Documentation

fpExplorer

Contents

1. Description

This application should be used to preview and perform basic analysis of fiber photometry data. The application was designed for data collected by TDT system.

2. Quick guide

2.1. Start

fpExplorer.exe file should be used to start the application. Then, the user will be presented with a “Select Data” window. This window can be also opened later at any time using “Select Data” button in the main menu at the top of application main window. (Fig.1)

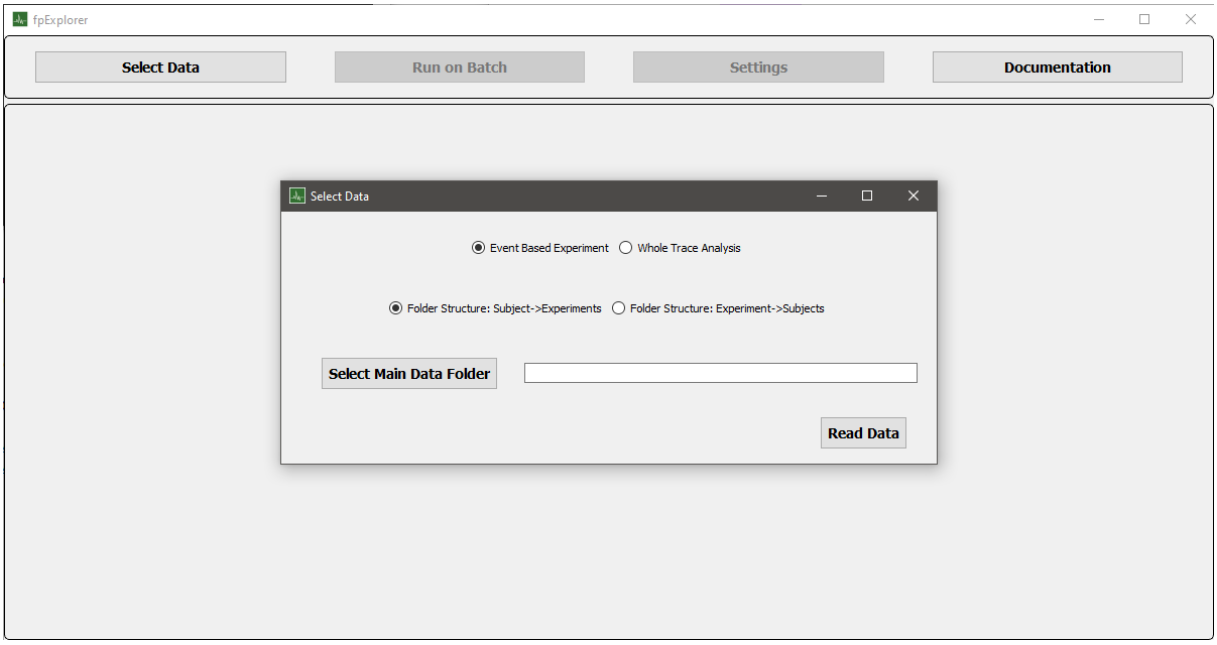

Fig 1. Application main window with “Select Data” window where user defines what kind of data will be analyzed.

2.1.1. Event Based Experiment

If the dataset contains recorded events that don’t start with Cam or Tick(i.e., PrtA 253, PrtA 249), the user can analyze it as event based experiment. In this case, the user will be able to preview raw data, trim the data, downsample it, normalize it and detect spikes. Moreover, the user will be able to plot events and to perform peri-event analysis.

2.1.2. Whole Trace Analysis

If the dataset does not contain recorded events that don’t start with Cam or Tick(i.e., PrtA 253, PrtA 249), the user can analyze it as the whole trace. In this case user will be able to preview raw data, trim the data, downsample, normalize and detect spikes.

2.1.3. Folder Structure

The application is designed to analyze two kinds of data structures from TDT system. (https://www.tdt.com/docs/synapse/managing-data-for-your-lab/)

Subject -> Experiments

A folder that contains subfolders with subject names. Within subject subfolders, there should be subfolders with experiment names. Within experiment subfolders, there should be user's TDT files (i.e., *.Tbk, *.Tdx, *.tev, *.tin, *.tnt, *.tsq).(Fig2.)

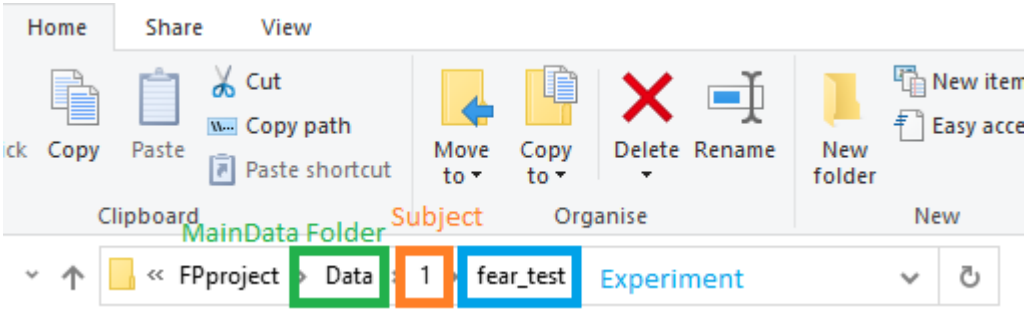

Fig2. Subject->Experiment folder structure example, where fear_test is an experiment name with TDT files (i.e., *.Tbk, *.Tdx, *.tev, *.tin, *.tnt, *.tsq)

Experiment -> Subjects

A folder that contains subfolders with experiment names. Within experiment subfolders, there should be subfolders with subject names. Within subject subfolders, there should be user's TDT files (i.e., *.Tbk, *.Tdx, *.tev, *.tin, *.tnt, *.tsq).(Fig3.)

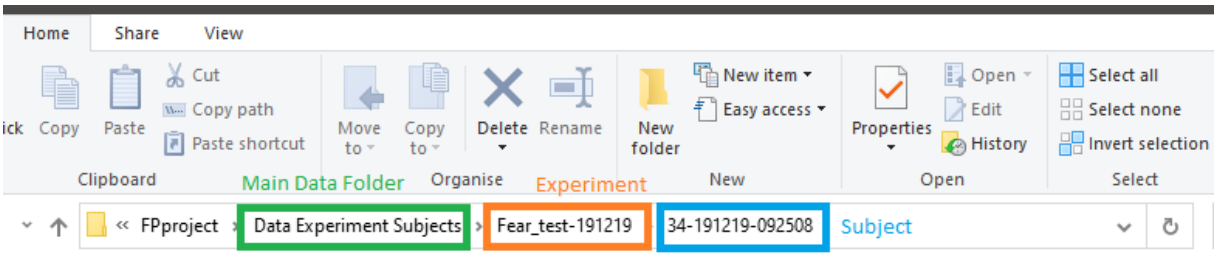

Fig3. Experiment->Subject folder structure example, where 34-191219-092508 is a subject name with TDT files (i.e., *.Tbk, *.Tdx, *.tev, *.tin, *.tnt, *.tsq)

2.1.4. Select Experiment Name

The user is prompted with the experiment names available from the first data set in the main folder. (Fig4.)

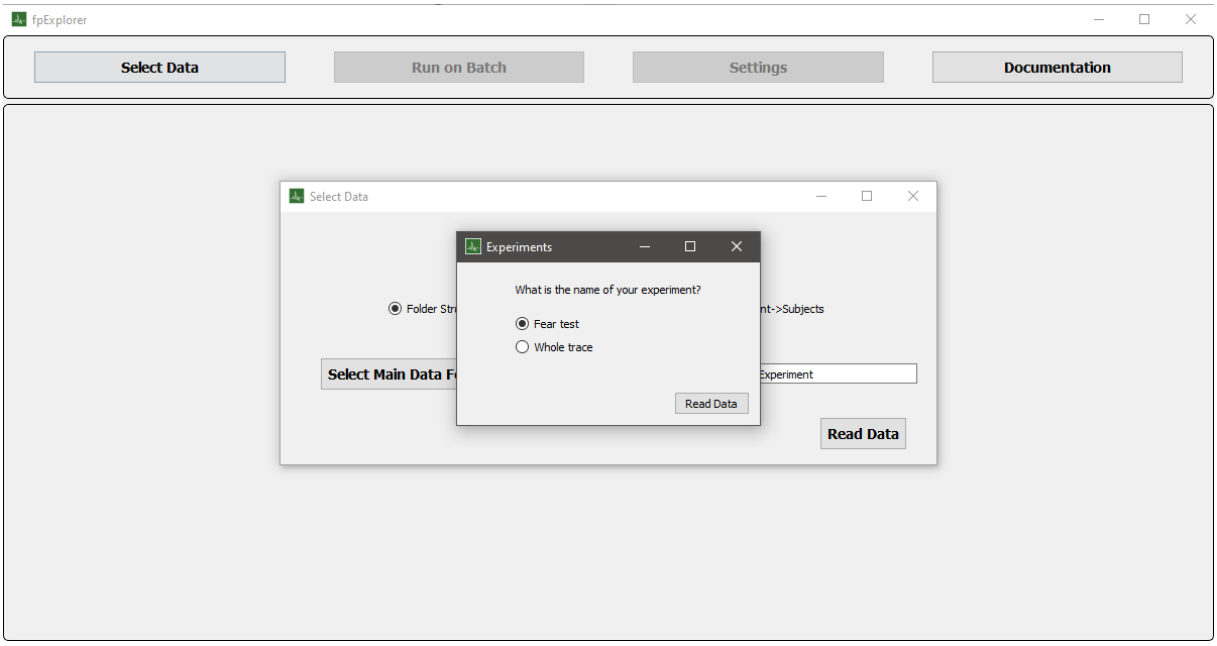

Fig4. Select experiment that will be analyzed

2.1.5. Select Signal and Control Channels

The user is prompted with channel names available from the first data set in the main folder. The user should select signal channel and control channel (Fig5.)

