IBL dataset types

Title	Version	Last consented (Due for review)
IBL dataset types	v0.0.0	Never
Authors		
Olivier Winter, Gaelle Chapuis, Niccolò Bonacchi, Miles Wells		
Editors		**Working Group**
		Data Architecture Working Group
Abstract		
Explanation of some of the variables listed in PyBpod DatasetTypes		

Table of contents

Introduction to the IBL dataset

The aim of the International Brain Laboratory (IBL) is to understand the brain functions underlying decision making. Understanding these processes is a problem with a scale and complexity that far exceed what can be tackled by any single laboratory and that demands computational theory to be interwoven with experimental design and analysis in a manner not yet achieved. To overcome these challenges, we have created a virtual laboratory, unifying a group of 22 highly experienced neuroscience groups distributed across the world.

The data consists of neurophysiological and behavior measurements acquired in mice, with the aim to record from the entire mouse brain across animals. Neuropixels probes are placed in the mouse brain whilst it performs a decision-making task. Various sensors are used to monitor the animal's performance and condition (e.g. cameras, humidity and temperature sensors, microphone etc. see https://elifesciences.org/articles/63711 for details). The data are acquired in various laboratories across the world, and are typically processed (using heavy algorithms such as e.g. Deep Lab Cut (https://github.com/DeepLabCut/DeepLabCut) for tracking points on video data; Kilosort (https://github.com/MouseLand/Kilosort) for detecting and sorting the spikes of cells on the ephys traces) locally before being sent to a centralized database (see https://www.biorxiv.org/content/10.1101/827873v3 for details).
These data are used by a large group of theoreticians and analysts across the globe, with the aim to uncover what are the sets of brain functions that guide decision-making.

Introduction to ALF

IBL datasets are typically saved into files that follow the ALF naming convention, a way of representing collections of related data files with a given directory and filename structure. For more information, see the ALF reference page.

Folder tree structure

See the documentation on Data organisation .

How to get probe type

From spikeglx meta data file

```
from ibllib.io import spikeglx
ap_file = '/mnt/Data/Subjects/ZM_1735/2019-08-01/001/raw_ephys_data.ap.bin'
sr = spikeglx.Reader(ap_file)
```

```
sr.version
```

Returns '3A' or '3B1' or '3B2'

From folder structure

```
from ibllib.io import spikeglx
session_path = '/mnt/Data/Subjects/ZM_1735/2019-08-01/001/raw_ephys_data'
spikeglx.get_neuropixel_version_from_folder(session_path)
```

Returns '3A' or '3B' as it's impossible to differentiate between 3B1 and 3B2

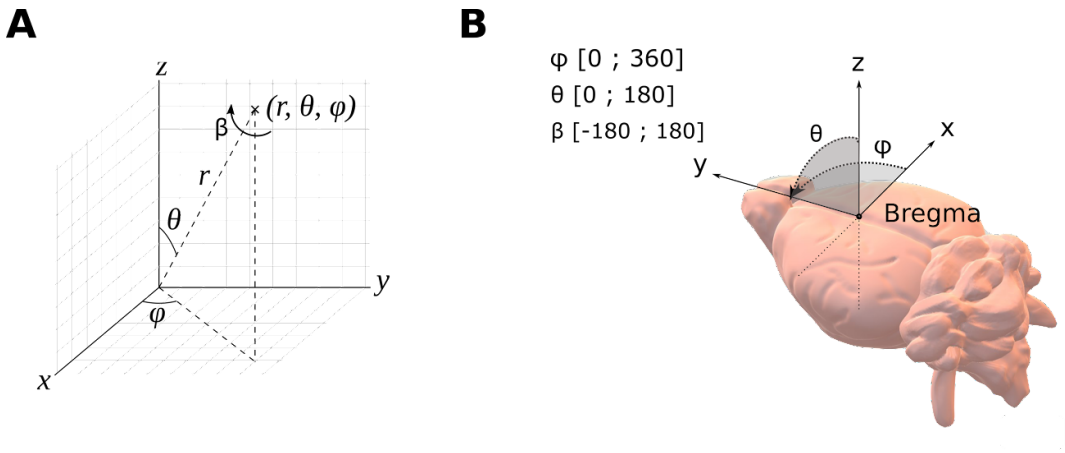
How to load spike sorting data

https://github.com/int-brain-lab/ibllib/blob/develop/examples/one/ephys/get_spikeData_and_brainLocations.py

TODO Explain spikesorting structure and relationship with histology

Coordinate System Within Allen CCF

Bregma is defined as Voxel ML-566, AP-540, DV-33 within the 10um volume of the Allen CCF mouse Atlas[15]



Coordinate System: ***A****) Standard polar angles are used, alongside the standard [x,y,z] base. In total, an electrode trajectory registers 7 entries (3 angles, 3 coordinates, depth) to define a probe insertion.* ***B****) The coordinate system is aligned to the ML (x, Right: positive), AP (y, A: positive), DV (z, D: positive) stereotaxic axis and zeroed at Bregma. The boundaries for each angle are indicated.*

Computing response type using choice and stimulus side

Response type are: correct or incorrect.

3 variables are needed to compute it:

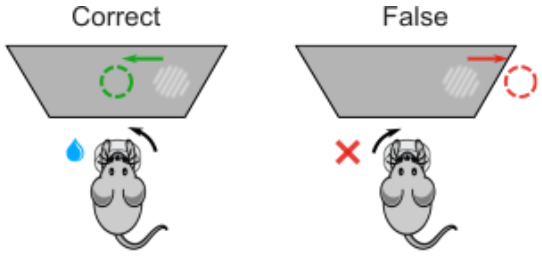
Variable	Type	Definition
_ibl_trials.choice	int64	which choice was made in choiceworld: -1 (turn CCW), +1 (turn CW), or 0 (nogo)
_ibl_trials.contrastLeft	float64	contrast of left-side stimulus (0...1) nan if trial is on other side
_ibl_trials.contrastRight	float64	contrast of right-side stimulus (0...1) nan if trial is on other side

CCW : counter-clockwise
CW : clockwise

`CW` and `CCW` indicate how the mouse turns the wheel (from the perspective of the mouse).

