session_demonstration_script

April 5, 2023

1 Example code for using session.Session

Note: This notebook covers several relevant methods of the `Session` and `Stim` objects, detailing some of their arguments, as well. For more details, take a look at the docstring associated with a method of interest.

Import notes:

- Any python packages required by the codebase should be installed and available, if the required conda environment, installed from `osca.yml`, has been activated.
- `util` is a Github repo of mine, and the correct branch `osca_mult` is automatically installed from `osca.yml`.

- **Potential updates:** Errors internal to the codebase involving `util` code and occurring *after* new changes have been pulled from the `OpenScope_CA_Analysis` repo *may* be due to an update of the `osca_mult` branch of `util` that breaks backwards compatibility.
  Though I will try to avoid this, if running the notebook locally and an error occurs, make sure to check whether there are updates to the utility, and update your installation as needed, e.g., by running, from the command line: `pip install -U util-colleenjg`

1.1 1. Setup

1.1.1 Update logging and plot formatting

If you wish to use the same logging and formatting style as I do:

1.1.2 Data directory

The data directory should contain the session data in **NWB format**, at any depth.

1.1.3 Running in a docker, or specifically in Binder

If the notebook is **running in a docker**, the dataset is downloaded in NWB format from the Dandi archive first, and the data directory is set accordingly.
In addition, if the notebook is **running specifically in Binder**, some analyses are slightly altered later, in order to reduce memory use.

If the notebook is **not running in a docker**, the dataset should be downloaded manually. It can be stored in any location. Ensure that you update `datadir` below to point to that location.

Be sure to download the dataset, if needed, and update `datadir` to point the correct location. Currently it points to ../../data/OSCA_NWB.

1.1.4 Mouse dataframe

The mouse dataframe, contains the metadata for each session, including its 9-digit `sessid`, the `mouse_n`, `sess_n`, etc.

Mouse dataframe columns:
* **`sessid`**: Unique session ID (9-digit)
* **`full_timestamp`**: Full timestamp for the session (in UTC).
* **`mouse_n`**: Mouse number
* **`mouseid`**: Unique mouse ID (6-digit)
* **`date`**: Recording date
* **`depth`**: Recording depth (um)
* **`plane`**: Recording plane ("dend" or "soma")
* **`line`**: Cell line ("L2/3-Cux2" or "L5-Rbp4")
* **`runtype`**: Type of session ("pilot" or "prod"). Only production data is available in NWB dataset on Dandi.
* **`sess_n`**: Session number
* **`nrois`**: Number of valid ROIs (see ***Note***)
* **`nrois_tracked`**: Number of ROIs tracked across sessions (-1 for sessions with no tracking).
* **`nrois_all`**: Same as **`nrois`**, but including bad (non valid) ROIs.
* **`nrois_allen`**: Number of valid ROIs when using the **`allen`** segmentation for dendritic ROIs, instead of the `extr` segmentation (see ***Note***).
* **`nrois_allen_all`**: Same as **`nrois_allen`**, but including bad (non valid) ROIs.
* **`pass_fail`**: Whether the session passed (`P`) or failed (`F`) quality control.
* **`all_files`**: Whether all files are available for the session (original data format).
* **`any_files`**: Whether any files are available for the session (original data format).
* **`incl`**: Whether the session can be included in analyses (looser criterion than **`pass_fail`**).
* **`stim_seed`**: Seed used to initialize stimuli for the session, during recording.
* **`notes`**: Any notes on the session.

Note: The `allen` segmentations were used for all **somatic** data. The `extr` segmentations were preferred for all **dendritic** data, and are the only type included for **dendritic** data in the NWB dataset.

[6]:
```
      sessid   full_timestamp  mouse_n  mouseid sex       DOB      date  \
0   758519303  20180926T172917        1   408021   M  20180623  20180926
1   759189643  20180927T182632        1   408021   M  20180623  20180927
2   759660390  20181001T172833        1   408021   M  20180623  20181001
3   759666166  20181001T180256        2   411400   F  20180711  20181001
4   759872185  20181002T173740        2   411400   F  20180711  20181002
5   760269100  20181003T180253        2   411400   F  20180711  20181003
6   761730740  20181009T175037        2   411400   F  20180711  20181009
7   762415169  20181011T174057        2   411400   F  20180711  20181011
8   763646681  20181015T173410        2   411400   F  20180711  20181015
9   761624763  20181009T152254        3   411424   F  20180711  20181009
10  761944562  20181010T160230        3   411424   F  20180711  20181010
```

11	762250376	20181011T155936	3	411424	F	20180711	20181011
12	760260459	20181003T174857	4	411771	M	20180713	20181003
13	760659782	20181004T173816	4	411771	M	20180713	20181004
14	761269197	20181008T172946	4	411771	M	20180713	20181008
15	763949859	20181016T220226	5	412933	M	20180718	20181016
16	764897534	20181017T221505	5	412933	M	20180718	20181017
17	765427689	20181019T001829	5	412933	M	20180718	20181018
18	766755831	20181022T225131	5	412933	M	20180718	20181022
19	767254594	20181023T220932	5	412933	M	20180718	20181023
20	768807532	20181025T222758	5	412933	M	20180718	20181025
21	764704289	20181017T173352	6	413663	M	20180721	20181017
22	765193831	20181018T173052	6	413663	M	20180721	20181018
23	766502238	20181022T172824	6	413663	M	20180721	20181022
24	777496949	20181113T015148	7	418779	F	20180820	20181112
25	778374308	20181114T030141	7	418779	F	20180820	20181113
26	779152062	20181115T010609	7	418779	F	20180820	20181114
27	777914830	20181113T180512	8	420011	F	20180826	20181113
28	778864809	20181114T180624	8	420011	F	20180826	20181114
29	779650018	20181115T183654	8	420011	F	20180826	20181115
30	826187862	20190220T212644	9	433414	F	20181106	20190220
31	826773996	20190221T202814	9	433414	F	20181106	20190221
32	827833392	20190222T221848	9	433414	F	20181106	20190222
33	826338612	20190220T230630	10	433448	M	20181106	20190220
34	826819032	20190221T220634	10	433448	M	20181106	20190221
35	828816509	20190225T214952	10	433448	M	20181106	20190225
36	829283315	20190226T224301	10	433448	M	20181106	20190226
37	823453391	20190215T165328	11	433451	M	20181106	20190215
38	824434038	20190218T162355	11	433451	M	20181106	20190218
39	825180479	20190219T162113	11	433451	M	20181106	20190219
40	826659257	20190221T162838	12	433458	M	20181106	20190221
41	827300090	20190222T161948	12	433458	M	20181106	20190222
42	828475005	20190225T162214	12	433458	M	20181106	20190225
43	829520904	20190227T162140	12	433458	M	20181106	20190227
44	832883243	20190306T163903	13	440889	F	20181212	20190306
45	833704570	20190307T163524	13	440889	F	20181212	20190307
46	834403597	20190308T164555	13	440889	F	20181212	20190308
47	836968429	20190314T152429	13	440889	F	20181212	20190314
48	837360280	20190315T152224	13	440889	F	20181212	20190315
49	838633305	20190318T232807	13	440889	F	20181212	20190318

	age_weeks	depth	plane	…	nrois_tracked	nrois_all	nrois_allen	\
0	13.571429	175	soma	…	59	116	96	
1	13.714286	175	soma	…	59	86	74	
2	14.285714	175	soma	…	59	121	107	
3	11.714286	20	dend	…	-1	1009	70	
4	11.857143	20	dend	…	-1	309	53	
5	12.000000	20	dend	…	-1	539	75	

