session_demonstration_script

May 19, 2021

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:

- These packages should be present if installing the conda environment from `osca.yml`.
- `util` is a Github repo of mine, and the correct branch `osca_mult` is automatically installed from `osca.yml`. 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`. Though I will try to avoid this, consider updating the utility under those circumstances.

1.1 Set paths to main data directory and the mouse dataframe

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

1.2 1. Basics of initializing a Session object

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

1.2.1 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…
Creating stimulus objects…
Loading ROI trace info…
    WARNING: 3 noisy ROIs (mean below 0 or median above midrange) are also
included in the NaN ROI attributes (but not set to NaN): 244, 298, 305.
Loading running info…
    WARNING: 211 dropped running frames (~0.1%) (in pre-processing).
Loading pupil info…
```

[5]:

	stimType	stimPar1	stimPar2	surp	stimSeg	gabfr	start2pfr	end2pfr
0	-1	-1	-1	-1	-1	-1	143	143
1	g	135	16	0	0	0	1046	1055
2	g	135	16	0	1	1	1055	1064
3	g	135	16	0	2	2	1064	1073
4	g	135	16	0	3	3	1073	1082
…	…	…	…	…	…	…	…	…
7476	b	128	right (temp)	0	2035	-1	125522	125552
7477	b	128	right (temp)	0	2036	-1	125552	125582
7478	b	128	right (temp)	1	2037	-1	125582	125612
7479	b	128	right (temp)	1	2038	-1	125612	125642
7480	b	128	right (temp)	1	2039	-1	125642	125672

	num2pfr	display_sequence_n	block_n
0	0	NaN	NaN
1	9	0.0	0.0
2	9	0.0	0.0
3	9	0.0	0.0
4	9	0.0	0.0
…	…	…	…
7476	30	1.0	1.0
7477	30	1.0	1.0
7478	30	1.0	1.0
7479	30	1.0	1.0
7480	30	1.0	1.0

[7481 rows x 11 columns]

1.2.2 Some information contained in the session object

Note: `Stim` objects (subclasses: `Gabors`, `Bricks`, `Grayscr`) are a separate class from `Session` objects. However, each can by accessed from the other using: - from `Session objects`: `self.stims`, `self.gabors`, `self.bricks` - from `Stim objects`: `self.sess`

```
number of rois: 644
mouse number: 6
mouse ID: 413663
gabor object: Gabors (stimulus 0 of session 764704289)
gabor orientation standard deviation(s): 0.25
2p frames per sec: 30.08
pupil frames per sec: 30.00
stimulus frames per sec: 60.00
```

1.3 2. Identifying segments of interest

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

Then, you can access the frame numbers.

Note: 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 of end of the recording (based on `pre`/`post` values), will be dropped.

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

You can also directly obtain statistics on the data of interest

[11]:
```
datatype                roi_traces
nan_rois_removed               yes
scaled                          no
baseline                        no
integrated                     yes
smoothing                       no
fluorescence                   dff
general ROIs sequences
stats   None stat_mean    0.026752
             error_SEM    0.000911
```

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, syntaxically.

[12]:
```
datatype                    roi_traces
nan_rois_removed                   yes
scaled                             yes
baseline                            no
integrated                          no
smoothing                           no
fluorescence                       dff
ROIs sequences time_values
0    0         -1.000000    -0.009556
               -0.966102    -0.644810
               -0.932203    -0.214521
               -0.898305    -0.116127
               -0.864407    -0.318214
…                                  …
643  95         0.864407     0.050568
                0.898305     0.445153
                0.932203     0.108850
                0.966102     0.116475
                1.000000     0.213779

