

Simple demo for analysing imaging data in python

Balazs Lükő and Balázs B Ujfalussy

1. Folder structure

There is a new script, `ImageAnal.py` which contains all the programs for analysing imaging data, and a simple interface `Mouse_ImView.py` with examples of data analysis. These python scripts should be placed in the same folder as the other python analysis scripts and all your data to be analysed.

All the behavioral data should be in a `data` subfolder as usual.

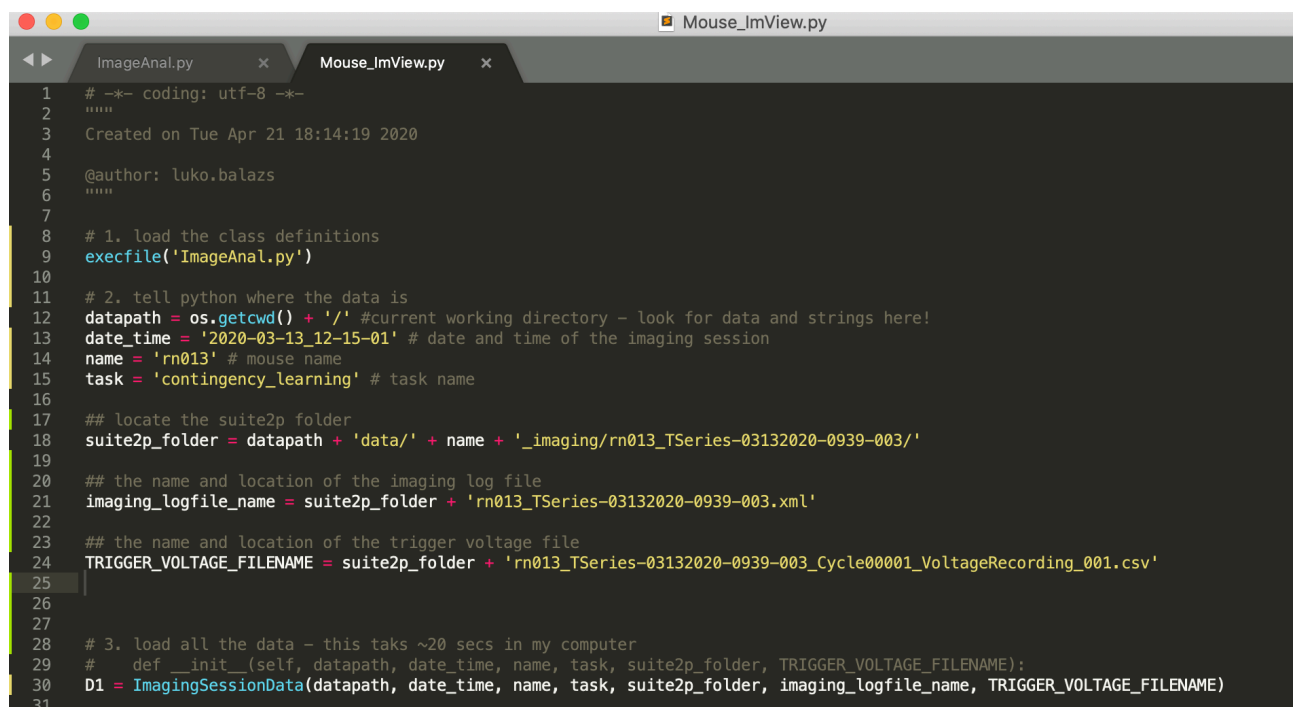
The imaging data could be placed in the `data/rn013_imaging/rn013_TSeries-03132020-0939-003` folder, but also somewhere else. Importantly, the imaging data folder should contain the following files:

- the data files `F.npy`, `spikes.npy` and `iscell.npy` generated by suite2p
- the xml file describing the imaging experiment, e.g.:
`rn013_TSeries-03132020-0939-003.xml`
- the csv file of the recorded trigger signal, e.g.:
`rn013_TSeries-03132020-0939-003_Cycle00001_VoltageRecording_001.csv`

where `rn013` should be replaced by the name and the task. The name of the imaging folder and the `xml` and `csv` files will be set individually.