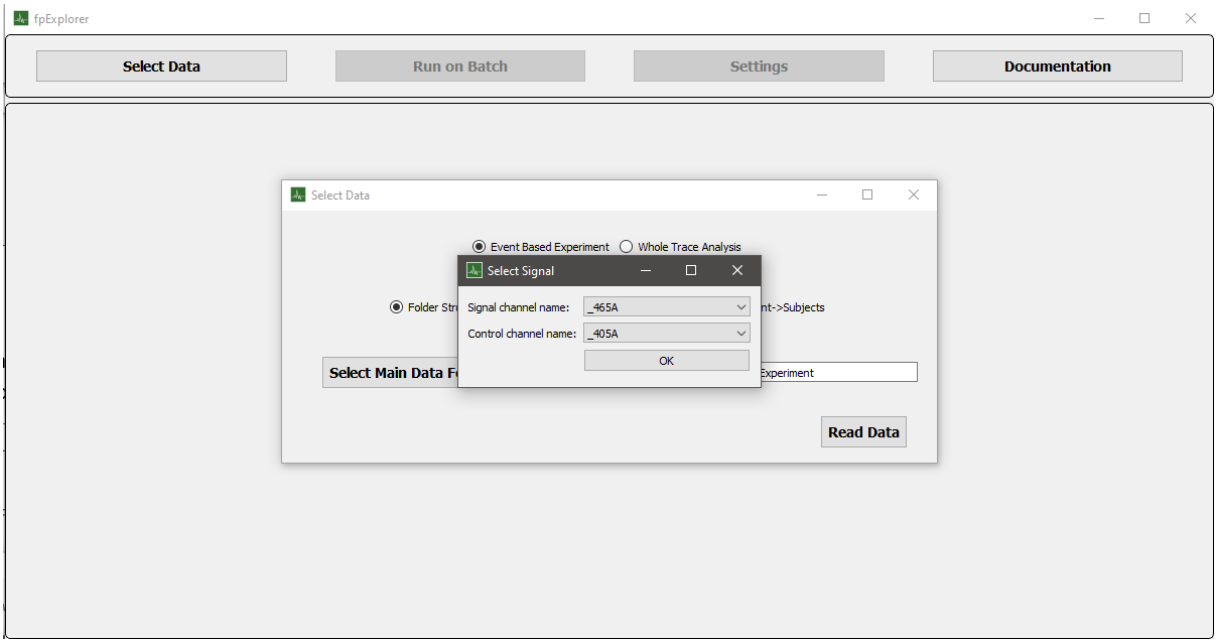

Fig5. Select signal channel name and control channel name.

2.2. Preview

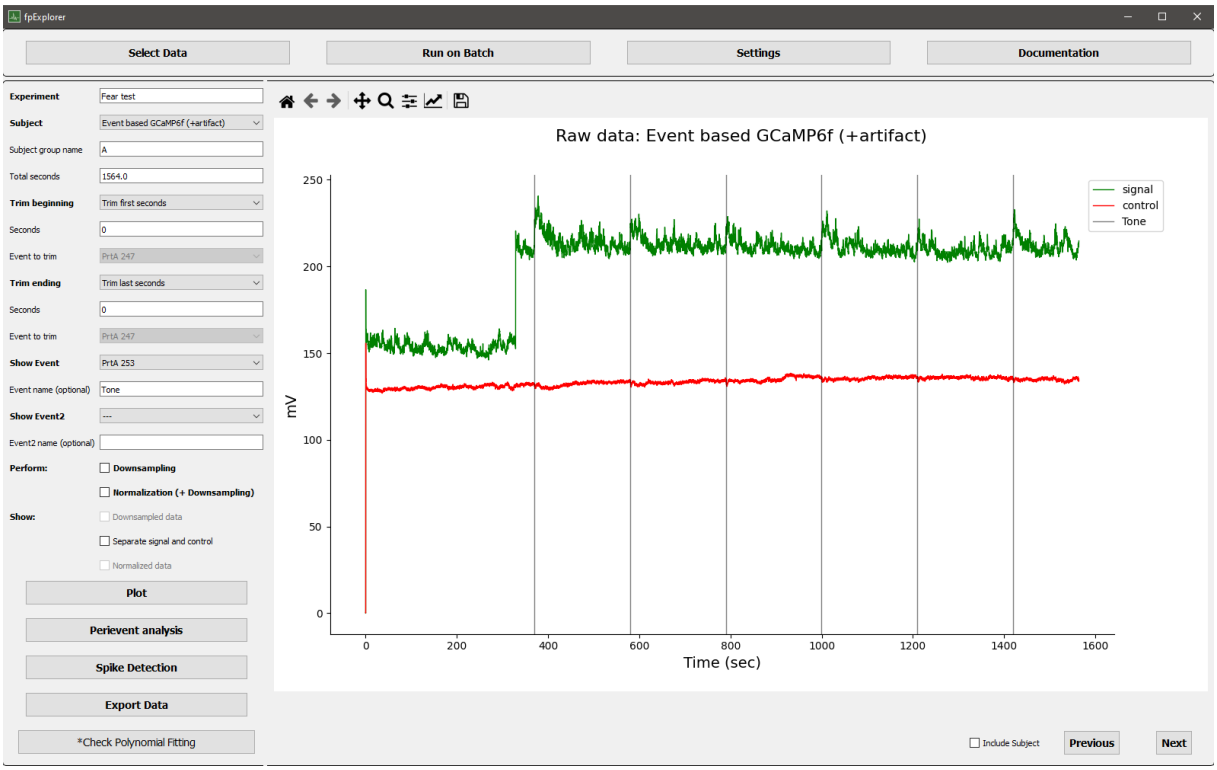

Fig6. The example preview of raw data read from TDT data files. Main options for subject based data analysis are visible to the left of the plot.

2.2.1. Options

Subject

The user can select any subject from the drop down menu or use “Next” and “Previous” buttons to move between different subjects.

Subject group name

Here, the user can assign a group name to the subject.

Trimming

The user can decide to remove certain number of seconds from the beginning or ending of the recording before the analysis. The user can also opt to trim beginning or ending based on the first or last onset of the recorded event.

Events

The user can visualize up to two events available from the drop down menu. It is also possible to enter a custom name for each selected event.

Downsampling and Normalization

The user can choose to downsample the signal (0.5%-50% of the original sampling rate) and to normalize (to fit the control to the signal in order to reduce photobleaching in post processing).