6	12.857143	20	dend	…	162	657	57
7	13.142857	20	dend	…	162	662	25
8	13.714286	20	dend	…	162	602	33
9	12.857143	175	soma	…	55	96	87
10	13.000000	175	soma	…	55	94	90
11	13.142857	175	soma	…	55	88	80
12	11.714286	375	soma	…	47	104	90
13	11.857143	375	soma	…	47	76	70
14	12.428571	375	soma	…	47	99	9
15	12.857143	75	dend	…	-1	1073	12
16	13.000000	75	dend	…	-1	992	11
17	13.142857	75	dend	…	-1	881	12
18	13.714286	50	dend	…	98	350	63
19	13.857143	50	dend	…	98	172	70
20	14.142857	50	dend	…	98	256	108
21	12.571429	50	dend	…	136	644	86
22	12.714286	50	dend	…	136	359	158
23	13.285714	50	dend	…	136	529	104
24	12.000000	375	soma	…	12	31	15
25	12.142857	375	soma	…	12	56	26
26	12.285714	375	soma	…	12	55	29
27	11.285714	20	dend	…	51	222	57
28	11.428571	20	dend	…	51	166	94
29	11.571429	20	dend	…	51	192	165
30	15.142857	75	dend	…	118	736	91
31	15.285714	75	dend	…	118	311	88
32	15.428571	75	dend	…	118	340	106
33	15.142857	20	dend	…	112	1728	150
34	15.285714	20	dend	…	112	462	313
35	15.857143	20	dend	…	112	523	282
36	16.000000	20	dend	…	-1	456	256
37	14.428571	20	dend	…	353	1003	53
38	14.857143	20	dend	…	353	1070	63
39	15.000000	20	dend	…	353	1022	39
40	15.285714	375	soma	…	70	126	99
41	15.428571	375	soma	…	70	107	87
42	15.857143	375	soma	…	70	118	97
43	16.142857	375	soma	…	-1	109	88
44	12.000000	175	soma	…	147	251	224
45	12.142857	175	soma	…	147	251	224
46	12.285714	175	soma	…	147	228	210
47	13.142857	175	soma	…	-1	217	205
48	13.285714	175	soma	…	-1	244	217
49	13.714286	175	soma	…	-1	256	227

	nrois_allen_all	pass_fail	all_files	any_files	incl	stim_seed	\
0	116	P	1	1	yes	30587	

1	86	P	1	1	yes	5730
2	121	P	1	1	yes	36941
3	79	F	1	1	yes	11883
4	58	F	1	1	yes	8005
5	86	F	1	1	yes	34380
6	65	P	1	1	yes	44023
7	31	P	1	1	yes	29259
8	44	P	1	1	yes	1118
9	96	P	1	1	yes	997
10	94	P	1	1	yes	33856
11	88	P	1	1	yes	23187
12	104	P	1	1	yes	33767
13	76	P	1	1	yes	32698
14	99	P	1	1	yes	17904
15	12	F	1	1	yes	44721
16	11	F	1	1	yes	32579
17	12	F	1	1	yes	26850
18	75	P	1	1	yes	39002
19	84	P	1	1	yes	6698
20	129	P	1	1	yes	8612
21	102	P	1	1	yes	12470
22	193	P	1	1	yes	7038
23	127	P	1	1	yes	23433
24	31	P	1	1	yes	32706
25	56	P	1	1	yes	8114
26	55	P	1	1	yes	11744
27	61	P	1	1	yes	20846
28	101	P	1	1	yes	35159
29	178	P	1	1	yes	34931
30	110	P	1	1	yes	303
31	99	P	1	1	yes	13515
32	113	P	1	1	yes	32899
33	163	P	1	1	yes	38171
34	339	P	1	1	yes	38273
35	320	P	1	1	yes	18246
36	288	P	1	1	yes	17769
37	63	P	1	1	yes	18665
38	71	P	1	1	yes	36
39	47	P	1	1	yes	7754
40	126	P	1	1	yes	35969
41	107	P	1	1	yes	10378
42	118	P	1	1	yes	10576
43	109	P	1	1	yes	42270
44	251	P	1	1	yes	27797
45	251	P	1	1	yes	16745
46	228	P	1	1	yes	10210
47	217	P	1	1	yes	24253

48	244	F	1	1	yes	19576
49	256	F	1	1	no	30582

	notes
0	NaN
1	NaN
2	NaN
3	dropped beh and eye tracking frames (5), stim …
4	pupil recording truncated (half missing), drop…
5	dropped beh and eye tracking frames (7), stim …
6	NaN
7	NaN
8	NaN
9	NaN
10	dropped stim frames (17), 2nd stim2twop alignm…
11	NaN
12	NaN
13	NaN
14	NaN
15	ROIs not in retinotopic center
16	ROIs not in retinotopic center
17	ROIs not in retinotopic center
18	NaN
19	NaN
20	NaN
21	NaN
22	NaN
23	NaN
24	NaN
25	NaN
26	NaN
27	NaN
28	NaN
29	NaN
30	dropped stim frames (133)
31	NaN
32	NaN
33	dropped stim frames (126)
34	NaN
35	NaN
36	NaN
37	NaN
38	NaN
39	NaN
40	NaN
41	NaN
42	z-drift (10 um), passed QC after revision

```
43                                              NaN
44                             dropped stim frames (257)
45  stim2twop alignment shifted corrected with 2nd…
46  dropped beh and eye tracking frames (6), stim …
47  FOV shifted (poor alignment with previous sess…
48                                      z-drift (14 um)
49  laser wavelength incorrectly set to 800 nm (in…

[50 rows x 24 columns]
```

1.2 2. Basics of initializing a Session object

Sessions can be intialized with their 9-digit `sessid`:

or with their `mouse_n`, `sess_n` and `runtype`:

1.2.1 Data format is identified automatically

During initialization, the code looks first for the session data in NWB format. If it doesn't find it, it looks for the data in its original format. If neither are found, an error is thrown.

1.2.2 Loading the data after initialization.

After creating the session, you must run `self.extract_info()`. This wasn't amalgamated into the `__init__` to reduce the amount of information needed to just create a session object.

1.2.3 Loading ROI/running/pupil info

You can load this information when you call `self.extract_info()` or manually later by calling `self.load_roi_info()`, `self.load_run_data()` and `self.load_pup_data()`.

```
Loading stimulus and alignment info…
Loading ROI trace info…
Loading running info…
Loading pupil info…
```

1.2.4 Stimulus dataframe

The stimulus dataframe, stored under `sess.stim_df`, details the stimulus feature for each segment of the presentation.
A **segment** is the minimal subdivision of the stimulus presentation: **0.3 sec** for the Gabor stimulus, and **1s** for the visual flow, and grayscreen stimuli.

