

CureQ MEA Analysis tool User Guide

01/02/2024

K. J. van Beem – k.j.van.beem@hva.nl

Contents

Installation and setup	2
Official website	2
Microsoft store	3
Installing the library	4
Launch graphical user interface	6
Set parameters.....	9
Analyse single file	9
Analysis output.....	11
Batch processing	12
Inspect results.....	14
Single Electrode View.....	15
Whole Well View.....	17
Compare groups	18
Features over time	20
Boxplots	21
Parameters	23
Convert Axion .raw to hdf5	26
Compress/Rechunk files.....	27
Features	28
Python Library	31

Installation and setup

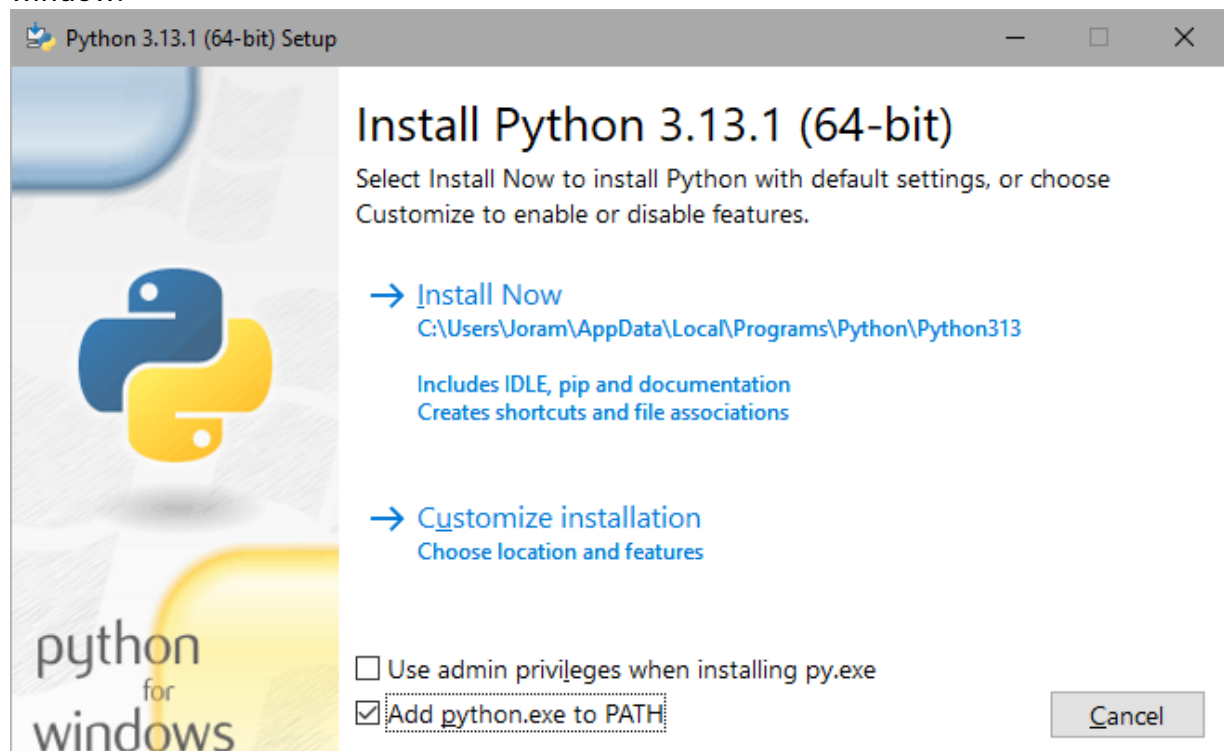
Since this tool is written in python, python is also required to run the application. The first step to installing the MEA analysis tool is installing python. There are two ways to install python.

Official website

Firstly, python can be installed using the official python installer. All python versions can be found on <https://www.python.org/downloads/>. The MEA analysis tool has been developed and tested on python 3.11, but will most likely work on newer versions. Download the desired python version.



Next, execute the file you have just downloaded, you will be presented with the following window:



Select the option “**Add python.exe to PATH**”. This will allow your operating system to recognise the python installation. Then, press “Install Now”. Finally, let’s confirm that both python and pip are successfully installed by running the following commands in the command prompt.

```
C:\>python --version  
Python 3.11.9
```

```
C:\>pip --version  
pip 23.3.1
```

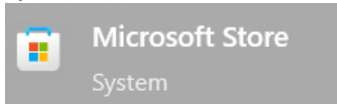
If both outputs look like this (can be different versions) both python and pip have been successfully installed.

Microsoft store

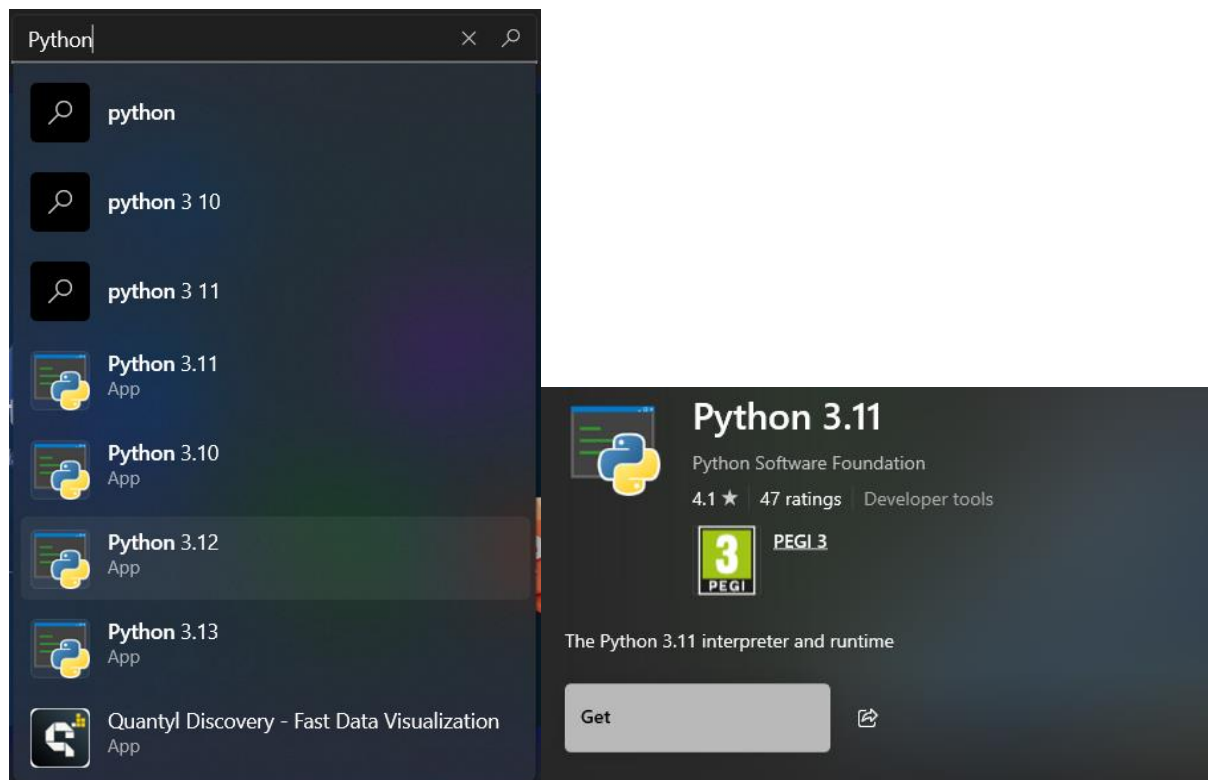
The following text originates from <https://learn.microsoft.com/en-us/windows/python/beginners> and has been slightly altered.

To install Python using the Microsoft Store:

Go to your Start menu (lower left Windows icon), type "Microsoft Store", select the link to open the store.



Once the store is open, select Search from the upper-right menu and enter "Python". Select which version of Python you would like to use from the results under Apps. The MEA analysis tool has been developed and tested using python 3.11, however, newer python versions should work as well. Once you've determined which version you would like to install, select Get.



Once Python has completed the downloading and installation process, open Windows Command Prompt using the Start menu (lower left Windows icon). Once Command Prompt is open, enter `Python --version` to confirm that Python3 has installed on your machine. The output should look like this:

```
C:\>python --version
Python 3.11.9
```

The Microsoft Store installation of Python includes pip, the standard package manager. Pip allows you to install and manage additional packages, such as the MEA analysis tool, that are not part of the Python standard library. To confirm that you also have pip available to install and manage packages, enter `pip --version`. The output should look like this:

```
C:\>pip --version
pip 23.3.1
```

If both outputs look like this (can be different versions) both python and pip have been successfully installed.

Installing the library

Once both python and pip have been installed, acquiring the MEA analysis tool should be simple. For the next step we will use pip to install the tool.

First, open the command prompt, then enter “pip install cureq”. Pip will now collect the cureq package from the internet, collect and install dependencies and the library. The last few lines should look like this:

```
Installing collected packages: cureq
Successfully installed cureq-1.1.1
```

The MEA analysis tool is now successfully installed on your machine and can be executed from a python script.

Add to path

When downloading the MEA analysis tool, the following warning might appear:

```
Using cached CureQ-1.2.5-py3-none-any.whl (140 kB)
Installing collected packages: cureq
  WARNING: The script cureq.exe is installed in 'C:\Users\jvbga\AppData\Local\Packages\PythonSoftwareFoundation.
Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-locati
on.
Successfully installed cureq-1.2.5
```

This usually occurs when python is installed using the Microsoft store. It means that the directory containing python scripts is not added to the PATH environment variable yet. This will mean that the MEA analysis library commands will not be able to be executed from the command line. To fix this, the file path will have to be manually added to the environment variables. This can be done using one of the following guides:

- <https://stackoverflow.com/questions/44272416/how-to-add-a-folder-to-path-environment-variable-in-windows-10-with-screensho>
- <https://www.eukhost.com/kb/how-to-add-to-the-path-on-windows-10-and-windows-11/>

In this example, the folder that should be added to PATH is:

```
C:\Users\jvbga\AppData\Local\Packages\PythonSoftwareFoundation
.Python.3.12_qbz5n2kfra8p0\LocalCahce\local-
packages\Python312\Scripts
```

Upgrade Library

The MEA Analysis Tool might receive further updates to enhance the analysis, or fix problems. To check the current version of the library, open the command prompt, and enter: “cureq –version”.

```
C:\> cureq --version
CureQ MEA analysis tool - Version: 1.2.7
```

The most recent available version of the library can be found on the [pypi page](#). The library can be upgraded using the command “pip install cureq –upgrade”. This will upgrade the library to the newest available version.

```
C:\> pip install cureq --upgrade
Successfully installed cureq-1.2.7
```

The first and last row of the output should look something like this.

