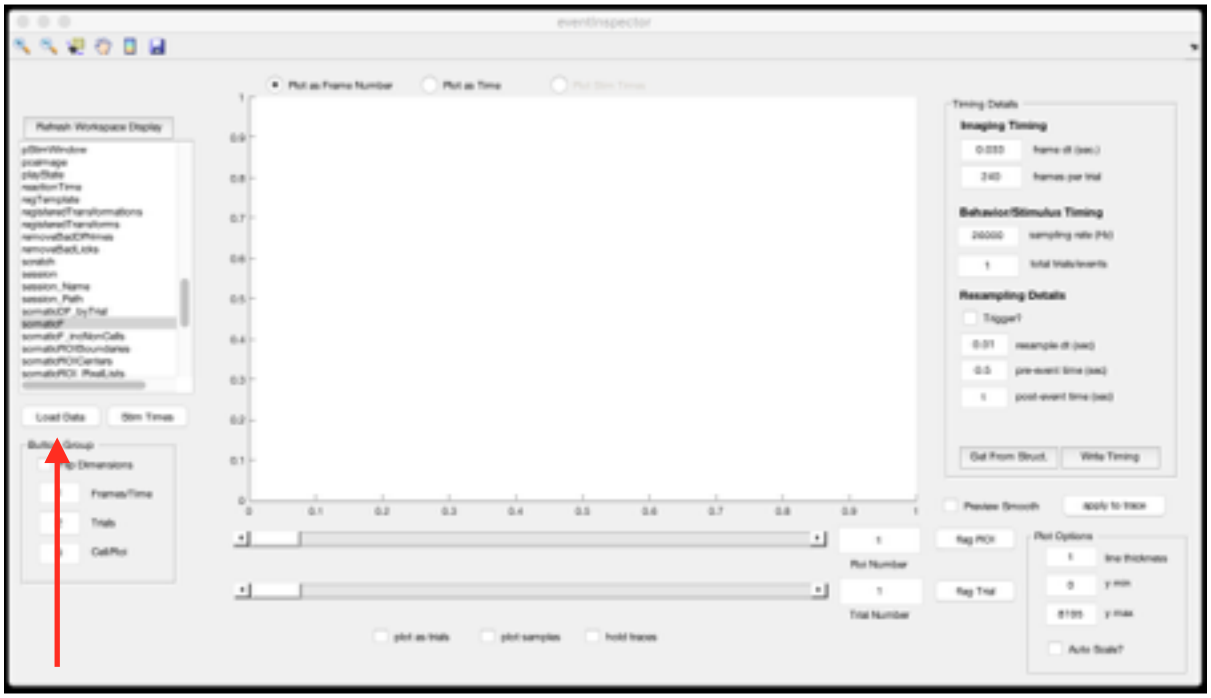
eventInspector

documentation started: 4/10/2016; updated: ; cdeister@brown.edu

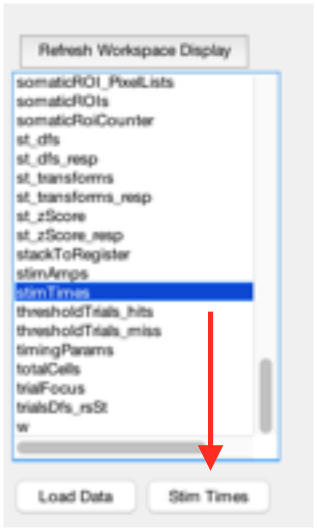
The point of eventInspector is to look at your data in the context of 'events.' Events are most often a stimulus or cue time, but could be anything. The main thing you need to make use of this is a matrix of data, a vector of event times, and some basic knowledge of how your data and events were collected/generated. Optionally, you can configure and use a struct with relevant timing info using fields we will discuss later. For now let's start working through some data.

To open eventInspector, make sure imageAnalysisGui and all subfolders are in your path. If you care about how *eventInspector* works you probably made it this far already.

type 'eventInspector' in the matlab command line. You should see this:



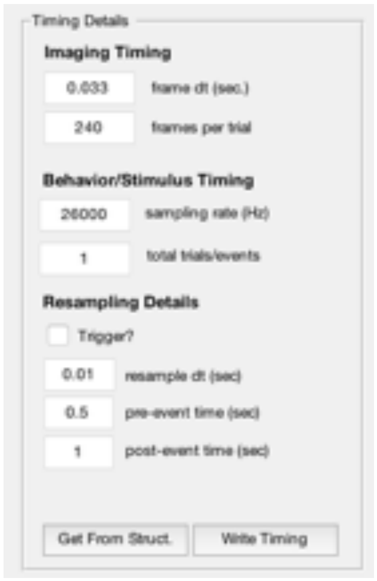
This screenshot is from version 2.9 of the package, so it may not look exactly like this, but it will be close. The first thing you should do is select a matrix from the 'workspace browser.' Here I have selected a matrix that I have extracted with extractor from a bunch of somas called 'somaticF', this is the default from *extractor*, so you probably have one of these too. eventInspector works best with a 2-D matrix that contains M(rois) x N(frames), but you can do the opposite, and let eventInspector know, by toggling the 'Flip Dimensions' button that is slightly obscured by my red arrow. Once you know what you want to load hit the 'Load Data' button that I've pointed to above with the red arrow.



When you do this, you will see a trace plotted for the first roi and all frames. From then on, eventInspector understands what you are working on so you can choose something else from the workspace browser. Go ahead and now select a vector of event times, and then click the button called 'Stim Times' next to the load button:

The screenshot to the right shows the button called out in red. Once you click 'Stim Times' this will load the times into a struct in your workspace called 'scratch.' The whole point of scratch is to store relevant temp variables, that you don't need to worry about. Point is eventInspector now 'knows' these times. It then tries to help you out and make some assumptions, or figure out some of your timing information for you. Don't worry if it gets it wrong you will be able to modify things. Speaking of which let's turn our attention to that real quick.

On timing. At this point the necessary timing details you need to see something are minimal, but here is what happens when you load stim times. The left image below shows the initial state of the 'Timing Details' portion of the program. Once I load my 'stimTimes' vector, which is a 253 element vector of times, some fields update as you can see on the right.



Timing Details

Imaging Timing

0.033 frame dt (sec.)
240 frames per trial

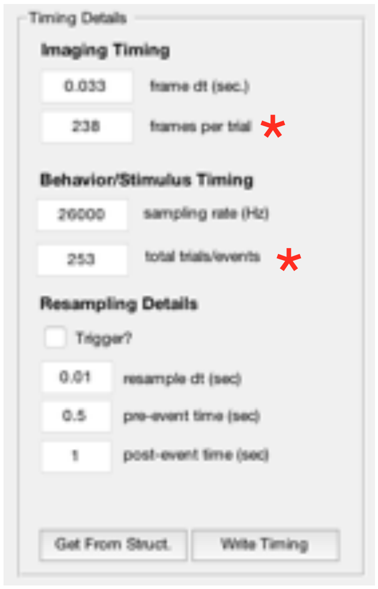
Behavior/Stimulus Timing

26000 sampling rate (Hz)
1 total trials/events

Resampling Details

Trigger?
0.01 resample dt (sec)
0.5 pre-event time (sec)
1 post-event time (sec)

Get From Struct. | Write Timing



Timing Details

Imaging Timing

0.033 frame dt (sec.)
238 frames per trial *

Behavior/Stimulus Timing

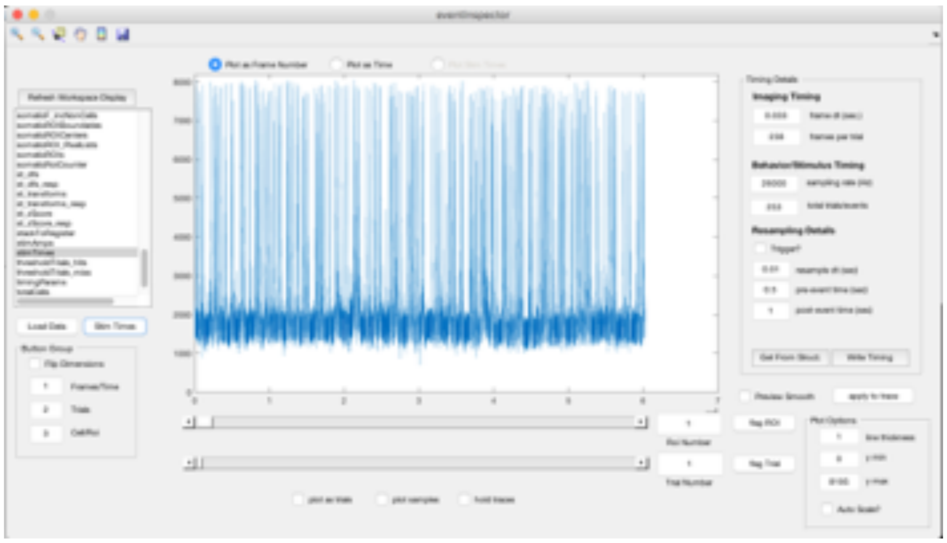
26000 sampling rate (Hz)
253 total trials/events *

Resampling Details

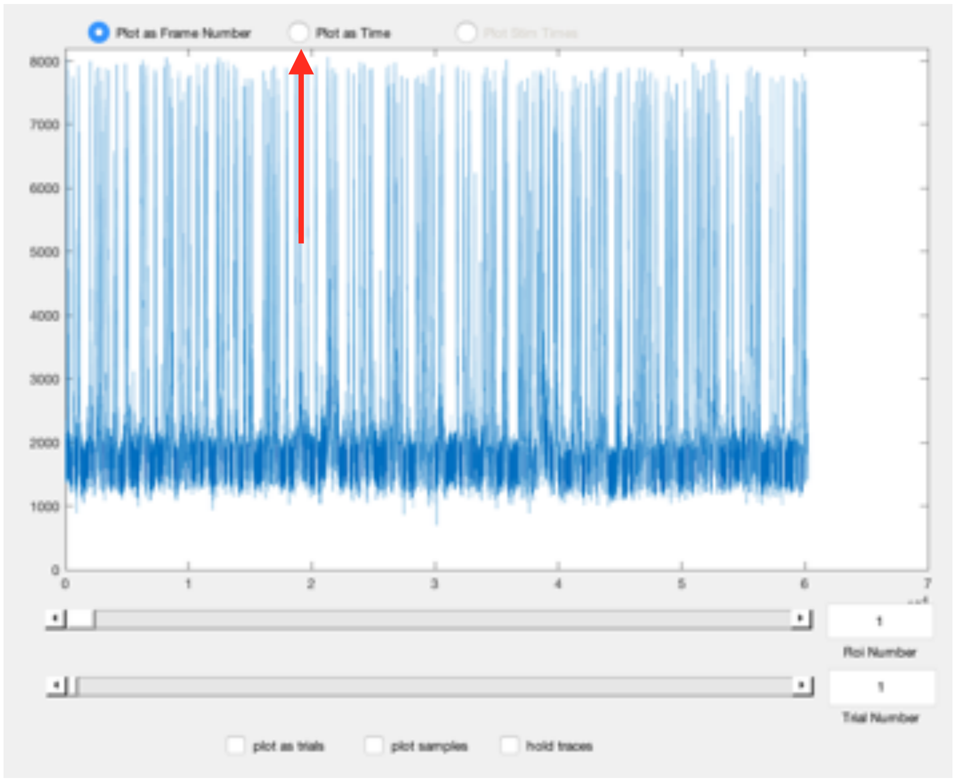
Trigger?
0.01 resample dt (sec)
0.5 pre-event time (sec)
1 post-event time (sec)