2. Starting the analysis

Open the `ImageAnal.py` in an editor. This is what you should see:



```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Apr 21 18:14:19 2020
4
5  @author: luko.balazs
6  """
7
8  # 1. load the class definitions
9  execfile('ImageAnal.py')
10
11 # 2. tell python where the data is
12 datapath = os.getcwd() + '/' #current working directory - look for data and strings here!
13 date_time = '2020-03-13_12-15-01' # date and time of the imaging session
14 name = 'rn013' # mouse name
15 task = 'contingency_learning' # task name
16
17 ## locate the suite2p folder
18 suite2p_folder = datapath + 'data/' + name + '_imaging/rn013_TSeries-03132020-0939-003/'
19
20 ## the name and location of the imaging log file
21 imaging_logfile_name = suite2p_folder + 'rn013_TSeries-03132020-0939-003.xml'
22
23 ## the name and location of the trigger voltage file
24 TRIGGER_VOLTAGE_FILENAME = suite2p_folder + 'rn013_TSeries-03132020-0939-003_Cycle00001_VoltageRecording_001.csv'
25
26
27
28 # 3. load all the data - this takes ~20 secs in my computer
29 # def __init__(self, datapath, date_time, name, task, suite2p_folder, TRIGGER_VOLTAGE_FILENAME):
30 D1 = ImagingSessionData(datapath, date_time, name, task, suite2p_folder, imaging_logfile_name, TRIGGER_VOLTAGE_FILENAME)
31
```

1. Start python (anaconda)
2. Change your working directory in the directory where the python scripts are
`os.chdir("your/Working/Directory")`

- Read the comments start executing the lines one-by one. You need to set a few variables that describes your experiment:
 - `datapath`: this is your current working directory
 - `date_time`: date and time of the recordings
 - `name`: mouse name
 - `task`: task name
 - `suite2p_folder`: the folder that contains the imaging data
 - `imaging_logfile_name`: the name and location of the imaging log file - this should be in the suite2p folder
 - `TRIGGER_VOLTAGE_FILENAME`: the name and location of the trigger voltage file - this should be in the suite2p folder. CAPITAL letters means that this data was recorded using the imaging time axis. All other data will be immediately converted to the labview time axis.

We may simplify this process, but we need to decide how to do it (what are the naming conventions for the `TRIGGER_VOLTAGE_FILE`, `imaging_logfile`, where do you want to put the suite2p files etc.)

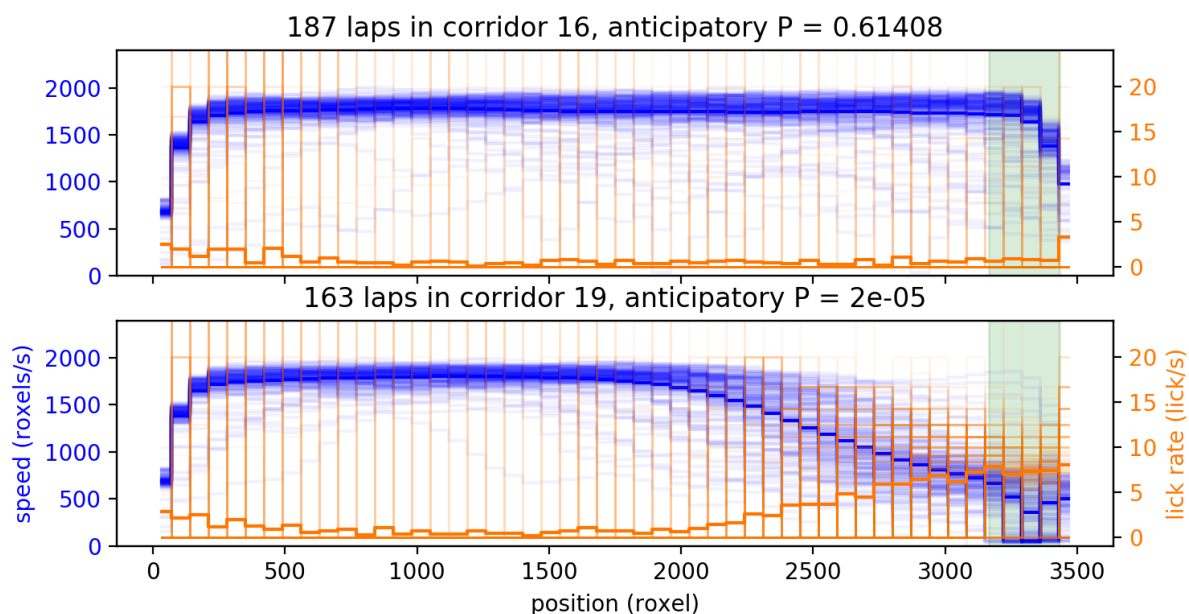
Once you defined the location of the data you can load the data. In this example the data is loaded in a `D1` variable, which is an object from the `ImagingSessionData` class. Lot's of analysis is done when you first load the data, including calculating dF/F , spikes rates and separating data into laps. It has many different methods for visualising the data.

