threshdetgui - a quick tutorial - Oct 2018

`threshdetgui` is a graphical user interface (GUI) for detection of all kinds of events in neuronal data via threshold methods. The events may be discrete/fixed-length (e.g. action potentials) or of a burst-type with variable duration (e.g. UP states as visible in the field potential). The crucial output of this program consists of time stamp lists (tsl, for discrete events) or 'extended time stamp lists' (etsl, for burst-type events). The code was developed over roughly a decade by Harald Hentschke (HH). Several batch analysis routines and other GUIs developed by HH take these data as input.

As **`threshdetgui`** makes heavy use of functions which serve a role in several other projects the code is not distributed as a package but resides distributed in diverse Matlab directories (which you need to have). This is a little awkward for you as the user, but there is no good alternative except using **`threshdetgui`** as a standalone executable (available upon request). Code required for running **`threshdetgui`** is in the following of HH's matlab subdirectories/repositories (subject to change):

`/projects_ephys/threshDet`
`/projects_ephys/threshDet/private`
`/projects_ephys/threshDet/parameterFiles`
`/fileIO`
`/fileIO/abfload`
`/graphics`
`/graphics/GUIfunc`
`/sampledSeries/analysis`
`/sampledSeries/etslfunc`
`/sampledSeries/preprocessing`
`/utilities`
`/utilities/datetime`
`/utilities/geometry`
`/utilities/timeseries`

This tutorial is a quick tour through the major functionality of **`threshdetgui`**. Topics not yet covered are spike sorting and burst detection (these will be added later upon request).

To get started, take the following steps:
- get all required code files and set the Matlab paths accordingly (it is assumed you know how to do that)
- in the Matlab command window, issue the command **`threshdetgui`**
- if two multi-panel windows pop up, you have at least the basic code and paths set correctly
- if Matlab tells you at this stage or in later steps that it doesn't know a function or variable of this or that name, get the missing code files and/or set the paths accordingly.

1. Options window

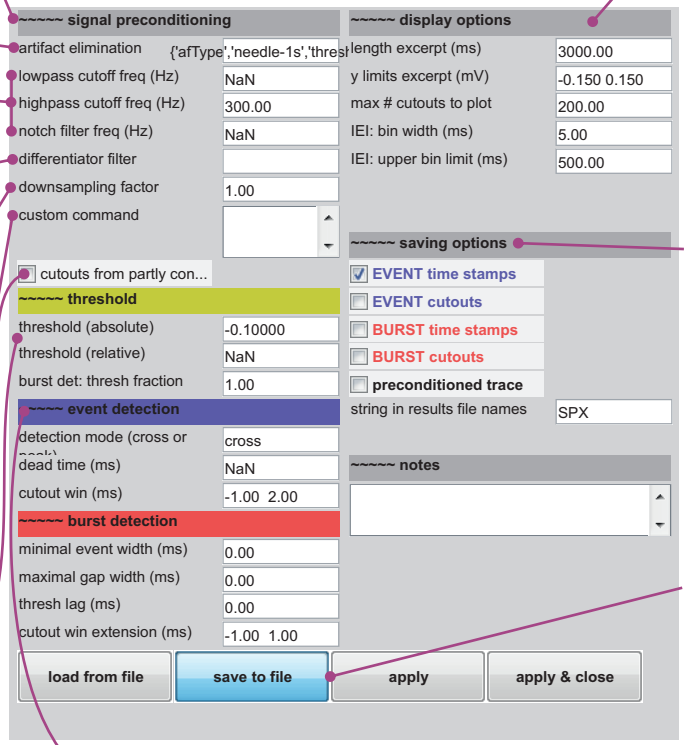
- pops up automatically after starting threshdetgui
- the settings govern detection of 'events' and 'bursts', their display and the kinds of results to be saved
- conceptually, '**events**' are discrete signals and/or signals of fixed length, e.g. spikes, as opposed to '**bursts**', e.g. UP states, which have variable duration and for which duration is an interesting parameter. Event detection and burst detection are based on different algorithms, hence the settings are placed under different headings
- the concrete values in the example below are useful for detection of spikes. The explanations below don't deal with burst detection (which will be dealt with later because it is more complicated)

preconditioning

- operations will run in this order: notch->artifact removal->lowpass->resample->hipass->diff-filter ->custom cmd
- input arguments into function elim_artefact
- general rule: enter NaN if no filtering requested
- differentiation filter: a means of detecting sharp transitions in the data (we use it for detection of IPSCs; leave field empty if no filtering requested)
- a downsampling factor of 1 means no down-sampling, a factor of 2 means data points 1, 3, 5,.. are retained, etc.
- custom command is the magic wand: it allows you to perform additional operations on the data; e.g.

```
d(:,2,:)=abs(d(:,2,:));
```