Get From Struct. | Write Timing

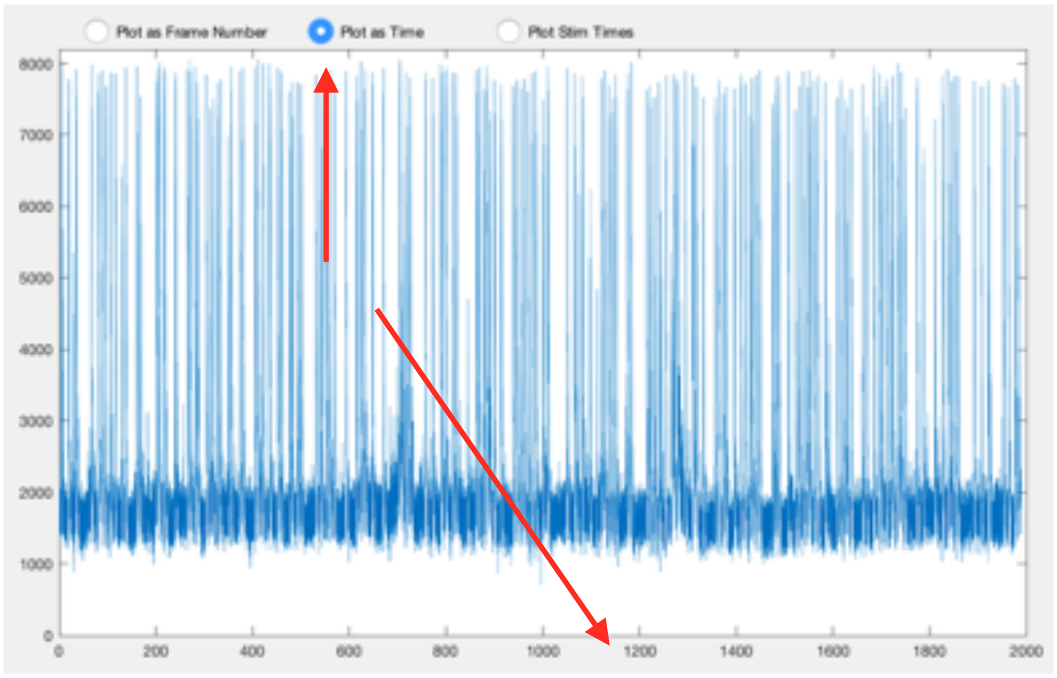
You can, and likely, deliver your stimuli differently, but what I say should generalize. I collect trials of images whose onset are synchronized to a behavior control computer that uses standard AD/DA boards to generate and deliver stimuli. I have 8 second long trials in this example, but within the 8 seconds the stimuli came on at variable times, which are listed in 'stimTimes'. So trial 1 will have a time of 1.23 seconds, but trial 2 might be 2.6 seconds. eventInspector will use the number of stimTimes as my 'total trials/events' value denoted above with a red star. My data in this example is a 24 x 60214 matrix. There are 24 neurons and 60214 frames. The imaging/behavioral session was broken up into 253 trials, so the number of frames per trial were 60214/253, which comes to 238 and is calculated and entered for you in the 'frames per trial' entry box also pointed out with a red star. If it is wrong, feel free to enter the real value. You should now see something like this:



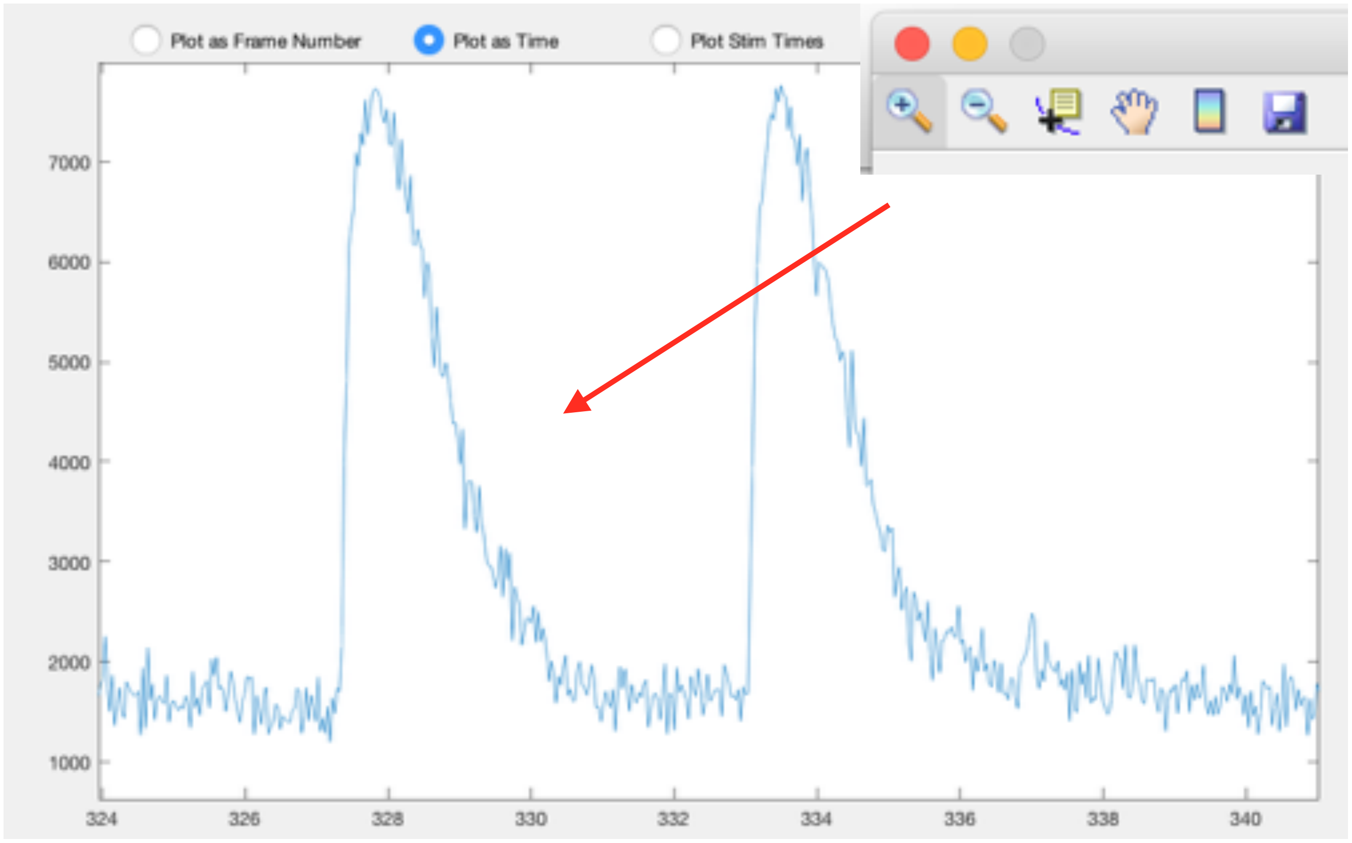
Now let's turn our attention to the trace and how it is plotted:

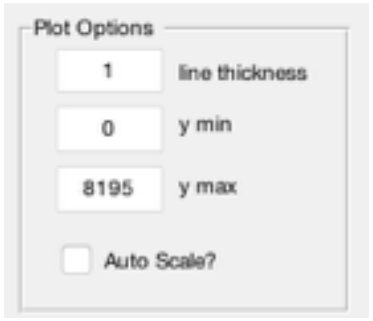


Here you will see that the initial plotting is done for Roi Number 1, with the Y values being the raw values I loaded (raw fluorescence values in time), and the X values are frame number. The frame interval is specified in the timing window, so if you click 'Plot as Time' (red arrow) you will now see your trace plotted in time (seconds):



In this example my data are an imaging frame's values for that ROI across 60214 frames collected at a sampling interval (frame interval) of 0.033 seconds, which I had specified in the timing window. It therefore made a time vector from the first point in time to 60214 * 0.033, which is 1987.062 seconds. Now the standard matlab plot tools are in eventInspectors toolbar so you can zoom in with those:





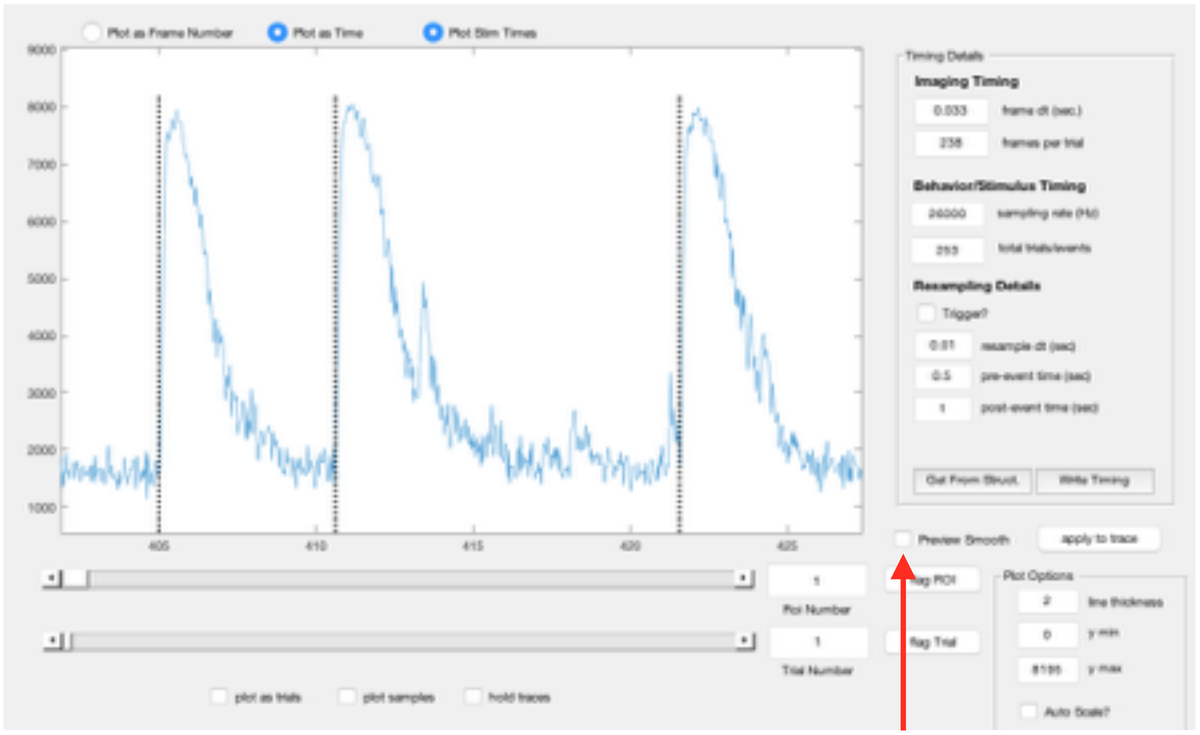
1	line thickness
0	y min
8195	y max

☐ Auto Scale?

