VIEW Handbook

VIEW: a program to analyze optical imaging data in neuroscience, combining interactive and off-line high level data analysis

See the publication: #reference#

Introduction

View is a powerful program to analyze optical imaging data interactively, and to analyze multiple data sets off line using uniform settings, thus allowing for rigorous statistical data. The program has been developed over many years to study odor-evoked activity patterns in insect brains, for a variety of quite different experimental approaches. Therefore, many different settings were implemented, resulting in a strong and powerful platform: different data formats, different dyes used, different calculations and corrections for bleaching, noise, movement artefacts. Data reporting as images, as movies, as time-traces. Using information about single stimuli, complex stimuli, repeated stimuli – it is all in there. Most importantly: everything that can be done interactively can also be done off-line, to create statistically coherent datasets.

We have taught our students and our colleagues how to use VIEW, and we learned that it can be an overwhelming experience. Settings are expressed in *flags*, and there are so many of them, that keeping track is difficult. Also, as in all these instances: the help section in the Wiki works well for those who already know what they are looking for, but is difficult to use for a novice.

This handbook is hands-on: after a short introduction about the philosophy of the program, you will be guided through a synthetic data set, and learn about the possibilities of VIEW.

Table of content

Philosophy of the program

View is organized in a series of separate steps/sections, each with its own *flags* (settings).

Define experiment -> create a list file

An experiment consists of multiple measurements in one animal (or preparation). For example, we record odor-evoked activity patterns in the brain of an insect: a single animal is exposed to different odorants and controls, and maybe treated with pharmacological substances. A list needs to be created to collect this information: stimulus identity, stimulus timing, pharmacological treatment, measurement length, storage place for original data, technical details of the measurements (e.g. frame rate), …
You can create a list file using Excel (or any other spreadsheet program), or create it adapting the scripts in the *log2list_examples* folder. (Many setups create a .log file during measurements which contains, among others, the file names of the data, and the time of measurements, and the python scripts use those .log files to create the .list files).

Load data and calculate signals-> yields raw data and signals

Where does the data come from, i.e. what format is the raw data: Zeiss confocal microscope, Till Photonics imaging system, 3D-TIFF-stack, etc. Do you want to apply filters when loading (e.g. against bleaching, movement correction, noise filters…), or reduce data size (e.g. apply post-hoc binning).
The raw data is converted into signals, e.g. deltaF/F, or ratiometric calculations. Settings for signal calculation include, for example, which frames to use for F_0 calculation.

Interact with data -> use ILTIS, yields .coor and .area files

The ILTIS package within VIEW allows to interactively look at imaging data. Multiple traces from different areas within a measurement, or multiple traces from a location in different measurements. Raw data and signals can be compared. Movement artefacts, other artefacts, peculiarities – all is accessible in a fast and efficient way.
Within ILTIS, you can define areas of interest (ROIs, defined here in the .coor file) for calculating traces (e.g. the location of cell bodies, or of olfactory glomeruli or other brain areas). Furthermore, you can define the area of the brain in the image – this is useful for visualization purposes, or for bleach correction settings.

Create overviews -> yields activity images

Within VIEW, you can create false-color coded images of your measurements. Many settings (*flags*) are needed: do you want to calculate a maximum response, or the absolute response at a given time point? What filter? Color-table? Value for maximum-minimum activity? Activity shown only in the brain (use .area file) or also outside? Additional information? All flags can be tested in detail.

Work off line 1 -> yields tapestries

With all the settings that were experimented with in VIEW, you can now run the identical mathematics on all measurements for all animals in all of your experiments. This results in a "tapestry", which allows you to compare, for example, response magnitude across animals, or to compare response patterns (scale each image to its own maximum) or compare response magnitude (scale all images to a global maximum). You need a dedicated file that contains all the settings (a .yml file), and a file that defines which measurements to show in what order along the tapestry. A bit of organization, but great insight into your data!

Create time courses -> yields traces

Within VIEW, you can create activity traces, using the areas (ROIs) defined in the .coor file.

Work off line 2 -> yields time trace files

With all the settings that were experimented with in VIEW, you can now calculate time traces for all areas and all animals, writing the resulting data into a spreadsheet file. The important thing here: all animals, all measurements were treated with the identical settings (filters etc), making the resulting data statistically sound and comparable.
The spreadsheet files contain the measurements, and all metadata (stimulus identity, timing of the stimulus, name of the area, etc). It allows to efficiently do statistics or trace visualization using any statistical package (R, S, SPSS, even Excel…)

Create movies -> yields single movies

Within VIEW, you can create movies of your measurements. Many settings (*flags*) are needed: do you want to apply a filter? Which color-table? Additional information, such as time-point of stimulus, or numbers indicating time passed? All flags can be tested in detail.

Work off line 3 -> yields movies

With all the settings that were experimented with in VIEW, you can now create movies for all your measurements in all animals using the exact same settings. Off-line movie creation is feasible even with a slow computer – just come back the next day.

Hands on handbook: synthetic data

View contains an option to create synthetic data, which mimics real data (including bleaching, broken pixels on the CCD chip, axial movement, lateral movement…). You can use that dataset to test and practice many of the flags, settings, and options in VIEW. We will guide you through the process. Please take 10-30 minutes for this process, afterwards using VIEW with your own data will be much easier.
You can find all the example files in the folder *synthetic_data* in the repository. Please copy that folder into your own DATA folder before using it. It contains a suitable list file and a corresponding .yml file, allowing you to start right away.

Define experiment -> create a list file

An example for a measurement list file is given in the file *Synthetic_data_strip.lst.xls* in the folder *synthetic_data/02_LISTS*.

	Measu	Analyze	Animal	Comment	Cycle	DBB1	Label	Line	OConc	Odour	StimON	StimOFF	setting
0	1	1	Constructed_00	raw666	200.0	noFile	raw666_00	constructed	-10	FAKE	25	35	setting
1	2	1	Constructed_00	raw666	200.0	noFile	raw666_01	constructed	-1	FAKE	25	35	setting
2	3	1	Constructed_00	raw666	200.0	noFile	raw666_02	constructed	0	FAKE	25	35	setting
3	4	1	Constructed_00	raw666	200.0	noFile	raw666_03	constructed	0.5	FAKE	25	35	setting
4	5	1	Constructed_00	raw666	200.0	noFile	raw666_04	constructed	1	FAKE	25	35	setting
5	6	1	Constructed_00	raw666	200.0	noFile	raw666_05	constructed	2	FAKE	25	35	setting
6	7	1	Constructed_00	raw666	200.0	noFile	raw666_06	constructed	5	FAKE	25	35	setting
7	8	1	Constructed_00	raw666	200.0	noFile	raw666_07	constructed	10	FAKE	25	35	setting
8	9	1	Constructed_00	raw666	200.0	noFile	raw666_08	constructed	20	FAKE	25	35	setting

This mimics 9 measurements (lines) in a single animal (file). Each measurements has a unique value for “measu”, which is used to select that measurement. Columns include a label for the animal, cycle (this is ms for each frame), DBB1 (location of the file with the data, in the case of synthetic data no file), OConc (concentration of the stimulus – in this particular case used to create fictive responses of different size and polarity), Odour (nature of the stimulus), StimON-StimOFF (given here as frame numbers).

The file *view/flags_and_metadata_definitions/metadata_definition.csv* in the repository contains all possible column names that are used in VIEW. This includes information about more complex stimulus timing settings, treatments with pharmacological substances, absolute time of measurement, time of measurement relative to beginning of the experiment, size of pixels, age or sex of the animal, and many more.

Define experimental settings -> create a .yml file

An example for a .yml file is given in the file *view_synthetic_666.yml* in the folder *synthetic_data*.
This file contains all *flag* settings. When working with VIEW, you can change the flags interactively. When you have found flag values that are better suited for your data in VIEW, you can save them to the .yml file from within VIEW.
The file *view/flags_and_metadata_definitions/view_flags_definition.csv* in the repository contains all possible flag names that are used in VIEW.

LE_loadExp: 666 defines that the data be synthetic. A value of "3" is data format "Till Photonics". You can find all setup definition (i.e. readable data formats) in the file *view/flags_and_metadata_definitions/setup_definitions.csv.*
LE_CalcMethod: 3 defines that signals be calculated as deltaF/F. This method calculates relative changes in fluorescence, relative to the background which is defined as the average of frames **LE_StartBackground: 4** to **LE_PrestimEndBackground: 1**. (Of course, you can change these numbers). LE_PrestimEndBackground is relative to the start of the stimulus, which is known from the column StimON in the list file.
STG_Datapath: 01_DATA defines where measurement data are stored. It is part of the philosophy of VIEW that files in this folder are never touched or modified. (For synthetic data, that folder is empty).
STG_OdorInfoPath: 01_LISTS defines where the list files are to be found. Other flags define where results are to be stored. The folder structure defined by these flags is created when VIEW is used, if those folders are not yet present.
Other flags include which filters to be used, etc.

