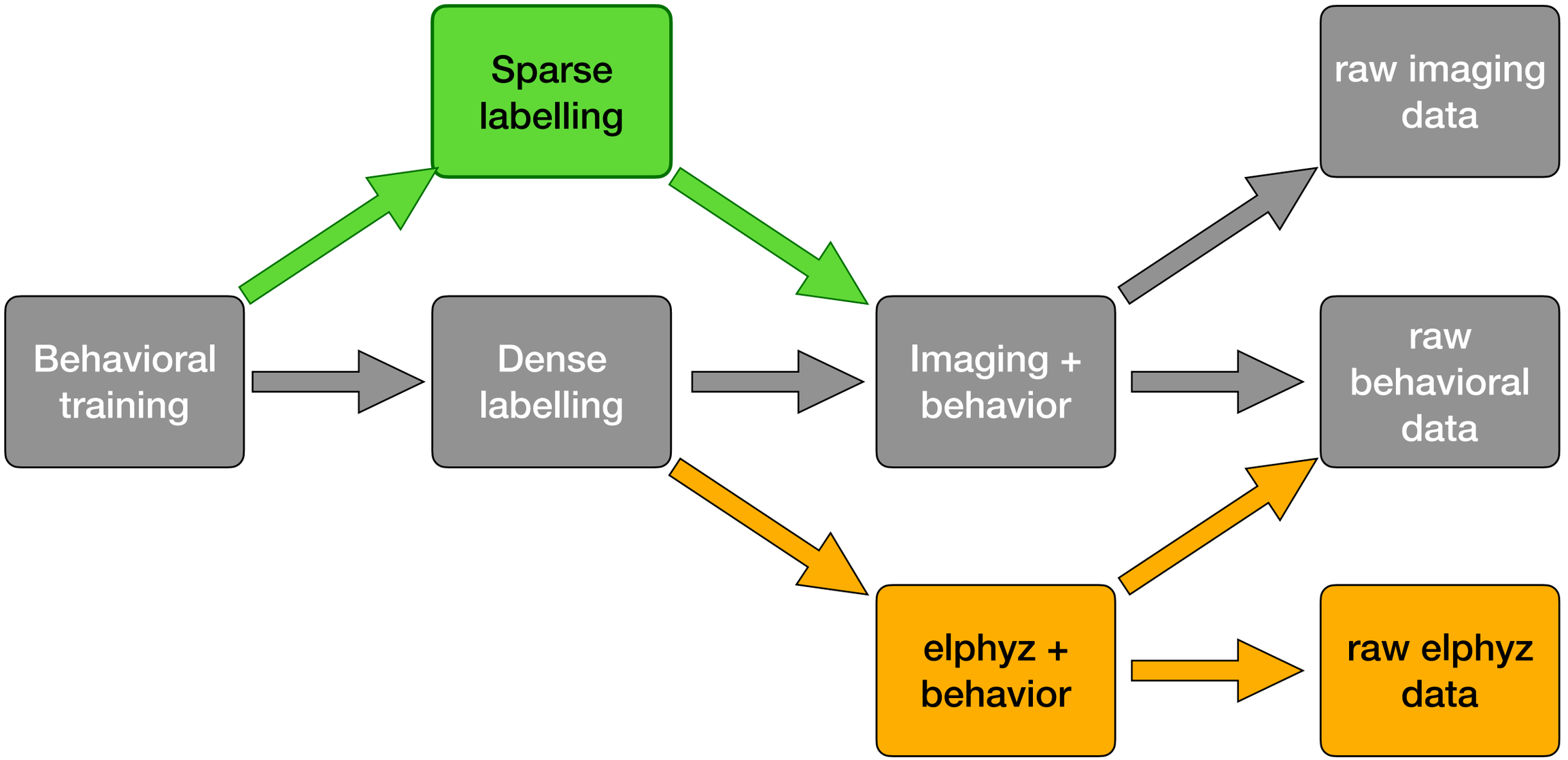
ABmice

https://github.com/bbujfalussy/ABmice

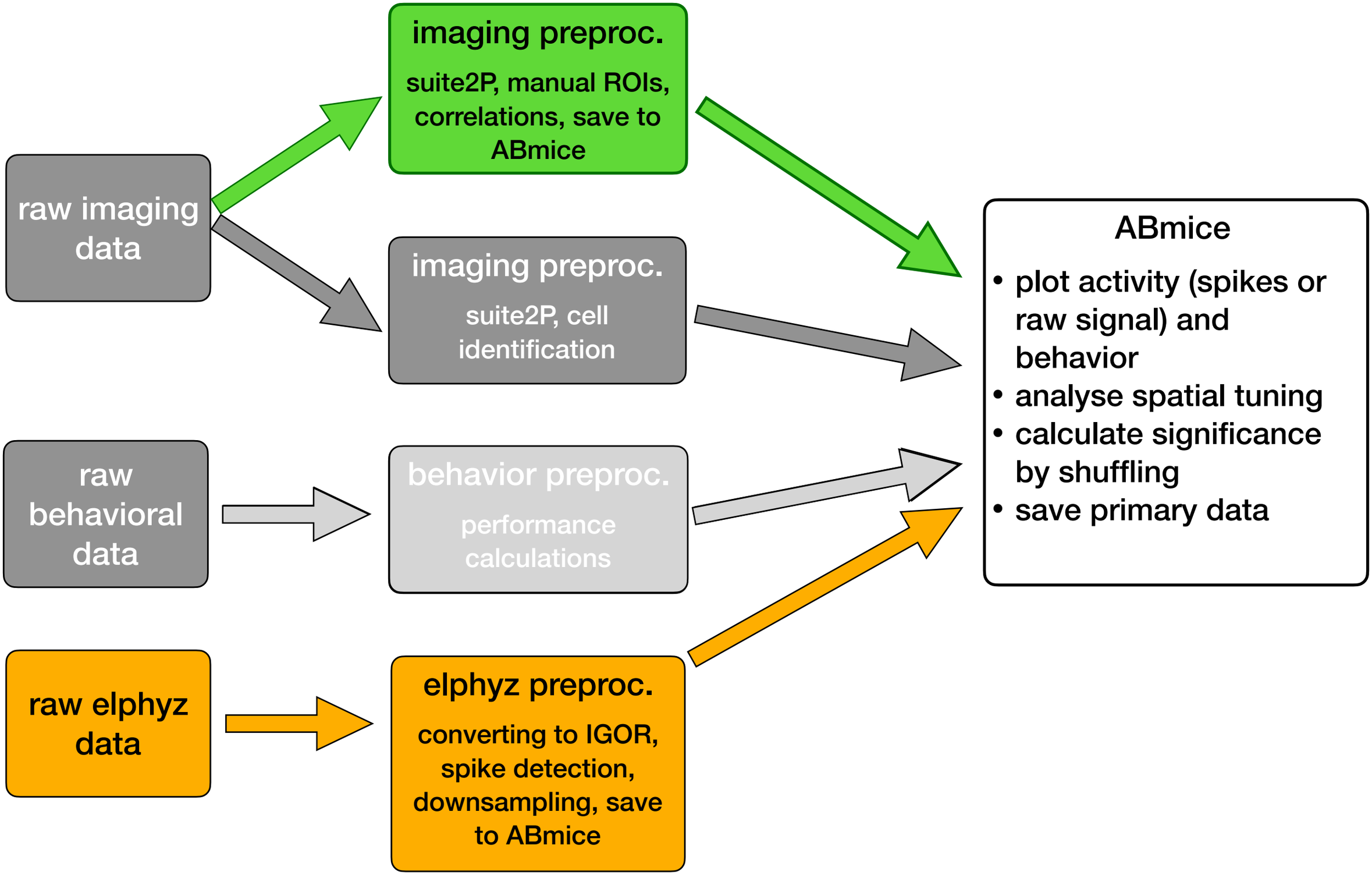
Balázs[2], 21 July, 2021

Experiments pipeline



Snezana
Rita, Kata
Berci

Analysis pipeline



Folders and files

~ root

- Data
- ABmice (python scripts)

Corridors.py
ImageAnal_corrid0.py
ImageAnal_new.py
ImageAnal.py
ImShuffle.py
LogAnal.py
Maze_Labview2Ogre.py
Mice.py
Mouse_Close.py
…

Data/AB057

- processed behavioral
  dataAB057_NearFarLong.pkl

Data/AB057_NearFarLong

- raw behavioral data and Elphys data ***for each session***
  2021-06-25_13-04-27
  2021-06-21_19-21-14
  2021-06-17_10-51-47
  …
- processed behavioral data ***for each session***
  behaviour_data

/AB057_NearFarLong/
2021-06-17_10-51-47/

- **behavioral log files**
  (4 x *.txt, 3 x *.bin)
- raw Elphys data ***for this session***
  AB2021_06_25mc1/
  AB2021_06_25_0000.abf
  AB2021_06_25_0001.abf
- **processed Elphys data:**
  frame_times_0001.npy
  spikes_0001.npy
  Vm_0001.npy
  Trigger_0001.csv

Data/AB057_imaging

- imaging data ***for each session***
  Suite2P_1_06-07-2021
  Suite2P_1_06-07-2021_manual_ROI

/Suite2P_1_06-07-2021

- **raw imaging log files**
  (xml and csv files)
- **Suite2P files** (F.npy, spks.npy, iscell.npy)

Folders and files

~ root

- Data
- ABmice (python scripts)

