Setup

1. Install python with version >=3.9

2. Install the following dependencies, as appropriate for your python version/environment
   - numpy
   - scipy
   - matplotlib
   - Bottleneck
   - pyabf

   Note that pyabf is only available via pip, it is not available via conda

3. Extract the contents of the zip into a folder of your choice (this folder is hereby referred to as the project root directory)

4. Done

Known-working configuration

If your particular setup does not work, you may wish to try the following exact versions of python and the dependencies – these are the versions used by the developer and is guaranteed to work.

- Python==3.9.13

- numpy==1.23.1
- scipy==1.9.1
- matplotlib==3.5.2
- Bottleneck==1.3.5
- pyabf==2.3.7

The dependencies in their exact versions can be installed easily using either of the following commands, if using pip.

- pip install -r requirements.txt

OR

- pip install "numpy==1.23.1" "scipy==1.9.1" "matplotlib==3.5.2" "Bottleneck==1.3.5" "pyabf==2.3.7"

Note: You may need to use pip3 instead of pip depending on your exact python setup

Usage Overview

The workflow when using the software consists of three main stages:

1. Prepare/label the data
2. Process the data to produce dataset(s)
3. Do machine learning with the produced dataset(s)

Supported data formats

- Axon Binary Format v2 (as openable by pyabf)
  - The first signal/trace in the file must be the current trace
  - Timestamps should be in seconds
  - Current values should be in nanoamperes

Where to put data

Your data files can be placed anywhere. They do not need to be in the project root directory or any subdirectory of the project root directory.

The software includes file discovery features for convenience; you should group relevant data files into their own (sub)directories.

Whilst the software works with files in network drives, we recommend you copy the files to the local computer/server before using the data with the software. The software requires the full data file, it does not "stream" the data – if you have not copied the full data file, it will be copied in full when you use the software on it anyway. The downside to letting the software do the copy is that if you have to abort you won't have a copy of the file readily available in the filesystem to simply restart, as the file would have been copied to memory or some other temporary location.

File structure in examples

For all terminal commands in this user guide, the working directory of the terminal should be the project root directory unless specified otherwise.

Note: You may need to use python3 instead of python in the commands depending on your exact python setup.

In the examples in this user guide, our project root directory is E:\Softwares\Demo\ML-Project and our data is stored in E:\Softwares\Demo\ML-Project\codetest_data, you need not follow the exact same directory structure and data location.

Pore info database

Certain features of the software require information about the nanopore used in data collection.

Pore information should be entered into poreinfo.csv in the project root directory.

This CSV file contains three columns:

1. pore_id – an arbitrary string identifier for the pore
2. open_pore_conductance (nS) – the open pore conductance of the pore, in units of nanosiemens.

3. pore_diameter (nm) – the diameter of the nanopore, in units of nanometers

The first row of the CSV is the header, it is ignored when parsing.

All non-empty rows after the header will be parsed as individual entries. For a row to be valid, the pore id and open pore conductance entries must be non-empty and valid. The pore diameter is currently unused and is hence optional (it is still recommended to be filled though for future use and reference).

Preparing data

Data is prepared using the script prepare_data.py. This script interactively lets you pick settings to be used for your data, and generates a JSON formatted settings file which can then be used in the processing stage.

To prepare a single data file:

- python prepare_data.py -f abf <PATH TO FILE>

To scan for and select data files:

- python prepare_data.py -f abf -s <PATH OF DIR TO SCAN>

Here -f abf specifies that the file format is ABF, and -s means to scan.

Target selection

If you launched on a single file, you will enter the data preparation/labelling session immediately for that file.

```
(base) E:\Softwares\Demo\ML-Project>python prepare_data.py -f abf codetest_data\DNA_pH8_LiCl_CH001_000_
CHN46_Unfiltered.abf
========== prepare_data.py ==========
Started at time: Fri Nov 25 12:37:15 2022
CWD: E:\Softwares\Demo\ML-Project
Arguments: Namespace(format='abf', scan=False, path=WindowsPath('codetest_data/DNA_pH8_LiCl_CH001_000_C
HN46_Unfiltered.abf'))

Trace was created from: "E:\Softwares\Demo\ML-Project\codetest_data\DNA_pH8_LiCl_CH001_000_CHN46_Unfilt
ered.abf"
Overwrite sampling rate loaded from data (100000.0Hz)? [1=Y,Enter=N] >
```

If you launched with scanning, you will be prompted for each file found on if you want to exclude that file.

```
(base) E:\Softwares\Demo\ML-Project>python prepare_data.py -f abf -s codetest_data
========== prepare_data.py ==========
Started at time: Fri Nov 25 12:41:37 2022
CWD: E:\Softwares\Demo\ML-Project
Arguments: Namespace(format='abf', scan=True, path=WindowsPath('codetest_data'))

Found "codetest_data\10MHz_BSA_2uM_4uL_pH7_1M_KCl_700mV_000.abf", exclude? [1=Y,Enter=N] > 1
Found "codetest_data\10MHz_HSA_2uM_4uL_pH7_1M_KCl_300mV_000.abf", exclude? [1=Y,Enter=N] > 1
Found "codetest_data\BSA_10uM_2uL_1MKCl_pH_7_CH001_000.abf_applied_voltage_300mV_sample_freq_200kHz_fil
tered_at_10kHz.abf", exclude? Existing settings file found. [1=Y,Enter=N] > 1
Found "codetest_data\DNA_pH8_LiCl_CH001_000.abf_applied_voltage_400mV_sample_freq_100kHz_filtered_at_5k
Hz.abf", exclude? Existing settings file found. [1=Y,Enter=N] >
Found "codetest_data\DNA_pH8_LiCl_CH001_000_CHN46_Unfiltered.abf", exclude? [1=Y,Enter=N] >
Found "codetest_data\DNA_longest_run", exclude? This is a set. Existing settings file found. [1=Y,Enter
=N] >
Confirm targets? [1=Y,0=N] >
```

In the example here we have excluded all files except the last 3.

We also see that for some of the files, the script reports "Existing settings file found" – if you have previously done preparation for a data, the script will pick it up and load the existing settings as prefill for this new preparation session.

We also see that for one folder, the script reported that it "is a set" – if you wish to use the same settings for a set of data files (e.g. large data that was saved as multiple smaller files), you can make a file named exactly this_is_a_set.txt in that directory (the content of the file can be anything, including blank). This makes the script treat the directory as a single set of data, rather than pick up individual files in that directory.
If we did not have the magic file in that directory above, the below is what we will instead see.

```
(base) E:\Softwares\Demo\ML-Project>python prepare_data.py -f abf -s codetest_data
========== prepare_data.py ==========
Started at time: Fri Nov 25 12:50:19 2022
CWD: E:\Softwares\Demo\ML-Project
Arguments: Namespace(format='abf', scan=True, path=WindowsPath('codetest_data'))

Found "codetest_data\10MHz_BSA_2uM_4uL_pH7_1M_KCl_700mV_000.abf", exclude? [1=Y,Enter=N] > 1
Found "codetest_data\10MHz_HSA_2uM_4uL_pH7_1M_KCl_300mV_000.abf", exclude? [1=Y,Enter=N] > 1
Found "codetest_data\BSA_10uM_2uL_1MKCl_pH_7_CH001_000.abf_applied_voltage_300mV_sample_freq_200kHz_fil
tered_at_10kHz.abf", exclude? Existing settings file found. [1=Y,Enter=N] > 1
Found "codetest_data\DNA_pH8_LiCl_CH001_000.abf_applied_voltage_400mV_sample_freq_100kHz_filtered_at_5k
Hz.abf", exclude? Existing settings file found. [1=Y,Enter=N] >
Found "codetest_data\DNA_pH8_LiCl_CH001_000_CHN46_Unfiltered.abf", exclude? [1=Y,Enter=N] >
Found "codetest_data\DNA_longest_run\DNA_pH8_LiCl_CH001_000.abf_applied_voltage_400mV_sample_freq_100kH
z_filtered_at_5kHz.abf", exclude? [1=Y,Enter=N] >
Found "codetest_data\DNA_longest_run\DNA_pH8_LiCl_CH001_001.abf_applied_voltage_400mV_sample_freq_100kH
z_filtered_at_5kHz.abf", exclude? [1=Y,Enter=N] >
Found "codetest_data\DNA_longest_run\DNA_pH8_LiCl_CH001_003.abf_applied_voltage_400mV_sample_freq_100kH
z_filtered_at_5kHz.abf", exclude? [1=Y,Enter=N] >
Found "codetest_data\DNA_longest_run\DNA_pH8_LiCl_CH001_004.abf_applied_voltage_400mV_sample_freq_100kH
z_filtered_at_5kHz.abf", exclude? [1=Y,Enter=N] >
Found "codetest_data\DNA_longest_run\DNA_pH8_LiCl_CH001_005.abf_applied_voltage_400mV_sample_freq_100kH
z_filtered_at_5kHz.abf", exclude? [1=Y,Enter=N] >
Found "codetest_data\DNA_longest_run\DNA_pH8_LiCl_CH001_006.abf_applied_voltage_400mV_sample_freq_100kH
z_filtered_at_5kHz.abf", exclude? [1=Y,Enter=N] >
Found "codetest_data\DNA_longest_run\DNA_pH8_LiCl_CH001_007.abf_applied_voltage_400mV_sample_freq_100kH
z_filtered_at_5kHz.abf", exclude? [1=Y,Enter=N] >
Found "codetest_data\DNA_longest_run\DNA_pH8_LiCl_CH001_008.abf_applied_voltage_400mV_sample_freq_100kH
z_filtered_at_5kHz.abf", exclude? [1=Y,Enter=N] >
Confirm targets? [1=Y,0=N] >
```

Where individual files in the directory are now picked up, rather than just the directory.