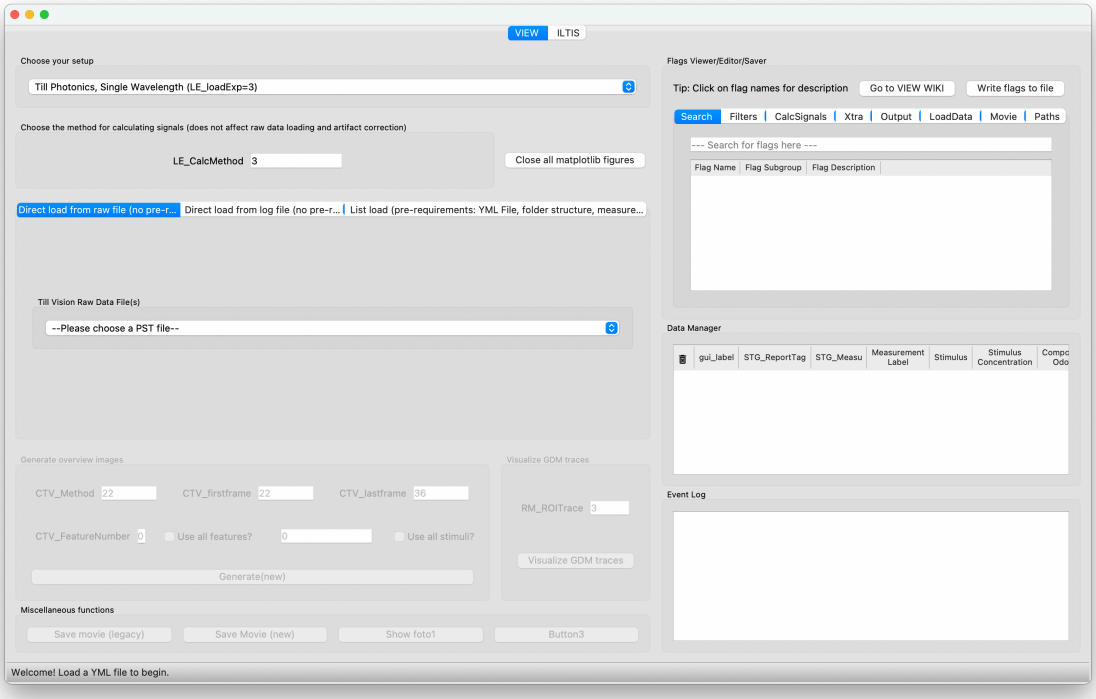
In this hands-on handbook we do not go through all flag settings: use the Wiki instead.

Run VIEW -> getting started

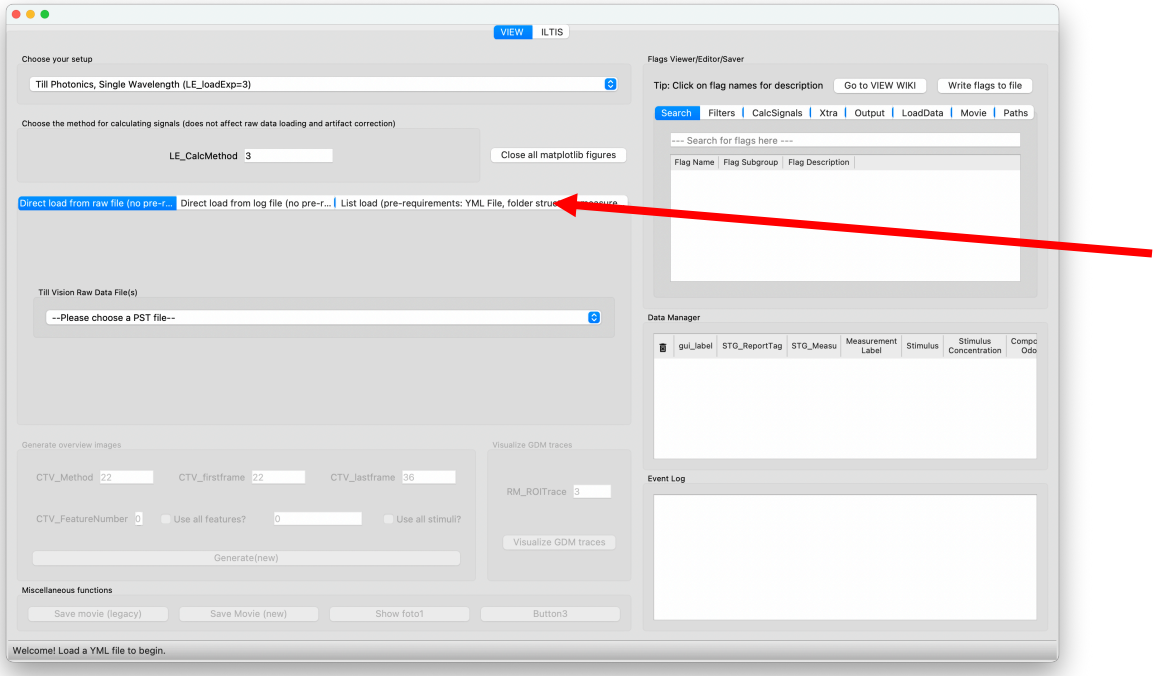
Here we assume that:

- You have installed python, created the appropriate environment, and installed view
- You have created a copy of the folder *syntetic_data* into your own data repository folder.
- You are in that directory, and
- You have started VIEW within python.

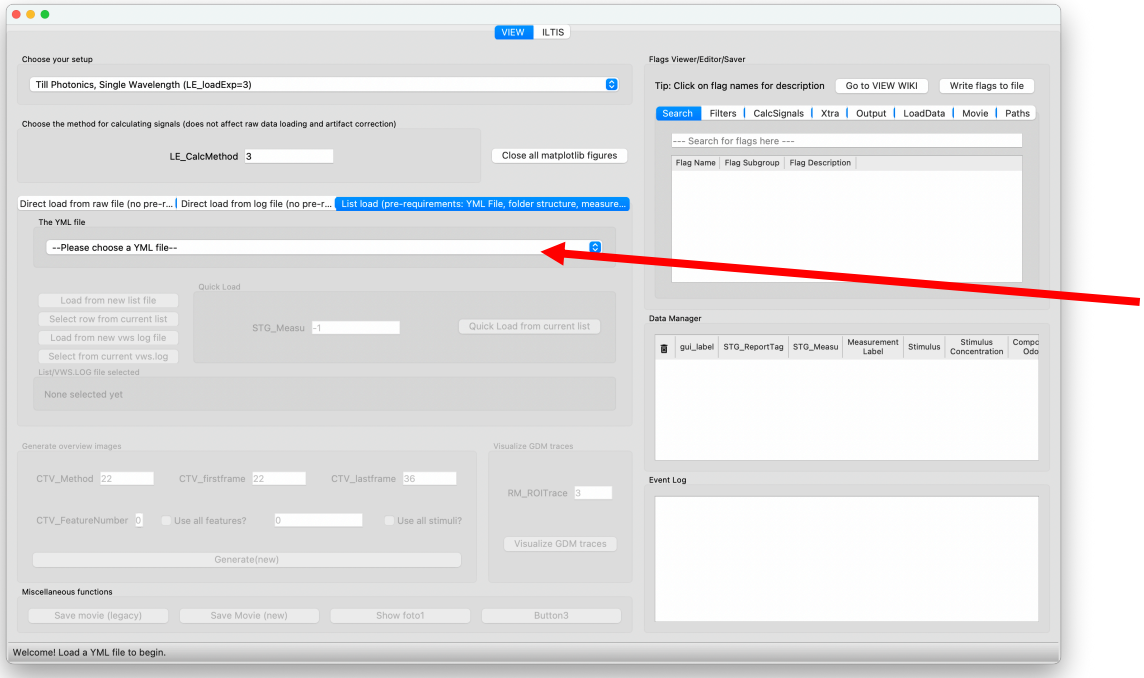
You will see this (details and graphical look depend on your operating system):