If a feature **does not apply** to certain segments (e.g., `gabor_number` for visual flow stimulus segments), the values for those segments will be `None`, `NaN` or `[]`, depending on the column's datatype.

Missing columns: Note that a few columns are missing, since the session was loaded with `full_table=False`. * `"gabor_orientations"`: Specific orientation of each Gabor patch, for each segment. * "`square_locations_x`": Specific x location of each visual flow square, at **each frame** of each segment. * "`square_locations_y`": Specific y location of each visual flow square, at **each frame** of each segment.

This is primarily to save memory, when loading a session, as this information is not typically needed. To load all columns, re-run `sess.extract_info()` with `full_table=True`. Data that is already loaded will not be re-loaded.

[10]:

```
     stimulus_type stimulus_template_name  unexpected gabor_frame  \
id
0       grayscreen             grayscreen         NaN
1           gabors                 gabors         0.0           A
2           gabors                 gabors         0.0           B
3           gabors                 gabors         0.0           C
4           gabors                 gabors         0.0           D
…              …                    …         …           …
8839       visflow          visflow_right         0.0
8840       visflow          visflow_right         0.0
8841       visflow          visflow_right         1.0
8842       visflow          visflow_right         1.0
8843    grayscreen             grayscreen         NaN

      gabor_kappa  gabor_mean_orientation  gabor_number  \
id
0             NaN                     NaN           NaN
1            16.0                     0.0          30.0
2            16.0                     0.0          30.0
3            16.0                     0.0          30.0
4            16.0                     0.0          30.0
…           …                       …           …
8839          NaN                     NaN           NaN
8840          NaN                     NaN           NaN
8841          NaN                     NaN           NaN
8842          NaN                     NaN           NaN
8843          NaN                     NaN           NaN

                                        gabor_locations_x  \
id
0                                                      []
1     [-314.2481536790383, 726.6351926350328, -609.4…
2     [278.93714376420894, -895.0169462360316, 830.4…
3     [-694.2565883378384, 458.8415680953749, -472.6…
4     [-631.2261180219028, -600.2310528361336, -887…
…                                                   …
8839                                                   []
8840                                                   []
8841                                                   []
8842                                                   []
```

```
8843                                              []

                                      gabor_locations_y  \
id
0                                                 []
1     [519.3985635606798, 429.54112277826425, 482.75…
2     [-62.92603512701612, -329.96944361291634, -332…
3     [162.5089263895926, 433.50619201931613, 567.71…
4     [-21.003509639097615, -271.4924294875755, 555…
…                                              …
8839                                              []
8840                                              []
8841                                              []
8842                                              []
8843                                              []

                                            gabor_sizes  …  \
id                                                       …
0                                                 []  …
1     [237, 382, 341, 269, 332, 300, 256, 322, 252, …  …
2     [355, 245, 207, 246, 209, 371, 209, 400, 214, …  …
3     [270, 274, 369, 230, 364, 205, 360, 315, 396, …  …
4     [228, 332, 237, 248, 346, 308, 333, 277, 232, …  …
…                                              …   …
8839                                              []  …
8840                                              []  …
8841                                              []  …
8842                                              []  …
8843                                              []  …

     start_frame_stim_template  start_frame_stim  stop_frame_stim  \
id
0                            0                 0             1800
1                            0              1800             1818
2                            1              1818             1836
3                            2              1836             1854
4                            3              1854             1872
…                          …                 …               …
8839                     60960            249960           250020
8840                     61020            250020           250080
8841                     61080            250080           250140
8842                     61140            250140           250200
8843                         0            250200           251999

      num_frames_stim  start_frame_twop  stop_frame_twop  num_frames_twop  \
id
0                1800               143             1046              903
```

```
1                    18             1046             1055               9
2                    18             1055             1064               9
3                    18             1064             1073               9
4                    18             1073             1082               9
…                  …                …                …                …
8839                 60           125551           125581              30
8840                 60           125581           125611              30
8841                 60           125611           125641              30
8842                 60           125641           125672              31
8843               1799           125672           126575             903

      start_time_sec  stop_time_sec  duration_sec
id
0           14.30646      44.332150     30.025690
1           44.33215      44.639380      0.307230
2           44.63938      44.939040      0.299660
3           44.93904      45.232430      0.293390
4           45.23243      45.526750      0.294320
…                 …             …             …
8839      4183.68954    4184.690500      1.000960
8840      4184.69050    4185.691070      1.000570
8841      4185.69107    4186.692190      1.001120
8842      4186.69219    4187.690570      0.998380
8843      4187.69057    4217.673903     29.983333

[8844 rows x 24 columns]
```

1.2.5 Stimulus objects

Once `sess.extract_info()` is run, each Session object also contains `Stim` objects.

These come in one of three subclasses: `Gabors`, `Visflow`, `Grayscr`, and can be accessed with: `sess.stims`, `sess.gabors`, `sess.visflow`, `sess.grayscr`.

The the `Stim` object `stim`, the Session object can be accessed with `stim.sess`.

```
number of rois          : 90
mouse number            : 4
mouse ID                : 411771
gabor object            : Gabors (stimulus from session 760260459)
2p frames per sec       : 30.08
stimulus frames per sec: 59.95
```

1.3 3. Retrieving data of interest

1.3.1 Identifying stimulus segments of interest

From a `Session`'s `Stim`, you can get a list of segments that fit a specific criterion, e.g. **U segments** (unexpected, 3rd Gabor frame).

1.3.2 Identifying frame numbers of interest, to index the data

Then, you can retrieve the exact frame numbers that match these segments.

Specifically, you can access: * `twop` frame numbers, which index the two-photon data and pupil data, and * `stim` frame numbers, which index the running data.

Note: When retrieving the frame numbers, specifying `ch_fl` (check flanks) ensures that only frame numbers whose flanks are within the recording are returned. In other words, any frame number too close to the start or end of the recording (based on `pre`/`post` values), will be dropped.

1.3.3 Retrieving the data of interest

You can now get the **ROI / running / pupil data** corresponding to these reference frames and the specified `pre` / `post` periods (in sec).

1.3.4 Retrieving data statistics of interest

You can also directly obtain statistics on the data of interest.

[15]:
```
datatype                  roi_traces
bad_rois_removed                 yes
scaled                            no
baseline                          no
integrated                       yes
smoothing                         no
fluorescence                     dff
general ROIs sequences
stats    None stat_mean     0.062516
              error_SEM     0.017370
```

1.3.5 Using hierarchical dataframes

Data and statistics are returned in a hierarchical dataframe with **columns** and **indices**.

This has the advantage of allowing metadata to be stored in dummy columns, however extracting data from these dataframes can be tricky, syntactically.

[16]:
```
datatype                      roi_traces
bad_rois_removed                     yes
scaled                               yes
baseline                              no
integrated                            no
smoothing                             no
```

```
fluorescence                          dff
ROIs sequences time_values
0    0          -1.000000    -0.338172
                -0.966102     0.155122
                -0.932203     0.150821
                -0.898305     0.053135
                -0.864407    -0.100729
…                                …
89   81          0.864407     0.297796
                 0.898305    -0.031714
                 0.932203     0.339743
                 0.966102     0.524661
                 1.000000    -0.289968

[442800 rows x 1 columns]
```

To **extract a numpy array** with the correct dimensions from a hierarchical dataframe, you can use the following utility function: `gen_util.reshape_df_data()`.

