids() (*allensdk.api.queries.biophysical_api.BiophysicalApi method*), 70

get_well_known_files_by_name() (*in module allensdk.internal.core.lims_utilities*), 310

get_well_known_files_by_type() (*in module allensdk.internal.core.lims_utilities*), 310

get_wkf() (*in module allensdk.internal.pipeline_modules.run_ophys_eye_calibration*), 356

GLIF_TYPES (*allensdk.api.queries.glif_api.GlifApi attribute*), 76

GlifApi (*class in allensdk.api.queries.glif_api*), 76

GlifBadInitializationException, 326

GlifBadResetException, 363

GlifExperiment (*class in allensdk.internal.model.glif.glif_experiment*), 324

GlifNeuron (*class in allensdk.model.glif.glif_neuron*), 363

GlifNeuronException, 326

GlifNeuronMethod (*class in allensdk.model.glif.glif_neuron_methods*), 367

GlifOptimizer (*class in allensdk.internal.model.glif.glif_optimizer*), 326

GlifOptimizerNeuron (*class in allensdk.internal.model.glif.glif_optimizer_neuron*), 326

grating_to_screen() (*allensdk.brain_observatory.stimulus_info.BrainObservatoryMonitor method*), 224

grating_to_screen() (*allensdk.brain_observatory.stimulus_info.Monitor method*), 225

GRID_DATA_DIRECTORY (*allensdk.internal.api.queries.grid_data_api_prerelease.GridDataApiPrerelease attribute*), 290

grid_[illegible]() (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 378

GridDataApi (*class in allensdk.api.queries.grid_data_api*), 77

GridDataApiPrerelease (*class in allensdk.internal.api.queries.grid_data_api_prerelease*), 290

H

h (*allensdk.model.biophys_sim.neuron.hoc_utils.HocUtils attribute*), 359

h5_object_matcher_relname_in() (*in module allensdk.core.h5_utilities*), 250
handle_output_image() (*in module allensdk.internal.mouse_connectivity.projection_thumbnail.generate_projection_strip*), 347
handle_pyramid() (*in module allensdk.mouse_connectivity.grid.writers*), 380
has_fixed_dt() (*in module allensdk.ephys.ephys_features*), 286
has_overlaps() (*allensdk.core.structure_tree.StructureTree method*), 270
Hdf5Util (*class in allensdk.config.model.formats.hdf5_util*), 235
height (*allensdk.brain_observatory.stimulus_info.Monitor attribute*), 225
hex_pack() (*in module allensdk.brain_observatory.circle_plots*), 202
hex_to_rgb() (*allensdk.core.structure_tree.StructureTree static method*), 271
HocUtils (*class in allensdk.model.biophys_sim.neuron.hoc_utils*), 359
hook() (*in module allensdk.brain_observatory*), 232
host (*allensdk.core.authentication.DbCredentials attribute*), 241
HUMAN (*allensdk.api.queries.cell_types_api.CellTypesApi attribute*), 74
HUMAN_ORGANISM (*allensdk.api.queries.mouse_atlas_api.MouseAtlasApi attribute*), 82

I

identify_valid_masks() (*in module allensdk.brain_observatory.demixer*), 203
identify_valid_masks() (*in module allensdk.internal.brain_observatory.demixer*), 296
Image (*class in allensdk.brain_observatory.behavior.image_api*), 142
image_from_array() (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 378
image_names (*allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.StimulusTemplate attribute*), 127
image_set_name (*allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.StimulusTemplate attribute*), 127
ImageApi (*class in allensdk.brain_observatory.behavior.image_api*), 142
ImageDownloadApi (*class in allensdk.api.queries.image_download_api*), 79
ImageOutputStream (*class in allensdk.internal.brain_observatory.frame_stream*), 301
images (*allensdk.brain_observatory.behavior.stimulus_processing.stimulu attribute*), 127
ImageSeriesGridder (*class in allensdk.mouse_connectivity.grid.image_series_gridder*), 380
ImageSheet (*class in allensdk.internal.mouse_connectivity.projection_thumbnail.image_*), 347
imaging_depth (*allensdk.brain_observatory.behavior.metadata.behavior_ophys_me attribute*), 112
imaging_plane_group (*allensdk.brain_observatory.behavior.metadata.behavior_ophys_me attribute*), 112
imaging_plane_group_count (*allensdk.brain_observatory.behavior.metadata.behavior_ophys_me attribute*), 112
INCLUDE (*allensdk.api.queries.rma_api.RmaApi attribute*), 90
INDETERMINATE (*allensdk.core.cell_types_cache.ReporterStatus attribute*), 249
indicator (*allensdk.brain_observatory.behavior.metadata.behavior_oph attribute*), 112
infer_dims() (*allensdk.brain_observatory.circle_plots.CoronaPlotter method*), 201
infer_dims() (*allensdk.brain_observatory.circle_plots.FanPlotter method*), 201
info() (*allensdk.internal.model.biophysical.run_optimize.RunOptimize method*), 321
init_by_mask() (*allensdk.brain_observatory.roi_masks.NeuropilMask method*), 214
init_by_mask() (*allensdk.brain_observatory.roi_masks.RoiMask method*), 214
init_by_pixels() (*allensdk.brain_observatory.roi_masks.Mask method*), 214
initial_cr_point() (*in module allensdk.internal.brain_observatory.itracker_utils*), 302
initial_pupil_point() (*in module allensdk.internal.brain_observatory.itracker_utils*), 302
initialize_coarse_volume() (*allensdk.mouse_connectivity.grid.image_series_gridder.ImageSerie method*), 380
initialize_hoc() (*allensdk.model.biophys_sim.neuron.hoc_utils.HocUtils*

method), 360
initialize_image() (*allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile method*), 351
initialize_image() (*in module allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher*), 350
initialize_images() (*in module allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher*), 350
initiate_unique_seed() (*allensdk.internal.model.glif.glif_optimizer.GlifOptimizer method*), 326
INJECTION_DENSITY (*allensdk.api.queries.grid_data_api.GridDataApi attribute*), 77
INJECTION_DENSITY_KEY (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute*), 253
INJECTION_ENERGY (*allensdk.api.queries.grid_data_api.GridDataApi attribute*), 77
INJECTION_FRACTION (*allensdk.api.queries.grid_data_api.GridDataApi attribute*), 77
INJECTION_FRACTION_KEY (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute*), 253
input_data() (*allensdk.internal.core.lims_pipeline_module.PipelineModule method*), 309
input_resistance() (*in module allensdk.ephys.ephys_extractor*), 280
InputFile (*class in allensdk.brain_observatory.argschema_utilities*), 197
insert_iclamp() (*allensdk.internal.model.biophysical.deap_utils.Utils method*), 320
INTEGER (*allensdk.api.queries.connected_services.ConnectedServices attribute*), 76
INTENSITY (*allensdk.api.queries.grid_data_api.GridDataApi attribute*), 77
IntensitySubImage (*class in allensdk.mouse_connectivity.grid.subimage.base_subimage*), 374
interlength (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise attribute*), 208
interlength (*allensdk.brain_observatory.natural_scenes.NaturalScenes attribute*), 210
interlength (*allensdk.brain_observatory.static_gratings.StaticGratings attribute*), 221
interpolate_RF() (*in module allensdk.brain_observatory.receptive_field_analysis.chisquarerf*), 190
interpolate_spike_time() (*in module allensdk.model.glif.glif_neuron*), 366
interpolate_spike_value() (*in module allensdk.model.glif.glif_neuron*), 366
interpolate_spike_voltage() (*in module allensdk.internal.model.glif.glif_optimizer_neuron*), 329
intersection() (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303
intersection_size() (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303
IntervalUnionizer (*class in allensdk.internal.mouse_connectivity.interval_unionize.interval_un*
343
invnl() (*in module allensdk.internal.model.GLM*), 334
IS (*allensdk.api.queries.rma_api.RmaApi attribute*), 90
is_change_event() (*in module allensdk.brain_observatory.behavior.stimulus_processing*), 131
is_empty() (*allensdk.config.model.description.Description method*), 237
is_equal() (*in module allensdk.brain_observatory.session_api_utils*), 220
is_spike_feature_affected_by_clipping() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 277
is_well_known_file_type() (*allensdk.api.queries.biophysical_api.BiophysicalApi method*), 70
isicv() (*allensdk.ephys.feature_extractor.EphysFeatureExtractor method*), 288
items() (*allensdk.brain_observatory.behavior.stimulus_processing.stimu*
method), 127

J

json_handler() (*in module allensdk.core.json_utilities*), 251
json_msg_query() (*allensdk.api.api.Api method*), 104
json_remove_keys() (*allensdk.api.warehouse_cache.cache.Cache static method*), 98
json_rename_columns() (*allensdk.api.warehouse_cache.cache.Cache static method*), 98
JsonComments (*class in allensdk.core.json_utilities*), 251
JsonDescriptionParser (*class in allensdk.config.model.formats.json_description_parser*), 235

JSONEncoder (*class in allensdk.brain_observatory*), 232

K

keyed_locate_h5_objects() (*in module allensdk.core.h5_utilities*), 250
keys() (*allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.StimulusTemplate method*), 127

L

label_data() (*allensdk.internal.brain_observatory.roi_filter_utils.TrainingMultiLabelClassifier method*), 305
latency() (*in module allensdk.ephys.ephys_features*), 287
LazyProperty (*allensdk.core.lazy_property.lazy_property_mixin.LazyPropertyMixin attribute*), 241
LazyProperty (*class in allensdk.core.lazy_property.lazy_property*), 241
LazyPropertyMixin (*class in allensdk.core.lazy_property.lazy_property_mixin*), 241
least_squares_RCEl_calc_tested() (*in module allensdk.internal.model.glif.rc*), 330
legacy() (*in module allensdk.deprecated*), 382
legend_artist() (*allensdk.brain_observatory.observatory_plots.DimensionPatchHandler method*), 210
lims_working_directory() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 289
lims_working_directory() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 291
LimsApi (*class in allensdk.internal.api.lims_api*), 293
line_crossing_x() (*in module allensdk.model.glif.glif_neuron*), 366
line_crossing_y() (*in module allensdk.model.glif.glif_neuron*), 366
line_stats() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 231
linear_transform_from_intervals() (*in module allensdk.brain_observatory.ecephys.align_timestamps.barcode*), 166
linux_to_windows() (*in module allensdk.internal.core.lims_utilities*), 310
list_cells() (*allensdk.api.queries.cell_types_api.CellTypesApi method*), 75
list_cells_api() (*allensdk.api.queries.cell_types_api.CellTypesApi method*), 75
list_column_definition_class_names() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method*), 73
list_isi_experiments() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method*), 73
list_neuronal_models() (*allensdk.api.queries.glif_api.GlifApi method*), 77
list_of_dicts_to_dict_of_lists() (*in module allensdk.brain_observatory.receptive_field_analysis.tools*), 193
list_stimuli() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method*), 246
load() (*allensdk.brain_observatory.ecephys.file_io.continuous_file.Conti method*), 172
load() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 231
load() (*allensdk.config.app.application_config.ApplicationConfig method*), 234
load() (*allensdk.model.biophys_sim.config.Config method*), 360
load_annotation() (*in module allensdk.internal.mouse_connectivity.interval_unionize.data_utilitie*), 343
load_api_schema() (*allensdk.api.api.Api method*), 104
load_arrays() (*in module allensdk.internal.pipeline_modules.run_annotated_region_metrics*), 352
load_cell_parameters() (*allensdk.internal.model.biophysical.deap_utils.Utils method*), 320
load_cell_parameters() (*allensdk.model.biophysical.utils.AllActiveUtils method*), 362
load_cell_parameters() (*allensdk.model.biophysical.utils.Utils method*), 363
load_config() (*allensdk.config.manifest.Manifest method*), 239
load_csv() (*allensdk.api.warehouse_cache.cache.Cache method*), 98
load_datasets_by_relnames() (*in module allensdk.core.h5_utilities*), 250
load_description() (*in module allensdk.model.biophysical.runner*), 361
load_experiment() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
load_experiment() (*in module allensdk.internal.ephys.plot_qc_figures3*), 315
load_eye_tracking_hdf() (*in module allensdk.brain_observatory.behavior.eye_tracking_processing*),

