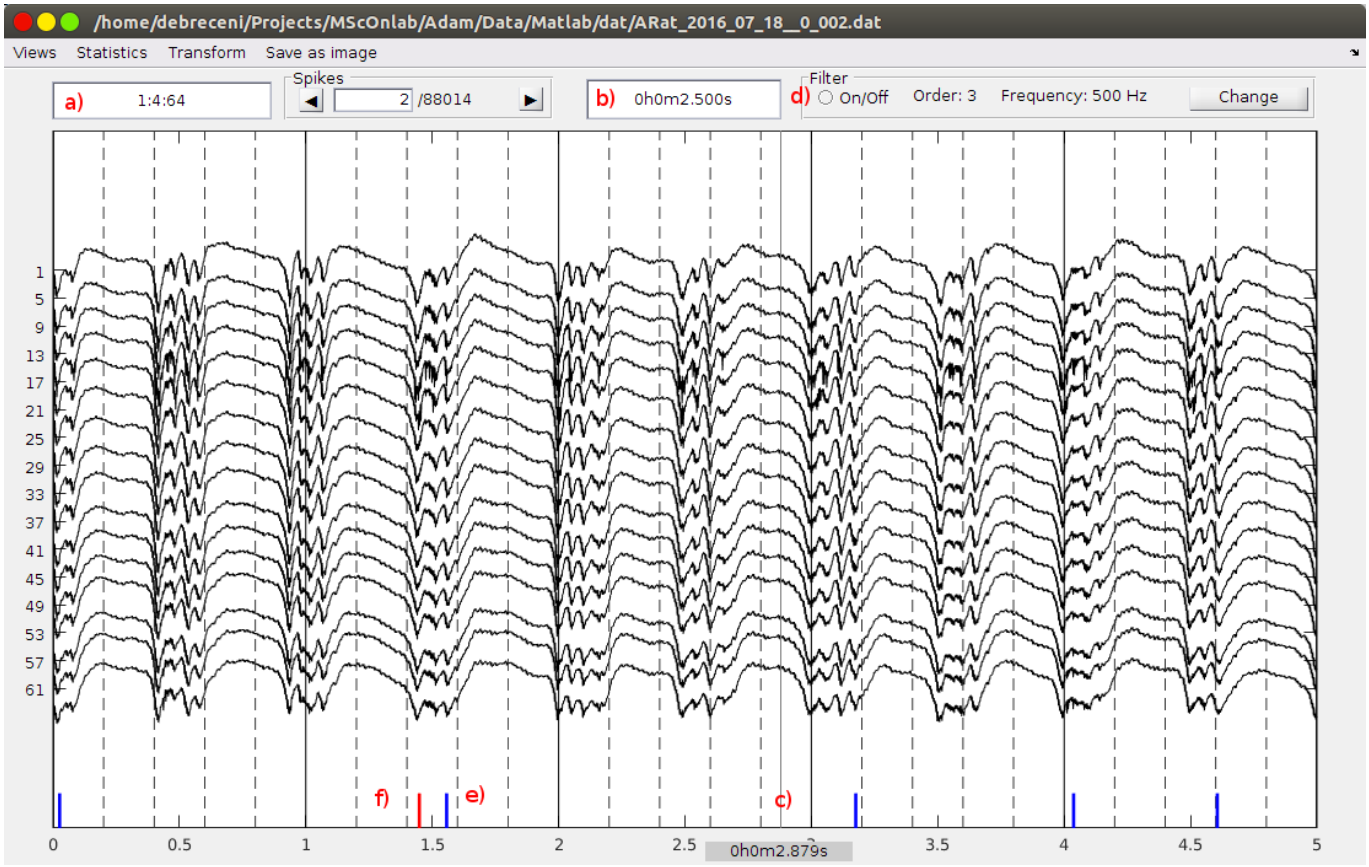
Launching the application

The program can be started with `sdemi` command, then input the number of channels you are going to be working with.