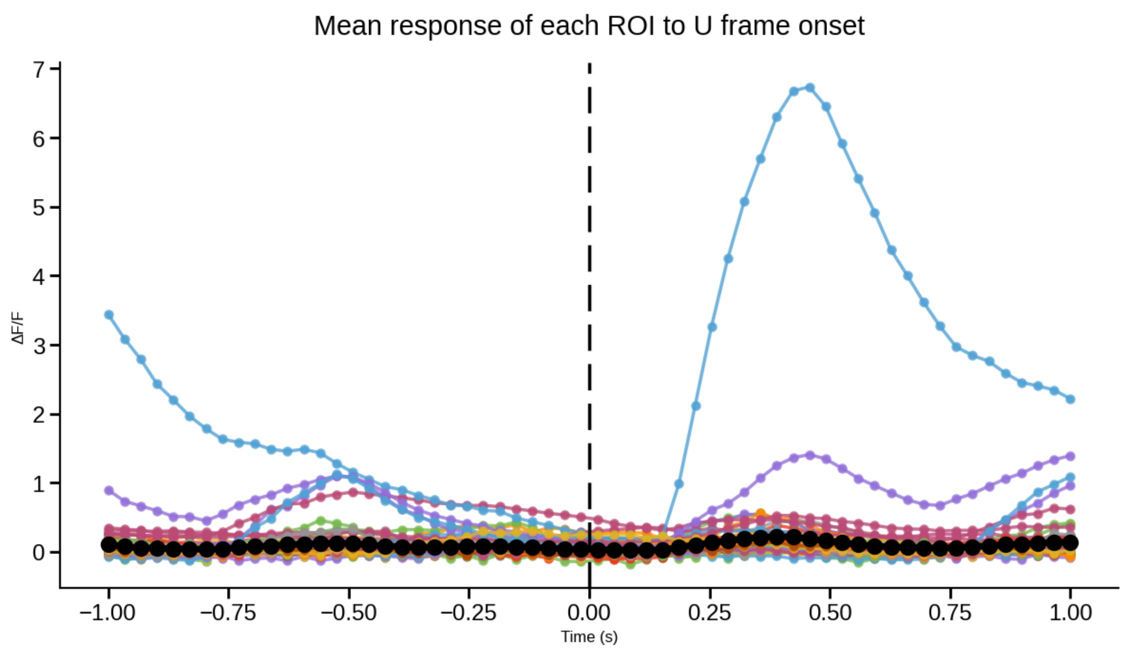
Here, each index level, then column level is turned into a new axis, **i.e. ROIs x sequences x time_values** (In this case, `squeeze_cols` is set to True to prevent each dummy column from becoming its own axis.)

```
ROI data shape: 90 ROIs x 82 sequences x 60 time values
```

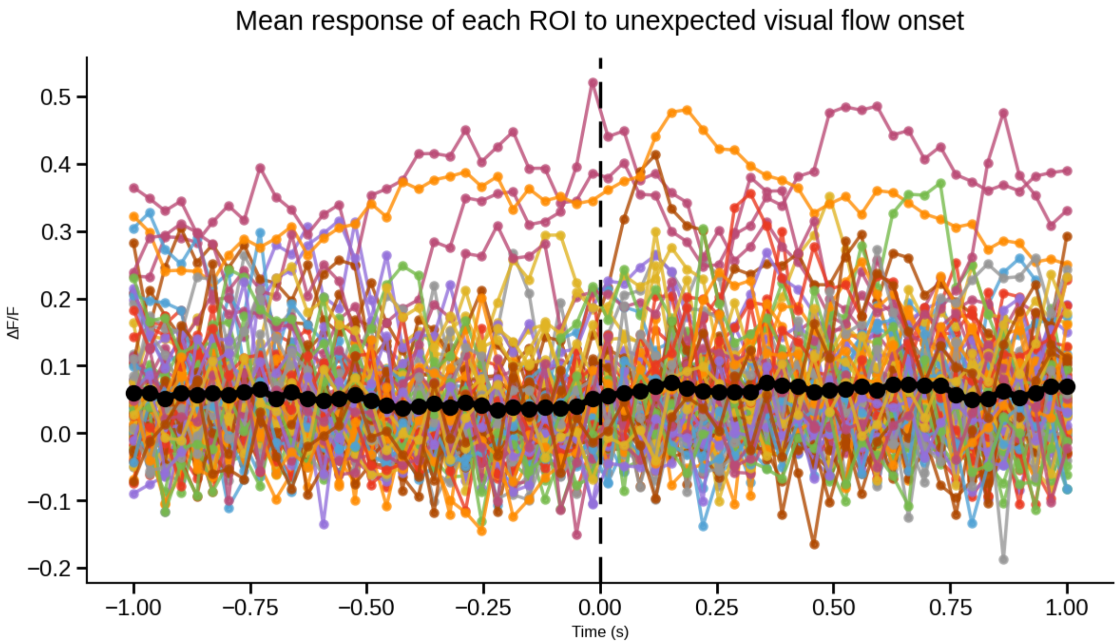
You can also retrieve the time stamps for each frame.

1.3.6 Visualizing the data

Finally, we can plot each ROIs mean activity across Gabor sequences, as well as a mean across ROIs.



1.3.7 The same steps apply to analysing the visual flow stimulus



1.4 4. Tracked ROIs

ROI tracking was performed on the `production` data.
At any point, it is possible to **restrict the data returned** to only the tracked ROIs, called `sess.set_only_tracked_rois(True)`.

Here, we retrieve the data, **integrated over each sequence**.

The dataframe returned contains data only for tracked ROIs.

[23]:
```
datatype          roi_traces
bad_rois_removed         yes
scaled                   yes
baseline                  no
integrated               yes
smoothing                 no
fluorescence             dff
ROIs sequences
27   0              0.101591
     1              0.188843
     2             -0.072082
```

```
     3          0.685275
     4          0.033439
…                  …
21   27         0.026887
     28         0.523182
     29        -0.039192
     30        -0.059955
     31         0.080472

[1504 rows x 1 columns]
```

1.4.1 Extracting tracked ROI data *correctly* (!)

Importantly, the ROIs are now sorted in their tracking order, which ensures that they are correctly aligned across sessions.
As a result, the **"ROIs" index** may no longer be in increasing order, like in this example.

```
ROI numbers, ordered for tracking:
27, 60, 81, 2, 49, 6, 4, 64, 88, 15, 84, 76, 75, 16, 14, 17, 28, 25, 12, 45, 1,
65, 24, 78, 63, 79, 80, 67, 0, 69, 58, 51, 54, 46, 39, 38, 40, 13, 31, 5, 42,
43, 86, 41, 26, 33, 21
```

Note that these **ROI numbers** are assigned for each session separately. Thus, e.g., ROI #28 from session 1 is most likely not the same neuron/dendrite as the ROI #28 from session 2, for the same mouse. For this reason, **the tracking ordering** is important: it allows ROI traces from different sessions to be lined up correctly with one another.