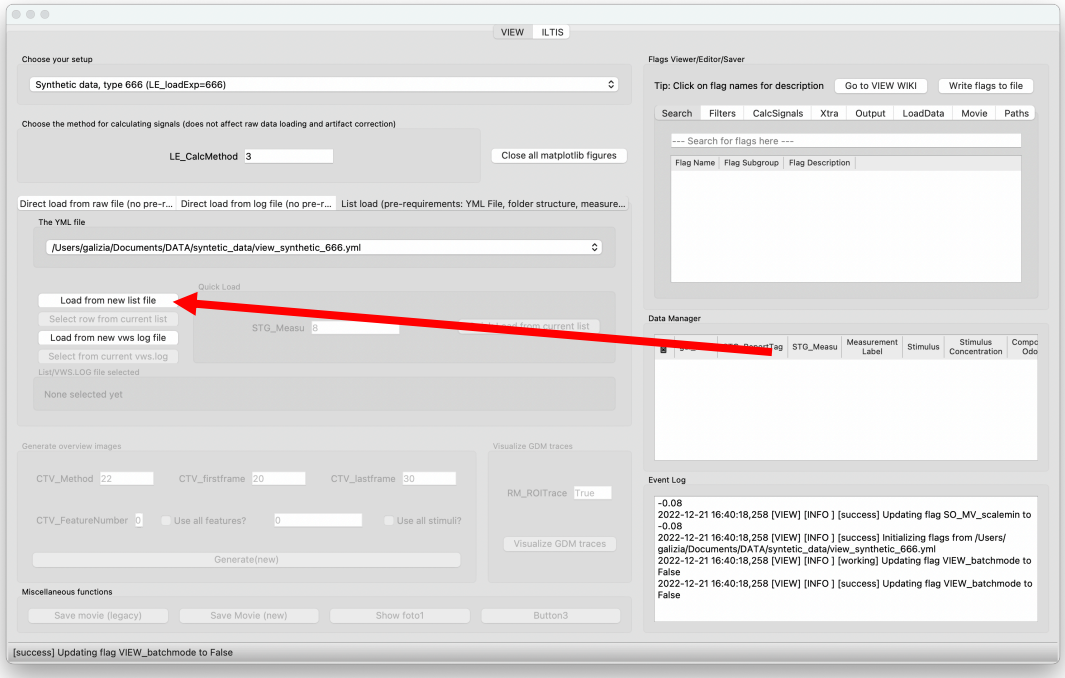
Run VIEW -> load .yml file



Click “List load” to load measurements using a .list file (preferred option). (Settings are remembered, so the second time you do this, the pull-down will look different).

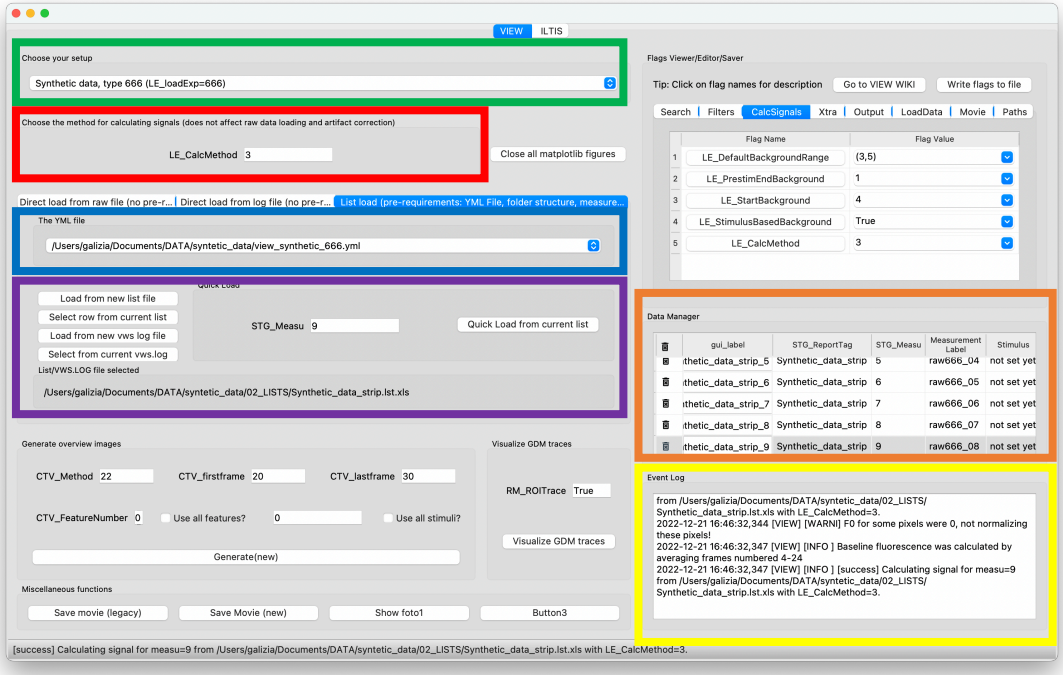


Choose the view_synthetic_666.yml file in the folder *syntetic_data* in your own data repository folder. (The program will remember the file, so the second time you want to load it, you will be faster).



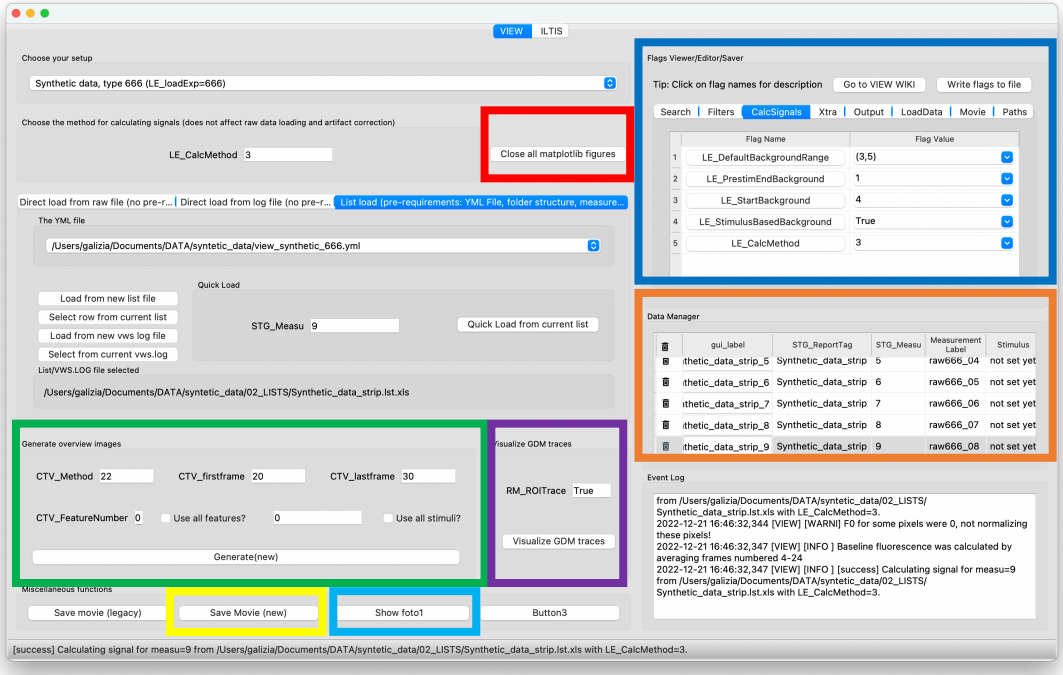
The Window now shows you which .yml file is loaded. The window at the bottom right has given some feedback about what has happened (e.g. created the necessary folders, loaded the flags, etc).
Now choose the **Synthetic_data_strip.lst.xls** file in the folder *syntetic_data/02_LISTS* in your own data repository folder. (The program will remember the file, so the second time you want to load it, you will be faster).
A window opens, and you can select to load one or several single measurements from this animal, or all (click top-left in the list). Select the measurements you want, and load the data. For this tutorial, select all measurements.

Let’s have a look at the components of the VIEW window now:



Information about the setup used (here: synthetic data, setup 666).
Information about how signals were calculated (here: deltaF/F, value for LE_CalcMethod is 3)
Information about the .yml file loaded
Options for quickly loading new measurements
List of measurements that have been loaded (that are in memory). Single lines can be selected, all data can be deleted, or single measurements can be deleted.
Log window reporting about events/calculations/errors. (Please check also the terminal window for error reports from python).

More about the VIEW window:



Flag settings. Here, you can look up each flag setting, search for flags, change flags. There is a direct hyperlink to the VIEW Wiki page. And there is a button allowing to overwrite the current .yml file with the values selected here. Flags are sorted into different categories (Filters, Output, LoadData etc.)

Close all matplotlib figures: with data analysis, sometimes many windows clutter the screen. Close them all in one click!