By clicking on the “Plot” button, user can visualize the data. (Fig7)

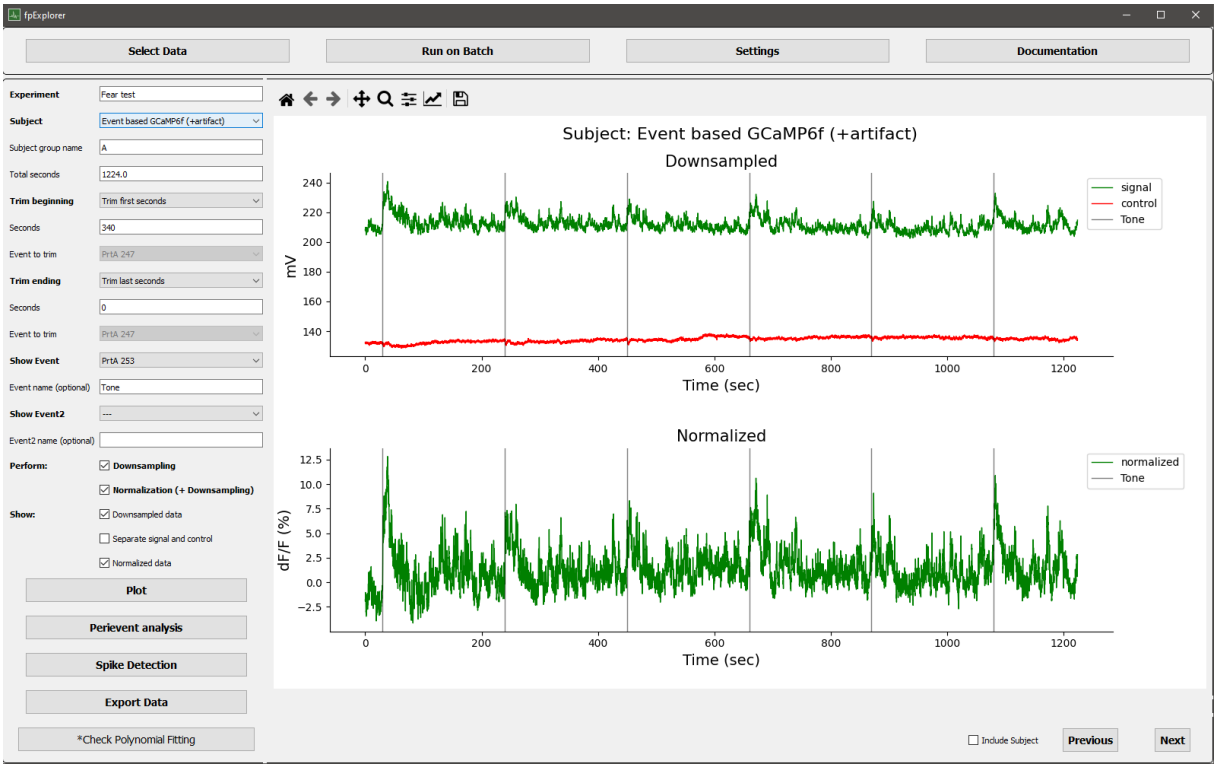

Fig7. The example of data visualization.

In order to check how well the polynomial fit worked, the user can click on ‘*Check Polynomial Fitting’ button. (Fig8)

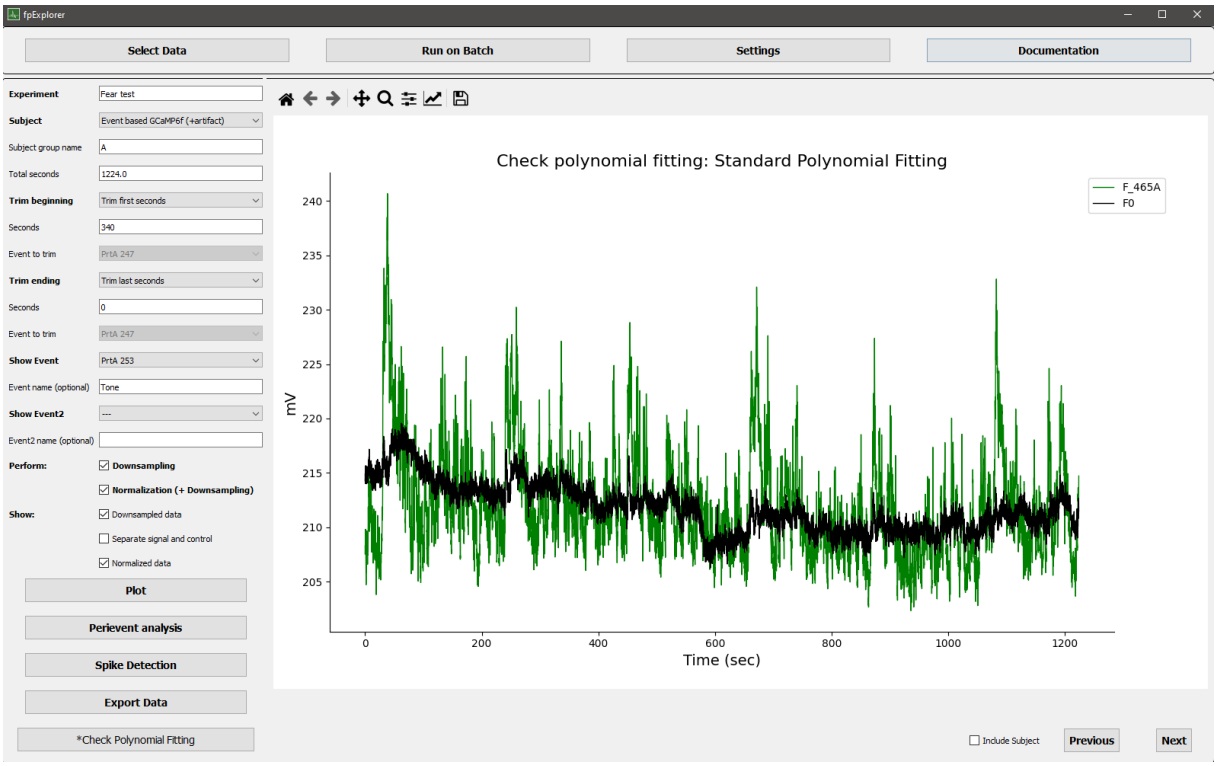

Fig8. Option to visualize how well the polynomial fitting works.

2.2.2. Settings

Downsample

The user can modify the sampling rate or normalization method at any time by clicking on "Settings" button from the main menu at the top of main application window.

The recommended downsampling should be between 1 and 2% of the sampling frequency, so as not to lose too much information.

Normalize

At the moment, the application allows to normalize using either Standard Polynomial Fitting (based on fitting applied in David Barker's pMAT application) or Modified Polynomial Fitting (Mulholland's version of polynomial fitting). These methods will be described later. (Standard Polynomial Fitting is the default). Then the user has the option to either show normalized data as df/F in % or as a Z-score.

Filter

The user can also choose to smooth data using filter with a window around each sample of the data between 0 and 100 samples. (10 is the default).

We are using a digital filter forward and backward to the signal from Python's scipy.signal package called filtfilt:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.filtfilt.html

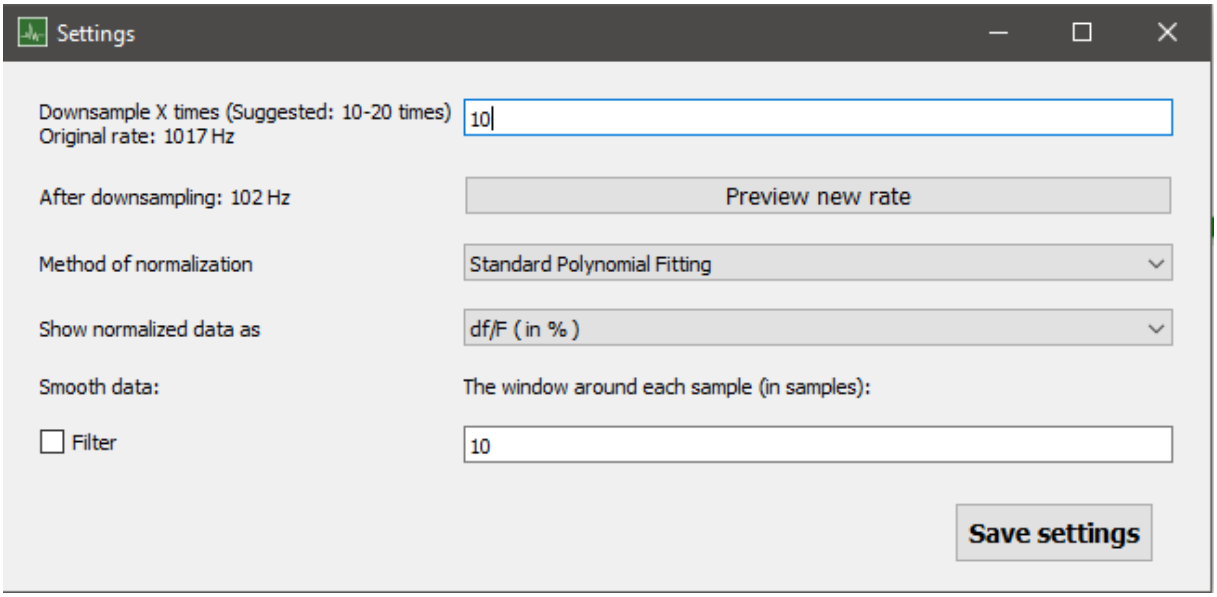

Fig9. Settings window where user can change normalization method and downsampling applied during the analysis.

2.2.3. Plots

Zoom

User can zoom in on the data by scrolling the mouse (zooms only along x axis) or by clicking on a magnifying glass and drawing a rectangle over a place of interest. Left and right arrows go back and forward between rectangle zooming.

Move

User can move the plots by selecting a cross and then by dragging and dropping plots.