[3617280 rows x 1 columns]
```

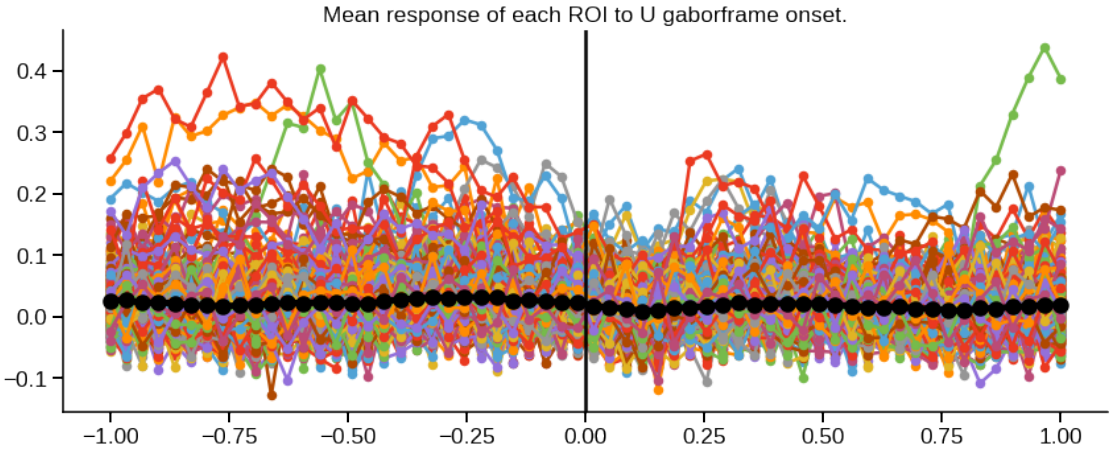
To **extract a numpy array** with the correct dimensions from a hierarchical dataframe, you can use the following utility.

Here, each index level, then column level becomes an 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 an axis.)

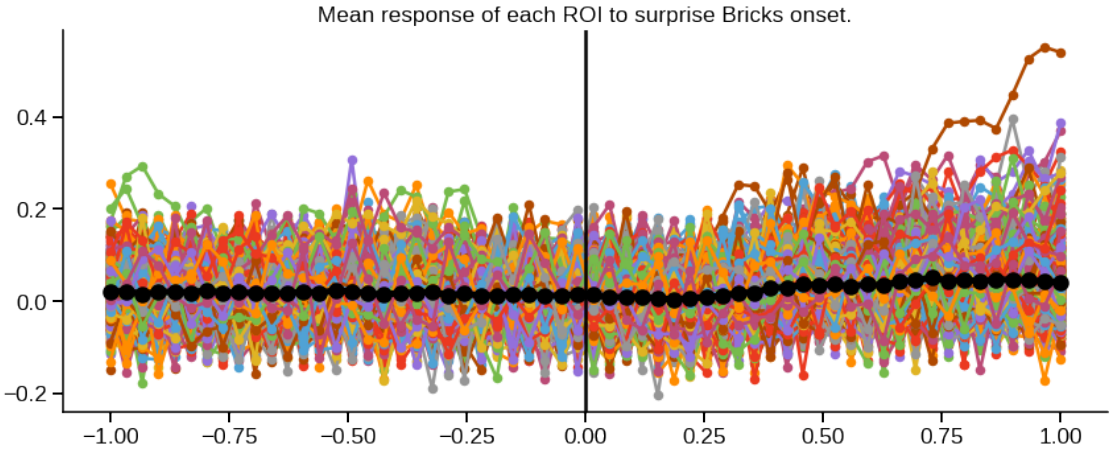
```
ROI data shape: 628 ROIs x 96 sequences x 60 time values
```

You can also retrieve the time stamps for each frame.

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



1.3.1 The same steps apply for Bricks



1.4 3. Additional tips on indexing a hierarchical dataframe

[17]:
```
scaled                          yes
baseline                         no
integrated                       no
smoothing                        no
```

```
fluorescence                    dff
ROIs sequences time_values
0    0         -1.000000    0.045433
               -0.966102   -0.303140
               -0.932203   -0.301464
               -0.898305   -0.219794
               -0.864407    0.065587
…                              …
643  4          0.864407   -0.513739
                0.898305   -0.143983
                0.932203    0.000693
                0.966102    0.140502
                1.000000    0.283351

[113040 rows x 1 columns]
```

1.4.1 4. Retrieving several Session objects, based on criteria

This function keeps track of which Sessions or Mice must be left out (e.g., due to a problem with the session data or the mouse didn't see the stimulus of interest <- the latter only comes up with pilot data).

You can now retrieve the mouse number, session number and ID that fit specific the criteria,

e.g., **session number 1, 2 or 3**, **production**, **dendritic plane**

```
WARNING: Sorted and unique will be set to False as multiple labels are
requested.
```

```
mouse 6: 764704289 (session 1)
mouse 6: 765193831 (session 2)
mouse 6: 766502238 (session 3)
mouse 8: 777914830 (session 1)
mouse 8: 778864809 (session 2)
mouse 8: 779650018 (session 3)
mouse 9: 826187862 (session 1)
mouse 9: 826773996 (session 2)
mouse 9: 827833392 (session 3)
mouse 10: 826338612 (session 1)
mouse 10: 826819032 (session 2)
mouse 10: 828816509 (session 3)
mouse 11: 823453391 (session 1)
mouse 11: 824434038 (session 2)
mouse 11: 825180479 (session 3)
```

You can now **initialize the Sessions** using this function which does the additional extraction steps automatically.