  No error checks whatsoever against faulty code!
- cutouts of detected events are usually generated from the conditioned data trace. Ticking this box will result in the cutouts being generated from partially preprocessed data (up to resampling)



threshold

- absolute: enter a specific value of the threshold (or set via mouse click in main window)
- relative: specify relative to the base line noise (the exact definition is a bit technical); -3.5 is a reasonable value. **ATTENTION: will override the absolute threshold value; set to NaN unless you know the implications**
- burst det thresh fraction is only relevant for burst detection, ignore

event detection-specific settings

- detection mode: 'cross' counts threshold crossings as events (standard for spikes); 'peak' counts each peak beyond threshold as an event (useful for PSCs)
- dead time: the time interval after an event in which the trigger shall be 'dead' (=ignore events)
- cutout win: length of cutouts for display and output into separate *.mat file (if requested)

data display options

- have no effect whatsoever on the analysis
- if you'd like automatic y axis scaling in the excerpt window (see next pages) enter `NaN NaN` in the 'y limits excerpt' field

PLEASE NOTE: these settings (as well as the threshold value) will register in the main window only after you perform some operation in that window, e.g. shifting the excerpt (clicking in the overview window), detecting events, etc.. In other words, the settings here will only take visible effect after some user action.

saving data

- time stamps and cutouts will be saved in different files
- you can also have the preconditioned data trace saved as *.mat file
- string in results file names: useful or even essential for automated batch processing of the data

buttons

- after entering values in the numerical fields above, hit 'apply' or 'apply & close' for these settings to take effect
- load & save should be clear
- there are some example parameter files in \threshDet\parameterFiles

2. Main window after loading data channel

'overview window'
- display of whole data trace (one channel)
- automatic scaling of y axis
- mouse click in this window = excerpt in window below will be centered on the mouse's x coordinate

'excerpt window'
- display of excerpt of arbitrary length (to be set in options)
- scaling of y axis automatic or manual
- mouse click in this window shifts threshold (yellow line) and **deletes all results of previous detections, including manual deletion of artifacts & spike sorting**

a propos mouse click...
the current implementation requires a mouse click somewhere in the 'open space' (at least 3 pixels away from any data trace) for anything to happen

buttons

read single data channel

opens a window in which all detection and display parameters are set (see preceding page)

preconditions data: filters, applies custom commands, etc.

detects events
(again: events are signals of fixed length, e.g. spikes, as opposed to e.g. UP states, which may have variable length)

computes event amplitudes (useful for PSCs)

does principal components analysis (PCA) on events, allowing spike sorting (separation of cutouts into populations of similar shape, or 'units')