Thus, if the stimulus appears on the **right** side of the screen, and the mouse turns the wheel **`CCW`**, the response type is **correct**.

Similarly, if the stimulus appears on the **right** side of the screen, and the mouse turns the wheel **`CW`**, the response type is **false**.



Example

Variable	Trials							
choice	+1 (CW)	+1 (CW)	+1 (CW)	-1 (CCW)	-1 (CCW)	-1 (CCW)	0 (nogo)	0 (nogo)
contrastLeft	0.5	nan	nan	1	nan	nan	nan	1

contrastRight	nan	1	0	nan	0.5	0	0	nan
feedbackType	+1	-1	-1	-1	+1	+1	-1	-1

Aligning events to the onset of specific feedback type

The feedback variable contains all types of feedback received by the animal (e.g. noise sound, reward).

When wanting to align events (e.g. spikes) to the onset of one specific type of feedback, make sure to select only the corresponding feedback times.

Variable	Type	Definition
_ibl_trials.feedbackType	int64	Whether feedback is positive or negative in choiceworld (-1 for negative feedback, +1 for positive feedback, no_go trials receive negative feedback)
_ibl_trials.feedback_times	float64	Time of feedback delivery (reward or not) in choiceworld - in absolute seconds, rather than relative to trial onset

Example

Variable	Trials					
_ibl_trials.feedbackType	+1	0	+1	-1	0	+1
_ibl_trials.feedback_times	2.1	3.5	4.1	6.0	8.4	9.5
REWARD	x		x			x

REWARD trial ID : [1, 3, 6]
REWARD times: [2.1, 4.1, 9.5]

Spikes

Collection: alf/probe00, alf/prob01, etc.

spikes.amps

Units: V

Type: double

Dimensions: [number of spikes]

Format: npy

Description: Amplitude of each spike (V)

spikes.clusters

Units: index

Type: int

Dimensions: [number of spikes]

Format: npy

Description: Cluster assignments for each spike (integers counting from 0). Can be used as direct indexing of the cluster.* attributes.

spikes.depths

Units: um

Type: double

Dimensions: [number of spikes]

Format: npy

Description: Depth along probe of each spike (µm; computed from waveform center of mass). 0 means probe tip, positive upwards.

spikes.samples

Units: seconds

Type: int

Dimensions: [number of spikes]

Format: npy

Description: Sample index of spikes in their own electrophysiology binary file.

spikes.times

Units: seconds

Type: float64

Dimensions: [number of spikes]

Format: npy

Description: Times of spikes (seconds, relative to experiment onset).

spikes.templates

Units: index

Type: int

Dimensions: [number of spikes]

Format: npy

Description: Detection template assignment for each spike (integers counting from 0). Can be used as direct indexing of the templates.* attributes.

Clusters

Collection: alf/probe00, alf/prob01, etc.

clusters.amps

Units: V

Type: double

Dimensions: [number of clusters]

Format: npy

Description: Average spike amplitude for the cluster (V)

clusters.depths

Units: µm

Type: double

Dimensions: [number of clusters]

Format: npy

Description: Depth along probe of each cluster (µm; computed from waveform center of mass). 0 means deepest site, positive means above this.

clusters.channels

Units: index

Type: int

Dimensions: [number of clusters]

Format: npy

Description: Channel assignments for each cluster (integers counting from 0). Refers to the channel that contains peak amplitude. Can be used as direct indexing of the channel.* attributes.

clusters.metrics

Units: N/A

Type: N/A

Dimensions: [number of clusters]

Format: csv

Description: Quality control metrics at the cluster level. Fields may vary but a basic subset includes: cluster_id, num_spikes, firing_rate, presence_ratio, presence_ratio_std, isi_viol, amplitude_cutoff, amplitude_std, ks2_label, ks2_contamination_pct.
Detailed description of each metric here: https://docs.google.com/document/d/1ba_krsfm4epiAd0zbQ8hdvDN908P9VZOpTxkkH3P_ZY/edit

clusters.probes

Units: index

Type: int

Dimensions: [number of clusters]

Format: npy

Description: Probe assignments for each cluster (integers counting from 0). Can be used as direct indexing of the probes.* attributes.

clusters.peakToTrough

Units: ms

Type: double

Dimensions: [number of clusters]

Format: npy

Description: Template waveform duration for the cluster. The duration is the time elapsed from peak to through (peak_sample - through_sample). Value can be negative if peak happens first (ms).

clusters.uuids

Units: int128

Type: UUIDv4

Dimensions: [number of clusters]

Format: csv

Description: Unique identifier assigned to each cluster when ALF files created and during manual curation.

clusters.waveforms

Units: V

Type: single

Dimensions: [number of clusters, number of time samples, number of selected channels]

Format: npy

Description: Waveform from spike sorting templates (stored as a sparse array, only for a subset of channels closest to the peak channel).

clusters.waveformsChannels

Units: Index

Type: int

Dimensions: [number of clusters, number of selected channels]

Format: npy

Description: Index of channels that are stored for each cluster waveform. Sorted by increasing distance from the maximum amplitude channel.

Drift Registration

drift.um

Units: um

Type: double

Dimensions: [ntimes, ndepths]

Format: npy

Description: "Result of the drift registration. 1D vector of drift estimation in um. A cell moving downwards relative to the probe will show as negative drift (ie. it overlays on the raster).

For non-rigid drift, 2D vector where the first dimension corresponds to the time of the estimation and the second dimension to the depth of the estimate (see the scale datasets)"

drift.times

Units: secs

Type: double

Dimensions: [ntime]

Format: npy

Description: Time of the probe corresponding to each of the drift matrix row.

drift_depths.um

Units: um

Type: double

Dimensions: [ntime]

Format: npy

Description: For non-rigid registration, depth corresponding to each of the drift matrix column.