140
load_frame() (*in module allensdk.internal.brain_observatory.ophys_session_decomposition*), 304
load_json() (*allensdk.api.warehouse_cache.cache.Cache method*), 98
load_manifest() (*allensdk.api.warehouse_cache.cache.Cache method*), 99
load_manifest() (*allensdk.internal.model.biophysical.run_optimize.RunOptimize method*), 321
load_manifest() (*allensdk.model.biophysical.run_simulate.RunSimulate method*), 361
load_morphology() (*in module allensdk.internal.model.biophysical.passive_fitting.neuron_utils*), 319
load_pickle() (*in module allensdk.brain_observatory.behavior.stimulus_processing*), 131
load_sweep() (*in module allensdk.internal.model.data_access*), 334
load_sweep() (*in module allensdk.model.glif.simulate_neuron*), 372
load_sweeps() (*in module allensdk.internal.model.data_access*), 334
local_path (*allensdk.api.cloud_cache.file_attributes.CacheFileAttributes attribute*), 66
local_time() (*in module allensdk.brain_observatory.behavior.trials_processing*), 164
LocallySparseNoise (*class in allensdk.brain_observatory.locally_sparse_noise*), 207
locate_h5_objects() (*in module allensdk.core.h5_utilities*), 250
locate_median() (*in module allensdk.brain_observatory.receptive_field_analysis.chisquare*), 190
log (*allensdk.config.manifest.Manifest attribute*), 239
log (*allensdk.config.model.description_parser.DescriptionParser attribute*), 237
log (*allensdk.config.model.formats.json_description_parser.JsonDescriptionParser attribute*), 235
log (*allensdk.config.model.formats.pycfg_description_parser.PycfgDescriptionParser attribute*), 236
long_squares_features() (*allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor method*), 276
long_squares_stim_amps() (*allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor method*), 276
LSN (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise attribute*), 207
lsn_coordinate_to_monitor_coordinate() (*in module allensdk.brain_observatory.stimulus_info*), 225
LSN_GREY (*allensdk.brain_observatory.locally_sparse_noise.LocallySpars attribute*), 207
lsn_image_to_screen() (*allensdk.brain_observatory.stimulus_info.BrainObservatoryMonito method*), 224
lsn_image_to_screen() (*allensdk.brain_observatory.stimulus_info.Monitor method*), 225
LSN_mask (*allensdk.brain_observatory.locally_sparse_noise.LocallySpars attribute*), 208
LSN_OFF (*allensdk.brain_observatory.locally_sparse_noise.LocallySparse attribute*), 207
LSN_OFF_SCREEN (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoi attribute*), 207
LSN_ON (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseN attribute*), 208
lsna_check_hvas() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 356

M

main() (*in module allensdk.brain_observatory.dff*), 205
main() (*in module allensdk.brain_observatory.ecephys.stimulus_table.visualization.vie*), 176
main() (*in module allensdk.brain_observatory.session_analysis*), 219
main() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
main() (*in module allensdk.internal.model.biophysical.passive_fitting.neuron_passive_*), 319
main() (*in module allensdk.internal.model.biophysical.passive_fitting.neuron_passive_*), 319
main() (*in module allensdk.internal.model.biophysical.passive_fitting.neuron_passive_*), 319
main() (*in module allensdk.internal.model.biophysical.passive_fitting.preprocess*), 319
main() (*in module allensdk.internal.model.biophysical.run_optimize*), 321
main() (*in module allensdk.internal.model.biophysical.run_passive_fit*), 322

main() (*in module allensdk.internal.model.biophysical.run_simulate_lims*), 322
main() (*in module allensdk.internal.model.glif.find_sweeps*), 324
main() (*in module allensdk.internal.morphology.validate_swc*), 342
main() (*in module allensdk.internal.pipeline_modules.gbm.generate_gbm_analysis_run_records*), 351
main() (*in module allensdk.internal.pipeline_modules.gbm.generate_gbm_heatmap*), 351
main() (*in module allensdk.internal.pipeline_modules.gbm.generate_gbm_sample_metadata*), 352
main() (*in module allensdk.internal.pipeline_modules.run_annotated_region_metrics*), 352
main() (*in module allensdk.internal.pipeline_modules.run_demixing*), 352
main() (*in module allensdk.internal.pipeline_modules.run_dff_computation*), 352
main() (*in module allensdk.internal.pipeline_modules.run_neuropil_correction*), 353
main() (*in module allensdk.internal.pipeline_modules.run_observatory_analysis*), 353
main() (*in module allensdk.internal.pipeline_modules.run_observatory_thumbnails*), 356
main() (*in module allensdk.internal.pipeline_modules.run_ophys_eye_calibration*), 356
main() (*in module allensdk.internal.pipeline_modules.run_ophys_session_decomposition*), 356
main() (*in module allensdk.internal.pipeline_modules.run_ophys_time_sync*), 358
main() (*in module allensdk.internal.pipeline_modules.run_tissuecyte_unionize_cluster*), 359
main() (*in module allensdk.model.biophysical.run_simulate*), 361
main() (*in module allensdk.model.glif.simulate_neuron*), 372
main() (*in module allensdk.morphology.validate_swc*), 373
make_bbs() (*in module allensdk.internal.brain_observatory.mask_set*), 303
make_blended_tile() (*in module allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher*), 350
make_category_dummy() (*in module allensdk.brain_observatory.chisquare_categorical*), 201
make_cell_html() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
make_cell_html() (*in module allensdk.internal.ephys.plot_qc_figures3*), 315
make_cell_page() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
make_cell_page() (*in module allensdk.internal.ephys.plot_qc_figures3*), 315
make_display_mask() (*in module allensdk.brain_observatory.stimulus_info*), 225
make_fit() (*allensdk.internal.model.biophysical.run_optimize.RunOpti method*), 321
make_fit_json_file() (*allensdk.internal.model.biophysical.make_deap_fit_json.Report method*), 321
make_pincushion_plot() (*in module allensdk.brain_observatory.circle_plots*), 202
make_pixel_counter() (*allensdk.mouse_connectivity.grid.subimage.base_subimage.SubIma method*), 375
make_ratio_volume() (*allensdk.mouse_connectivity.grid.image_series_gridder.ImageSerie method*), 380
make_spontaneous_activity_tables() (*in module allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spik*), 180
make_structure_mask() (*allensdk.core.reference_space.ReferenceSpace method*), 261
make_sweep_html() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
make_sweep_html() (*in module allensdk.internal.ephys.plot_qc_figures3*), 316
make_sweep_page() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
make_sweep_page() (*in module allensdk.internal.ephys.plot_qc_figures3*), 316
makeBasis_StimKernel() (*in module allensdk.internal.model.GLM*), 334

makeBasis_StimKernel_exp() (*in module allensdk.internal.model.GLM*), 334
makeFitStruct_GLM() (*in module allensdk.internal.model.GLM*), 334
Manifest (*class in allensdk.api.cloud_cache.manifest*), 66
Manifest (*class in allensdk.config.manifest*), 238
manifest_dataframe() (*allensdk.api.warehouse_cache.cache.Cache method*), 99
MANIFEST_VERSION (*allensdk.core.cell_types_cache.CellTypesCache attribute*), 247
MANIFEST_VERSION (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute*), 253
MANIFEST_VERSION (*allensdk.core.reference_space_cache.ReferenceSpaceCache attribute*), 263
ManifestBuilder (*class in allensdk.config.manifest_builder*), 240
ManifestVersionError, 240
many_structure_masks() (*allensdk.core.reference_space.ReferenceSpace method*), 262
map_column_names() (*in module allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities*), 181
map_monitor_coordinate_to_stimulus_coordinate() (*in module allensdk.brain_observatory.stimulus_info*), 226
map_monitor_coordinate_to_template_coordinate() (*in module allensdk.brain_observatory.stimulus_info*), 226
map_stimulus() (*allensdk.brain_observatory.stimulus_info.Monitor method*), 225
map_stimulus() (*in module allensdk.brain_observatory.stimulus_info*), 226
map_stimulus_coordinate_to_monitor_coordinate() (*in module allensdk.brain_observatory.stimulus_info*), 226
map_stimulus_names() (*in module allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities*), 181
map_template_coordinate_to_monitor_coordinate() (*in module allensdk.brain_observatory.stimulus_info*), 226
Marker (*class in allensdk.core.swc*), 271
Marker (*class in allensdk.internal.core.swc*), 312
MARKER_FILE_TYPE (*allensdk.api.queries.cell_types_api.CellTypesApi attribute*), 74
MARKER_KEY (*allensdk.core.cell_types_cache.CellTypesCache attribute*), 247
mask (*allensdk.brain_observatory.stimulus_info.Monitor attribute*), 225
Mask (*class in allensdk.brain_observatory.roi_masks*), 214
mask() (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303
mask_is_union_of_set() (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303
mask_nulls() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
mask_nulls() (*in module allensdk.internal.ephys.plot_qc_figures3*), 316
mask_stimulus_template() (*in module allensdk.brain_observatory.stimulus_info*), 226
MaskSet (*class in allensdk.internal.brain_observatory.mask_set*), 303
match_barcodes() (*in module allensdk.brain_observatory.ecephys.align_timestamps.barcode*), 167
max_cb() (*in module allensdk.internal.mouse_connectivity.projection_thumbnail.generat*
347
max_of_line_and_const() (*in module allensdk.model.glif.glif_neuron_methods*), 369
max_projection() (*in module allensdk.internal.mouse_connectivity.projection_thumbnail.projecti*
348
max_voxel_density (*allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_u*
attribute), 345
max_voxel_index (*allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_u*
attribute), 345
mean_features_spike_zero() (*in module allensdk.ephys.extract_cell_features*), 287
mean_response (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoi*
attribute), 208
mean_sweep_response (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 223
medfilt_custom() (*in module allensdk.internal.brain_observatory.itracker_utils*), 302
median_absolute_deviation() (*in module al-*

lensdk.internal.brain_observatory.itracker_utils), 302
meets_engagement_criteria() (*in module allensdk.brain_observatory.behavior.criteria*), 136
membrane_time_constant() (*in module allensdk.ephys.ephys_extractor*), 280
memoize() (*in module allensdk.api.warehouse_cache.cache*), 100
merge_mean_response() (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise static method*), 208
metadata_file_attributes() (*allensdk.api.cloud_cache.manifest.Manifest method*), 66
metadata_file_names (*allensdk.api.cloud_cache.manifest.Manifest attribute*), 67
METHOD (*allensdk.core.authentication.CredentialProvider attribute*), 241
METHOD (*allensdk.core.authentication.EnvCredentialProvider attribute*), 242
midpoint() (*in module allensdk.internal.morphology.node*), 342
min_of_line_and_zero() (*in module allensdk.model.glif.glif_neuron_methods*), 369
minmax_norm() (*in module allensdk.internal.mouse_connectivity.projection_thumbnail.visualization_utilities*), 348
MissingDataError, 250
MissingStimulusException, 200
MissingSweepException, 323
MLIN() (*in module allensdk.internal.model.glif.MLIN*), 322
MLIN_list_error() (*in module allensdk.internal.model.glif.error_functions*), 323
mod_file_entries() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 289
mod_file_entries() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 291
mod_file_paths() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 289
mod_file_paths() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 292
MOD_FILE_TYPE_ID (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader attribute*), 289
MOD_FILE_TYPE_ID (*al-*
lensdk.internal.api.queries.optimize_config_reader.OptimizeConf *attribute*), 291
MODEL (*allensdk.api.queries.rma_api.RmaApi attribute*), 90
model_query() (*allensdk.api.queries.rma_api.RmaApi method*), 91
model_stage() (*allensdk.api.queries.rma_api.RmaApi method*), 92
model_type() (*allensdk.internal.api.queries.biophysical_module_reade method*), 289
modify_parameter() (*allensdk.model.glif.glif_neuron_methods.GlifNeuronMethod method*), 367
moments2() (*in module allensdk.brain_observatory.receptive_field_analysis.fitgaussian2D*), 192
Monitor (*class in allensdk.brain_observatory.stimulus_info*), 224
monitor_coordinate_to_lsn_coordinate() (*in module allensdk.brain_observatory.stimulus_info*), 227
monitor_coordinate_to_natural_movie_coordinate() (*in module allensdk.brain_observatory.stimulus_info*), 227
monitor_delay (*allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute*), 307
Morphology (*class in allensdk.core.swc*), 271
Morphology (*class in allensdk.internal.morphology.morphology*), 335
MORPHOLOGY_FEATURES_KEY (*allensdk.core.cell_types_cache.CellTypesCache attribute*), 247
morphology_entries() (*allensdk.internal.api.queries.biophysical_module_reader.Biophysic method*), 289
morphology_entries() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConf method*), 292
morphology_path() (*allensdk.internal.api.queries.biophysical_module_reader.Biophysic method*), 289
morphology_path() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConf method*), 292
MORPHOLOGY_TYPE_ID (*allensdk.internal.api.queries.biophysical_module_reader.Biophysic attribute*), 289

MORPHOLOGY_TYPE_ID (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader attribute*), 291
MorphologyColors (*class in allensdk.internal.morphology.morphvis*), 339
mostly_useful() (*in module allensdk.brain_observatory.behavior.criteria*), 136
MOTION_CORRECTION_DATASETS (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet attribute*), 242
MOUSE (*allensdk.api.queries.cell_types_api.CellTypesApi attribute*), 74
MOUSE_2011 (*allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute*), 88
MOUSE_ATLAS_PRODUCTS (*allensdk.api.queries.mouse_atlas_api.MouseAtlasApi attribute*), 82
mouse_id (*allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata attribute*), 110
MOUSE_ORGANISM (*allensdk.api.queries.mouse_atlas_api.MouseAtlasApi attribute*), 82
MouseAtlasApi (*class in allensdk.api.queries.mouse_atlas_api*), 82
MouseConnectivityApi (*class in allensdk.api.queries.mouse_connectivity_api*), 83
MouseConnectivityApiPrerelease (*class in allensdk.internal.api.queries.mouse_connectivity_api_prerelease*), 291
MouseConnectivityCache (*class in allensdk.core.mouse_connectivity_cache*), 252
MouseConnectivityCachePrerelease (*class in allensdk.internal.core.mouse_connectivity_cache_prerelease*), 310
movingaverage() (*in module allensdk.brain_observatory.dff*), 205
movingmode_fast() (*in module allensdk.brain_observatory.dff*), 205
multi_dataframe_merge() (*in module allensdk.brain_observatory.session_analysis*), 219