To **ensure that the tracked ROI order is preserved** when extracting the data, the safest option is to use the utility function introduced above, i.e. `gen_util.reshape_data_df()`. It will ensure that the order is preserved.

```
Tracked ROI data shape using the correct method, i.e.,
gen_util.reshape_df_data()
47 ROIs x 32 sequences
```

Do not use the `.unstack()` method for hierarchical dataframes!

Even though the `.unstack()` method is typically a convenient way to extract a 2D array from a hierarchical dataframe, it will cause major problems here. Specifically, `.unstack()` internally triggers a resorting of the hierarchical indices. Thus, using it will completely mess up the tracked ROI order.

```
Tracked ROI data shape using the wrong method, i.e., .unstack()
47 ROIs x 32 sequences
```

As you can see, the dimensions are still correct. However, the **ROI sorting is actually lost!**

For example, **ROI #5**, which should appear at index 6 in the array, is now at index 3.

```
Data for the tracked ROI at index 6, when using the correct method: i.e.,
gen_util.reshape_df_data()
0.005, 0.024, -0.050, -0.553, 0.629, -0.051, -0.057, 0.071, -0.003, -0.067 …
```

```
Data for the tracked ROI at index 6, when using the wrong method: i.e.,
.unstack()
0.053, 0.191, 0.232, -0.004, -0.008, 0.060, 0.083, -0.031, 0.015, 0.024 …
```

```
Data for the tracked ROI that should be at index 6 is instead at index 3,
when using the wrong method: i.e., .unstack()
0.005, 0.024, -0.050, -0.553, 0.629, -0.051, -0.057, 0.071, -0.003, -0.067 …
```

1.4.2 Reset the session to start using all ROIs, again

1.5 5. Additional tips on indexing a hierarchical dataframe

[31]:
```
scaled                          yes
baseline                         no
integrated                       no
smoothing                        no
fluorescence                    dff
ROIs sequences time_values
0    1         -1.0       -0.183646
     20        -1.0        0.013693
     21        -1.0       -0.091127
3    1         -1.0        0.221201
     20        -1.0        0.347209
     21        -1.0       -0.163844
5    1         -1.0       -0.219216
     20        -1.0        0.047849
     21        -1.0        0.031890
```

1.6 6. Retrieving several Session objects, based on criteria

1.6.1 Retrieving mouse / session numbers and IDs that fit specific criteria

`sess_gen_util.get_sess_vals()` can be used to retrieve information for sessions that meet certain criteria.
e.g., **session number 1, 2 or 3**, **production**, **dendritic plane**

```
mouse  1: 758519303 (session 1)
mouse  1: 759189643 (session 2)
mouse  1: 759660390 (session 3)
mouse  3: 761624763 (session 1)
mouse  3: 761944562 (session 2)
mouse  3: 762250376 (session 3)
mouse  4: 760260459 (session 1)
mouse  4: 760659782 (session 2)
mouse  4: 761269197 (session 3)
mouse  7: 777496949 (session 1)
```

```
mouse  7: 778374308 (session 2)
mouse  7: 779152062 (session 3)
mouse 12: 826659257 (session 1)
mouse 12: 827300090 (session 2)
mouse 12: 828475005 (session 3)
mouse 13: 832883243 (session 1)
mouse 13: 833704570 (session 2)
mouse 13: 834403597 (session 3)
```

1.6.2 Loading the sessions

`sess_gen_util.init_sessions()` can be used to **initialize the sessions** and **extract the requested data**.

```
Creating session 758519303…
```

```
/home/colleen/Documents/PhD/OpenScope_CA_Analysis/analysis/session.py:236:
UserWarning:
Several NWB files were found for this session. When loading data, the first file
listed that contains the required data will be used.
```

```
Loading stimulus and alignment info…
Loading ROI trace info…
Loading running info…
Finished creating session 758519303.

Creating session 759189643…
Loading stimulus and alignment info…
Loading ROI trace info…
Loading running info…
Finished creating session 759189643.

Creating session 759660390…
Loading stimulus and alignment info…
Loading ROI trace info…
Loading running info…
Finished creating session 759660390.

Creating session 764704289…
Loading stimulus and alignment info…
Loading ROI trace info…
Loading running info…
Finished creating session 764704289.
```

1.6.3 Using the loaded sessions

Now, one can run through the sessions, and run whatever analysis is needed.

Note here that, when calling `stim.get_segs_by_criteria()`, **features that do not apply to**

the stimulus (e.g., `gabfr` for the `visflow` stimulus) **are simply ignored**.

```
Session ID: 758519303 (mouse 1, session 1)
    visflow: 31 sequences
    gabors: 94 sequences
Session ID: 759189643 (mouse 1, session 2)
    visflow: 34 sequences
    gabors: 90 sequences
Session ID: 759660390 (mouse 1, session 3)
    visflow: 33 sequences
    gabors: 105 sequences
Session ID: 764704289 (mouse 6, session 1)
    visflow: 33 sequences
    gabors: 96 sequences
```

1.7 7. Retrieving ROI masks from a session

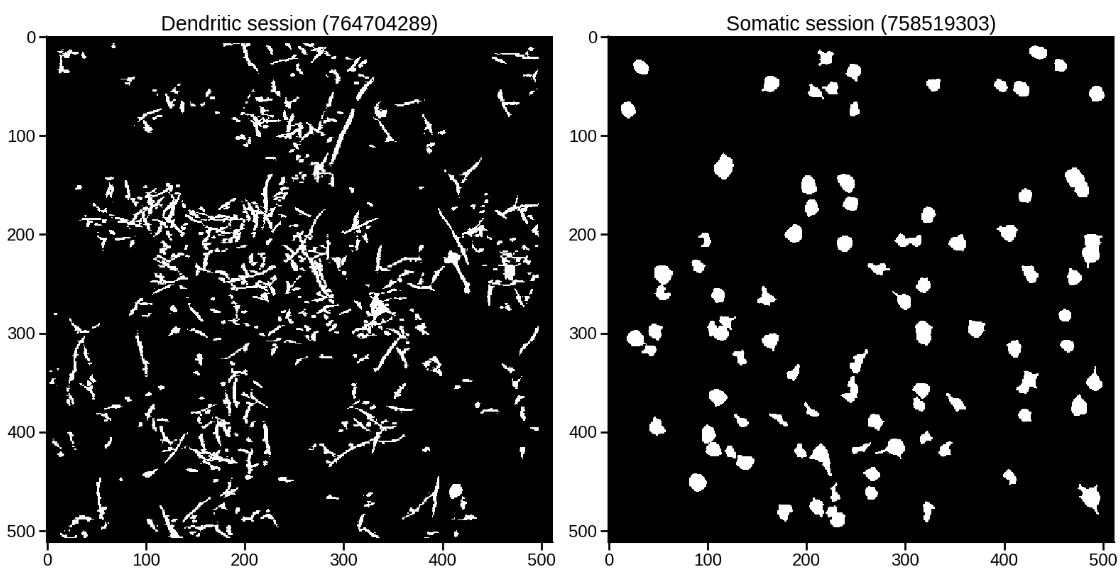
1.7.1 Loading masks

Boolean ROI masks can be obtained for each Session. Masks can be loaded as follows, with dimensions: **ROI x height x width**, retrieving only masks for ROIs that are valid (when evaluated by their dF/F traces).