Probes

probes.description

Units: N/A

Type: text

Dimensions: [number of probes]

Format: json

Description: JSON with one entry per probe containing label, model (3A, 3B1, 3B2),serial and raw_file_name.

```
import json
 with open(probe_description_file) as fid:
   probes_description = json.reads(fid.read())
```

Channels

Collection: alf/probe00, alf/prob01, etc.

channels.localCoordinates

Units: index

Type: float

Dimensions: [number of channels]

Format: npy

Description: relative to probe coordinate system (µm): x (first) dimension is on the width of the shank while (y) is the depth where 0 is the deepest site, and positive above this.

channels.probes

Units: index

Type: int

Dimensions: [number of channels]

Format: npy

Description: Probe assignments for each channel (integers counting from 0). Can be used as direct indexing for the probes.* attributes.

channels.rawInd

Units: index

Type: int

Dimensions: [number of channels]

Format: npy

Description: array of integers saying which index in the raw recording file (of its home probe) that the channel corresponds to (counting from zero). Can be used as direct indexing for the raw binary files (AP and LF).

channels.mlapdv

Units: µm

Type: double

Dimensions: [number of channels, 3]

Format: npy

Description: 3d location of the channels relative to bregma following ephys alignment - mediolateral; anterior-posterior; dorsoventral coordinates (um)

channels.brainLocationIds_ccf_2017

Units: Index

Type: int

Dimensions: [number of channels]

Format: npy

Description: Brain location id of channels following ephys alignment obtained from 25um resolution 2017 Allen Common Coordinate Framework

Templates

Collection: alf/probe00, alf/prob01, etc.

template.amps

Units: V

Type: double

Dimensions: [number of templates]

Format: npy

Description: Average spike amplitude for the template (V)

templates.waveforms

Units: V

Type: single

Dimensions: [number of templates, number of time samples, number of selected channels]

Format: npy

Description: Waveform of automatic spike sorting templates (stored as a sparse array, only for a subset of channels closest to the peak channel)

templates.waveformsChannels

Units: Index

Type: int

Dimensions: [number of templates, number of selected channels]

Format: npy

Description: Index of channels that are stored for each template. Sorted by increasing distance from the maximum amplitude channel.

Sample waveforms

_phy_spikes_subset.waveforms

Units: V

Type: double

Dimensions: [number of spikes, number of time samples, number of selected channels]

Format: npy

Description: Sample waveform of spike traces extracted from raw data

_phy_spikes_subset.channels

Units: Index

Type: int

Dimensions: [number of spikes, number of selected channels]

Format: npy

Description: Index of channels that are stored for each sample spike waveform. Sorted by increasing distance from the maximum amplitude channel.

_phy_spikes_subset.spikes

Units: Index

Type: int

Dimensions: [number of spikes]

Format: npy

Description: Index of spikes

Raw Data

The following datasets can be loaded and read by our automatic extraction pipeline.

_ibl_experiment.description

Units: N/A

Type: text

Dimensions: N/A

Format: yaml

Collection: N/A

Description: A list of acquisition devices and behaviour protocols, along with the data and sync file location. The root keys are (devices, sync, task, procedures).

*_*_DAQdata.raw*

Units: voltes, samples, seconds

Type: float64

Dimensions: [nSamples, nch]

Format: bin, tdms, mat

Collection: raw_sync_data, raw_widefield_data, raw_photometry_data, etc.

Description: The output of the DAQ software for a given device. The namespace should identify the DAQ model, e.g. ‘mcc’, ‘ni’ or ‘ni-usb-6211’.

*_*_DAQdata.wiring*

Units: N/A

Type: text

Dimensions: [channels, device_name]

Format: json

Collection: raw_widefield_data, raw_photometry_data, etc.

Description: A map of the channel IDs and their corresponding device label, e.g. {ai1: bpod}. There are three root keys: 'SYSTEM', 'SYNC_WIRING_DIGITAL' and 'SYNC_WIRING_ANALOG'. The corresponding 'SYSTEM' value is a label such as 'Widefield'.

Raw Electrophysiology Data

Collection: raw_ephys_data

_iblqc_ephysSpectralDensity.power

Units: V**2/Hz

Type: float32

Dimensions: [nfreqs, nch]

Format: npy

Description: Power Spectral Density for all channels (V**2/Hz).

_iblqc_ephysSpectralDensity.freqs

Units: Hz

Type: float32

Dimensions: [nfreqs]

Format: npy

Description: Frequency scale for the spectrogram (Hz).

_iblqc_ephysTimeRms.rms

Units: V

Type: float32

Dimensions: [ntwin, nch]

Format: npy

Description: RMS amplitude as a function of time (V)

_iblqc_ephysTimeRms.timestamps

Units: s

Type: float32

Dimensions: [ntwin]

Format: npy

Description: Time scale for the RMS amplitude as a function of time, relative to the raw binary ephys file (s).

_iblqc_ephysChannels.apRMS

Units: V

Type: float32

Dimensions: [nchan]

Format: npy

Description: Median RMS per channel from samples of raw [nch, 0] and destriped [nch, 1] ephys data (V). (ap: computed for the AP band)

_iblqc_ephysChannels.labels

Units: N/A

Type: int

Dimensions: [nchan]

Format: npy

Description: Labels of automatic bad channel detections (int) 0: ok, 1: dead or low amplitude, 2: noisy, 3: out of the brain.

_iblqc_ephysChannels.rawSpikeRates

Units: Hz

Type: float32

Dimensions: [nchan]

Format: npy

Description: Raw spike rate per channel without clustering or collision handling - for early stages raw electrophysiology data QC purposes. [Hz]

_spikeglx_sync.channels

Units: index

Type: int

Dimensions: [nsync_pulses]

Format: npy

Description: 0 based index corresponding to the raw file sync channel used [0-15 on 3A, 0-8 on 3B].