N

n_complete() (*in module allensdk.brain_observatory.behavior.criteria*), 136
NA (*allensdk.core.cell_types_cache.ReporterStatus attribute*), 249
name (*allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.StimulusImage attribute*), 127
nan_get() (*in module allensdk.internal.ephys.core_feature_extract*), 313
natural_movie_coordinate_to_monitor_coordinate() (*in module allensdk.brain_observatory.stimulus_info*), 227
natural_movie_image_to_screen() (*allensdk.brain_observatory.stimulus_info.Monitor method*), 225
natural_scene_coordinate_to_monitor_coordinate() (*in module allensdk.brain_observatory.stimulus_info*), 227
natural_scene_image_to_screen() (*allensdk.brain_observatory.stimulus_info.Monitor method*), 225
NaturalMovie (*class in allensdk.brain_observatory.natural_movie*), 208
NaturalScenes (*class in allensdk.brain_observatory.natural_scenes*), 209
nearest_neuron_sampling_rate() (*allensdk.model.biophysical.utils.Utils static method*), 363
needs_data_refresh() (*allensdk.brain_observatory.session_api_utils.ParamsMixin method*), 220
NEGATIVE (*allensdk.core.cell_types_cache.ReporterStatus attribute*), 249
neurodata_doc (*allensdk.brain_observatory.behavior.schemas.BehaviorMetadataSc attribute*), 147
neurodata_doc (*allensdk.brain_observatory.behavior.schemas.BehaviorTaskParame attribute*), 149
neurodata_doc (*allensdk.brain_observatory.behavior.schemas.OphysBehaviorMetac attribute*), 153
neurodata_doc (*allensdk.brain_observatory.behavior.schemas.OphysEyeTrackingRi attribute*), 155
neurodata_doc (*allensdk.brain_observatory.behavior.schemas.SubjectMetadataSche attribute*), 157
neurodata_skip (*allensdk.brain_observatory.behavior.schemas.BehaviorMetadataSc attribute*), 147
neurodata_skip (*allensdk.brain_observatory.behavior.schemas.OphysBehaviorMetac attribute*), 153
neurodata_skip (*allensdk.brain_observatory.behavior.schemas.SubjectMetadataSche*

attribute), 157
neurodata_type (*allensdk.brain_observatory.behavior.schemas.BehaviorMetadataSchema attribute*), 148
neurodata_type (*allensdk.brain_observatory.behavior.schemas.BehaviorTaskParametersSchema attribute*), 149
neurodata_type (*allensdk.brain_observatory.behavior.schemas.OphysBehaviorMetadataSchema attribute*), 154
neurodata_type (*allensdk.brain_observatory.behavior.schemas.OphysEyeTrackingRigMetadataSchema attribute*), 155
neurodata_type (*allensdk.brain_observatory.behavior.schemas.SubjectMetadataSchema attribute*), 157
neurodata_type_inc (*allensdk.brain_observatory.behavior.schemas.BehaviorMetadataSchema attribute*), 148
neurodata_type_inc (*allensdk.brain_observatory.behavior.schemas.BehaviorTaskParametersSchema attribute*), 149
neurodata_type_inc (*allensdk.brain_observatory.behavior.schemas.OphysBehaviorMetadataSchema attribute*), 154
neurodata_type_inc (*allensdk.brain_observatory.behavior.schemas.OphysEyeTrackingRigMetadataSchema attribute*), 155
neurodata_type_inc (*allensdk.brain_observatory.behavior.schemas.SubjectMetadataSchema attribute*), 158
neuron (*allensdk.model.biophys_sim.neuron.hoc_utils.HocUtils attribute*), 360
neuron_parameter_count() (*allensdk.internal.model.glif.glif_experiment.GlifExperiment method*), 324
neuronal_model_optimize_dir() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 292
NEURONAL_MODEL_PARAMETERS (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader attribute*), 291
neuronal_model_run_dir() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 289
NeuropilMask (*class in allensdk.brain_observatory.roi_masks*), 214
NeuropilSubtract (*class in allensdk.brain_observatory.r_neuropil*), 212
new_image() (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 378
nlin() (*in module allensdk.internal.model.GLM*), 334
NLL_to_pvalue() (*in module allensdk.brain_observatory.receptive_field_analysis.chisquarerf*), 187
NO_RECONSTRUCTION (*allensdk.core.swc.Marker attribute*), 271
NO_RECONSTRUCTION (*allensdk.internal.core.swc.Marker attribute*), 312
no_response_bias() (*in module allensdk.brain_observatory.behavior.criteria*), 136
nocache_dataframe() (*allensdk.api.warehouse_cache.cache.Cache static method*), 99
nocache_json() (*allensdk.api.warehouse_cache.cache.Cache static method*), 99
Node (*class in allensdk.internal.morphology.node*), 342
node() (*allensdk.core.simple_tree.SimpleTree method*), 266
node() (*allensdk.core.swc.Morphology method*), 274
node() (*allensdk.internal.core.simpletree.SimpleTree method*), 311
node() (*allensdk.internal.morphology.morphology.Morphology method*), 337
node_ids() (*allensdk.core.simple_tree.SimpleTree method*), 266
node_ids() (*allensdk.internal.core.simpletree.SimpleTree method*), 311
node_list (*allensdk.internal.morphology.morphology.Morphology attribute*), 337
node_list_by_type() (*allensdk.internal.morphology.morphology.Morphology method*), 337
NODE_TYPES (*allensdk.core.swc.Morphology attribute*), 272
NODE_TYPES (*allensdk.internal.morphology.morphology.Morphology attribute*), 335
nodes() (*allensdk.core.simple_tree.SimpleTree method*), 267
nodes() (*allensdk.internal.core.simpletree.SimpleTree method*), 311
nodes_by_property() (*allensdk.core.simple_tree.SimpleTree method*), 267
NoEyeTrackingException, 200
noise_std() (*in module allensdk.brain_observatory.dff*), 205
nonraising_ks_2samp() (*in module allensdk.brain_observatory.stimulus_analysis*), 224
norm_diff() (*in module allensdk.ephys.ephys_features*), 287
norm_sq_diff() (*in module allensdk.ephys.ephys_features*), 287

normalize_actual_parameters() (allensdk.internal.model.biophysical.deap_utils.Utils method), 320
normalize_F() (in module allensdk.brain_observatory.r_neuropil), 213
normalize_intensity() (in module allensdk.internal.mouse_connectivity.projection_thumbnail.visualization_utilities), 348
normalizecols() (in module allensdk.internal.model.GLM), 334
np_sitk_convert() (in module allensdk.mouse_connectivity.grid.utilities.image_utilities), 379
nrn (allensdk.model.biophys_sim.neuron.hoc_utils.HocUtils attribute), 360
nrnivmodl() (allensdk.internal.model.biophysical.run_optimize.RunOptimize method), 321
nrnivmodl() (allensdk.model.biophysical.run_simulate.RunSimulate method), 361
num_contingent_trials() (in module allensdk.brain_observatory.behavior.session_metrics), 158
num_nodes (allensdk.core.swc.Morphology attribute), 274
num_nodes (allensdk.internal.morphology.morphology.Morphology attribute), 338
NUM_ROWS (allensdk.api.queries.rma_api.RmaApi attribute), 90
num_trees (allensdk.core.swc.Morphology attribute), 274
num_trees (allensdk.internal.morphology.morphology.Morphology attribute), 338
number_of_cells (allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet attribute), 246
number_ori (allensdk.brain_observatory.drifting_gratings.DriftingGratings attribute), 207
number_ori (allensdk.brain_observatory.static_gratings.StaticGratings attribute), 221
number_phase (allensdk.brain_observatory.static_gratings.StaticGratings attribute), 221
number_scenes (allensdk.brain_observatory.natural_scenes.NaturalScenes attribute), 210
number_sf (allensdk.brain_observatory.static_gratings.StaticGratings attribute), 221
number_tf (allensdk.brain_observatory.drifting_gratings.DriftingGratings attribute), 207
numbercells (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute), 223
NWB_FILE_TYPE (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi attribute), 71
NWB_FILE_TYPE (allensdk.api.queries.cell_types_api.CellTypesApi attribute), 74
NWB_FILE_TYPE (allensdk.api.queries.glif_api.GlifApi attribute), 76
NwbDataSet (class in allensdk.core.nwb_data_set), 257
NwbOphysMetadataSchema (class in allensdk.brain_observatory.behavior.schemas), 151

O

object_norm_eye_coordinates() (in module allensdk.internal.brain_observatory.eye_calibration), 299
object_rotation_matrix() (in module allensdk.internal.brain_observatory.eye_calibration), 299
one() (in module allensdk), 382
one_file_call_caching() (in module allensdk.api.warehouse_cache.caching_utilities), 101
OneOrMoreResultExpectedError, 293
OneResultExpectedError, 382
ONLY (allensdk.api.queries.rma_api.RmaApi attribute), 90
only_except_tabular_clause() (allensdk.api.queries.rma_api.RmaApi method), 92
OntologiesApi (class in allensdk.api.queries.ontologies_api), 86
Ontology (class in allensdk.core.ontology), 259
open() (allensdk.internal.brain_observatory.frame_stream.CvInputStream method), 301
open() (allensdk.internal.brain_observatory.frame_stream.FfmpegInputStream method), 301
open() (allensdk.internal.brain_observatory.frame_stream.FfmpegOutputStream method), 301
open() (allensdk.internal.brain_observatory.frame_stream.FrameInputStream method), 301
open() (allensdk.internal.brain_observatory.frame_stream.FrameOutputStream method), 301
open_corona_plot() (allensdk.brain_observatory.natural_scenes.NaturalScenes method), 210
open_fan_plot() (allensdk.brain_observatory.static_gratings.StaticGratings method), 221
open_pincushion_plot() (allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise method), 208
open_star_plot() (allensdk.brain_observatory.drifting_gratings.DriftingGratings method), 207

open_track_plot() (allensdk.brain_observatory.natural_movie.NaturalMovie method), 209
open_view_on_binary() (in module allensdk.internal.brain_observatory.ophys_session_decomposition), 304
OPHYS_ANALYSIS_FILE_TYPE (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi attribute), 71
ophys_delta (allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncOutputs attribute), 357
OPHYS_EVENTS_FILE_TYPE (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi attribute), 71
ophys_experiment_id (allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata.BehaviorOphysMetadata attribute), 112
ophys_frame_rate (allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata.BehaviorOphysMetadata attribute), 112
ophys_session_id (allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata.BehaviorOphysMetadata attribute), 112
ophys_times (allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncOutputs attribute), 357
ophys_timestamps (allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute), 307
OphysBehaviorMetadataSchema (class in allensdk.brain_observatory.behavior.schemas), 152
OphysEyeTrackingRigMetadataSchema (class in allensdk.brain_observatory.behavior.schemas), 154
OphysMetadataSchema (class in allensdk.brain_observatory.behavior.schemas), 155
OphysMixin (class in allensdk.brain_observatory.behavior.behavior_project_cache.tables.ophys_mixin), 108
OphysTimeAligner (class in allensdk.internal.brain_observatory.time_sync), 307
OptimizeConfigReader (class in allensdk.internal.api.queries.optimize_config_reader), 291
optional_lims_inputs() (in module allensdk.brain_observatory.argschema_utilities), 199
optional_polys (allensdk.mouse_connectivity.grid.subimage.base_subimage.PolygonSubImage attribute), 374
optional_polys (allensdk.mouse_connectivity.grid.subimage.classic_subimage.Class attribute), 376
OPTIONS (allensdk.api.queries.rma_api.RmaApi attribute), 90
options_clause() (allensdk.api.queries.rma_api.RmaApi method), 93
ORTOGONAL_STIMULATION_KEYS (allensdk.brain_observatory.sync_dataset.Dataset attribute), 238
opts (allensdk.brain_observatory.argschema_utilities.RaisingSchema attribute), 199
opts (allensdk.brain_observatory.behavior.mtrain.ExtendedTrialSchema attribute), 143
opts (allensdk.brain_observatory.behavior.schemas.BehaviorMetadataSch attribute), 148
opts (allensdk.brain_observatory.behavior.schemas.BehaviorTaskParamet attribute), 149
opts (allensdk.brain_observatory.behavior.schemas.CompleteOphysBehav attribute), 150
opts (allensdk.brain_observatory.behavior.schemas.EyeTrackingRigGeom attribute), 151
opts (allensdk.brain_observatory.behavior.schemas.NwbOphysMetadataS attribute), 152
opts (allensdk.brain_observatory.behavior.schemas.OphysBehaviorMetad attribute), 154
opts (allensdk.brain_observatory.behavior.schemas.OphysEyeTrackingRig attribute), 155
opts (allensdk.brain_observatory.behavior.schemas.OphysMetadataSchem attribute), 156
opts (allensdk.brain_observatory.behavior.schemas.RaisingSchema attribute), 156
opts (allensdk.brain_observatory.behavior.schemas.SubjectMetadataSche attribute), 158
ORDER (allensdk.api.queries.rma_api.RmaApi attribute), 90
order_clause() (allensdk.api.queries.rma_api.RmaApi method), 93
order_roi_by_object_list() (in module allensdk.internal.brain_observatory.roi_filter_utils), 306
organize_sweeps_by_name() (in module allensdk.internal.model.glif.find_sweeps), 324
orivals (allensdk.brain_observatory.drifting_gratings.DriftingGratings attribute), 207
orivals (allensdk.brain_observatory.static_gratings.StaticGratings attribute), 222
outdated (allensdk.config.manifest.ManifestVersionError attribute), 240
outlier_cost() (allensdk.brain_observatory.fit_ellipse.FitEllipse method), 300
output() (allensdk.internal.mouse_connectivity.interval_unionize.tissued

method), 345
output() (allensdk.internal.mouse_connectivity.interval_unionize.unionize_record.Unionize method), 346
output_directory() (allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method), 292
OutputFile (class in allensdk.brain_observatory.argschema_utilities), 198
OutputGrabber (class in allensdk.internal.model.biophysical.passive_fitting.output_grabber), 319
overlap_fraction() (allensdk.internal.brain_observatory.mask_set.MaskSet method), 303
overlaps_motion_border (allensdk.brain_observatory.roi_masks.Mask attribute), 214