You can also change the y-axis values using the 'Plot Options' entries (look to the left). It will default to using the 'y min' and 'y max' values in these fields, which is the valid range for a 16-bit image (2^16; minus one for 0). But, you can also toggle 'Auto Scale?' and it will find the min and max of the ROI's data you have loaded and use those as its bounds.

The other thing to realize is that if you look at the trace above you will see to the right of 'Plot as Time' and option called 'Plot Stim Times' is available. It is grayed out and unavailable to you until you load some stim times. It is also unavailable to you if you are plotting data as frame number. If you toggle this it will display some dashed lines over where your stim times are, the lines will have a thickness of 1, by default, but this can be changed by changing 'line thickness' in the Plot Options you see to the left.

Once you do toggle 'Plot Stim Times' you will now see:



Timing Details

Imaging Timing

0.033	frame dt (sec.)
238	frames per trial

Behavior/Stimulus Timing

26000	sampling rate (Hz)
253	total trials/events

Resampling Details

Trigger?

0.01	resample dt (sec)
0.5	pre-event time (sec)
1	post-event time (sec)

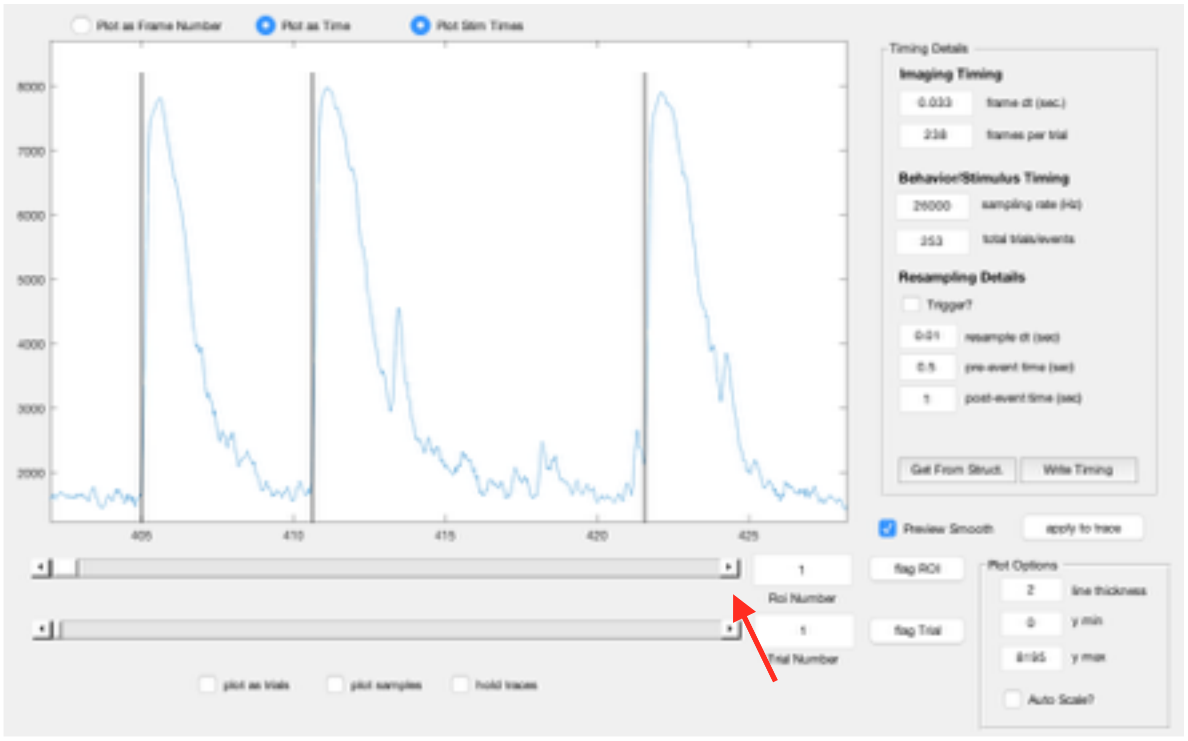
Roi Number: 1

Trial Number: 1

Plot Options

2	line thickness
0	y min
8195	y max

The dashed black lines are stimulus times, and the blue traces are still the data. Note that I changed the thickness of the stim lines to 2. Now I am going to toggle 'Preview Smooth,' as noted by the arrow:



Timing Details

Imaging Timing

0.033	frame dt (sec.)
238	frames per trial

Behavior/Stimulus Timing

26000	sampling rate (Hz)
253	total trials/events

Resampling Details

Trigger?

0.01	resample dt (sec)
0.5	pre-event time (sec)
1	post-event time (sec)

[✓ Preview Smooth]

Roi Number: 1

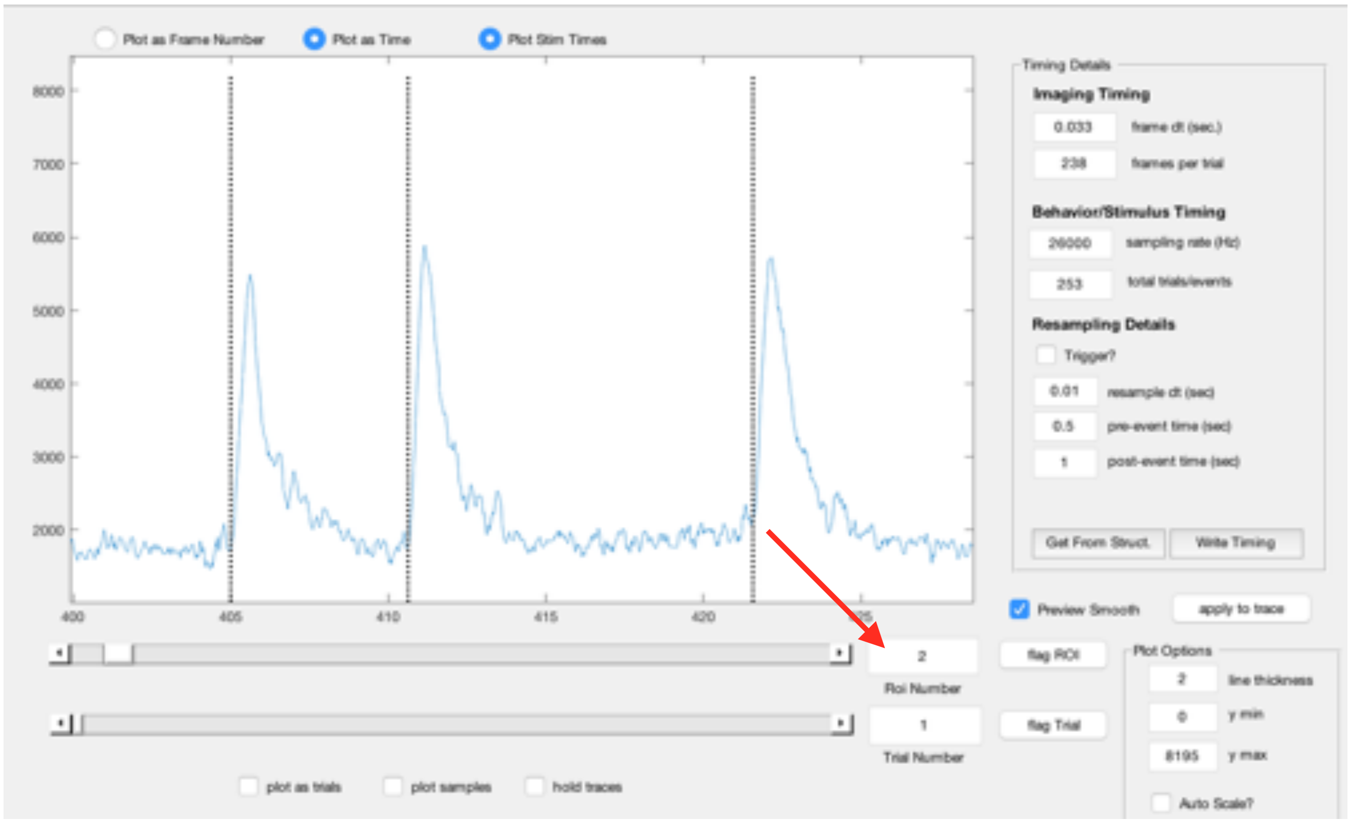
Trial Number: 1

Plot Options

2	line thickness
0	y min
8195	y max

Which applies a simple low-pass filter to the data. This can be left on, and you can save the results by clicking 'apply to trace,' which will save the filtered version to your matrix (more on that later).

If you toggle the 'Roi Number' (red arrow) you will get the next cell:



Plot as Frame Number [✓ Plot as Time] [✓ Plot Stim Times]

Timing Details

Imaging Timing

0.033 frame dt (sec.)

238 frames per trial

Behavior/Stimulus Timing

26000 sampling rate (Hz)

253 total trials/events

Resampling Details

Trigger?