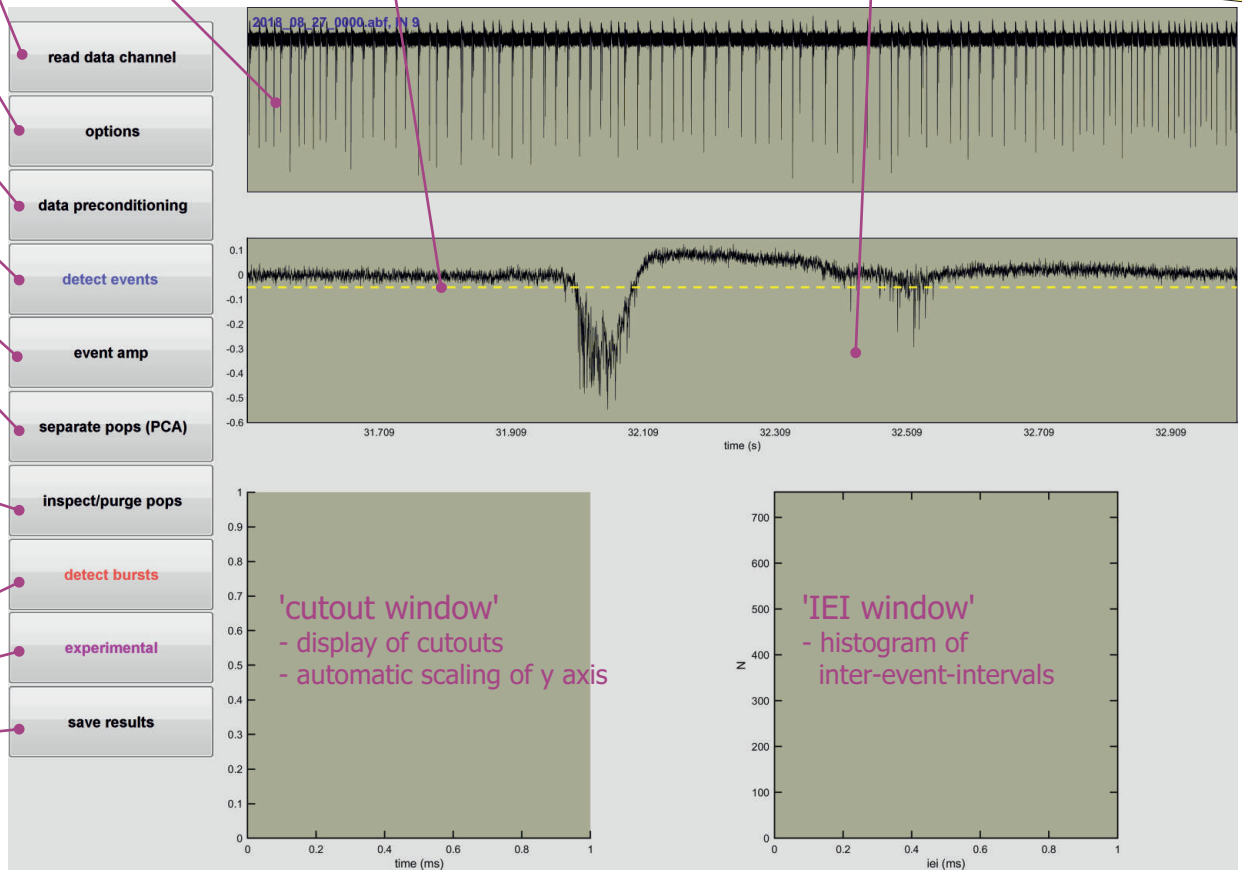
dual purpose:
i. displas all event cutouts with the option to delete individual ones
ii. if cutouts were previously separated into populations via PCA allows for deletion of individual populations

detects bursts, e.g. UP states in LFP

experimental code, ignore

saves results (it's not done automatically!)

threshold
- if above zero, positive-going transitions will be detected
- if below zero, negative-going transitions will be detected
- if fine-tuning of the threshold level is required, use the numerical field in the options window.