Opening .dat file

Click **Views** > **TraceView**, select the `.dat` file.



Navigating in the .dat file

Vertically: `scroll`, `up/down arrow`

Horizontally: `shift+scroll`, `left/right arrow`

Vertically zoom in/out: `ctrl+scroll`, `ctrl+up/down arrow`

Horizontally zoom in/out: `ctrl+shift+scroll`, `ctrl+left/right arrow`

When you vertically zoom in/out the space between the channels is also adjusted to keep the signals from overlapping.

To zoom in/out without changing this distance use `alt+up/down arrow` or `alt+scroll`.

Selecting channels to display

Enter a Matlab style list expression into `(a)` e.g.:

`1,2,3`, `16:32`, `1:3,12:4:28`

(Currently, channels are only displayed in order i.e. `4:-1:1` is invalid)

Jump to time

To jump to a specific time in the recording enter the time into `(b)`.
The format it accepts is `[<hour>h] [<minute>m] [<second>[.<millisecond>]s]`
e.g.: `2m`, `54m 23s`, `1h 5s`, `128s` (which is translated to `2m 8s`), `0.25s`
If the unit is omitted second is assumed
The millisecond component has a maximum precision of 1ms meaning `5.2135s` is invalid.

Filtering

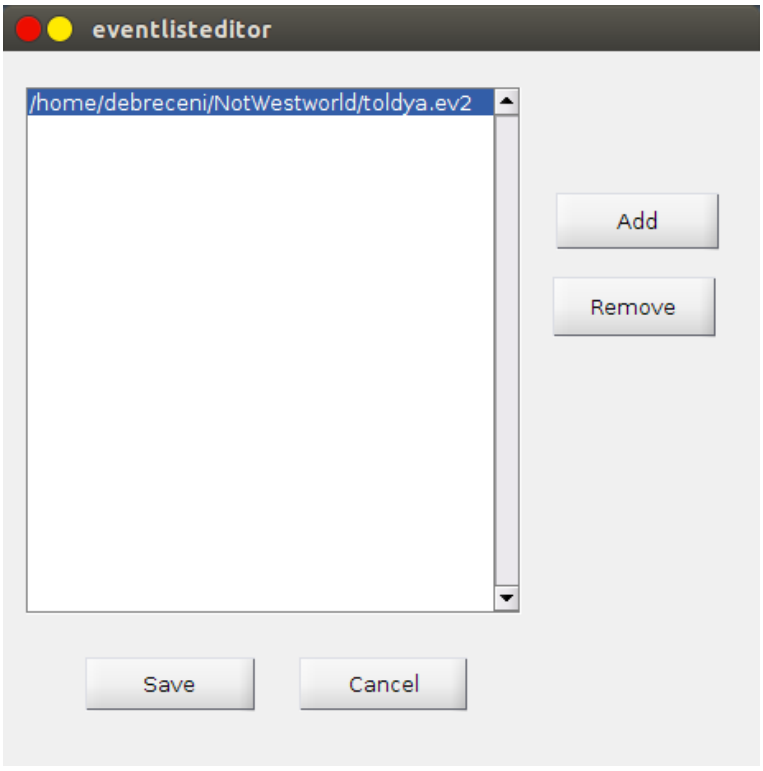
Simply toggle `(d)` on/off, it uses a high-pass Butterworth filter for which the cut-off frequency and the order is adjustable.

Exporting

To generate images of the currently visible region of the signal click `Save as image`. A temporary window will appear shortly and disappear. If it persists that is a bug.

Adding event files

To display stimulation events along the recording you can choose **EventFiles** > **Edit**.
For every visible event file's every event there is an indicator (a little colored line at the bottom) like (`(e)`,`(f)`).



Displaying the event files

All the event files available can be adjusted in **EventFiles** > **Appearance**. You can select which event file should be **active** from the drop-down menu, and select which event files should be visible and with what color should the indicators be drawn (`Visibility` panel).