Launch graphical user interface

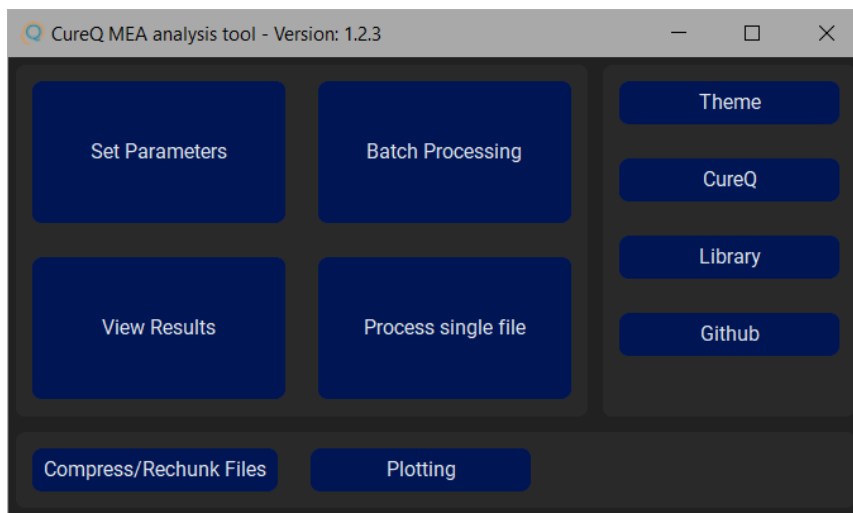
The graphical user interface is the simplest way to communicate with the library. The GUI can be launched in multiple ways:

Launch from command prompt

Firstly, the GUI can be launched from the command prompt. Simply open the command prompt, and enter “cureq”.

```
C:\Users>cureq  
Successfully launched MEA GUI
```

The output should look like this, and the GUI should appear on your screen.

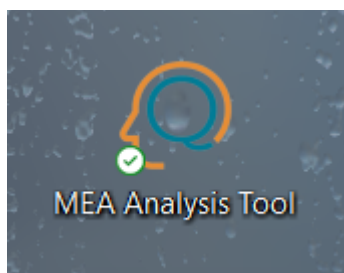


Create shortcuts

This process can be simplified by creating shortcuts that in essence perform the same process. In the command prompt, enter “cureq --create-shortcut”.

```
C:\Users>cureq --create-shortcut  
Desktop shortcut created at C:\Users\Desktop\CureQ.lnk
```

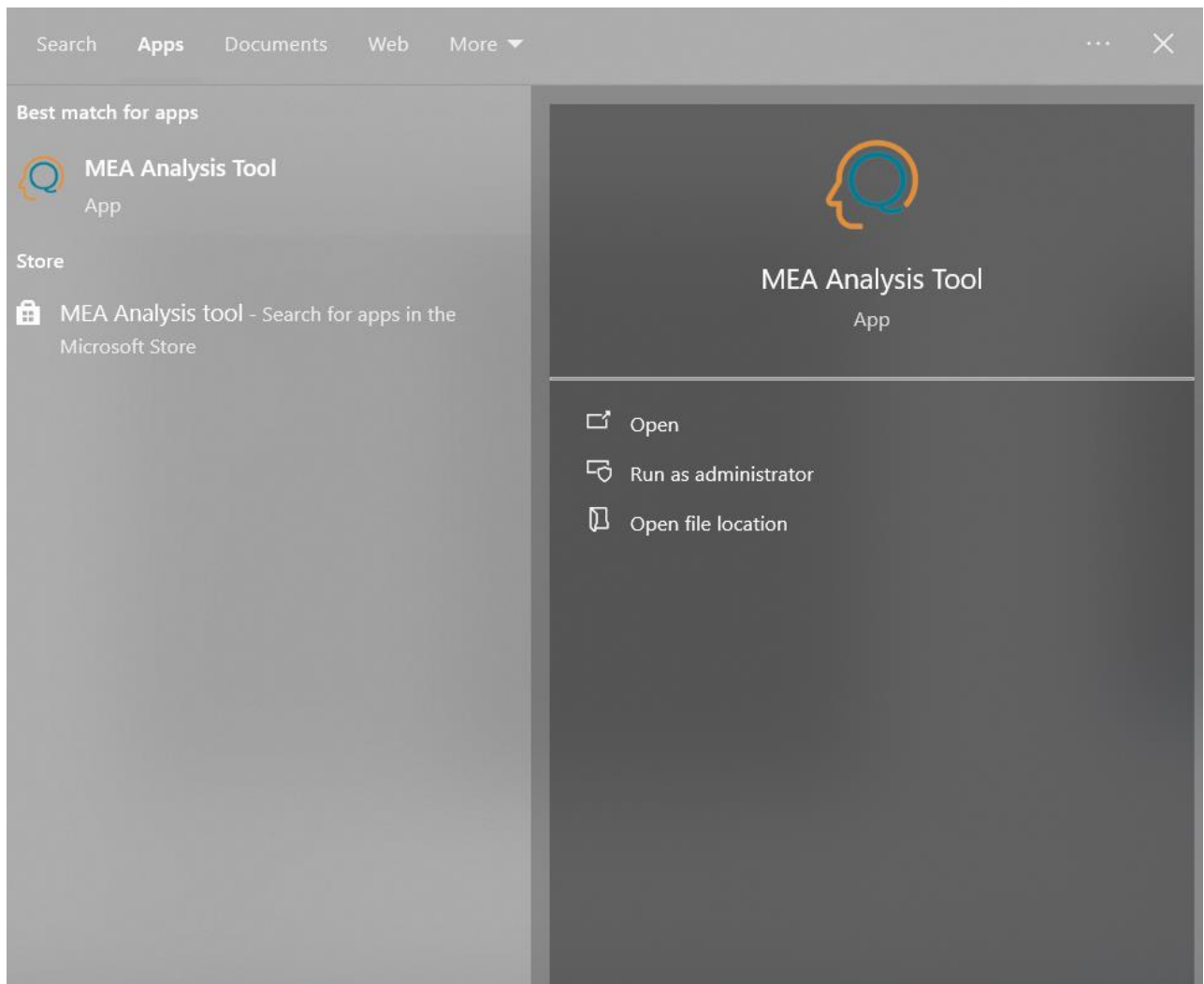
The output should look like this, and a shortcut should appear on your desktop:



Additionally a shortcut can be added to the start menu. In the command prompt, enter “cureq –add-to-start-menu”.

```
C:\Users>cureq --add-to-start-menu  
  
Start Menu shortcut created at C:\Users  
\AppData\Roaming\Microsoft\Windows\Start  
Menu\Programs\CureQ\CureQ.lnk
```

The output should look like this, and the shortcut should appear in your start menu.



The shortcut can also be added to the taskbar by pressing “Pin to taskbar”.

Launch from python script

Lastly, the GUI can be launched from a python script. Create a python file and execute the following code:

```
from CureQ.meanalysis_tool import MEA_GUI
```

```
if __name__ == '__main__':  
    MEA_GUI()
```

The GUI should always be opened inside the “if __name__ == '__main__'” guard. Otherwise, when using multiprocessing, the application will slowly create an infinite amount of processes and crash the application.

Set parameters

First, navigate to the “Set Parameters” tab to alter analysis parameters.

The screenshot displays the 'CureQ MEA analysis tool - Version: 1.2.3' window. It features three main panels for parameter configuration:

- Filter Parameters:** Includes input fields for 'Low cutoff:' (200), 'High cutoff:' (3500), and 'Filter order:' (2).
- Other Parameters:** Contains a 'Use multiprocessing:' checkbox (unchecked), a 'Remove inactive electrodes:' checkbox (checked), and an 'Activity threshold:' input field (0.1).
- Spike Detection Parameters:** Includes a 'Threshold Parameters' sub-panel with 'Threshold portion:' (0.1), 'Standard Deviation Multiplier:' (5), and 'RMS Multiplier:' (5). Below this is a 'Refractory Period:' input field (0.001), a 'Spike validation method:' dropdown menu (set to 'DMP_noisebased'), an 'Exit time:' input field (0.001), a 'Drop amplitude:' input field (5), and a 'Max drop:' input field (2).
- Burst Detection Parameters:** Includes 'Minimal amount of spikes:' (5), 'Default interval threshold:' (100), 'Max interval threshold:' (1000), and 'KDE bandwidth:' (1).
- Network Burst Detection Parameters:** Includes 'Min channels:' (0.5), a 'Thresholding method:' dropdown menu (set to 'Yen'), and 'KDE Bandwidth:' (0.05).

At the bottom of the interface are three buttons: 'Save parameters and return', 'Restore default parameters', and 'Import parameters'.

From here it is possible to alter the parameters or import settings from a previous experiment. To save the parameters, press **Save parameters and return**.

Restore default parameters can be used to restore all parameters to the default values.