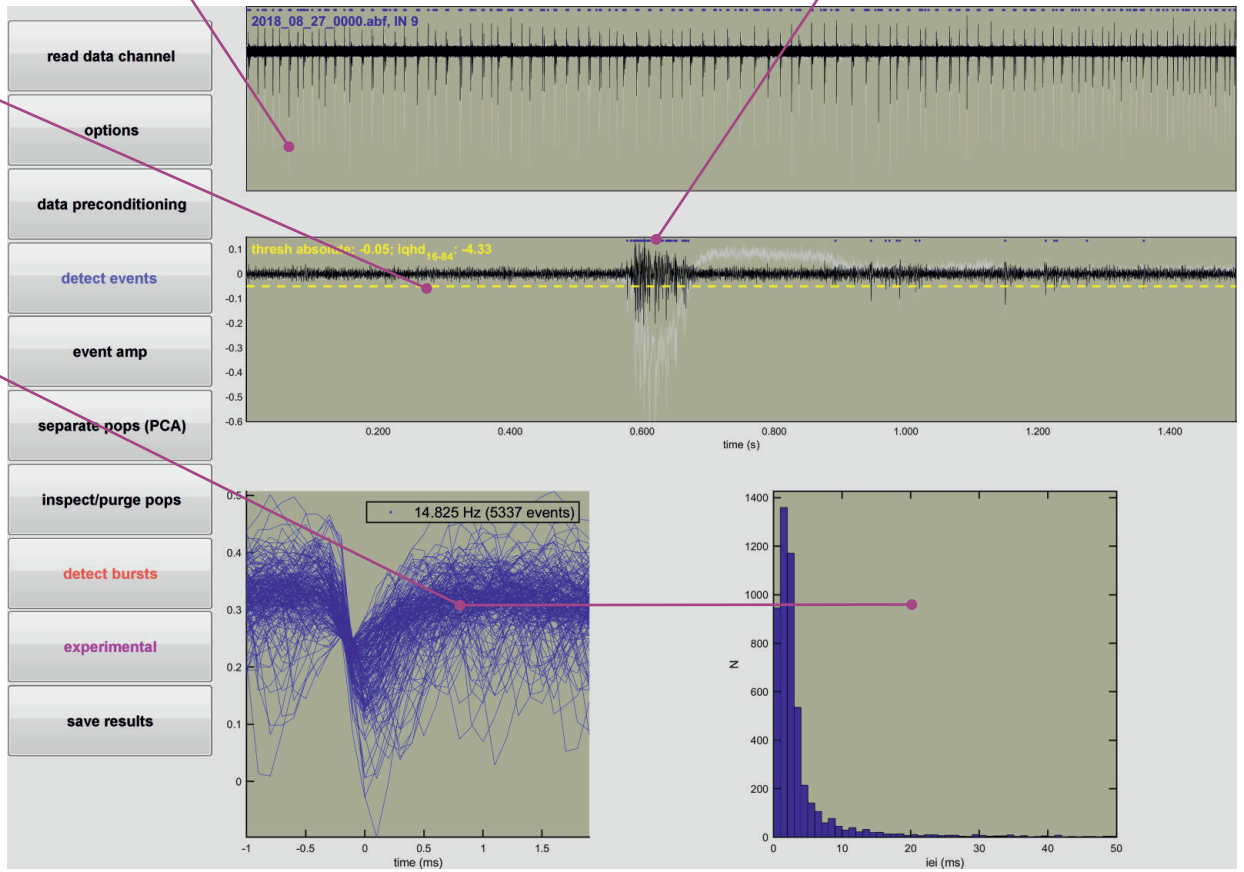
3. Main window after preconditioning & event detection

preconditioning
- after preconditioning the original data trace is displayed in white in the background, the preconditioned data in black (in both overview and excerpt window)

threshold
- in this specific example, the threshold had been set to a negative value, so negative-going spikes were asked for
- 'iqhd' = inter-quantile half-distance, a parametric equivalent of standard deviation (sd); informs us how far the threshold is from the baseline in quasi-sd units and can thus help us to set thresholds more consistently across experiments