After confirming targets, the program proceeds to launch preparation sessions for each target not excluded, in sequence. Only one session will be launched for each set, not one per file in the set. The session will use the first file in the set for previewing and prefilling settings.

If you do not confirm targets, you will be prompted again for all targets found in the scan on if you wish to exclude them.

Interactive preparation session

Let's now walkthrough a preparation session.

At the start of each session, the absolute path of the file (or of the first file in a set for sets) will be shown.

You will also be shown a prompt which includes the sampling rate pulled from the data file, asking if you wish to overwrite it.

```
Trace was created from: "E:\Softwares\Demo\ML-Project\codetest_data\DNA_longest_run\DNA_pH8_LiCl_CH001_
000.abf_applied_voltage_400mV_sample_freq_100kHz_filtered_at_5kHz.abf"
Overwrite sampling rate loaded from data (100000.0Hz)? [1=Y,Enter=N] >
```

Check that the sampling rate shown matches what you are expecting. The overwrite may be used if your sampling device does not save the correct sampling rate to the data file, or if your data is corrupted (not really applicable to ABF files, this feature is mostly for other formats which have metadata saved separately that can be missing or easily corrupted).

You will then be prompted to choose a label for the data.

```
Enter label
[0=DNA, 1=BSA, 2=ConA, 3=BovineHb, 4=HSA, 5=BSA_HSA_5050, 6=BSA_HSA_7525, 7=BSA_HSA_2575, 8=BSA_HSA_Con
A_1_1_1, 9=BSA_BovineHb_5050, 10=BSA_BovineHb_7525][Enter=DNA] >
```

Enter the number corresponding to the correct label. You will have a chance to confirm if you accidentally entered the wrong label.

```
Enter label
[0=DNA, 1=BSA, 2=ConA, 3=BovineHb, 4=HSA, 5=BSA_HSA_5050, 6=BSA_HSA_7525, 7=BSA_HSA_2575, 8=BSA_HSA_Con
A_1_1_1, 9=BSA_BovineHb_5050, 10=BSA_BovineHb_7525][Enter=DNA] > 0
Confirm label is "DNA"? [1=Y,0=N] > 1
```

After confirming the label, you will need to enter the bias voltage used, in units of volts. The bias voltage can be signed. Enter the number as a float. There are no limits on the precision of the number beyond the usual hardware limitations. You do not need to prepend "+" for positive bias voltages.

```
Enter SIGNED voltage in volts [Enter=0.4] > 0.4
```

You will be shown the parsed value for confirmation. You should check that this value matches what you intended to enter.

```
Enter SIGNED voltage in volts [Enter=0.4] > 0.4
Confirm voltage of +0.400000V? [1=Y,0=N] > 1
```

Next you will be asked to enter the ID of the nanopore used in the collection of the data. This is an arbitrary string field.

```
Enter pore id [Enter=CODETEST1] >
```

As usual, you will have the chance to confirm the ID entered.

```
Enter pore id [Enter=CODETEST1] >lalala
Confirm pore id is "lalala"? [1=Y,0=N] > 1
```

After confirmation, the pore ID database will be checked to pull the information associated with that pore.

If the ID entered does not exist in the database, you will get the following message.

```
Pore id invalid or not in database. Please add pore to database if new pore.
Reload pore info from database before prompt pore id again? [1=Y,0=N] >
```

If the ID entered is incorrect, simply reply N and you will be prompted to enter the pore ID again.

```
Enter pore id [Enter=CODETEST1] >lalala
Confirm pore id is "lalala"? [1=Y,0=N] > 1
Pore id invalid or not in database. Please add pore to database if new pore.
Reload pore info from database before prompt pore id again? [1=Y,0=N] > 0
Enter pore id [Enter=CODETEST1] >
```

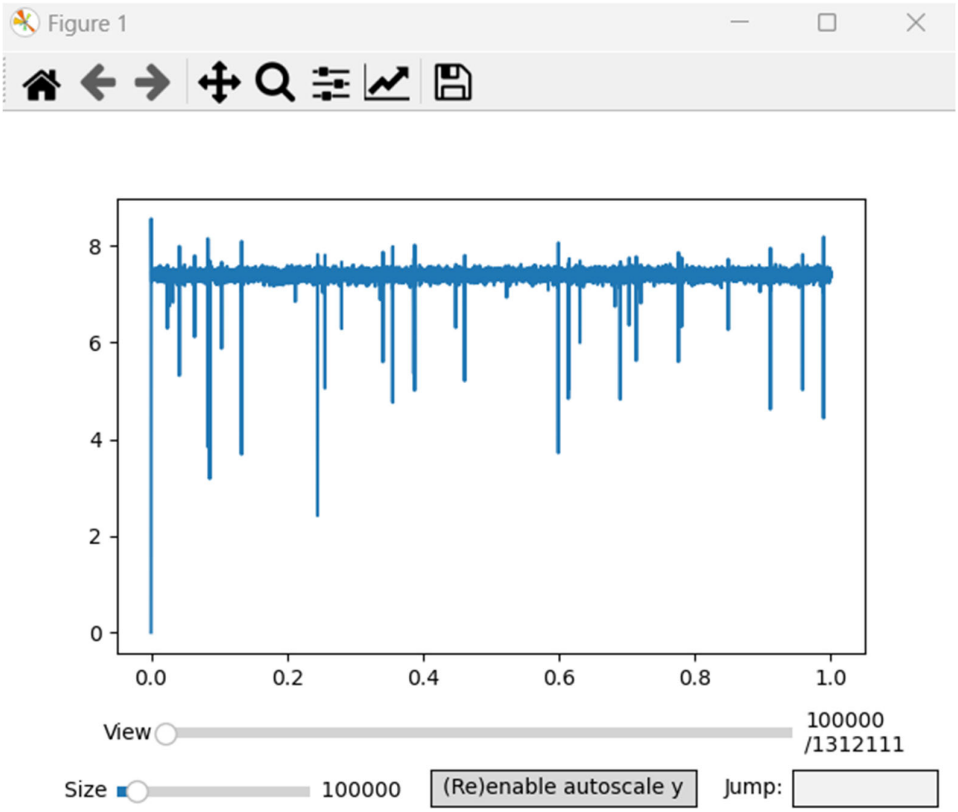
If the ID entered is correct, but just that the pore ID was not added to the pore database yet, you should add the ID and associated info into the pore database before replying Y. If you reply N, you will still be prompted to enter the pore ID again but the pore database will not be reloaded.

After entering a valid pore ID, the pore information fetched from the database will be displayed and you will be asked to confirm the pore information.

```
Enter pore id [Enter=CODETEST1] >
Confirm pore id is "CODETEST1"? [1=Y,0=N] > 1
Retrieved pore info:
PoreInfo({'open_pore_conductance': 100.0, 'pore_diameter': 20.0})
Confirm pore info? [1=Y,0=N] >
```

You will now be asked if event signals are peaks or dips. A matplotlib figure window will pop displaying the full trace to help you decide.

```
Are events peaks rather than dips? [1=Y,0=N][Enter=False] >
```



You can drag the "View" slider to move along in time. The "Size" slider controls the viewing window size. The number displayed at the end of the "View" slider is end index of the view. The "Jump" textbox can be used to jump to a particular view by entering the end index of that view. The "(Re)enable autoscale y" button re-enables y autoscaling, this is useful if you have used the zoom tool (magnifying glass), which will disable y autoscaling.

The data used for the example has dips not peaks.

```
Are events peaks rather than dips? [1=Y,0=N][Enter=False] >0
Confirm events are dips? [1=Y,0=N] > 1
```

Next you will be asked to enter the hard maximum event width in seconds. Any event wider/longer in duration than this value will be excluded from the results. This max event width will also be used as a base size we scale for windowing in event extraction later. We typically set this to 0.001 (1ms) for our data.