0.01 resample dt (sec)

0.5 pre-event time (sec)

1 post-event time (sec)

Get From Struct. | Write Timing

[✓ Preview Smooth] apply to trace

2 Roi Number | flag ROI

1 Trial Number | flag Trial

Plot Options

2 line thickness

0 y min

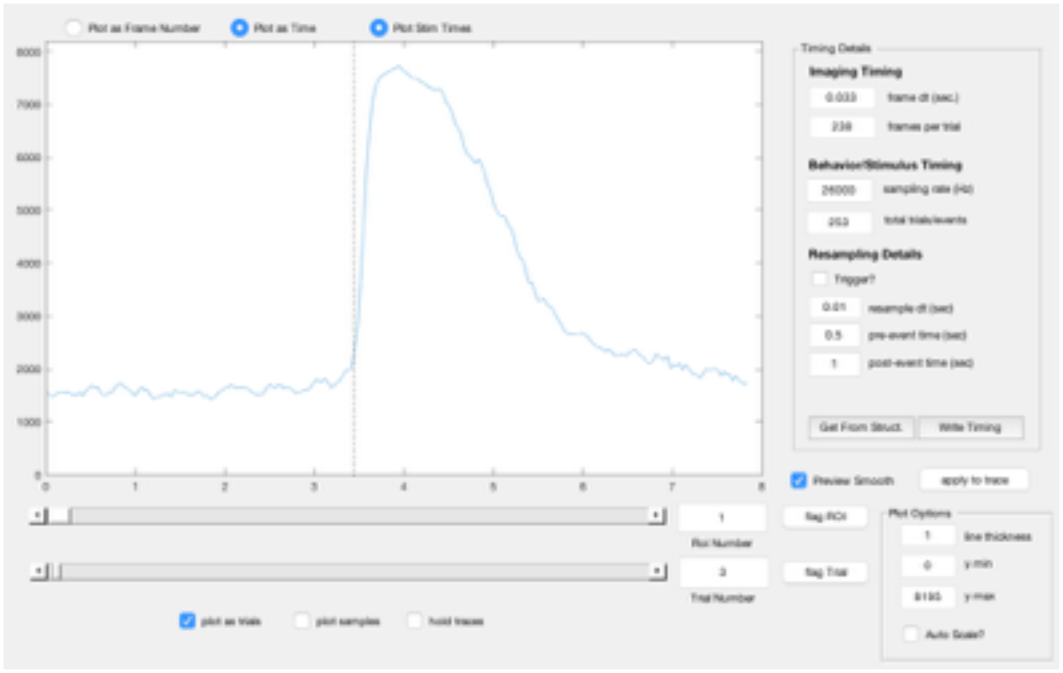
8195 y max

Auto Scale?

plot as trials | plot samples | hold traces

This is roi number 2, but similar time etc. to the one above. Now hitting the arrow, on the slider, changing the number and hitting enter, as well as clicking in the 'dead zone' of the slider itself will all change the roi by 1, do not drag the slider itself, as it does not 'scrub.' Instead, it will move in non-integer steps and throw and error. As far as I can tell, this is default behavior in matlab and I can't override it without subclassing that GUI element, which is not worth it.

Ok now let's look at how individual trials look. Select the button 'plot as trials' below the Trial Number slider:



Plot as Frame Number [✓ Plot as Time] [✓ Plot Stim Times]

Timing Details

Imaging Timing

0.033 frame dt (sec.)

238 frames per trial

Behavior/Stimulus Timing

26000 sampling rate (Hz)

253 total trials/events

Resampling Details

Trigger?

0.01 resample dt (sec)

0.5 pre-event time (sec)

1 post-event time (sec)

Get From Struct. | Write Timing

[✓ Preview Smooth] apply to trace

1 Roi Number | flag ROI

3 Trial Number | flag Trial

Plot Options

1 line thickness

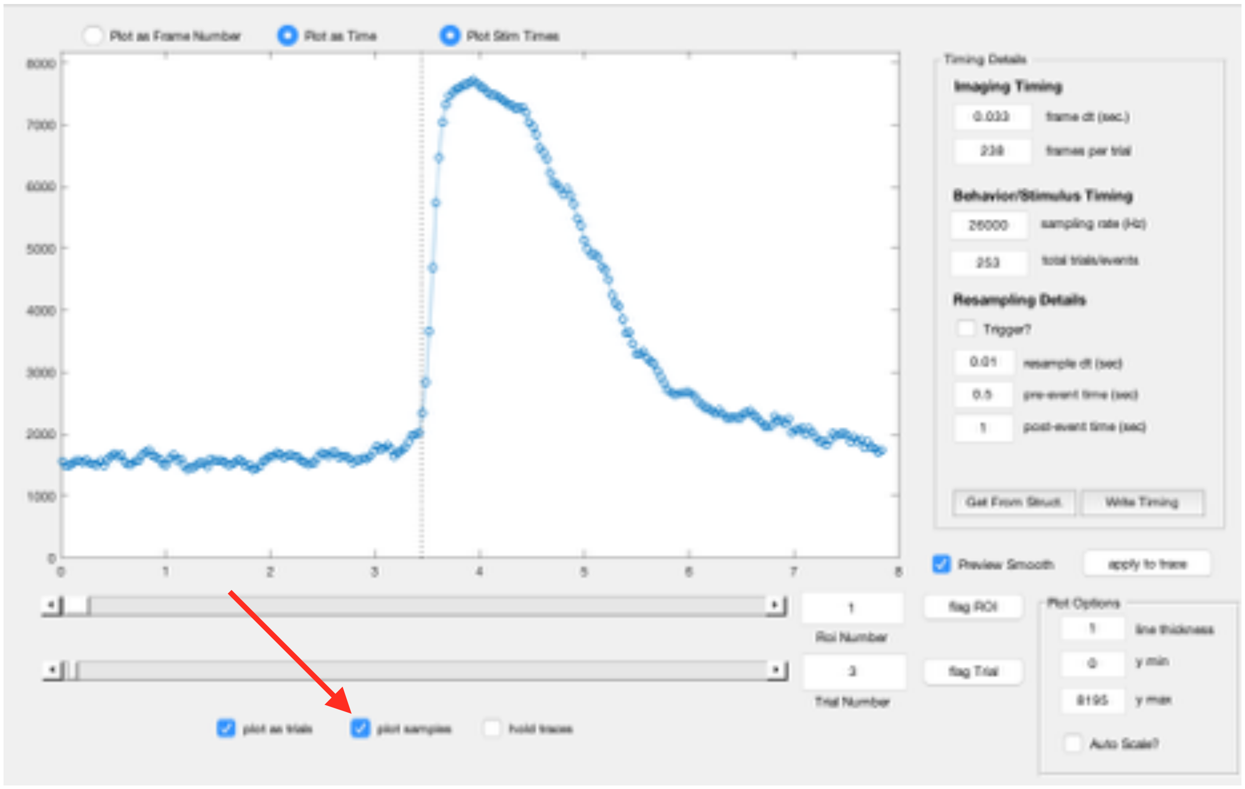
0 y min

8195 y max

Auto Scale?

[✓ plot as trials] | plot samples | hold traces

Now you can see just a single trial, in this case trial number 3, as you can see on the trial number slider. The x-axis is time because I can see the stim time mark (dashed black line), and the Plot as Time option is selected. If you select 'plot samples' you will see the points that make up the trace, this can slow things down so don't leave it on, but use it when you need it:



Plot as Frame Number [ ] Plot as Time [✓] Plot Stim Times [✓]

Imaging Timing

0.033	frame dt (sec.)
238	frames per trial

Behavior/Stimulus Timing

26000	sampling rate (Hz)
253	total trials/events

Resampling Details

Trigger? [ ]

0.01	resample dt (sec)
0.5	pre-event time (sec)
1	post-event time (sec)

Preview Smooth [✓]

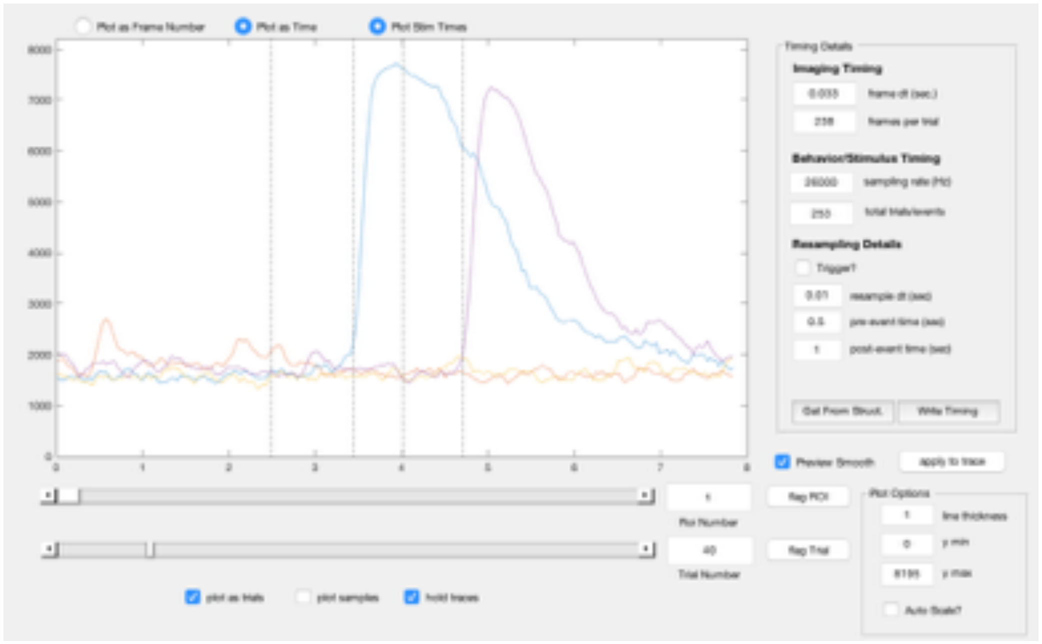
1	Roi Number
3	Trial Number

1	line thickness
0	y min
8195	y max

Auto Scale? [ ]

plot as trials [✓] plot samples [✓] hold traces [ ]

Now select 'hold traces,' this will let you plot a bunch of trial on top of each other. I will overlay trials 3, 10,20 and 40 by entering each number in the trial number box one by one:



Plot as Frame Number [ ] Plot as Time [✓] Plot Stim Times [✓]

Imaging Timing

0.033	frame dt (sec.)
238	frames per trial

Behavior/Stimulus Timing

26000	sampling rate (Hz)
253	total trials/events

Resampling Details

Trigger? [ ]

0.01	resample dt (sec)
0.5	pre-event time (sec)
1	post-event time (sec)