Corridors.py
ImageAnal_corrid0.py
ImageAnal_new.py
ImageAnal.py
ImShuffle.py
LogAnal.py
Maze_Labview2Ogre.py
Mice.py
Mouse_Close.py
…

Data/AB057

- processed behavioral
  dataAB057_NearFarLong.pkl

Data/AB057_NearFarLong

- raw behavioral data and Elphys data ***for each session***
  2021-06-25_13-04-27
  2021-06-21_19-21-14
  2021-06-17_10-51-47
  …
- processed behavioral data ***for each session***
  behaviour_data

Data/AB057_imaging

- imaging data ***for each session***
  Suite2P_1_06-07-2021
  Suite2P_1_06-07-2021_manual_ROI

/AB057_NearFarLong/
2021-06-17_10-51-47/

- **behavioral log files**
  (4 x *.txt, 3 x *.bin)

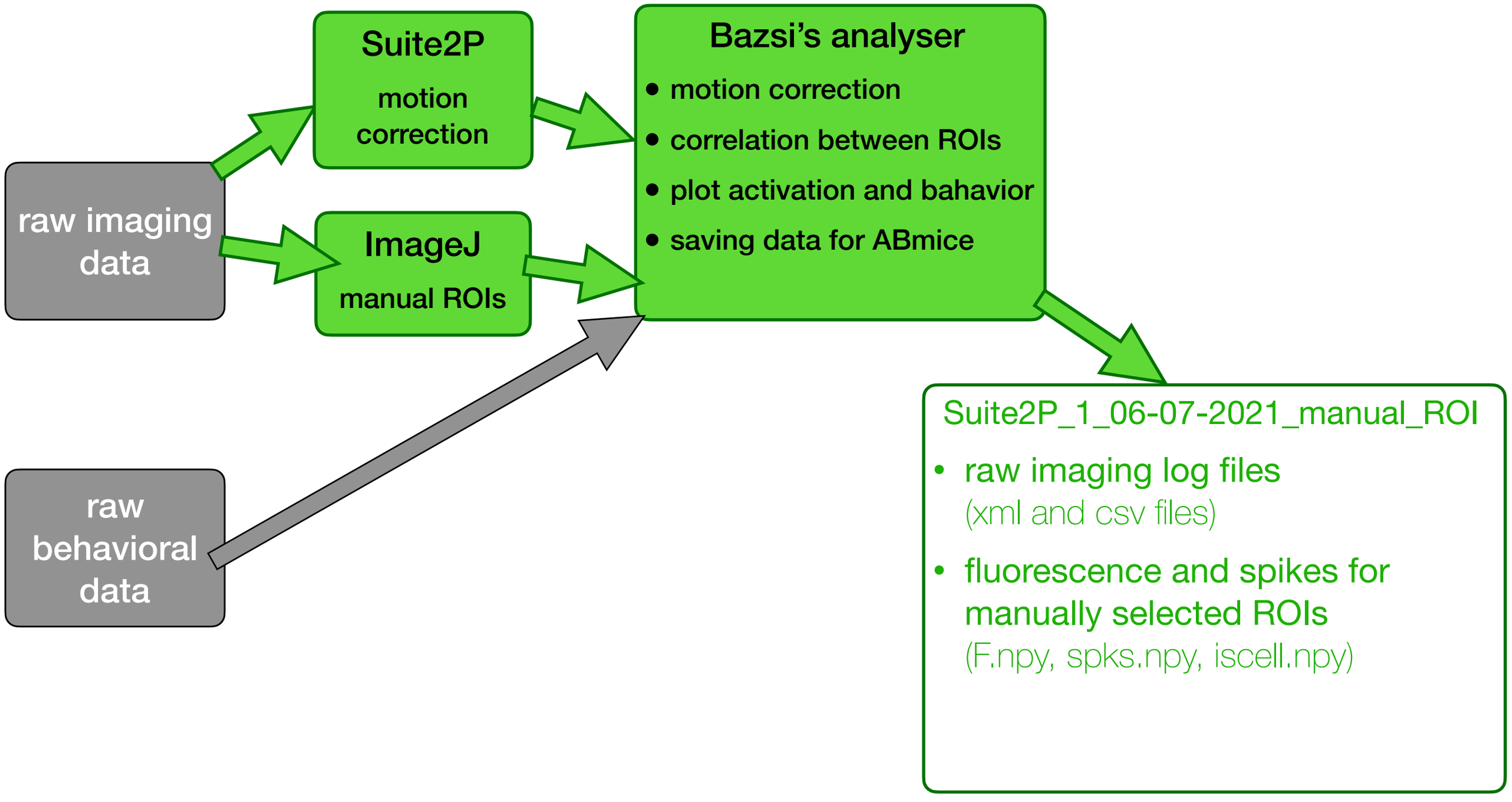
Suite2P_1_06-07-2021_manual_ROI

- raw imaging log files
  (xml and csv files)
- fluorescence and spikes for manually selected ROIs
  (F.npy, spks.npy, iscell.npy)

/Suite2P_1_06-07-2021

- **raw imaging log files**
  (xml and csv files)
- **Suite2P files** (F.npy, spks.npy, iscell.npy)

Preprocessing - sparse labeling



Preprocessing - elphys

raw behavioral data

raw elphyz data

Bazsi analyser.py

- Loads raw data
- Finds the matching position, lick and reward
- Saves it for IGOR in csv
- Downsampling to 5000 Hz
- Spike detection (user interaction needed)
- Removing stimulation artefacts
- Saving for ABmice

/AB057_NearFarLong/
2021-06-17_10-51-47/

- **behavioral log files**
  (4 x *.txt, 3 x *.bin)
- raw Elphys data ***for this session***
  AB2021_06_25mc1/
  AB2021_06_25_0000.abf
  AB2021_06_25_0001.abf
- **processed Elphys data:**
  frame_times_0001.npy
  spikes_0001.npy
  Vm_0001.npy
  Trigger_0001.csv

ABmice

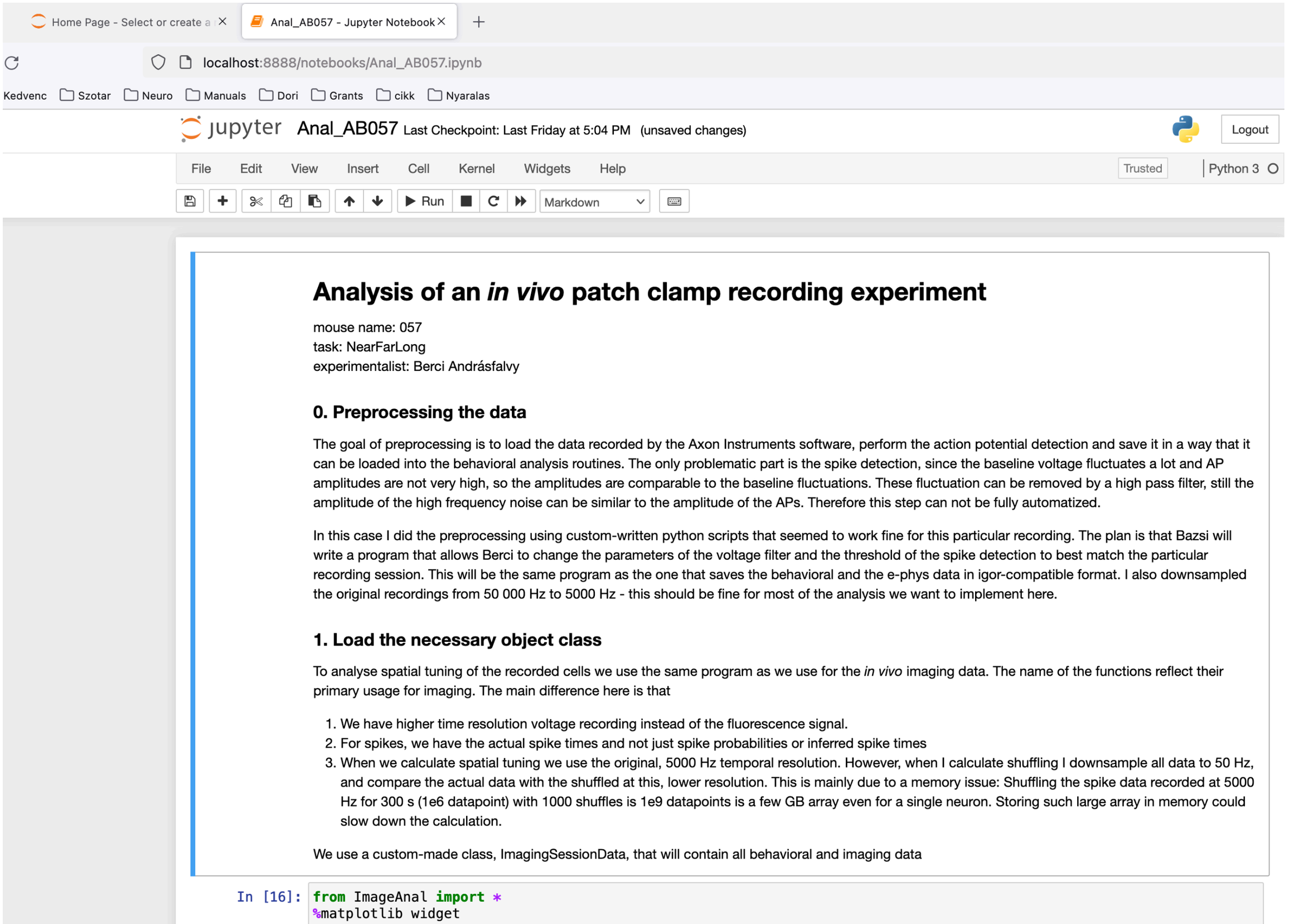
https://www.sourcetreeapp.com

https://github.com/bbujfalussy/ABmice

Latest changes:

1. Corrections
    a) synchronization between imaging and behavior
    b) speed threshold
2. Integration
    a) should work with any number of corridors
    b) can handle multiple reward zones and arbitrary corridor length
3. New features
    a) Jupyter notebook
    b) saving primary data
    c) handling data from elphyz experiments
    d) saves P-values after shuffling and uses the same file when performing shuffling again

Starting


Home Page - Select or create a
Anal_AB057 - Jupyter Notebook
localhost:8888/notebooks/Anal_AB057.ipynb
Kedvenc
Szotar
Neuro
Manuals
Dori
Grants
cikk
Nyaralas
jupyter
Anal_AB057
Last Checkpoint: Last Friday at 5:04 PM
(unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Markdown
Analysis of an in vivo patch clamp recording experiment
mouse name: 057
task: NearFarLong
experimentalist: Berci Andrásfalvy
0. Preprocessing the data
The goal of preprocessing is to load the data recorded by the Axon Instruments software, perform the action potential detection and save it in a way that it can be loaded into the behavioral analysis routines. The only problematic part is the spike detection, since the baseline voltage fluctuates a lot and AP amplitudes are not very high, so the amplitudes are comparable to the baseline fluctuations. These fluctuation can be removed by a high pass filter, still the amplitude of the high frequency noise can be similar to the amplitude of the APs. Therefore this step can not be fully automatized.
In this case I did the preprocessing using custom-written python scripts that seemed to work fine for this particular recording. The plan is that Bazsi will write a program that allows Berci to change the parameters of the voltage filter and the threshold of the spike detection to best match the particular recording session. This will be the same program as the one that saves the behavioral and the e-phys data in igor-compatible format. I also downsampled the original recordings from 50 000 Hz to 5000 Hz - this should be fine for most of the analysis we want to implement here.
1. Load the necessary object class
To analyse spatial tuning of the recorded cells we use the same program as we use for the in vivo imaging data. The name of the functions reflect their primary usage for imaging. The main difference here is that
1. We have higher time resolution voltage recording instead of the fluorescence signal.
2. For spikes, we have the actual spike times and not just spike probabilities or inferred spike times
3. When we calculate spatial tuning we use the original, 5000 Hz temporal resolution. However, when I calculate shuffling I downsample all data to 50 Hz, and compare the actual data with the shuffled at this, lower resolution. This is mainly due to a memory issue: Shuffling the spike data recorded at 5000 Hz for 300 s (1e6 datapoint) with 1000 shuffles is 1e9 datapoints is a few GB array even for a single neuron. Storing such large array in memory could slow down the calculation.
We use a custom-made class, ImagingSessionData, that will contain all behavioral and imaging data
In [16]:
from ImageAnal import *
%matplotlib widget

Folders

2. Tell python where the data is

The required file structure is the following:

- All data should be in the data folder
- Within the data folder separate subfolders are needed for each mouse. Folder name starts with the **name** of the mouse.
- For each mouse there should be a single folders containing **both** the **e-phys** data and the **behavioral** data.
- The folder is named as `MouseName_TaskName` - so we need a separate folder for each different task
- The behavioral log files are in separate subfolders named by the experiment's start time within the behavioral folder - e.g. `2021-02-03_10-15-50`
- There are several recordings (traces) within the same session. They should be numbered - `imaging_logfile_name` is the number we want to process.
- We need to tell python that the e-phys files are in the same folder. Remember, python thinks that these ar eimaging files!

In [17]:
```
datapath = os.getcwd() + '/' #current working directory - look for data and strings here!
date_time = '2021-06-25_13-04-27' # date and time of the imaging session
name = 'AB057' # mouse name
task = 'NearFarLong' # task name

suite2p_folder = datapath + 'data/' + name + '_NearFarLong/2021-06-25_13-04-27/' # the e-phys files should be here

## the name and location of the trigger voltage file
TRIGGER_VOLTAGE_FILENAME = suite2p_folder + 'Trigger_0001.csv'

## the name and location of the imaging log file - this is the number of the recording session...
imaging_logfile_name = '0001' # elfiz_session_number
```

Folders

3. Load all the data - this taks ~20 secs in my computer

Python looks for the data in the specified folders. It loads the behavioral data (position, lick and rewards) as well as the imaging data. It calculates the activity of the cells as a function of position in the different corridors and calculates their spatial tuning measures and corridor selectivity.
The name of the object that contains all the data is `D1` here - Data 1.

Important: to tell python that you want to load e-phys data, you need to write at the end that `elfiz = True`

Important 2.: Berci recorded from the same mouse for ~2 hours, and there are almost 1000 laps. Loading all the data needs a lots of memory, and takes forever. So I selected only the 20-80 laps that contain the laps during the current recordings. Iy you want to analyse a different recording, then you need to load the appropriate traces!

In [3]:
```
# 3. load all the data - this taks ~20 secs in my computer
D1 = ImagingSessionData(datapath, date_time, name, task, suite2p_folder, imaging_logfile_name, TRIGGER_VOLTAGE_FILEN
```

```
TRIGGER_VOLTAGE_FILENAME, selected_laps=np.arange(20,80), speed_threshold=5, randseed=123, elfiz=True)
```

```
trigger logfile loaded
trigger voltage signal loaded
triggers after: 22
n_extra_indexes 5
candidate log indexes [48, 147, 246, 344, 345, 443, 444, 542, 543, 641, 642, 740, 839, 938]
min recorded trigger length: 0.4898000000000007
relevant behavior located, lap time of the first frame: 1360.813762 , log reference index: 48
elfiz data loaded
elfiz time axis loaded
ExpStateMachineLog time interval > 1s:  293  times
############################################################
substage change detected!
first lap in substage  ['1'] is lap 0 , which started at t 1051.0210371398189
the time of the change in imaging time is:  -309.7927248601811
############################################################
calculating rate, reliability and Fano factor...
calculating Skaggs spatial info...
```

Laps with imaging data and behavior

The behavior is divided into laps (trials or runs). You can check the **number of laps** and which lap is associated with imaging data in the following way:

In [4]:
```
print(D1.n_laps)
print(D1.i_Laps_ImData)
```

```
60
[28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50]
```

So we have 60 laps and laps 28-50 contains imaging (elphys) data.

Plot the behavioral data

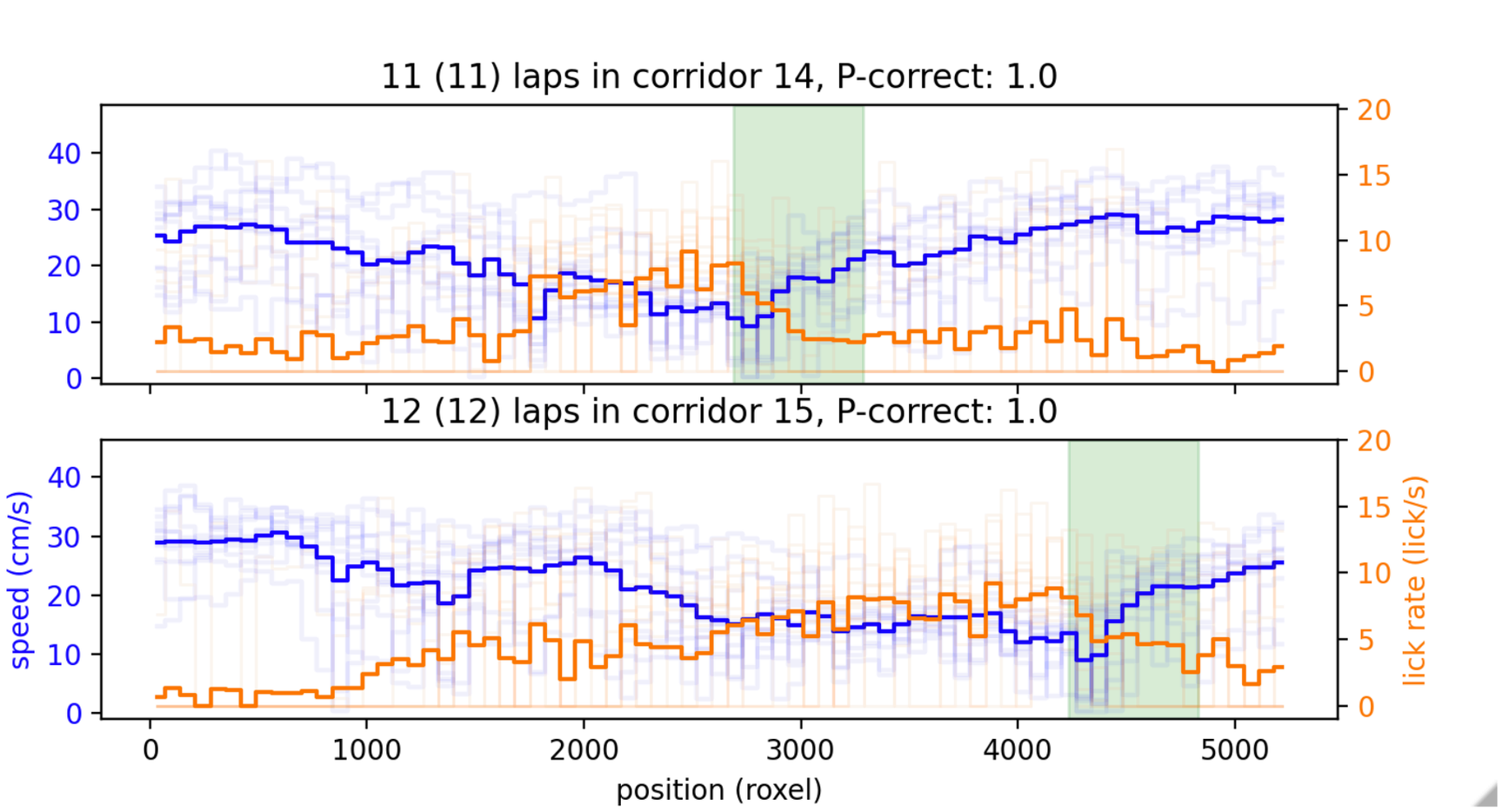
```
    def plot_session(self, selected_laps=None):
        ## plot the behavioral data during one session.
            # - speed
            # - lick rate
        ## selected laps: numpy array indexing the laps to be included in the plot
```

4. Plotting the behavioral data

You can plot the behavioral data of the session. This mouse runs a lot, but the performance is not great: it lick almost everywhere and does not systematically slow down before reward zone.