_spikeglx_sync.polarities

Units: sign [-1, 1]

Type: float32

Dimensions: [nsync_pulses]

Format: npy

Description: Polarity of detected front: 1: rising, -1 falling

ephysData.raw.lf

Units: samples

Type: int16

Dimensions: [number of samples, number of channels]

Format: flat binary (*.bin), or mtscomp compressed (*.cbin, *.ch)

Description: Raw electrophysiology data. Low Field Potential.

```
# the following examples will read and convert to Volts
from ibllib.io import spikeglx
lf_file = "/path/to/my/_spikeglx_ephysData_g0_t0.imec1.lf.cbin"
sr = spikeglx.Reader(lf_file)
sr[0:10000]
# this assumes you have 3 files on disk:
# _spikeglx_ephysData_g0_t0.imec1.lf.cbin
# _spikeglx_ephysData_g0_t0.imec1.lf.meta
# _spikeglx_ephysData_g0_t0.imec1.lf.ch
```

Note: The sampling frequency can be accessed via `sr.fs` (cf SpikGLX doc)

ephysData.raw.ap

Units: samples

Type: int16

Dimensions: [number of samples, number of channels]

Format: flat binary (*.bin), or mtscomp compressed (*.cbin, *.ch)

Description: Raw electrophysiology data. Action Potential.

```
# the following examples will read and convert to Volts
from ibllib.io import spikeglx
ap_file = "/path/to/my/_spikeglx_ephysData_g0_t0.imec1.ap.cbin"
sr = spikeglx.Reader(ap_file)
sr[0:10000]
# this assumes you have 3 files on disk:
# _spikeglx_ephysData_g0_t0.imec1.ap.cbin
# _spikeglx_ephysData_g0_t0.imec1.ap.meta
# _spikeglx_ephysData_g0_t0.imec1.ap.ch
```

ephysData.raw.nidq

Units: samples

Type: int16

Dimensions: [number of samples, number of channels]

Format: flat binary (*.bin), or mtscomp compressed (*.cbin, *.ch)

Description: Raw electrophysiology data from the breakout box. Contains synchronization pulses from behaviour devices for 3B Neuropixel.

ephysData.raw.meta

Format: text

Description: Raw electrophysiology data support file containing acquisition information.

```
from ibllib.io import spikeglx
metadata_file = "/path/to/my/_spikeglx_ephysData_g0_t0.imec1.lf.meta"
spikeglx.read_meta_data(metadata_file)
```

ephysData.raw.sync

Units: s

Type: float64

Dimensions: [number of sync samples, 2]

Format: npy

Description: Synchronization for binary file: 2 columns giving self-time in seconds and reference time in seconds

ephysData.raw.timestamps

Units: samples/seconds

Type: float64

Dimensions: [number of sync samples, 2]

Format: npy

Description: 2 columns file containing time synchronisation information for the AP binary file: sample index in the first column and session time in the second column. Note that sample indices may not be integers

ephysData.raw.wiring

Format: json

Description: Description of wirings of the synchronization traces. Examples for 3A and 3B.

Raw Behavior Data

Collection: raw_behavior_data, raw_task_data_XX

audioOnsetGoCue.times_mic

Units: seconds

Type: double

Dimensions: [number of go Cues detected]

Format: npy

Description: GoCue Times extracted from microphone data (s), in the time reference of the microphone. If the extraction algorithm managed to align to the behaviour timing, another `_iblmic_onsetGoCue.times` file will be present.

_iblmic_audioSpectrogram.frequencies

Units: Hz

Type: double

Dimensions: [1, number of frequencies]

Format: npy

Description: Frequency scale of audio spectrogram from microphone (Hz).

_iblmic_audioSpectrogram.power

Units: Hz

Type: double

Dimensions: [number of time windows, number of frequencies]

Format: npy

Description: Audio spectrogram from microphone, power.

_iblmic_audioSpectrogram.times_mic

Units: s

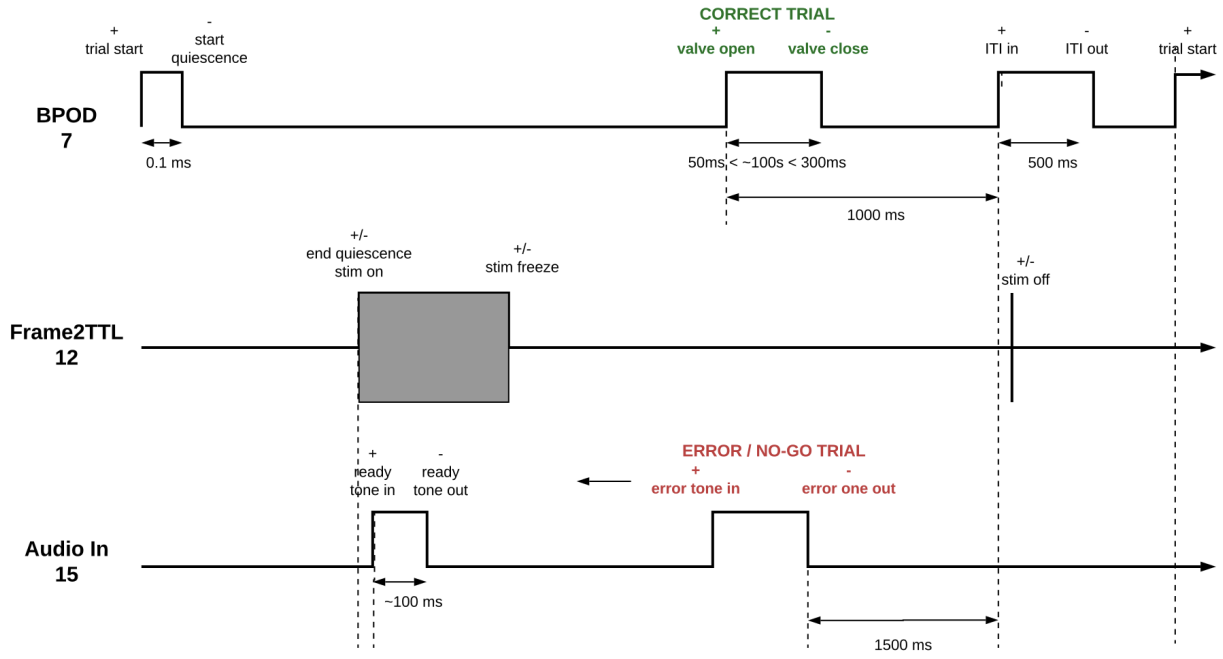
Type: double

Dimensions: [number of time windows]

Format: npy

Description: Time of the middle sample of the time window (s). Relative to the microphone samples time.