Preview Smooth [✓]

1	Roi Number
40	Trial Number

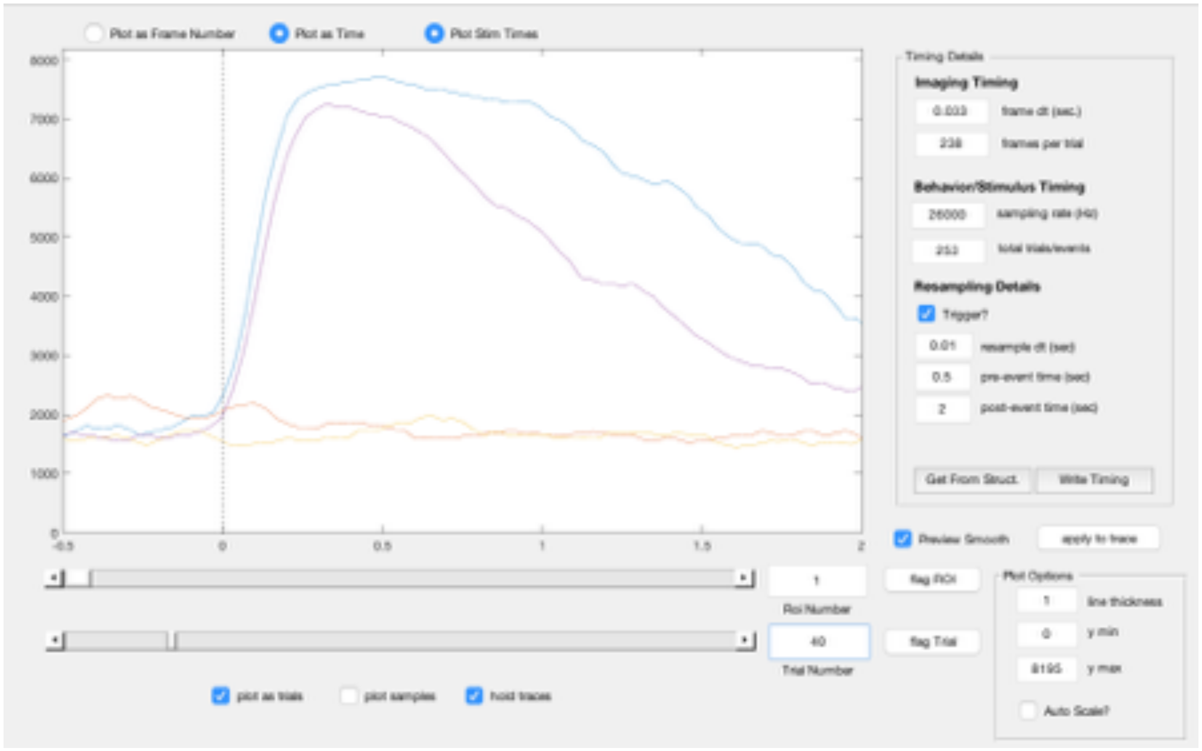
1	line thickness
0	y min
8195	y max

Auto Scale? [ ]

plot as trials [✓] plot samples [ ] hold traces [✓]

Notice that

whole we can see a few traces, the timing is all over the place. This is due to the way in which I deliver stimuli (the timing is variable). I can select 'Trigger' in the Timing Details panel to plot data triggered to the stimulus time. The 'pre-event time' option and 'post-even time' option allow for you to specify the time before and after the stim to look at too:



Plot as Frame Number [✓ Plot as Time] [✓ Plot Stim Times]

Timing Details
Imaging Timing
0.033 frame dt (sec.)
238 frames per trial
Behavior/Stimulus Timing
26000 sampling rate (Hz)
253 total trials/events
Resampling Details
[✓ Trigger?]
0.01 resample dt (sec)
0.5 pre-event time (sec)
2 post-event time (sec)

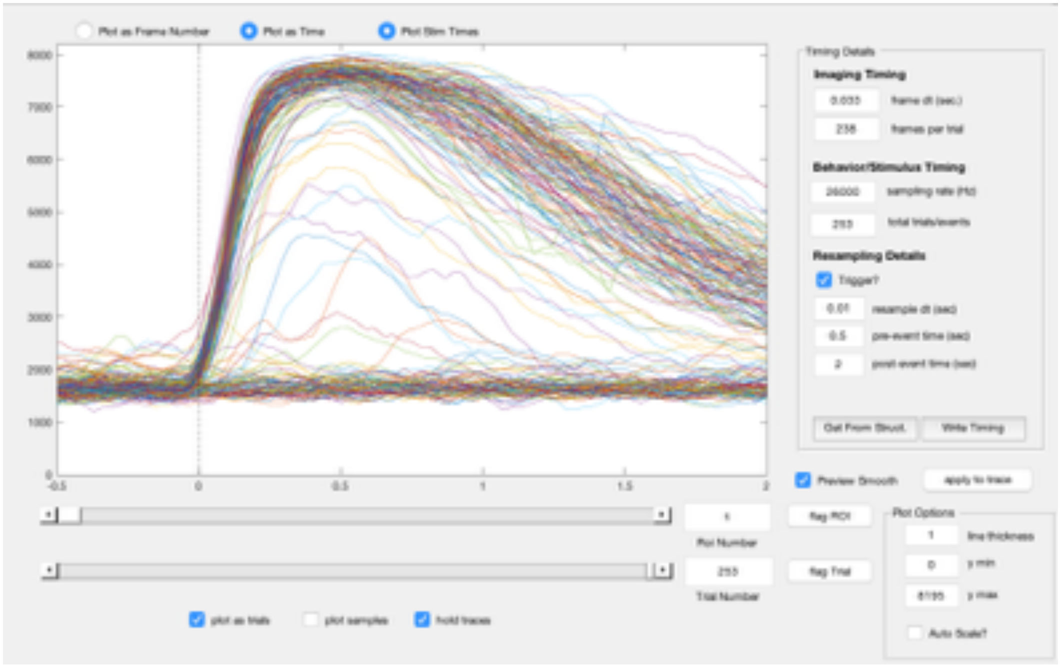
[✓ Preview Smooth]

1 Roi Number
40 Trial Number

Plot Options
1 line thickness
0 y min
8195 y max
Auto Scale?

[✓ plot as trials] plot samples [✓ hold traces]

To make this I just plotted trials 3,10,20 and 40 again, keeping hold traces on, and triggering at the event, with 500 ms before (negative time values), and 2 seconds after (stim onset is time 0). The neuron fired some action potentials on trials 3 and 40 (blue and purple), but not 10 and 20 (orange and yellow). If I just mash down on the slider arrow and collect all the trials this is what we see for this neuron:



Plot as Frame Number [✓ Plot as Time] [✓ Plot Stim Times]

Timing Details
Imaging Timing
0.033 frame dt (sec.)
238 frames per trial
Behavior/Stimulus Timing
26000 sampling rate (Hz)
253 total trials/events
Resampling Details
[✓ Trigger?]
0.01 resample dt (sec)
0.5 pre-event time (sec)
2 post-event time (sec)

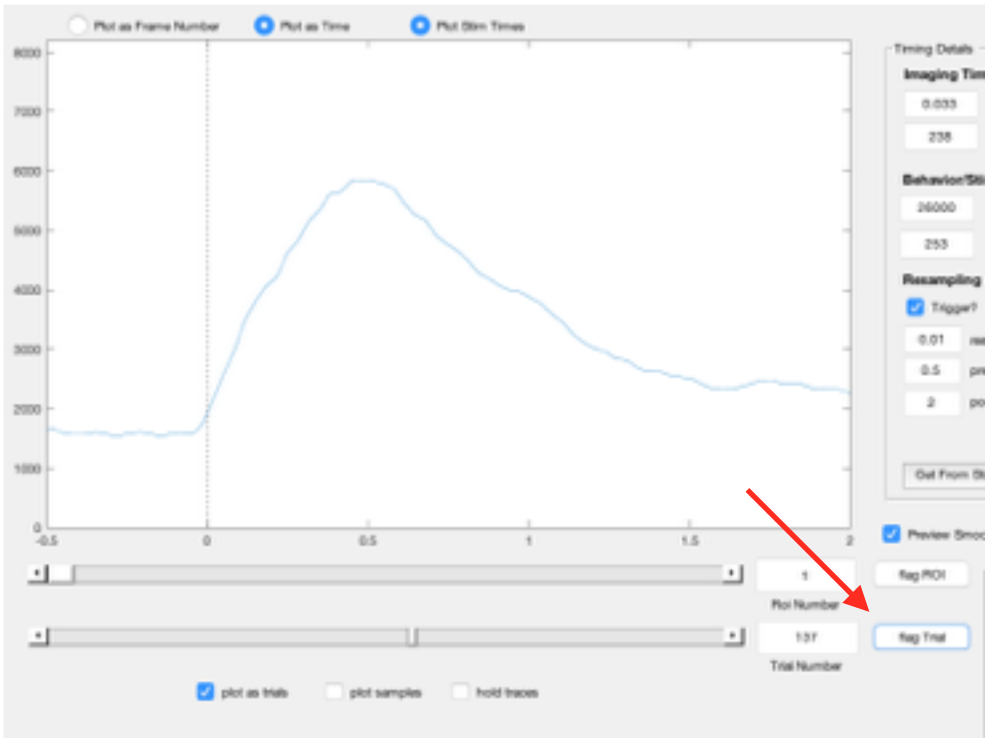
[✓ Preview Smooth]

1 Roi Number
253 Trial Number

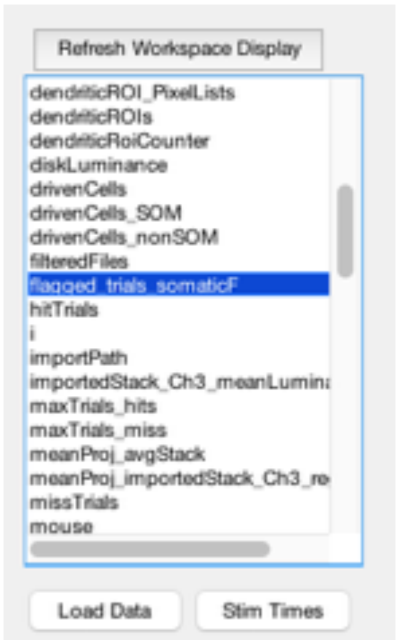
Plot Options
1 line thickness
0 y min
8195 y max
Auto Scale?