You will then be asked to enter the hard minimum event width in seconds. Any event shorter in duration than this value will be excluded from the results.

```
Enter (hard) minimum event width [Enter=2e-05] >
Confirm min event width of 2e-05? [1=Y,0=N] > 1
```

This then brings us to the end of the part of the session that deals with the trace information. A summary of the settings will be shown.

```
{'sampling_rate': 100000.0,
 'label': 'DNA',
 'signed_voltage': 0.4,
 'pore_id': 'CODETEST1',
 'peaks_not_dips': False,
 'max_event_width_seconds': 0.001,
 'min_event_width_seconds': 2e-05}
```

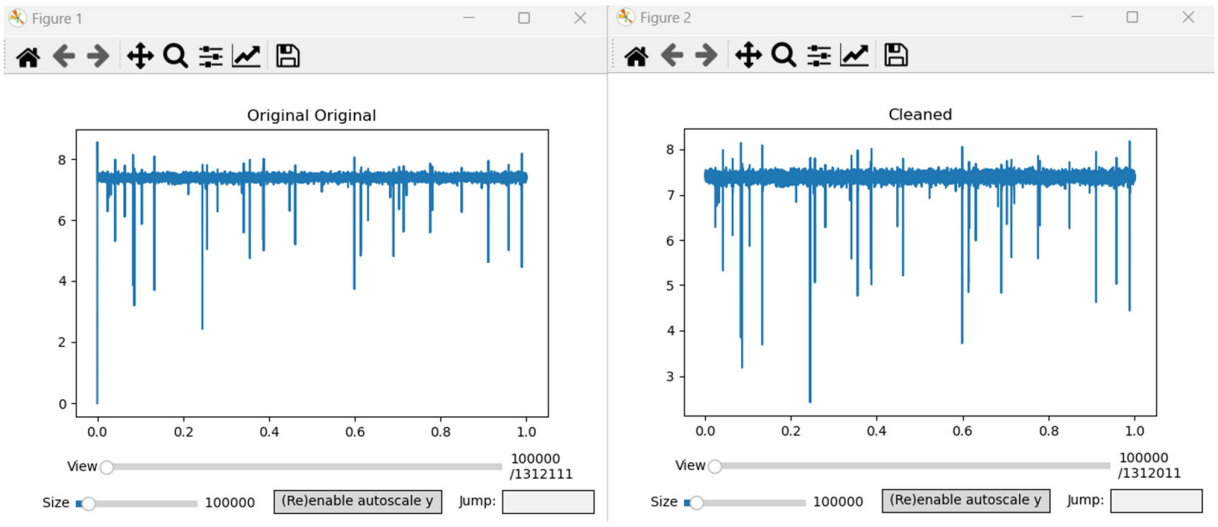
Now the session proceeds to the cleaning part.

If you have an existing settings file, you will be shown the existing settings and asked if you are happy with the existing settings.

```
Existing cleaners settings:
        0 zapmasker
                {'roll_seconds': 0.1, 'positive_zap_threshold': 10.0, 'negative_zap_threshold': -5.0, '
extra_collateral_damage_rolls': 2}
        1 trimmer
                {'slice_start': 100, 'slice_end': None}
Happy with existing cleaning settings? [1=Y,0=N] >
```

The number before the cleaner names are the index. Cleaners are applied in the sequence dictated by the indices. The settings of each cleaner are also displayed.

The original and cleaned trace will also be displayed in separate windows for your reference. If you cant find one of the windows, check if they are stacked on top of each other.

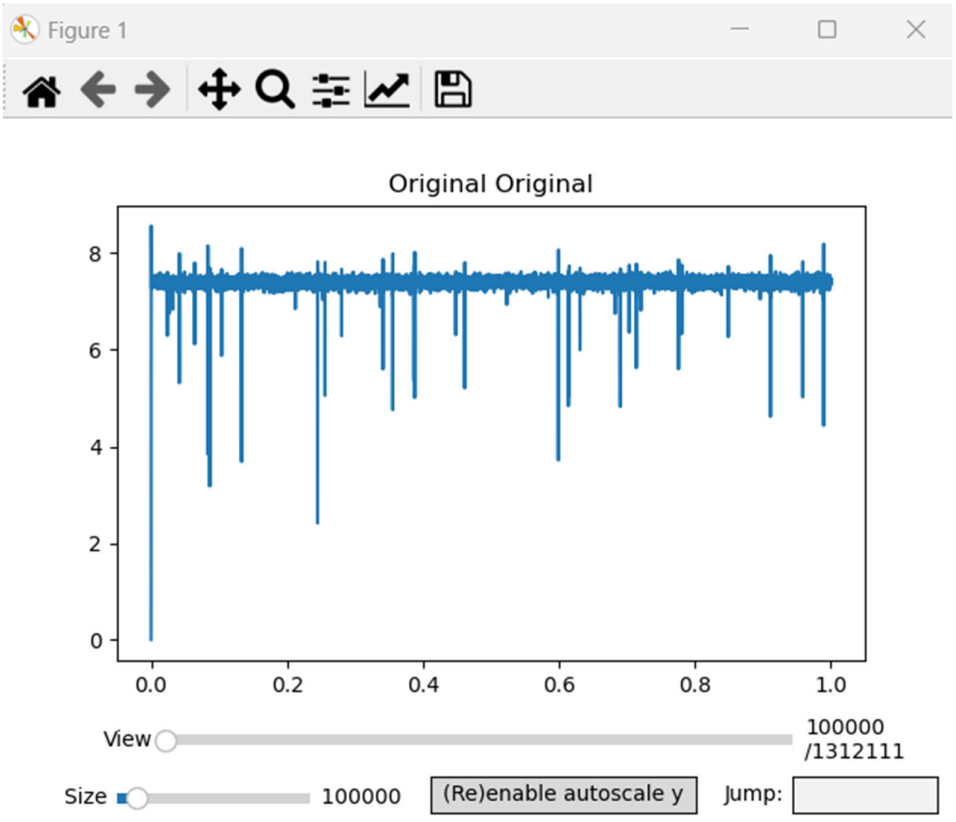


If you reply Y, you will proceed directly to event extraction settings setting.

If you reply N, or if you have no existing settings, you will be asked which cleaner you want to add.

```
Choose cleaner to add
[0=presetfilter, 1=lowpassfilter, 2=zapmasker, 3=trimmer, 4=manualmasker, 5=DONE] >
```

A window showing the trace (original) will also appear for your reference.



Multiple cleaners are supported, but note that you add cleaners one at a time. When shown the above menu, you will get to choose ONE cleaner, following which you will enter the settings setting for that cleaner. You will then be returned to the menu where you can add further cleaners. When you are done adding cleaners (or if you don't want to add any cleaner) choose the DONE option.

Note that cleaners are applied in the sequence they were added. You also cannot add the same cleaner twice.

The cleaner options are as follows:

- presetfilter – Apply a user defined custom filter (defined in code)
  - The only preset filter at the moment is "Elements200kHz_Custom35kLPF", which is a 35kHz LPF followed by a 35k-60k critical frequency bandstop, both butterworth and 8th order. Based on our spectrum analysis, we did not see any meaningful frequency content above 35kHz. The bandpass is to make the cutoff sharper.
- lowpassfilter – Apply a low pass filter
  - Note that a forward and backward pass is done when applying the filter. This eliminates the phase response of the filter (which is undesirable as it distorts the signals) entirely.
- zapmasker – Detect and nan out zaps
- trimmer – Trim the start and/or end of the trace off
- manualmasker – Manually nan out regions of the trace

Normally you will want the zapmasker and the lowpassfilter (or presetfilter). The order between these two is not important, though we have usually applied lowpassfilter first habitually.

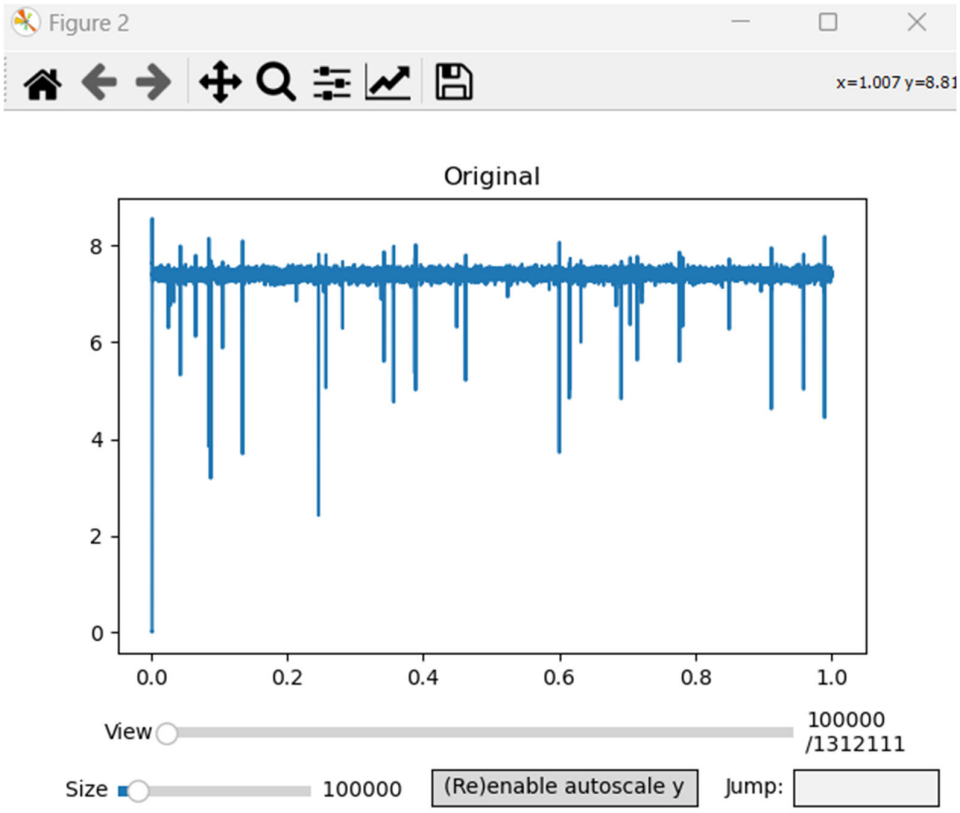
We shall just go through the zapmasker and the lowpassfilter as part of the user guide.
The lowpassfilter cleaner is very straightforward to use. You will be prompted for the critical frequency in Hz, and then the number of taps (order of the filter) and thats it. The result will not be shown at the end of the process, but a summary of the (two) parameters are displayed at the end as usual.

```
Choose cleaner to add
[0=presetfilter, 1=lowpassfilter, 2=zapmasker, 3=trimmer, 4=manualmasker, 5=DONE] > 1
Enter critical frequency in Hz > 10000
Enter N [Enter=8] >
Confirm filter parameters? [1=Y,0=N] > 1
{'fc': 10000.0, 'N': 8}
```

After setting the lowpassfilter you will now be brought back to the cleaner selection as mentioned. You will notice that lowpassfilter has now disappeared from the options as the software does not allow the same cleaner to be added twice. Use presetfilter if you need more complicated filtering setups, including cascading filters.

```
Choose cleaner to add
[0=presetfilter, 1=zapmasker, 2=trimmer, 3=manualmasker, 4=DONE] >
```

On choosing the zapmasker, you will be shown the "original" trace. Here "original" means the trace before the application of the currently being configured cleaner, but after the application of previous cleaners; "Original original" refers to the trace before any cleaner was applied.