P

pageable() (in module allensdk.api.queries.rma_pager), 94
pager() (allensdk.api.queries.rma_pager.RmaPager static method), 94
panel_size (allensdk.brain_observatory.stimulus_info.Monitor attribute), 225
ParamsMixin (class in allensdk.brain_observatory.session_api_utils), 219
parent() (allensdk.core.simple_tree.SimpleTree method), 267
parent() (allensdk.internal.core.simpletree.SimpleTree method), 312
parent_id() (allensdk.core.simple_tree.SimpleTree method), 267
parent_id() (allensdk.internal.core.simpletree.SimpleTree method), 312
parent_ids() (allensdk.core.simple_tree.SimpleTree method), 267
parent_of() (allensdk.core.swc.Morphology method), 274
parent_of() (allensdk.internal.morphology.morphology.Morphology method), 338
parents() (allensdk.core.simple_tree.SimpleTree method), 267
parse_age_in_days() (allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata static method), 110
parse_arguments() (in module allensdk.internal.model.glif.find_sweeps), 324
parse_arguments() (in module allensdk.model.glif.simulate_neuron), 372
parse_command_line_args() (allensdk.config.app.application_config.ApplicationConfig method), 234
parse_cre_line() (allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata static method), 111
parse_indicator() (allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata static method), 111
parse_input() (in module allensdk.internal.pipeline_modules.run_demixing), 352
parse_input() (in module allensdk.internal.pipeline_modules.run_dff_computation), 352
parse_input() (in module allensdk.internal.pipeline_modules.run_observatory_thumbnails), 356
parse_input() (in module allensdk.internal.pipeline_modules.run_ophys_session_decomposition), 356
parse_input_data() (in module allensdk.internal.pipeline_modules.run_ophys_eye_calibration), 356
parse_neuron_output() (in module allensdk.internal.model.biophysical.passive_fitting.neuron_utils), 319
parse_obj() (in module allensdk.core.obj_utilities), 259
parse_reporter_line() (allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata static method), 111
parse_stim_repr() (in module allensdk.brain_observatory.ecephys.stimulus_table.stimulus_parameter_extraction), 183
parser_for_extension() (allensdk.config.model.description_parser.DescriptionParser method), 237
password (allensdk.core.authentication.DbCredentials attribute), 241
paste_slice() (allensdk.mouse_connectivity.grid.image_series_gridder.ImageSeriesGridder method), 380
paste_subimage() (allensdk.mouse_connectivity.grid.image_series_gridder.ImageSeriesGridder method), 380
path_to_list() (allensdk.core.structure_tree.StructureTree static method), 271
pathfinder() (allensdk.api.warehouse_cache.cache.Cache static method), 99
pause_metrics() (allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method), 277
peak (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute), 223

peak_run (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 223
period() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 231
phasevals (*allensdk.brain_observatory.static_gratings.StaticGratings attribute*), 222
PHOTODIODE_KEYS (*allensdk.brain_observatory.sync_dataset.Dataset attribute*), 228
PIPE (*allensdk.api.queries.rma_api.RmaApi attribute*), 90
pipe_stage() (*allensdk.api.queries.rma_api.RmaApi method*), 93
PIPELINE_DATASET (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryNwbDataSet attribute*), 242
PipelineModule (*class in allensdk.internal.core.lims_pipeline_module*), 309
pixel_counter (*allensdk.mouse_connectivity.grid.subimage.base_subimage.SubImage attribute*), 375
pixel_size (*allensdk.brain_observatory.stimulus_info.Monitor attribute*), 225
pixels_to_visual_degrees() (*allensdk.brain_observatory.stimulus_info.BrainObservatoryMonitor method*), 224
pixels_to_visual_degrees() (*allensdk.brain_observatory.stimulus_info.Monitor method*), 225
plot() (*allensdk.brain_observatory.circle_plots.CoronaPlotter method*), 201
plot() (*allensdk.brain_observatory.circle_plots.FanPlotter method*), 201
plot() (*allensdk.brain_observatory.circle_plots.TrackPlotter method*), 202
plot_all() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 231
plot_bit() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 231
plot_bits() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 231
plot_blocks() (*in module allensdk.brain_observatory.ecephys.stimulus_table.visualization_utilities*), 177
plot_cell_correlation() (*in module allensdk.brain_observatory.observatory_plots*), 211
plot_cell_figures() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
plot_cell_figures() (*in module allensdk.internal.ephys.plot_qc_figures3*), 316
plot_cell_receptive_field() (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise method*), 208
plot_chi_square_summary() (*in module allensdk.brain_observatory.receptive_field_analysis.visualization*), 194
plot_combined_speed() (*in module allensdk.brain_observatory.observatory_plots*), 211
plot_condition_histogram() (*in module allensdk.brain_observatory.observatory_plots*), 211
plot_direction_selectivity() (*allensdk.brain_observatory.drifting_gratings.DriftingGratings method*), 207
plot_drifting_grating_traces() (*in module allensdk.brain_observatory.brain_observatory_plotting*), 200
plot_ellipses() (*in module allensdk.brain_observatory.receptive_field_analysis.visualization*), 194
plot_fi_curve_figures() (*in module allensdk.internal.ephys.plot_qc_figures*), 313
plot_fi_curve_figures() (*in module allensdk.internal.ephys.plot_qc_figures3*), 316
plot_fdr_mask() (*in module allensdk.brain_observatory.receptive_field_analysis.visualization*), 194
plot_gaussian_fit() (*in module allensdk.brain_observatory.receptive_field_analysis.visualization*), 195
plot_hero_figures() (*in module allensdk.internal.ephys.plot_qc_figures*), 314
plot_hero_figures() (*in module allensdk.internal.ephys.plot_qc_figures3*), 316
plot_images() (*in module allensdk.internal.ephys.plot_qc_figures*), 314
plot_images() (*in module allensdk.internal.ephys.plot_qc_figures3*), 316
plot_instantaneous_threshold_thumbnail() (*in module allensdk.internal.ephys.plot_qc_figures*), 314
plot_instantaneous_threshold_thumbnail() (*in module allensdk.internal.ephys.plot_qc_figures3*), 316
plot_line() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 231
plot_lines() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 231
plot_long_square_summary() (*in module allensdk.internal.ephys.plot_qc_figures*), 314

plot_long_square_summary() (*in module allensdk.internal.ephys.plot_qc_figures3*), 316
plot_lsn_traces() (*in module allensdk.brain_observatory.brain_observatory_plotting*), 200
plot_mask() (*in module allensdk.brain_observatory.receptive_field_analysis.visualization*), 195
plot_mask_outline() (*in module allensdk.brain_observatory.observatory_plots*), 211
plot_masks() (*in module allensdk.internal.brain_observatory.demix_report*), 296
plot_mean_waveforms() (*in module allensdk.brain_observatory.ecephys.visualization*), 184
plot_msr_summary() (*in module allensdk.brain_observatory.receptive_field_analysis.visualization*), 195
plot_negative_baselines() (*in module allensdk.brain_observatory.demixer*), 203
plot_negative_baselines() (*in module allensdk.internal.brain_observatory.demixer*), 296
plot_negative_transients() (*in module allensdk.brain_observatory.demixer*), 203
plot_negative_transients() (*in module allensdk.internal.brain_observatory.demixer*), 296
plot_ns_traces() (*in module allensdk.brain_observatory.brain_observatory_plotting*), 200
plot_onetrace() (*in module allensdk.brain_observatory.dff*), 206
plot_orientation_selectivity() (*allensdk.brain_observatory.drifting_gratings.DriftingGratings method*), 207
plot_orientation_selectivity() (*allensdk.brain_observatory.static_gratings.StaticGratings method*), 222
plot_overlap_masks_lengthOne() (*in module allensdk.brain_observatory.demixer*), 203
plot_overlap_masks_lengthOne() (*in module allensdk.internal.brain_observatory.demixer*), 297
plot_p_values() (*in module allensdk.brain_observatory.receptive_field_analysis.visualization*), 195
plot_population_receptive_field() (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise method*), 208
plot_preferred_direction() (*allensdk.brain_observatory.drifting_gratings.DriftingGratings method*), 207
plot_preferred_orientation() (*allensdk.brain_observatory.static_gratings.StaticGratings method*), 222
plot_preferred_spatial_frequency() (*allensdk.brain_observatory.static_gratings.StaticGratings method*), 222
plot_preferred_temporal_frequency() (*allensdk.brain_observatory.drifting_gratings.DriftingGratings method*), 207
plot_pupil_location() (*in module allensdk.brain_observatory.observatory_plots*), 211
plot_radial_histogram() (*in module allensdk.brain_observatory.observatory_plots*), 211
plot_ramp_figures() (*in module allensdk.internal.ephys.plot_qc_figures*), 314
plot_ramp_figures() (*in module allensdk.internal.ephys.plot_qc_figures3*), 317
plot_receptive_field() (*in module allensdk.brain_observatory.observatory_plots*), 211
plot_receptive_field_analysis_data() (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise method*), 208
plot_receptive_field_data() (*in module allensdk.brain_observatory.receptive_field_analysis.visualization*), 195
plot_representational_similarity() (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis method*), 223
plot_representational_similarity() (*in module allensdk.brain_observatory.observatory_plots*), 211
plot_rheo_figures() (*in module allensdk.internal.ephys.plot_qc_figures*), 314
plot_rheo_figures() (*in module allensdk.internal.ephys.plot_qc_figures3*), 317
plot_rts_blur_summary() (*in module allensdk.brain_observatory.receptive_field_analysis.visualization*), 195
plot_rts_summary() (*in module allensdk.brain_observatory.receptive_field_analysis.visualization*), 195
plot_running_a() (*in module allensdk.brain_observatory.brain_observatory_plotting*), 200
plot_running_speed() (*in module allensdk.brain_observatory.visualization*), 196

plot_running_speed_histogram() (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis method), 223
plot_sag_figures() (in module allensdk.internal.ephys.plot_qc_figures), 314
plot_sag_figures() (in module allensdk.internal.ephys.plot_qc_figures3), 317
plot_selectivity_cumulative_histogram() (in module allensdk.brain_observatory.observatory_plots), 212
plot_sg_traces() (in module allensdk.brain_observatory.brain_observatory_plotting), 200
plot_short_square_figures() (in module allensdk.internal.ephys.plot_qc_figures), 314
plot_short_square_figures() (in module allensdk.internal.ephys.plot_qc_figures3), 317
plot_single_ap_values() (in module allensdk.internal.ephys.plot_qc_figures), 314
plot_single_ap_values() (in module allensdk.internal.ephys.plot_qc_figures3), 317
plot_speed() (in module allensdk.brain_observatory.observatory_plots), 212
plot_speed_tuning() (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis method), 223
plot_spike_counts() (in module allensdk.brain_observatory.ecephys.visualization), 184
plot_subthreshold_long_square_figures() (in module allensdk.internal.ephys.plot_qc_figures), 315
plot_subthreshold_long_square_figures() (in module allensdk.internal.ephys.plot_qc_figures3), 317
plot_sweep_figures() (in module allensdk.internal.ephys.plot_qc_figures), 315
plot_sweep_figures() (in module allensdk.internal.ephys.plot_qc_figures3), 317
plot_sweep_set_summary() (in module allensdk.internal.ephys.plot_qc_figures), 315
plot_sweep_set_summary() (in module allensdk.internal.ephys.plot_qc_figures3), 317
plot_sweep_value_figures() (in module allensdk.internal.ephys.plot_qc_figures), 315
plot_sweep_value_figures() (in module allensdk.internal.ephys.plot_qc_figures3), 318
plot_time_to_peak() (allensdk.brain_observatory.natural_scenes.NaturalScenes method), 210
plot_time_to_peak() (allensdk.brain_observatory.static_gratings.StaticGratings method), 222
plot_time_to_peak() (in module allensdk.brain_observatory.observatory_plots), 212
plot_traces() (in module allensdk.brain_observatory.demixer), 204
plot_traces() (in module allensdk.internal.brain_observatory.demixer), 297
plot_transients() (in module allensdk.brain_observatory.demixer), 204
plot_transients() (in module allensdk.internal.brain_observatory.demixer), 297
plotLineRegress1() (in module allensdk.internal.model.glif.plotting), 329
plotLineRegress1() (in module allensdk.internal.model.glif.spike_cutting), 330
plotLineRegressRed() (in module allensdk.internal.model.glif.plotting), 329
plotLineRegressRed() (in module allensdk.internal.model.glif.spike_cutting), 330
plotSpikes() (in module allensdk.internal.model.glif.plotting), 329
polar_line_circles() (in module allensdk.brain_observatory.circle_plots), 202
polar_linspace() (in module allensdk.brain_observatory.circle_plots), 202
polar_to_xy() (in module allensdk.brain_observatory.circle_plots), 202
PolarPlotter (class in allensdk.brain_observatory.circle_plots), 201
PolygonSubImage (class in allensdk.mouse_connectivity.grid.subimage.base_subimage), 374
populate_stimulus_table() (allensdk.brain_observatory.drifting_gratings.DriftingGratings method), 207
populate_stimulus_table() (allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoi method), 208
populate_stimulus_table() (allensdk.brain_observatory.natural_movie.NaturalMovie method), 209
populate_stimulus_table() (allensdk.brain_observatory.natural_scenes.NaturalScenes method), 210
populate_stimulus_table() (allensdk.brain_observatory.static_gratings.StaticGratings

method), 222

populate_stimulus_table() (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis method*), 223

population_correlation_scatter() (*in module allensdk.brain_observatory.observatory_plots*), 212

port (*allensdk.core.authentication.DbCredentials attribute*), 241

POSITIVE (*allensdk.core.cell_types_cache.ReporterStatus attribute*), 249

post_process_cr() (*in module allensdk.internal.brain_observatory.itracker_utils*), 302

post_process_pupil() (*in module allensdk.internal.brain_observatory.itracker_utils*), 302