In [5]:
```
#D1.plot_session()
D1.plot_session(selected_laps=D1.i_Laps_ImData)
```

Figure 1



Plot the laps with imaging (elphys) data - Vm and spikes as a function of time

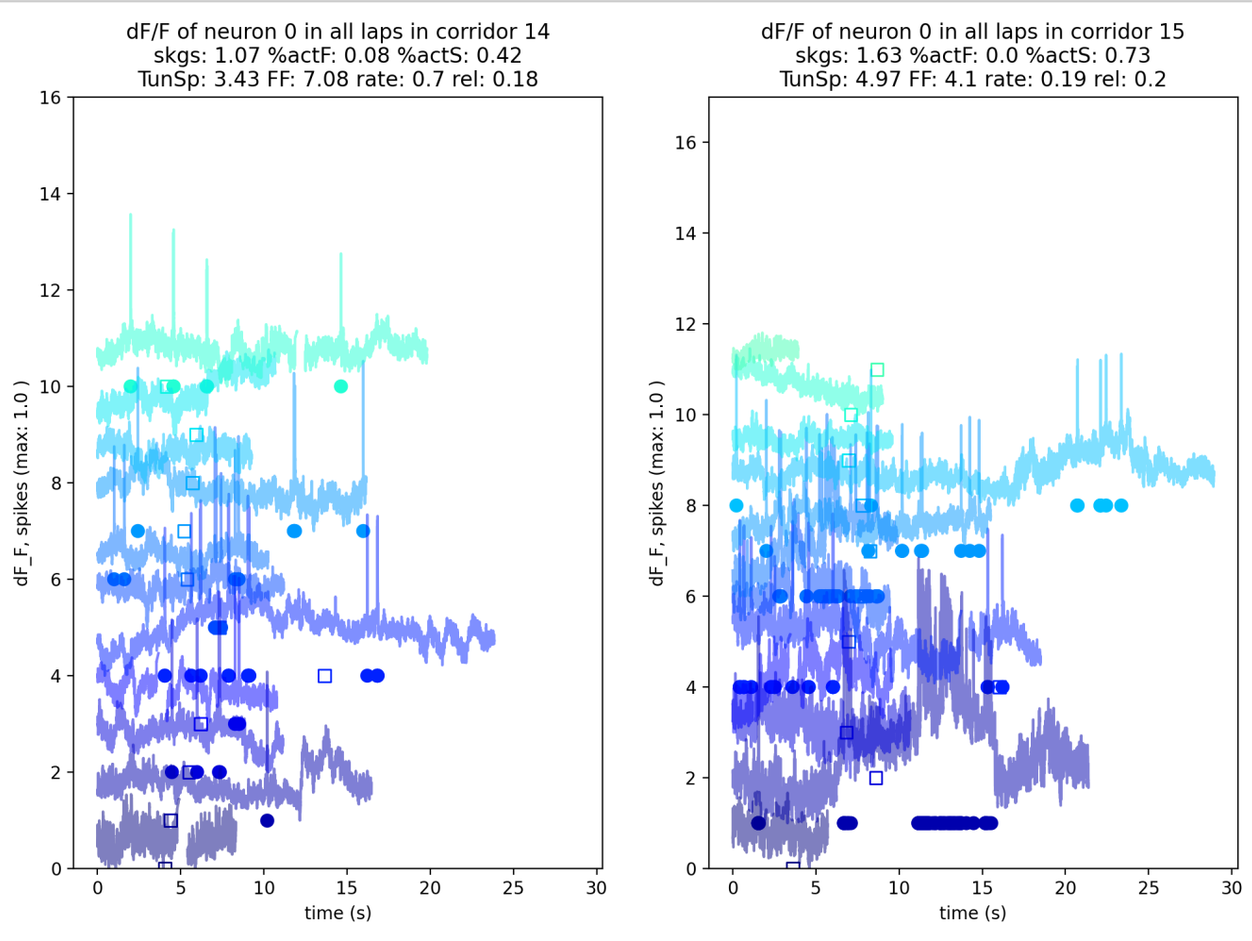
```
def plot_cell_laps(self, cellid, multipdf_object=-1, signal='dF', corridor=-1, reward=True, write_pdf=False, plot_laps='all'
    ## plot the activity of a single cell in all trials in a given corridor
    ## signal can be
    ##          'dF' when dF/F and spikes are plotted as a function of time
    ##          'rate' when rate vs. position is plotted
    ## plot_laps can be either 'all', 'correct' or 'error'
```

6. Plot the activity lap by lap

We can also plot the lap by lap activity of a the recorded cell. Again, there are several options, but first is to plot the Vm and the spikes as a function of time. To achieve this, we need to set `signal = 'dF'`.

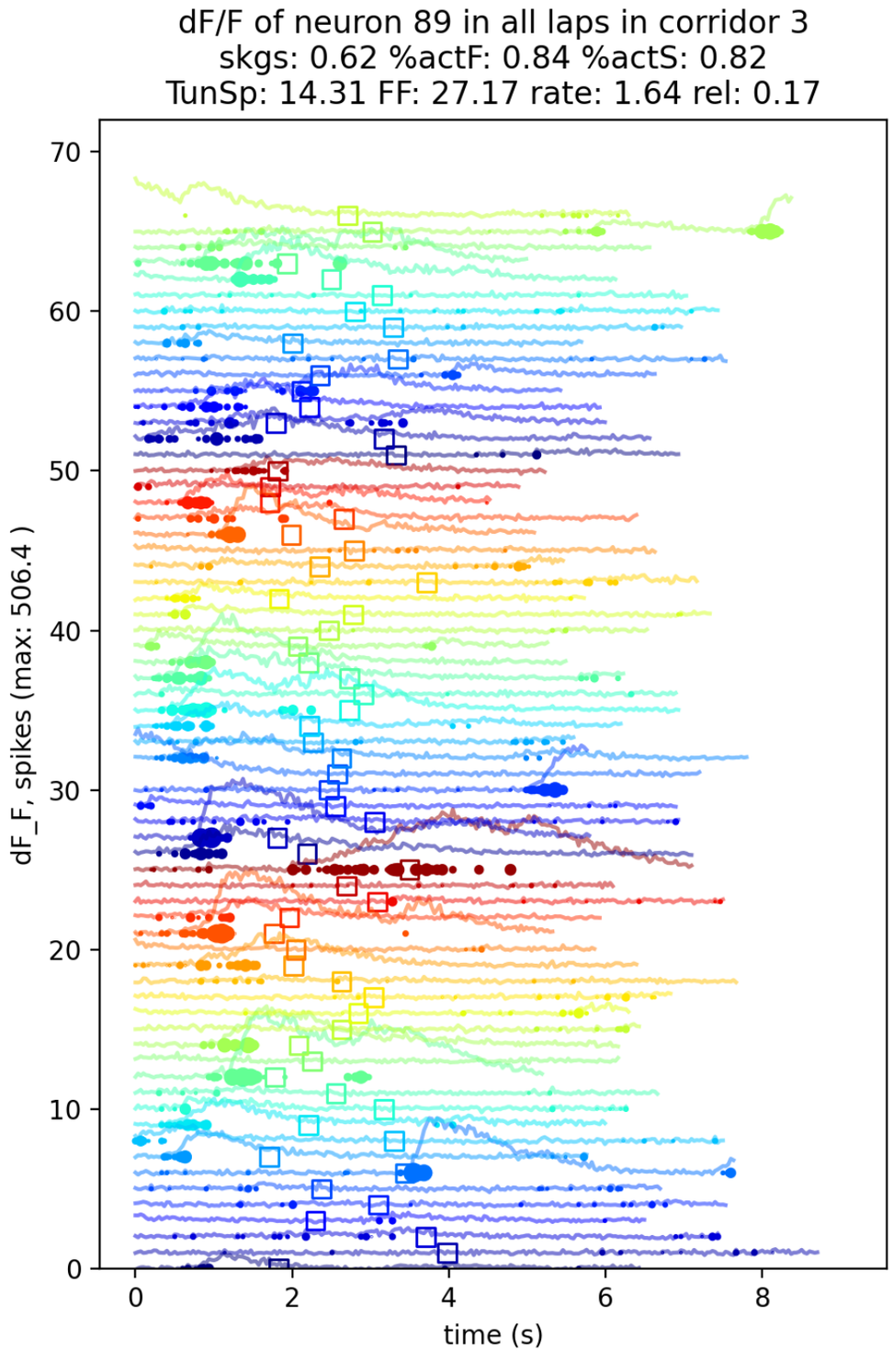
Different colors are different laps. Solid line is the recorded Vm - gaps correspond to calibration pulses that have been removed during preprocessing. Traces are shifted so that their minimum is at the corresponding point in the y-axis. Vm curves are transparentm so they do not cover each other. Open square is the reward time and colored dotes are the spikes - each at the y coordinate at the given lap.