You will also be prompted on whether there are any positive zaps in the trace, followed by whether there are any negative zaps in the trace. Positive zaps are those which give large positive currents, and negative zaps are those which give large negative currents.

For our example data, we only have negative zaps.

```
Any positive zaps? [1=Y,0=N] > 0
Any negative zaps? [1=Y,0=N] > 1
```

You will be then prompted to enter the threshold over (positive) or under (negative) which we will detect the relevant section of the trace as a zap. These sections will be masked (nan-ed). Note that negative thresholds must have a negative sign. For our data, we typically use -5.0V.

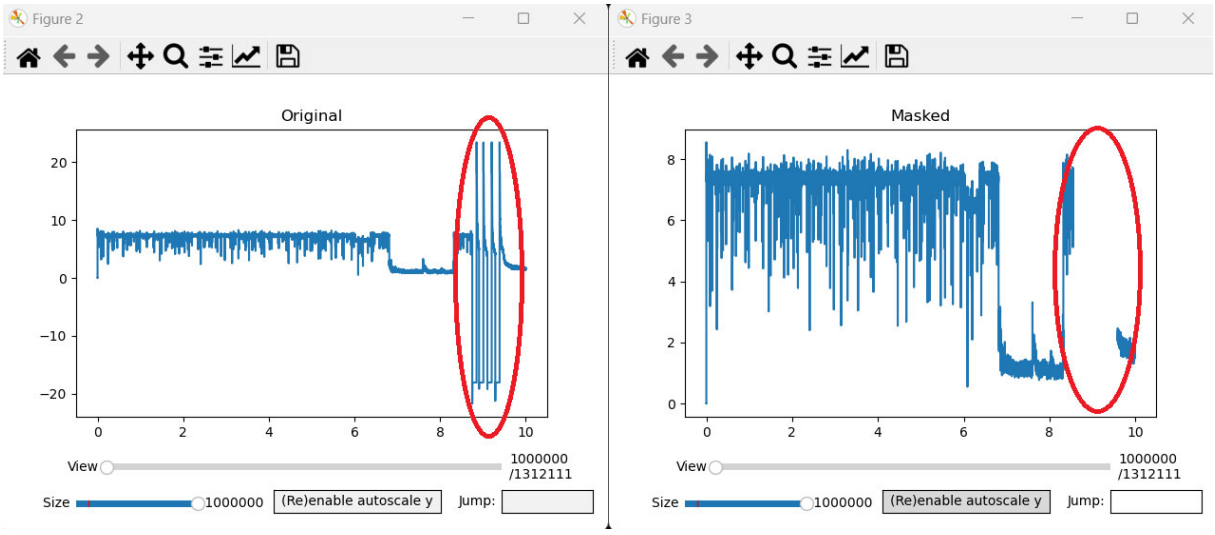
After specifying the threshold, you will be prompted to enter the number of seconds to use for rolling and the number of times to roll. Each section of trace detected as a zap, will be rolled left and right

by that amount of seconds and by the specified amount of times. Those regions affected by the rolling of the mask will also be masked. This deals with non-rectangular zap pulses and any transients leading up to or persisting a while after a zap, at the cost of some potential collateral damage. Note that the number of rolls specified is the total for one side only, not the total for both left and right. The total for left and right will be two times the entered value.

You should generally be liberal with the roll duration and number of rolls, as a long trace has plenty of events that we can afford a few being lost from collateral damage, and since non-events being picked up as events will degrade the quality of the dataset for machine learning. (Low SNR regions like within and in the transients of zaps can have high rates of non-events being picked up due to our SNR threshold based event detection method)

We typically use a roll of 0.1s and 2 rolls (4 total). 0.1s corresponds to the half period of the zap pulses used by our devices and also our manual momentary polarity flip zaps.

After entering the parameters you will be showed the results in a new figure window. You should verify that all zap containing regions have been satisfactorily masked without excessive collateral damage. Note that masked regions may be hidden automatically by matplotlib's auto xlim; use a larger view size to verify the masking for these regions.



Confirm you are satisfied in the prompt if the masking is ok. Otherwise enter N to repick parameters. As usual, after confirmation you will be shown a summary of your parameters. If your data has things that you want to remove but are unable to with the zapmasker, you can use the manualmasker (one-

off issues), write a custom cleaner (for classes of data with the same issue), or use an external program to clean your data.

```
Satisfied with results? [1=Y,0=N] > 1
{'roll_seconds': 0.1,
 'positive_zap_threshold': None,
 'negative_zap_threshold': -5.0,
 'extra_collateral_damage_rolls': 2}
```

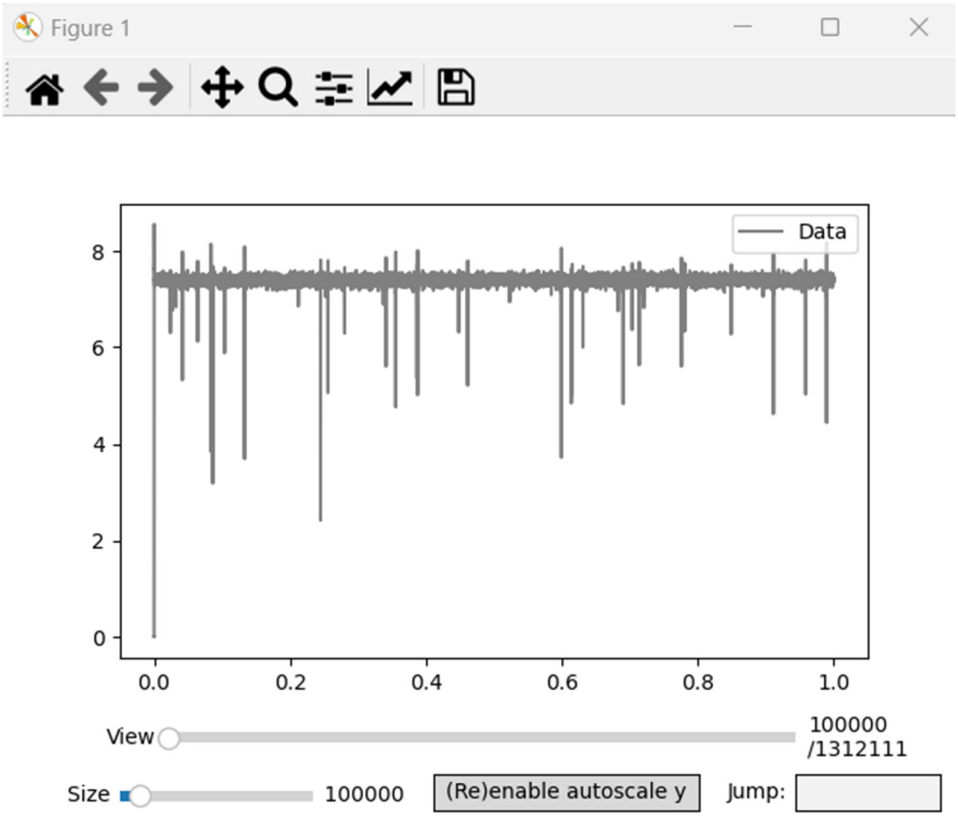
For this example, we are now happy with the cleaner settings, so simply reply DONE on the cleaner add prompt.

```
Choose cleaner to add
[0=presetfilter, 1=trimmer, 2=manualmasker, 3=DONE] > 3
```

The next step is to select an event extractor to use. Currently there is only one option – the ftrtextractor, so just reply that. (ftrtextractor stands for faster than real time extractor)

You will then be shown the base window size (in units of seconds) again (this is the max event width you set earlier) and be asked the baseline window scale. This is the factor the base window size is multiplied by to give the baseline window size. A **section** of the trace is also shown in a figure window. This section is only up to index 100,000 of the trace for performance reasons as it will be used for previews.