PostgresQueryMixin (*class in allensdk.internal.api*), 293

postprocess() (*allensdk.brain_observatory.behavior.behavior_project_cache.tables.project_table.ProjectTable method*), 109

postprocess_additional() (*allensdk.brain_observatory.behavior.behavior_project_cache.tables.experiments_table.ExperimentsTable method*), 108

postprocess_additional() (*allensdk.brain_observatory.behavior.behavior_project_cache.tables.ophys_sessions_table.BehaviorOphysSessionsTable method*), 109

postprocess_additional() (*allensdk.brain_observatory.behavior.behavior_project_cache.tables.project_table.ProjectTable method*), 109

postprocess_base() (*allensdk.brain_observatory.behavior.behavior_project_cache.tables.project_table.ProjectTable method*), 109

postprocess_unionizes() (*allensdk.internal.mouse_connectivity.interval_unionize.interval_unionizer.IntervalUnionizer method*), 344

postprocess_unionizes() (*allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionizer.TissuecyteUnionizer method*), 346

pre_blank_sec (*allensdk.brain_observatory.ecephys.file_io.stim_file.CamStimOnePickleStimFile attribute*), 173

prepare_nwb_output() (*in module allensdk.model.biophysical.runner*), 361

PREVIEW (*allensdk.api.queries.rma_api.RmaApi attribute*), 90

print_node() (*allensdk.core.swc.Compartment method*), 271

print_out() (*allensdk.ephys.feature_extractor.EphysFeatures method*), 288

print_summary() (*in module allensdk.brain_observatory.receptive_field_analysis.receptive_field*), 193

ProbeSynchronizer (*class in allensdk.brain_observatory.ecephys.align_timestamps.probe_synch*), 169

process() (*allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor method*), 276

process_eye_tracking_data() (*in module allensdk.brain_observatory.behavior.eye_tracking_processing*), 141

process_inputs() (*in module allensdk.internal.model.biophysical.passive_fitting.neuron_passive_*), 319

process_instance() (*allensdk.ephys.feature_extractor.EphysFeatureExtractor method*), 288

process_new_spike_feature() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 277

process_new_sweep_feature() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 277

process_segmentation() (*allensdk.mouse_connectivity.grid.subimage.base_subimage.Segmen method*), 375

process_segmentation() (*allensdk.mouse_connectivity.grid.subimage.classic_subimage.Class method*), 376

process_segmentation() (*allensdk.mouse_connectivity.grid.subimage.count_subimage.Count method*), 377

process_spikes() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 277

process_spikes() (*allensdk.ephys.ephys_extractor.EphysSweepSetFeatureExtractor method*), 279

PRODUCT_IDS (*allensdk.api.queries.mouse_connectivity_api.MouseConn attribute*), 83

project_code (*allensdk.brain_observatory.behavior.metadata.behavior_ attribute*), 112

project_name (*allensdk.api.cloud_cache.manifest.Manifest attribute*), 67

project_to_plane() (*in module allensdk.internal.brain_observatory.eye_calibration*), 300

PROJECTION_DENSITY (*allensdk.api.queries.grid_data_api.GridDataApi attribute*), 77

projection_density (*allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_u attribute*), 345

PROJECTION_DENSITY_KEY (*allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute*), 253

PROJECTION_ENERGY (allensdk.api.queries.grid_data_api.GridDataApi attribute), 77
projection_energy (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record.TissuecyteBaseUnionize attribute), 345
projection_intensity (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record.TissuecyteBaseUnionize attribute), 345
ProjectTable (class in allensdk.brain_observatory.behavior.behavior_project_cache.tables.project_table), 109
propagate() (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record.TissuecyteBaseUnionize method), 345
propagate() (allensdk.internal.mouse_connectivity.interval_unionize.unionize_record.Unionize method), 346
propagate_record() (allensdk.internal.mouse_connectivity.interval_unionize.interval_unionizer.IntervalUnionizer class method), 344
propagate_record() (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionizer.TissuecyteUnionizer class method), 346
propagate_to_bilateral() (allensdk.internal.mouse_connectivity.interval_unionize.interval_unionizer.IntervalUnionizer class method), 344
propagate_unionizes() (allensdk.internal.mouse_connectivity.interval_unionize.interval_unionizer.IntervalUnionizer class method), 344
provide() (allensdk.core.authentication.CredentialProvider method), 241
provide() (allensdk.core.authentication.EnvCredentialProvider method), 242
psycopg2_select() (in module allensdk.internal.api), 294
pupil_position_in_mouse_eye_coordinates() (allensdk.internal.brain_observatory.eye_calibration.EyeCalibration method), 298
pupil_position_on_monitor_in_cm() (allensdk.internal.brain_observatory.eye_calibration.EyeCalibration method), 298
pupil_position_on_monitor_in_degrees() (allensdk.internal.brain_observatory.eye_calibration.EyeCalibration method), 299
push_summary() (allensdk.ephys.feature_extractor.EphysFeatureExtractor method), 288
pval (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute), 224
pvalue_to_NLL() (in module allensdk.brain_observatory.receptive_field_analysis.chisquarerf), 190
pvalue_to_NLL() (in module allensdk.brain_observatory.receptive_field_analysis.visualization), 195
PycfgDescriptionParser (class in allensdk.config.model.formats.pycfg_description_parser), 236

Q

query() (in module allensdk.internal.core.lims_utilities), 310
quote_string() (allensdk.api.queries.rma_api.RmaApi method), 93

R

radial_arcs() (in module allensdk.brain_observatory.circle_plots), 202
radial_circles() (in module allensdk.brain_observatory.circle_plots), 202
RaisingSchema (class in allensdk.brain_observatory.argschema_utilities), 198
RaisingSchema (class in allensdk.brain_observatory.behavior.schemas), 156
RaisingSchema.Meta (class in allensdk.brain_observatory.argschema_utilities), 199
RaisingSchema.Meta (class in allensdk.brain_observatory.behavior.schemas), 156
ramps_features() (allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor method), 276
randomize_parameter_values() (allensdk.internal.model.glif.glif_optimizer.GlifOptimizer method), 326
rank_structures() (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache method), 256
ransac_fit() (allensdk.internal.brain_observatory.fit_ellipse.FitEllipse method), 300
raster_plot() (in module allensdk.brain_observatory.ecephys.visualization), 183
rasterize_polygons() (in module allensdk.mouse_connectivity.grid.utilities.image_utilities), 379
ratio_and_pyramid() (in module allensdk.mouse_connectivity.grid.writers), 380
read() (allensdk.config.model.description_parser.DescriptionParser method), 237
read() (allensdk.config.model.formats.hdf5_util.Hdf5Util method), 235
read() (allensdk.config.model.formats.json_description_parser.JsonDesc... method), 235

read() (*allensdk.config.model.formats.pycfg_description_parser.PycfgDescriptionParser method*), 236
read() (*in module allensdk.core.json_utilities*), 251
read() (*in module allensdk.internal.mouse_connectivity.interval_unionize.data_utilities*), 343
read_cell_index_receptive_field_analysis() (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise static method*), 208
read_data() (*allensdk.api.api.Api method*), 104
read_file() (*allensdk.core.json_utilities.JsonComments class method*), 251
read_h5_group() (*in module allensdk.brain_observatory.receptive_field_analysis.receptive_field*), 193
read_h5_group() (*in module allensdk.brain_observatory.receptive_field_analysis.tools*), 193
read_intensity_image() (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 379
read_json() (*allensdk.api.queries.biophysical_api.BiophysicalApi method*), 70
read_json() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 289
read_json() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 292
read_json_string() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 289
read_json_string() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 292
read_lims_file() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 289
read_lims_file() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 292
read_lims_message() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 290
read_lims_message() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 292
read_marker_file() (*in module allensdk.core.swc*), 275
read_marker_file() (*in module allensdk.internal.core.swc*), 312
read_model_description() (*allensdk.model.biophys_sim.config.Config method*), 360
read_ndarray_with_sitk() (*in module allensdk.core.sitk_utilities*), 268
read_neuron_fit_stdout() (*in module allensdk.internal.model.biophysical.passive_fitting.neuron_utils*), 319
read_obj() (*in module allensdk.core.obj_utilities*), 259
read_receptive_field_from_h5() (*in module allensdk.brain_observatory.receptive_field_analysis.receptive_field*), 193
read_segmentation_image() (*allensdk.mouse_connectivity.grid.subimage.base_subimage.Segmen method*), 375
read_segmentation_image() (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 379
read_stimulus() (*allensdk.model.biophysical.utils.Utils method*), 363
read_stimulus_name_from_path() (*in module allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spik*), 180
read_strided() (*in module allensdk.brain_observatory.ophys_session_decomposition*), 304
read_string() (*allensdk.config.model.description_parser.DescriptionParser method*), 238
read_string() (*allensdk.config.model.formats.json_description_parser.JsonDescrip method*), 235
read_string() (*allensdk.config.model.formats.pycfg_description_parser.PycfgDesc method*), 236
read_string() (*allensdk.core.json_utilities.JsonComments class method*), 251
read_swc() (*in module allensdk.core.swc*), 275
read_swc() (*in module allensdk.internal.core.swc*), 312
read_url() (*in module allensdk.core.json_utilities*), 251
read_url_get() (*in module allensdk.core.json_utilities*), 251
read_url_post() (*in module allensdk.core.json_utilities*), 251
readOutput() (*allensdk.internal.model.biophysical.passive_fitting.outpu method*), 319
receptive_field (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoi attribute*), 208
RECONSTRUCTION_KEY (*allensdk.core.cell_types_cache.CellTypesCache attribute*), 247

record_cb() (allensdk.internal.mouse_connectivity.interval_unionize.interval_unionizer.IntervalUnionizer class method), 344
record_cb() (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionizer.TissuecyteUnionizer class method), 346
record_values() (allensdk.internal.model.biophysical.deap_utils.Utils method), 320
record_values() (allensdk.model.biophysical.utils.Utils method), 363
REFERENCE_SPACE_VERSION_KEY (allensdk.core.reference_space_cache.ReferenceSpaceCache attribute), 263
ReferenceSpace (class in allensdk.core.reference_space), 260
ReferenceSpaceApi (class in allensdk.api.queries.reference_space_api), 87
ReferenceSpaceCache (class in allensdk.core.reference_space_cache), 263
refine_threshold_indexes() (in module allensdk.ephys.ephys_features), 287
relative_path_from_url() (in module allensdk.api.cloud_cache.utils), 67
remove_comments() (allensdk.core.json_utilities.JsonComments class method), 251
remove_keys() (allensdk.api.warehouse_cache.cache.Cache static method), 99
remove_lfp_noise() (in module allensdk.brain_observatory.ecephys.lfp_subsampling.subsampling), 173
remove_lfp_offset() (in module allensdk.brain_observatory.ecephys.lfp_subsampling.subsampling), 173
remove_multiline_comments() (allensdk.core.json_utilities.JsonComments class method), 251
remove_unassigned() (allensdk.core.reference_space.ReferenceSpace method), 262
rename_columns() (allensdk.api.warehouse_cache.cache.Cache static method), 99
renames() (allensdk.core.structure_tree.StructureTree static method), 271
Report (class in allensdk.internal.model.biophysical.make_deap_fit_json), 321
reporter_line (allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata attribute), 111
ReporterStatus (class in al-
lensdk.internal.core.cell_types_cache), 249
required_intensities (allensdk.mouse_connectivity.grid.subimage.base_subimage.Intensit attribute), 374
required_intensities (allensdk.mouse_connectivity.grid.subimage.classic_subimage.Class attribute), 376
required_polys (allensdk.mouse_connectivity.grid.subimage.base_subimage.Polygon attribute), 374
required_polys (allensdk.mouse_connectivity.grid.subimage.cav_subimage.CavSubI attribute), 375
required_polys (allensdk.mouse_connectivity.grid.subimage.classic_subimage.Class attribute), 376
required_polys (allensdk.mouse_connectivity.grid.subimage.count_subimage.Count attribute), 377
required_segmentations (allensdk.mouse_connectivity.grid.subimage.base_subimage.Segmen attribute), 375
required_segmentations (allensdk.mouse_connectivity.grid.subimage.classic_subimage.Class attribute), 376
required_segmentations (allensdk.mouse_connectivity.grid.subimage.count_subimage.Count attribute), 377
resample_into_volume() (in module allensdk.mouse_connectivity.grid.utilities.image_utilities), 379
resample_volume() (allensdk.mouse_connectivity.grid.image_series_gridder.ImageSerie method), 381
resample_volume() (in module allensdk.mouse_connectivity.grid.utilities.image_utilities), 379
resave_swc() (in module allensdk.internal.morphology.validate_swc), 342
reset() (allensdk.model.glif.glif_neuron.GlifNeuron method), 365
reset_AScurrent_none() (in module allensdk.model.glif.glif_neuron_methods), 370
reset_AScurrent_sum() (in module allensdk.model.glif.glif_neuron_methods), 370
reset_hex_pack() (in module allensdk.brain_observatory.circle_plots), 202
reset_long_squares_start() (in module allensdk.ephys.ephys_extractor), 280
reset_threshold_inf() (in module allensdk.model.glif.glif_neuron_methods),