3. Visualisation

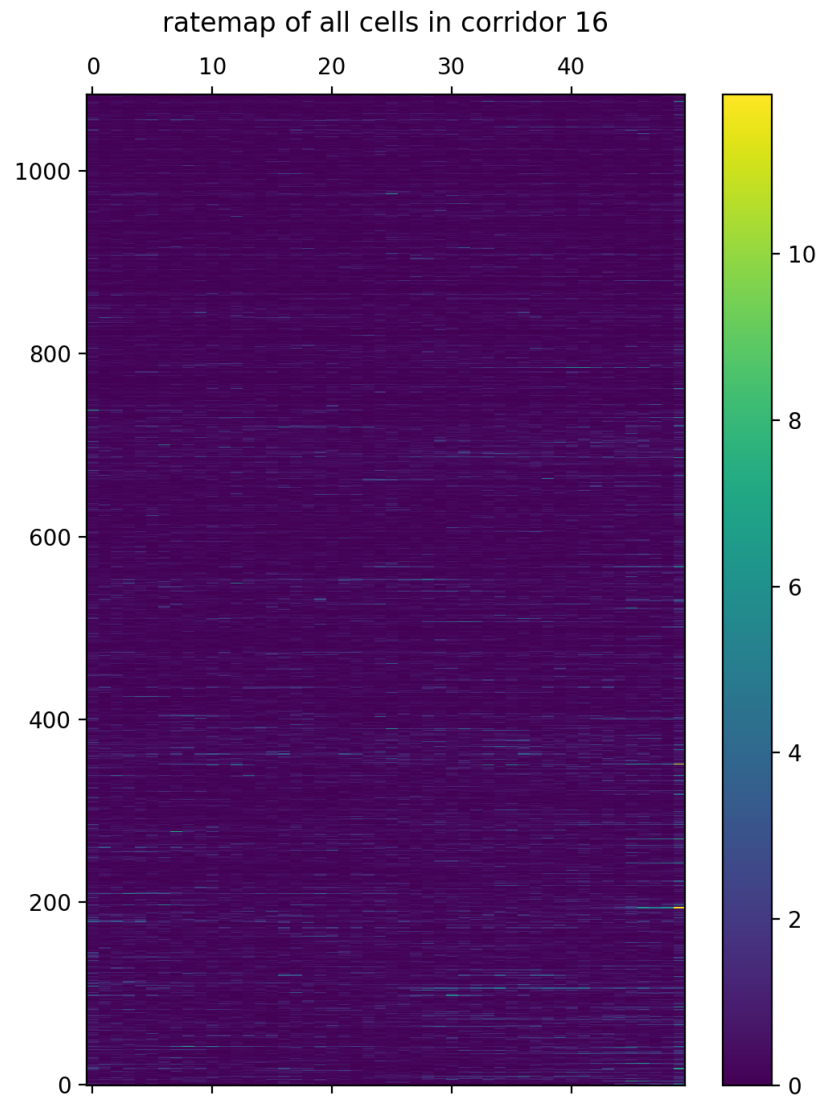
We will discuss the mothods for visualising the data.

Visualising data of the whole session.

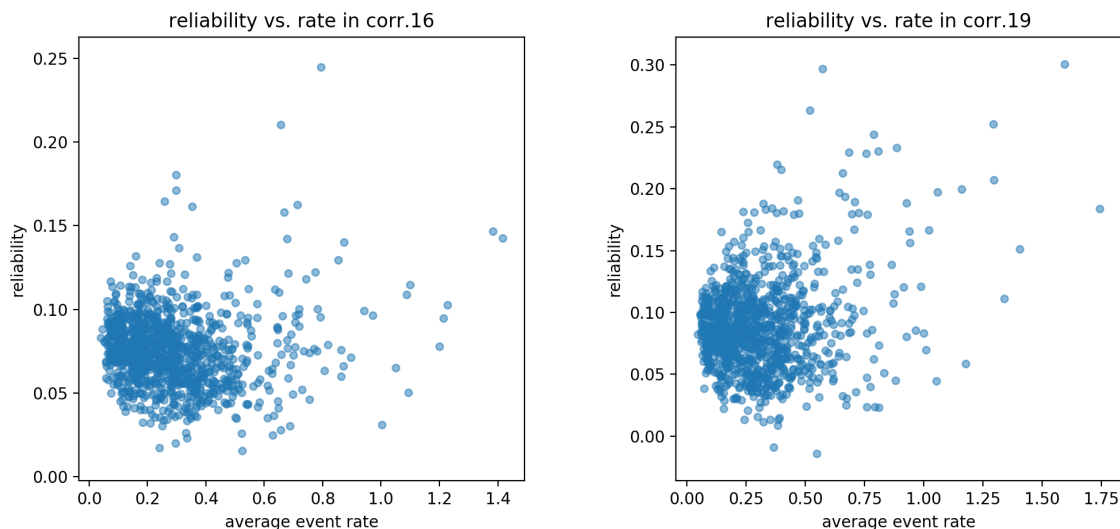
- `D1.plot_session()`: Plotting the behavioral data - this should work as before. You see that corridor 19 is rewarded, and the mouse slows down in that corridor.