Notes: - **If sessions are set to use only tracked ROIs**, as described above, only masks for the tracked ROIs (sorted in the tracking order) will be returned. - If running this notebook in **Binder**, the dendritic masks will not loaded, as the memory requirements are too high (~2-3GB).

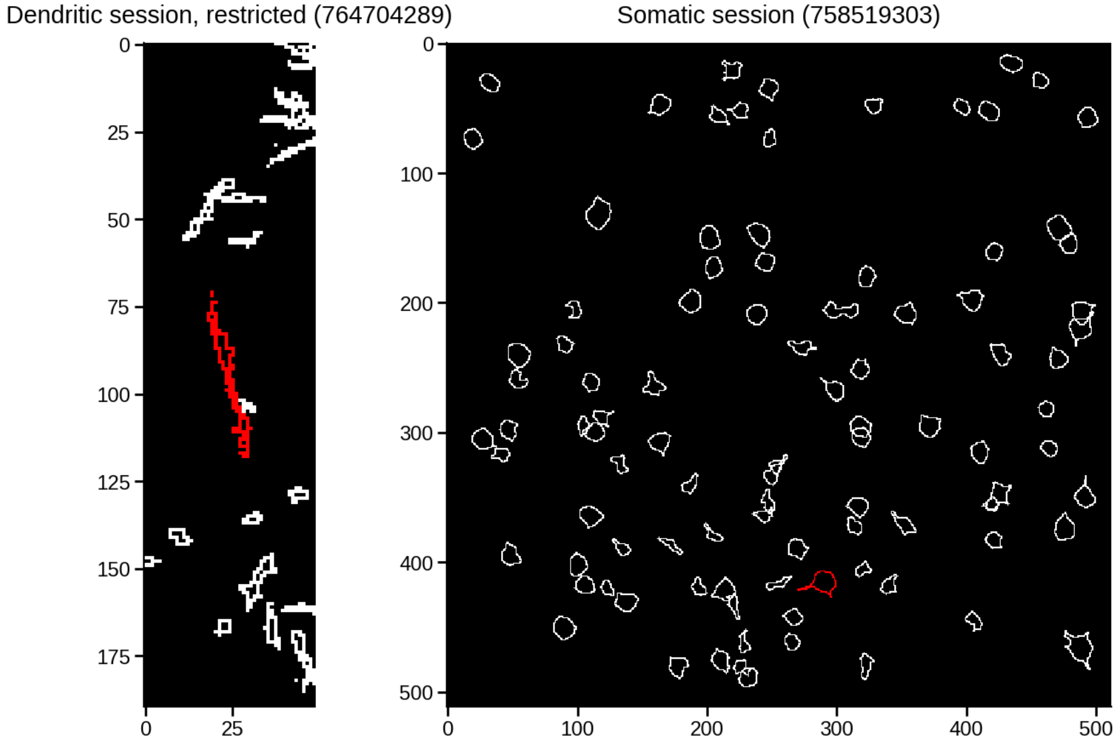
1.7.2 Visualizing ROI masks

`sess_plot_util.plot_ROIs()` can be used to visualize ROIs.



1.7.3 Visualizing ROI mask contours

`sess_plot_util.plot_ROI_contours()` can be used to visualize ROI contours, optionally restricted to around an ROI of interest.



1.8 8. Visualizing stimulus templates

One should note that different NWB versions are available for each session, on the Dandi archive.

The basic versions are the smallest ones (~130 MB to 1.7 GB each), and contain all the data needed for most analyses. In contrast, the versions with `+image` in the name also contain the stimulus templates, i.e. all unique stimulus frame images. They are typically ~1.5 GB larger than the corresponding basic versions.

We can load an example session: mouse 1, session 1, downloaded from the Dandi archive: sub-408021_ses-758519303_behavior+ophys.nwb.

```
Be sure to download the file, and place it in the `datadir` directory:
../../data/OSCA_NWB.
```

```
/home/colleen/Documents/PhD/OpenScope_CA_Analysis/analysis/session.py:236:
UserWarning:
```

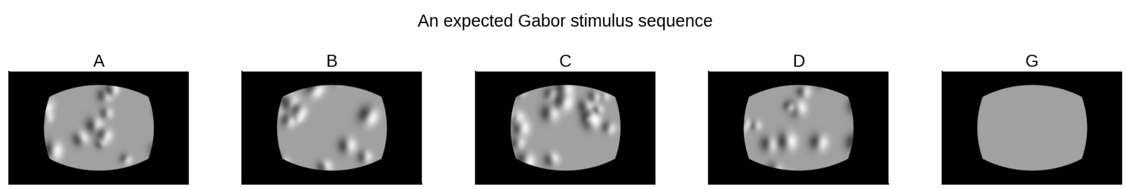
```
Several NWB files were found for this session. When loading data, the first file
listed that contains the required data will be used.
```

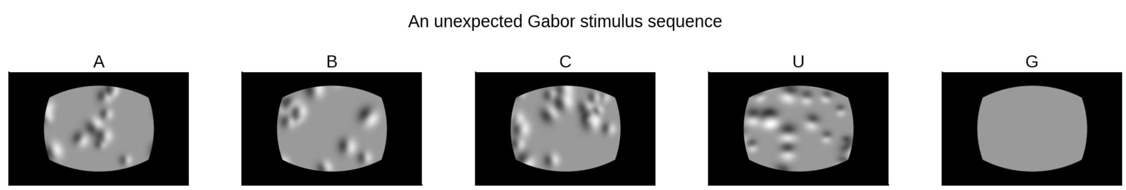
```
Loading stimulus and alignment info…
```

As the warning indicates, the `Session` object has found both the basic version of the data for this session, and the version that also contains the stimulus template (`+image`) in the specified data directory. At any step where data must be loaded, the `Session` object will load it from the first listed version (alphabetically) that contains the required data.

1.8.1 Gabor sequence images

We can now identify the frame numbers for the **first Gabor sequence**, and **visualize** the corresponding stimulus images.





As we can see, whereas the Gabor patch orientations are consistent across frames in the expected sequence, they are rotated by 90° in the U frame of the unexpected sequence.