[✓ plot as trials] plot samples [✓ hold traces]

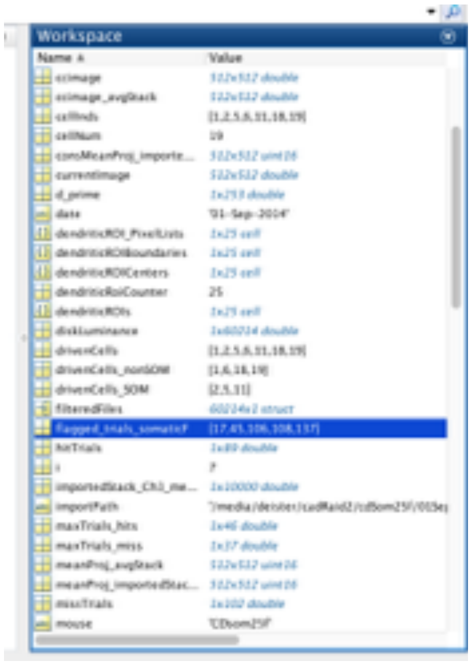
Now I want to turn your attention to a useful feature and that is the ‘flag ROI’ and ‘flag Trial.’ If you encounter an ROI that just looks particularly good or bad, you can ‘flag’ it. What you do with this is up to you, but I use it to track specific things later. For example, lets say I was curious about the intermediate range of responses we see in the above example. Most of the responses where close to max (~8000) and others were flat. Many though were in the middle. Maybe I want to have a record of which those are so I can inspect them closer later, I can flag them when I see them:



I flagged a few and now I have a matrix in my workspace called ‘flagged_somaticF_trials:’

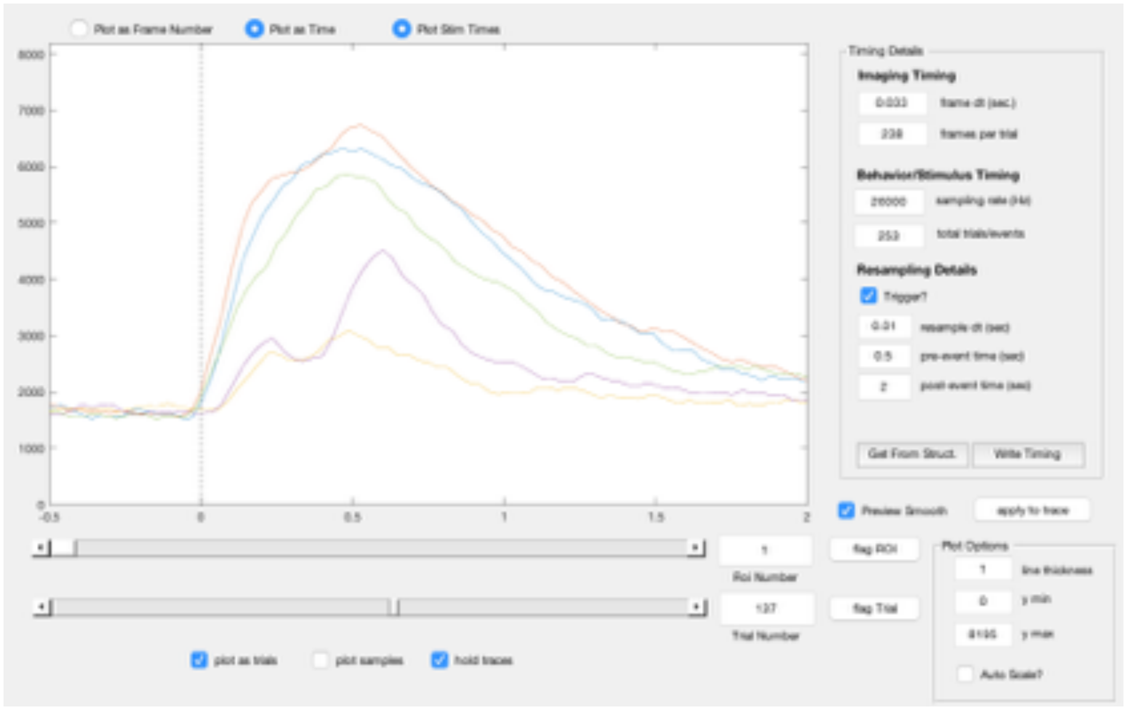


dendriticROI_PixelLists
dendriticROIs
dendriticRoiCounter
diskLuminance
drivenCells
drivenCells_SOM
drivenCells_nonSOM
filteredFiles
flagged_trials_somaticF
hitTrials
i
importPath
importedStack_Ch3_meanLumin
maxTrials_hits
maxTrials_miss
meanProj_avgStack
meanProj_importedStack_Ch3_re
missTrials
mouse



Name ▲	Value
ccimage	512x512 double
ccimage_avgStack	512x512 double
cellInds	[1,2,5,6,11,18,19]
cellNum	19
concMeanProj_importe...	512x512 uint16
currentImage	512x512 double
d_prime	1x253 double
date	'01-Sep-2014'
dendriticROI_PixelLists	1x25 cell
dendriticROIBoundaries	1x25 cell
dendriticROICenters	1x25 cell
dendriticRoiCounter	25
dendriticROIs	1x25 cell
diskLuminance	1x60214 double
drivenCells	[1,2,5,6,11,18,19]
drivenCells_nonSOM	[1,6,18,19]
drivenCells_SOM	[2,5,11]
filteredFiles	60214x1 struct
flagged_trials_somaticF	[17,45,106,108,137]
hitTrials	1x89 double
i	7
importedStack_Ch3_me...	1x10000 double
importPath	'/media/deisiler/cadRaid2/cdSom25F/015e
maxTrials_hits	1x46 double
maxTrials_miss	1x37 double
meanProj_avgStack	512x512 uint16
meanProj_importedStac...	512x512 uint16
missTrials	1x102 double
mouse	'CDsom25F'

So now I can look at 17,45,106,108 and 137 later:



Plot as Frame Number [✓ Plot as Time] [✓ Plot Stim Times]

Timing Details

Imaging Timing

0.033	frame dt (sec)
238	frames per trial

Behavior/Stimulus Timing

20000	sampling rate (Hz)
253	total trials/events

Resampling Details

[✓ Trigger?]

0.01	resample dt (sec)
0.5	pre-event time (sec)
2	post event time (sec)

[✓ Preview Smooth]

1	Roi Number
137	Trial Number

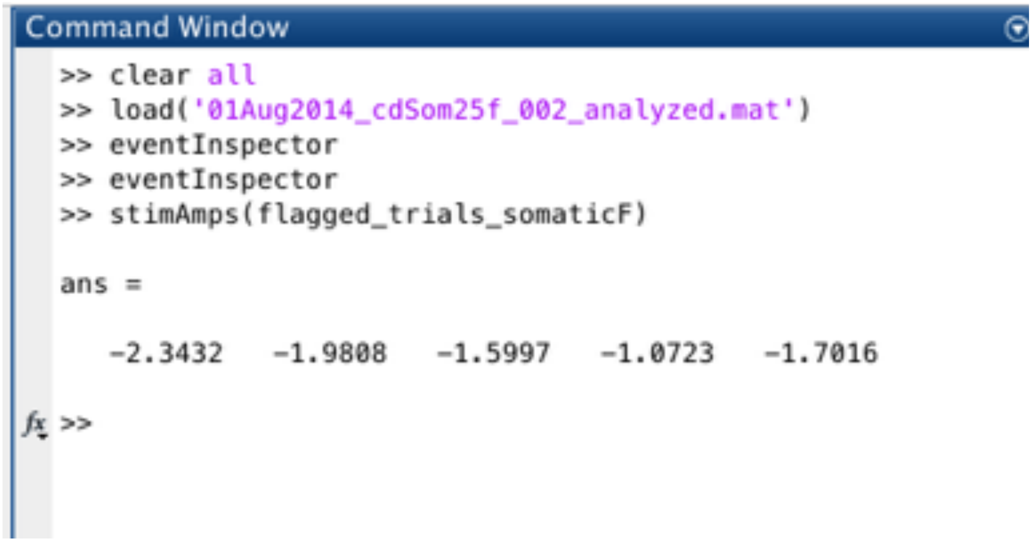
Plot Options

1	line thickness
0	y min
8195	y max

Auto Scale?

[✓ plot as trials] plot samples [✓ hold traces]

I can do things like check my record of stim amplitudes to see what value those were:



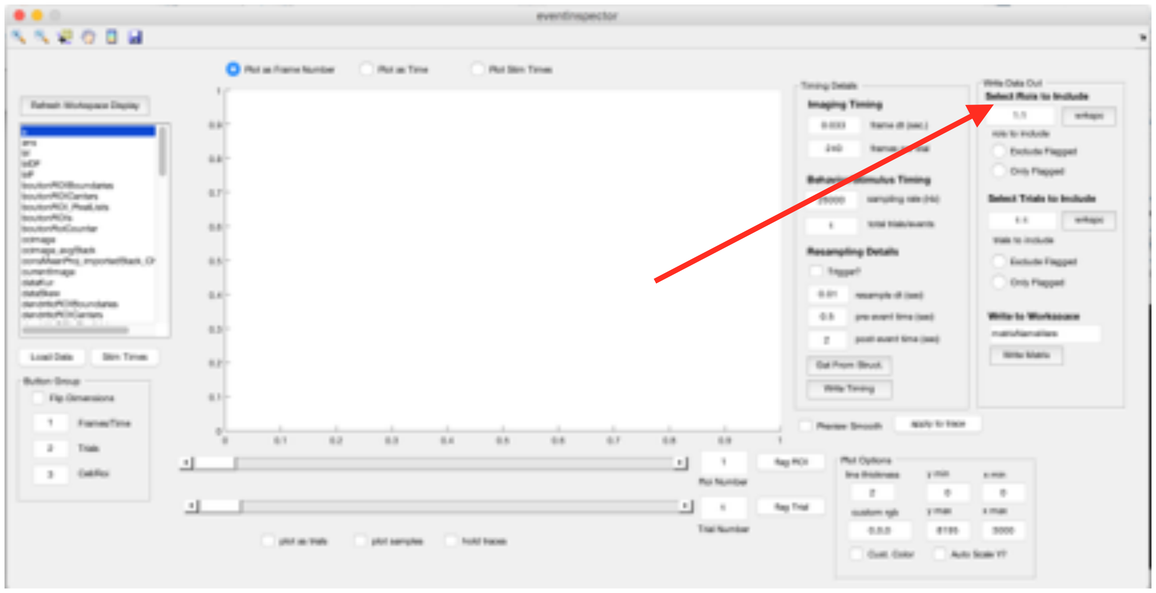
```
Command Window
>> clear all
>> load('01Aug2014_cdSom25f_002_analyzed.mat')
>> eventInspector
>> eventInspector
>> stimAmps(flagged_trials_somaticF)

ans =

   -2.3432   -1.9808   -1.5997   -1.0723   -1.7016

fx >>
```

Now these won't mean much to you, but these are some of my weak whisker deflections (the stimulus I gave to the mouse that contributed the data), these are the command voltages I gave to a whisker deflector and were about a 150 um deflection. In contrast, the larger responses we saw early came from deflections that were usually 3x this amplitude at ~450 um.