Adjust Plot Layout

User can modify the spaces between subplots by clicking on the icon with three sliders. (Fig10)

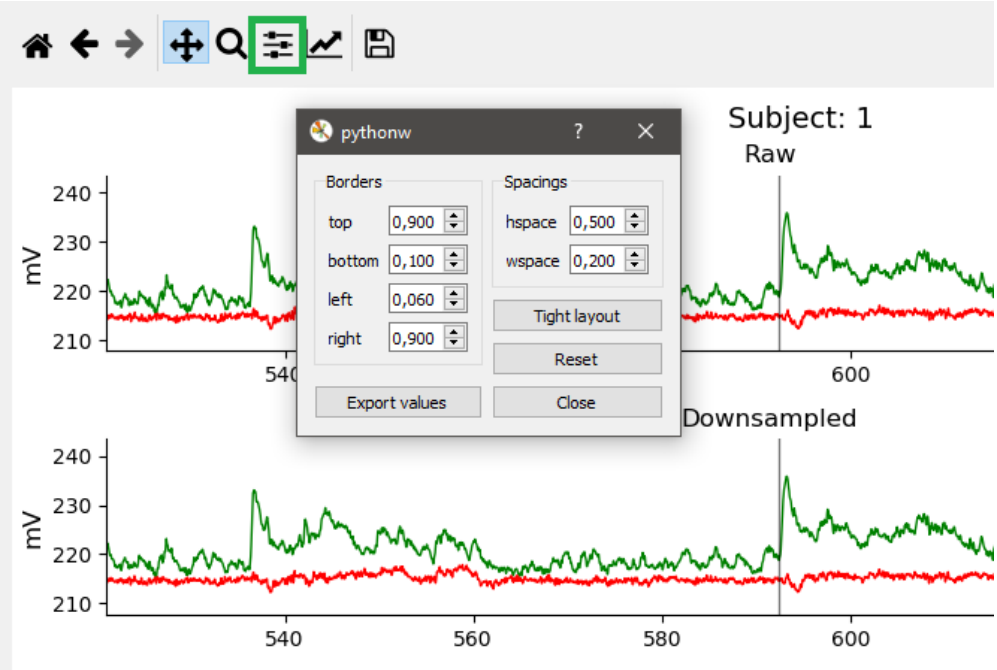

Fig10. Option to adjust subplots layout.

Adjust Labels and Range

User can also customize plot's label, axis labels and displayed data range. (Fig11)

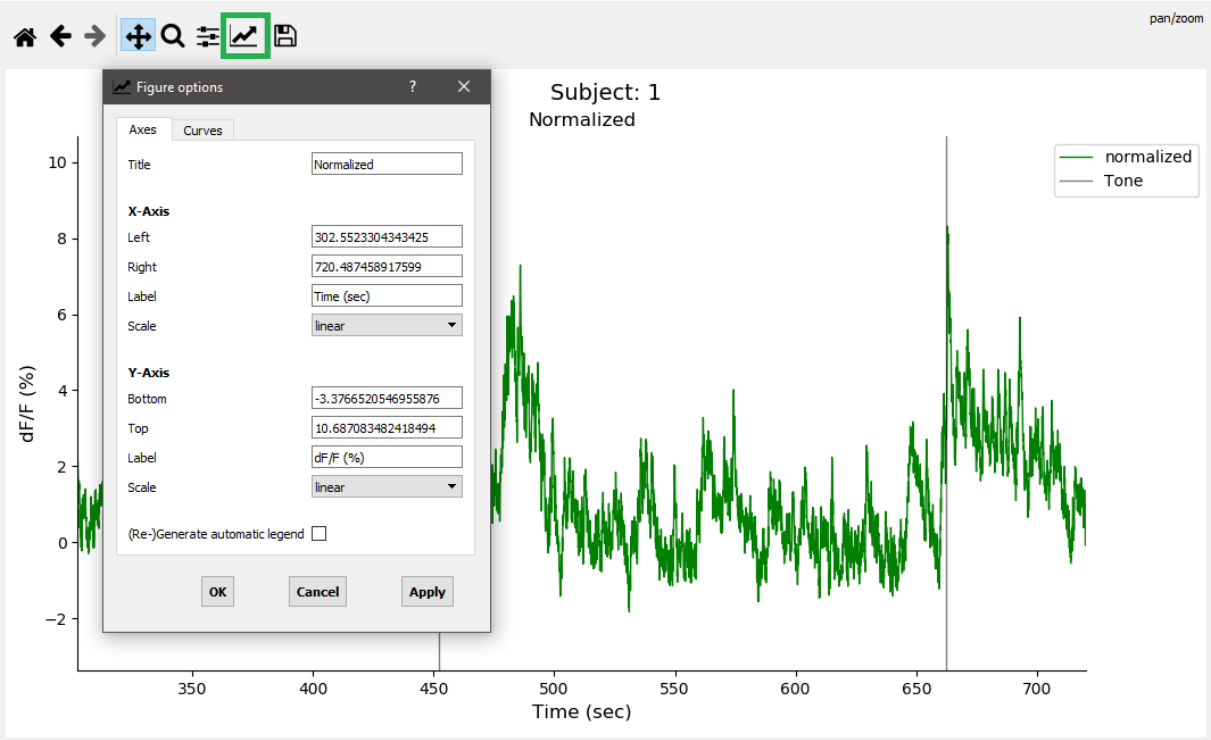

Fig11. Option to customize the plot.

Save

By clicking on a disk icon, user can save a custom plot at any time.

2.3. Analysis

2.3.1. Perievent Analysis

By clicking on a “Perievent analysis” button, the user can choose to perform some basic analysis around the selected event throughout the recording (Fig12). After the first perievent preview, the user will be presented with the option to include or exclude single trials (Fig13).

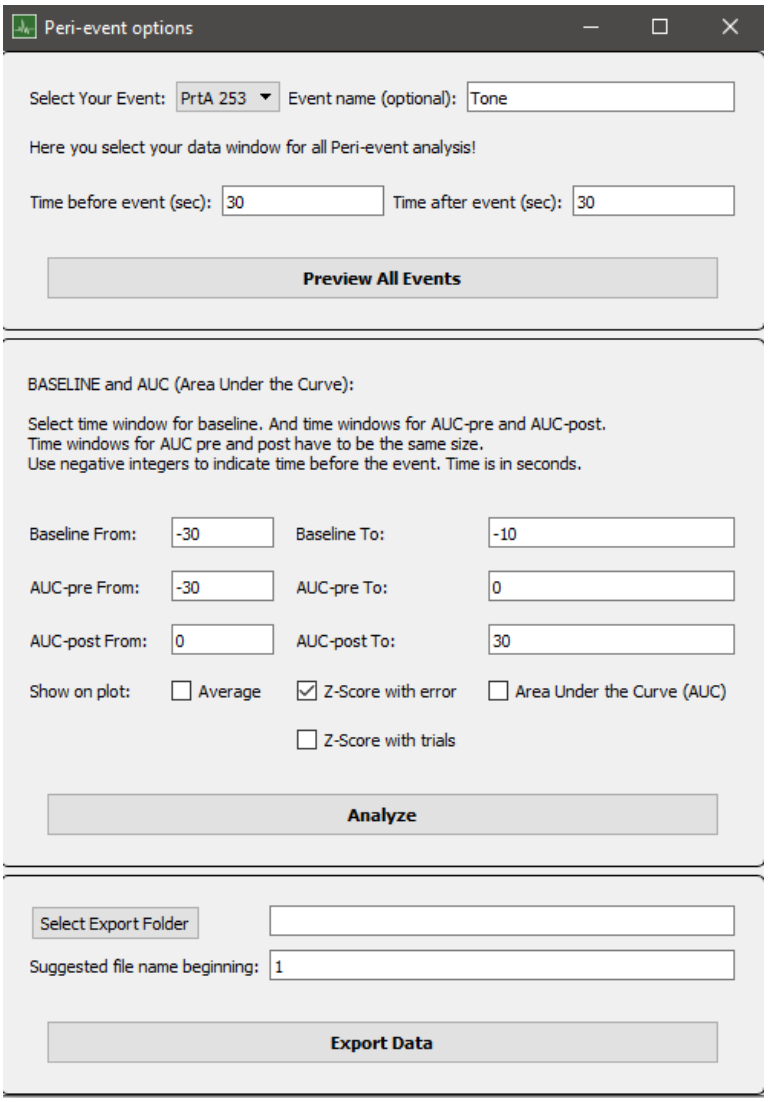

Fig12. Perievent analysis options.