2. `D1.plot_ratemaps()` : This function plot the ratemaps of the cells. Without argument it plots all cells an the default (first corridor). It is not very informative, since cells are not ordered, and there are too many.



3. `D1.plot_props(16)`: plotting cell properties - currently reliability as a function of event rate - in a given corridor.



You can select interesting cells, and only plot the ratemap of these cells. Here I select those cells that are reliable in corridor 19. To do this, I use the list `D1.cell_reliability` which contains the reliability of all cells in the two corridors. The vector at index `[0]` belongs to corridor 16 and at index `[1]` belongs to corridor 19.

```
cellids = np.nonzero(D1.cell_reliability[1] > 0.15)[0]
D1.plot_ratemaps(corridor=19, cellids = cellids)
```

Here `cellids` contains the indices of the selected cells.

Note, that y axis of this plot is the index of the cell in the vector `cellids`. So if I want to analyse the cell in the 29th row, I need to see the 29th value of the `cellids` vector:

```
cellids[29]
```

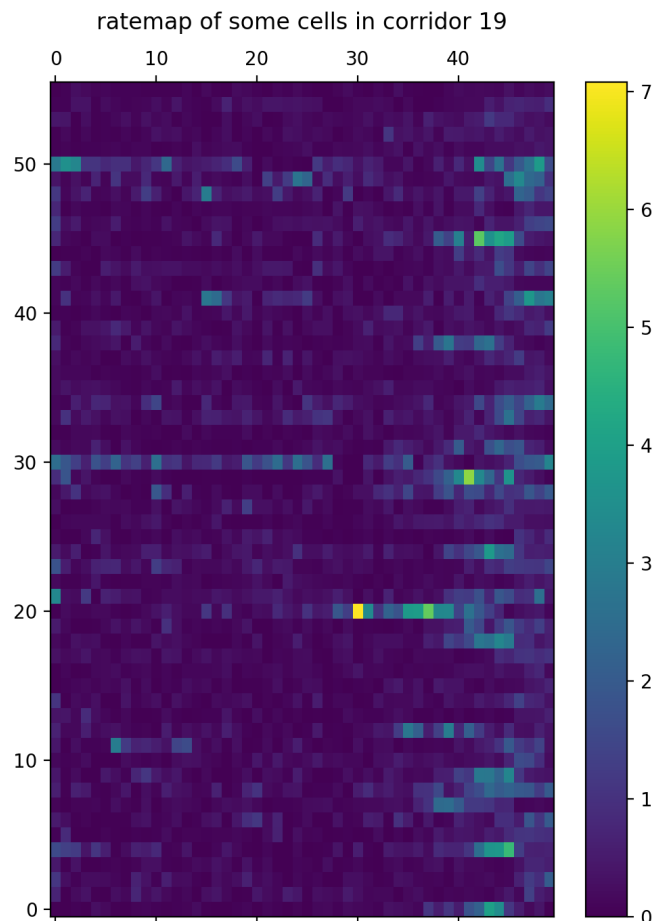
which is 194. So this was the 194th cell. We will analyse this neuron in the remaining part of this example.