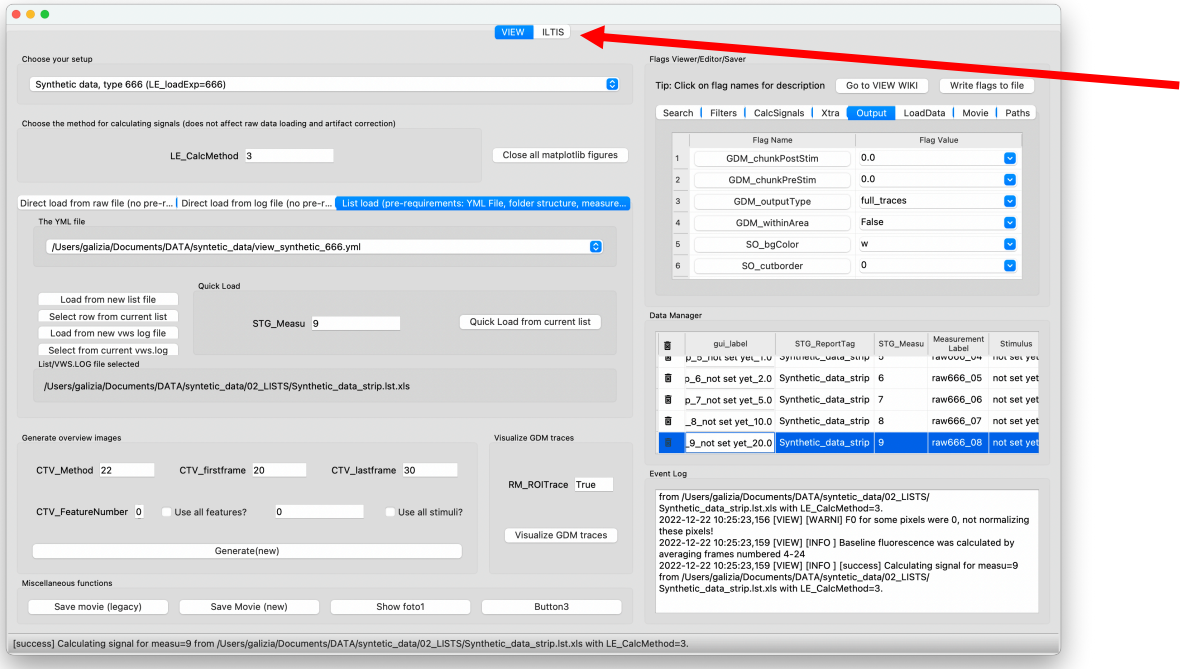
Generate overviews of the measurement selected in Data Manager (orange square), using the flags selected (blue square), the most important flags are repeated here (CTV_Method etc.). Once your flags are set, click “Generate(new)”

Generate traces of the measurements (using .roi file)

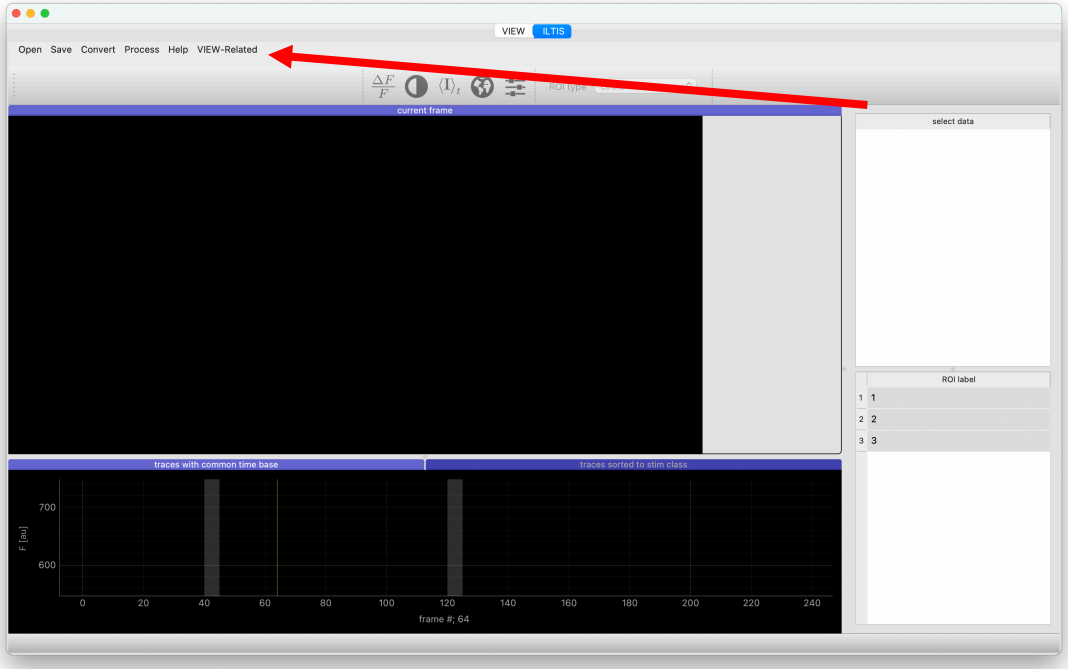
Save a movie file for the measurement selected in Data Manager (orange square).

Show a foto of the raw data (generally an averaged frame).

So, the data is loaded. Now, the next step in data analysis, is to look at the data interactively. For this purpose, change into the ILTIS section of VIEW.



ILTIS is empty – load the data from the VIEW section into the ILTIS section. To this purpose use the drop-down menu “VIEW-Related” and select “Quickly import all data from VIEW”. (If you want to load only some measurements, or if you want to adjust the name of each measurement that is shown in the window, select “Selectively import data from VIEW”.) Here we assume you chose the first option.



The ILTIS panel allows to visualize single frames, time traces, and to select areas of interest. Chose the ROI type (arrow), and then click into the frame to select an area (circle in this case). Click multiple times to create more areas.

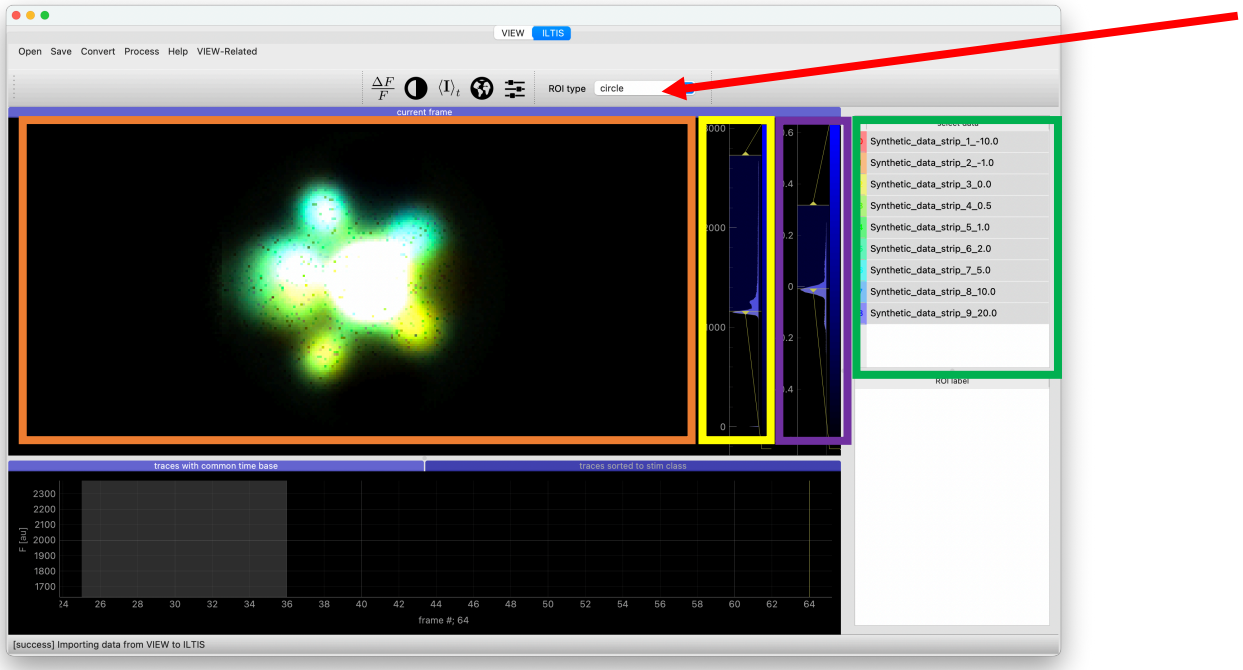
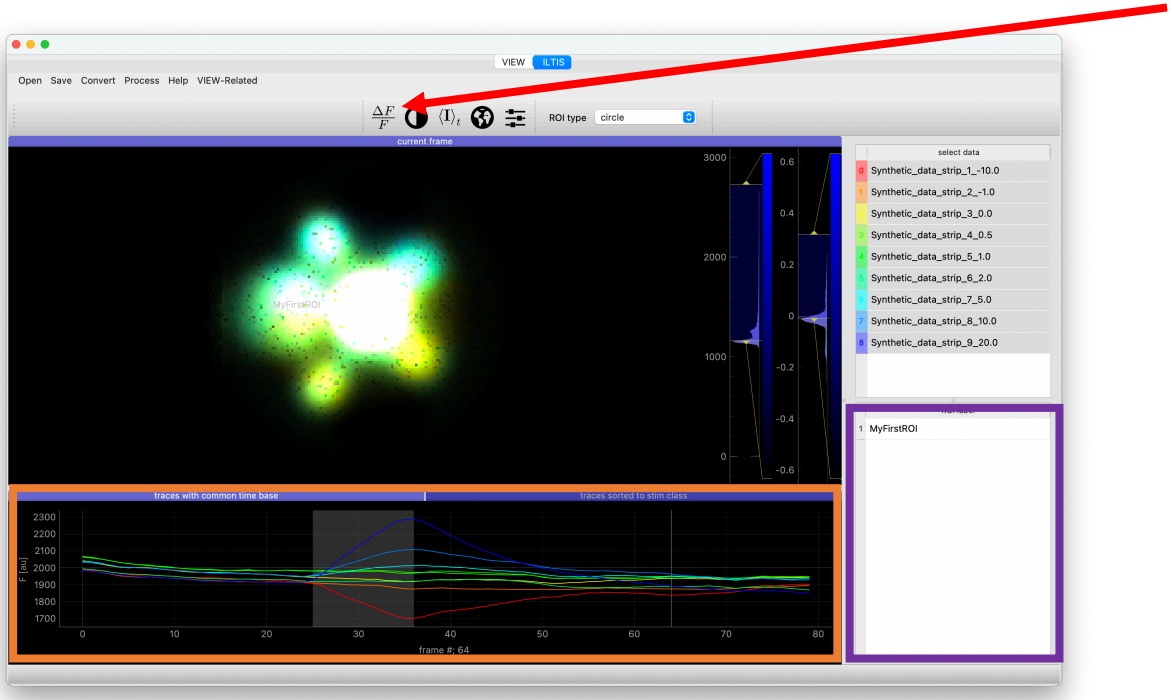


Image area for one selected frame (“current frame”), superimposing several measurements each with a different color if more than one measurement is selected.