Preview All Events

Here, the user can visualize normalized data around each instance of selected event throughout the recording. First, user needs to select the event from the drop down menu and enter how many seconds before and after each event instance to show. This data window will also be used for the next peri-event data analysis. Optionally, user can enter a custom event name. (Fig13)

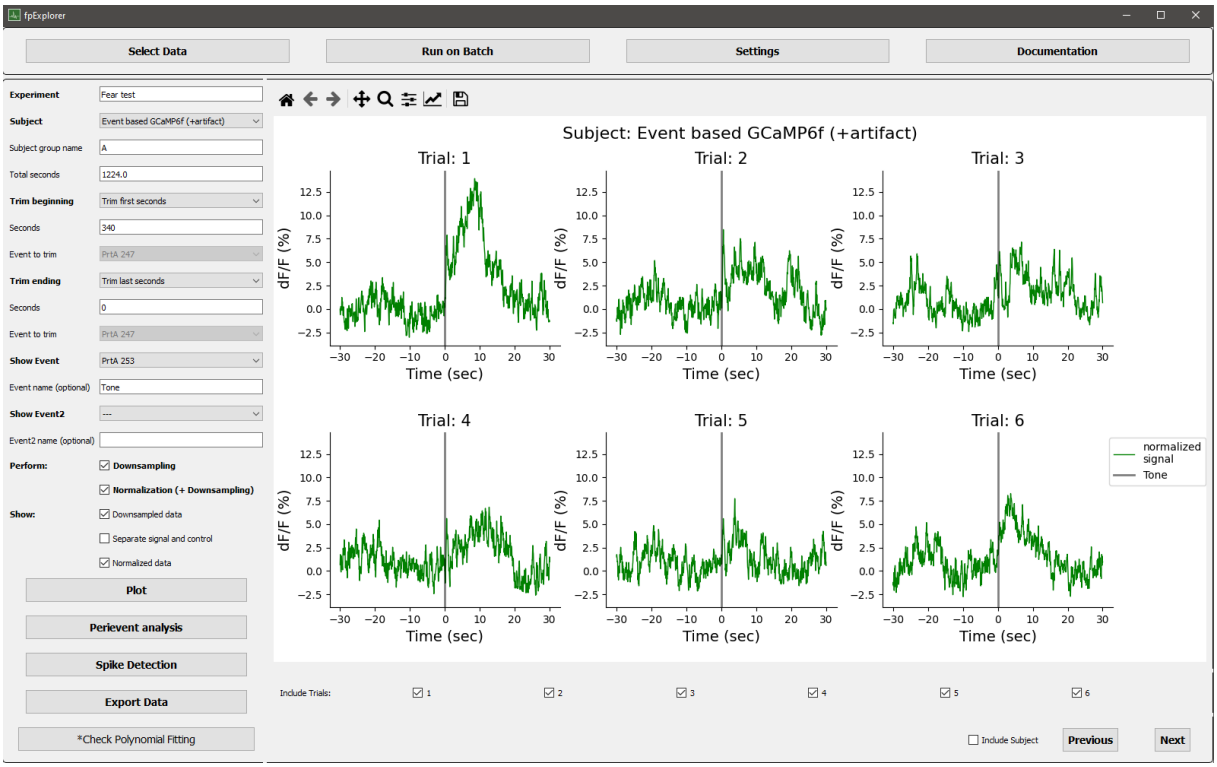

Fig13. The example of perievent preview.

Analyze

- Average

Here, the user can visualize average downsampled data around the selected event throughout the recording. First, user needs to select the event from the drop down menu and enter how many seconds before and after the event to show. Optionally, user can enter a custom event name. (Fig14)

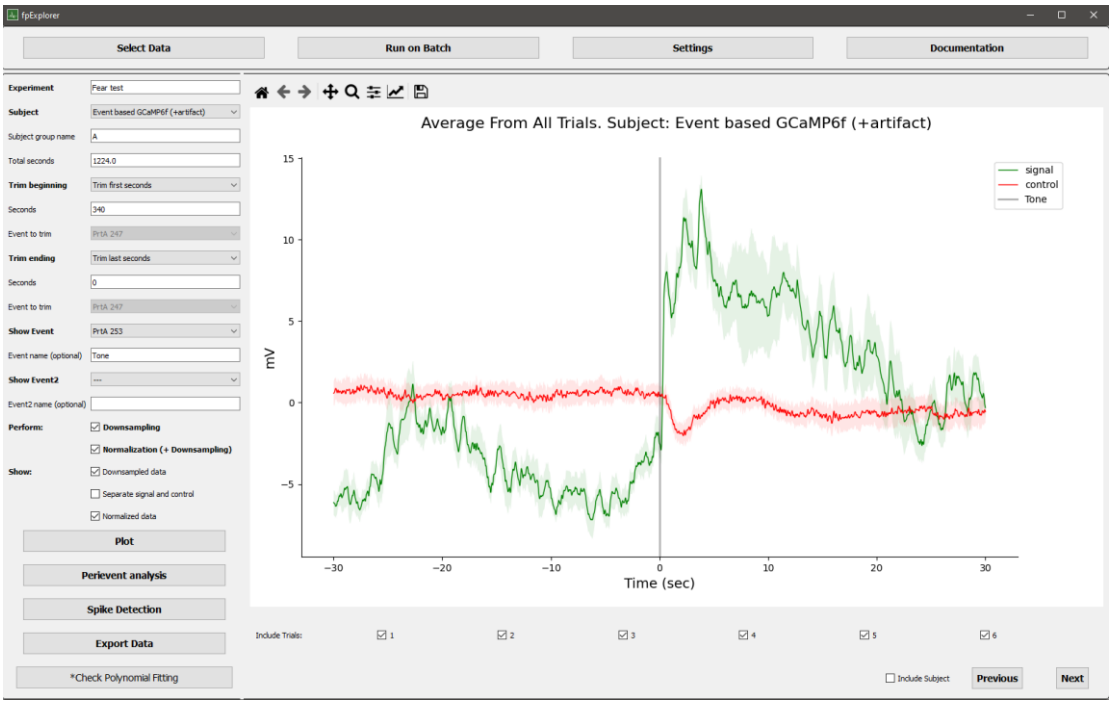

Fig14. The example of average perievent analysis.

- Z-score

Here, the user can visualize individual z-score traces around the selected event throughout the recording. First, user needs to select the event from the drop down menu and enter how many seconds before and after the event to show. Optionally, user can enter a custom event name. Then, user needs to define the time window for baseline (in seconds). That baseline must be within the selected data window. Negative integers must be used to indicate time before the event. This baseline window will be used to calculate median and median absolute deviation required for calculating z-score. (Fig15)

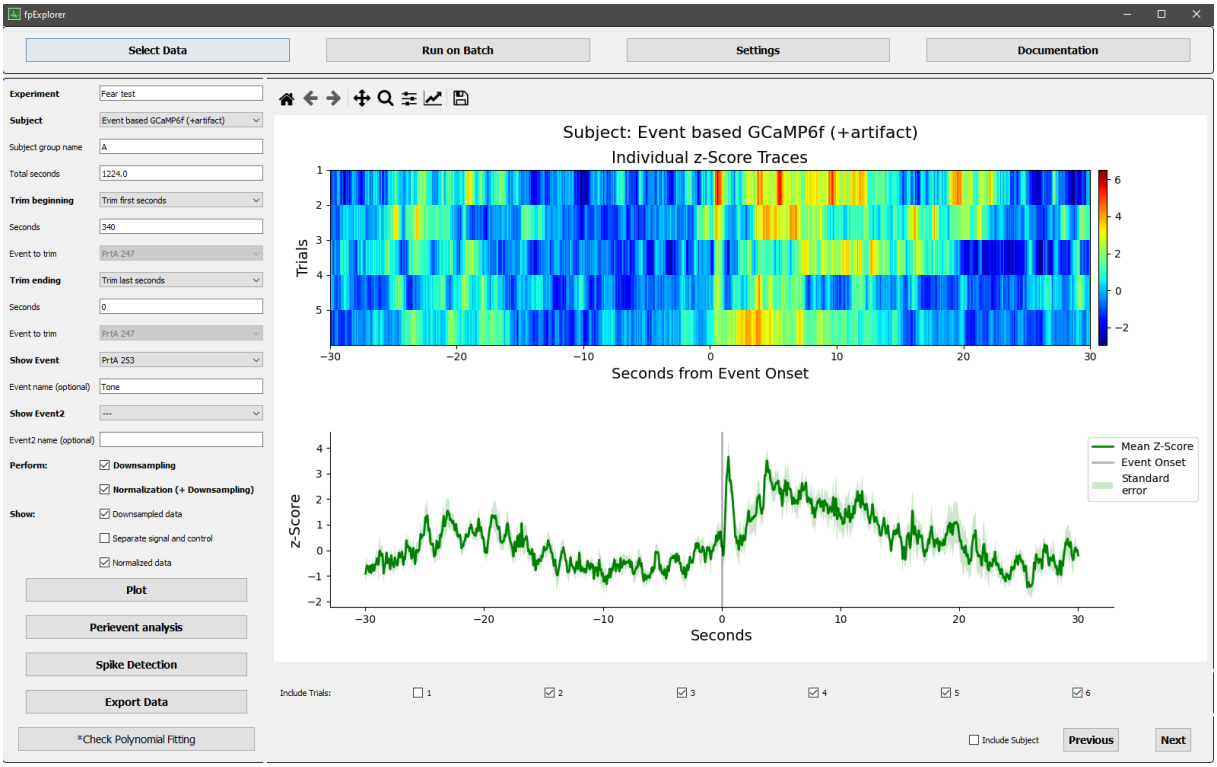

Fig15. The example of z-score visualization

- Area Under the Curve (AUC)