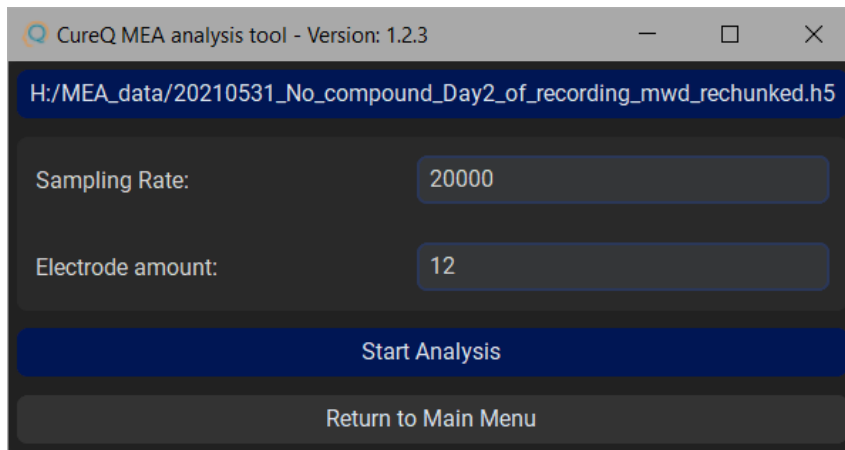
Parameters from previous experiments can be loaded using **Import parameters**. Every experiment will generate a file called “parameters.json” in the output folder. These files can be selected to copy the parameters to the current analysis.

For more information about each parameter, see [Parameters](#).

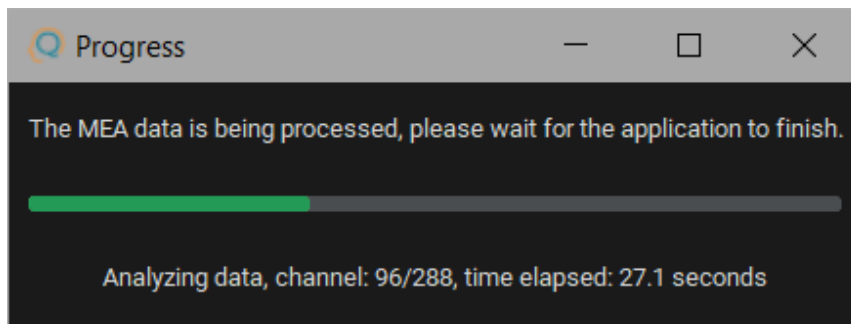
Analyse single file

In the main menu, select **Process single file**.

First, press **Select a file** and select the file you want to process. Next, insert the sampling rate of the experiment, and the amount of electrode that each well contains. Then, press **Start Analysis**.



A new window will appear, keeping the user up to date with the progress. The analysis can be cancelled by closing this window. This might not be instant.



Axion .raw files

Axion .raw files can unfortunately not be accessed by python, so they will first have to be converted using MATLAB using our custom script available on GitHub. For more information, see [Convert Axion .raw to hdf5](#).

Rechunking

If your file has not been rechunked/compressed yet, the application will first rechunk the file, creating a rechunked copy, and then process the rechunked file. Files can also be rechunked manually. For more information about why files are rechunked, see [Compress/Rechunk files](#).

Analysis output

During the analysis, the application will create an “outputfolder” where it stores all results and details from the analysis. This outputfolder can be found in the same location as the raw data file. The outputfolder will share the same name as the raw data file, with the data and time of the analysis attached at the end. The outputfolder contains several different folder and files:

Name	Date modified	Type	Size
burst_values	23/01/2025 10:49	File folder	
figures	23/01/2025 10:49	File folder	
network_data	23/01/2025 10:49	File folder	
spike_values	23/01/2025 10:49	File folder	
20210531_No_compound_Day2_of_recording_mwd_rechunked_output_2025-01-23_10-49-08_Features.csv	23/01/2025 10:50	Microsoft Excel Co...	9 KB
parameters.json	23/01/2025 10:49	JSON File	1 KB

Burst_values, **network_data** and **spike_values** contain files that are used to save the results of the spike, burst and network burst detection process. These files are used to calculate the output features and visualise the process in the GUI. These folders are generally not important for the end user.

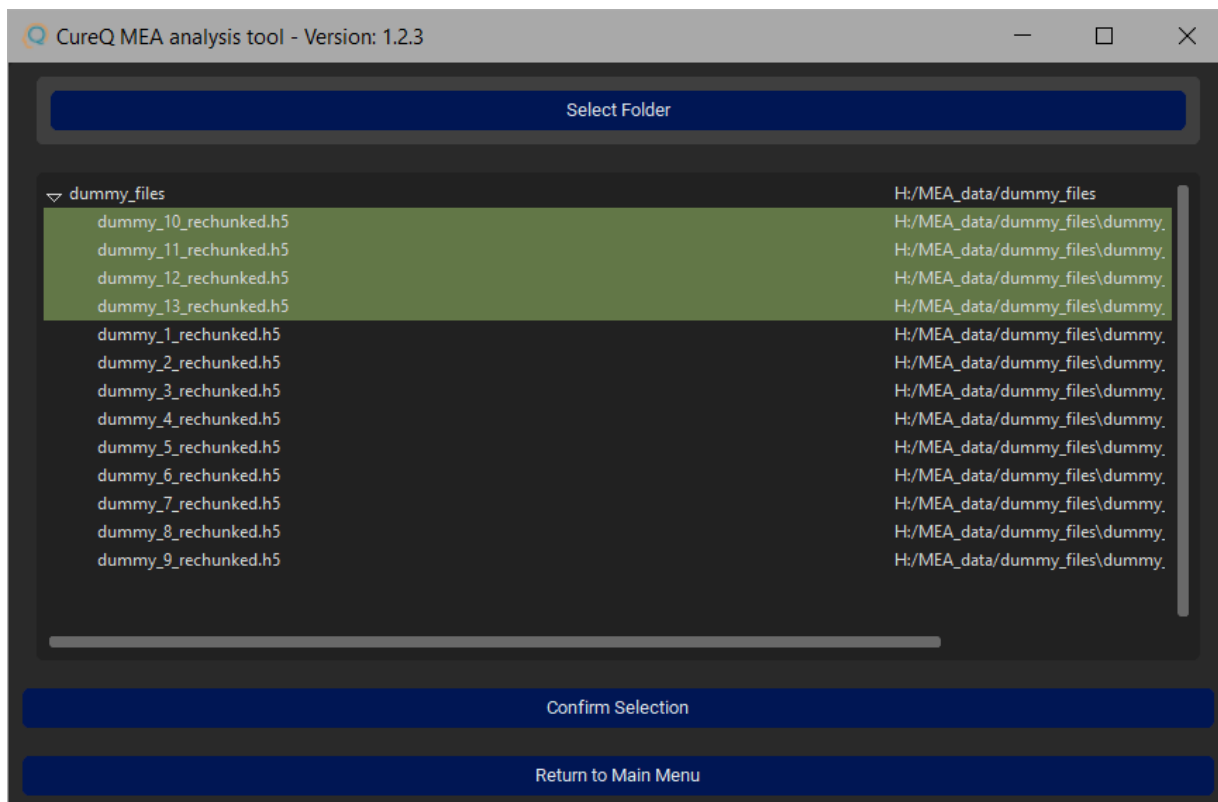
Figures contains visualisations for the network burst detection process, and can be used to get a quick overview of network/well activity without having to load in the results in the GUI.

The file ending with **Features.csv** will contain the output features that are calculated by the analysis tool. Here, every column is a feature, and every row is a well. For more information about the features the MEA analysis tool calculates, see [Features](#).

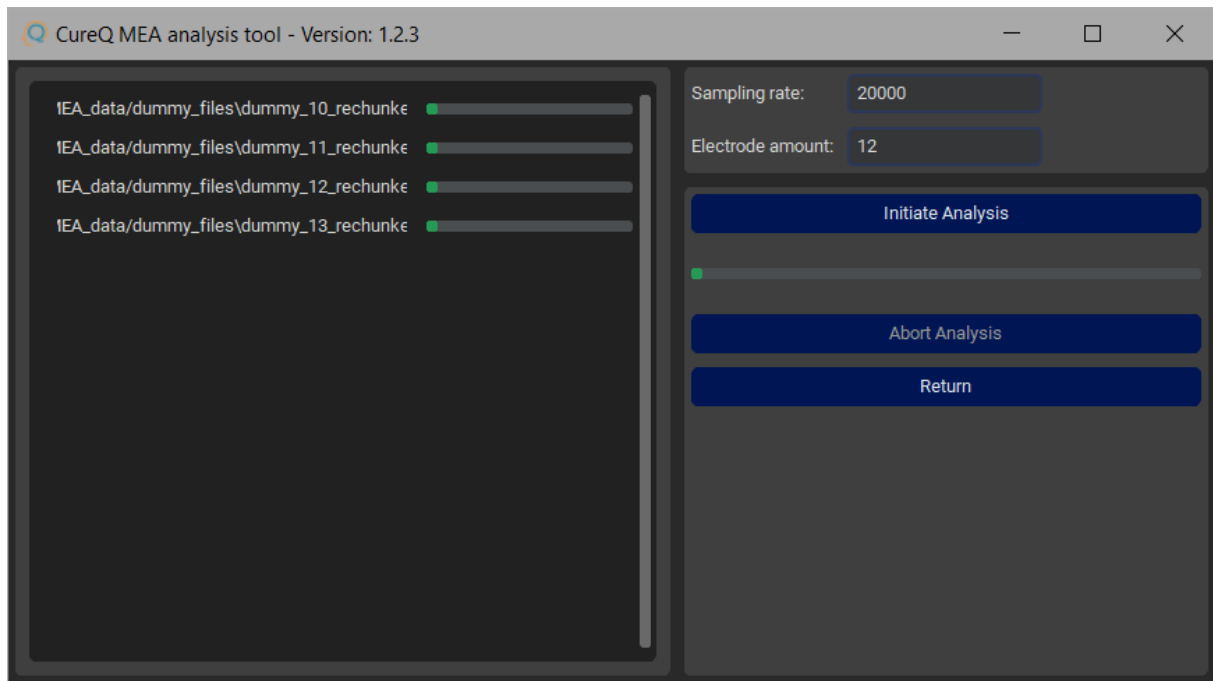
Lastly, the **parameters.json** file contains information about the analysis such as which parameters were used, some information about the dataset, and which version of the tool was used.