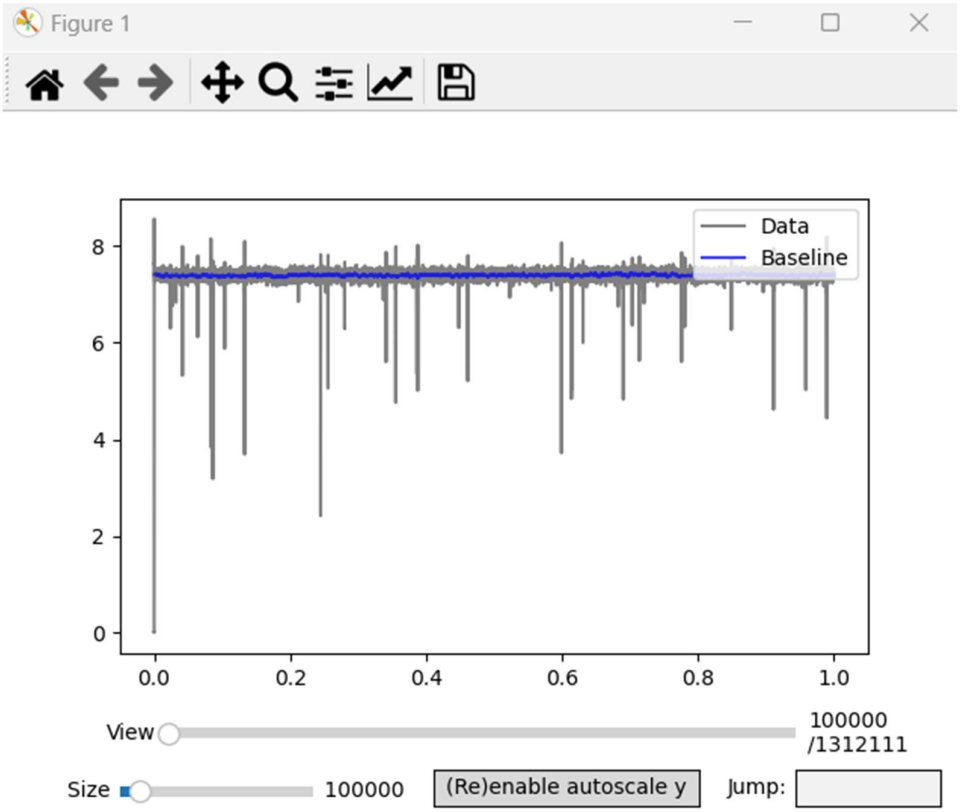
```
Base window size is 0.001s
Enter baseline window scale [Enter=3.0] >
```



Typically a baseline scale of 3-10 suffices. The baseline is a moving median. A smaller scale will tend to better follow short-duration changes in the trace whilst a larger scale will tend to ignore short-duration changes more.

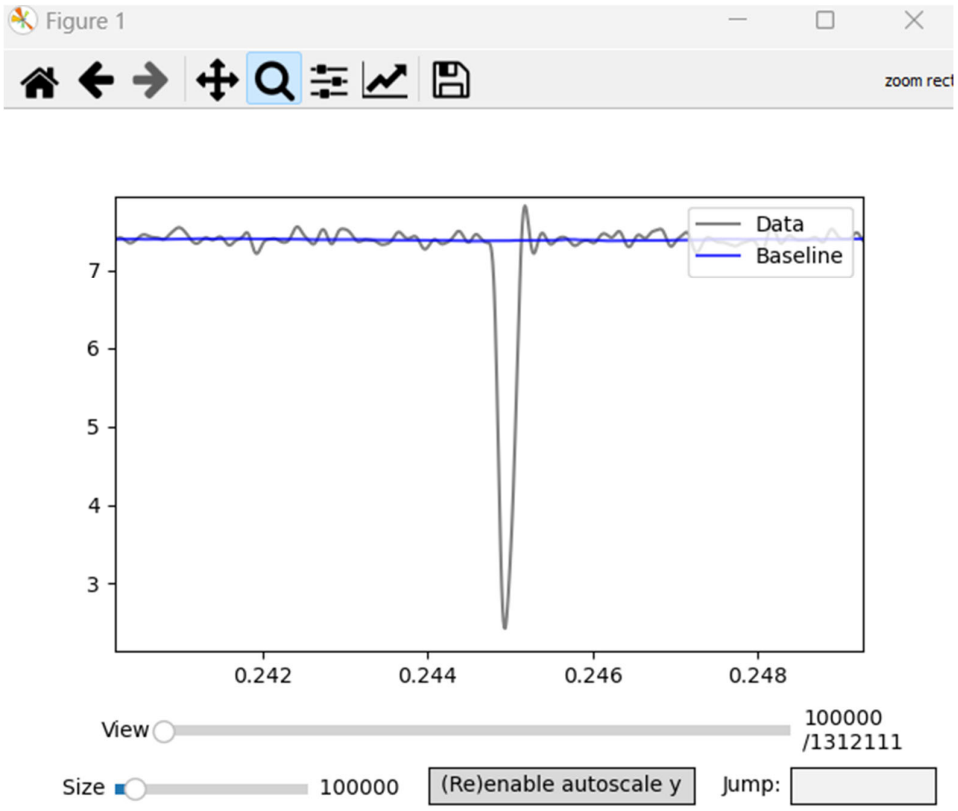
After you enter a scale and confirm you will be shown the baseline window size in units of samples and a preview of the baseline will be shown on the figure.

```
Enter baseline window scale [Enter=3.0] > 3
Baseline window size is 301 samples
Accept baseline? [1=Y,0=N] >
```

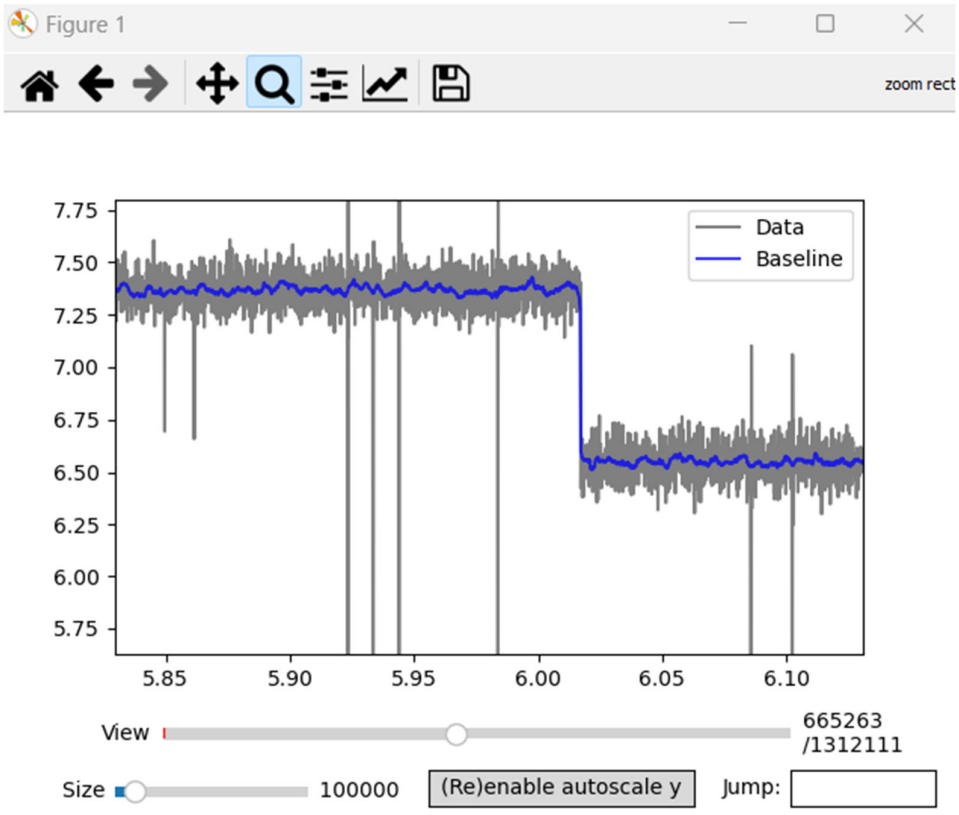


You should use the zoom tool to check the baseline is large enough that it is "flat" within events, but small enough that changes in the real baseline due to blockages/blockage-like interactions are still followed. You may find the "(Re)enable autoscale y" button, the home view tool (house icon) and the previous view (left arrow icon) tool helpful in switching between the zoom-out view and the zoomed in view

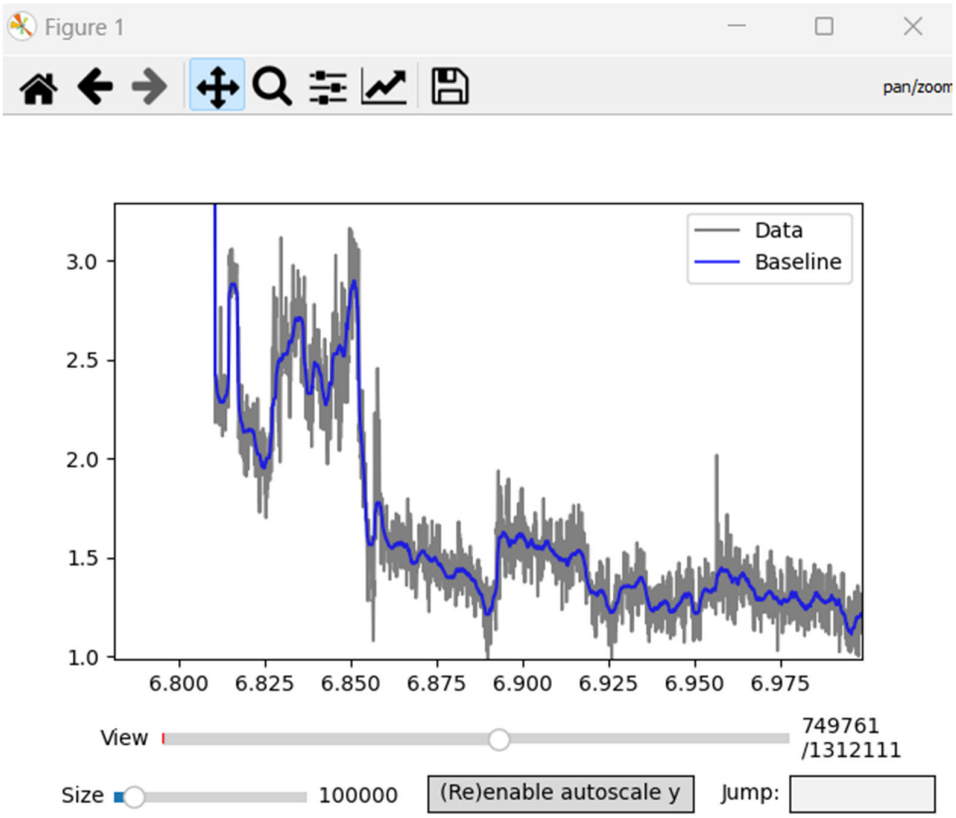
Below is an example of an event for which the baseline is good.