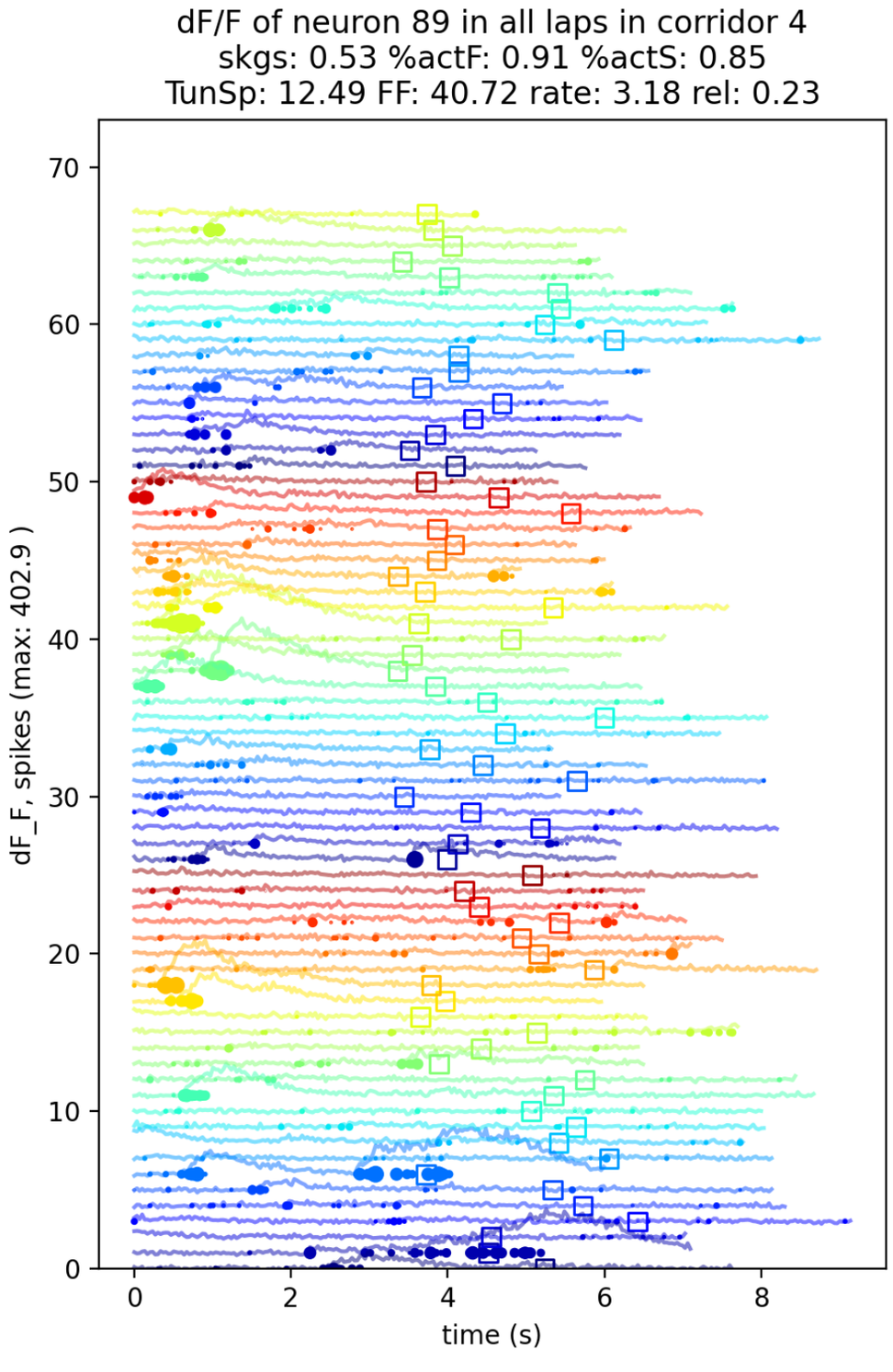
In [6]:
```
D1.plot_cell_laps(cellid=0, signal='dF') ## look at lap 20
```



Plot the laps with imaging (elphys) data - spikes and dF/F as a function of time

```
In [12]: D1.plot_cell_laps(cellid=89, signal='dF')
```