Its reliability is high in both corridors:

```
D1.cell_reliability[0][194]: 0.245
D1.cell_reliability[1][194]: 0.242
```

And its rate is also similarly high:

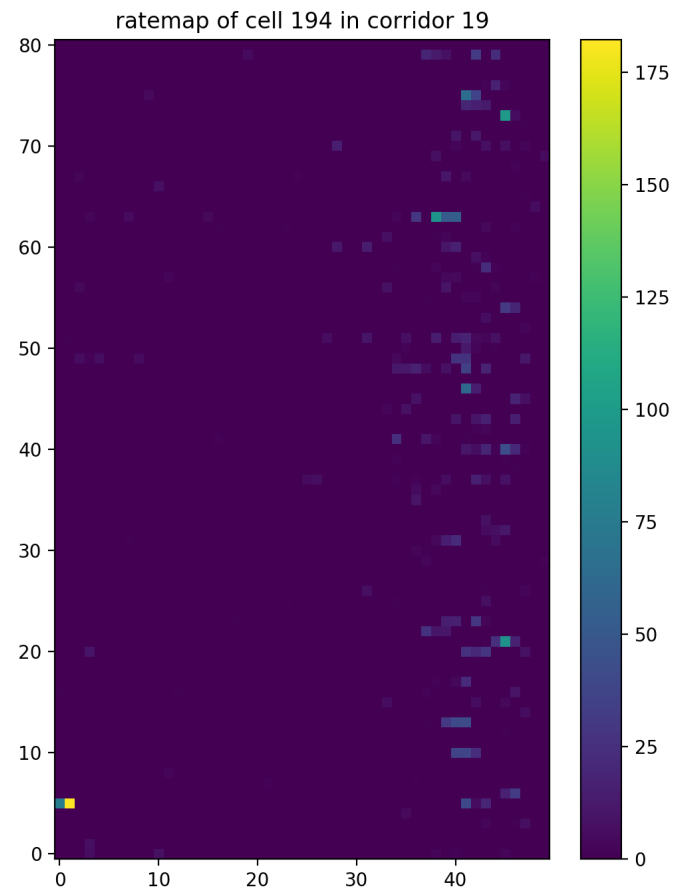
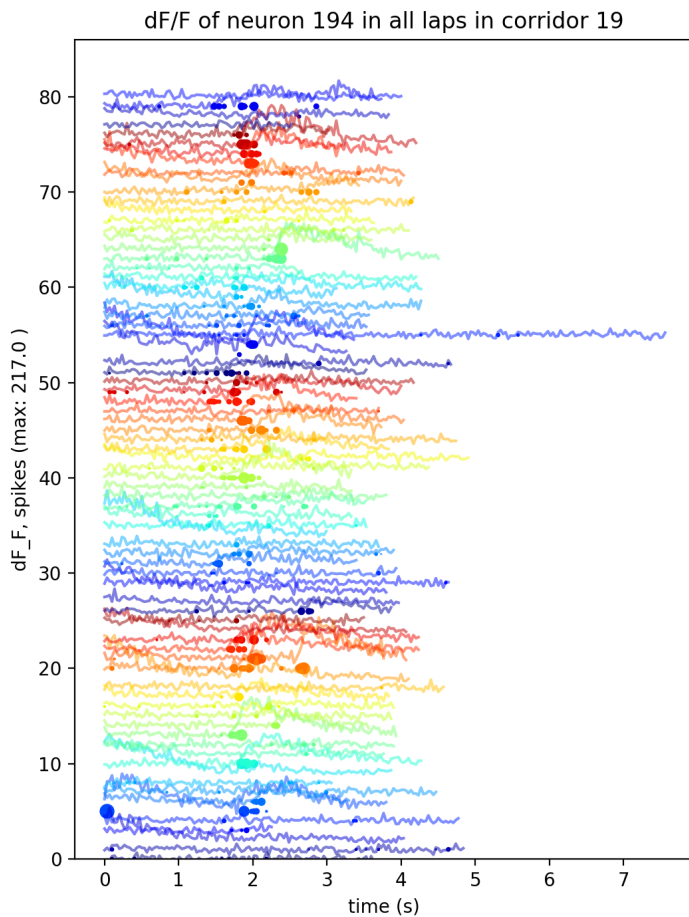
```
D1.cell_rates[1][194]: 0.791
D1.cell_rates[0][194]: 0.794
```



4. For each cell you can plot its activity in a given corridor in two different ways. Either showing dF/F and spikes as a function of **time** or plotting the event rate as a function of **space**.

```
D1.plot_cell_laps(corridor=19, cellid=194, signal='dF')
```