and an example of where the baseline followed the step change in the real baseline well

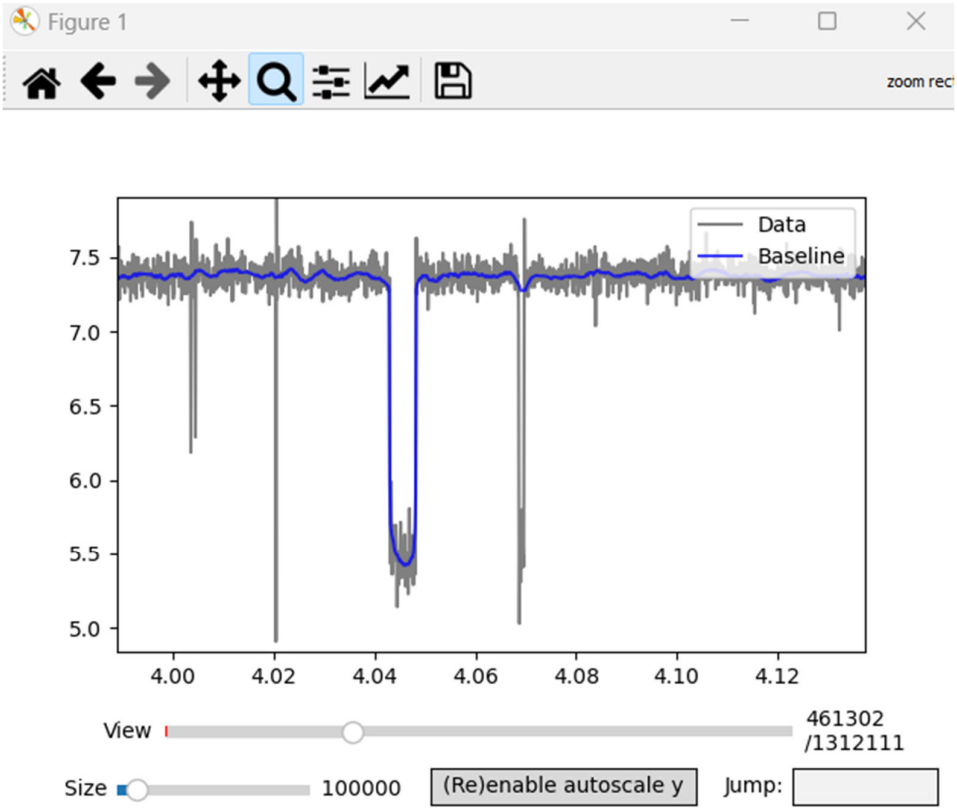


It may however be tricky to decide whether the baseline is good or not for some regions though.



These are typically in regions where there is no real event, so consider masking or trimming these sections away if you have a lot of trouble with them.

You may also have trouble deciding whether some "events" are real events or not. One such example is below, where the "event" is just below the 1ms maximum but also is not exactly short width-ed and neither a flat level nor a sharp event.



Regardless of if you decide to include or exclude these, you should be consistent in applying your decision across your dataset.

Note also that if you have very high event density, which can occur if you used high concentration samples in your data collection experiments, you may also have trouble setting a good baseline – multiple events that are not very distinguishable from noise from a signal point of view, especially if these events are of very short duration. A few long duration events scattered within these regions however are the telltale signs to the human that these noise looking signals are not actually noise.

Whilst it is best to simply avoid getting into this situation using lower concentration samples, you can technically still establish a reasonable baseline, as long as you are happy to only cater for the longer duration events, discarding the shorter duration events treating them as noise when setting the baseline. That or you will have to accept that your baseline will just be poor if you want to keep most of your events.

Anyway, once you are happy with the baseline, confirm in the prompt. If you are not happy, reply N and you will get to choose the baseline scale again. The preview will also be updated to the new baseline you set.

After confirming the baseline, the next step is to set the parameters for the "moving standard deviation" which characterises the amount of noise and which will be scaled and added to the baseline later to make the event threshold line. "Moving standard deviation" here as we will do a smoothing of the actual moving standard deviation, not just simply take it as is. Events spike the standard deviation, smoothing removes that effect. When we refer to the standard deviation line, we generally refer to the line given by "adding" the smoothed moving standard deviation shifted to the baseline, where "adding" is either addition or subtraction depending on if events are peaks or dips respectively.

```
Accept baseline? [1=Y,0=N] > 1
Enter std window scale [Enter=3.0] >
Std window size is 301 samples
Enter std smoothing scale factor [Enter=2.0] > 6
Accept std? [1=Y,0=N] >
```

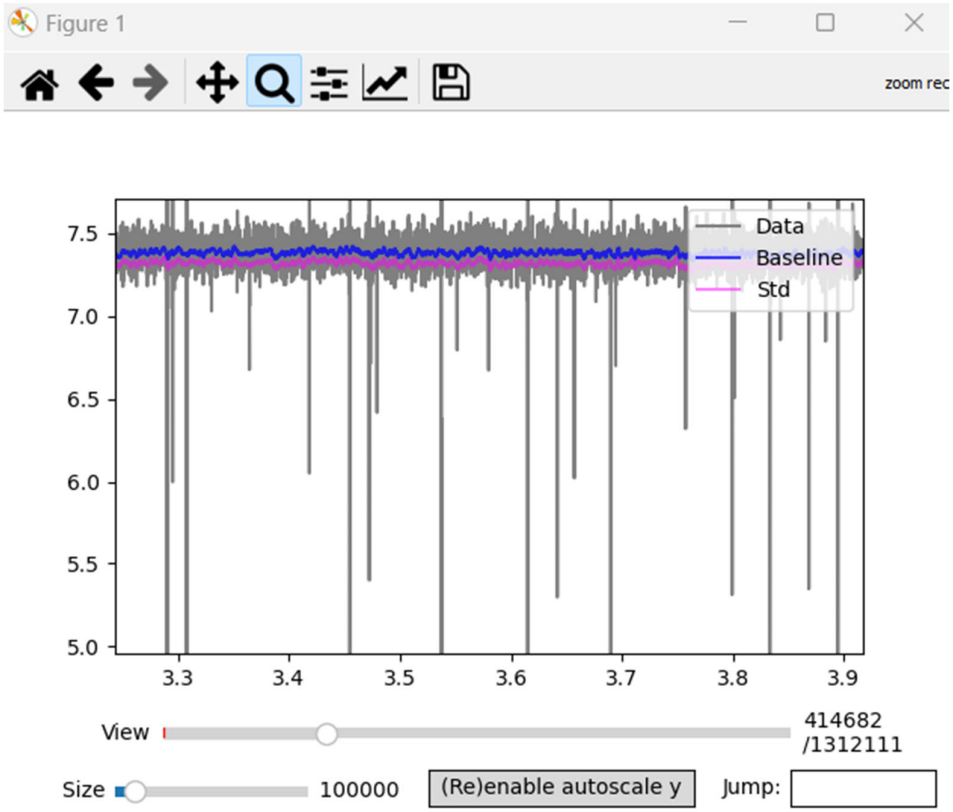
Generally, a scale same as the baseline usually works. Increasing the scale increases the width over which standard deviation is computed and is useful if you have high event density so that noise is considered over a larger region, though beyond a certain point it becomes worse again as you have included too many events in a window.

The smoothing scale factor is the scale is used for the smoothing moving median applied on the moving standard deviation line. It is NOT multiplied by the standard deviation scale; it is directly multiplied to the base window size. Generally either 2-3 or two times your standard deviation scale works well, though in some cases (extremely tall events, or high event density) you may have to go up to 20-80 or multiple times your standard deviation scale.