Batch processing

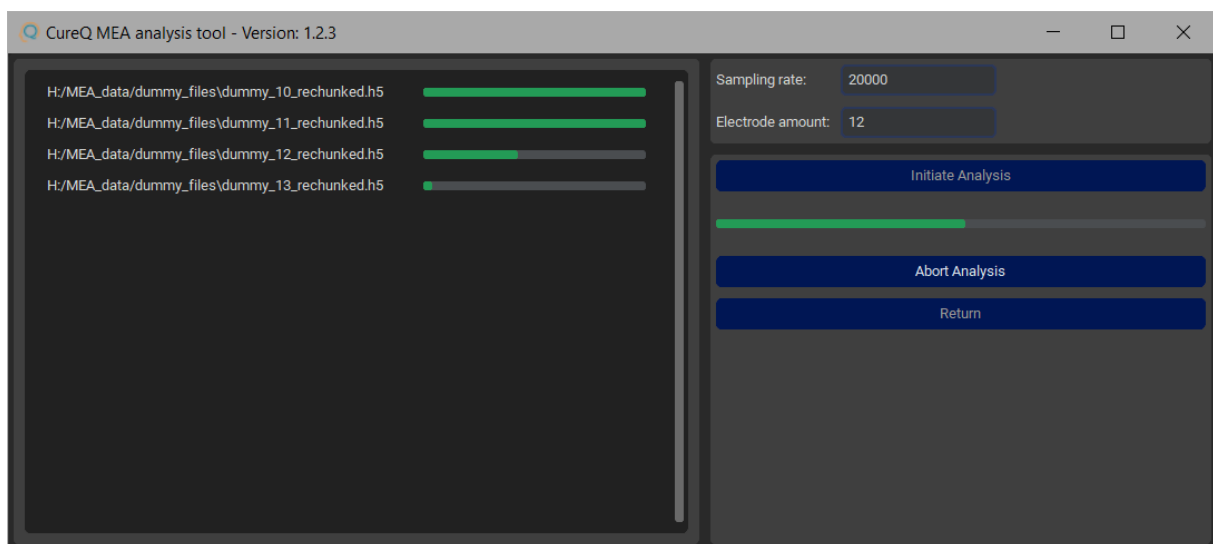
In the main menu, select **Batch Processing**. Next, press **Select Folder** and select the folder where your MEA files are located.



In this menu, the user can create a selection for the batch processing. It is possible to expand subfolder. Single files can be selected by clicking on them, and the content of the entire folder can be selected by selecting the folder name. Once the selection has been made, press **Confirm Selection**.



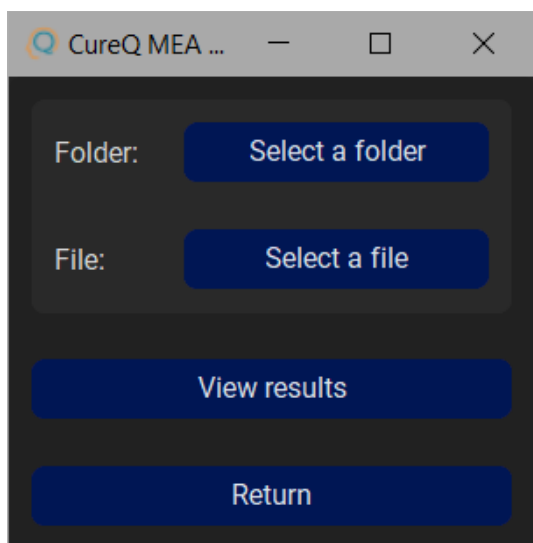
Next, insert the sampling rate of the experiment, and the amount of electrode that each well contains. Then, press **Initiate Analysis**.



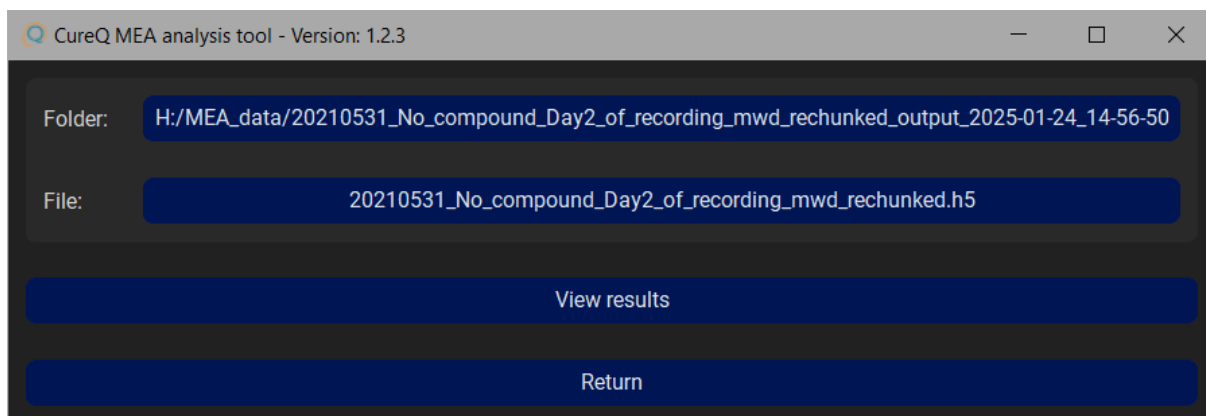
It is possible to cancel the analysis by pressing **Abort Analysis**. The application will finish processing the current file, after which it will quit the analysis.

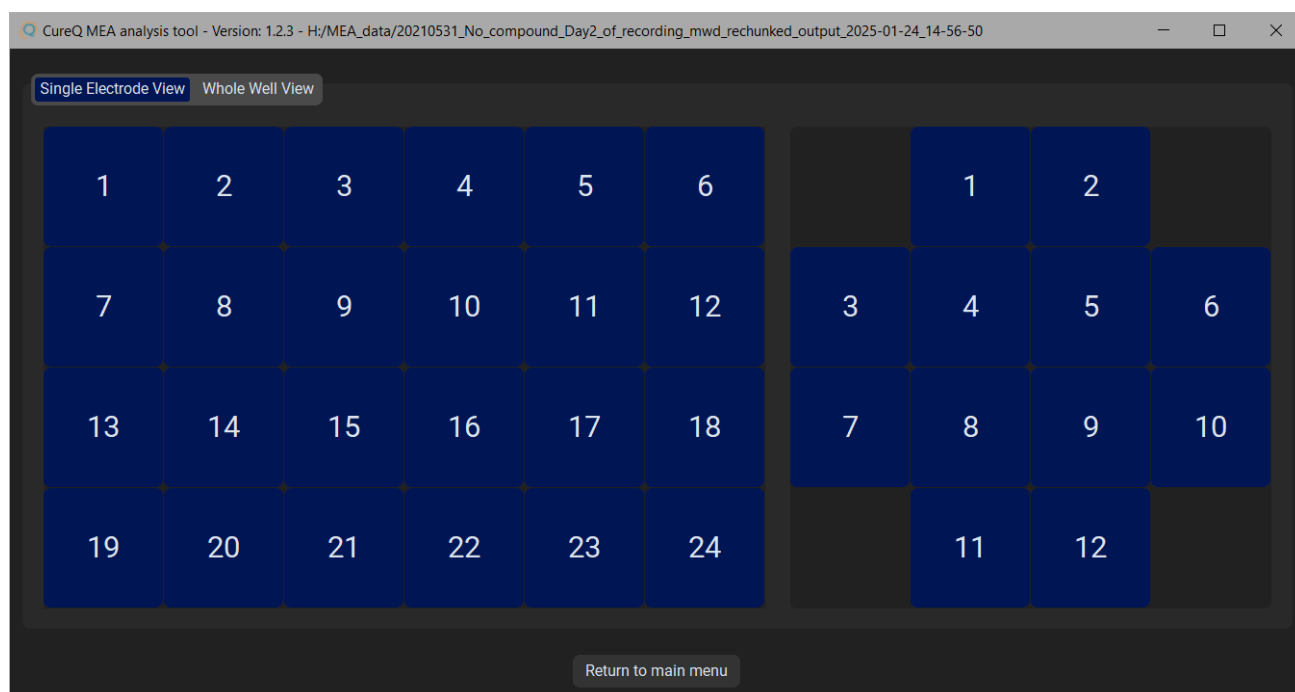
Inspect results

The CureQ MEA Analysis tool allows the user to inspect the results after the analysis has been completed. In the main menu, select **View Results**.



Here, select the output folder that was generated by the analysis. Next, select the raw data file that was used for the experiment. Then press **View Results**.