Trial Data



Note: All times are in absolute seconds from session start.

Collection: alf

_ibl_trials.table

Units: various

Type: various

Dimensions: [number of trials, n]

Format: pqt

Description: A table of trial events, one per trial. Events include: intervals, goCue_times, response_times, choice, stimOn_times, contrastLeft, contrastRight, feedback_times, feedbackType, rewardVolume, probabilityLeft, firstMovement_times

Table column descriptions:

intervals

Units: seconds

Type: double

Dimensions: [number of trials, 2]

Description: 2 column array giving each trial's start (i.e. beginning of quiescent period) and end (i.e. stimulus off + 500ms*) times of trials in universal seconds. As a DataFrame, this will be split into 2 columns, intervals_0 and intervals_1.

*Note that the minimum ITI is 500ms, however due to implementation there is also a significant variable delay between intervals (typically an ITI is around a second). Therefore the true ITI can be found by taking intervals[n+1, 0] - stimOff_times[n].

goCue_times

Units: seconds

Type: double

Dimensions: [number of trials]

Description: The start time of the go cue tone. This is the time the sound is actually played, that is, the command sent through soundcard sync was fed back into Bpod. The go cue tone is a 100ms 5 kHz sine wave and occurs approximately at the time of stimulus onset.

response_times

Units: seconds

Type: double

Dimensions: [number of trials]

Description: The time at which a response was recorded. This marks the end of the closed loop state in Bpod and occurs when either 60 seconds have elapsed since the go cue, or the rotary encoder reaches a position equivalent to the stimulus on the screen reaching + or - 35° azimuth.

choice

Units: sign [-1, 0, 1]

Type: int64

Dimensions: [number of trials]

Description: The response type registered for each trial where -1 corresponds to turning the wheel CCW, +1 turning CW, and 0 being a timeout ('no-go') where the wheel wasn't moved to threshold within the 60 second time window. See Computing response type using choice and stimulus side.

stimOn_times

Units: seconds

Type: double

Dimensions: [number of trials]

Description: The time at which the visual stimulus appears on the screen, as detected by the photodiode which is placed over the sync square that flips colour each time the screen is redrawn.

contrastLeft

Units: normalized

Type: double

Dimensions: [number of trials]

Description: The contrast of the stimulus that appears on the left side of the screen (-35° azimuth). When there is a non-zero contrast on the right, contrastLeft == 0, when there is no contrast on either side (a 'catch' trial), contrastLeft == NaN.

contrastRight

Units: normalized

Type: double

Dimensions: [number of trials]

Description: The contrast of the stimulus that appears on the right side of the screen (35° azimuth). When there is a non-zero contrast on the left, contrastRight == 0, when there is no contrast on either side (a 'catch' trial), contrastRight == NaN.

feedback_times

Units: seconds

Type: double

Dimensions: [number of trials]

Description: The time of feedback delivery. For correct trials this is the time of the valve TTL trigger command, for incorrect trials this is the time of the white noise output trigger.

feedbackType

Units: sign [-1, 1]

Type: int64

Dimensions: [number of trials]

Description: Whether the feedback was positive (+1) or negative (-1). Positive feedback indicates a correct response rewarded with sugar water. Negative feedback indicates a trial timeout or incorrect response followed by a white noise burst.

rewardVolume

Units: µl

Type: double

Dimensions: [number of trials]

Description: The volume of reward delivered on each trial. On trials where feedbackType == -1, rewardVolume == 0. The reward volume is typically within the range of 1.5 to 3 and should not change within a session.

probabilityLeft

Units: normalized

Type: double

Dimensions: [number of trials]

Description: Probability that the stimulus will be on the left-hand side for the current trial. The probability of right is 1 minus this. For repeat trials (trainingChoiceWorld) the probability is N(bias,4), where bias is calculated using responses from the last 10 trials. During biased blocks (biasedChoiceWorld) the probability may be 0.5, 0.8, or 0.2.

firstMovement_times

Units: seconds

Type: double

Dimensions: [number of trials]

Description: The time of the first detected movement of the trial with a sufficiently large amplitude. To be counted, the movement must occur within the open-loop period, which starts just before the go cue* and ends at feedback time. A NaN value indicates that no movement was detected for that trial.

*The first movement onset is sometimes just before the cue, occurring in the gap between quiescence end and cue start, or during the quiescence period but sub-threshold. The movement is sufficiently large if it is greater than or equal to .1 radians.

_ibl_trials.stimOff_times

Units: seconds

Type: float

Dimensions: [number of trials]

Format: npy

Description: Time in seconds, relative to the session start, of the stimulus offset, as recorded by an external photodiode.

_ibl_trials.quiescencePeriod

Units: seconds

Type: float

Dimensions: [number of trials]

Format: npy

Description: The pre-stimulus quiescence period in seconds. The exact time for each trial is drawn from a truncated exponential, range 0.2-0.5 seconds.

_ibl_subjectTrials.table

Collection: aggregates/Subjects/{lab}/{Subject}/

Units: various

Type: various

Dimensions: [number of all trials, n]

Format: pqt

Description: Aggregate of all trials of one subject, aggregated from all sessions that have complete trials data. One row per trial, columns represent trial events and include all columns of _ibl_trials.table (`intervals`, `goCue_times`, `response_times`, `choice`, `stimOn_times`, `contrastLeft`, `contrastRight`, `feedback_times`, `feedbackType`, `rewardVolume`, `probabilityLeft`, `firstMovement_times`) plus information about the session the trial stems from (`session`, `session_start_time`, `session_number` `task_protocol`), plus the two columns `stimOnTrigger_times` and `quiescence` (equal to the dataset _ibl_trials.quiescencePeriod.npy.

_ibl_subjectTraining.table

Collection: aggregates/Subjects/{lab}/{Subject}/

Units: various

Type: various

Dimensions: [number of training criteria reached, 2]

Format: pqt

Description: Overview of training status for subject, each row contains the date and session at which the subject reached different training criterion. Index is date, columns are training_status and session .