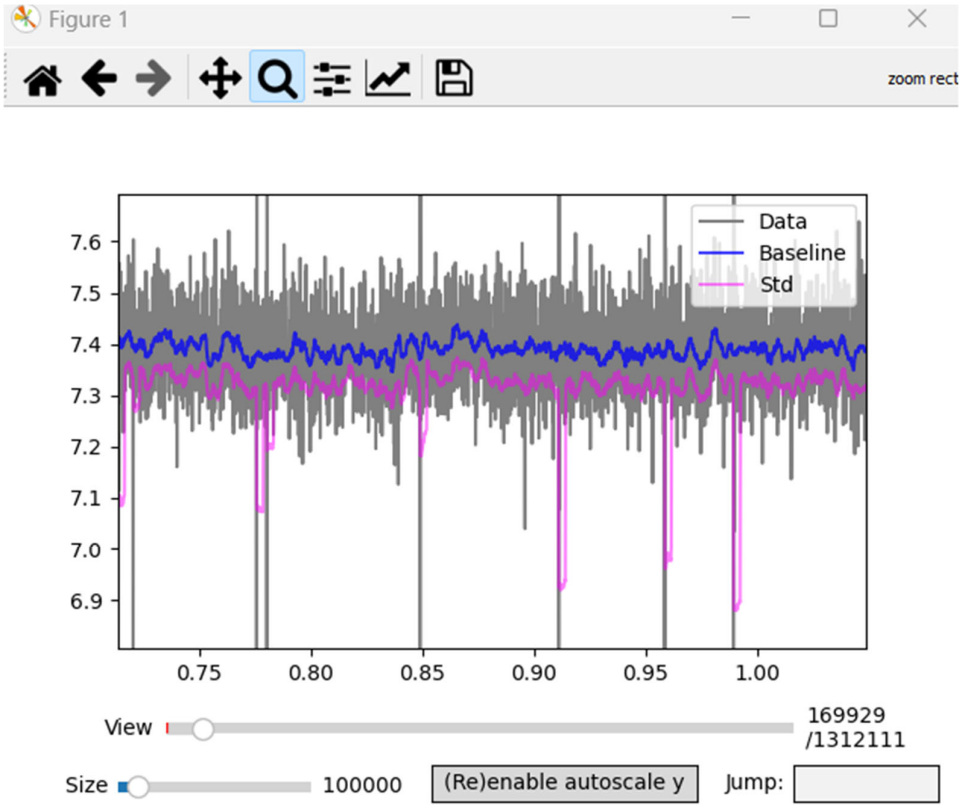
As before a preview of the standard deviation line is shown on the figure. If you chose good parameters, you should not observe any bumps or spikes in the line around events. The line should also generally be at the same distance from the baseline throughout the trace. If this is not the case, or if you see the separation increasing in a region you will need to fix that with a larger smoothing factor. The base scaling factor only affects the separation with the baseline in most cases.

Note that smaller bumps or spikes may not be visible until later on, where we scale the standard deviation before adding to the baseline to make the threshold line, so do not spend too much time on this step on the first try.

Example of a good standard deviation line below.



An example of a poor standard deviation line:

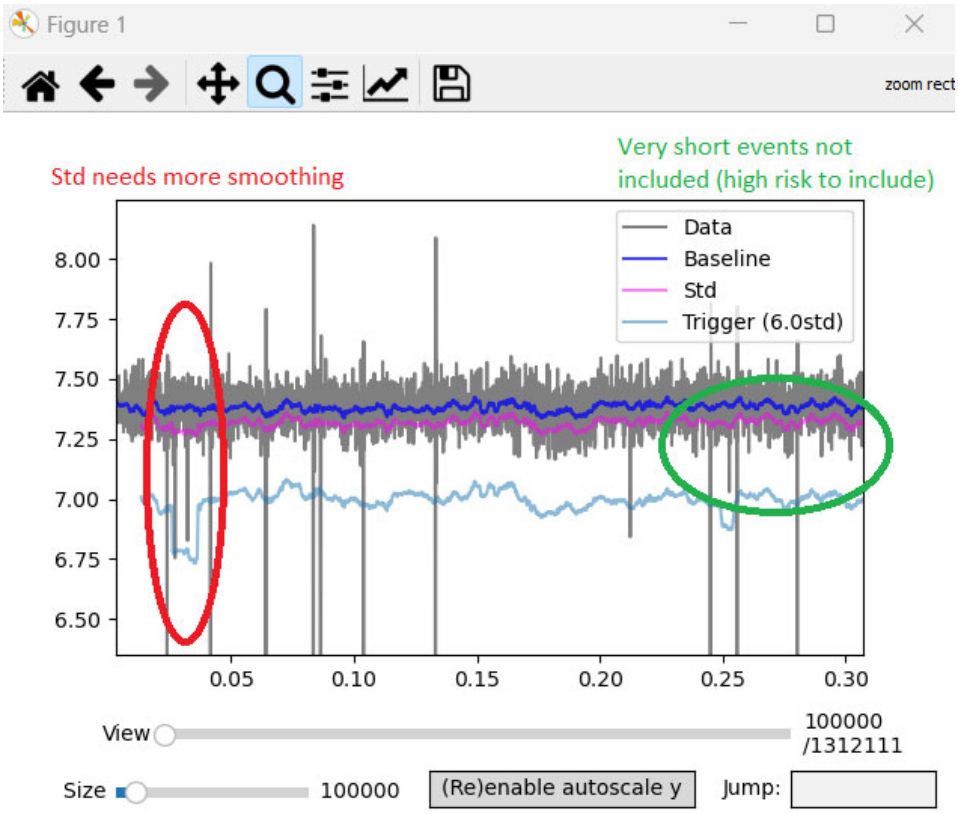


Once the standard deviation line is set, the next three steps are to set the threshold line, the start line and the end line. Each of these are the smoothed standard deviation multiplied by a factor (which we simply say units of standard deviation) added to the baseline.

The threshold line is used to detect events whilst the start and end line are used to define the expanse of each detected event.

The threshold line should be set such that most events cross the threshold. Be careful however to not be confused by large noises, the threshold line should not be too close to the noise floor. Non-events being detected at events will have a severe impact on the machine learning accuracy. Typically 6-8stds is the sweet spot, 10 onwards you are excluding most medium height events and 3 is typically too low. If your data is very clean (events all very tall and noise is very low) 4-6 or lower may work.

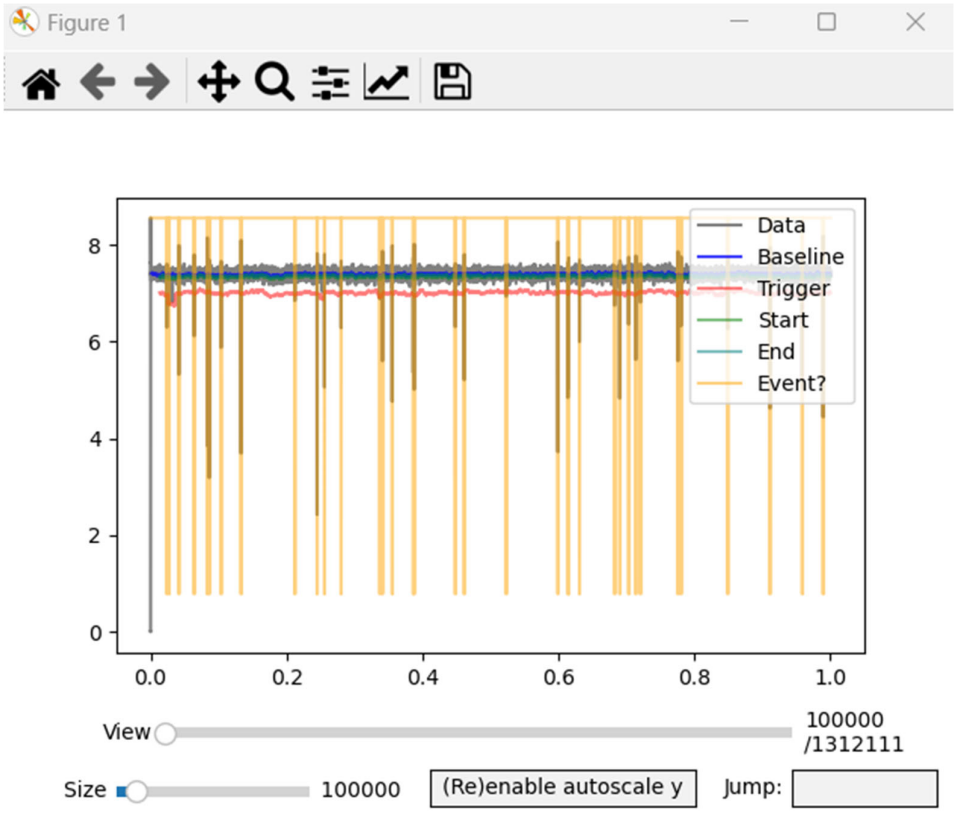
Below is an example with a 6std trigger line.



From the trigger line, you may notice issues with the standard deviation that you may not have seen previously as well. Simply just go through the remaining two steps of setting the start and end lines with arbitrary values then reply N when asked if you are happy with the results and you will be able to start again from baseline setting.