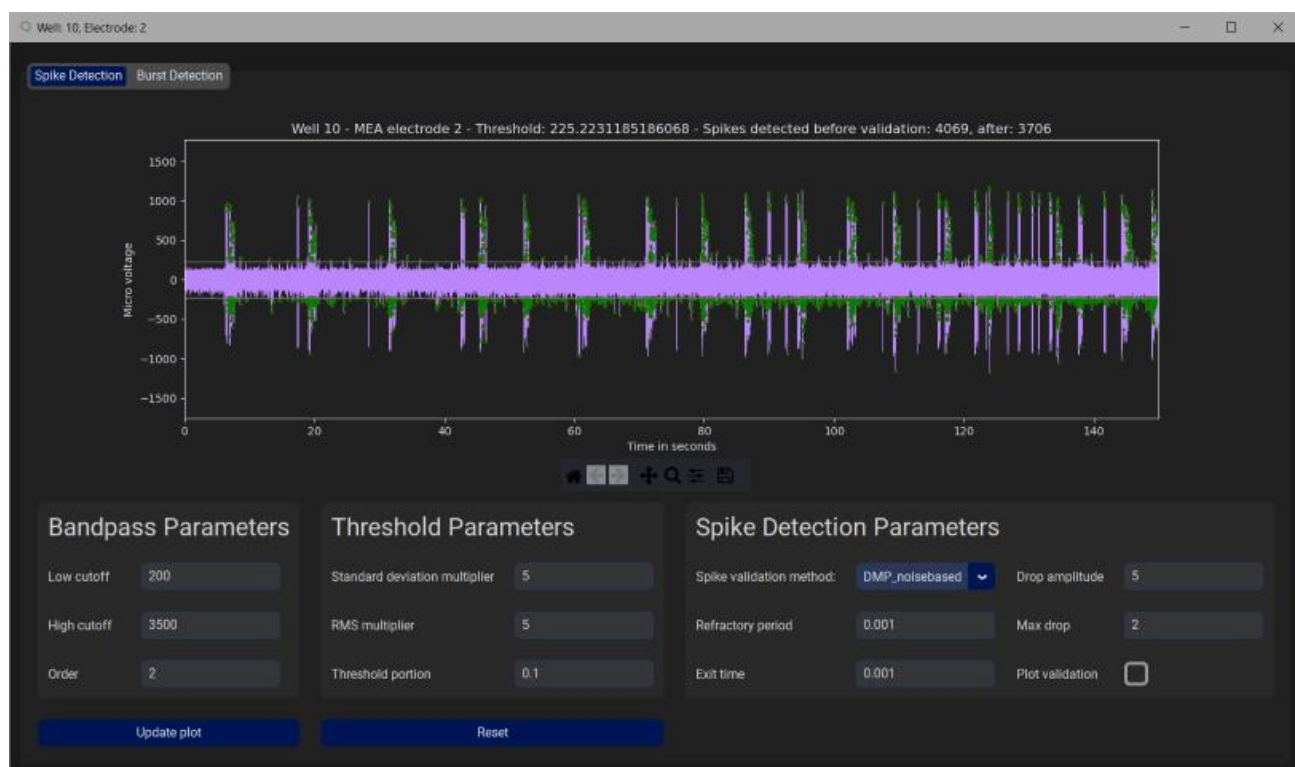




This window contains two tabs, **Single Electrode View** and **Whole Well View**.

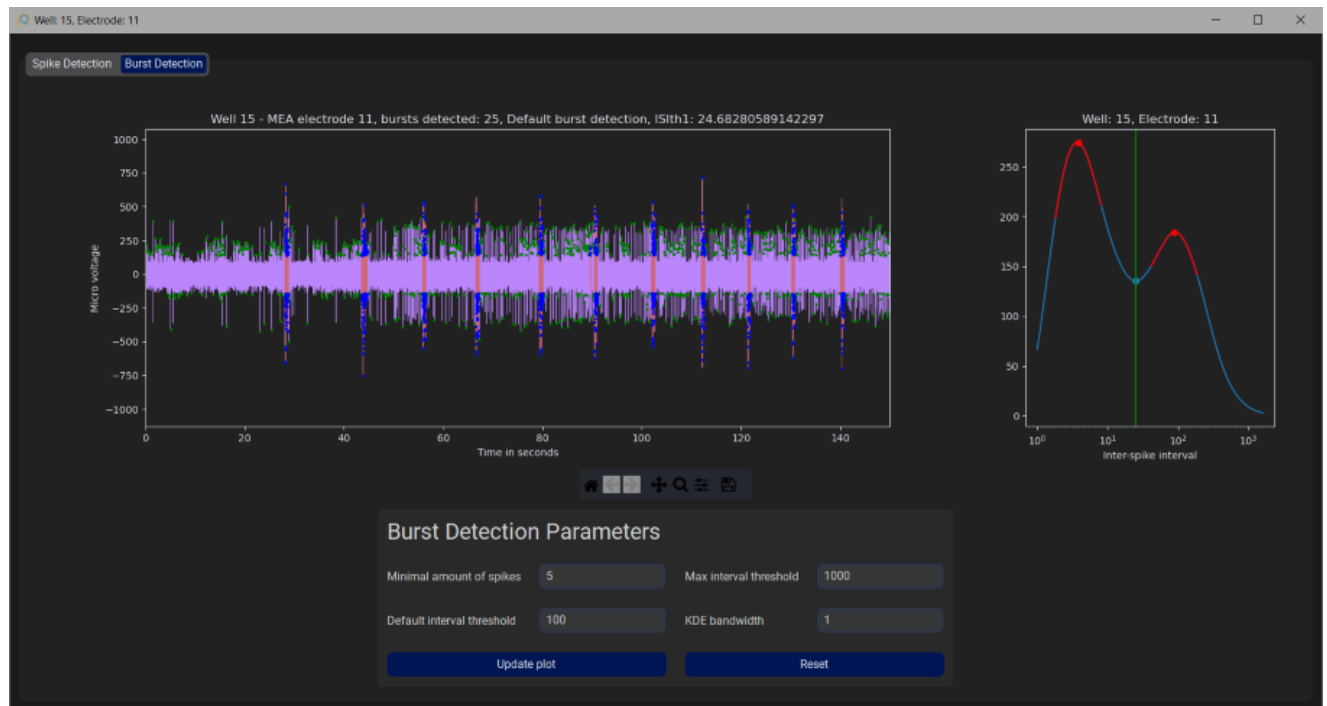
Single Electrode View

From the **Single Electrode View** tab, the user can select the desired well and electrode, after which a new window will open.



Again, this window contains two tabs, **Spike Detection** and **Burst Detection**. The first tab displays the spike detection applied to the raw electrode data. It also contains all

the parameters that influence the spike detection. The user can alter these parameters to see how they could affect the analysis, although they will not have any effect on the outcomes of the current experiment, or further steps, such as burst or network burst detection.



In the second tab, the burst detection is applied to the electrode data, as well as the kernel density estimate that was used to determine the burst detection parameters. Again, the user can change certain parameters to see how they could affect the burst detection process without influencing the outcome of the current experiment.

The raw data plot on both tabs is interactive and allows the user to zoom in on the data using the toolbar.



Whole Well View



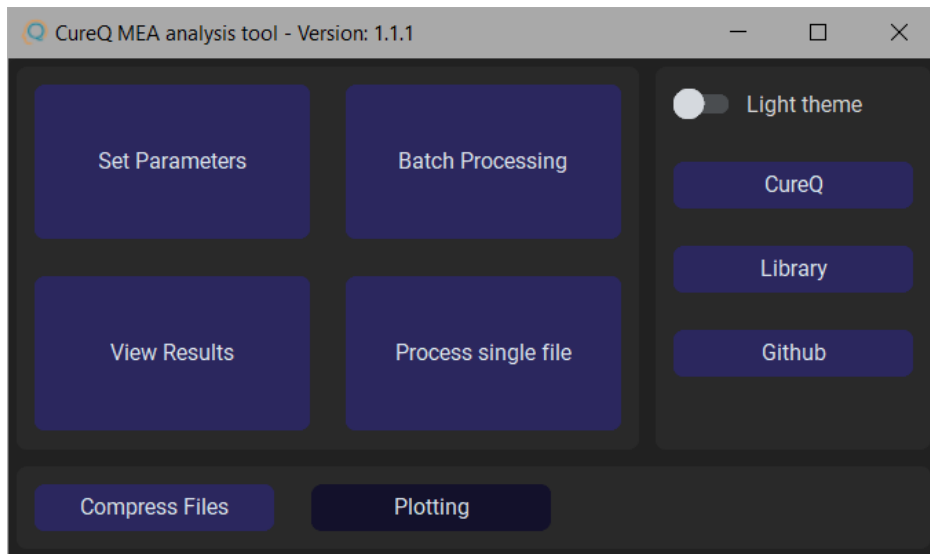
From the Whole Well View tab, the user can select the desired well, after which a new window will open.



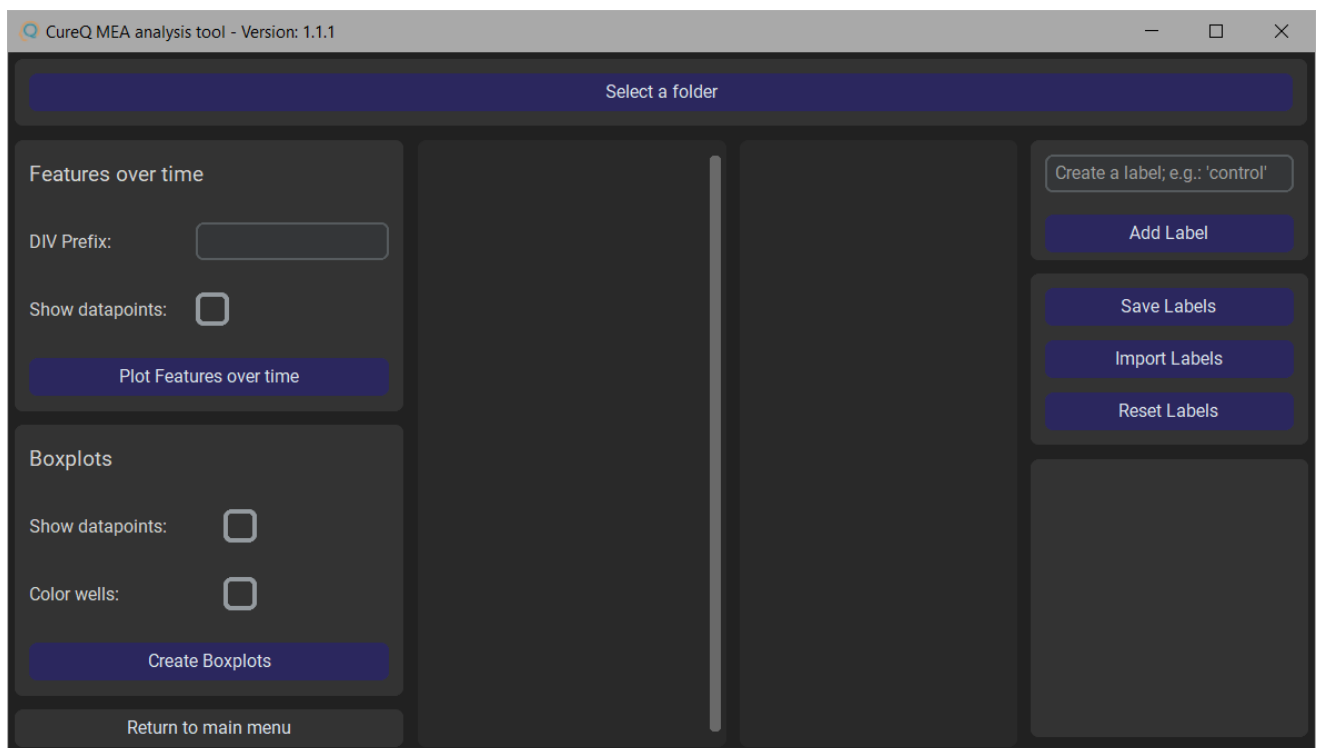
Here, the user can inspect the network burst detection process on this particular well, and alter parameters to see the effect on the analysis. Again, this will not have any effect on the outcomes of the current experiment.