Possible status
['habituation', 'in training', 'trained 1a', 'trained 1b', 'ready4ephysrig',, 'ready4delay', 'ready4recording', 'untrainable', 'unbiasable', 'not_computed']

Trial Data (optogenetics)

Collection: alf

_ibl_trials.laserStimulation

Units: seconds

Type: float

Dimensions: [number of trials]

Format: npy

Description: Whether or not the laser was actuating during the trial (outcome): 0:no laser, 1: laser, np.nan: no info.

_ibl_trials.laserProbability

Units: normalized

Type: float

Dimensions: [number of trials]

Format: npy

Description: Probability of the laser being actuating (prior).

_ibl_trials.laserIntervals

Units: seconds

Type: float

Dimensions: [number of trials, 2]

Format: npy

Description: Time in seconds, relative to the session start, of the onset and offset of the laser stimulation.

Wheel Data

Collection: alf

_ibl_wheel.position

Units: radians

Type: double

Dimensions: [number of wheel ticks]

Format: npy

Description: Absolute unwrapped angle of the wheel from session start. The sign from the subject perspective corresponds to mathematical convention: counter clockwise is positive.

_ibl_wheel.timestamps

Units: seconds

Type: double

Dimensions: [number of wheel ticks]

Format: npy

Description: Time in seconds of the corresponding wheel event, relative to session start. Note that the wheel delivers a sample only when crossing a tick, which makes this time-serie very irregular.

_ibl_wheel.velocity

Units: radians per second

Type: double

Dimensions: [number of wheel ticks]

Format: npy

Description: Velocity of the wheel. Same sign convention as position.

_ibl_wheelMoves.intervals

Units: seconds

Type: double

Dimensions: [number of movements, 2]

Format: npy

Description: The onset and offset times of all detected movements. Movements are defined as a wheel movement of at least 0.012 rad over 200ms. For a rotary encoder of resolution 1024 in X4 encoding, this is equivalent to around 8 ticks. Movements below 50ms are discarded and two detected movements within 100ms of one another are considered as a single movement. For the onsets a lower threshold is used to find a more precise onset time.

_ibl_wheelMoves.peakAmplitude

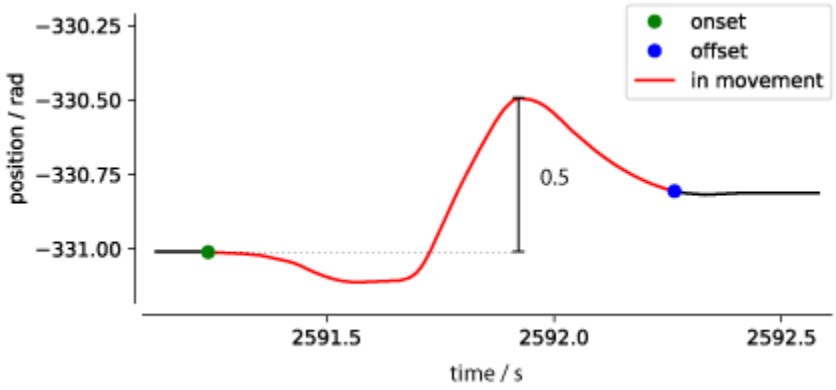
Units: radians

Type: double

Dimensions: [number of movements]

Format: npy

Description: The absolute maximum amplitude of each detected wheel movement, relative to onset position.


onset
offset
in movement
−330.25
−330.50
−330.75
−331.00
position / rad
0.5
2591.5
2592.0
2592.5
time / s

Camera Data

camera.times

Units: seconds

Type: double

Dimensions: [number of frames]

Format: npy

Collection: alf

Description: Time in seconds of the corresponding frame, relative to session start. These are the times of the camera TTL pulse during every frame capture. There are three ALFs associated with this dataset: _ibl_leftCamera.times, _ibl_rightCamera.times and _ibl_bodyCamera.times.

camera.raw

Units: frames

Type: double

Dimensions: [1280 (640), 1024 (512), number of frames]

Format: mp4 (h.264)

Collection: raw_video_data

Description: The compressed video from the cameras. There are three ALFs associated with this dataset: _iblrig_leftCamera.raw, _iblrig_rightCamera.raw, and _iblrig_bodyCamera.raw. The right side cameras will run at 150 Hz (spatial resolution 640 x 512) and the left side camera at 60 Hz (spatial resolution 1280 x 1024), the body camera at 30 Hz (spatial resolution 640 x 512)

_iblrig_*Camera.frame_counter

Units: number

Type: float64

Dimensions: [number of frames]

Format: bin

Collection: raw_video_data

Description: The frame counter from the camera hardware starts at arbitrary value. Saved as floating point 64 bits

_iblrig_*Camera.GPIO

Units: number

Type: float64

Dimensions: [number of frames]

Format: bin

Collection: raw_video_data

Description: The GPIO pin state from the camera hardware, saved as floating point 64 bits.

_iblrig_*Camera.frameData

Units: [

- UTC ticks from BehaviorPC,
- CameraTimestamps (Needs unclycling and conversion),
- Frame counter,
- GPIO pin state integer representation of 4 pins

]

Type: float64

Dimensions: [number of frames x 4]

Format: bin

Collection: raw_video_data

Description: Binary file of nframes X 4 columns storing the raw representation of:

Timestamp → UTC ticks from BehaviorPC (100's of ns since midnight 1/1/0001)
embeddedTimeStamp → Camera timestamp (Needs unclycling and conversion)
embeddedFrameCounter → Frame counter
embeddedGPIOPinState → GPIO pin state integer representation of 4 pins

camera.dlc

Units: number

Type: float32

Dimensions: [number of frames, number of tracked points x 3]

Format: pqt

Collection: alf

Description: Coordinates of points tracked in the videos using DeepLabCut (DLC). Can be read with pandas.read_parquet. For each tracked point, the data frame contains a row each for the x coordinate, y coordinate, and likelihood over time. There are three ALF files associated with this dataset type: _ibl_leftCamera.dlc.pqt, _ibl_rightCamera.dlc.pqt, and _ibl_bodyCamera.dlc.pqt The number of tracked points is 11 for the left and right camera and 1 for the body camera.