Here, the user can visualize Area Under the Curve as a bar chart of before and after the event. First, user needs to select the event from the drop down menu and enter how many seconds before and after the event. Optionally, user can enter a custom event name. Then, user needs to define the time window in seconds for the “Pre” event and the time window in seconds for the “Post” event. Negative integers must be used to indicate time before the event. The windows “Pre” and “Post” must be the same size. (Fig16)

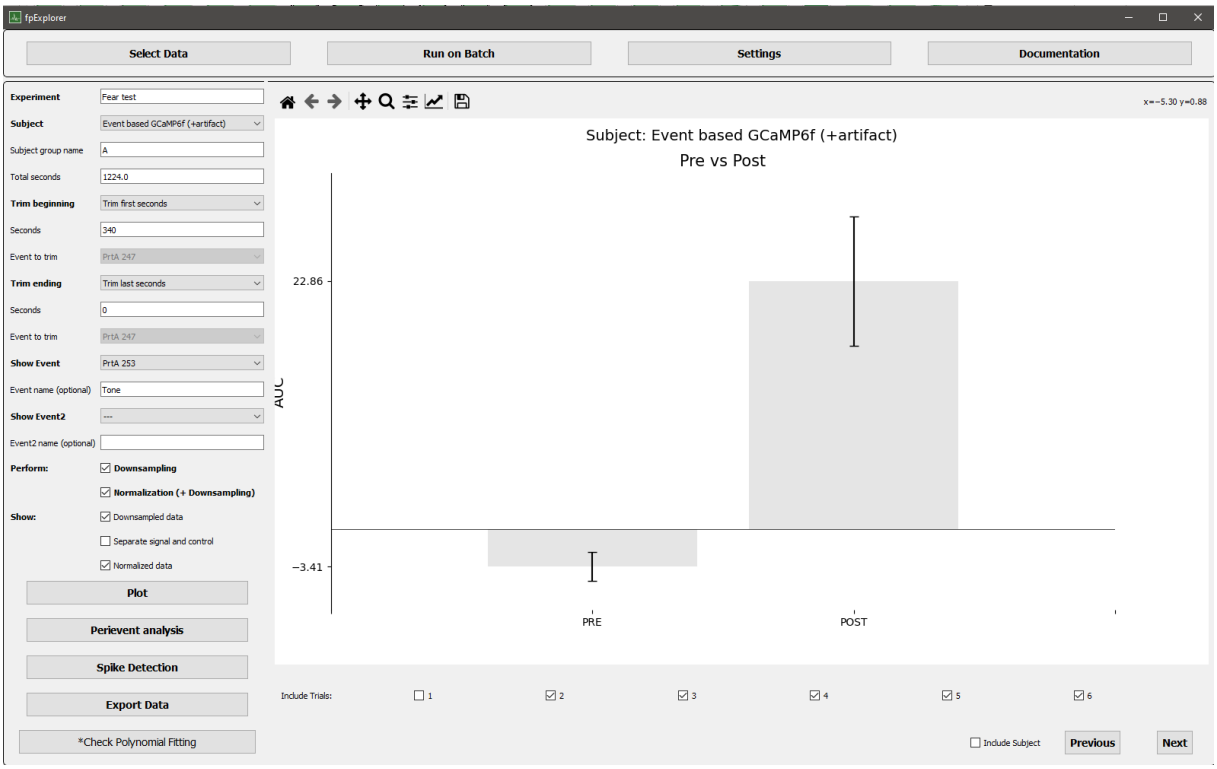

Fig16. The example of Area Under the Curve analysis.

Export Data

Here, the user can decide to export data for all of the above analysis. User also has the option to save these plots as both png and svg files(Fig17)

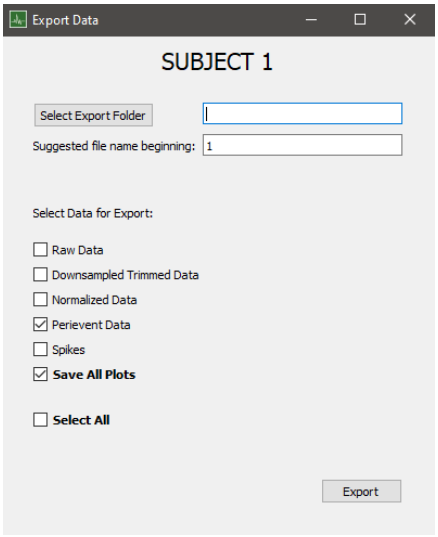

Fig17. Export Data window, from where the user can export both plots and data for perievent analysis.

2.3.2. Spike Detection

By clicking on “Spike Detection” button the user can define different parameters that will allow to detect potential spikes. (Fig18) The results of the spike detecting algorithm (from python’s numpy package) will be displayed as a graph to quickly verify the selected parameters. (Fig 19)

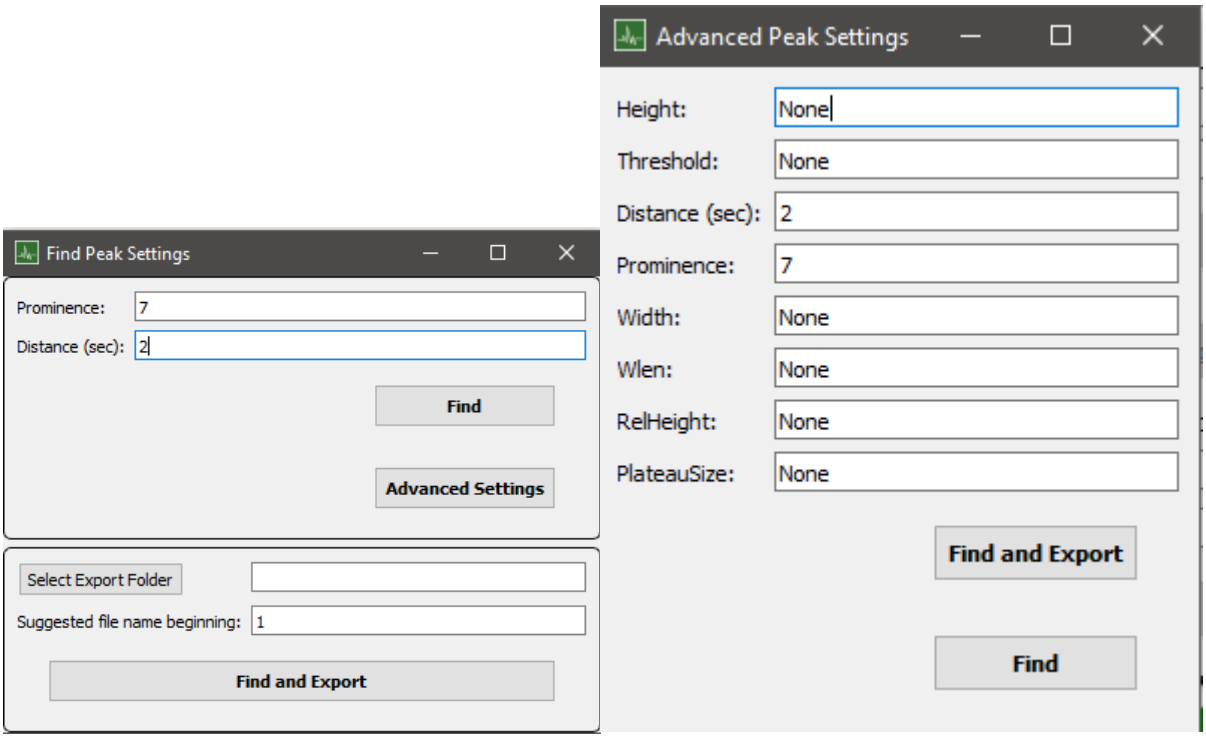

Fig18. Possible parameters that can help detect spikes withing the recording.