We can also similarly flag rois the same way. You can rename these vectors in the workspace and have multiple. This is it for the basics. Now we will move on to saving triggered data out.

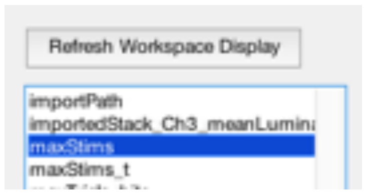
Saving/Analyzing Results:

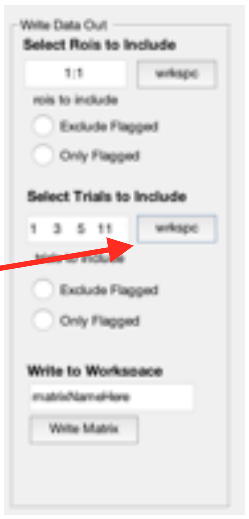
The save portion is a work in progress, but functioning well enough to get real results from your data. On the right of the program is a 'Write Data Out' panel. This is not in the screenshots above so don't worry:



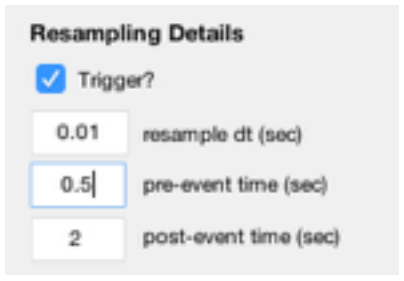
From here you can specify ROIs and trials to write out to the workspace. This is pretty self-explanatory, but some highlights are that you can manually enter the cells and trials with terms that would be appropriate for any matlab index. So 1:1 would be ROI 1, 1:10 would be 1 through 10, 1 2 5 20 would be 1,2,5,20, etc. If you toggle exclude flagged that will happen, but it will exclude from what is entered. If you hit the 'wrkspc' button it will load a vector of integers you have selected in the workspace. It will save to the workspace the time vector and data using the name you enter below and will append _d for the data and _t for the time. Plot options mostly apply so if you want triggered data, you select that as a plot option and specify the time etc. Let's take a look at an example.

I want to see how my first 3 neurons respond to maximal stimuli and 'catch' trials, which were trials with a stimulus amplitude of 0. So, I have made a vector of 'maxStims' and 'noStims,' which are trials of the right type. First let's get trials and a PSTH. I'm going to load the maxStims first:





Highlight maxStims, I made this by doing a maxStims=find(stimAmps==-6.5); where -6.5 was my biggest amplitude and stimAmps was the master record of amplitudes. Then I move over to the Write Data Out panel and click 'wrkspc' in the trial tab, you can now see a random list of trials. I then make sure the triggering params are set the way I want to make my PSTH in the old Timing Details panel:


Resampling Details

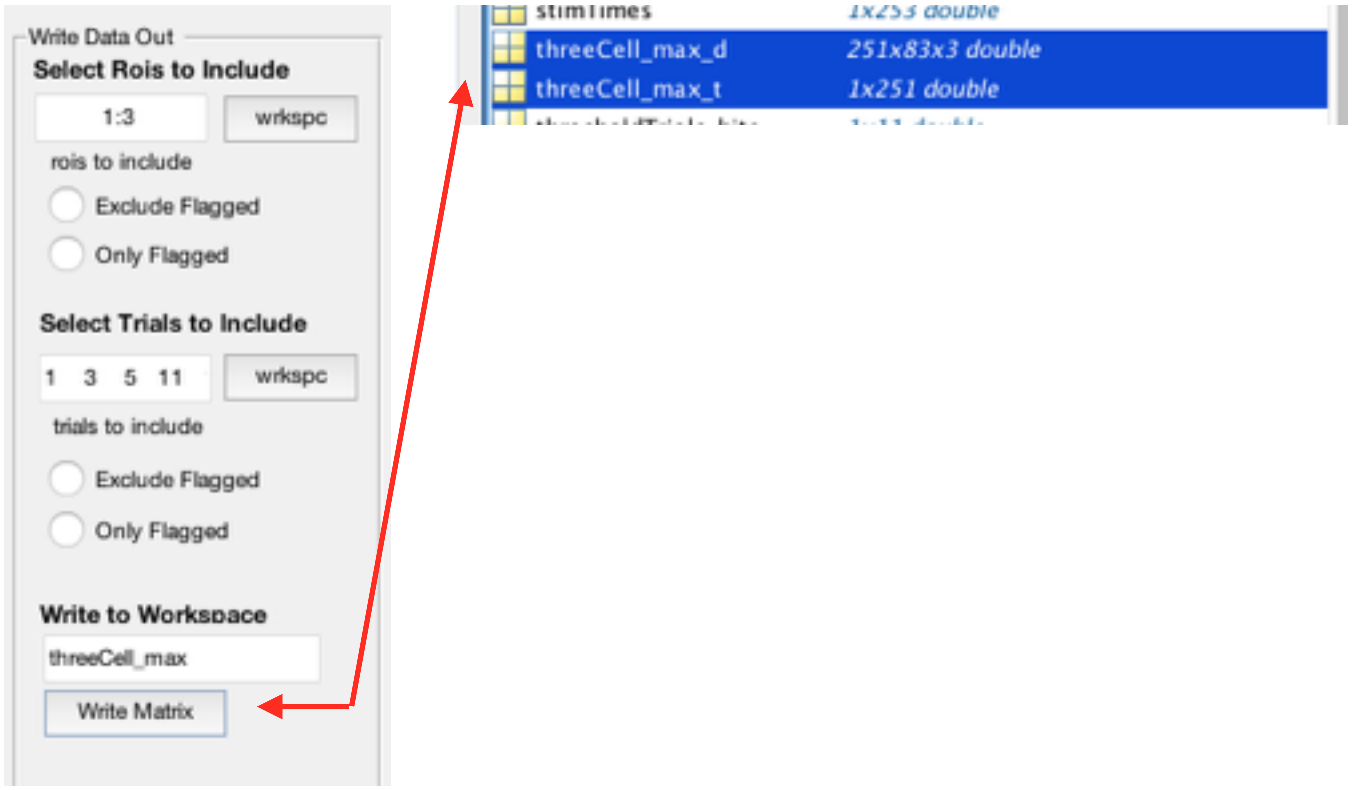
✓ Trigger?

0.01 resample dt (sec)

0.5 pre-event time (sec)

2 post-event time (sec)

Then I name the matrix in the entry and select 'Write Matrix:'


Write Data Out

Select Rois to Include

1:3 wrkspc

rois to include

Exclude Flagged

Only Flagged

Select Trials to Include

1 3 5 11 wrkspc

trials to include

Exclude Flagged

Only Flagged

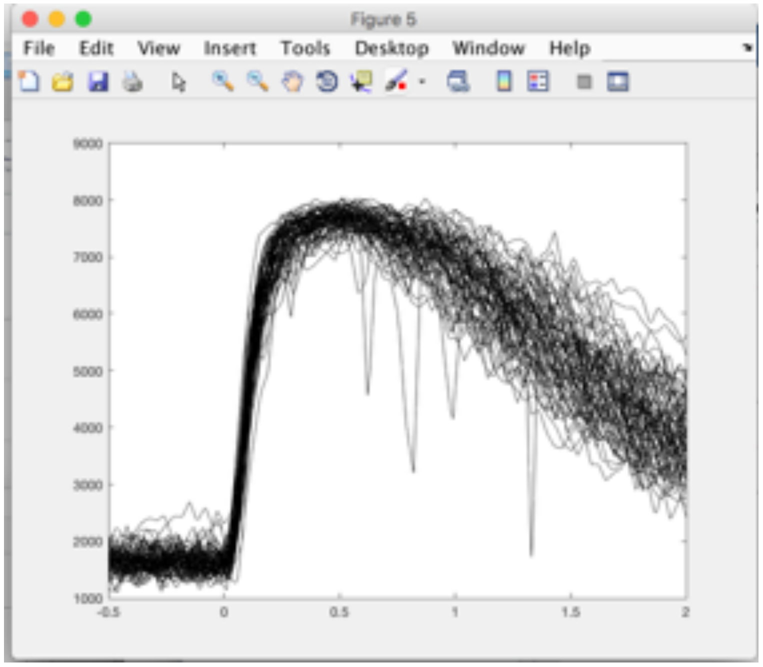
Write to Workspace

threeCell_max

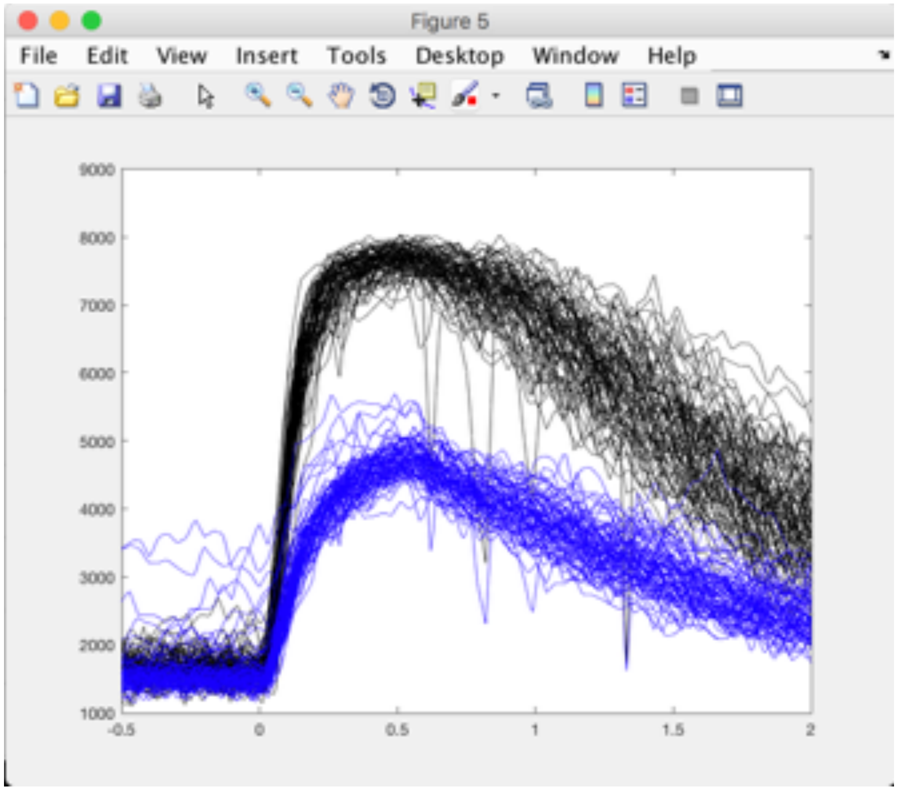
Write Matrix

```
stimTimes             1x253 double
threeCell_max_d       251x83x3 double
threeCell_max_t       1x251 double
```