Navigating the active event file

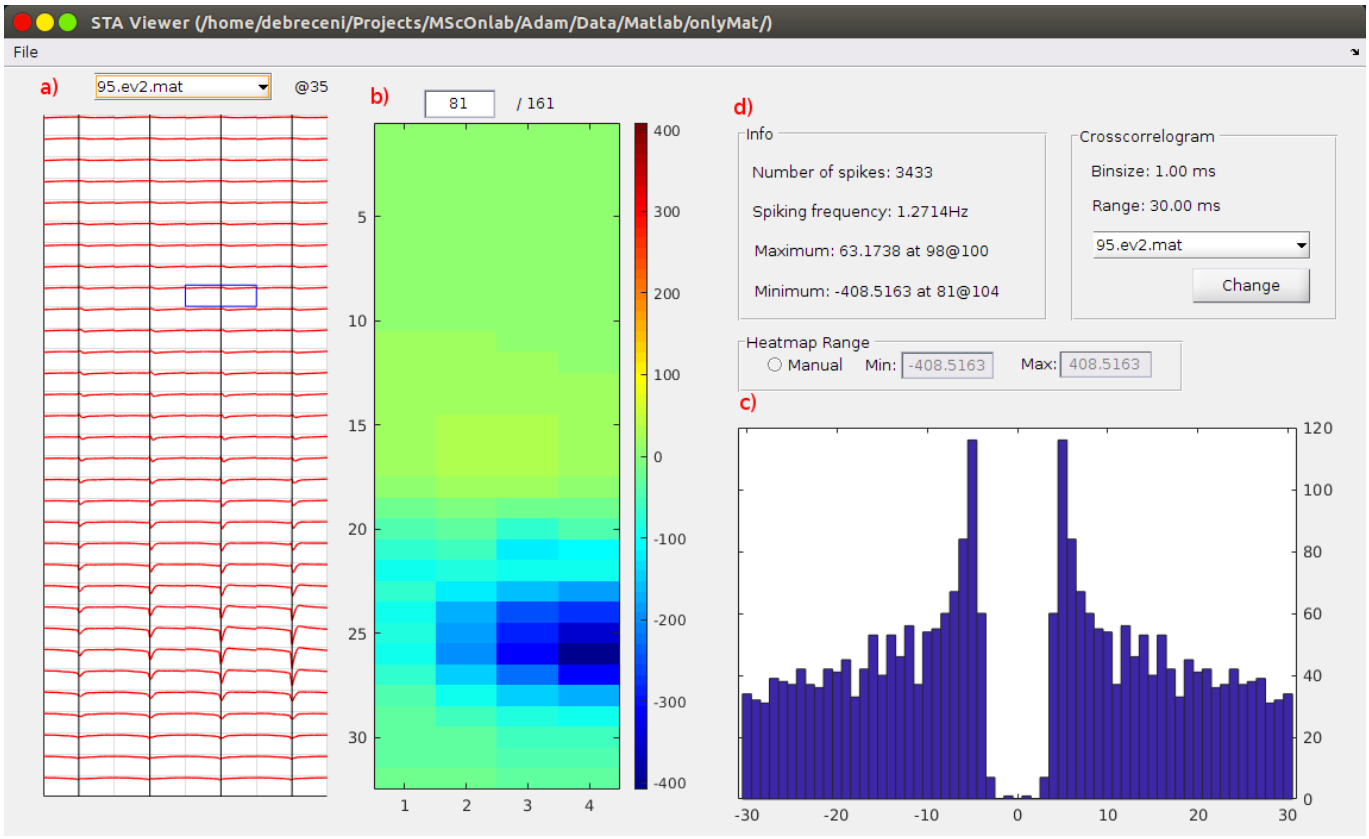
There is at most one event file marked **active**. The current event and the total number of events is displayed in the `Spikes` panel. You can either click on the arrows or enter a number into the textfield to jump to that event.
You can also click on any indicators to center on that.