370
reset_threshold_three_components() (*in module allensdk.model.glif.glif_neuron_methods*), 370
reset_voltage_v_before() (*in module allensdk.model.glif.glif_neuron_methods*), 371
reset_voltage_zero() (*in module allensdk.model.glif.glif_neuron_methods*), 371
reshape_response_array() (*allensdk.brain_observatory.drifting_gratings.DriftingGratings method*), 207
reshape_response_array() (*allensdk.brain_observatory.natural_scenes.NaturalScenes method*), 210
reshape_response_array() (*allensdk.brain_observatory.static_gratings.StaticGratings method*), 222
resolve_initial_image() (*in module allensdk.brain_observatory.behavior.trials_processing*), 164
resolve_paths() (*allensdk.config.manifest.Manifest method*), 239
response (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 224
response_bias() (*in module allensdk.brain_observatory.behavior.session_metrics*), 158
retinotopy_metric() (*in module allensdk.internal.brain_observatory.annotated_region_metrics*), 295
retrieve_file_from_storage() (*allensdk.internal.api.api_prerelease.ApiPrerelease method*), 293
retrieve_file_over_http() (*allensdk.api.api.Api method*), 104
retrieve_parsed_json_over_http() (*allensdk.api.api.Api method*), 105
retrieve_xml_over_http() (*allensdk.api.api.Api method*), 105
return_mask_cb() (*allensdk.core.reference_space.ReferenceSpace static method*), 262
reward_rate() (*in module allensdk.brain_observatory.behavior.trial_masks*), 158
rings_in_hex_pack() (*in module allensdk.brain_observatory.circle_plots*), 202
rma_templates (*allensdk.api.queries.biophysical_api.BiophysicalApi attribute*), 70
rma_templates (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi attribute*), 73
rma_templates (*allensdk.api.queries.glif_api.GlifApi attribute*), 77
rma_templates (*allensdk.api.queries.image_download_api.ImageDownloadApi attribute*), 81
rma_templates (*allensdk.api.queries.ontologies_api.OntologiesApi attribute*), 87
rma_templates (*allensdk.internal.api.queries.biophysical_module_api.BiophysicalM attribute*), 289
RmaApi (*class in allensdk.api.queries.rma_api*), 89
RmaPager (*class in allensdk.api.queries.rma_pager*), 94
RmaTemplate (*class in allensdk.api.queries.rma_template*), 94
robust_std() (*in module allensdk.brain_observatory.dff*), 206
roi_id (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 224
RoiMask (*class in allensdk.brain_observatory.roi_masks*), 214
rolling_window() (*in module allensdk.brain_observatory.demixer*), 204
rolling_window() (*in module allensdk.internal.brain_observatory.demixer*), 297
root (*allensdk.core.swc.Morphology attribute*), 274
rotate() (*allensdk.internal.mouse_connectivity.projection_thumbnail.vo method*), 349
rotate() (*in module allensdk.brain_observatory.stimulus_info*), 227
rotate_and_extract() (*allensdk.internal.mouse_connectivity.projection_thumbnail.volume_ method*), 349
rotate_ray() (*in module allensdk.internal.brain_observatory.itracker_utils*), 303
rotate_vector() (*in module allensdk.internal.brain_observatory.fit_ellipse*), 300
row_from_cell_id() (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis method*), 224
run() (*allensdk.internal.model.glif.glif_experiment.GlifExperiment method*), 324
run() (*allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher.S method*), 349
run() (*allensdk.model.glif.glif_neuron.GlifNeuron method*), 366
run() (*in module allensdk.internal.mouse_connectivity.interval_unionize.run_tissuec*

345
run() (*in module allensdk.internal.mouse_connectivity.projection_thumbnail.generate_projection_strip*), 347
run() (*in module allensdk.model.biophysical.runner*), 361
run_base_model() (*allensdk.internal.model.glif.glif_experiment.GlifExperiment method*), 325
run_many() (*allensdk.internal.model.glif.glif_optimizer.GlifOptimizer method*), 326
run_module() (*in module allensdk.internal.core.lims_pipeline_module*), 309
run_module() (*in module allensdk.model.biophys_sim.bps_command*), 360
run_once() (*allensdk.internal.model.glif.glif_optimizer.GlifOptimizer method*), 326
run_once_bound() (*allensdk.internal.model.glif.glif_optimizer.GlifOptimizer method*), 326
run_ophys_time_sync() (*in module allensdk.internal.pipeline_modules.run_ophys_time_sync*), 358
run_passive_fit() (*in module allensdk.internal.model.biophysical.run_passive_fit*), 322
run_postprocessing() (*in module allensdk.brain_observatory.receptive_field_analysis.postprocessing*), 192
run_session_analysis() (*in module allensdk.brain_observatory.session_analysis*), 219
run_subimage() (*in module allensdk.mouse_connectivity.grid.subimage*), 377
run_subimage() (*in module allensdk.mouse_connectivity.grid.subimage.base_subimage*), 375
run_sync() (*in module allensdk.model.biophysical.runner*), 361
run_until_biological_spike() (*allensdk.internal.model.glif.glif_optimizer_neuron.GlifOptimizerNeuron method*), 327
run_with_biological_spikes() (*allensdk.internal.model.glif.glif_optimizer_neuron.GlifOptimizerNeuron method*), 327
RunningSpeed (*class in allensdk.brain_observatory.running_speed*), 216
RunOptimize (*class in allensdk.internal.model.biophysical.run_optimize*), 321
RunSimulate (*class in allensdk.model.biophysical.run_simulate*), 361
RunSimulateLims (*class in allensdk.internal.model.biophysical.run_simulate_lims*), 322

S

safe_factory() (*allensdk.internal.mouse_connectivity.projection_thumbnail.volume_ class method*), 349
safe_make_parent_dirs() (*allensdk.config.manifest.Manifest class method*), 239
safe_mkdir() (*allensdk.config.manifest.Manifest class method*), 240
safe_system_path() (*in module allensdk.internal.core.lims_utilities*), 310
SAG_TARGET (*allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor attribute*), 276
sameconv() (*in module allensdk.internal.model.GLM*), 334
sample_freq (*allensdk.brain_observatory.sync_dataset.Dataset attribute*), 231
sample_frequency (*allensdk.brain_observatory.ecephys.file_io.ecephys_sync_dataset.E attribute*), 172
sampling_rate_scale (*allensdk.brain_observatory.ecephys.align_timestamps.probe_synch attribute*), 169
save() (*allensdk.core.swc.Morphology method*), 274
save() (*allensdk.internal.morphology.morphology.Morphology method*), 338
save_analysis_arrays() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method*), 246
save_analysis_dataframes() (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN method*), 246
save_cell_index_receptive_field_analysis() (*allensdk.brain_observatory.locally_sparse_noise.LocallySparseN static method*), 208
save_ephys_data() (*allensdk.api.queries.cell_types_api.CellTypesApi method*), 75
save_figure() (*in module allensdk.internal.ephys.plot_qc_figures*), 315
save_figure() (*in module allensdk.internal.ephys.plot_qc_figures3*), 318
save_nwb() (*in module allensdk.model.biophysical.runner*), 362
save_ophys_experiment_analysis_data() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryAp*

method), 73
save_ophys_experiment_data() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method*), 73
save_ophys_experiment_event_data() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method*), 73
save_ophys_experiment_eye_gaze_data() (*allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method*), 73
save_qc_figures() (*in module allensdk.internal.ephys.core_feature_extract*), 313
save_reconstruction() (*allensdk.api.queries.cell_types_api.CellTypesApi method*), 75
save_reconstruction_markers() (*allensdk.api.queries.cell_types_api.CellTypesApi method*), 75
save_session_a() (*allensdk.brain_observatory.session_analysis.SessionAnalysis method*), 217
save_session_b() (*allensdk.brain_observatory.session_analysis.SessionAnalysis method*), 217
save_session_c() (*allensdk.brain_observatory.session_analysis.SessionAnalysis method*), 218
save_session_c2() (*allensdk.brain_observatory.session_analysis.SessionAnalysis method*), 218
save_voltage() (*allensdk.core.dat_utilities.DatUtilities class method*), 249
schema() (*allensdk.api.queries.connected_services.ConnectedServices class method*), 76
score_feature_set() (*allensdk.ephys.feature_extractor.EphysFeatureExtractor method*), 288
search() (*allensdk.brain_observatory.stimulus_info.BinaryIntervalSearchTree method*), 224
search() (*allensdk.brain_observatory.stimulus_info.StimulusSearch method*), 225
section_image_query() (*allensdk.api.queries.image_download_api.ImageDownloadApi method*), 81
SegmentationSubImage (*class in allensdk.mouse_connectivity.grid.subimage.base_subimage*), 374
select() (*allensdk.internal.api.PostgresQueryMixin method*), 293
select() (*in module allensdk.internal.core.lims_utilities*), 310
select_channels() (*in module allensdk.brain_observatory.ecephys.lfp_subsampling.subsampling*), 174
select_one() (*allensdk.internal.api.PostgresQueryMixin method*), 293
serialize() (*allensdk.brain_observatory.behavior.image_api.ImageApi static method*), 142
SERVICE (*allensdk.api.queries.rma_api.RmaApi attribute*), 90
service_query() (*allensdk.api.queries.rma_api.RmaApi method*), 93
service_stage() (*allensdk.api.queries.rma_api.RmaApi method*), 94
session_a() (*allensdk.brain_observatory.session_analysis.SessionAnaly method*), 218
session_b() (*allensdk.brain_observatory.session_analysis.SessionAnaly method*), 218
session_c() (*allensdk.brain_observatory.session_analysis.SessionAnaly method*), 219
session_c2() (*allensdk.brain_observatory.session_analysis.SessionAna method*), 219
session_na (*allensdk.brain_observatory.ecephys.ecephys_session_api.e attribute*), 171
session_type (*allensdk.brain_observatory.behavior.metadata.behavior attribute*), 111
SessionAnalysis (*class in allensdk.brain_observatory.session_analysis*), 217
sessions_are_equal() (*in module allensdk.brain_observatory.session_api_utils*), 220
sessions_with_stimulus() (*in module allensdk.brain_observatory.stimulus_info*), 227
set_actual_parameters() (*allensdk.internal.model.biophysical.deap_utils.Utils method*), 320
set_api_urls() (*allensdk.api.api.Api method*), 105
set_apical_color() (*allensdk.internal.morphology.morphvis.MorphologyColors method*), 339
set_axon_color() (*allensdk.internal.morphology.morphvis.MorphologyColors method*), 339
set_basal_color() (*allensdk.internal.morphology.morphvis.MorphologyColors method*), 339
set_coarse_grid_parameters() (*allensdk.mouse_connectivity.grid.image_series_gridder.ImageSerie method*), 381
set_credential_provider() (*in module allensdk.core.authentication*), 242
set_default_working_directory() (*allensdk.api.api.Api method*), 105