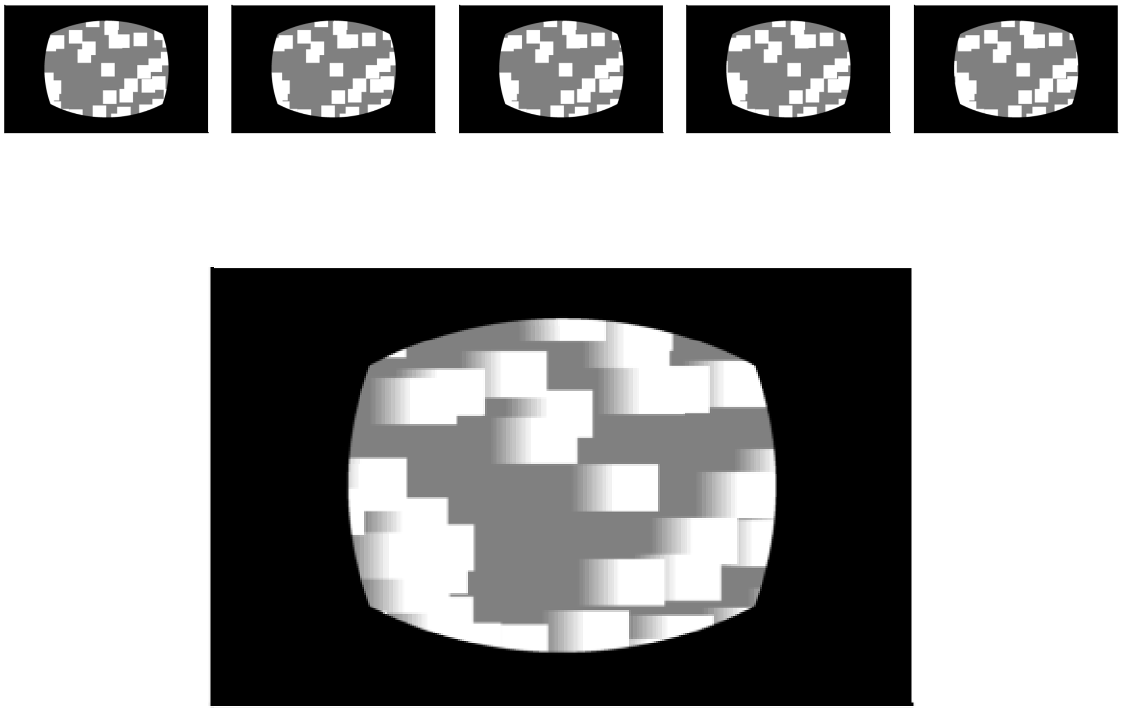
1.8.2 Warping

Note that the periphery of the images is masked in black. This is because, during the actual stimulus presentation, the images were presented on a **flat screen, and spherically warped**. This ensured that the apparent properties of the stimuli (size, speed, spatial frequency, etc.) were constant across the monitor, as seen from the mice's perspectives. The black masks overlayed on the unwarped stimuli stored in the NWB file, therefore, **mask the parts of the stimuli that were** outside the edges of the screen, due to warping, and thus **not visible** to the mice during the experiments.

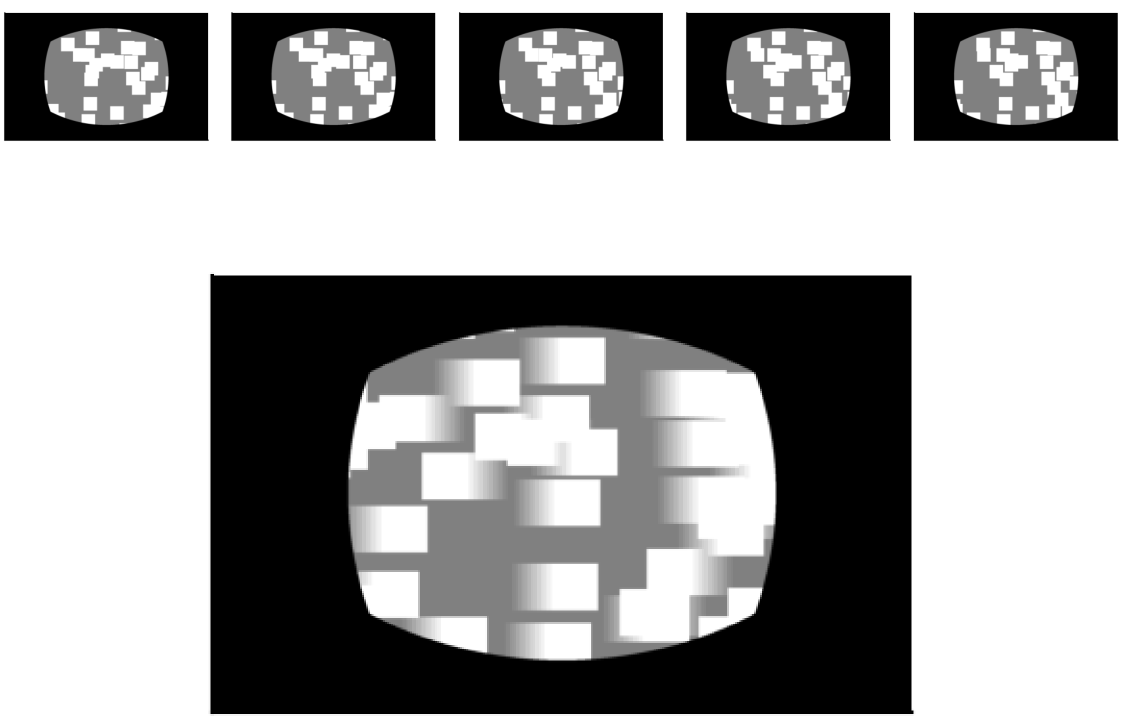
1.8.3 Visual flow sequence images

We can also visualize the **visual flow stimulus**. It is important to note that, whereas the Gabor images are static for each segment, the visual flow stimulus is in motion, and therefore changes at each frame. For this reason, we will simply identify the first visual flow segment in a sequence, and visualize the first few frames in that follow it.

An expected visual flow sequence



An unexpected visual flow sequence



We plot the first few frames in each sequence separately. We then also plot all the frames retained for each sequence, **overlayed in a graded way**, in order to visualize the squares **in motion**.

As we can see, in the **expected** sequence, all of the squares are moving rightward, uniformly. In contrast, in the **unexpected** sequence, although most squares are still moving rightward, ~25% of them are moving in the opposite direction, i.e., leftward.

1.8.4 Stimulus generating code and examples

This repository contains the code to generate these stimuli, as well as some example videos.

1.9 9. Last notes

There is much more to the codebase, and even to the `Session` and `Stim` objects, and almost all functions and methods are thoroughly documented.

When looking to implement a new analysis, consider checking to see whether relevant functions have already been implemented in:
* `analysis/session.py`
* `analysis/basic_analys.py`
* `sess_util/sess_gen_util.py`