Generating STA files

Click **Transform** > **Generate STA from...**, then select the appropriate files and directories.

Opening STA files

Click **Views** > **WaveformView**, enter the number of columns of the electrode, select the directory containing the STA files. In case the directory contains the original event files as well, they are loaded automatically and can be displayed on the `TraceView`.



Selecting the cluster

Use the drop-down selector `(a)` to switch between clusters or `up/down arrows`.

Navigating in the heatmap

Hover over the WaveformPanel (panel under `(a)`) and `scroll` or press `left/right arrow`.
You can also click in the WaveformPanel and jump to that point.
Alternatively enter a number in range into `(b)`.

Adjusting heatmap colors

By default for every cluster the range is determined as the `[-max(cluster), max(cluster)]`.
You can enter custom values by toggling `Manual` on the `Heatmap Range` panel.

Calculating cross/auto-correlogram

By default an autocorrelogram is calculated, from the drop-down menu on the `Crosscorrelogram` panel you can pick a different cluster for comparison.
The `Binsize` and the `Range` can be adjusted by clicking on `Change`.

Exporting STAs

By right clicking on any of the three panels (the waveforms, the heatmap or the correlogram) you can save a picture in the specified file format. A temporary window will appear shortly and then disappear. If the window persists that is an unfortunate bug.

Visualizing signal as heatmap Views > HeatmapView will open a window where the same region (as on `TraceView`) is displayed, the local field potential, the current source density (disabled for now) and the multi unit activity. The range of each heatmap is adjustable.

Visualizing an event Views > ClusterView opens a window displaying a detrended slice of the `.dat` file centered around the selected event. Only works with a layout of 4x32. Can use to reassign or discard events.

Calculating Features

Select **Statistics** menu on the `TraceView`. Specify the input `.mat` files containing the STAs, specify the output directory. `Add` or `Remove` features you want to calculate. Some features require a parameter, to change the defaults double-click on the Feature in the list. Click `Calculate`.

Visualizing Features

To inspect the features open **Views > StatisticsView** on the `TraceView`. Click `Open` and select a `summary*` or `zscore_summary*` file. You can select which feature should be assigned to which axis, you can click `Rotate/Datacursor` to rotate the plot.

Clustering Units

Click on **Cluster > Hierarchical/K-means** on `TraceView` to bring up the clustering window. Specify the input feature file(`summary*` or `zscore_summary*`), the output file's name, and select/unselect those features which should be included in the clustering. Click `Calculate`, if it is finished `(Done)` is presented in the window's header.

Adding new features

Add a new file to the project preferably with name `"calc*"` (e.g. `calcFiringRate`, although it is not a requirement).

In `statView.m` add a new line to the `handles.features` with the following format:

`<feature_name>, struct('callback', @<new_function_name>, 'args',{{<param_name>; <default value>}})`

if there are no parameters, set the `args` to `[]`

```
handles.features = {'Firing rate',struct('callback',@calcFiringRate,'args',[]);...
            'Burstiness',struct('callback',@calcBurstiness,'args',{{'Threshold (ms)';5}});...
            'Spike duration',struct('callback',@calcSpikeDuration,'args',[]);...
            'Maximal spike amplitude',struct('callback',@calcMaxAmplitude,'args',[]);...
            'P1 amplitude',struct('callback',@calcP1Amplitude,'args',[]);...
            'P2 amplitude',struct('callback',@calcP2Amplitude,'args',[]);...
            'N amplitude',struct('callback',@calcNAmplitude,'args',[]);...
            'P1-P2 ratio',struct('callback',@calcP1overP2,'args',[]);...
            'Spike asymmetry',struct('callback',@calcAsymmetry,'args',[]);...
            'P1-N duration',struct('callback',@calcP1N,'args',[]);...
```