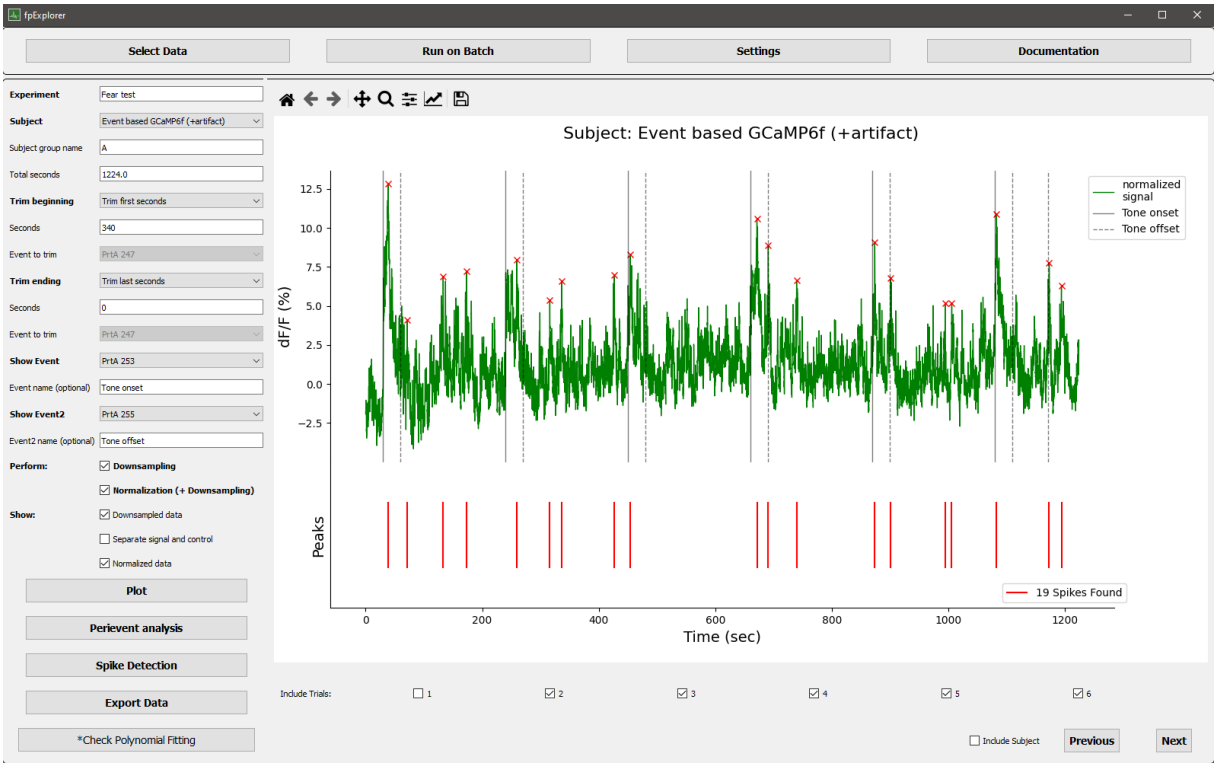

Fig19. The example result of spike detecting algorithm with user defined parameters.

2.4. Export Data

By clicking on “Export Data” button, user can choose to save all data as csv files as well as the corresponding plots (as png and svg). This option only allows to export one selected subject’s data at

a time. (Fig17) By default the application will recommend to save that data in the subfolder of each subject called "_fpExplorerAnalysis". But this can be changed by the user to any custom folder.

2.5. Run on Batch

"Run on Batch" button in the main application window opens a window with options that allow user to run automated analysis on multiple subjects. (Fig20)

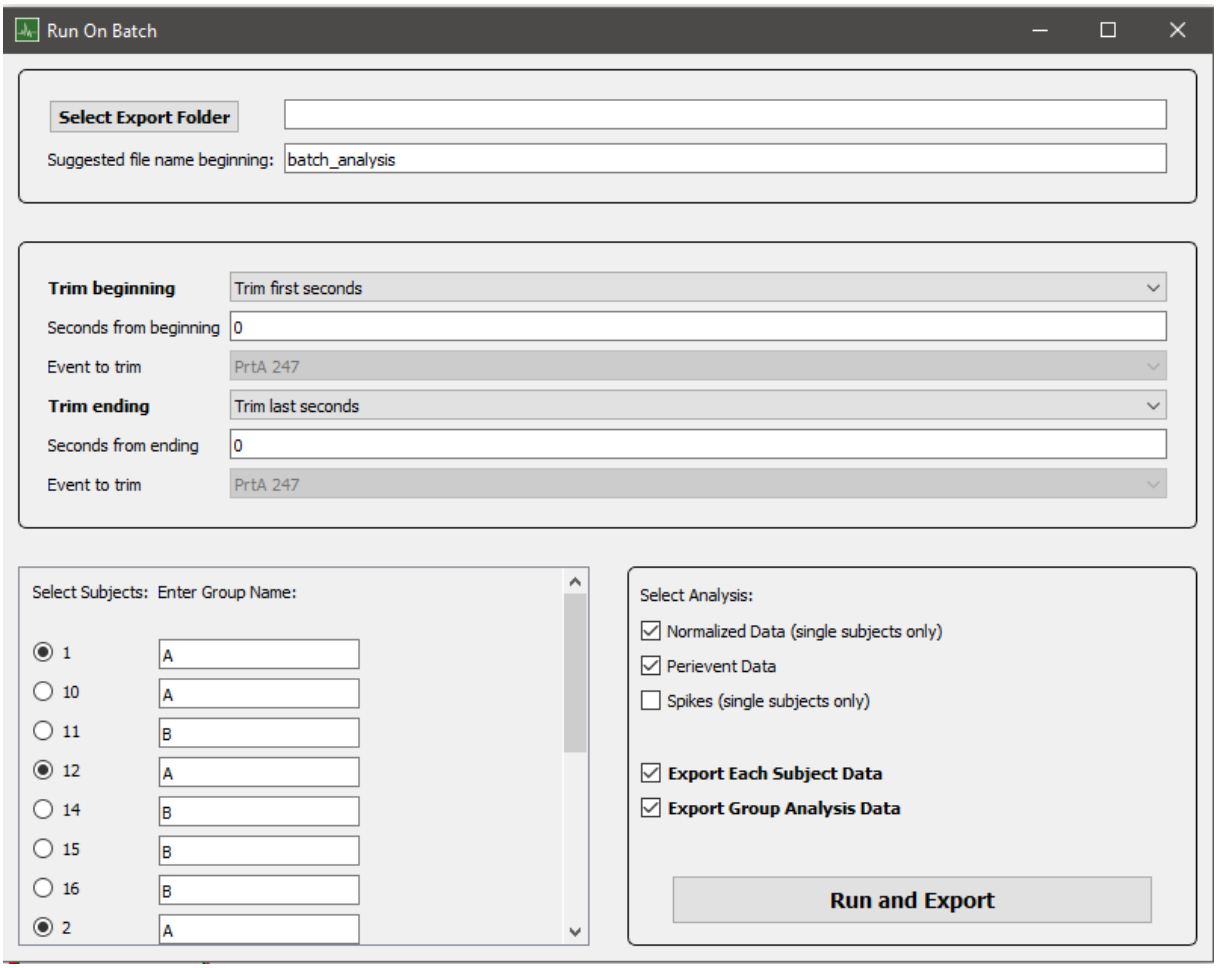

Fig20. Run on Batch window for multiple subjects data analysis.

The subjects can be selected within this window. However, the user also can "remember" interesting subjects by checking "Include Subject" checkbox at the bottom of each subject plot. Selected subjects will automatically appear as checked in the "Run on Batch" window.

The trimming will be applied to all analyzed data. Same as the current downsample, normalization and filter settings.

If the user selected any event in the Options next to the subject plots, the selected event will be shown on the plots.

"Export Each Subject Data" will save csv files with the selected single subject analyzed data as well as the plots.