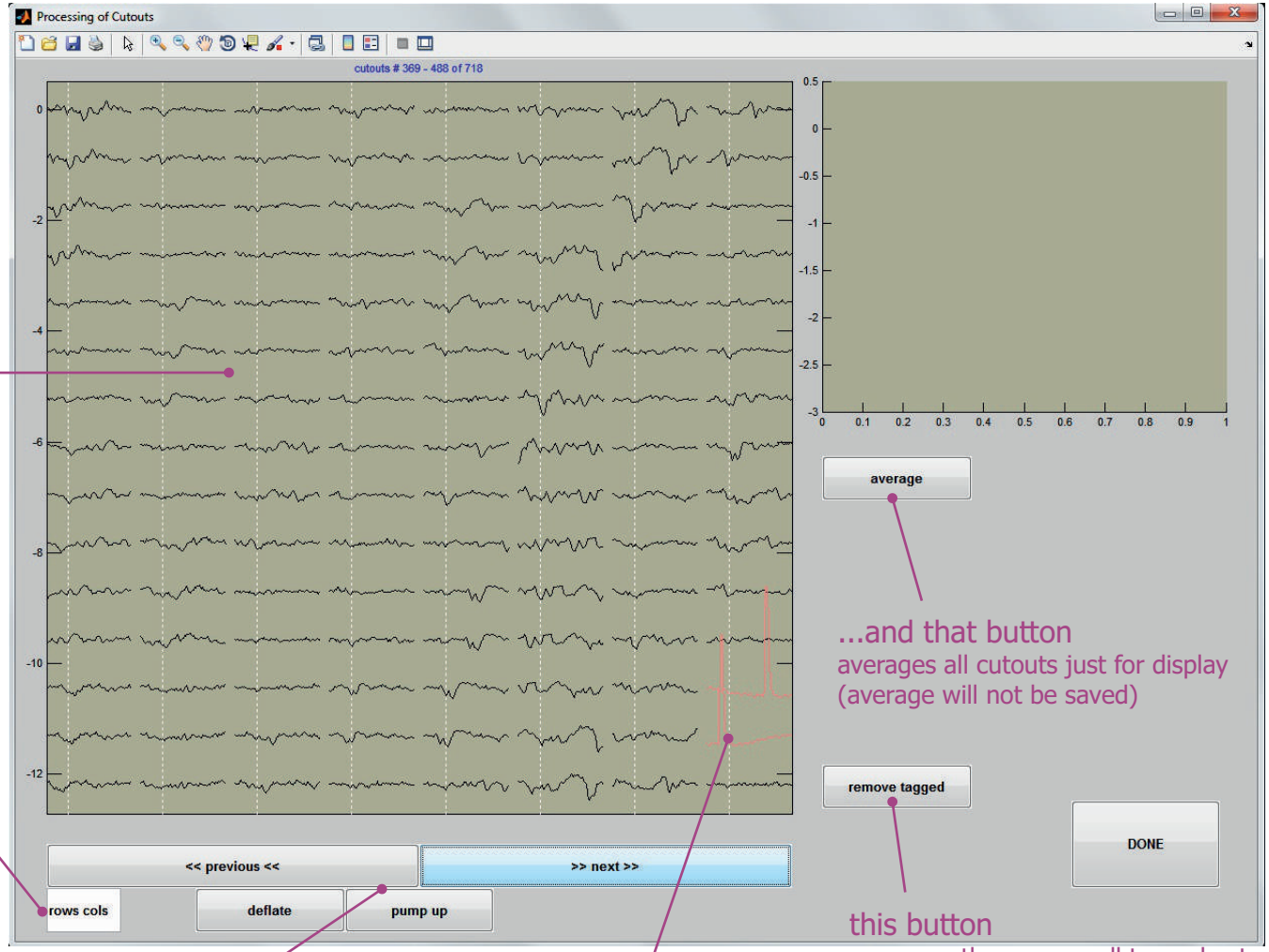
detected events (1)
- after event detection the events (spikes) are displayed as dots in the same color as the cutouts (in both overview and excerpt window)

detected events (2)
- a subset of the cutouts, the number of events, firing rate and the inter-event-interval histogram (IEI) are displayed in the two windows at the bottom
- the large counts in the first two bins in the IEI leaves little doubt that this is a multi-unit recording



4. Inspect/purge pops window

- this window opens up if you hit the 'inspect/purge pops' button and the detected events were NOT previously sorted into populations



cutouts
scroll-thru display
of all cutouts

layout
if you prefer a number of rows and columns different from the default, enter the numbers here (all tagged cutouts will be un-tagged)

...and that button
averages all cutouts just for display
(average will not be saved)

this button
permanently removes all tagged cutouts and the corresponding time stamps (this action can only be undone by re-detecting events in the main window)