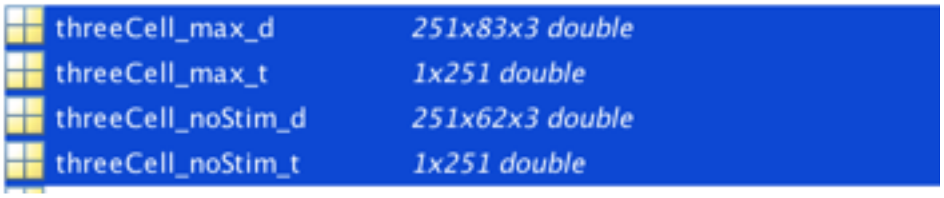
This will then assign two new matrices in your workspace called threeCell_max_d and threeCell_max_t. The _t is the 'triggered time' meaning -0.5 to 2 seconds, with the stim at 0. And the data are _d and are a matrix composed of (frames x trials x rois). So if I want to look at all trials of cell one I would type in the command window: figure,plot(threeCell_max_t,threeCell_max_d(:,:,1),'k-') :



Now let's say I want to compare that to cell number 2. I will then overlay cell 2's data by typing:
hold all,plot(threeCell_max_t,threeCell_max_d(:,:,2),'b-'):



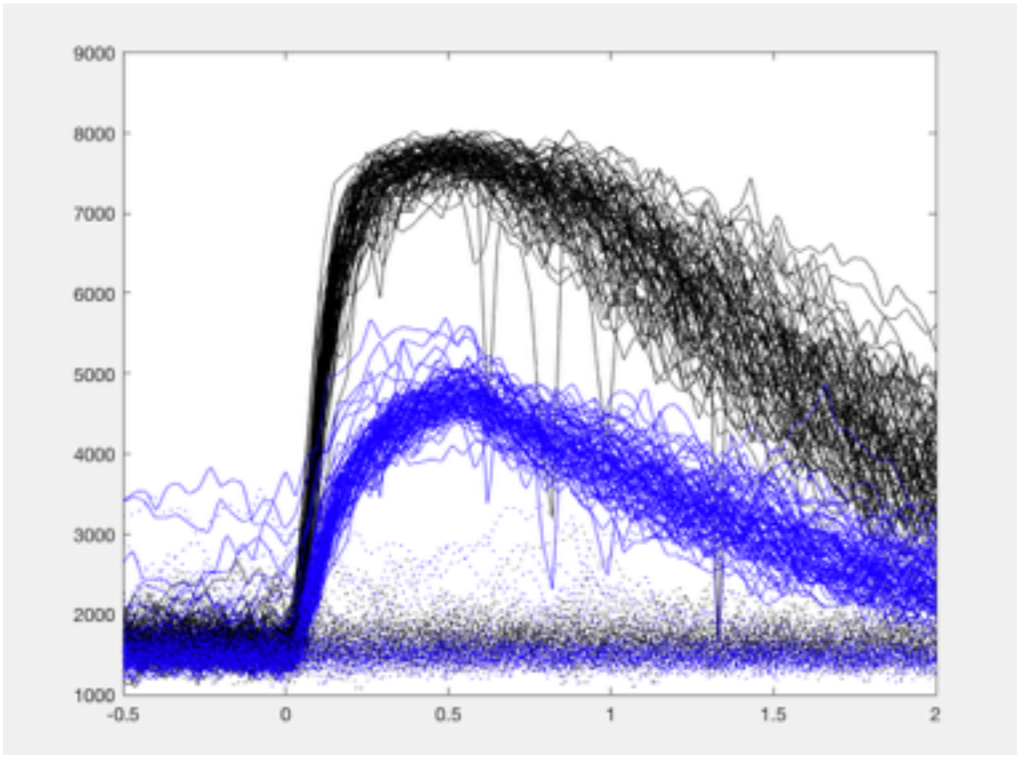
Now we can take the means and compare to no stims as well. First no stims. I make the noStim PSTH data the same way we did for max:



```
threeCell_max_d       251x83x3 double
threeCell_max_t       1x251 double
threeCell_noStim_d    251x62x3 double
threeCell_noStim_t    1x251 double
```

And I can then add to the ongoing PSTH using the following two lines:
hold all,plot(threeCell_max_t,threeCell_noStim_d(:,:,1),'k:')
hold all,plot(threeCell_max_t,threeCell_noStim_d(:,:,2),'b:')

Next is the plot (dashed lines for no stims; same colors):

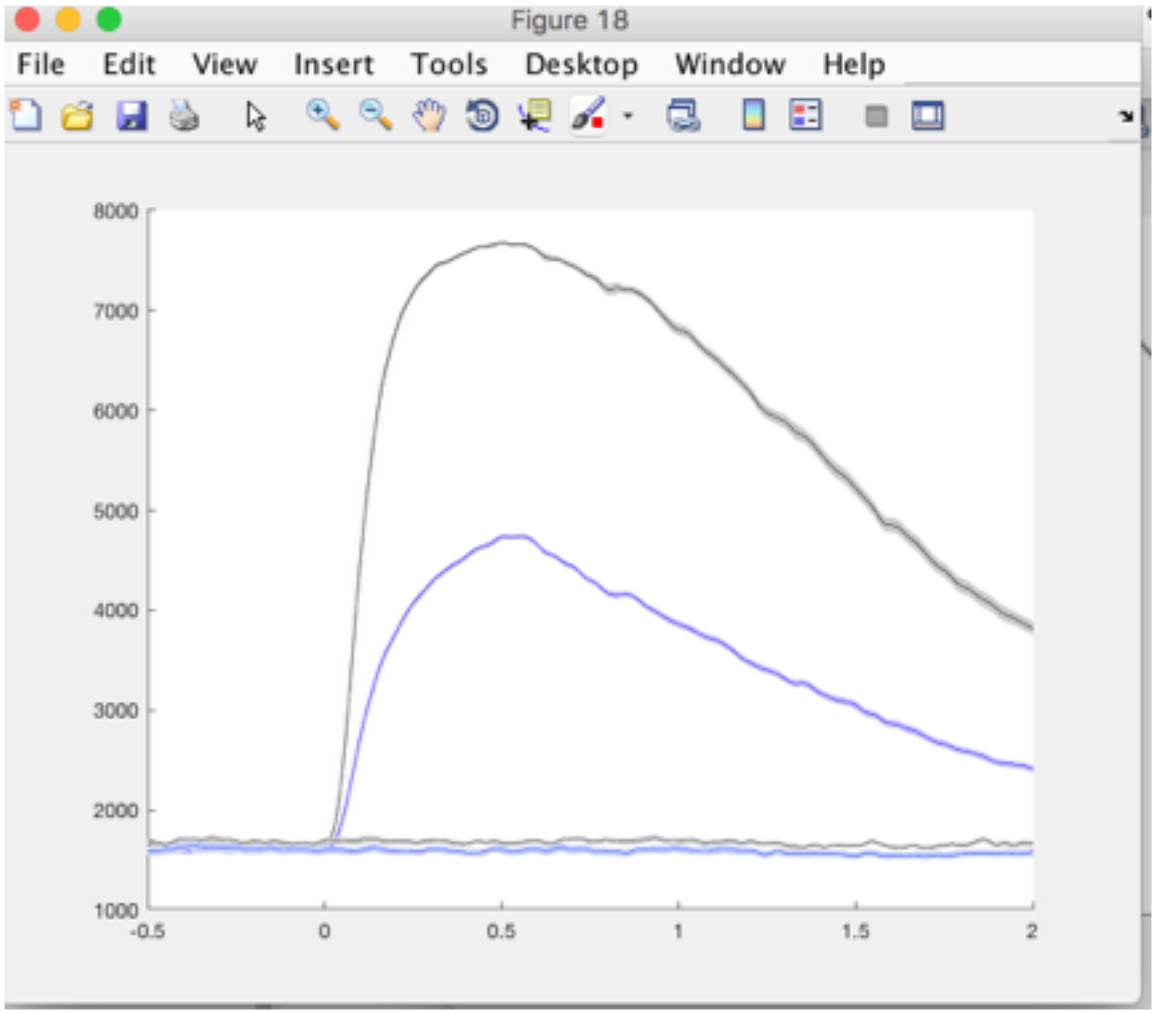


Now I will plot the means, with s.e.m using the following lines:

```
figure,boundedline(threeCell_max_t,mean(threeCell_max_d(:,:,1),2), ...
std(threeCell_max_d(:,:,1),1,2)./sqrt(size(threeCell_max_d,2)-1),'cmap',[0 0 0])

hold all,boundedline(threeCell_max_t,mean(threeCell_max_d(:,:,2),2), ...
std(threeCell_max_d(:,:,2), 1,2)./sqrt(size(threeCell_max_d,2)-1),'cmap',[0 0 1])

hold all,boundedline(threeCell_noStim_t,mean(threeCell_noStim_d(:,:,1), ...
2),std(threeCell_noStim_d(:,:,1),1,2)./sqrt(size(threeCell_noStim_d,2)-1),’cmap’,[0.2 0.2 0.2])

hold all,boundedline(threeCell_noStim_t,mean(threeCell_noStim_d(:,:,2), ...
2),std(threeCell_noStim_d(:,:,2),1,2)./sqrt(size(threeCell_noStim_d,2)-1),’cmap’,[0 0.2 1])
```

This uses ‘boundedline’ which is a function I have in the package, but is not my work that is a great file on filexchange by Kelly Kearney: http://www.mathworks.com/matlabcentral/fileexchange/27485-boundedline-m

Now if you do that you should see something like this:



Clearly these neurons were stimulus driven. From here you can do many many other things.