```
Creating session 764704289…
```

```
Loading stimulus and alignment info…
Creating stimulus objects…
Loading ROI trace info…
    WARNING: 3 noisy ROIs (mean below 0 or median above midrange) are also
included in the NaN ROI attributes (but not set to NaN): 244, 298, 305.
Loading running info…
    WARNING: 211 dropped running frames (~0.1%) (in pre-processing).
Finished creating session 764704289.

Creating session 765193831…
Loading stimulus and alignment info…
Creating stimulus objects…
Loading ROI trace info…
    WARNING: 4 noisy ROIs (mean below 0 or median above midrange) are also
included in the NaN ROI attributes (but not set to NaN): 3, 63, 88, 134.
Loading running info…
    WARNING: 345 dropped running frames (~0.1%) (in pre-processing).
Finished creating session 765193831.

Creating session 766502238…
Loading stimulus and alignment info…
Creating stimulus objects…
Loading ROI trace info…
    WARNING: 4 noisy ROIs (mean below 0 or median above midrange) are also
included in the NaN ROI attributes (but not set to NaN): 18, 114, 136, 240.
Loading running info…
    WARNING: 387 dropped running frames (~0.2%) (in pre-processing).
Finished creating session 766502238.

Creating session 777914830…
Loading stimulus and alignment info…
Creating stimulus objects…
Loading ROI trace info…
    WARNING: 1 noisy ROIs (mean below 0 or median above midrange) are also
included in the NaN ROI attributes (but not set to NaN): 45.
Loading running info…
    WARNING: 381 dropped running frames (~0.2%) (in pre-processing).
Finished creating session 777914830.

Creating session 778864809…
Loading stimulus and alignment info…
Creating stimulus objects…
Loading ROI trace info…
Loading running info…
    WARNING: 630 dropped running frames (~0.3%) (in pre-processing).
Finished creating session 778864809.

Creating session 758519303…
```

```
Loading stimulus and alignment info…
Creating stimulus objects…
Loading ROI trace info…
Loading running info…
    WARNING: 175 dropped running frames (~0.1%) (in pre-processing).
Finished creating session 758519303.
```

Then run through the sessions and do whatever with them.