buttons
for scaling and scrolling through cutouts

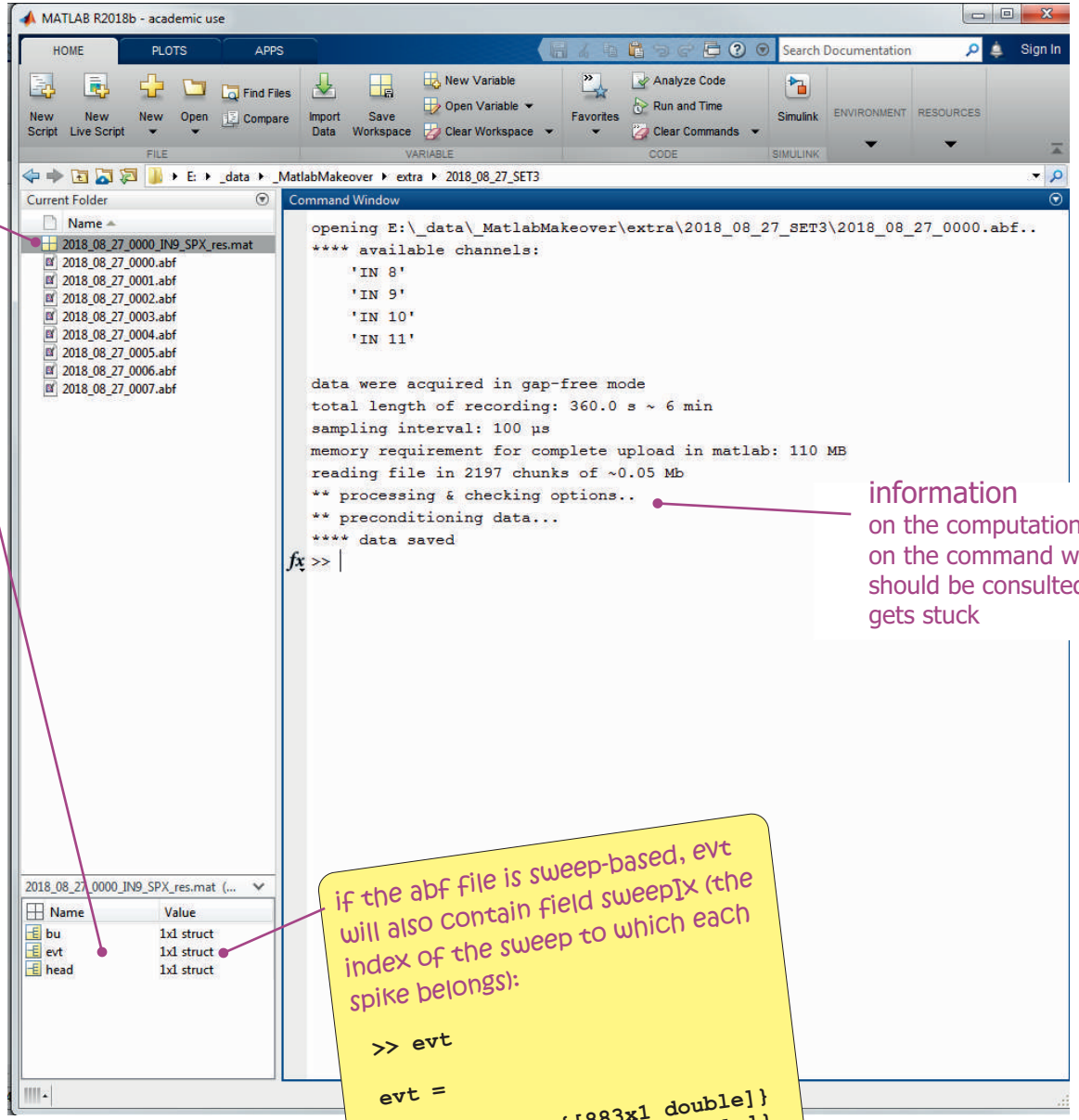
manual removal of artifacts
click on the cutouts (yes, ON the data traces) you wish to remove; these will be tagged in a nasty pink (they will not get lost when you scroll through the data)

5. See what we've achieved so far

results

- pressing the 'save results' button will make Matlab save the specified data in the same directory as the abf file

- we requested only the events' time stamps; these are saved in the high-lighted file (note the 'SPX' string and the fact that the blank in the chan-nel name ('IN 0') is gone)
- **`evt`** is a struct which contains the time stamp list in field tsl, which is a cell array (populations resulting from spike sorting would be placed in the different elements)
- **`bu`** is a struct containing results of burst detection (empty in the current example because no bursts were detected)
- **`head`** is a struct containing lots of information on the recording/data channel, most of which is taken from **`abfload.m`**



information

on the computations is put out on the command window, which should be consulted if the code gets stuck

if the abf file is sweep-based, evt will also contain field sweepIx (the index of the sweep to which each spike belongs):

```
>> evt

evt =

        tsl: {[883x1 double]}
    sweepIx: {[883x1 double]}
```