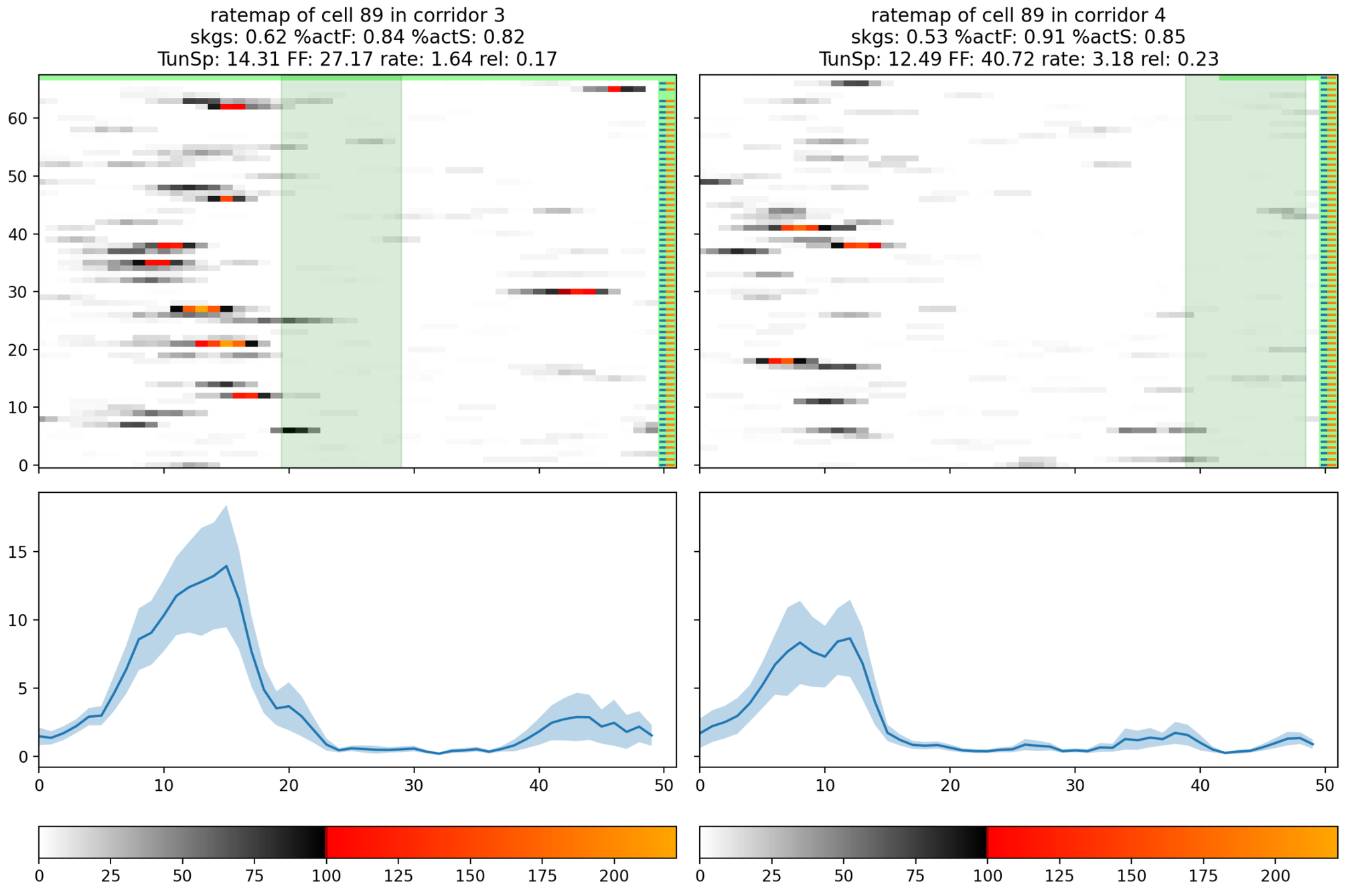


Plot the laps with imaging (elphys) data - firing rate as a function of space

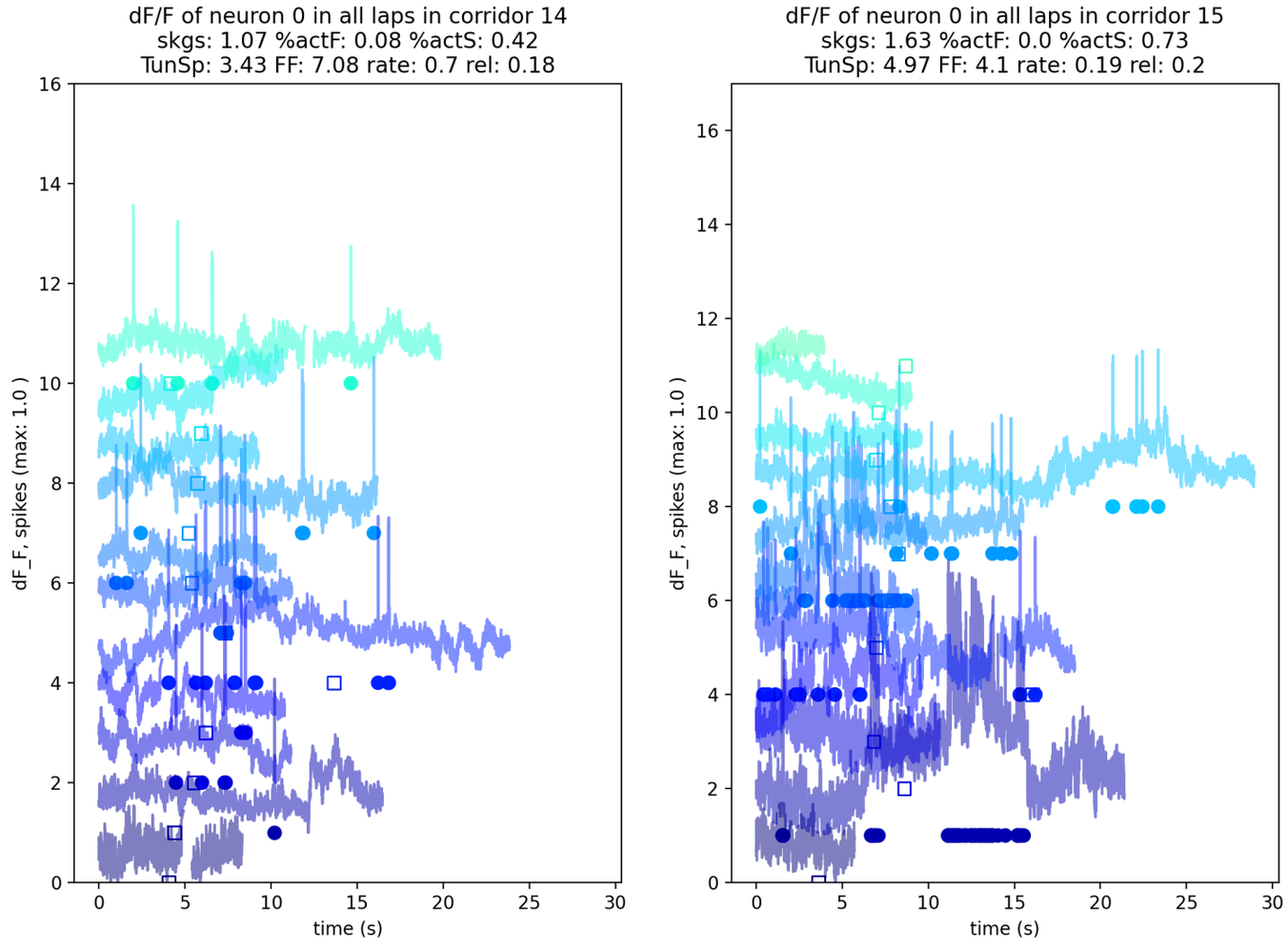
8. Plot the activity lap by lap

We can also plot the lap by lap activity of a selected cell. Again, there are several options, but the simplest is to plot the rate as a function of position.

In [13]:
```
D1.plot_cell_laps(cellid=89, signal='rate')
```



Laps with imaging data and behavior



We see that in lap 6 in corridor 15 the cell fired a lots of spikes. We can check the index of this lap using the following function:

In [10]:
```
D1.get_lap_indexes(corridor=15, i_lap=6)
```

```
lap # in corridor 15 with imaging data;    lap # within session
6        35
```

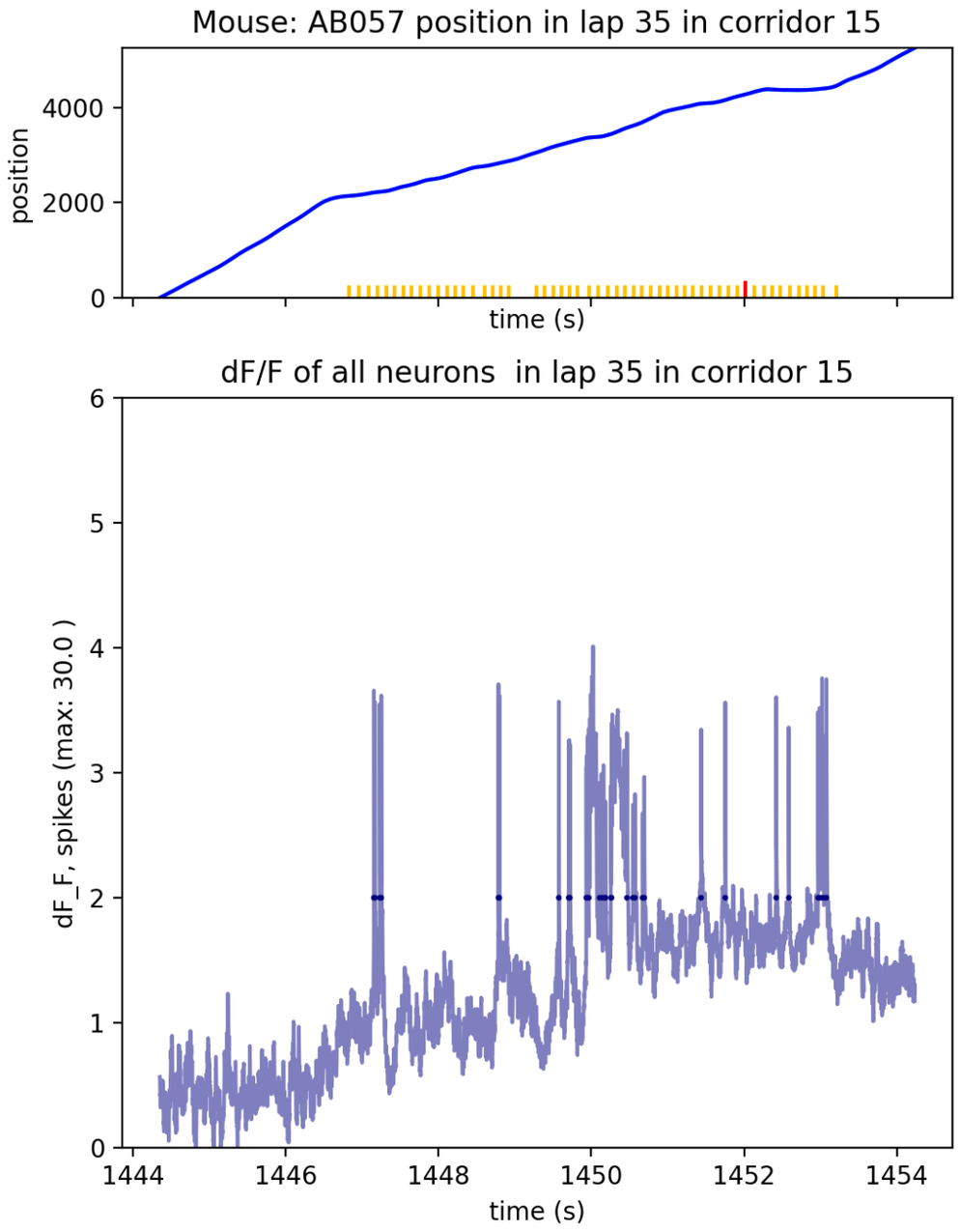
We can also plot the activity in this particular lap. The argument `th = -0.5` tells python to include all cells with spike count larger than `-0.5` which is all neurons. `flue = True` means that the flourescence (voltage) is also plotted.

In [11]:
```
D1.ImLaps[35].plot_tx(th=-0.5, fluo=True)
```

Single laps with elphys data and behavior

```
    def plot_tx(self, fluo=False, th=25):
        ## fluo: True or Fasle, whether fluoresnece data should be plotted.
        ## th: threshold for plotting the fluorescence data - only cells with spikes > th are shown
        ##      when plotting elphys data, th should be -0.5 to show the voltage trace
```

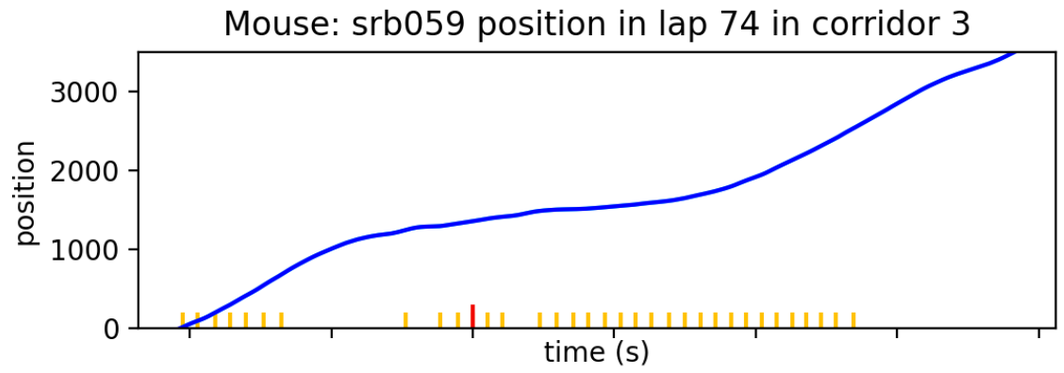
In [11]:
```
D1.ImLaps[35].plot_tx(th=-0.5, fluo=True)
```