Compare groups

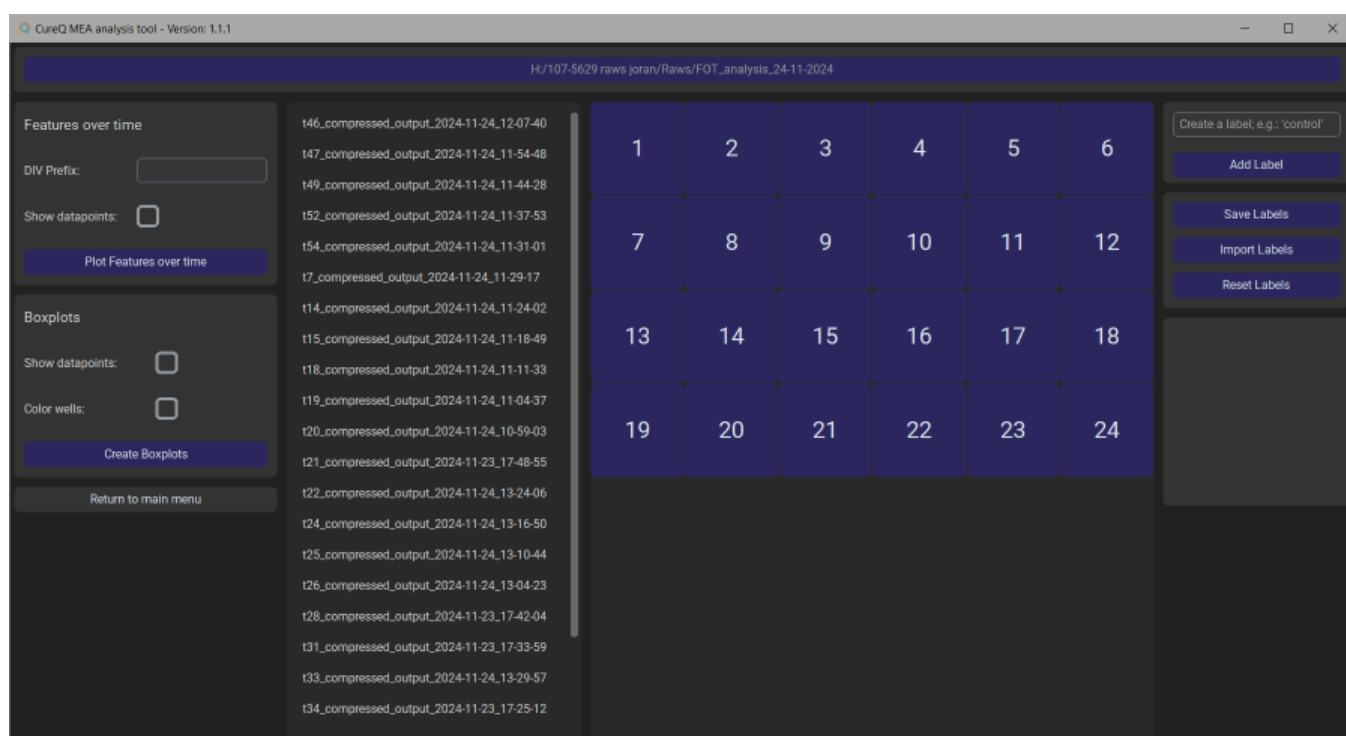
Besides functionality for analysing raw data files, this tool also provides methods to compare the outcomes of the analysis with each other.



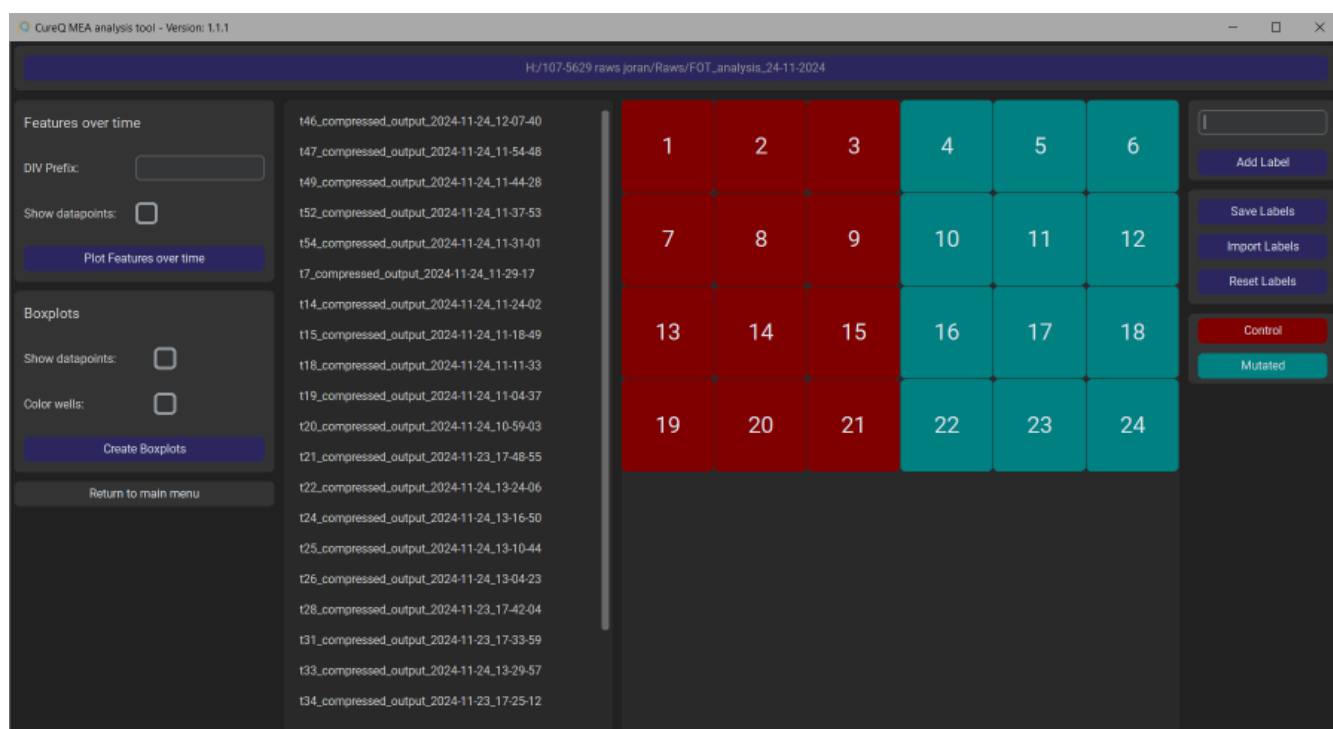
In the main menu, navigate to **Plotting**.



At the top of the window, press Select a folder, and select the folder that contains the outputfolder(s) of your experiment(s). The application will recursively loop through all the files/folder in the directory, and look for files that end with "Features.csv". These files will be collected for visualisation and can be seen in the second column.



In the fourth column, the user can create new labels, e.g. “Control” or “Mutated”, and assign these labels to a specific well by first selecting the label followed by the desired well.

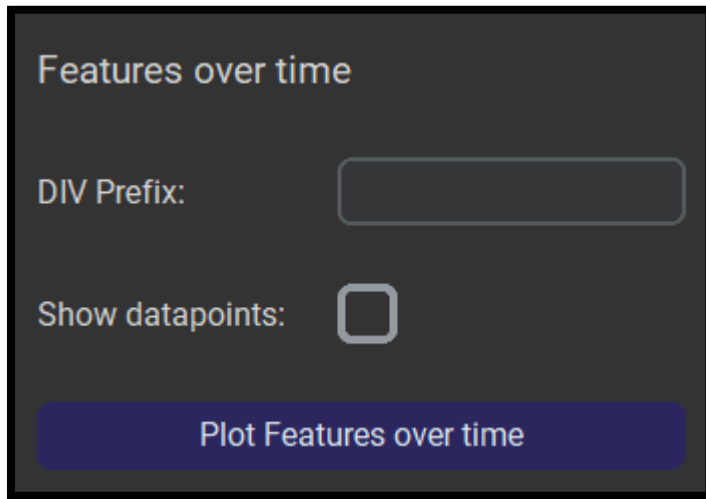


These labels can be saved, imported or reset using the buttons on the right side of the window.

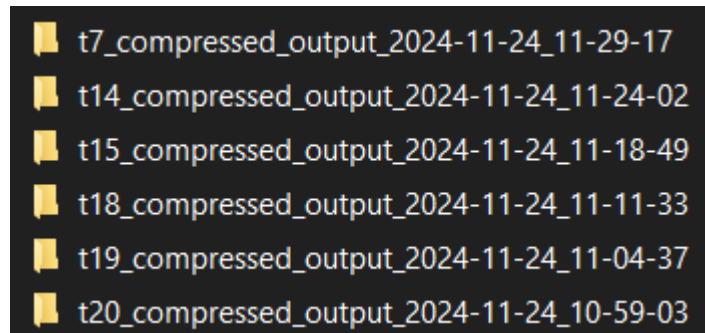
After the labels have been successfully assigned to their respective wells, there are two visualisation options, **Features over time** and **Boxplots**.