"Export Group Analysis Data" will save csv files with group analysis data as well as the plots. (Fig21)

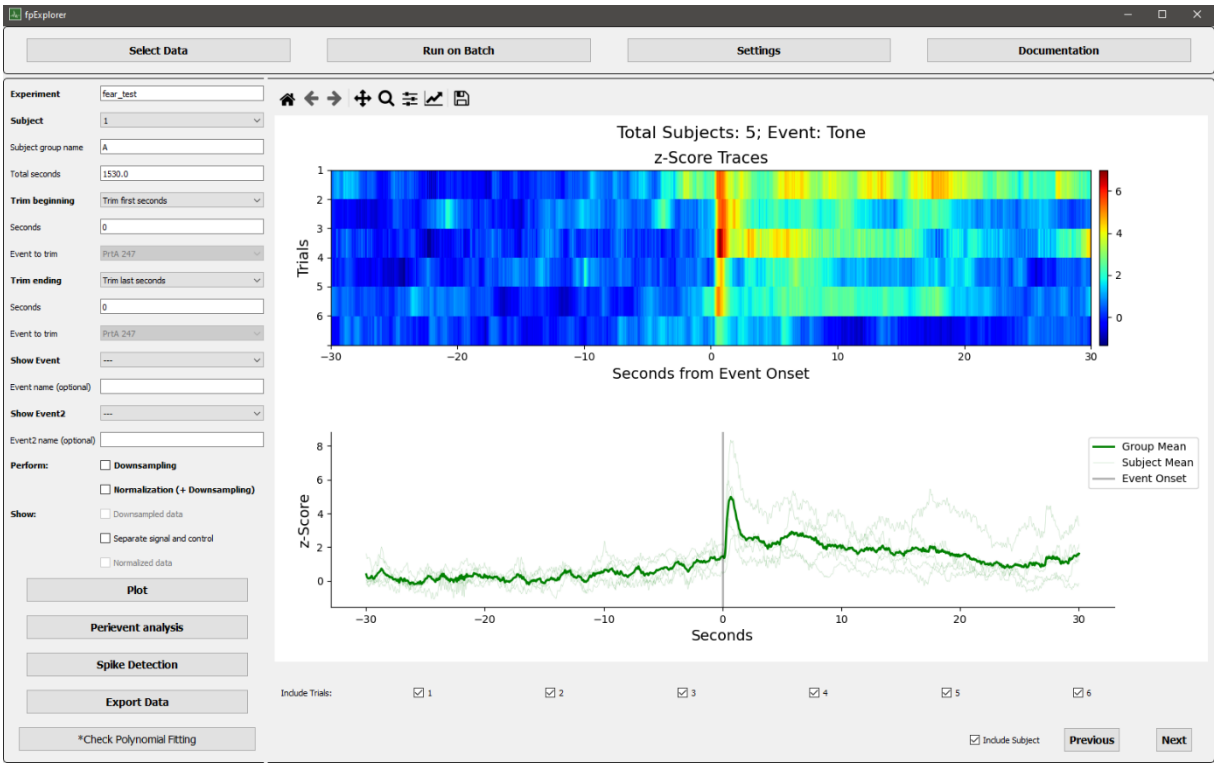

Fig21. The example of batch analysis.

3. Analysis Methods

3.1. Normalization

Before each normalization, data is downsampled by default. Then, the user has two methods to choose from when it comes to polynomial fitting. The following methods allow to fit the control to the signal, in order to reduce photobleaching in post processing.

The application relies on Python's numpy package to calculate polyfit.

(https://numpy.org/doc/stable/reference/generated/numpy.polynomial.polynomial.Polynomial.fit.html

https://numpy.org/doc/stable/reference/generated/numpy.polynomial.polynomial.polyval.html#numpy.polynomial.polynomial.polyval)

3.1.1. Modified Polynomial Fitting (Mulholland dF/F)

Patrick Mulholland noticed that often signals in both channels do not decrease equally. Therefore, he corrects both signals for decay over time by fitting them independently onto the time vector. Then he calculates deltaF for both channels, and then he substracts deltaF of the control channel from deltaF of the GCaMP to correct the GCaMP signal for motion artifacts.

Pseudo code:

```
# fit time axis to the GCaMP stream (numpy.polynomial.polynomial.Polynomial.fit)

bls_Ca = Polynomial.fit (time, GCaMP signal,1)

# numpy.polynomial.polynomial.polyval

F0Ca = polyval(time,bls_Ca.convert().coef)

# dF/F for the GCaMP channel
```

```
dFFCa = (GCaMP signal - F0Ca)/F0Ca *100
# fit time axis to the control stream
bls_ref = Polynomial.fit (time, control,1)
F0Ref = polyval(time,bls_ref.convert().coef)
# dF/F for the control channel
dFFRef = (control - F0Ref)/F0Ref *100
dFFnorm = dFFCa - dFFRef
# find all values of the normalized DF/F that are negative so you can next shift up the curve
# to make 0 the mean value for DF/F
negative = dFFnorm[dFFnorm<0]
dFF = dFFnorm - mean(negative)
```

3.1.2. Standard Polynomial Fitting (David Barker's dF/F)

This method assumes an equal decrease in signal in both channels over time.

The standard in the field has been to do polynomial fitting of the control signal onto the GCaMP signal, and use this fitted signal as "F". Then calculate "deltaF" by subtracting F from GCaMP signal at each time point. In David Barker's pMAT (Photometry Modular Analysis Tool), the following method is used after smoothing.*

Pseudo code:

```
bls = polyfit(control, GCaMP signal,1)
fit_line = multiply(bls[0], control) + bls[1]
dFF = (GCaMP signal - fit_line)/fit_line * 100
```

We decided to modify this method after some normalization problems when control channel started below signal and ended above signal. Numpy Python package was used for calculations and programming solution explained on forum.**

Pseudo code:

```
mu = mean(control_)
std = std(control)
# call numpy.polynomial.polynomial.Polynomial.fit, using the shifted and scaled version of control
cscaled = Polynomial.fit ((control - mu)/std, signal, 1)
# create a poly1d object that can be called
pscaled = Polynomial(cscaled.convert().coef)
# inputs to pscaled must be shifted and scaled using mu and std
F0 = pscaled((control - mu)/std)
dffnorm = (signal - F0)/F0 * 100
# find all values of the normalized DF/F that are negative so you can next shift up the curve
# to make 0 the mean value for DF/F
negative = dffnorm[dffnorm<0]
dff = dffnorm - mean(negative)
```

**https://github.com/djamesbarker/pMAT*

***https://stackoverflow.com/questions/45338872/matlab-polyval-function-with-three-outputs-equivalent-in-python-numpy*

What polyval with the extra mu-input in Matlab does is to use the polynomial function obtained with polyfit using a transformed (z-score transformation) x-vector that is scaled (i.e. std=1) and centered (i.e. mean=0), apply this polynomial function to the transformed x-vector to regress it onto the y-vector, and then "untransform" the regressed x-vector so that its values are in a numerical domain again (like the original x and y vectors) instead of in the z-score domain. The code on the forum should do exactly that, it's just more lines of code because it's basically writing things out step by step instead of using an all-in-one function in Matlab.

3.2. Z-score

Z-score operation is applied within the perievent analysis. It is performed on normalized data. Signals are converted into a robust z-score.

Pseudo code:

```
mad = median_absolute_deviation(dFF[baseline])

zscore = ((dFF - median(dFF[baseline]))/mad)
```

3.2.1. Z-score group analysis

In order to calculate group Z-score, mean values from all trials from all subjects (not from means from each subject) are calculated as well as standard errors. There is also an option to plot each subject's mean values from that subject's trials instead of standard errors.

Heat map plot represents each trial's mean values from all analyzed subjects.

3.3. Area Under the Curve (AUC)

The application calculates area under the curve using Python's sklearn package. It represents the area under z-scored values.
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html

3.3.1. Area Under the Curve (AUC) group analysis

For group analysis, area under the mean value from all subject's z-scores is calculated.
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html

3.4. Spike detection

The algorithms used in the application to detect spikes are explained here:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

scipy.signal.find_peaks(x, height=None, threshold=None, distance=None, prominence=None, width=None, wlen=None, rel_height=0.5, plateau_size=None)

Find peaks inside a signal based on peak properties. This function takes a 1-D array and finds all local maxima by simple comparison of neighboring values. Optionally, a subset of these peaks can be selected by specifying conditions for a peak's properties.

Parameters

x: sequence

A signal with peaks.

height: number or ndarray or sequence, optional

Required height of peaks. Either a number, None, an array matching x or a 2-element sequence of the former. The first element is always interpreted as the minimal and the second, if supplied, as the maximal required height.

threshold: number or ndarray or sequence, optional

Required threshold of peaks, the vertical distance to its neighboring samples. Either a number, None, an array matching x or a 2-element sequence of the former. The first element is always interpreted as the minimal and the second, if supplied, as the maximal required threshold.

distance: number, optional

Required minimal horizontal distance (>= 1) in samples between neighbouring peaks. Smaller peaks are removed first until the condition is fulfilled for all remaining peaks.

prominence: number or ndarray or sequence, optional

Required prominence of peaks. Either a number, None, an array matching x or a 2-element sequence of the former. The first element is always interpreted as the minimal and the second, if supplied, as the maximal required prominence.

width: number or ndarray or sequence, optional

Required width of peaks in samples. Either a number, None, an array matching x or a 2-element sequence of the former. The first element is always interpreted as the minimal and the second, if supplied, as the maximal required width.

wlen: int, optional

Used for calculation of the peaks prominences, thus it is only used if one of the arguments prominence or width is given. See argument wlen in peak_prominences for a full description of its effects.

rel_height: float, optional

Used for calculation of the peaks width, thus it is only used if width is given. See argument rel_height in peak_widths for a full description of its effects.

plateau_size: number or ndarray or sequence, optional

Required size of the flat top of peaks in samples. Either a number, None, an array matching x or a 2-element sequence of the former. The first element is always interpreted as the minimal and the second, if supplied as the maximal required plateau size.