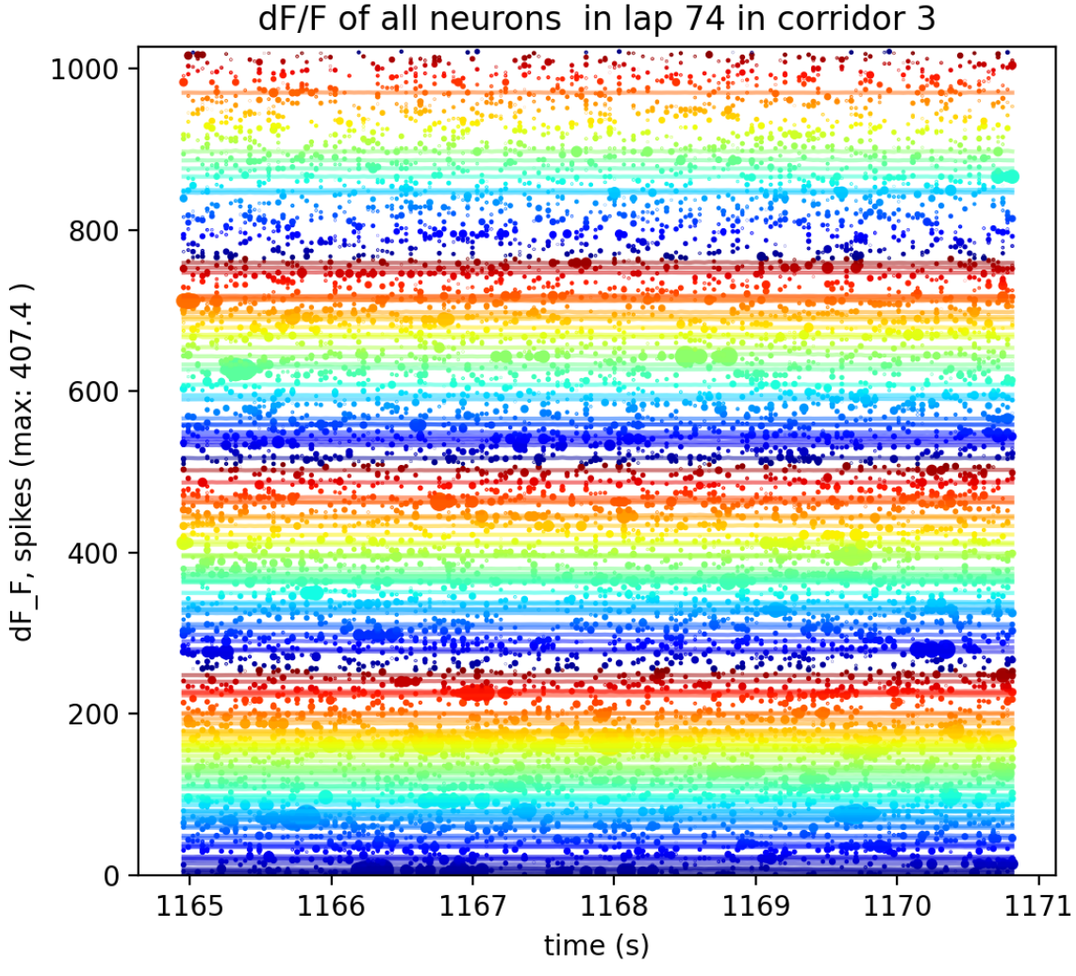


Single laps with imaging data and behavior

```
    def plot_tx(self, fluo=False, th=25):
        ## fluo: True or Fasle, whether fluoresnece data should be plotted.
        ## th: threshold for plotting the fluorescence data - only cells with spikes > th are shown
        ##      when plotting elphys data, th should be -0.5 to show the voltage trace
```

```
D1.ImLaps[74].plot_tx(fluo=True, th=50)
```

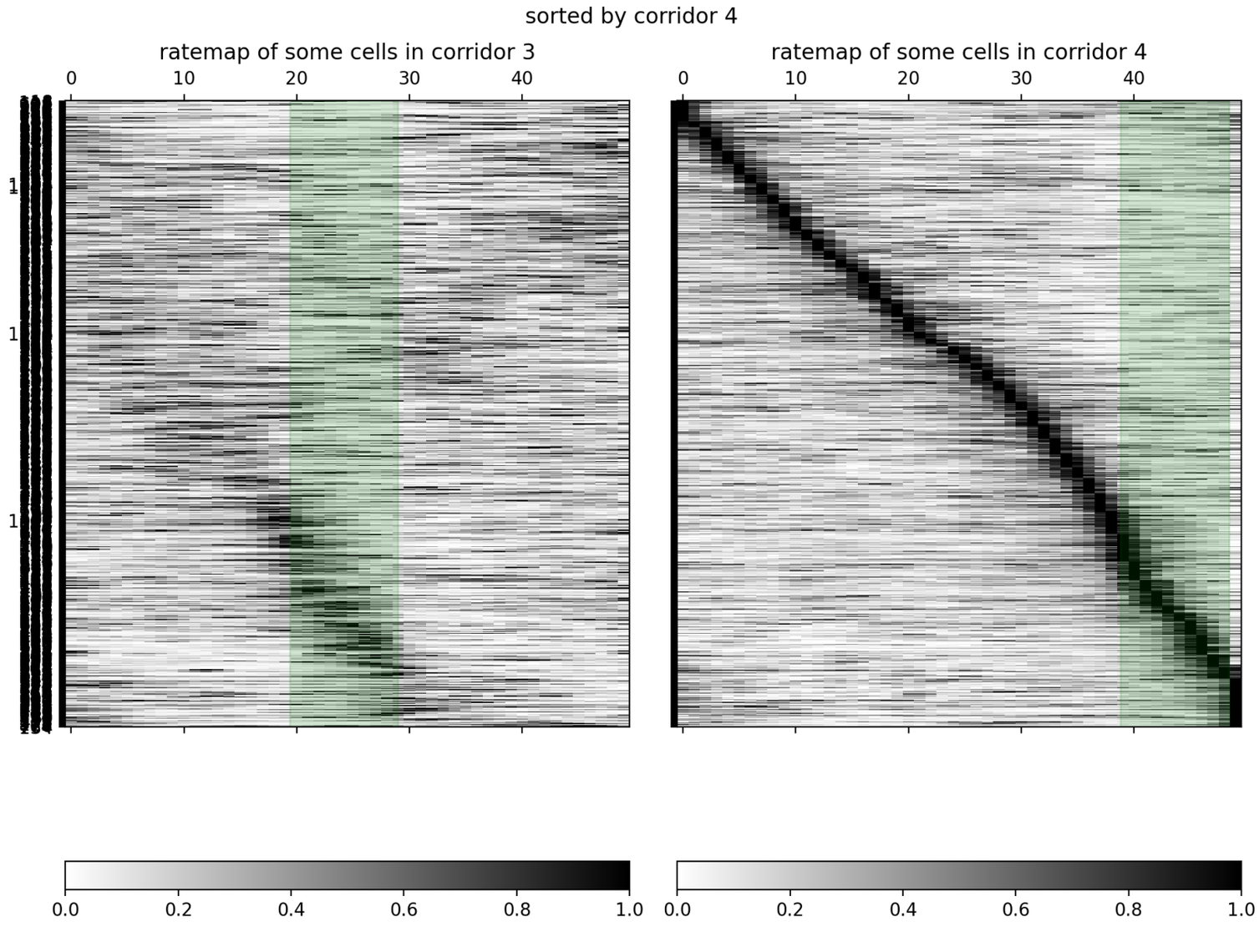




Ratemaps

```
def plot_ratemaps(self, corridor=-1, normalized=False, sorted=False, corridor_sort=-1, cellids=np.array([-1]), vmax=0):
    ## plot the average event rate of all cells in a given corridor
    ## corridor: integer, the ID of a valid corridor
    ##      if corridor == -1 then all corridors are used
    ## normalized: True or False. If True then  each cell ratemap is normalized to have a max = 1
    ## sorted: sorting the rememaps by their peaks
    ## corridor_sort: which corridor to use for sorting the ratemaps
    ## cellids: np array with the indexes of the cells to be plotted. when -1: all cells are plotted
    ## vmax: float. If ratemaps are not normalised then the max range of the colors will be at least vmax.
            # If one of the ratemaps has a higher peak, then vmax is replaced by that peak
```

```
cellids = np.nonzero(((D1.cell_activelaps[0]>0.2) + (D1.cell_activelaps[1]>0.2)))[0]
D1.plot_ratemaps(cellids = cellids, sorted=True, corridor_sort=4, normalized=True)
```