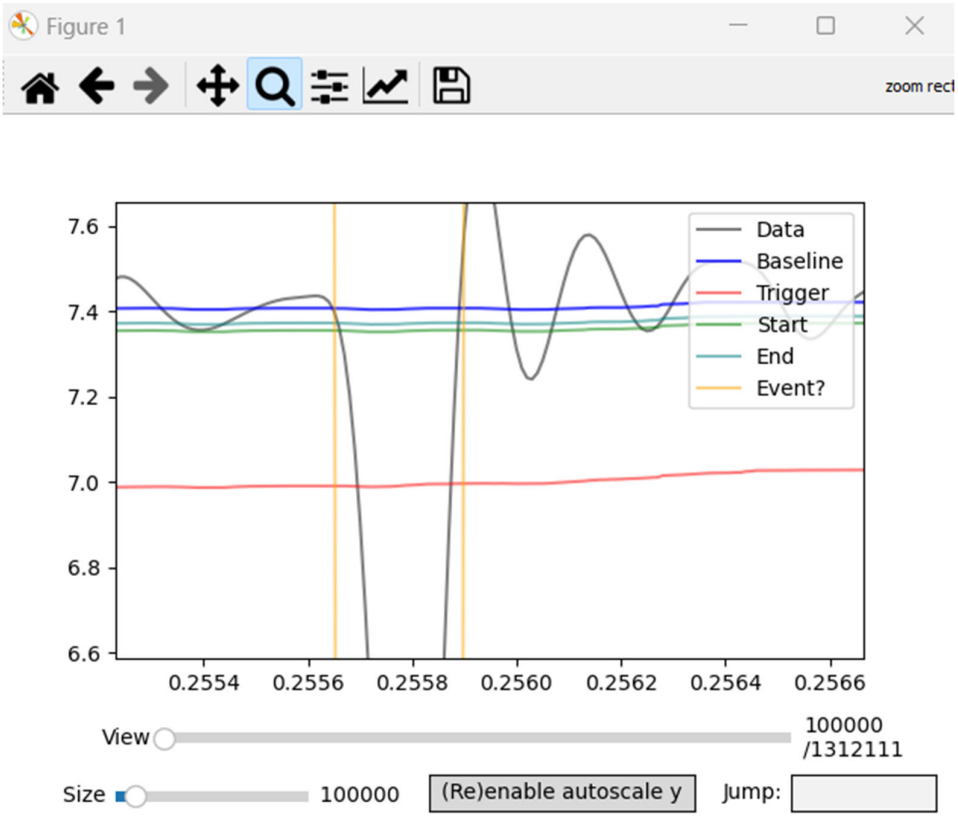
After the trigger line are the start and end lines. The last point before the trigger point that is below the start line is taken as the start of each event and the first point after the trigger point that is below the end line is taken as the end of each event. Typical values are (0.75stds and 0.5stds) or (1stds and 0.75stds) where you see that the start line std is generally larger than that of the end line. From our experience setting the values this way gives the best boundaries, these values also work for most data quite generally and do not require much checking.

Once you have confirmed the end line, event extraction is ran on the 100,000 size preview section with the results shown.



The yellow line is normally high and goes low within events. The height of the yellow line is arbitrary and is only for visualisation. It does not correspond to the event height; in fact event height is not measured/used at all at this stage.

You should check that the majority of events have been successfully captured. You should also zoom in to a few events to check that the boundaries are good, such as as shown in the below example.



Once you are happy, confirm it in the prompt. If you reply N in the prompt you will be brought back to the start of baseline setting.

```
Enter trigger threshold in units of stds [Enter=6.0] >
Confirm trigger threshold? [1=Y,0=N] > 1
Enter start threshold in units of stds [Enter=0.75] >
Confirm start line placement? [1=Y,0=N] > 1
Enter end threshold in units of stds [Enter=0.5] >
Confirm end line placement? [1=Y,0=N] > 1
Happy with results? [1=Y,0=N] > 1
```

After the first confirmation you will be asked to confirm one last time. Confirming here finishes the preparation session for that data and saves its settings to disk. Replying N there brings you all the way to the start of the session for that data.

If an existing settings file exists, you will be prompted on whether you wish to overwrite. In either case, the settings is printed to the console, so if you choose not to overwrite you can simply copy the settings to save it somewhere else of your choosing.

If your data is a set, the settings file will be saved in the same directory that the directory is in, with the file being named the same as the directory. If not, the settings file will be in the same directory as the file, with the file being named the same as the file.

```
Happy with settings? [1=Y,0=N] > 1
{'trace_info': {'sampling_rate': 100000.0,
                'label': 'DNA',
                'signed_voltage': 0.4,
                'pore_id': 'CODETEST1',
                'peaks_not_dips': False,
                'max_event_width_seconds': 0.001,
                'min_event_width_seconds': 2e-05},
 'cleaners': ['lowpassfilter', 'zapmasker'],
 'cleaner_params': {'trimmer': {'slice_start': 100, 'slice_end': None},
                    'zapmasker': {'roll_seconds': 0.1,
                                  'positive_zap_threshold': None,
                                  'negative_zap_threshold': -5.0,
                                  'extra_collateral_damage_rolls': 2},
                    'lowpassfilter': {'fc': 10000.0, 'N': 8}},
 'eventextractor': 'ftrtextractor',
 'eventextractor_params': {'ftrtextractor': {'baseline_window_scale': 3.0,
                                             'std_window_scale': 3.0,
                                             'std_smoothing_scale_factor': 6.0,
                                             'trig_std': 6.0,
                                             'start_std': 0.75,
                                             'end_std': 0.5}}}
Settings file destination "E:\Softwares\Demo\ML-Project\codetest_data\DNA_longest_run.json" exists, overwrite? [1=Y,0=N]
 >
```

That concludes the session for one data. If you have selected multiple targets, the session for the next data will immediately begin.

Processing data after preparation

Once data is prepared, they can be processed using the script run_process.py.

Processing stages

Prepared data is processed into a machine learning usable dataset in multiple stages:

1. Prepared data (data/trace + settings file) → extracted events (trace and baseline within events only, along with original indices for each event)
2. Extracted events → raw signals (only the current data of each event; events are still variable length)
3. Raw signals → standardised signals (raw signals but values are standardized across nanopores to allow valid comparison of signals from different nanoporoes; raw signals are still variable length)
4. Standardised signals → dataset (Fixed length vectors constructed from raw signals following a feature scheme)

A staged process is used so that intermediate files are produced, which can be easily reused, or used for debugging.

Script usage

The script the run_process.py's input varies depending on what stage/file/intermediate file you wish to start from.

The help for the script is shown below.

```
(base) E:\Softwares\Demo\ML-Project>python run_process.py
usage: run_process.py [-h] [-a {data,event,rawsignal,stdsignal}] [-z {event,rawsignal,stdsignal,dataset}] [-s]
                      [-f {abf}] [--standard {root_G0_dG_nS}]
                      [--scheme {geometric_features,geometric_features_plus,averaging_sample_10,averaging_sample_50,aver
aging_sample_10_wslicesize,averaging_10_wslicesize_wgeometricplus,full_res_200,full_res_20000,full_res_200_wgeometricplu
s,full_res_20000_wgeometricplus}]
                      path
```

Below are what each switch is for:

- The -a switch specifies what stage you are starting in. If not specified, it defaults to data. The input file will need to be that corresponding to the stage you chose.
- The -z switch specifies what stage you want to stop at. If not specified, it defaults to dataset. Ending at a stage means you **will** get the file corresponding to that stage.
- The -s switch specifies that you want to scan for input files. The type of files scanned for will depend on the starting stage specified. If you use this switch, the input must be a directory, not a file.
- The -f switch specifies the data format and is same as the -f switch of the data preparation stage. This switch is **required** if starting from the data stage. It is not required and ignored if specified otherwise without warning.
- The --standard switch specifies the standard you wish to use for standardising raw signals. There is only one option – root_G0_dG_ns. If not specified, you will be prompted interactively
- The --scheme switch specifies the feature scheme you wish to use for converting standardised signals to machine learning usable datasets. If not specified, you will be prompted interactively.

If you use the scan option, you will be prompted for each file found if you wish to exclude any of the files.

The output file(s) for each input file will have the same name as the input file (suffix will be different). If the output file for the upcoming stage exists, you will be asked if you wish to overwrite before the start of the stage transition.

The scan option supports sets, just like the data preparation script. Follow instructions for the preparation script for marking a directory as a set. Note however that intermediate files will be produced on a per-file-within-set basis. If the end stage is dataset, the datasets from the files within the set will be combined into a single dataset however.

Notes:

- The run_process.py script is simply a convenience launcher for single-stage, single-file processing scripts run_pipeline.py, events_to_signals.py , signals_to_std_signals.py and make_dataset.py. These component scripts may be used standalone if you so find the need to.
- If errors occur, the launcher will report the non-zero exit code returned by the component script (corresponding to the stage) together with which was the problematic file. Please check the script for the corresponding stage to investigate their meaning. Exit codes may have different meanings in each script, any common meaning is purely coincidental.
- If using scripts standalone, more verbose error messages will be printed to console so you do not need to go through the trouble of echoing the exit code returned.
- A 105 exit code when making event files typically indicates that that data file has no events and is not a true error. If a large majority of your data files show code 105 you may wish to review your extraction settings
- A code 3 when making raw signals from an events file indicates that there was an error loading the events file. A common cause of corrupted events file is if you ran out of disk space during the events extraction process. In this case some events file may not have been completely written. Most archive viewing software should report this issue either as a warning or as an error if you try to open the events file. Always remember to check that you have more than sufficient disk space before doing event extraction to ensure you do not end up having to rerun the time consuming process due to lack of disk space. Events files are generally smaller than their source data file, but if your event density is high, they may only be approximately 1.5x-5x smaller than the source data file, which may still be a large amount of space. For lower event densities, the events file can be more than 30x smaller than the source file.

Machine Learning: Use files “ML_main.py” and “ML_clusters.py” for running machine learning analysis. The details are mentioned in the manuscript.