6. Batch detection of events

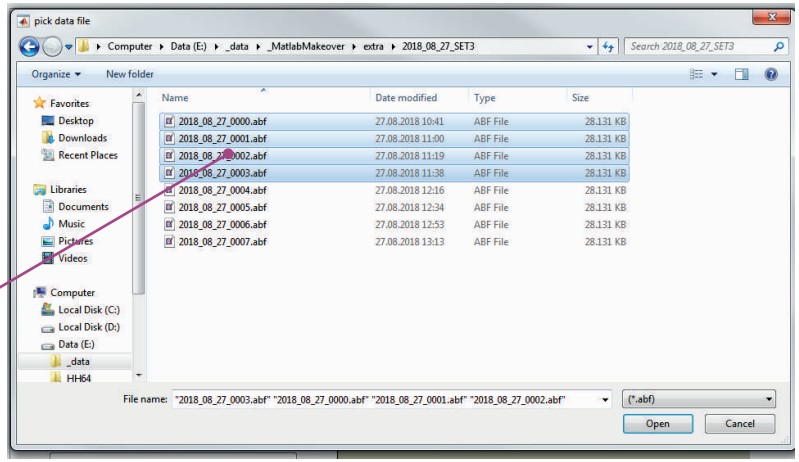
Yes, it's possible! The idea is to save time and effort with multiple recordings per experiment, as all data processing and event detection parameters should be kept constant within experiments.

1. start as usual

- read data channel
- set detection parameters and saving options
- preprocess data
- detect events
- save results

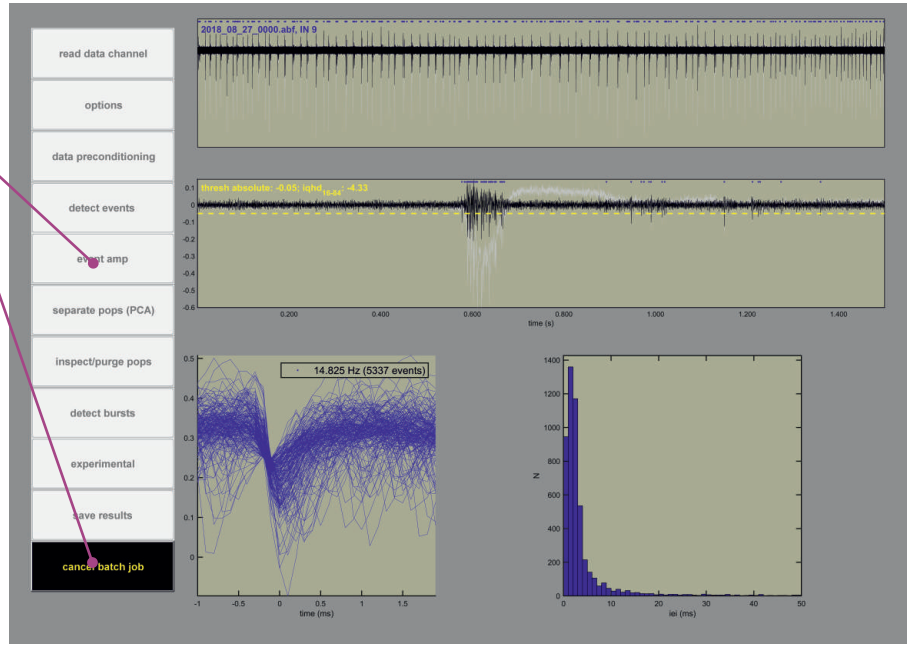
2. press button 'read data channel' again, now selecting all files you wish to process in the exact same way

- threshdetgui will process only the channel which was selected in the previous analysis



3. threshdetgui will immediately jump into batch processing mode

- note that the regular buttons are inactivated and a new button appears which allows you cancel batch processing. Also, the background of the main figure window is a little darker to indicate batch mode.
- threshdetgui will perform data preconditioning as specified in the options, including the same filters, threshold, etc., and also save the data automatically
- once it is done, results from the last file processed can be seen and the button arrangement is restored



4. Notes

- be sceptical of the results! Batch processing is convenient, but in experiments which alter neuronal activity substantially adjustment of parameters from recording to recording may be warranted
- the comment above applies even more to burst detection; here, adjustment of parameters between recordings is the rule, not the exception, which is why batch burst detection is not implemented