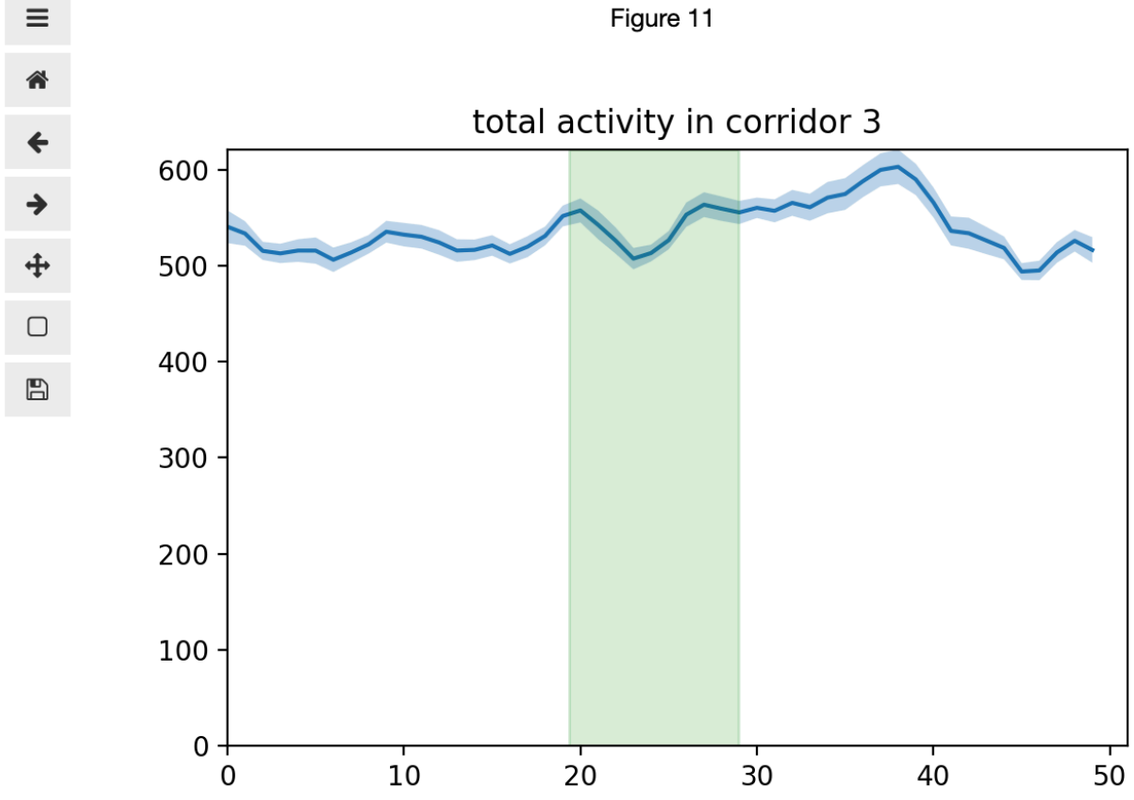


Total population activity

```
def plot_popact(self, cellids, corridor=-1, name_string='selected_cells', bylaps=False):
    ## plot the total population activity in all trials in a given corridor
    ## cellids: numpy array. The index of the cells included in the population activity
    ## corridor: integer, the ID of a valid corridor
    ##      if corridor == -1 then all corridors are used
    ## bylaps: True or False. If True then population activity is plotted lap by lap
```

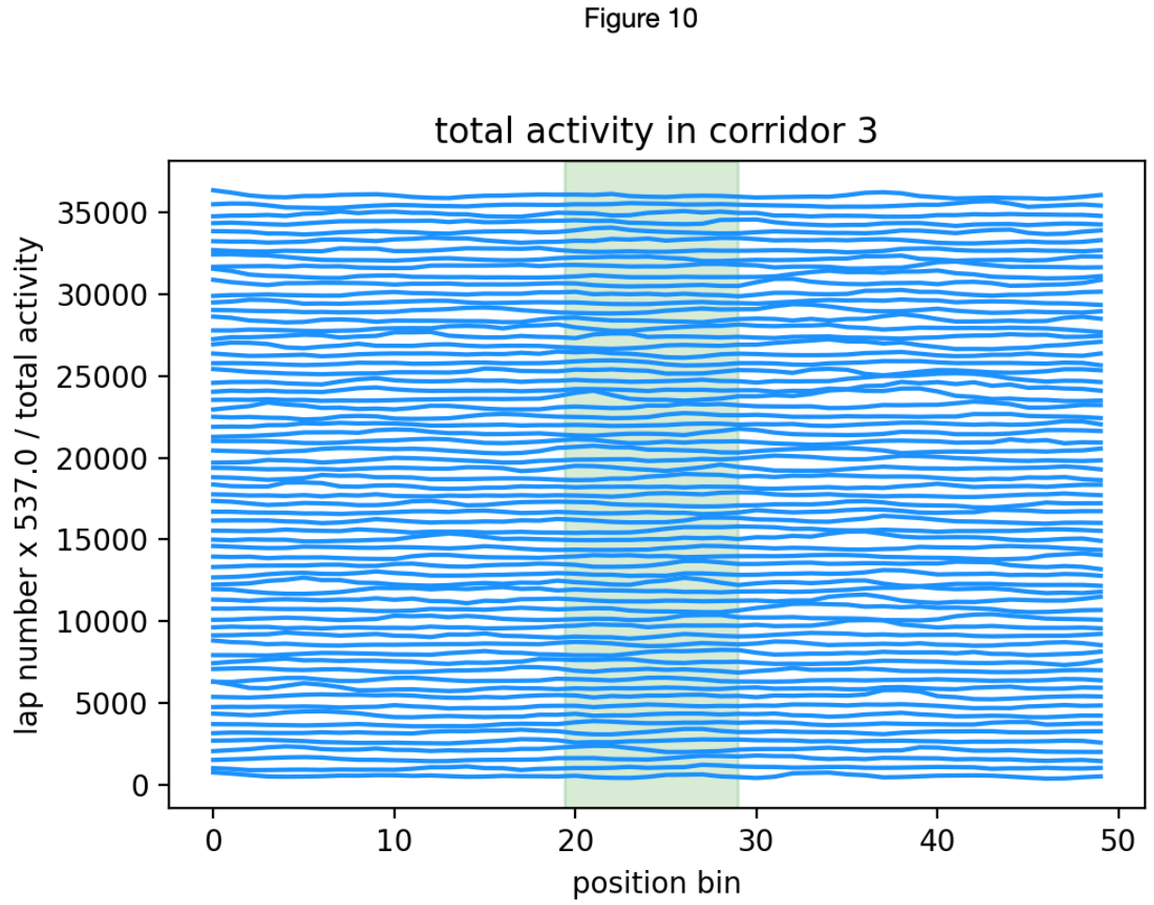
```
D1.plot_popact(cellids, corridor=3)
```

Figure 11




```
D1.plot_popact(cellids, bylaps=True, corridor=3)
```

Figure 10



Calculating shuffling stats

```
    def calc_shuffle(self, cellids, n=1000, mode='shift', batchsize=25, verbous=True):
        ## cellids: numpy array - the index of the cells to be included in the analysis
        ## n: integer, number of shuffles
        ## mode: 'random' or 'shift'.
            # random: totally randomize the spike times;
            # shift: circularly shift spike times
        ## batchsize: integer. To make computation faster, shuffling is done in batches of size batchsize
        ## verbous: True of False: information is gizev about the progress
```

In [8]:
```
D1.calc_shuffle(cellids, 1000, 'shift', batchsize=50)
```

```
loading shuffling P-values from file...
tuned cells: [array([   2,    5,    8,    9,   10,   12,   13,   14,   16,   19,   22,
         23,   24,   26,   27,   28,   30,   31,   35,   36,   38,   40,
         41,   42,   43,   45,   46,   47,   49,   51,   52,   56,   65,
```

Saving the data

```
def save_data(self, save_properties=True, save_ratemaps=True, save_laptime=True):
    # Saves the primary data into a folder in csv format.
    # separate file is created for each corridor and lap.
    #
    # save_properties: True or False. If True, the cell properties are saved.
    # save_ratemaps: True or False. If True, the ratemaps are saved.
    # save_laptime: True or False. If True, the raw data is saved for each lap.
```

```
D1.save_data()
```

```
Session parameters written into file:  cell_properties_corridor_3_N1022.csv
cell properties for corridor 3  saved into file:  /Users/ubi/Projects/KOKI/VR/MiceData/data/srb059_imaging/Suite2P_
4_19-05-2021/analysed_data/cell_properties_corridor_3_N1022.csv
Session parameters written into file:  cell_properties_corridor_4_N1022.csv
cell properties for corridor 4  saved into file:  /Users/ubi/Projects/KOKI/VR/MiceData/data/srb059_imaging/Suite2P_
4_19-05-2021/analysed_data/cell_properties_corridor_4_N1022.csv
Session parameters written into file:  ratemaps_corridor_3_N1022.csv
ratemap for corridor 3  saved into file:  /Users/ubi/Projects/KOKI/VR/MiceData/data/srb059_imaging/Suite2P_4_19-05-
2021/analysed_data/ratemaps_corridor_3_N1022.csv
Session parameters written into file:  ratemaps_corridor_4_N1022.csv
ratemap for corridor 4  saved into file:  /Users/ubi/Projects/KOKI/VR/MiceData/data/srb059_imaging/Suite2P_4_19-05-
2021/analysed_data/ratemaps_corridor_4_N1022.csv
Session parameters written into file:  lapdata_lap_33_N1022.csv
Session parameters written into file:  lapdata_lap_34_N1022.csv
Session parameters written into file:  lapdata_lap_35_N1022.csv
Session parameters written into file:  lapdata_lap_36_N1022.csv
Session parameters written into file:  lapdata_lap_37_N1022.csv
Session parameters written into file:  lapdata_lap_38_N1022.csv
Session parameters written into file:  lapdata_lap_39_N1022.csv
```