set_dims() (*allensdk.brain_observatory.circle_plots.CoronaPlotter method*), 201
set_dims() (*allensdk.brain_observatory.circle_plots.FanPlotter method*), 201
set_F() (*allensdk.brain_observatory.r_neuropil.NeuropilSubtract method*), 212
set_iclamp_params() (*allensdk.internal.model.biophysical.deap_utils.Utils method*), 320
set_image_spacing() (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 379
set_max_voxel() (*allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record.TissuecyteBaseUnionize method*), 345
set_neuron_parameters() (*allensdk.internal.model.glif.glif_experiment.GlifExperiment method*), 325
set_normalized_parameters() (*allensdk.internal.model.biophysical.deap_utils.Utils method*), 320
set_params() (*allensdk.brain_observatory.session_api_utils.ParamsMixin method*), 220
set_sitk_image_information() (*in module allensdk.core.sitk_utilities*), 268
set_soma_color() (*allensdk.internal.morphology.morphvis.MorphologyColors method*), 339
set_spatial_unit() (*allensdk.brain_observatory.stimulus_info.Monitor method*), 225
set_spike_times() (*allensdk.core.nwb_data_set.NwbDataSet method*), 258
set_stimulus_amplitude_calculator() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 278
set_sweep() (*allensdk.core.nwb_data_set.NwbDataSet method*), 258
set_version() (*allensdk.config.manifest_builder.ManifestBuilder method*), 240
set_workflow_state() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 290
setup_iclamp() (*allensdk.model.biophysical.utils.Utils method*), 363
setup_images() (*allensdk.mouse_connectivity.grid.subimage.base_su*... *method*), 374
setup_images() (*allensdk.mouse_connectivity.grid.subimage.base_subimage.PolygonSubImage method*), 374
setup_images() (*allensdk.mouse_connectivity.grid.subimage.base_subimage.Segmen*... *method*), 375
setup_images() (*allensdk.mouse_connectivity.grid.subimage.base_subimage.SubIma*... *method*), 375
setup_interval_map() (*allensdk.internal.mouse_connectivity.interval_unionize.interval_un*... *method*), 344
setup_model() (*allensdk.internal.model.biophysical.make_deap_fit_json.Report method*), 321
setup_subimages() (*allensdk.mouse_connectivity.grid.image_series_gridder.ImageSerie*... *method*), 381
sex (*allensdk.brain_observatory.behavior.metadata.behavior_metadata.Be*... *attribute*), 111
sfvals (*allensdk.brain_observatory.static_gratings.StaticGratings attribute*), 222
short_squares_features() (*allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor method*), 276
short_string() (*allensdk.internal.morphology.node.Node method*), 342
show_angle_labels() (*allensdk.brain_observatory.circle_plots.FanPlotter method*), 201
show_arrow() (*allensdk.brain_observatory.circle_plots.CoronaPlotter method*), 201
show_arrow() (*allensdk.brain_observatory.circle_plots.TrackPlotter method*), 202
show_axes() (*allensdk.brain_observatory.circle_plots.FanPlotter method*), 201
show_circle() (*allensdk.brain_observatory.circle_plots.CoronaPlotter method*), 201
show_group_labels() (*allensdk.brain_observatory.circle_plots.FanPlotter method*), 201
show_image() (*allensdk.brain_observatory.stimulus_info.Monitor method*), 225
show_r_labels() (*allensdk.brain_observatory.circle_plots.FanPlotter method*), 201
simple_rotation() (*in module allensdk.internal.mouse_connectivity.projection_thumbnail.generat*...), 347
SimpleTree (*class in allensdk.core.simple_tree*), 265
SimpleTree (*class in allensdk.internal.core.simpletree*), 311
simplify_cells_api() (*allensdk.api.queries.cell_types_api.CellTypesApi method*), 75

simplify_experiment_containers() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 73
simplify_ophys_experiments() (allensdk.api.queries.brain_observatory_api.BrainObservatoryApi method), 73
simulate() (allensdk.model.biophysical.run_simulate.RunSimulate method), 361
simulate_neuron() (in module allensdk.model.glif.simulate_neuron), 372
simulate_sweep() (in module allensdk.model.glif.simulate_neuron), 372
simulate_sweep_from_file() (in module allensdk.model.glif.simulate_neuron), 372
sitk_get_center() (in module allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities), 349
sitk_get_diagonal_length() (in module allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities), 349
sitk_get_image_parameters() (in module allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities), 349
sitk_get_size_parity() (in module allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities), 349
sitk_np_convert() (in module allensdk.mouse_connectivity.grid.utilities.image_utilities), 379
sitk_paste_into_center() (in module allensdk.internal.mouse_connectivity.projection_thumbnail.volume_utilities), 349
sitk_safe_ln() (in module allensdk.internal.mouse_connectivity.projection_thumbnail.visualization_utilities), 348
size() (allensdk.internal.brain_observatory.mask_set.MaskSet method), 303
slice_arrays() (allensdk.internal.mouse_connectivity.interval_unionize.interval_unionizer.IntervalUnionizer method), 346
smooth() (in module allensdk.brain_observatory.receptive_field_analysis.utilities), 194
smooth_STA() (in module allensdk.brain_observatory.receptive_field_analysis.chisquarerf), 190
sobel_grad() (in module allensdk.internal.brain_observatory.itracker_utils), 303
SOMA (allensdk.core.swc.Morphology attribute), 272
soma (allensdk.core.swc.Morphology attribute), 274
SOMA (allensdk.internal.morphology.morphology.Morphology attribute), 335
soma_root() (allensdk.internal.morphology.morphology.Morphology method), 338
sort_data_arrays() (allensdk.internal.mouse_connectivity.interval_unionize.interval_unionizer.IntervalUnionizer method), 344
sort_trials() (allensdk.brain_observatory.locally_sparse_noise.LocallySparseNoise method), 208
spacing (allensdk.brain_observatory.behavior.image_api.Image attribute), 142
SPACING (allensdk.core.swc.Marker attribute), 271
SPACING (allensdk.internal.core.swc.Marker attribute), 312
sparsify() (allensdk.core.swc.Morphology method), 274
sparsify() (allensdk.internal.morphology.morphology.Morphology method), 338
spatial_frequency_to_pix_per_cycle() (allensdk.brain_observatory.stimulus_info.Monitor method), 225
spike_component_of_threshold_exact() (in module allensdk.model.glif.glif_neuron_methods), 371
spike_component_of_threshold_forward_euler() (in module allensdk.model.glif.glif_neuron_methods), 371
spike_feature() (allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method), 278
spike_feature_averages() (allensdk.ephys.ephys_extractor.EphysSweepSetFeatureExtractor method), 279
spike_feature_keys() (allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method), 278
SPIKE_TIMES (allensdk.core.nwb_data_set.NwbDataSet attribute), 257
spikes() (allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method), 278
spiral_trials() (in module allensdk.brain_observatory.circle_plots), 202
spiral_trials_polar() (in module allensdk.brain_observatory.circle_plots), 203
split_column() (in module allensdk.brain_observatory.ecephys.stimulus_table.ephys_pre_spikes), 180
standardize_movie_numbers() (in module allensdk.brain_observatory.ecephys.stimulus_table.naming_utilities), 181
start() (allensdk.internal.model.biophysical.passive_fitting.output_grabber.OutputGrabber method), 319
START_ROW (allensdk.api.queries.rma_api.RmaApi attribute), 90

start_specimen() (*allensdk.internal.model.biophysical.run_optimize.RunOptimize method*), 321
StaticGratings (*class in allensdk.brain_observatory.static_gratings*), 221
stats() (*allensdk.brain_observatory.sync_dataset.Dataset method*), 231
stim_table (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 224
stim_table_to_categories() (*in module allensdk.brain_observatory.chisquare_categorical*), 201
stim_timestamps (*allensdk.internal.brain_observatory.time_sync.OphysTimeAligner attribute*), 307
stimuli (*allensdk.brain_observatory.ecephys.file_io.stim_file.CamStimOnePickleStimFile attribute*), 173
stimuli_in_session() (*in module allensdk.brain_observatory.stimulus_info*), 227
stimulus_alignment (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncOutputs attribute*), 357
stimulus_amplitude() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 278
STIMULUS_CONTENT_TYPE (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader attribute*), 289
STIMULUS_CONTENT_TYPE (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader attribute*), 291
stimulus_delay (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncOutputs attribute*), 357
stimulus_delta (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncOutputs attribute*), 357
stimulus_file_entries() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 290
stimulus_file_entries() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 292
stimulus_frame_rate (*allensdk.brain_observatory.behavior.metadata.behavior_metadata.BehaviorMetadata attribute*), 111
stimulus_path() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 290
stimulus_path() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 292
stimulus_search (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN attribute*), 246
STIMULUS_TABLE_TYPES (*allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN attribute*), 242
stimulus_times (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSync attribute*), 357
StimulusAnalysis (*class in allensdk.brain_observatory.stimulus_analysis*), 222
StimulusImage (*class in allensdk.brain_observatory.behavior.stimulus_processing.stimulus_*), 127
StimulusImageFactory (*class in allensdk.brain_observatory.behavior.stimulus_processing.stimulus_*), 127
StimulusSearch (*class in allensdk.brain_observatory.stimulus_info*), 225
StimulusTemplate (*class in allensdk.brain_observatory.behavior.stimulus_processing.stimulus_*), 127
StimulusTemplateFactory (*class in allensdk.brain_observatory.behavior.stimulus_processing.stimulus_*), 127
stitch() (*allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitc method*), 349
Stitcher (*class in allensdk.internal.mouse_connectivity.tissuecyte_stitching.stitcher*), 349
stop() (*allensdk.internal.model.biophysical.passive_fitting.output_grabb method*), 319
STORAGE_DIRECTORIES_PRERELEASE_KEY (*allensdk.internal.core.mouse_connectivity_cache_prerelease.Mouse attribute*), 311
stream_file_over_http() (*in module allensdk.api.api*), 105
stream_zip_directory_over_http() (*in module allensdk.api.api*), 105
STRING (*allensdk.api.queries.connected_services.ConnectedServices attribute*), 76
strip_all_other_types() (*allensdk.core.swc.Morphology method*), 274
strip_all_other_types() (*allensdk.internal.morphology.morphology.Morphology method*), 338
strip_type() (*allensdk.core.swc.Morphology method*), 275
strip_type() (*allensdk.internal.morphology.morphology.Morphology method*), 338
structure_descends_from() (*allensdk.core.ontology.Ontology method*),

260
structure_descends_from() (allensdk.core.structure_tree.StructureTree method), 271
STRUCTURE_MASK_KEY (allensdk.core.reference_space_cache.ReferenceSpaceCache attribute), 263
STRUCTURE_MESH_KEY (allensdk.core.reference_space_cache.ReferenceSpaceCache attribute), 263
STRUCTURE_TREE_KEY (allensdk.core.reference_space_cache.ReferenceSpaceCache attribute), 263
STRUCTURE_UNIONIZES_KEY (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute), 253
STRUCTURES_KEY (allensdk.core.reference_space_cache.ReferenceSpaceCache attribute), 263
StructureTree (class in allensdk.core.structure_tree), 269
stumpify_axon() (allensdk.core.swc.Morphology method), 275
stumpify_axon() (allensdk.internal.morphology.morphology.Morphology method), 339
SubImage (class in allensdk.mouse_connectivity.grid.subimage.base_subimage), 375
SubjectMetadataSchema (class in allensdk.brain_observatory.behavior.schemas), 156
subsample_data() (in module allensdk.internal.model.data_access), 335
subsample_lfp() (in module allensdk.brain_observatory.ecephys.lfp_subsampling.subsampling), 174
subsample_timestamps() (in module allensdk.brain_observatory.ecephys.lfp_subsampling.subsampling), 175
SUBTHRESH_MAX_AMP (allensdk.ephys.ephys_extractor.EphysCellFeatureExtractor attribute), 276
sum_pixel_intensity (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record.TissuecyteBaseUnionize attribute), 345
sum_pixels (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record.TissuecyteBaseUnionize attribute), 345
sum_projection_pixel_intensity (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record.TissuecyteBaseUnionize attribute), 345
sum_projection_pixels (allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record.TissuecyteBaseUnionize attribute), 345
summarize() (allensdk.ephys.feature_extractor.EphysFeatureExtractor method), 288
SUMMARY_STRUCTURE_SET_ID (allensdk.core.mouse_connectivity_cache.MouseConnectivityCache attribute), 253
summer_over() (in module allensdk.brain_observatory.behavior.criteria), 137
SUPPORTED_PIPELINE_VERSION (allensdk.core.brain_observatory_nwb_data_set.BrainObservatoryN attribute), 242
SupportsStr (class in allensdk.core.typing), 275
SvgApi (class in allensdk.api.queries.svg_api), 95
SWC_FILE_TYPE (allensdk.api.queries.cell_types_api.CellTypesApi attribute), 74
sweep_entries() (allensdk.internal.api.queries.biophysical_module_reader.Biophysic method), 290
sweep_entries() (allensdk.internal.api.queries.optimize_config_reader.OptimizeConf method), 292
sweep_feature() (allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method), 278
sweep_feature_keys() (allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method), 278
sweep_features() (allensdk.ephys.ephys_extractor.EphysSweepSetFeatureExtractor method), 279
sweep_numbers() (allensdk.internal.api.queries.biophysical_module_reader.Biophysic method), 290
sweep_numbers() (allensdk.internal.api.queries.optimize_config_reader.OptimizeConf method), 292
sweep_numbers_by_type() (allensdk.internal.api.queries.biophysical_module_reader.Biophysic method), 290
sweep_response (allensdk.brain_observatory.natural_movie.NaturalMovie attribute), 209
sweep_response (allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute), 224
sweep_response (allensdk.brain_observatory.locally_sparse_noise.LocallyS attribute), 208
sweeplength (allensdk.brain_observatory.natural_movie.NaturalMovie attribute), 209
sweeplength (allensdk.brain_observatory.natural_scenes.NaturalScenes attribute), 210
sweeplength (allensdk.brain_observatory.static_gratings.StaticGratings attribute), 222

sweeps() (*allensdk.ephys.ephys_extractor.EphysSweepSetFeatureExtractor method*), 279
SynchronizationApi (*class in allensdk.api.queries.synchronization_api*), 95
synthesize_F() (*in module allensdk.brain_observatory.r_neuropil*), 213