camera.ROIMotionEnergy

Units: number

Type: float32

Dimensions: [number of frames]

Format: npy

Collection: alf

Description: The motion energy between the current frame and the subsequent frame, calculated within a ROI (see ROIMotionEnergy.position). There are three ALF files associated with this dataset type: leftCamera.ROIMotionEnergy.npy, rightCamera.ROIMotionEnergy.npy, and bodyCamera.ROIMotionEnergy.npy

ROIMotionEnergy.position

Units: number

Type: int64

Dimensions: [4,]

Format: npy

Collection: alf

Description: The coordinates of the region of interest (ROI) in which the camera.ROIMotionEnergy is calculated. The entries of this array are (w, h, x, y) where w and h are the width and height of the ROI in pixels, and x and y are the coordinates of the upper left corner pixel of the ROI. There are three ALF files associated with this dataset type: leftROIMotionEnergy.position.npy, rightROIMotionEnergy.position.npy, and bodyROIMotionEnergy.position.npy

CAUTION: As each software will load the video in a different orientation, the ROI might need to be adapted. For example, when loading the video with cv2 in Python, x and y axes are flipped from the convention used above. The ROI then is this box: [y:y+h, x:x+w]

camera.features

Units: number

Type: float32

Dimensions: [number of frames, number of features]

Format: pqt

Collection: alf

Description: Features computed from the dlc trace of each camera. Currently, this contains the raw and smoothed pupil Diameter ('pupilDiameter_raw' and 'pupilDiameter_smooth'). As these are only available for left and right camera, there are currently two ALF files associated with this dataset type: _ibl_leftCamera.features.pqt and _ibl_rightCamera.features.pqt (but _ibl_bodyCamera.features.pqt might be added in the future).

licks.times

Units: number

Type: float32

Dimensions: [number of licks,]

Format: npy

Collection: alf

Description: Time stamps of licks as detected from tongue dlc traces. If left and right camera exist, the licks detected from both cameras are combined.

Passive Data

NB: All times are in absolute seconds from session start.
Passive protocol runs after ephys protocol description here.

_ibl_passivePeriods.intervalsTable*.csv

Units: seconds

Type: float64

Dimensions: [2, 4]
lines = ['start', 'stop']
columns = ['passiveProtocol', 'spontaneousActivity', 'RFM', 'taskReplay'],

Format: csv

Description: Table of times of different passiveCW periods.

_ibl_passiveRFM.times*.npy

Units: seconds

Type: float64

Dimensions: [nFrames]

Format: npy

Description: Interpolated times for all frame presentations in Receptive Field Mapping period.

_ibl_passiveGabor.table*.csv

Units: [seconds, seconds, visual degrees, [0, 1], [0, 2*Pi]

Type: float64

Dimensions: [nPresentations, 5]
lines = nPresentations = 180
columns = [start, stop, position, contrast, phase]

Format: csv

Description: Gabor patch presentations table of details of visual stimulus per presentation. Presentations with negative visual degrees appeared in the left visual hemifield; positive visual degrees, in the right.

_ibl_passiveStims.table*.csv

Units: seconds

Type: float64

Dimensions: [nRepeats, 6]
lines = nRepeats = 40
columns = [valveOn, valveOff, toneOn, toneOff, noiseOn, noiseOff]

Format: csv

Description: Onset and offset times of all other stimuli presented during the passiveCW protocol.

_iblrig_RFMapStim.raw*.bin

Units: color

Type: uint8

Dimensions: [nframes, nx, ny]
Where nx == ny == 15

Format: bin

Description: Raw receptive field mapping stimulus matrix.

Imaging Data (multi-photon calcium imaging)

Collection: alf/FOV_XX

mpci.times

Units: seconds

Type: float64

Dimensions: [nframes]

Format: npy

Description: The time of each frame. Note that there may be further offsets available for each scanline or pixel/voxel in the mpciStack.timeshift.npy dataset.

mpci.badFrames

Units: N/A

Type: boolean

Dimensions: [nframes]

Format: npy

Description: Bad frames as identified by processing software. Typically these are frames where the image registration was considered poor, for instance because of high SNR on the reference channel.

mpci.mpciFrameQC

Units: N/A

Type: uint8

Dimensions: [nframes]

Format: npy

Description: An enumeration of frame-level quality control as added by experimenter. 0 means good, other values defined in mpciFrameQC.names.tsv

mpciFrameQC.names

Units: N/A

Type: str

Dimensions: [nQCtypes, 2]

Format: tsv

Description: Human-readable definition of QC types. First column, 'qc_values', contains unsigned ints where 0 always should mean good. Second column, 'qc_labels', contains a short human-readable description.

mpci.ROIActivityF

Units: photodetector output units

Type: float32

Dimensions: [nFrames, nROIs]

Format: npy

Description: Mean activity of all pixels in each ROI.

mpci.ROINeuropilActivityF

Units: photodetector output units

Type: float32

Dimensions: [nFrames, nROIs]

Format: npy

Description: Mean activity neuropil pixels neighboring each ROI.

mpci.ROIActivityDeconvolved

Units: AU

Type: float32

Dimensions: [nFrames, nROIs]

Format: npy

Description: Neuropil-subtracted deconvolved activity of each ROI using standard parameters.

mpciROIs.mpciROITypes

Units: N/A

Type: int16

Dimensions: [nROIs]

Format: npy

Description: Numerical enumeration of ROI type for each ROI.

mpciROITypes.names

Units: N/A

Type: str

Dimensions: [nROItypes, 2]

Format: tsv

Description: Human-readable definition of ROI types (neuron, dendrite, etc). First column, 'roi_values', contains the ROI type enumeration. The second, 'roi_labels', contains a string description.

mpciROIs.cellClassifier

Units: N/A

Type: float64

Dimensions: [nROIs]

Format: npy

Description: Floating-point cell classifier score for each ROI, ranging between 0 and 1.

mpciROIs.masks

Units: AU

Type: float

Dimensions: [nROIs, H, W]