1.9.1 Methods/properties attached to `Session` and `Stim` objects.

```
Loading stimulus and alignment info…
Loading ROI trace info…
Loading running info…
Loading pupil info…
```

```
Session (758519303)

  Public properties:
    self.DOB
    self.age_weeks
    self.all_files
    self.any_files
    self.date
    self.dend
    self.depth
    self.drop_tol
    self.gabors
    self.grayscr
    self.home
    self.line
    self.max_proj
    self.mouse_df
    self.mouse_n
    self.mouseid
    self.n_stims
```

```
  self.notes
  self.only_tracked_rois
  self.pass_fail
  self.plane
  self.pup_data
  self.roi_facts_df
  self.roi_masks
  self.roi_names
  self.run_data
  self.runtype
  self.sess_files
  self.sess_n
  self.sessid
  self.sex
  self.stim2twopfr
  self.stim_df
  self.stim_fps
  self.stim_seed
  self.stims
  self.stimtypes
  self.tot_stim_fr
  self.tot_twop_fr
  self.tracked_rois
  self.twop2stimfr
  self.twop_fps
  self.visflow

Public methods:
  self.check_flanks()
  self.convert_frames()
  self.data_loaded()
  self.extract_info()
  self.get_active_rois()
  self.get_bad_rois()
  self.get_fr_ran()
  self.get_frames_timestamps()
  self.get_local_nway_match_path()
  self.get_nrois()
  self.get_plateau_roi_traces()
  self.get_pup_data()
  self.get_registered_max_proj()
  self.get_registered_roi_masks()
  self.get_roi_masks()
  self.get_roi_seqs()
  self.get_roi_traces()
  self.get_roi_traces_by_ns()
  self.get_run_velocity()
  self.get_run_velocity_by_fr()
```

```
    self.get_single_roi_trace()
    self.get_stim()
    self.load_pup_data()
    self.load_roi_info()
    self.load_run_data()
    self.set_only_tracked_rois()

Gabors (stimulus from session 758519303)

  Public properties:
    self.all_gabfr
    self.all_gabfr_mean_oris
    self.block_params
    self.deg_per_pix
    self.exp_gabfr
    self.exp_gabfr_mean_oris
    self.exp_max_s
    self.exp_min_s
    self.kappas
    self.n_patches
    self.n_segs_per_seq
    self.ori_ran
    self.phase
    self.seg_len_s
    self.sess
    self.sf
    self.size_ran
    self.stim_fps
    self.stim_params
    self.stimtype
    self.unexp_gabfr
    self.unexp_gabfr_mean_oris
    self.unexp_max_s
    self.unexp_min_s
    self.win_size

  Public methods:
    self.get_A_frame_1s()
    self.get_A_segs()
    self.get_all_unexp_segs()
    self.get_all_unexp_stim_fr()
    self.get_fr_by_seg()
    self.get_frames_by_criteria()
    self.get_n_fr_by_seg()
    self.get_pup_diam_data()
    self.get_pup_diam_stats_df()
    self.get_roi_data()
    self.get_roi_stats_df()
```

```
    self.get_run()
    self.get_run_data()
    self.get_run_stats_df()
    self.get_segs_by_criteria()
    self.get_segs_by_frame()
    self.get_start_unexp_segs()
    self.get_start_unexp_stim_fr_trans()
    self.get_stats_df()
    self.get_stim_beh_sub_df()
    self.get_stim_df_by_criteria()
    self.get_stim_images_by_frame()
    self.get_stim_par_by_frame()
    self.get_stim_par_by_seg()

Visflow (stimulus from session 758519303)

  Public properties:
    self.block_params
    self.deg_per_pix
    self.exp_max_s
    self.exp_min_s
    self.main_flow_direcs
    self.n_squares
    self.prop_flipped
    self.seg_len_s
    self.sess
    self.speed
    self.square_sizes
    self.stim_fps
    self.stim_params
    self.stimtype
    self.unexp_max_s
    self.unexp_min_s
    self.win_size

  Public methods:
    self.get_all_unexp_segs()
    self.get_all_unexp_stim_fr()
    self.get_dir_segs_exp()
    self.get_fr_by_seg()
    self.get_frames_by_criteria()
    self.get_n_fr_by_seg()
    self.get_pup_diam_data()
    self.get_pup_diam_stats_df()
    self.get_roi_data()
    self.get_roi_stats_df()
    self.get_run()
    self.get_run_data()
```

```
    self.get_run_stats_df()
    self.get_segs_by_criteria()
    self.get_segs_by_frame()
    self.get_start_unexp_segs()
    self.get_start_unexp_stim_fr_trans()
    self.get_stats_df()
    self.get_stim_beh_sub_df()
    self.get_stim_df_by_criteria()
    self.get_stim_images_by_frame()
```

```
Grayscr (session 758519303)

  Public properties:
    self.sess
    self.stimtype

  Public methods:
    self.get_all_fr()
    self.get_start_fr()
    self.get_stim_images_by_frame()
    self.get_stop_fr()
```

1.9.2 Example `Session` and `Stim` object property values.

Properties with long values (e.g., long dataframes, arrays, lists, strings) are omitted, for brevity.

```
Session (758519303)

  Public property values:
    self.DOB: 20180623
    self.age_weeks: 13.5714285714286
    self.all_files: True
    self.any_files: True
    self.date: 20180926
    self.dend: extr
    self.depth: 175
    self.drop_tol: 0.0003
    self.gabors: Gabors (stimulus from session 758519303)
    self.grayscr: Grayscr (session 758519303)
    self.line: L23-Cux2
    self.mouse_n: 1
    self.mouseid: 408021
    self.n_stims: 2
    self.notes: nan
    self.only_tracked_rois: False
    self.pass_fail: P
    self.plane: soma
    self.runtype: prod
    self.sess_n: 1
```

```
    self.sessid: 758519303
    self.sex: M
    self.stim_fps: 59.951703429774675
    self.stim_seed: 30587
    self.stimtypes: ['gabors', 'visflow']
    self.tot_stim_fr: 251999
    self.tot_twop_fr: 126741
    self.twop2stimfr: [nan nan nan … nan nan nan]
    self.twop_fps: 30.078983328254086

Gabors (stimulus from session 758519303)

  Public property values:
    self.all_gabfr: ['A', 'B', 'C', 'D', 'G', 'U']
    self.all_gabfr_mean_oris: [0.0, 45.0, 90.0, 135.0, 180.0, 225.0]
    self.deg_per_pix: 0.06251912565744862
    self.exp_gabfr: ['A', 'B', 'C', 'D', 'G']
    self.exp_gabfr_mean_oris: [0.0, 45.0, 90.0, 135.0]
    self.exp_max_s: 90
    self.exp_min_s: 30
    self.kappas: [16]
    self.n_patches: 30
    self.n_segs_per_seq: 5
    self.ori_ran: [0, 360]
    self.phase: 0.25
    self.seg_len_s: 0.3
    self.sess: Session (758519303)
    self.sf: 0.04
    self.size_ran: [204, 408]
    self.stim_fps: 59.951703429774675
    self.stim_params: ['gabor_kappa']
    self.stimtype: gabors
    self.unexp_gabfr: ['U']
    self.unexp_gabfr_mean_oris: [90.0, 135.0, 180.0, 225.0]
    self.unexp_max_s: 6
    self.unexp_min_s: 3
    self.win_size: [1920, 1200]

Visflow (stimulus from session 758519303)

  Public property values:
    self.deg_per_pix: 0.06251912565744862
    self.exp_max_s: 90
    self.exp_min_s: 30
    self.main_flow_direcs: ['left (nasal)', 'right (temp)']
    self.n_squares: [105]
    self.prop_flipped: 0.25
    self.seg_len_s: 1
    self.sess: Session (758519303)
```

```
    self.speed: 799.7552664756905
    self.square_sizes: [128]
    self.stim_fps: 59.951703429774675
    self.stimtype: visflow
    self.unexp_max_s: 4
    self.unexp_min_s: 2
    self.win_size: [1920, 1200]

Grayscr (session 758519303)

  Public property values:
    self.sess: Session (758519303)
    self.stimtype: grayscreen
```