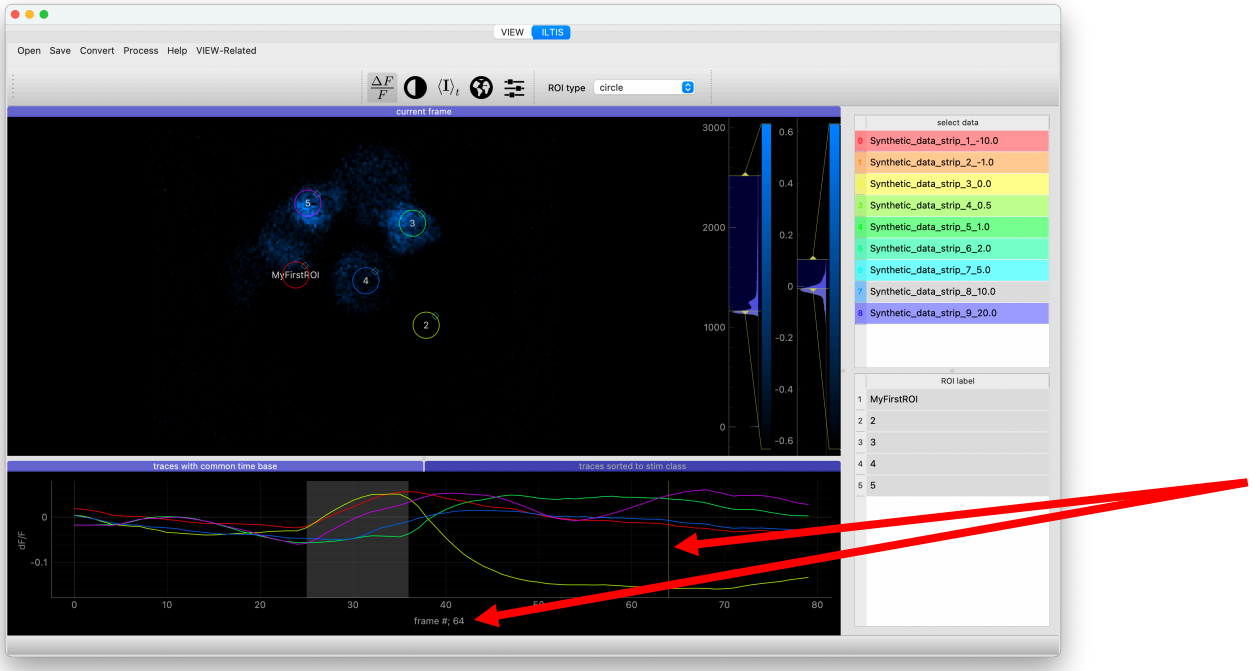
Area to adjust range for false-color code mapping, and to select the color table, for raw data display.

Area to adjust range for false-color code mapping, and to select the color table, for signals display (e.g. deltaF/F).

List of measurements. The names are created from columns in the .list file, and can be adjusted (flag: LE_labelColumns).



Area to show time traces. Here, multiple measurements are chosen, and one ROI, showing the responses to different stimuli in one location. .
Area to select ROIs. Change the name of the ROI here.
(Arrow) Button (let's call it "signal button") to flip between Raw data and Signals (deltaF/F, or, in case of ratiometric data, the ratio (even if the image in the button still remains deltaF/F)
Now click on the signals button (arrow), and then select a few more ROIs. Then in the "select data" panel select a single measurement (for the tutorial, select measurement 7), and in the "ROI label" panel select all ROIs.



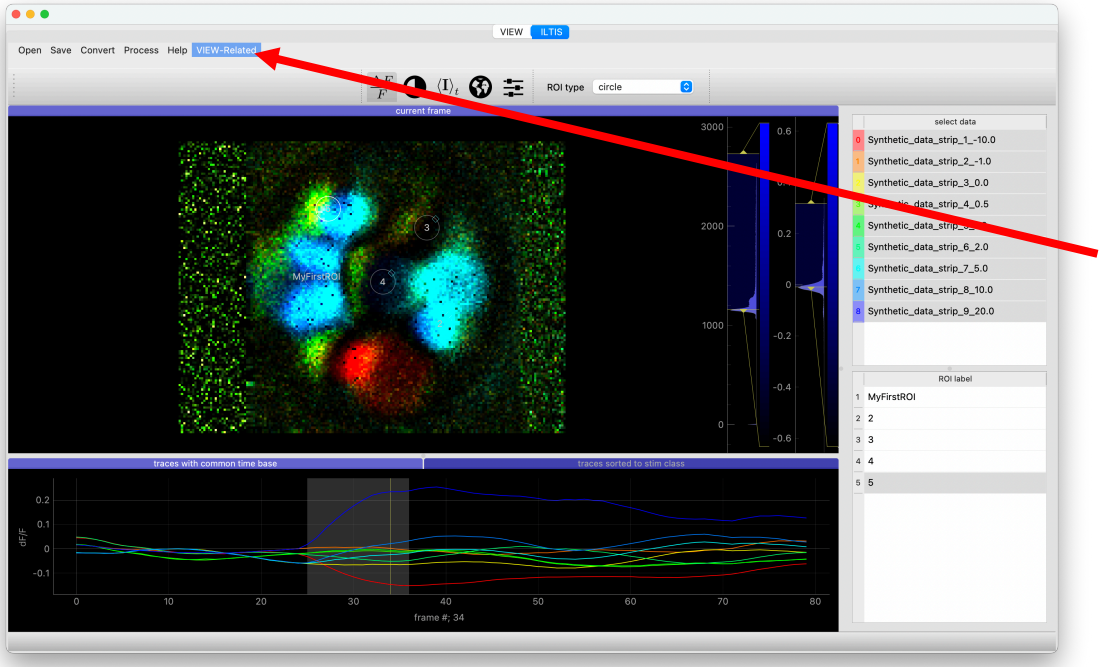
Here, we've selected 5 ROIs in measurement 7. ROI 2 responds to the stimulus with a fluorescence increase, and a decrease below baseline after the stimulus, ROI 3 has an OFF response (increased fluorescence after stimulus offset), "MyFirstROI" has slow response to the stimulus, and a slow decay. When your mouse clicked into the traces window, you can adjust the axes ranges. In the bottom left there is an "autoscale" button, useful if you've lost all of your traces.

The grey area is the stimulus. Stimulus timing information is taken from the list file (**Synthetic_data_strip.lst.xls)**.

There is a vertical yellow line (arrow – at the beginning it is located at position 0). Click that line, keep clicked, and move it sideways to select the frame that is displayed in the frame panel. Use the color bar to the right of the "current frame" window to change color boundaries and color scale (right click on color palette). Play around with your data – many of the possibilities in this section are intuitive in the sense that they use common practice effects for right-mouse click, for scrolling, for selecting, etc.

BTW: at the bottom, you can see the frame number that is selected (arrow). This is useful, for example, to identify the frame number with the maximum response.

Now we need to save the ROIs, in order to work with them later.