Features over time

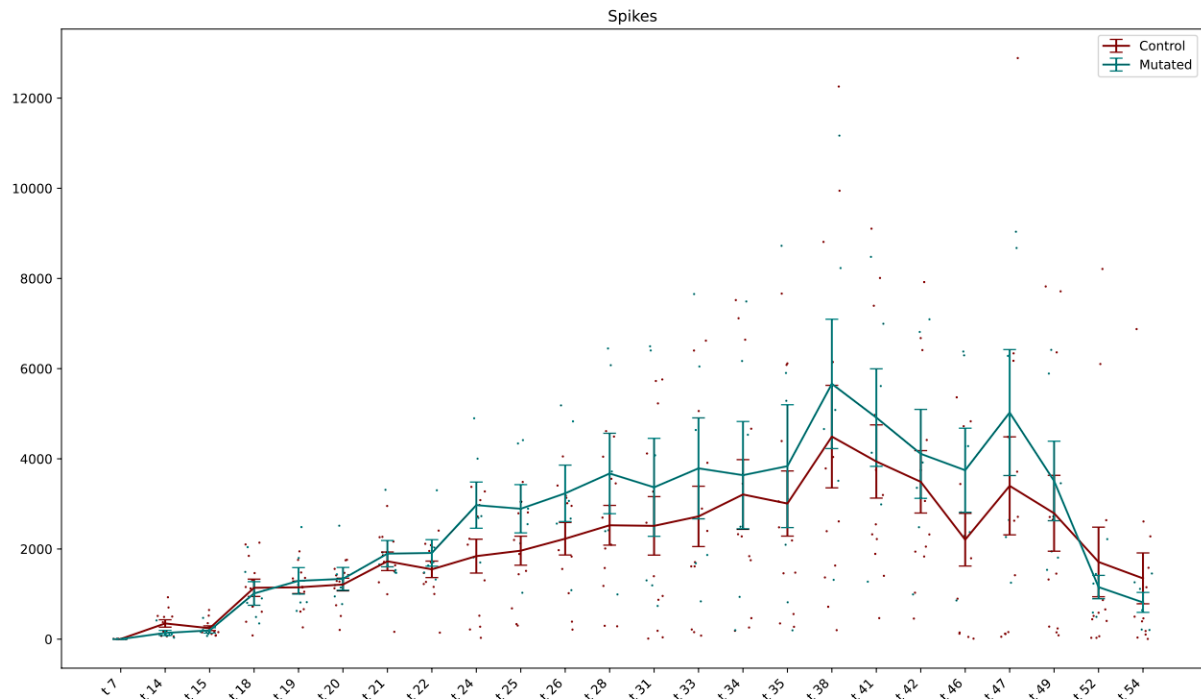
The **Features over time** function allows the user to select multiple measurements at once and visualise how the different features develop over multiple days.

A screenshot of a software interface titled "Features over time". It has a dark background. At the top, the title "Features over time" is displayed in a light blue font. Below the title, there are two input fields. The first is labeled "DIV Prefix:" in a light blue font, followed by a white rectangular text input box. The second is labeled "Show datapoints:" in a light blue font, followed by a white square checkbox. At the bottom of the interface, there is a large, rounded rectangular button with a dark blue background and the text "Plot Features over time" in a light blue font.

This functionality requires the selected files to follow a certain naming convention, where the relative date of the measurement is present in the name of the output features.csv file, and is preceded by a certain prefix such as '**DIV**' or '**t**'. This can be seen in the following example:

A screenshot of a file explorer showing a list of files. Each file name starts with a yellow folder icon, followed by a prefix, then "compressed_output", then a date and time in YYYY-MM-DD_HH-MM-SS format. The files are: t7_compressed_output_2024-11-24_11-29-17, t14_compressed_output_2024-11-24_11-24-02, t15_compressed_output_2024-11-24_11-18-49, t18_compressed_output_2024-11-24_11-11-33, t19_compressed_output_2024-11-24_11-04-37, and t20_compressed_output_2024-11-24_10-59-03.

This prefix should be supplied to the application using the **DIV Prefix** entry. As an extra option, the user can choose whether to show the individual datapoints on the final figures. Lastly, press **Plot Features over time** to generate the visualisations, this might take a little bit.

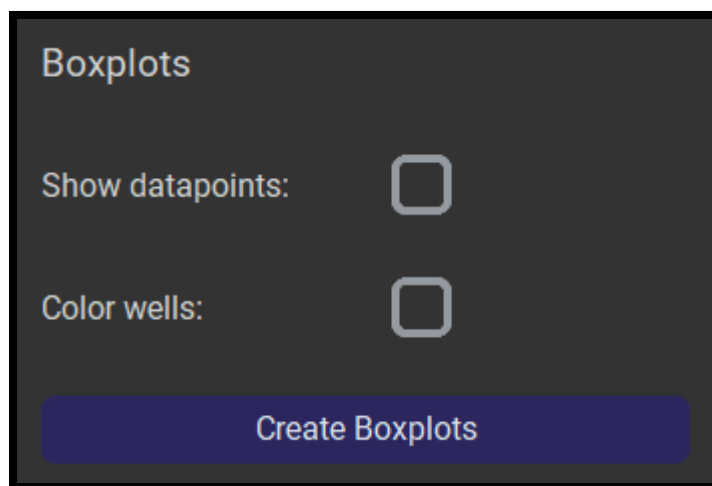


The error bars represent the standard error of the mean

(<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.sem.html>).

Boxplots

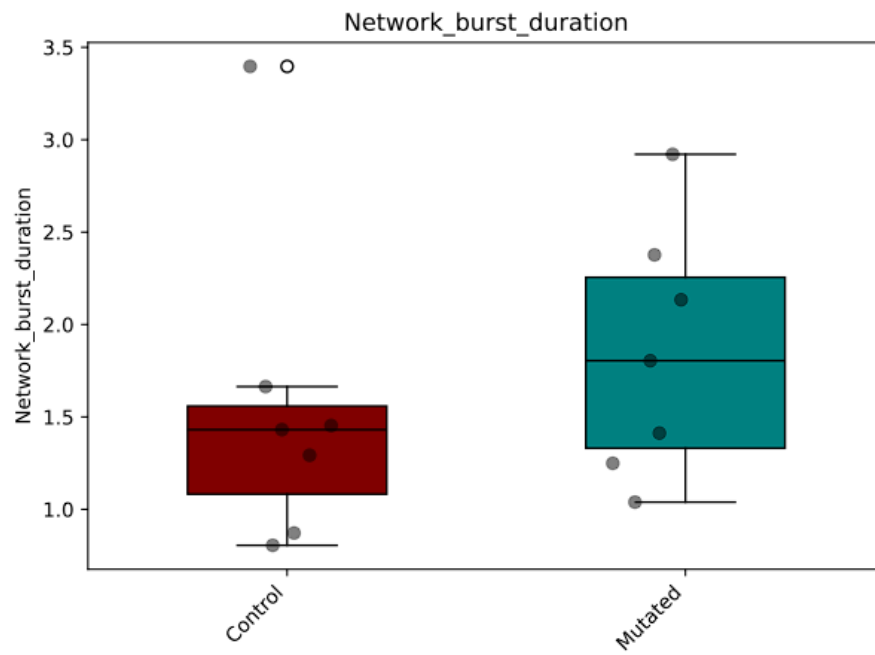
The boxplot functions will compare multiple groups against each other using boxplots.



The function has multiple options;

- **Show datapoints**
Whether to show the individual datapoints in the figures.
- **Colour Wells**
Whether to assign a specific colour to each well. Can be used to discern the results of different wells when combining multiple measurements.

Lastly, press **Create Boxplots** to generate the figures. Depending on the amount of data, this might take a little bit.



Parameters

The CureQ MEA library contains a wide range of parameters that can be used to alter the analysis pipeline. Described below are all the parameters, and how they affect the analysis pipeline.

The default parameter values are stored as a dictionary in the library, and can be accessed as follows:

```
from CureQ.mea import get_default_parameters
print(get_default_parameters())
```

Bandpass filter

Low cutoff

Low cutoff frequency for the Butterworth bandpass filter.

Default value: 200 Hz

High cutoff

High cutoff frequency for the Butterworth bandpass filter.

Default value: 3500 Hz

Order

The filter order used for the Butterworth bandpass filter.

Default value: 2

Threshold

Threshold portion

Portion of the data that is used to calculate the threshold value. Higher values can give a more accurate estimate of the background noise level, but takes longer to compute.

Default value: 0.1

Standard deviation multiplier

Value that is multiplied with the standard deviation of a portion of the data to create a threshold that determines whether this portion of the data contains purely noise or potential spikes.

Default value: 5

RMS multiplier

Value that is multiplied with the root mean square (RMS) of the background noise to determine the threshold for the spike detection.

Default value: 5

Spike detection

Refractory period

The refractory period of the spike. The time in which only one spike can be detected.

Value should be given in seconds.

Default value: 0.001

Spike validation

Spike validation method

The spike validation method used to determine whether a threshold crossing should be considered a neuronal signal or discarded.

Possible options: Noisebased, none.

Default value: Noisebased

Exit time

The time the signal must drop/rise below a certain amplitude before/after it has reached its absolute peak.

Value should be given in seconds.

Default value: 0.001

Drop amplitude

Multiplied with the surrounding background noise of the spike to determine the amplitude the signal must drop.

Default value: 5

Max drop

Value multiplied with the threshold value of the electrode to determine the maximum amplitude the signal can be required to drop.

Default value: 2

Burst detection

Minimal amount of spikes

The smallest amount of spikes that can form a single channel burst.

Default value: 5

Default interval threshold

The default inter-spike interval (ISI) threshold used for burst detection. Value should be given in milliseconds.

Default value: 100

Max interval threshold

The maximum ISI threshold that can be used for burst detection. Values should be given in milliseconds.

Default value: 1000

KDE bandwidth

The bandwidth of the Kernel Density Estimate (KDE) used for determining the burst detection parameters.

Default value: 1

Network burst detection

Min channels

The minimal amount of channels that should be participating in a network burst to consider it as one. Value should be given as a percentage.

Default value: 0.5

Thresholding method

The method used to automatically calculate the threshold used for determining high-activity periods. Possible values: Yen, Otsu, Li, Isodata, Mean, Minimum, Triangle. See <https://scikit-image.org/docs/0.24.x/api/skimimage.filters.html#>

Default value: Yen

KDE bandwidth

The bandwidth used for the kernel density estimate representing spike activity.