```
Session ID: 764704289 (mouse 6, session 1)
    bricks: 33 sequences
    gabors: 96 sequences
Session ID: 765193831 (mouse 6, session 2)
    bricks: 34 sequences
    gabors: 98 sequences
Session ID: 766502238 (mouse 6, session 3)
    bricks: 29 sequences
    gabors: 94 sequences
Session ID: 777914830 (mouse 8, session 1)
    bricks: 32 sequences
    gabors: 83 sequences
Session ID: 778864809 (mouse 8, session 2)
    bricks: 29 sequences
    gabors: 88 sequences
Session ID: 758519303 (mouse 1, session 1)
    bricks: 31 sequences
    gabors: 94 sequences
```

1.4.2 5. Retrieving ROI masks from session.

Boolean ROI masks can be obtained for Session.

For **dendritic sessions**, the Session is built to assume that `EXTRACT` (not `allen`) ROI data is to be used. This can be checked by checking `self.dend`. As long as `self.dend` is properly set, the correct masks will be loaded.

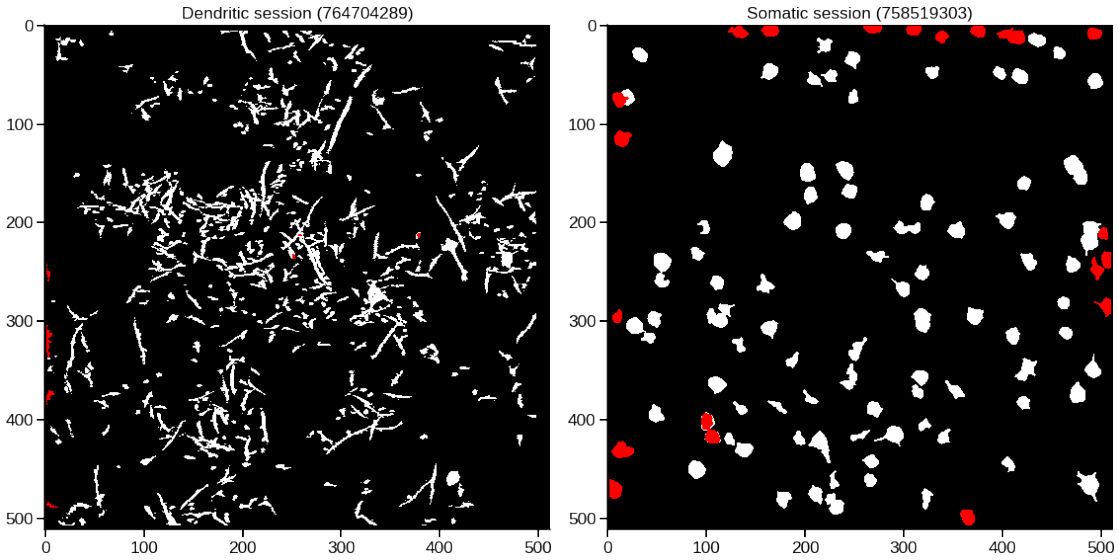
```
Dendritic session, ROI type: extr
Somatic session, ROI type: allen
```

Masks can be loaded as follows, with dimensions: **ROI x height x width**, retrieving only masks for ROIs that are valid for dF/F traces.

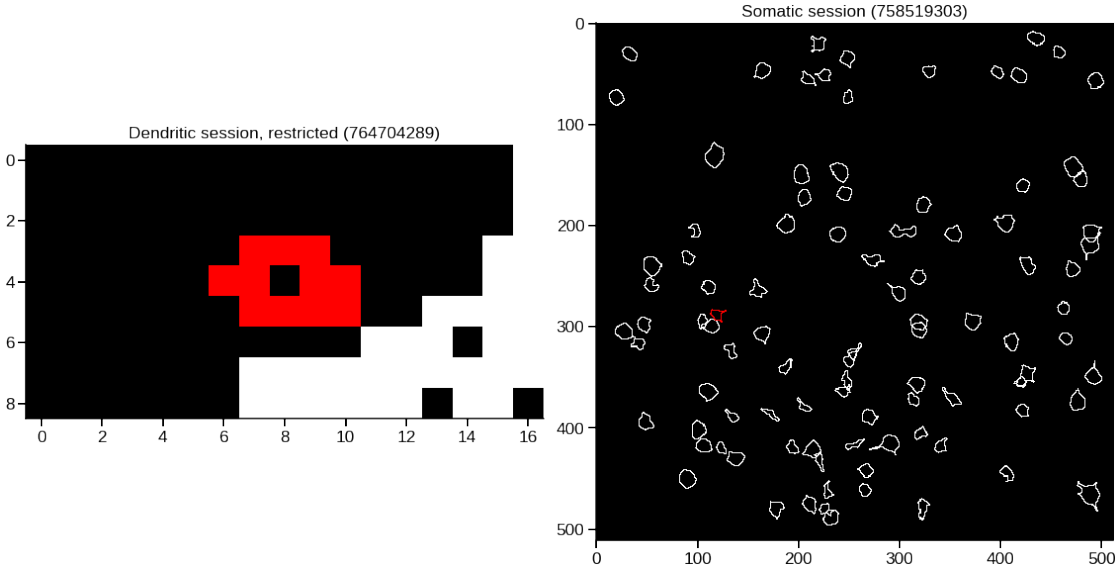
Or if all masks are needed, the attributes can simply be retrieved.

One way to check which ROIs are not valid, is using `self.get_nanrois()`

This is a tool to visualize ROIs, where specific ROIs can be set to red using a `valid_mask`.



This is a tool to visualize ROI contours, optionally localized around an ROI of interest.



1.4.3 6. Last notes

List of the methods/attributes attached to `Session` and `Stim` objects.

`Session: _align_df_loaded, _dend, _get_roi_facts, _init_directory, _init_roi_facts_df, _load_align_df, _load_stim_dict, _load_stims, _load_sync_h5_data, _roi_masks, _set_dend_type, _set_matched_rois, _set_nanrois, _set_roi_attributes, _stim_dict_loaded, _sync_h5_loaded, align_pkl, all_files, any_files, behav_video_h5, bricks, check_flanks, correct_data_h5, date, dend,`

depth, dir, drop_stim_fr, drop_tol, expdir, expid, extract_info, extract_sess_attribs, gabors, get_active_rois, get_frames_timestamps, get_matched_rois, get_nanrois, get_nrois, get_plateau_roi_traces, get_pup_data, get_pup_fr_by_twop_fr, get_roi_masks, get_roi_seqs, get_roi_traces, get_run_velocity, get_run_velocity_by_fr, get_stim, get_stim_twop_fr_ns, get_twop_fr_ran, grayscr, home, line, load_pup_data, load_roi_info, load_run_data, mouse_dir, mouse_n, mouseid, n_drop_stim_fr, n_stims, nanrois_dff, notes, nrois, only_matched_rois, pass_fail, plane, post_blank, pre_blank, procdir, pup_data_h5, pup_fps, pup_fr_interv, pup_video_h5, roi_extract_json, roi_facts_df, roi_mask_file, roi_masks, roi_names, roi_objectlist, roi_trace_dff_h5, roi_trace_h5, run_data, runtype, segid, sess_gen, sess_n, sess_stim_seed, sess_within, sessid, stim2twopfr, stim2twopfr2, stim_df, stim_dict, stim_fps, stim_pkl, stim_sync_h5, stims, stimtype_order, stimtypes, time_sync_h5, tot_pup_fr, tot_run_fr, tot_stim_fr, tot_twop_fr, twop2bodyfr, twop2pupfr, twop_fps, twop_fr_stim, zstack_h5

Gabors: _add_stim_fr_info, _add_twop_fr_info, _check_brick_prod_params, _set_block_params, _update_block_params, act_n_blocks, blank_per, block_params, deg_per_pix, disp_seq, exp_block_len_s, exp_n_blocks, extra_segs, get_A_frame_1s, get_A_segs, get_all_surp_segs, get_all_surp_stim_fr, get_first_surp_segs, get_first_surp_stim_fr_trans, get_n_twop_fr_by_seg, get_pup_diam_data, get_pup_diam_stats_df, get_roi_data, get_roi_stats_df, get_run, get_run_data, get_run_stats_df, get_segs_by_criteria, get_segs_by_twopfr, get_stats_df, get_stim_beh_sub_df, get_stim_df_by_criteria, get_stim_fr_by_criteria, get_stim_fr_by_seg, get_stim_par_by_seg, get_stim_par_by_twopfr, get_twop_fr_by_seg, n_patches, n_seg_per_set, ori_kaps, ori_ran, ori_std, oris, oris_pr, phase, pos_x, pos_x_ran, pos_y, pos_y_ran, post, pre, reg_max_s, reg_min_s, seg_len_s, seg_ps_nobl, seg_ps_wibl, sess, set_len_s, sf, size_ran, sizes_pr, stim_fps, stim_n, stim_seg_list, stimtype, surp_max_s, surp_min_s, units, win_size

Bricks: _add_stim_fr_info, _add_twop_fr_info, _check_brick_prod_params, _set_block_params, _update_block_params, act_n_blocks, blank_per, block_params, deg_per_pix, direcs, disp_seq, exp_block_len_s, exp_n_blocks, extra_segs, flipfrac, get_all_surp_segs, get_all_surp_stim_fr, get_dir_segs_reg, get_first_surp_segs, get_first_surp_stim_fr_trans, get_n_twop_fr_by_seg, get_pup_diam_data, get_pup_diam_stats_df, get_roi_data, get_roi_stats_df, get_run, get_run_data, get_run_stats_df, get_segs_by_criteria, get_segs_by_twopfr, get_stats_df, get_stim_beh_sub_df, get_stim_df_by_criteria, get_stim_fr_by_criteria, get_stim_fr_by_seg, get_twop_fr_by_seg, n_bricks, reg_max_s, reg_min_s, seg_len_s, seg_ps_nobl, seg_ps_wibl, sess, sizes, speed, stim_fps, stim_n, stim_n_all, stim_seg_list, stimtype, surp_max_s, surp_min_s, units