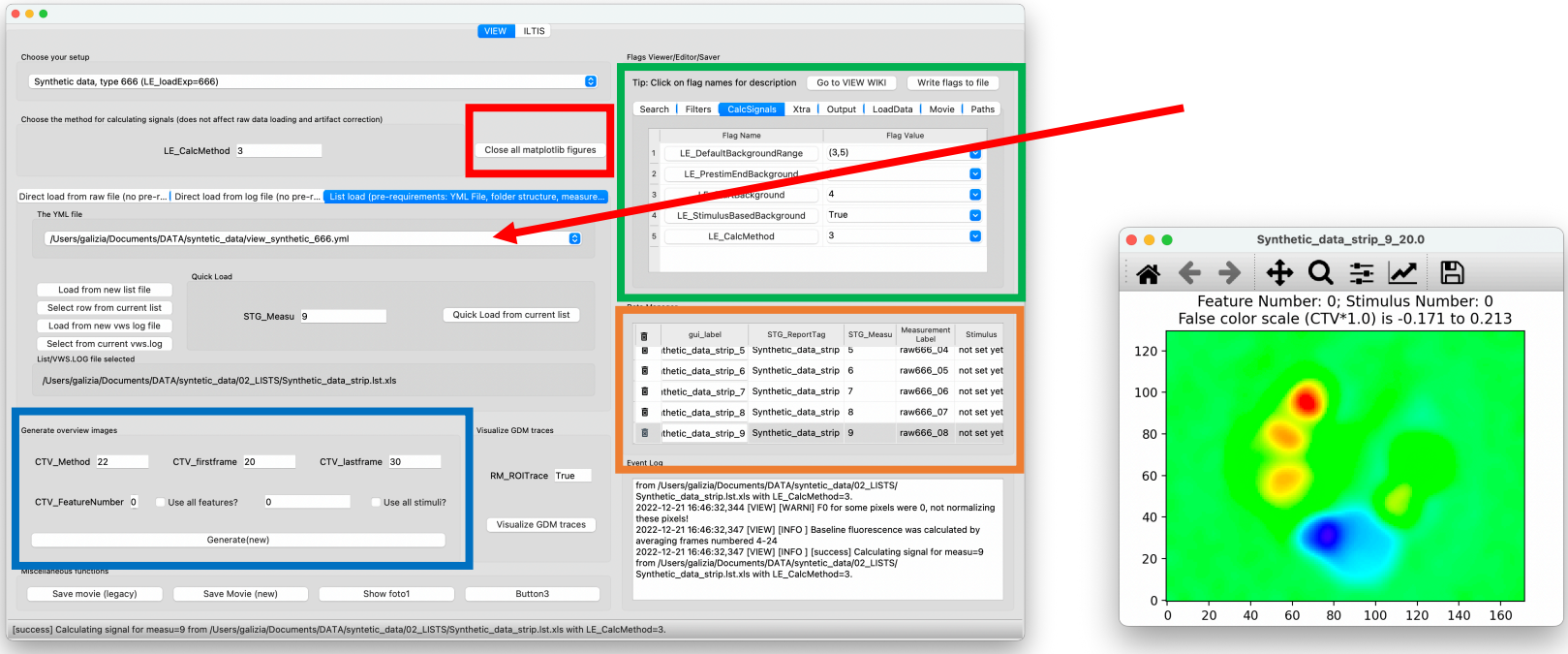
Here the view in “current frame” is: all measurements overlayed, each with its own color as indicated in the “select data” frame, traces shown for ROI 5, as indicated in the “ROI label” window, colors of the traces as for the frames from “select data”. The frame shown is indicated by the vertical line in the traces window (frame 34). The ROIs are shown in the frame view. (BTW: you can change ROI position and size any time using the mouse).

Use the pull-down menu “VIEW-Related” to save the ROIs. There is a choice if you have both circular and polygon ROIs, whether you want to save only the circular ROIs or all. This opens a window where you have to select which ROIs to save. (Advanced users may save separate ROIs for different measurements even within one animal; for the tutorial, choose: “save FOR ALL measurements of this animal”).

If you have a polygon ROI, you can save that one as an .area file. That is useful if the brain area is surrounded by non-brain, and the non-brain contains artefacts that should not be considered in some steps of the data analysis.

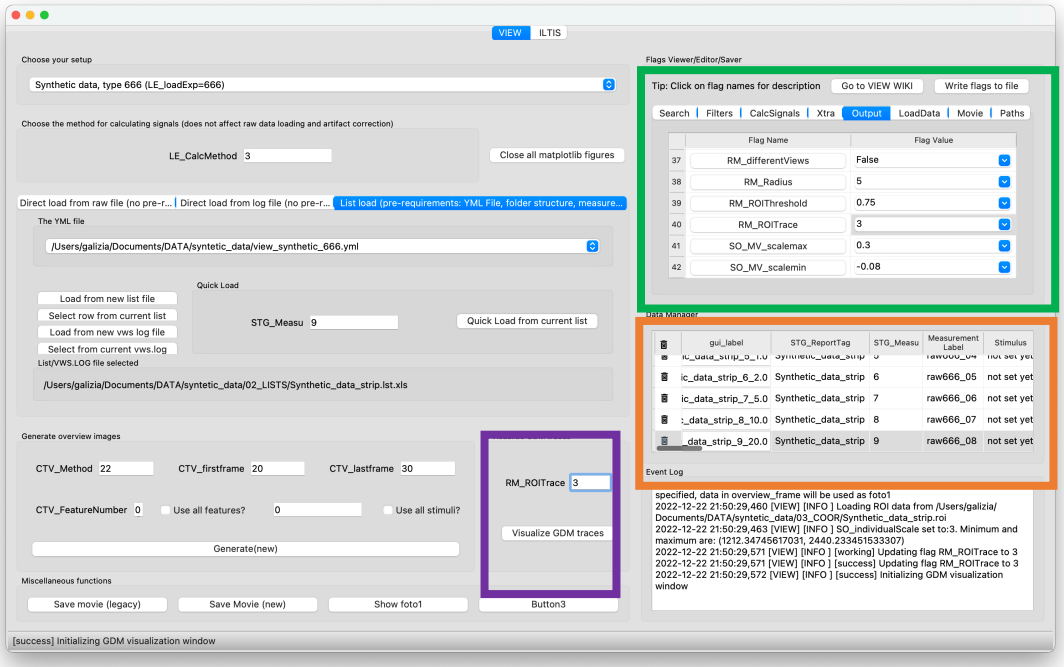
Where are ROIs saved? That is stated in the .yml file, in the flag STG_OdormaskPath. (After saving, a pop-up window also reports the saving location)

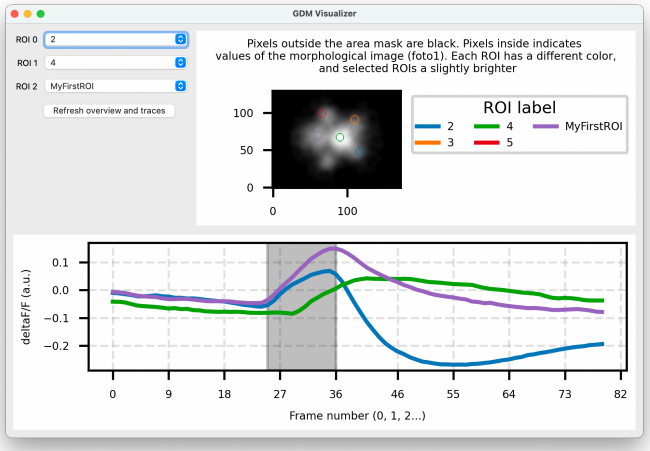
Back to the VIEW tab, to create overviews (interactive analysis mode):



An overview is a false-color coded display of response patterns. How to calculate responses? That is given by the CTV-Method (in the blue box). Method 22 calculates the difference between two time points: CTV_lastframe – CTV_firstframe (average of three frames each). Here, frame 20 is well before the stimulus, frame 30 is in the middle of the stimulus. Which measurement is used is selected in “Data Manager” (orange box). Click on Generate(new) to get the false-color coded image – shown here to the right. Many settings allow for fine tuning (see green box, most in the section “output”). Play with the flags – for example, try the difference between frames 45 and 25. Most important: Signal_FilterSpaceFlag and Signal_FilterSpaceSize (for filtering); SO_individualScale (value 3 for min/max taken from image, value 0 for min/max taken from flags SO_MV_scalemax and SO_MV_scalemin). When you have created too many overviews and you are confused, close all windows (red box). (Many flags have names with ‘SO_’ at the beginning, for ‘Show Overviews’. ‘LE_’ stands for ‘Load Experiment’ related flags. ‘CTV_’ stands for ‘Curve to Value’, since a time trace/curve is converted to a single value to show, for example, response strength).
If you played around too much with flags, and want to go back to the initial flag settings, you can always reload the .yml file (arrow)

Continue in the VIEW tab, to create traces (interactive analysis mode):