T

table (*allensdk.brain_observatory.behavior.behavior_project_cache.tables.project_table.ProjectTable attribute*), 109
TABULAR (*allensdk.api.queries.rma_api.RmaApi attribute*), 90
targeted_structure (*allensdk.brain_observatory.behavior.metadata.behavior_ophys_metadata.BehaviorOphysMetadata attribute*), 112
tau_m (*allensdk.model.glif.glif_neuron.GlifNeuron attribute*), 366
temp_dir() (*in module allensdk.test_utilities.temp_dir*), 381
TEMPLATE_KEY (*allensdk.core.reference_space_cache.ReferenceSpaceCache attribute*), 263
template_projection() (*in module allensdk.internal.mouse_connectivity.projection_thumbnail.projection_functions*), 348
template_query() (*allensdk.api.queries.rma_template.RmaTemplate method*), 94
test() (*allensdk.brain_observatory.ecephys.ecephys_session_api.ecephys_session_api.EcephysSessionApi method*), 171
test_fit() (*in module allensdk.internal.brain_observatory.fit_ellipse*), 300
TestNode (*class in allensdk.internal.morphology.validate_swc*), 342
tfvals (*allensdk.brain_observatory.drifting_gratings.DriftingGratings attribute*), 207
Tile (*class in allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile*), 350
timestamps (*allensdk.brain_observatory.running_speed.RunningSpeed attribute*), 216
timestamps (*allensdk.brain_observatory.stimulus_analysis.StimulusAnalysis attribute*), 224
TimeSyncOutputs (*class in allensdk.internal.pipeline_modules.run_ophys_time_sync*), 357
TimeSyncWriter (*class in allensdk.internal.pipeline_modules.run_ophys_time_sync*), 357
TissuecyteBaseUnionize (*class in allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_unionize_record*), 345
TissuecyteProjectionUnionize (*class in allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_u*
346
TissuecyteProjectionUnionize (*class in allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_u*
346
TissuecyteUnionizer (*class in allensdk.internal.mouse_connectivity.interval_unionize.tissuecyte_u*
346
to_config_string() (*allensdk.config.app.application_config.ApplicationConfig method*), 234
to_dataframe() (*allensdk.brain_observatory.behavior.stimulus_processing.stimulus_ method*), 127
to_dict() (*allensdk.brain_observatory.behavior.metadata.behavior_met method*), 111
to_dict() (*allensdk.brain_observatory.behavior.metadata.behavior_oph method*), 112
to_dict() (*allensdk.internal.model.glif.glif_optimizer.GlifOptimizer method*), 326
to_dict() (*allensdk.internal.model.glif.glif_optimizer_neuron.GlifOptim method*), 328
to_dict() (*allensdk.internal.morphology.morphology.Morphology method*), 339
to_dict() (*allensdk.internal.morphology.node.Node method*), 342
to_dict() (*allensdk.model.glif.glif_neuron.GlifNeuron method*), 366
to_dict() (*allensdk.model.glif.glif_neuron_methods.GlifNeuronMethod method*), 367
to_filter_rhs() (*allensdk.api.queries.rma_template.RmaTemplate method*), 94
to_manifest() (*allensdk.internal.api.queries.biophysical_module_reader.Biophysic method*), 290
to_manifest() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConf method*), 292
total_voxel_counts() (*allensdk.core.reference_space.ReferenceSpace method*), 262
total_voxel_map (*allensdk.core.reference_space.ReferenceSpace attribute*), 262
TrackPlotter (*class in allensdk.brain_observatory.circle_plots*), 202
TrainingLabelClassifier (*class in allensdk.internal.brain_observatory.roi_filter_utils*), 305
TrainingMultiLabelClassifier (*class in allensdk.internal.brain_observatory.roi_filter_utils*), 305

transform (*allensdk.mouse_connectivity.grid.image_series_gridder.ImageSeriesGridder attribute*), 381
translate_image_and_fill() (*in module allensdk.brain_observatory.stimulus_info*), 227
traverse_h5_file() (*in module allensdk.core.h5_utilities*), 250
tree() (*allensdk.core.swc.Morphology method*), 275
tree() (*allensdk.internal.morphology.morphology.Morphology method*), 339
TreeSearchApi (*class in allensdk.api.queries.tree_search_api*), 97
trial_data_from_log() (*in module allensdk.brain_observatory.behavior.trials_processing*), 164
trial_number_limit() (*in module allensdk.brain_observatory.behavior.dprime*), 138
trial_types() (*in module allensdk.brain_observatory.behavior.trial_masks*), 158
trim() (*allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile method*), 351
trim_border_pulses() (*in module allensdk.brain_observatory.ecephys.stimulus_sync*), 186
trim_discontiguous_times() (*in module allensdk.brain_observatory.sync_utilities*), 196
trim_self() (*allensdk.internal.mouse_connectivity.tissuecyte_stitching.tile.Tile method*), 351
trimmed_stats() (*in module allensdk.brain_observatory.ecephys.stimulus_sync*), 186
TRUE (*allensdk.api.queries.rma_api.RmaApi attribute*), 90
tuple_filters() (*allensdk.api.queries.rma_api.RmaApi method*), 94
two_out_of_three_aint_bad() (*in module allensdk.brain_observatory.behavior.criteria*), 137
TYPE (*allensdk.internal.model.glif.glif_optimizer_neuron.GlifOptimizerNeuron attribute*), 326
TYPE (*allensdk.model.glif.glif_neuron.GlifNeuron attribute*), 365

U

union() (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303
union_size() (*allensdk.internal.brain_observatory.mask_set.MaskSet method*), 303
Unionize (*class in allensdk.internal.mouse_connectivity.interval_unionize.unionize_record*), 346
unit (*allensdk.brain_observatory.behavior.image_api.Image attribute*), 142
unknown (*allensdk.brain_observatory.argschema_utilities.RaisingSchema attribute*), 199
unknown (*allensdk.brain_observatory.behavior.schemas.RaisingSchema.M attribute*), 156
unpack() (*allensdk.config.model.description.Description method*), 237
unpack_change_log() (*in module allensdk.brain_observatory.behavior.stimulus_processing*), 131
unpack_manifest() (*allensdk.config.model.description.Description method*), 237
unpack_structure_set_ancestors() (*allensdk.api.queries.ontologies_api.OntologiesApi method*), 87
unpack_uint32() (*in module allensdk.brain_observatory.sync_dataset*), 231
update_data() (*allensdk.config.model.description.Description method*), 237
update_default_cell_hoc() (*allensdk.model.biophysical.utils.Utils method*), 363
update_output_sweep_features() (*in module allensdk.internal.ephys.core_feature_extract*), 313
update_well_known_file() (*allensdk.internal.api.queries.biophysical_module_reader.Biophysic method*), 290
update_well_known_file() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConf method*), 292
upsample_image_to_degrees() (*in module allensdk.brain_observatory.receptive_field_analysis.utilities*), 194
url (*allensdk.api.cloud_cache.file_attributes.CacheFileAttributes attribute*), 66
user (*allensdk.core.authentication.DbCredentials attribute*), 242
Utils (*class in allensdk.internal.model.biophysical.deap_utils*), 320
Utils (*class in allensdk.model.biophysical.utils*), 362

V

validate_epoch_durations() (*in module allensdk.brain_observatory.ecephys.stimulus_table.output_validatio*), 182
validate_epoch_order() (*in module allensdk.brain_observatory.ecephys.stimulus_table.output_validatio*), 182

validate_mask() (*in module allensdk.brain_observatory.roi_masks*), 216
validate_max_spontaneous_epoch_duration() (*in module allensdk.brain_observatory.ecephys.stimulus_table.output_validation*), 182
validate_paths() (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncWriter method*), 357
validate_structure_id() (*allensdk.core.reference_space_cache.ReferenceSpaceCache class method*), 265
validate_structure_ids() (*allensdk.core.reference_space_cache.ReferenceSpaceCache class method*), 265
validate_structures() (*allensdk.core.reference_space.ReferenceSpace method*), 262
validate_swc() (*in module allensdk.internal.morphology.validate_swc*), 342
validate_swc() (*in module allensdk.morphology.validate_swc*), 373
validate_trial_condition_exclusivity() (*in module allensdk.brain_observatory.behavior.trials_processing*), 165
validate_with_synthetic_F() (*in module allensdk.brain_observatory.r_neuropil*), 213
value_map() (*allensdk.core.simple_tree.SimpleTree method*), 267
values (*allensdk.brain_observatory.running_speed.RunningSpeed attribute*), 216
values() (*allensdk.brain_observatory.behavior.stimulus_processing.stimulus_templates.StimulusTemplate method*), 127
verify_roi_lists_equal() (*allensdk.brain_observatory.session_analysis.SessionAnalysis method*), 219
version (*allensdk.api.cloud_cache.manifest.Manifest attribute*), 67
VERSION (*allensdk.config.manifest.Manifest attribute*), 238
version_id (*allensdk.api.cloud_cache.file_attributes.CacheFileAttributes attribute*), 66
vin (*allensdk.brain_observatory.ecephys.file_io.stim_file.CamStimOnePickleStimFile attribute*), 173
visual_degrees_to_pixels() (*allensdk.brain_observatory.stimulus_info.BrainObservatoryMonitor method*), 224
visual_degrees_to_pixels() (*allensdk.brain_observatory.stimulus_info.Monitor method*), 225
voltage_component_of_threshold_exact() (*in module allensdk.model.glif.glif_neuron_methods*), 371
voltage_component_of_threshold_forward_euler() (*in module allensdk.model.glif.glif_neuron_methods*), 372
voltage_deflection() (*allensdk.ephys.ephys_extractor.EphysSweepFeatureExtractor method*), 278
VolumeProjector (*class in allensdk.internal.mouse_connectivity.projection_thumbnail.volume_*), 349
VOXEL_RESOLUTION_100_MICRONS (*allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute*), 88
VOXEL_RESOLUTION_10_MICRONS (*allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute*), 88
VOXEL_RESOLUTION_25_MICRONS (*allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute*), 88
VOXEL_RESOLUTION_50_MICRONS (*allensdk.api.queries.reference_space_api.ReferenceSpaceApi attribute*), 88
vsig (*allensdk.brain_observatory.ecephys.file_io.stim_file.CamStimOnePi attribute*), 173

W

warp_coordinates (*allensdk.brain_observatory.stimulus_info.ExperimentGeometry attribute*), 224
warp_image() (*allensdk.brain_observatory.stimulus_info.BrainObserva method*), 224
warp_stimulus_coords() (*in module allensdk.brain_observatory.stimulus_info*), 227
wedge_ring() (*in module allensdk.brain_observatory.circle_plots*), 203
whitelist() (*allensdk.core.structure_tree.StructureTree static method*), 271
WhitespaceStrippedString (*class in allensdk.test_utilities.custom_comparators*), 381
whole_lotta_trials() (*in module allensdk.brain_observatory.behavior.criteria*), 137
width (*allensdk.brain_observatory.stimulus_info.Monitor attribute*), 225
window_average() (*in module allensdk.mouse_connectivity.grid.utilities.downsampling_utilities*), 378
wrap() (*allensdk.api.warehouse_cache.cache.Cache method*), 99

write() (*allensdk.config.model.description_parser.DescriptionParser method*), 238
write() (*allensdk.config.model.formats.hdf5_util.Hdf5Util method*), 235
write() (*allensdk.config.model.formats.json_description_parser.JsonDescriptionParser method*), 235
write() (*allensdk.config.model.formats.pycfg_description_parser.PycfgDescriptionParser method*), 236
write() (*allensdk.core.swc.Morphology method*), 275
write() (*allensdk.internal.brain_observatory.frame_stream.FrameOutputStream method*), 301
write() (*allensdk.internal.morphology.morphology.Morphology method*), 339
write() (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncWriter method*), 357
write() (*in module allensdk.core.json_utilities*), 252
write_file() (*allensdk.internal.api.queries.biophysical_module_reader.BiophysicalModuleReader method*), 290
write_file() (*allensdk.internal.api.queries.optimize_config_reader.OptimizeConfigReader method*), 292
write_itksnap_labels() (*allensdk.core.reference_space.ReferenceSpace method*), 263
write_json_file() (*allensdk.config.manifest_builder.ManifestBuilder method*), 240
write_json_string() (*allensdk.config.manifest_builder.ManifestBuilder method*), 240
write_ndarray_with_sitk() (*in module allensdk.core.sitk_utilities*), 268
write_or_print_outputs() (*in module allensdk.brain_observatory.argschema_utilities*), 199
write_output() (*in module allensdk.internal.pipeline_modules.run_ophys_eye_calibration*), 356
write_output_data() (*allensdk.internal.core.lims_pipeline_module.PipelineModule method*), 309
write_output_h5() (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncWriter method*), 358
write_output_json() (*allensdk.internal.pipeline_modules.run_ophys_time_sync.TimeSyncWriter method*), 358
write_receptive_field_to_h5() (*in module allensdk.brain_observatory.receptive_field_analysis.receptive_field*), 193
write_string() (*allensdk.config.model.formats.json_description_parser.JsonDescriptionParser method*), 235
write_string() (*allensdk.config.model.formats.pycfg_description_parser.PycfgDescriptionParser method*), 236
write_string() (*in module allensdk.core.json_utilities*), 252
write_sweep_response_file() (*in module allensdk.model.glif.simulate_neuron*), 373
write_volume() (*in module allensdk.mouse_connectivity.grid.utilities.image_utilities*), 379

Y

yesterday_was_good() (*in module allensdk.brain_observatory.behavior.criteria*), 187