Format: sparse_npy

Description: Floating-point mask of each ROI, in 2D or 3D according to how you did detection. Saved as a sparse array. sum(roi_mask * frames[ypix,xpix,:]) = fluorescence

mpciROIs.neuropilMasks

Units: AU

Type: boolean

Dimensions: [nROIs, H, W]

Format: sparse_npy

Description: Boolean neuropil mask for each ROI, in 2D or 3D according to how you did detection. Saved as a sparse array.

mpciROIs.mlapdv

Units: um

Type: float64

Dimensions: [nROIs, 3]

Format: npy

Description: 3D location of the ROI centroid relative to bregma - mediolateral; anterior-posterior; dorsoventral coordinates (um). '_estimate' means not resolved with histology.

mpciROIs.brainLocationIds

Units: ID

Type: int32

Dimensions: [nROIs]

Format: npy

Description: Brain location acronym of each ROI centroid. A '_ccf_2017' in the dataset means an alignment obtained from 25um resolution 2017 Allen Common Coordinate Framework. '_estimate' means not resolved with histology.

mpciROIs.stackPos

Units: pixels

Type: int64

Dimensions: [nROIs, 3]

Format: npy

Description: X, Y, and Z (plane) pixel coordinates of each ROI's centroid.

_suite2p_ROIData.raw

Format: zip of npy

Description: Raw cell detection output ROI data.

Raw Neuroimaging Data

Collection: raw_imaging_data_XX, raw_widefield_data

imaging.frames

Units: pixels

Type: uint8

Dimensions: [nframes, nx, ny(, nchannels)]

Format: bin, dat, tar.bz2, tif

Description: Raw frame images with the dimensions [nframes, nx, ny(, nchannels)]. Note that depending on modality, multiple planes and channels may be stitched together and/or presented in a different order.

referenceImage.raw

Collection: raw_imaging_data_XX/reference

Units: pixels

Type: uint8

Dimensions: [nframes, nx, ny(, nchannels)]

Format: tif

Description: A fullfield multiphoton imaging picture for use in histology alignment.

referenceImage.stack

Collection: raw_imaging_data_XX/reference

Units: pixels

Type: uint8

Dimensions: [nframes, nx, ny(, nchannels)]

Format: tif

Description: A stitched mean multiphoton imaging stack, containing X and Y resolution/size information in the tiff header. Used in histology alignment.

referenceImage.meta

Collection: raw_imaging_data_XX/reference

Units: N/A

Type: string

Format: json

Description: Stack image meta-data listing the depth in microns of each slice in the stack, and ML and AP coordinates of the center.

Imaging Data (widefield, photometry, etc.)

imaging.times

Units: seconds

Type: float64

Dimensions: [nframes,]

Format: npy

Collection: alf/widefield, etc.

Description: The times of frame acquisition.

imaging.imagingLightSource

Units: index

Type: int32

Dimensions: [nframes,]

Format: npy

Description: Integer representing which channel was acquired for each frame. 0 channel indicates no light.

imagingLightSource.properties

Units: nm

Type: int32

Dimensions: [nLightSources,]

Format: htsv

Description: A map of channel number, colour and wavelength. NB: This is the light source wavelength, not the photosensor channel / emitted wavelength.

Widefield Data

widefieldEvents.raw

Units: N/A

Type: float

Dimensions: [nFrames, n]

Format: camlog

Collection: raw_widefield_data

Description: The output of the Camlog software used to collect widefield images. Contains frame times and LED number for each frame.

widefieldU.mask

Units: binary

Type: int32

Dimensions: [nH, nW]

Format: npy

Collection: raw_widefield_data

Description: User selected mask to restrict pixels in imaging.frames before applying preprocessing

widefieldChannels.wiring

Units: N/A

Type: text

Dimensions: [nLED, 2]

Format: json

Description: A map of LED wiring number to wavelength. Wiring number matches those in XXX column of widefieldEvents.raw.camlog

widefield.U.images

Units: pixels

Type: float

Dimensions: [nW, nH, nComponents]

Format: npy

Collection: alf/widefield

Description: Projected SVD spatial components (denoised/decomposed) where nW and nH are the dimensions of a single frame.

widefieldSVT.uncorrected

Units: pixels

Type: float

Dimensions: [nComponents, nFrames]

Format: npy

Collection: alf/widefield

Description: SVD temporal components (denoised/decomposed) of data.

widefieldSVT.haemoCorrected

Units: pixels

Type: float

Dimensions: [nComponents, nFrames]

Format: npy

Collection: alf/widefield

Description: Haemodynamic corrected SVD temporal components of data.

widefieldChannels.frameAverage

Units: pixels

Type: float

Dimensions: [nLED, nW, nH]

Format: npy

Collection: alf/widefield

Description: Average image across each frame for each LED channel used in imaging.

widefieldLandmarks.dorsalCortex

Units: N/A

Type: text

Dimensions: N/A

Format: json

Collection: alf/widefield

Description: User selected landmarks and transform for widefield imaging to allow images to be transformed and aligned to Allen Atlas

subjectWidefieldU.mask_atlasTransformed

Units: binary

Type: int32

Dimensions: [nW, nH]

Format: npy

Collection: aggregates/Subjects/{lab}/{Subject}/

Description: User selected mask to apply to processed atlas transformed widefeild images. One mask per subject.

Photometry Data

fpData.raw

Units: various

Type: various

Dimensions: [nFrames, n]

Format: pqt

Collection: raw_photometry_data

Description: Raw photometry data dataframe containing fluorescence data for every active channel (max: isosbestic, red and green) as well as information of coactive components with each frame (E.g TTL syncing pulse)

photometry.signal

Units: various

Type: various

Dimensions: [nFrames, n]

Format: pqt

Collection: alf/photometry

Description: A table containing photometry data for all photometry ROIS in a single session, one row per imaging frame. Columns include Region0G, Region1G … , the raw data from the individual regions. Times, the time of each frame. Wavelength and name, which describe the LED used to collect each frame. Include, a mask indicating which frames should be used in analysis.

photometryROI.locations

Units: various

Type: various

Dimensions: [nROI, n]

Format: pqt

Collection: alf/photometry

Description: Look up table from photometry ROI, to fiber name registered in the database and Allen brain location