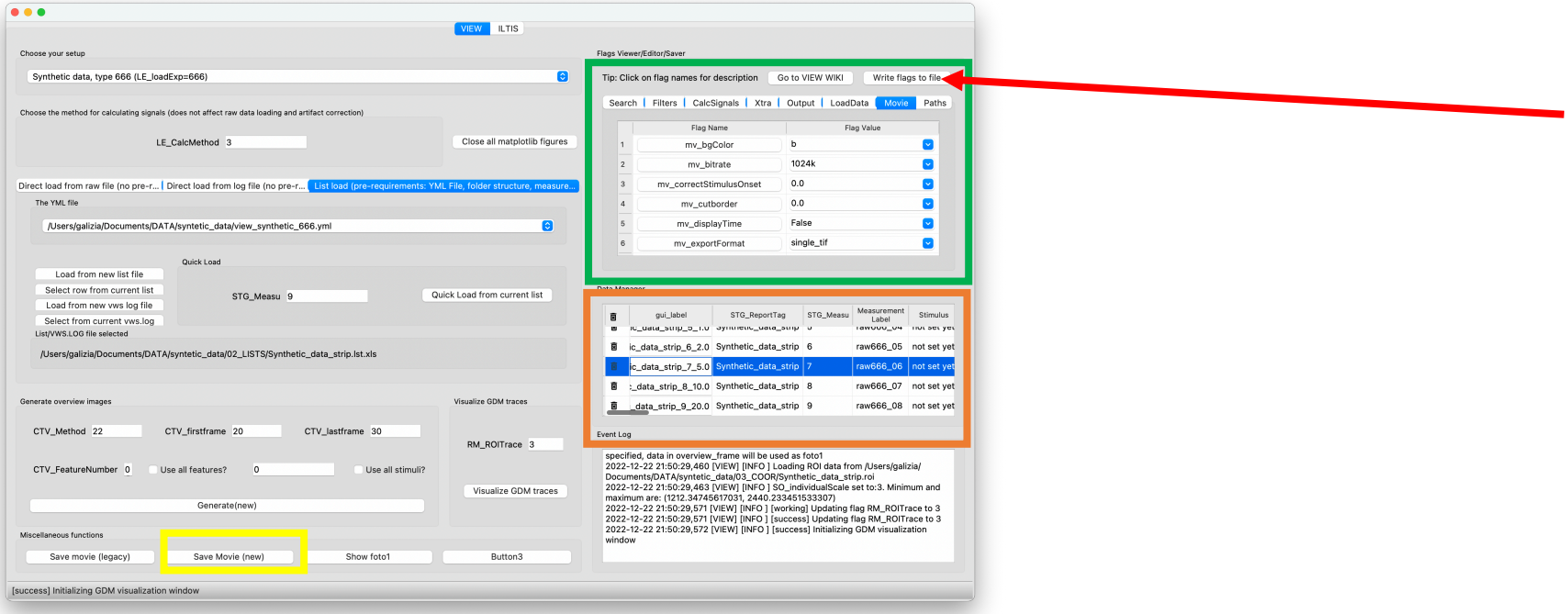




Traces are created in “Visualize GDM traces” (purple box), for the measurement selected in “Data Manager” (orange box), with the flag settings defined (green box). The flag RM_ROITrace selects which format traces are stored in. In our case, that is “3”.

Click on “Visualize GDM Traces” to open the appropriate window. In this window, you can display traces for up to three areas. This view is different from the view in ILTIS in that it uses the same code to be generated that is used for batch analysis (see below).

Continue in the VIEW tab, to create movies (interactive analysis mode):



Movies are created by clicking on the "Save Movie (new)" button. A movie will be calculated for the measurement selected in "Data Manager" (orange box), with the settings given in the flags (green box). Flags include format (single frames, frame stacks, codec used), filters, upper and lower limits for false-color coding, background color, foreground color, etc. – plenty of possibilities. Most flags related to movies start with 'MV_' for 'movie'.

The movie is saved to a subfolder of the output folder that is defined in flag STG_OdorReportPath, and needs to be opened from there. ('STG_' flags (for 'string') are collected in the 'paths' tab of the flags viewer (green box).

All work in interactive analysis mode leads to selecting the right flags values. Once you are done, you can save all flags with their values into a .yml file – or overwrite the current .yml file (arrow).

Batch analysis: movies

With this, we have seen the three main output options: overviews, traces, movies.
Now it is time to automate the output, in order to apply the same flags (i.e. the same mathematics) to all measurements, to all animals, so that we can compare the results across conditions in a rigorous way.
Automation is done off-line. The important aspect here: off-line analysis and interactive analysis in VIEW use the same code! Therefore, preparing all flags in VIEW will then allow to run the off-line system. **That is: you test overviews, traces and movies in interactive mode, select the best flags, and save those flags into a .yml file. And then you use that .yml file for batch analysis.**

In this tutorial, let's first export movies.

Batch analysis is done by running a python script.
In the folder "06_PROGS", you will find the file *export_movies_synthetic.py*.

You can run this file directly in python, remember that you need to be in the environment created for view (command line: *'python export_movies_synthetic.py'*, or calling it from a python editor).
Please adapt this file to your own data/settings. Note the following most important things:

- Within each animal, specify which measurements are to be evaluated (in column "Analyze" of the animal.list file, i.e. the file **Synthetic_data_strip.lst.xls** in this tutorial; see below).
- Specify which animals are to be evaluated (here it is "Synthetic_data_strip". It is always the stem of the animal.list file)
- Define "moaf", the mother of all folders, if you are using a different folder structure from our standard

In the code of *export_movies_synthetic.py* you will find the variable **Analyze_column_values_to_use**:
This variable indicates which lines of the animal.list file to use (of the 02_LISTS/Synthetic_data_strip.lst.xls in our case).
Open that file, and in column "Analyze", insert, for each row, what you want to do with that row. In our example, all lines have the value "1". Since in our example *analyze_values_to_use=(-1,1)*, all lines will be analyzed. You could set some lines to value "2", or value "3", and thus create different subgroups. In our group, the tradition is this: value '-1' for measurements that we did not yet look at in detail, value '0' for measurements that need to be excluded (e.g. strong movement artefacts), value '1' for good measurements, other values for special cases.
Running the program will create movies for each measurement in subfolders of 04_OUTPUT, using the flags specified in *view_synthetic_666.yml* (or the .yml file specified).

Batch analysis: tapestries

Overviews, when there are many, are called tapestries.
In VIEW and ILTIS, we have looked at the traces. Stimulus is at frame 25, peak response in most ROIs is around frame 45. Let's calculate the difference between 45 and 25, using CTV 22 (as experimented in VIEW, see above).

Similar to the situation for movies, do the following:
In the folder "06_PROGS", you will find the file *create_tapestry_synthetic.py*.
You can run this file directly in python (command line, or calling it from a python editor), within the right environment.

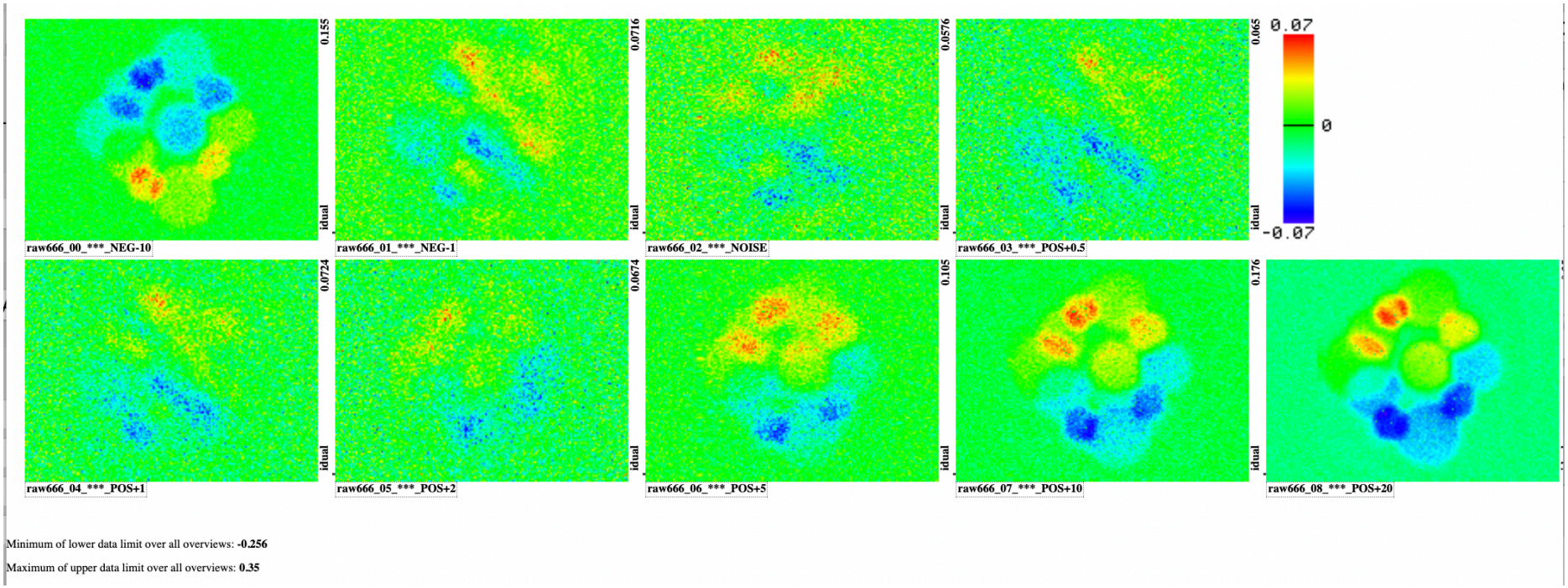
You should adapt this file to your own data. Most importantly, for each animal, you have to create a layout (the order of images in the tapestry). And you need to specify flags, in particular for the limits of false-color coding. Therefore, **each animal needs its own tapestry_animal.yml** file. In the file create_tapestry_synthetic.py, you will then list all tapestry .yml files

- Specify which animals/.yml combinations are to be evaluated (here it is "04_PROGS/tapestry_ctv22_individual.yml", and a second tapestry "04_PROGS/tapestry_ctv22_global.yml")
- You can add options for additional text to be written next to the overviews.