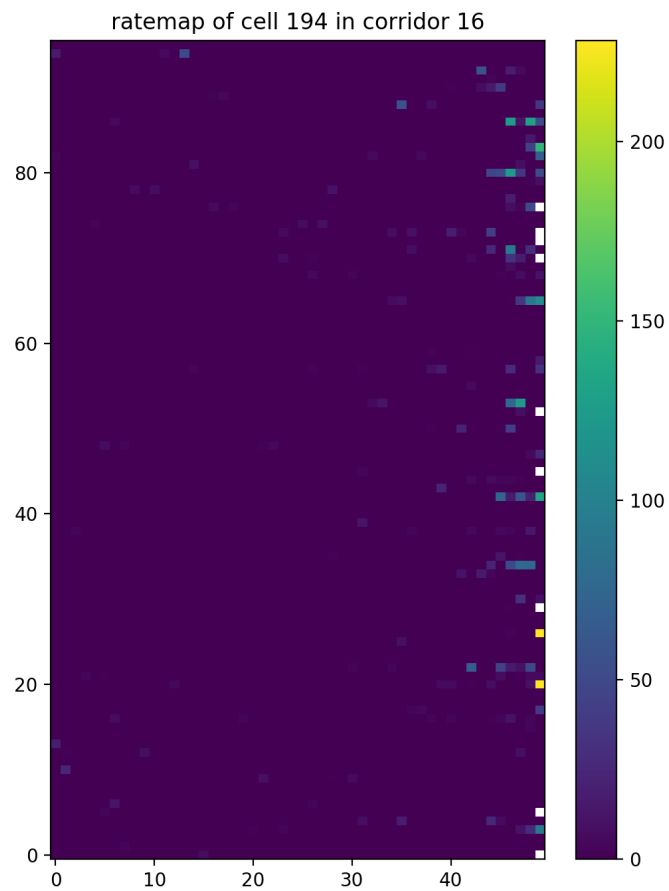
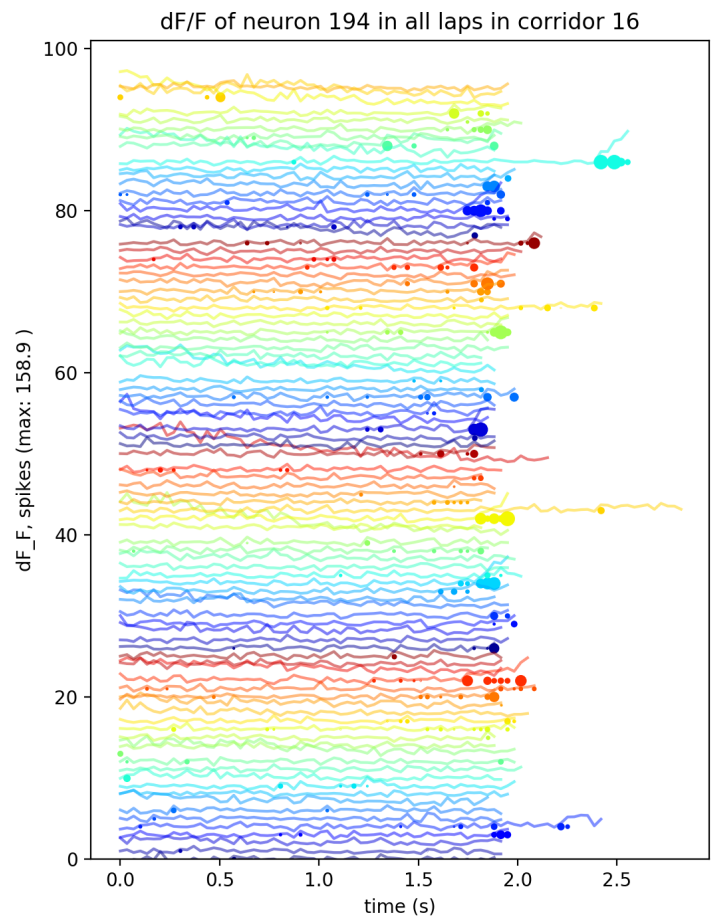
```
D1.plot_cell_laps(corridor=19, cellid=194, signal='rate')
```



Here each row shows an individual lap. Each lap can have a different duration. Remember: lap 19 is a rewarded corridor, and the animal stops at the reward zone. Spikes appearing around 2s are therefore around the end of the maze.

Currently all data is used so there is no speed thresholding. You can zoom in the interactive plots and try to match the laps. As always, there can be bugs in the scripts, so try to find traces that should match but do not. I checked a few cells, but they seemed OK.

The activity of the same neuron in the different corridor:



5. Plotting a single lap's data.

You can either plot as a function of time (fluorescence and spikes) or space (spike rate):

```
D1.ImLaps[188].plot_tx(fluo=True)
```

```
D1.ImLaps[188].plot_xv()
```