Default value: 0.05

Other

Remove inactive electrodes

Whether to remove inactive electrodes from the calculation when averaging single electrode features.

Default value: True

Activity threshold

The lowest spiking frequency and electrode can have before being considered 'inactive'. Value should be given in hertz.

Default value: 0.1

Use multiprocessing

Whether to use multiprocessing to analyse the different electrode in parallel. In machines with sufficient amount of CPU-power and RAM, this will speed up the analysis significantly. This setting will not alter the outcome of the experiment.

Default value: False

Convert Axion .raw to hdf5

Data generated from an Axion Biosystems MEA will be saved in the .raw format. Unfortunately, it is not possible to open this format in python. For the MEA-library to analyse these files, they must first be converted to the hdf5 format. The only way to do so, is to read the .raw files in MATLAB using the [AxionFileLoader](#). This repository must first be downloaded and added to the MATLAB path.

A script for this is available in the [github repository](#). This script will read the raw voltage data from the file, and save them in a new hdf5 file. After this, the hdf5 file can be analysed by the MEA-library. For this script to work, the AxionFileLoader must be installed in MATLAB.

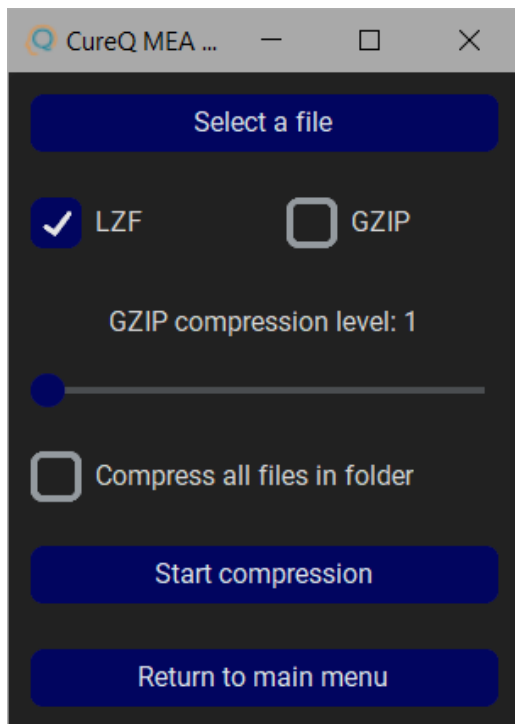
After the conversion has been completed, these files are not compressed yet, and might take up a lot of storage space. Hdf5 files can be compressed using the library, the GUI (see [Compress/Rechunk files](#)) or the following tool:

https://support.hdfgroup.org/documentation/hdf5/latest/h5_tools_r_p_u_g.html

Compress/Rechunk files

Files that originate from Multichannel Systems MEAs, or files that have been converted from the Axion .raw format are inefficiently chunked. This means that if the application wants to read the data of a single electrode, it must read the data of all electrodes at the same time. If the files are rechunked, this means that single electrode data can be loaded faster, and the analysis can be completed with less use of RAM. The application will automatically rechunk files if they are inefficiently chunked, and create a copy. Additionally, compression/rechunking can be performed manually in the GUI.

In the main menu, navigate to **Compress/Rechunk** files.



Besides rechunking the data, a compression algorithm will be applied as well. The default is LZF compression, which is not as efficient as GZIP, but achieves the compression in a much faster time. When GZIP is selected, the user can select different compression levels, where higher values will lead to smaller files, but an increased processing time.

By ticking **Compress all files in folder** the application will loop through all files in the parent directory and apply the rechunking and compression.

Features

Feature	Description	Column name
Spike features		
Spikes	The total amount of spikes recorded in a single electrode (spikes)	Spike
Mean firing rate	The average activity of the electrode, calculated as the average amount of spikes per second. (spikes/s)	Mean_FiringRate
Mean ISI	Mean inter-spike interval. The average amount of time between two successive spikes. (s)	Mean_ISI
Median ISI	The median of the inter-spike intervals. (s)	Median_ISI
Ratio of median ISI over mean ISI	The ratio of the median ISI over mean ISI. Calculated by dividing the median by the mean.	Ratio_median_ISI_over_mean_ISI
Inter-spike interval variance	The average of the square deviations from the mean from the inter-spike intervals. (s)	Interspike_interval_variance
Coefficient of variation ISI	The coefficient of variation of the inter-spike interval. (s)	Coefficient_of_variation_ISI
Partial autocorrelation function of the ISI	The partial autocorrelation of lag 1 of the ISIs. This value is calculated using the “statsmodels” python library (Seabold & Perktold, 2010)	Partial_autocorrelation_function
Mean absolute spike amplitude	The average absolute amplitude of the spikes. Spike amplitude is measured from 0 to the absolute top of the spike	Mean_absolute_spike_amplitude
Median absolute spike amplitude	The median absolute amplitude of the spikes	Median_absolute_spike_amplitude
Coefficient of variation of absolute spike amplitude	The coefficient of variation of the absolute spike amplitude	Coefficient_of_variation_absolute_spike_amplitude
Burst features		
Total number of bursts	The amount of burst that are detected in a single electrode. (bursts)	Total_number_of_bursts
Average burst length	The average length of a burst in a single electrode. (s)	Average_length_of_bursts
Burst length variance	The variance in the length of the bursts. (s)	Burst_length_variance
Coefficient of variation burst length	The coefficient of variation of the burst length. (s)	Coefficient_of_variation_burst_length

Mean inter-burst interval	The average amount of time between two bursts. (s)	Mean_interburst_interval
Variance of inter-burst interval	The variance of the inter-burst interval. (s)	Variance_interburst_interval
Coefficient of variation IBI	The coefficient of variation of the inter-burst interval. (s)	Coefficient_of_variation_IBI
Inter-burst interval partial autocorrelation function	The partial autocorrelation of lag 1 of the inter burst interval.	Inter-burst_interval_PACF
Mean intra-burst firing rate	The average firing rate in a burst. (spikes/s)	Mean_intra_burst_firing_rate
Mean spikes per burst	The average amount of spikes per burst. (spikes)	Mean_spikes_per_burst
Mean absolute deviation of spikes per burst	The mean absolute deviation of the amount of spikes per burst. (spikes)	MAD_spikes_per_burst
Isolated spikes	The portion of spikes that are isolated (not part of a burst). (value from 0 to 1)	Isolated_spikes
Single channel burst rate	The average amount of bursts per second. (bursts/s)	Single_channel_burst_rate
Network Features		
Network bursts	The total amount of network bursts in a well (network bursts)	Network_bursts
Network burst duration	The average duration of a network bursts. Calculated as the distance between the outer edges of the network burst. (s)	Network_burst_duration
Network burst core duration	The average duration of the network burst core. (s)	Network_burst_core_duration
Network burst core duration coefficient of variation	The coefficient of variation of the network burst core durations. (s)	Network_burst_core_duration_CV
Network inter burst interval	The average time between the network bursts (s)	Network_interburst_interval
Network inter burst interval partial autocorrelation function	The partial autocorrelation of lag 1 of the network inter burst intervals	Network_IBI_PACF

Network burst to network burst core ratio	The average ratio between the length of the network burst and network burst core.	NB_to_NBc_ratio
Variation of network IBI	The variation between the network interburst intervals (s)	Network_IBI_variance
Coefficient of variation network IBI	The coefficient of variation between the network interburst intervals (s)	Network_IBI_coefficient_of_variation
Mean spikes per network burst	The average amount of spikes in a network burst	Mean_spikes_per_network_burst
Network burst firing rate	The average firing rate in the network bursts (spikes/s)	Network_burst_firing_rate
Network burst inter spike interval	The average inter spike interval in the network bursts (s)	Network_burst_ISI
Portion of spikes participating in network bursts	The portion of spikes that are participating in network bursts.	Portion_spikes_in_network_bursts
Portion of bursts participating in network bursts	The portion of bursts that are participating in network bursts.	Portion_bursts_in_network_bursts
Ratio of left outer burst	The average ratio of the left outer burst compared to the burst core	Ratio_left_outer_burst_over_core
Ratio of right outer burst	The average ratio of the right outer burst compared to the burst core	Ratio_right_outer_burst_over_core
Left right ratio	The ratio of the left outer burst compared to the right outer burst	Ratio_left_outer_right_outer
Participating channels in network burst	The average amount of channels that are actively bursting during network bursts	Participating_electrodes

Python Library

Besides the GUI, the MEA analysis tool can also be called as a python library, and has a few functions that can be of use to the user. Detailed information about these functions can be found on the documentation hosted on GitHub. For now, let's walk through an example to fully analyse a MEA file.

Analyse MEA file

Firstly, import the necessary functions:

```
from CureQ.mea import analyse_wells, get_default_parameters
```

Next, we define some variables that we later need to pass to the function.

```
fileaddress='C:/mea_data/mea_experiment.h5'  
sampling_rate=20000  
electrode_amount=12
```

Then, we retrieve the dictionary containing the default parameters so we can alter the analysis. In this case we turn on multiprocessing to speed up the analysis.

```
parameters = get_default_parameters()  
parameters['use multiprocessing'] = True
```

Finally, pass all the arguments to the `analyse_wells` function to initiate the analysis. Because we turned on multiprocessing, we must use and “if `__name__ == '__main__'`”: “guard here. Otherwise, the application will eventually create an infinite number of processes and eventually crash.

```
if __name__ == '__main__':  
    analyse_wells(fileaddress=fileaddress,  
                  sampling_rate=sampling_rate,  
                  electrode_amnt=electrode_amount,  
                  parameters=parameters  
                  )
```

In the end, it should look like this:

```
from CureQ.mea import analyse_wells, get_default_parameters  
  
fileaddress='C:/mea_data/mea_experiment.h5'  
sampling_rate=20000  
electrode_amount=12  
  
parameters = get_default_parameters()  
parameters['use multiprocessing'] = True  
  
if __name__ == '__main__':
```

```
analyse_wells(fileaddress=fileaddress,  
              sampling_rate=sampling_rate,  
              electrode_amnt=electrode_amount,  
              parameters=parameters  
              )
```