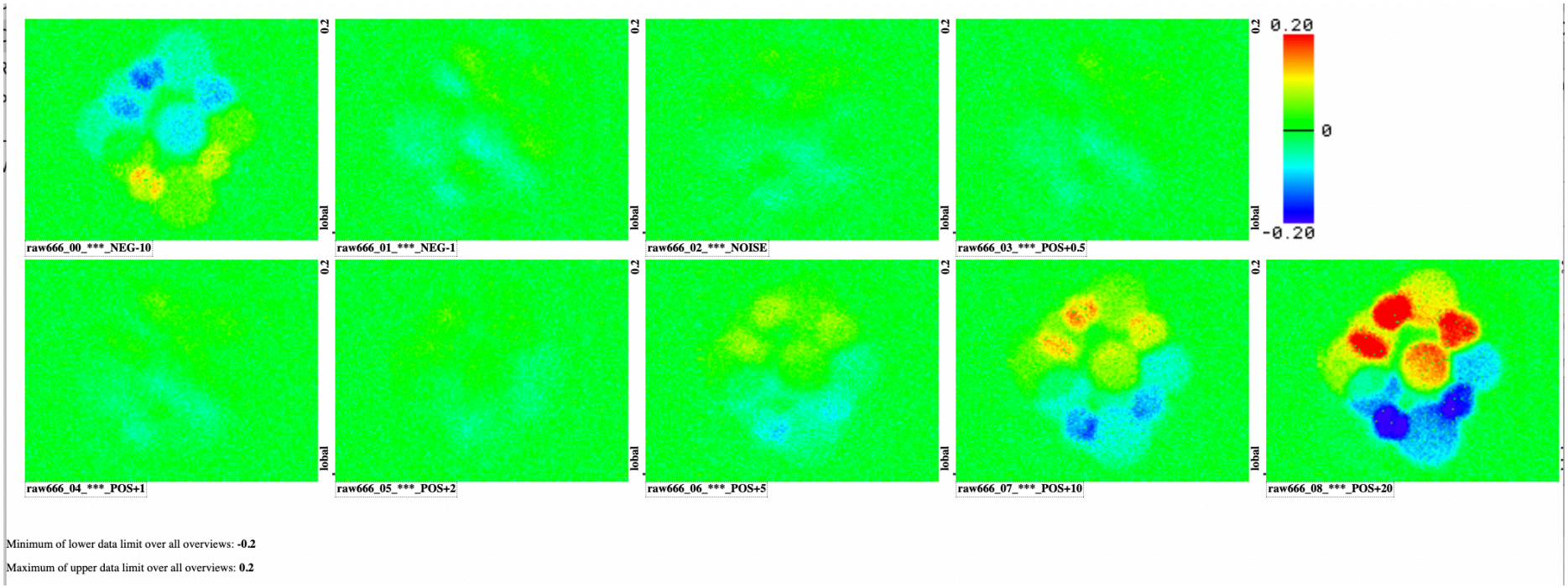
Run the python program – it will create a folder "04_OUTPUT/tapestries/". In that folder, we copied all relevant files (the python program and the .yml files used, for later reference), **a .html page for each tapestry**, and subfolders for the individual images, so that they can be used for images in publications.

Spatial filtering is controlled by Signal_FilterSpaceFlag and Signal_FilterSpaceSize (in our example: value 1).

In our example: tapestry_ctv22_individual (flag: SO_individualScale: 3) shows overviews where every overview is scaled individually. You will see consistent response patterns for POS+2 to POS+20, increasing concentration does not change the spatial pattern too much, though at lower concentrations the effect of noise is more visible. There is a negative pattern for POS-10, but all the (fictive) concentrations around 0 (NOISE) are so dominated by noise that the response pattern is not visible. Next to the individual overviews there are the scaling parameters, and at the bottom the maximum and minimum values across all overviews. Use these for deciding limits in the next tapestry.



In our example, tapestry_ctv22_global (flag: SO_individualScale: 0) has limits of -0.2 and 0.2 (in the “04_PROGS/tapestry_ctv22_global.yml” file). In the tapestry the increasing responses with increasing concentrations are clearly visible.



Batch analysis: traces

With this, we have seen the output options for overviews and movies. Now: export traces.
We have manually looked at traces in VIEW. They are based on the ROIs that were selected and saved in ILTIS.
To save traces, use the template program *06_PROGS/export_traces_synthetic.py*.
The syntax is very similar to what we described above for movies:

- You need a list of animals to be considered.
- You need to label, within the list file in column “Analyze”, every row with a number that creates subgroups (if needed)
- With the variable *Analyze_column_values_to_use* in *export_traces_synthetic.py*, define which values to consider.

As a result, you will obtain a file in a subfolder of 04_OUTPUT, subfolder with the name of the animal, name here: Synthetic_data_strip/Synthetic_data_strip.gloDatamix.csv.

This csv file fills the first columns with all information from the .list file, and the information about the ROI used in each row. In addition, there is a column with the CTV value, as calculated with the CTV chosen, here: CTV 22, taking the difference between frame 45 (average 44-46) and frame 25 (average 24-26). The right columns contain the time-traces.
These files can be loaded in a statistics program to analyze data statistically, or to use other graphics programs to display the traces.

Let’s look at the right columns:

Label	Analyze	DBB1	SampFr	Comment	setting	FrameSizeX	FrameSizeY	NumFrames	Unnamed	Line	GloInfo	CTV_22
raw666_00	1	noFile	5	raw666	setting	172	130	80	0	constructed	File: None; Pi	[-0.08892394125830348]
raw666_00	1	noFile	5	raw666	setting	172	130	80	0	constructed	File: None; Pi	[-0.09960158508750794]
raw666_00	1	noFile	5	raw666	setting	172	130	80	0	constructed	File: None; Pi	[-0.0675607276738547]
raw666_00	1	noFile	5	raw666	setting	172	130	80	0	constructed	File: None; Pi	[0.03164786508128937]
raw666_01	1	noFile	5	raw666	setting	172	130	80	1	constructed	File: None; Pi	[-0.013346298885560161]
raw666_01	1	noFile	5	raw666	setting	172	130	80	1	constructed	File: None; Pi	[-0.009066310302025031]
raw666_01	1	noFile	5	raw666	setting	172	130	80	1	constructed	File: None; Pi	[-0.021645277481029485]
raw666_01	1	noFile	5	raw666	setting	172	130	80	1	constructed	File: None; Pi	[-0.00973379597348134]
raw666_02	1	noFile	5	raw666	setting	172	130	80	2	constructed	File: None; Pi	[0.001200783406001816]
raw666_02	1	noFile	5	raw666	setting	172	130	80	2	constructed	File: None; Pi	[-0.00679452797608833]
raw666_02	1	noFile	5	raw666	setting	172	130	80	2	constructed	File: None; Pi	[-0.0007417916932436555]
raw666_02	1	noFile	5	raw666	setting	172	130	80	2	constructed	File: None; Pi	[-0.014292206251100774]

And the next columns, with the beginning of the traces:

TraceOffset	StimONms	StimLen	Odour	OConc	Cycle	GloTag	Measu	Animal	PlaceHolder	Frame0	Frame1
0	5000	2200	NEG-10	-10	200	1	1	Synthetic_data_strip	Trace begins->	0.04690345	0.050543227
0	5000	2200	NEG-10	-10	200	2	1	Synthetic_data_strip	Trace begins->	0.02358512	0.022887711
0	5000	2200	NEG-10	-10	200	3	1	Synthetic_data_strip	Trace begins->	0.05526854	0.049232533
0	5000	2200	NEG-10	-10	200	4	1	Synthetic_data_strip	Trace begins->	0.0186919	0.012240439
0	5000	2200	NEG-1	-1	200	1	2	Synthetic_data_strip	Trace begins->	0.03733792	0.03316526
0	5000	2200	NEG-1	-1	200	2	2	Synthetic_data_strip	Trace begins->	0.02926322	0.034838318
0	5000	2200	NEG-1	-1	200	3	2	Synthetic_data_strip	Trace begins->	-0.00302977	-0.004179659
0	5000	2200	NEG-1	-1	200	4	2	Synthetic_data_strip	Trace begins->	-0.01479057	-0.01877038
0	5000	2200	NOISE	0	200	1	3	Synthetic_data_strip	Trace begins->	0.01736772	0.006794039
0	5000	2200	NOISE	0	200	2	3	Synthetic_data_strip	Trace begins->	-0.03503633	-0.044773763
0	5000	2200	NOISE	0	200	3	3	Synthetic_data_strip	Trace begins->	0.0139288	0.019059641
0	5000	2200	NOISE	0	200	4	3	Synthetic_data_strip	Trace begins->	0.03576782	0.029267216

Playing with the traces file in Excel: