
Mesmerize Documentation

Release 0.1.0

Kushal Kolar

Jul 14, 2020

OVERVIEW

1	Overview	3
2	Installation	11
3	FAQs	17
4	Citation guide	19
5	Create a New Project	23
6	Project Browser	31
7	Viewer overview	45
8	Convert Meta Data	53
9	Add a Sample to the Project	55
10	Tiff file module	57
11	Batch Manager	59
12	Stimulus Mapping	63
13	ROI Manager	67
14	Caiman Motion Correction	71
15	CNMF	77
16	CNMF-E	81
17	Script Editor	87
18	Flowchart Overview	89
19	Nodes	95
20	Examples	117
21	Beeswarm	121
22	Consoles	123

23	Cross Correlation	125
24	Datapoint Tracer	127
25	Heatmap	129
26	KShape	135
27	Peak Editor	141
28	Proportions	143
29	Scatter	147
30	Simple Plot	149
31	SpaceMap	151
32	Welcome Window	155
33	Project Structure	157
34	Project Sample	159
35	Consoles	161
36	Saving plots	163
37	Plot Navbar	165
38	System Configuration	167
39	Indices and tables	169

bioRxiv: <https://doi.org/10.1101/840488>

GitHub: <https://github.com/kushalkolar/MESmerize>

Questions/Discussion: Gitter room

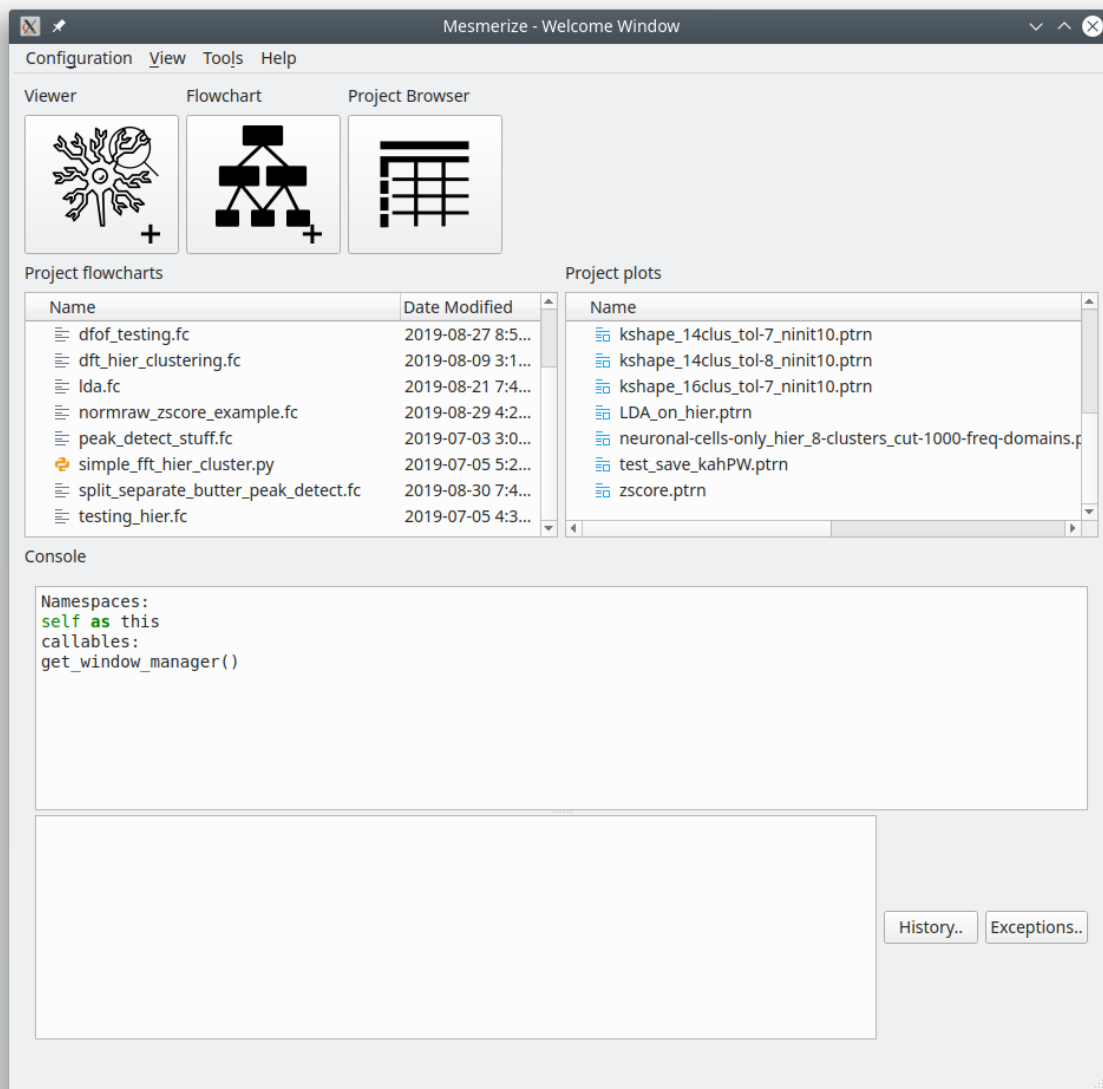
Contact: kushalkolar@alumni.ubc.ca

OVERVIEW

Mesmerize is a platform for the annotation and analysis of neuronal calcium imaging data. It encompasses the entire process of calcium imaging analysis from raw data to semi-final publication figures that are interactive, and aids in the creation of FAIR-functionally linked datasets. It is applicable for a broad range of experiments and is intended to be used by users with and without a programming background.

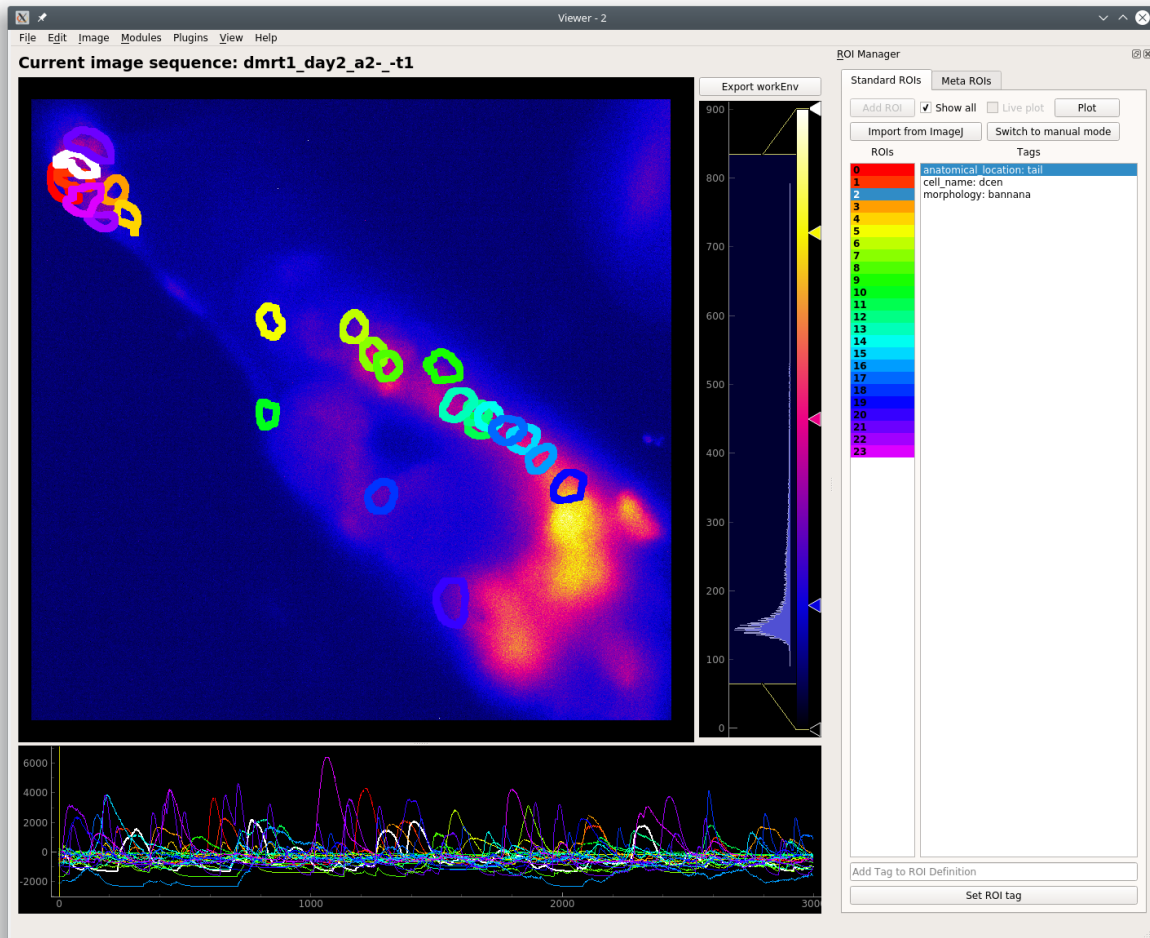
1.1 Welcome Window

Share your analysis pipelines and figures along with your publication



1.2 The Viewer

Explore image sequences, and use various modules for pre-processing and signal-extraction. Annotate regions of interest with any relevant information. Map stimuli/behavior periods.



1.3 CalmAn modules

Mesmerize contains front-end GUI modules for the CalmAn library. This makes it very easy for users without a programming background to use the library.

CalmAn Elastic Motion Correction

CalmAn Motion Correction

Rigid correction

Output bit depth: Do not convert

max shifts X (pixels): 0

max shifts Y (pixels): 0

iterations for rigid: 1

If do only rigid correction add here

Elastic correction

max deviation from rigid: 3

strides (pixels): 0



☐

overlaps (pixels): 0

☐

upsample grid: 4

CNMFE

CNMF-E



Input: Current Work Environment

Inspect Correlation and PNR

gaussian width of a 2D gaussian kernel, which approximates a neuron 3 times less than the average diamters of a neuron (pixels)

gSig: 14 Must be an even number

Stop here: Enter name Add to batch

CNMF-E

Minimum correlation of peak

min_corr: 0.98

Minimum peak to noise ratio

min_pnr: 10

Adaptive way to set threshold on the transient size

min_SNR: 1

Threshold on space consistency
If lower more components will be accepted, potentially with worse quality

r_values_min: 0.70

Average decay time of calcium spikes (seconds)

decay_time: 10

Half size of patch

rf: 50

Overlap of patches (at least 4 times the size of a neuron/cell)

overlap: 30

Global number of background components

gnb: 10

Background components per patch

nb_patch: 10

Number of neurons/cell per patch

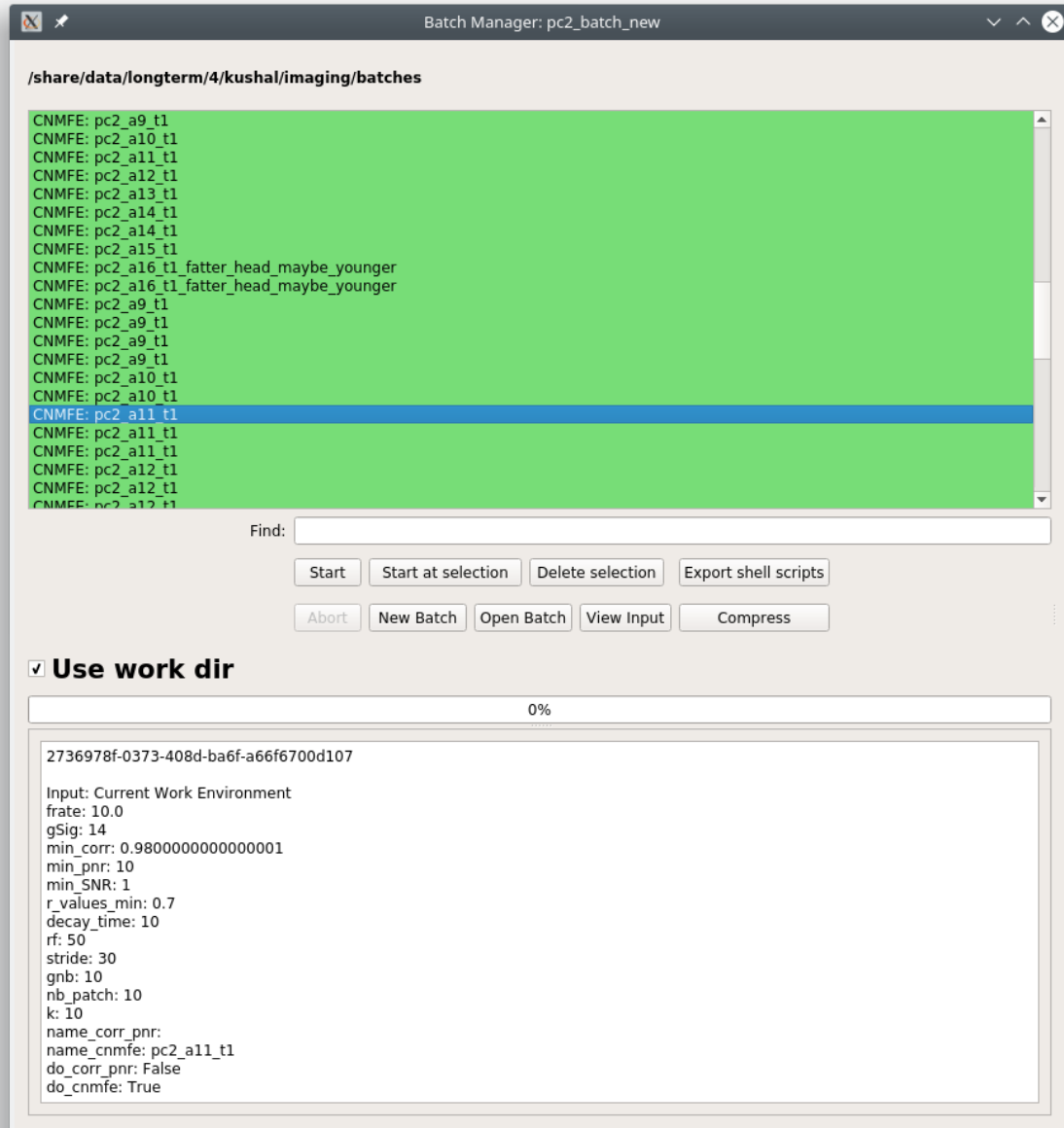
k: 10

Perform CNMF-E: Enter name Add to batch

Export Parameters
Import Parameters

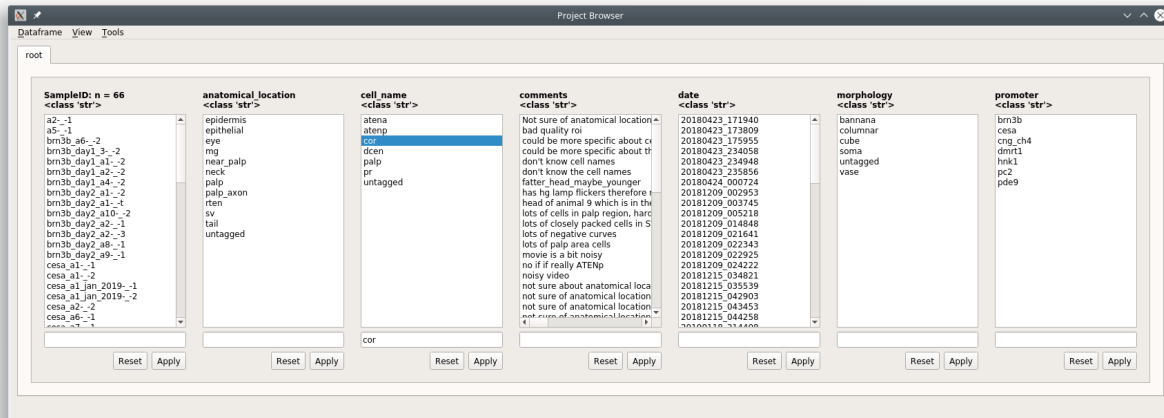
1.4 Batch Manager

Computationally intense procedures performed can be organized with the Mesmerize Batch Manager.



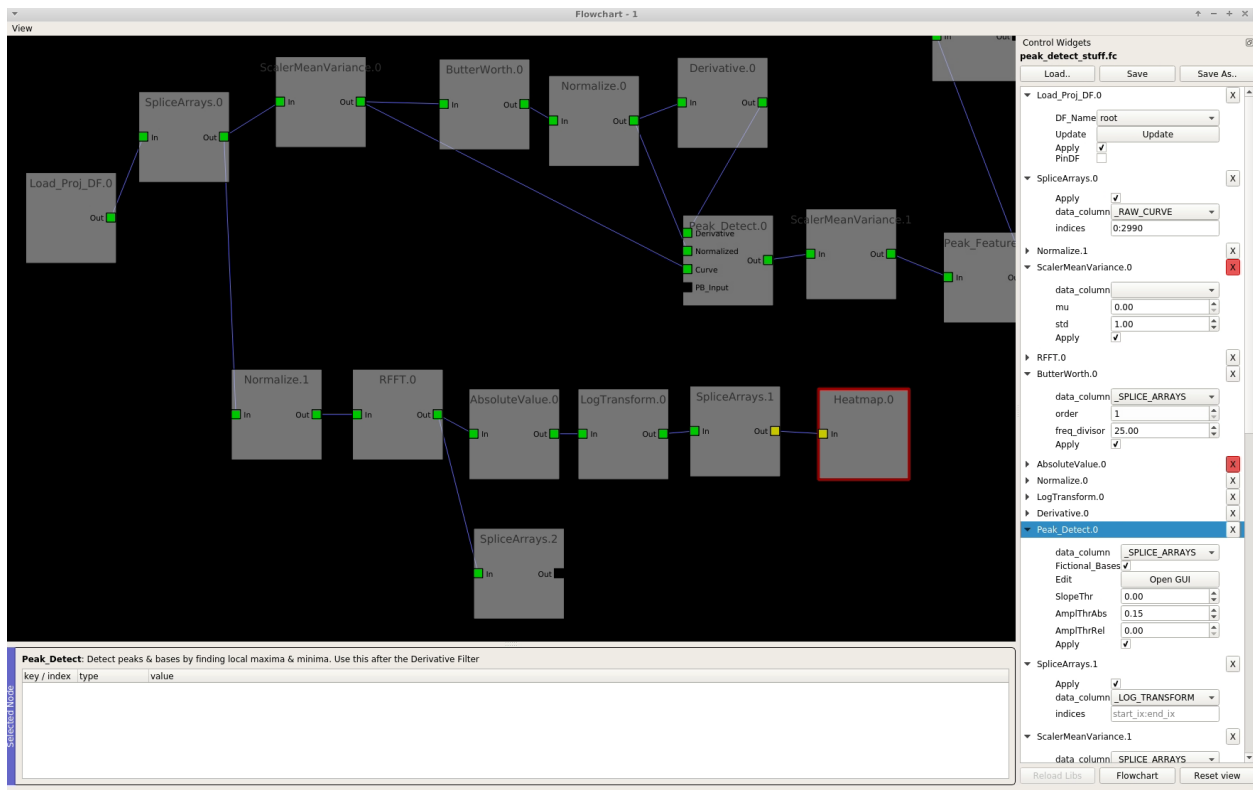
1.5 Project Organization

Explore project data and create experimental groups.



1.6 Data analysis - pyqtgraph programmable flowcharts.

Build your own analysis pipelines using flowcharts.



1.7 Interactive Plots

Create shareable interactive plots where the analysis history of every datapoint is traceable. Explore information associated with a datapoint, such as the spatial localization of its parent ROI and raw data.

Interactive Heatmaps

Interactive Cross-correlation analysis

Other types of plots: Beeswarm, Violins, KShape, Proportions, Scatter

INSTALLATION

We currently support Linux & Mac OS X. On Windows it is tricky to set up an environment that works reliably with compilers for building some dependencies. From our experience it's much easier to just install Linux and use the Snap.

2.1 Linux

The easiest way to get Mesmerize is through the Snap Store. You can also install from the GitHub repo.

2.1.1 Snap

Command line snap installation:

```
sudo snap install mesmerize
```

After installation simply run `mesmerize` in the terminal and the application will launch in ~10-30 seconds. Make sure your `PYTHONPATH` environment variable is empty otherwise it might conflict with the snap:

```
export PYTHONPATH=
```

You can also open an ipython console in the snap environment:

```
mesmerize.ipython
```

Note: You might get the following warnings when you launch the snap, this is normal. Just be patient and wait a few minutes:

```
warnings.warn('%s%s' % (e, warn))
/snap/mesmerize/x1/lib/python3.6/site-packages/matplotlib/font_manager.py:281: UserWarning: Matplotlib is
building the font cache using fc-list. This may take a moment.
'Matplotlib is building the font cache using fc-list. '
Bokeh could not be loaded. Either it is not installed or you are not running within a notebook
Loading, please wait...
Qt: Session management error: Authentication Rejected, reason : None of the authentication protocols specified are
supported and host-based authentication failed
```

Requirements

Make sure you have `snapd` installed, which is required for running snap apps. Ubuntu 16.04 and later usually come pre-installed with `snapd`.

You should be able to install `snapd` through `apt` for most Debian based distros:

```
sudo apt update
sudo apt install snapd
```

Installing `snapd` on Fedora:

```
sudo dnf install snapd
```

To install `snapd` on other distros please see: <https://docs.snapcraft.io/installing-snapd>

If you have trouble installing Mesmerize via snap you might need to install `core18` first:

```
sudo snap install core18
```

Limitations

The snap installation has several limitations, most importantly you will not be able to access arbitrary filesystems. If you need this you will have to install directly from the repo (see From GitHub). If you are able to mount your external filesystem in `/media` (or wherever your distro places removable media) then you should be able to access these filesystems if you do the following:

```
sudo snap connect mesmerize:removable-media
```

Alternatively you can install the snap in devmode (gives that snap broad access to the system):

```
sudo snap install mesmerize --devmode
```

Warning: Analysis graphs do not work in the snap version at the moment.

2.1.2 From GitHub

First, make sure you have compilers & python.

For Debian & Ubuntu based distros

Get build tools and other things:

```
sudo apt-get install build-essential
```

For other distros look for the equivalent meta-package that contains gcc, glibc, etc.

If you don't have python3.6:

```
sudo apt-get install python3.6
```


For other distros lookup how to install python3.6 through their package manager.

Install dependencies:

```
sudo apt-get install qt5-default tcl graphviz git
```

For other distros these packages probably have the same or similar names.

Create a virtual environment & install Mesmerize

1. Create a virtual environment:

```
# Choose a path to house the virtual environment
python3.6 -m venv /path/to/venv
```

2. Activate the virtual environment:

```
source /path/to/venv/bin/activate
```

3. Clone the repo:

```
git clone https://github.com/kushalkolar/MESmerize.git
```

4. cd & switch to the snap branch:

```
cd MESmerize
git checkout snap
```

5. Install some build dependencies:

```
pip install Cython numpy python-dateutil
```

6. Install remaining dependencies:

```
pip install -r requirements.txt
```

7. Build some things:

```
python setup.py build_ext -i
```

8. Add to PYTHONPATH environment variable. You will always need to add the path to MESmerize to the PYTHONPATH environment variable before launching.:

```
export PYTHONPATH=$PWD:$PYTHONPATH
```

9. Launch:

```
python ./mesmerize
```

2.2 Mac OSX

This requires Anaconda and will install Mesmerize in an Anaconda environment. Tested on macOS Catalina 10.15.1

Download Anaconda for Python 3: <https://www.anaconda.com/distribution/>

First make sure you have xcode:

```
xcode-select --install
```

This might take a while.

Create an environment & install Mesmerize

1. Create a new environment using python 3.6:

```
conda create --name mesmerize python=3.6
```

2. Enter the environment:

```
source activate mesmerize
```

3. Install cython, numpy and pandas:

```
conda install cython numpy pandas
```

4. Clone the mesmerize repo and enter it:

```
git clone https://github.com/kushalkolar/MESmerize.git
cd MESmerize
```

5. Checkout the snap branch:

```
git checkout snap
```

6. Install more dependencies:

```
pip install -r requirements.txt
```

7. Install Mesmerize:

```
CFLAGS='-stdlib=libc++' python setup.py build_ext -i
```

Launching Mesmerize

1. Export the path to the MESmerize repo directory:

```
export PYTHONPATH=<path_to_MESmerize_dir>
```

2. Launch. It may take a few minutes the first time:

```
python <path_to_MESmerize_dir>/mesmerize
```

You might get a matplotlib error, if so execute the following which appends the default matplotlib backend-option. Note that this will probably affect matplotlib in all your environments:

```
echo "backend: qt5" >> ~/.matplotlib/matplotlibrc
```

2.3 Customized

2.4 Troubleshooting

2.4.1 Qt version

2.5 PyPI

3.1 ROIs

1. **Can I delete an ROI?**

- See *ROI Manager guide*

2. **I don't want to delete ROIs but I want to mark them for exclusion in further analysis, how can I do this?**

- You can do this by creating an ROI type category. See [<link here> Add New ROI Type Later](#) which uses this as an example. You can also create this ROI Type category when you create a New Project, not necessarily when you already have a project as the example uses.

3. **Can I tag more than one piece of information to each ROI?**

- Yes, add as many ROI Type categories as you want in the Project Configuration.

See also:

Project Configuration

4. **I already have a Mesmerize project with many Samples in it. Can I add a new ROI Type category?**

- Yes, just add it to your *Project Configuration*

5. **Can some samples in my project have ROIs that originate from CNMF(E) and others that are manually drawn?**

- Yes, but be aware that you must separate the CNMF(E) and manual data in downstream analysis if you use flowchart nodes that only work with CNMF(E) data, such as *NormRaw* and *DetrendDFoF* `<node_DetrendDFoF>`.

3.2 CNMFE

1. **I have ROIs that clearly encompass multiple cells instead of just one**

- Increase *min_coor*
- Might help to reduce *gSig* as well

2. **I have too many bad ROIs around random regions that are clearly noise**

- Increase *min_pnr*

3. **Min_PNR image is completely blue and void of any signals**

- Increase *gSig*

4. **Vmin slider is stuck in Inspect Correlation & PNR GUI.**

- Close and reopen it. This is a matplotlib issue, not something I can fix.

3.3 Caiman Motion Correction

1. **I have video tearing**

- Try increasing *upsample grid*
- It's possible that the movement is too severe to be motion corrected. When the movement is so severe that the information do not exist, it is impossible to motion correct it.

2. **My animal is growing**

- This is growth, not motion. Unfortunately cannot be corrected for. If you have an idea for a technique I can try it out.

3. **The output actually has more motion, it has created false motion.**

- **Try these things:**
 - Reduce *Strides & Overlaps* by ~25%
 - Reduce *max shifts X & Y* by ~25%
 - Reduce *max deviation from rigid* by ~25%

3.4 Project Organization

1. **Can I modify a sample?**

- Yes. Double click the Sample ID in the Project Browser to open it in a viewer. You can then make any modifications you want and then go to File -> Add to Project and select the “Save Changes (overwrite)” option at the bottom. If you have not changed the image sequence itself you can uncheck “Overwrite image data”.

2. **Can I change the SampleID?**

- No this is fundamentally impossible.
- A work-around is to open that Sample in the viewer (double click it in the project browser), make any modifications if necessary, then go to File -> Add to Project, enter the the information for this sample and a new Animal ID (and Trial ID if wanted), and then select the option “Add to Project Dataframe” at the bottom and click Proceed. This will now add a new Sample to the project with this Sample ID. You can then delete the previous Sample.

3. **Can I add a new Custom Column, ROI Column, or Stimulus Column to my project when I already have samples in my pr**

- Yes, just modify your *Project Configuration*. In the Welcome Window go to Configure -> Project Configuration. Add anything that you want, and then click “Save and Apply”. **It's best to immediately restart Mesmerize whenever you change your project configuration.**
- If you are adding a new Custom Column you can enter a “Dataframe replace value”. This will allow you to set a value for all existing Samples in your project for this new column.
- If you do not set a Dataframe replace value it will label all existing as “untagged”

CITATION GUIDE

Mesmerize provides interfaces to many great tools that were created by other developers. Please cite the papers for the following Viewer Modules and analysis methods that you use in addition to citing Mesmerize. I would also suggest citing numpy, pandas, scipy, sklearn, and matplotlib.

Mesmerize relies heavily on [pyqtgraph](#) widgets. Citing [pyqtgraph](#).

4.1 Viewer

Module	Cite
<i>CNMF</i>	<p>Giovannucci A., Friedrich J., Gunn P., Kalfon J., Brown, B., Koay S.A., Taxidis J., Najafi F., Gauthier J.L., Zhou P., Baljit, K.S., Tank D.W., Chklovskii D.B., Pnevmatikakis E.A. (2019). CaImAn: An open source tool for scalable Calcium Imaging data Analysis. eLife 8, e38173. https://elifesciences.org/articles/38173</p> <p>Pnevmatikakis, E.A., Soudry, D., Gao, Y., Machado, T., Merel, J., ... & Paninski, L. (2016). Simultaneous denoising, deconvolution, and demixing of calcium imaging data. Neuron 89(2):285-299. http://dx.doi.org/10.1016/j.neuron.2015.11.037</p> <p>Pnevmatikakis, E.A., Gao, Y., Soudry, D., Pfau, D., Lacefield, C., ... & Paninski, L. (2014). A structured matrix factorization framework for large scale calcium imaging data analysis. arXiv preprint arXiv:1409.2903. http://arxiv.org/abs/1409.2903</p>
<i>CNMF-E</i>	<p>In addition to the above CNMF papers:</p> <p>Zhou, P., Resendez, S. L., Rodriguez-Romaguera, J., Jimenez, J. C., Neufeld, S. Q., Giovannucci, A., ... Paninski, L. (2018). Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data. ELife, 7. doi: https://doi.org/10.7554/eLife.28728.001</p>
<i>Caiman Motion Correction</i>	<p>Giovannucci A., Friedrich J., Gunn P., Kalfon J., Brown, B., Koay S.A., Taxidis J., Najafi F., Gauthier J.L., Zhou P., Baljit, K.S., Tank D.W., Chklovskii D.B., Pnevmatikakis E.A. (2019). CaImAn: An open source tool for scalable Calcium Imaging data Analysis. eLife 8, e38173. https://elifesciences.org/articles/38173</p> <p>Pnevmatikakis, E.A., and Giovannucci A. (2017). NoRMCorre: An online algorithm for piecewise rigid motion correction of calcium imaging data. Journal of Neuroscience Methods, 291:83-92. https://doi.org/10.1016/j.jneumeth.2017.07.031</p>

4.2 Nodes/Analysis

Node/Method	Cite
<i>k-Shape clustering</i>	<p>Paparrizos, J., & Gravano, L. (2016). k-Shape. ACM SIGMOD Record, 45(1), 69–76. doi: http://dx.doi.org/10.1145/2723372.2737793</p> <p>Romain Tavenard, Johann Faouzi, Gilles Vandewiele and Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Ruwurm, Kushal Kolar, & Eli Woods. (2017). Tslearn, A Machine Learning Toolkit for Time Series Data. Journal of Machine Learning Research, (118):16, 2020. http://jmlr.org/papers/v21/20-091.html</p>
<i>Cross-correlation</i>	<p>Romain Tavenard, Johann Faouzi, Gilles Vandewiele and Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Ruwurm, Kushal Kolar, & Eli Woods. (2017). Tslearn, A Machine Learning Toolkit for Time Series Data. Journal of Machine Learning Research, (118):16, 2020. http://jmlr.org/papers/v21/20-091.html</p>
<i>TVDiff Node</i>	<p>Rick Chartrand, “Numerical Differentiation of Noisy, Nonsmooth Data,” ISRN Applied Mathematics, vol. 2011, Article ID 164564, 11 pages, 2011. https://doi.org/10.5402/2011/164564.</p>

4.3 Scientific Libraries

Library	
numpy	Van Der Walt, S., Colbert, S. C. & Varoquaux, G. The NumPy array: A structure for efficient numerical computation. Comput. Sci. Eng. (2011) doi:10.1109/MCSE.2011.37
pandas	McKinney, W. Data Structures for Statistical Computing in Python. Proc. 9th Python Sci. Conf. (2010)
scipy	Virtanen, P., Gommers, R., Oliphant, T.E. et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods (2020). https://doi.org/10.1038/s41592-019-0686-2
sklearn	Pedregosa, F. et al. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. (2011)
matplotlib	Hunter, J. D. Matplotlib: A 2D graphics environment. Comput. Sci. Eng. (2007)
pyqtgraph	http://www.pyqtgraph.org/

CREATE A NEW PROJECT

5.1 Biological Questions

Before you create a new Mesmerize Project you must thoroughly think about the biological questions that you are interested in. Here are some thoughts to help you:

- The effects of different types of temporary stimulation? Such as poking or odors?
 - Are you interested in neuronal activity during specific behavioral periods?
 - Differences in calcium dynamics between different anatomical regions?
 - Chemogenetic experiments using transgenes to express DDREADs.
 - Non-temporary effects of drugs (for example, if the animal is bathed in drug for longer recordings).
 - For example, if you are inducing seizures with PTZ, where you are interested in the differences between a control recording of 5 minutes and subsequent 5 minute recordings where the animal is bathed in PTZ (or whatever duration you determine is biologically meaningful). You could also be interested in a recovery stage to see what happens to the calcium dynamics when you “perfuse-back” the liquid medium (such as seawater, steinberg’s solution etc.) without the drug.
 - Differences in calcium dynamics between different stages during development
 - Differences in calcium dynamics between different cell types using GCaMP driven by specific promoters.
-

5.2 New Project

To create a new project click **New Project** in the *Welcome Window*. You will then be prompted to choose a location and a name for the project. This will create a directory with the chosen name in the location you previously selected.

5.3 Project Configuration

After setting a project name you must configure it. This is where your biological questions of interest are important. You can change your project configuration later, but it is most time efficient if you enter all your categories of interest now.

Columns

Include in Project Browser View

SampleID
date
comments

Exclude in Project Browser View

CurvePath
ImgInfoPath
ImgPath
MaxProjPath
ROI_State
uuid_curve
misc

ROI Type Columns

Enter New ROI Column Name Add

Stimulus Type Columns

Custom Columns

Add new custom column

Column name

Choose data type:

☐ standard text (str)

☐ whole numbers (np.int64)

☐ decimal numbers (np.float64)

☐ boolean (True/False values)

Dataframe replacement value

Add

Enter New Stim Column Name Add

Close Save and apply

Warning: Restart Mesmerize whenever you change the project configuration.

Note: If you have Samples in your project and you change the project configuration at a later date to add new columns, all existing rows in your project DataFrame are labelled as “untagged” for the new columns.

See also:

Add To Project Guide to understand how the project configuration relates to the addition of data samples to your project

5.3.1 Categorical Data Columns

Mesmerize allows you to create three main different types of categorical data columns (for the project `DataFrame`), and an unlimited number of each type. These categorical data columns allow you to group your data during analysis, and therefore perform comparisons between experimental groups. In essence, these categorical data columns form scaffold with which you can create your experimental groups during analysis.

Note: You can change the project configuration at any point in the future by adding new columns or changing the visible/hidden columns.

ROI Type Columns

Create ROI-bound *categories* with which you want to group your data. Enter the desired name for the category and click **Add**. Here are some examples:

- If you are interested in calcium dynamics between different anatomical regions, you create a column named `anatomical_region`.
- You want to define defined notochord cell identities on a anterior-posterior axis, defined as “cell_1”, “cell_2”, ... “cell_n”. You can create an ROI Type Column named `notochord_cell_id`.

The screenshot shows the 'Columns' configuration window in Mesmerize. It is divided into several sections for managing project columns:

- Include in Project Browser View:** A list containing SampleID, date, comments, anatomical_location, and odor.
- Exclude in Project Browser View:** A list containing CurvePath, ImgInfoPath, ImgPath, MaxProjPath, ROI_State, uuid_curve, and misc.
- ROI Type Columns:** A list containing anatomical_location, with an 'Add' button below it.
- Stimulus Type Columns:** A list containing odor, with an 'Add' button below it.
- Custom Columns:** An empty list for user-defined columns.
- Add new custom column:** A section for creating new columns, featuring a text input (currently 'gene'), a 'Choose data type' section with radio buttons for 'standard text (str)', 'whole numbers (np.int64)', 'decimal numbers (np.float64)', and 'boolean (True/False values)', a 'Dataframe replacement value' input, and an 'Add' button.

At the bottom right of the window are 'Close' and 'Save and apply' buttons.

See also:

[ROI Manager](#) to understand how labels can be tagged onto ROIs using these categories that you have defined in the ROI Type Columns.

Stimulus Type Columns

If you're interested in mapping temporal information to your traces, such as stimuli or behavioral periods, add a "Stimulus Type column" for each type. This is only for temporary stimulation or behavioral periods that do not span the entire length of the video.

See also:

[Stimulus Mapping guide](#), to understand how stimuli can be labelled.

Custom Columns

Here you can create categories to tag any other piece of useful information to each Sample. i.e. to the entire video recording. For example:

- You are studying seizures, you perform a 5 minute recording in the medium, and then subsequent 5 minute recordings in PTZ. You can create a category called “drug_state”. When you add samples to your project you can tag drug states named “control”, “ptz_1”, “ptz_2”, “ptz_recovery_1” etc.
- This is also what you would use for chemogenetics experiments if you are recording for example without CNO for 5 minutes, and then with CNO for another 5 minutes.

Three different data types can be tagged to a category, **standard text**, **whole numbers**, and **decimal numbers**.

Warning: Data types cannot be changed later. If you are familiar with pandas you can manually change it, and the corresponding value in the project config file.

If you want to tag numerical information, such as the animal’s development stage, it can be useful to set the data type to **whole numbers**. This allows you to sort your data numerically. For example you may want to compare dynamics of all curves between stage 48 and 72.

The screenshot shows a 'Custom Columns' configuration window. It is divided into several sections:

- Include in Project Browser View:** A list box containing 'SampleID', 'date', 'comments', 'anatomical_location', 'odor', and 'gene'.
- Exclude in Project Browser View:** A list box containing 'CurvePath', 'ImgInfoPath', 'ImgPath', 'MaxProjPath', 'ROI_State', 'uuid_curve', and 'misc'.
- ROI Type Columns:** A list box containing 'anatomical_location'.
- Stimulus Type Columns:** A list box containing 'odor'.
- Custom Columns:** A list box containing 'gene'.
- Add new custom column:** A section with a text input field containing 'developmental_stage'. Below it, under 'Choose data type:', there are four radio buttons: 'standard text (str)', 'whole numbers (np.int64)' (which is selected), 'decimal numbers (np.float64)', and 'boolean (True/False values)'. Below the radio buttons is a text input field for 'Dataframe replacement value' and an 'Add' button.

At the bottom of the window, there are two buttons: 'Close' and 'Save and apply'.

If you are interested in dynamics between different cell types for which you are using specific GCaMP promoters, you can create a custom column called `promoter` or `cell_type` and select **standard text** as the data type.

The screenshot shows the 'Columns' dialog box with the following sections:

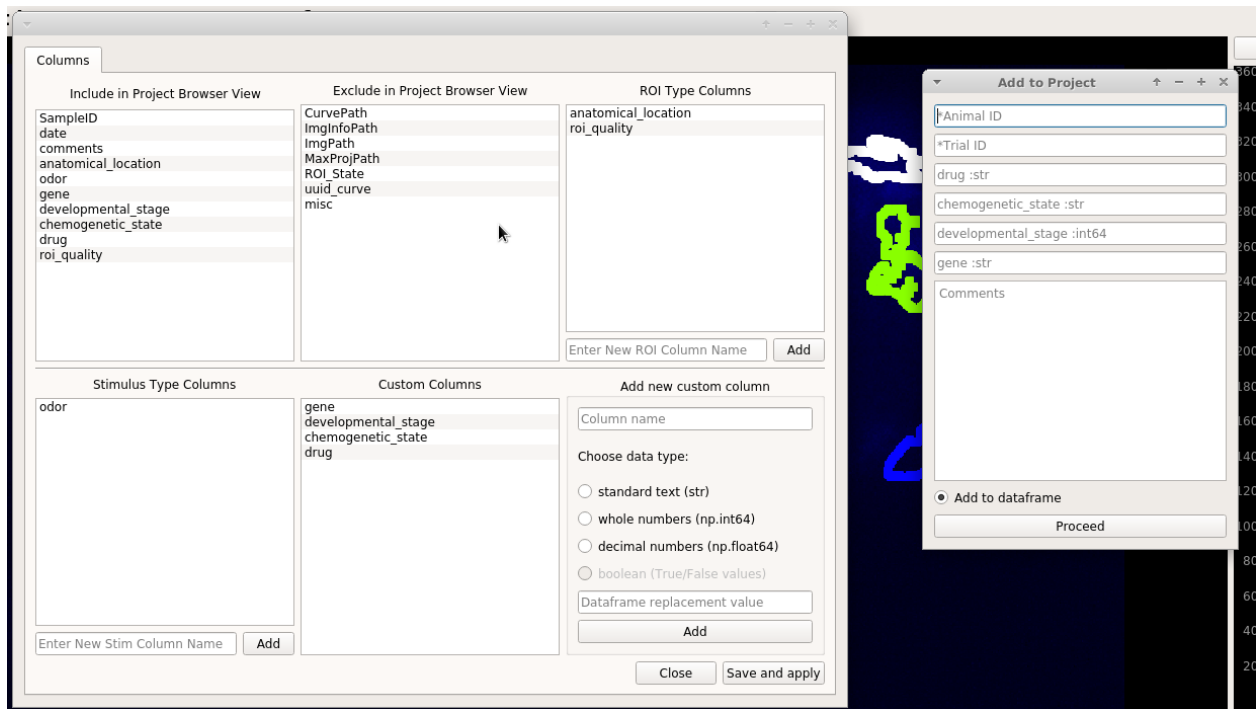
- Include in Project Browser View:** SampleID, date, comments, anatomical_location, odor.
- Exclude in Project Browser View:** CurvePath, ImgInfoPath, ImgPath, MaxProjPath, ROI_State, uuid_curve, misc.
- ROI Type Columns:** anatomical_location. Below this is a text input 'Enter New ROI Column Name' and an 'Add' button.
- Stimulus Type Columns:** odor. Below this is a text input 'Enter New Stim Column Name' and an 'Add' button.
- Custom Columns:** (Empty list)
- Add new custom column:**
 - Text input: gene
 - Choose data type:
 - ☒ standard text (str)
 - ☐ whole numbers (np.int64)
 - ☐ decimal numbers (np.float64)
 - ☐ boolean (True/False values)
 - Dataframe replacement value: (Empty text input)
 - Add button

At the bottom right are 'Close' and 'Save and apply' buttons.

When you add samples to your project from the viewer, you will be prompted to enter information that is directly based on the Custom Columns that you create here.

See also:

[Add to Project guide](#)



5.3.2 Visible / Hidden in Project Browser

You can drag and drop items (column names) between these two lists to set which ones are visible in the Project Browser. This is just to avoid clutter.

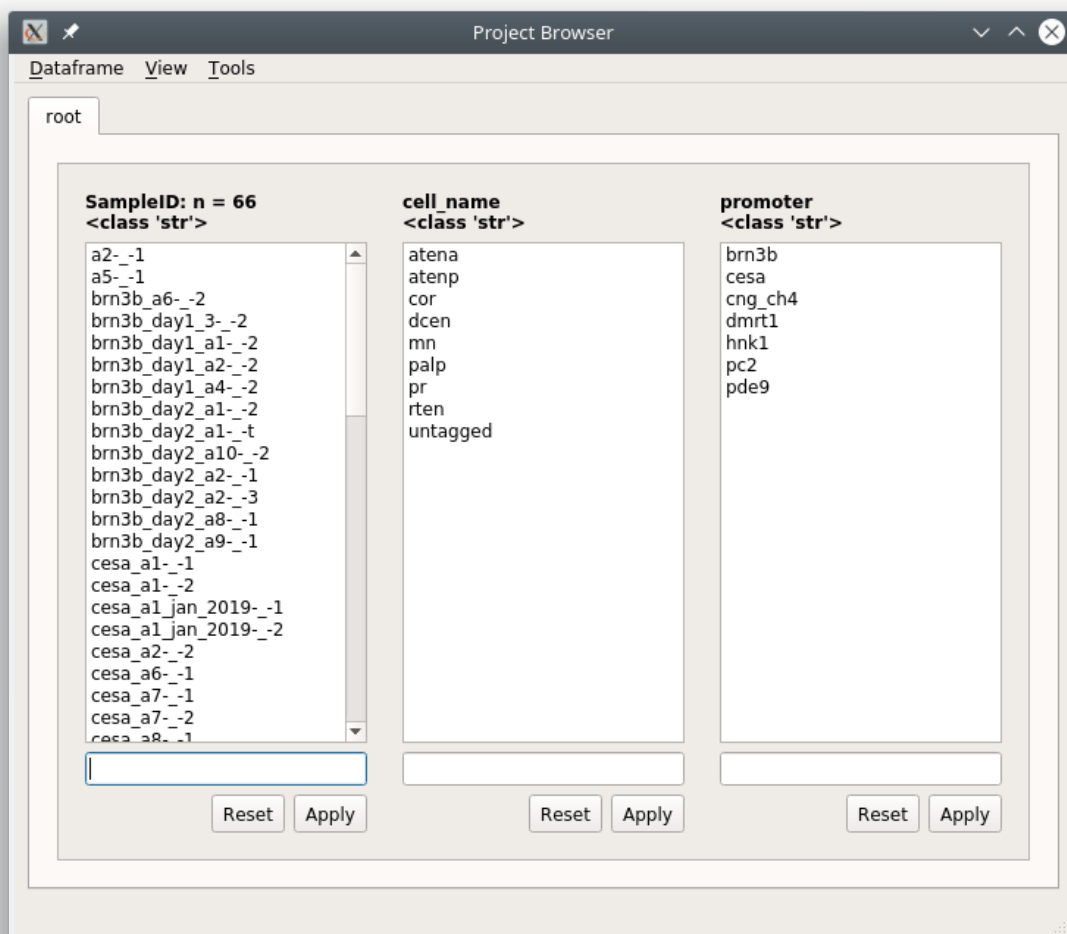
See also:

Project Browser

PROJECT BROWSER

Browse, edit and sort the project DataFrame

You can open the Project Browser from the *Welcome Window* after you have opened a project.



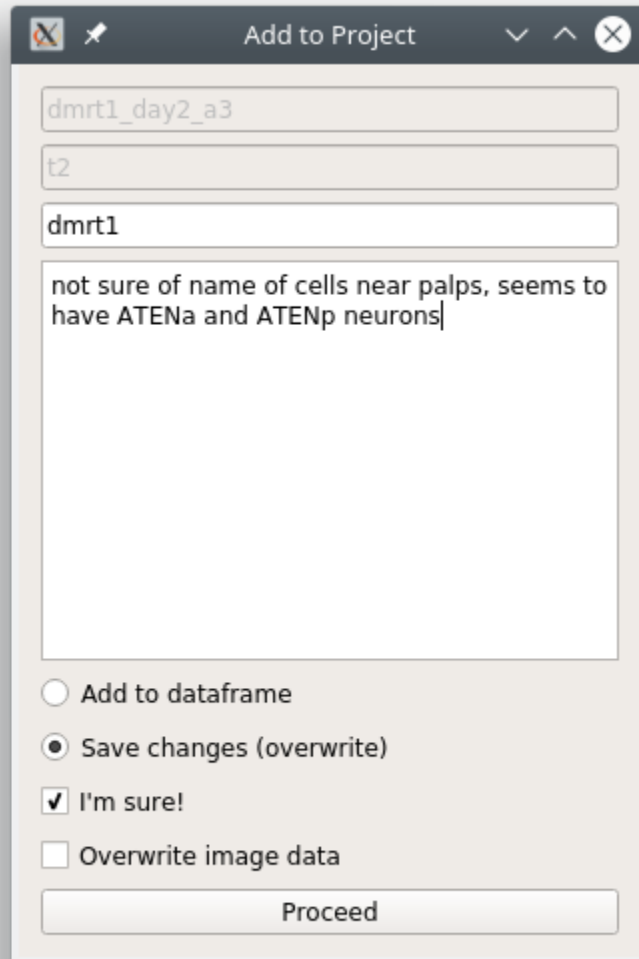
The columns that are visible in the Project Browser Window correspond to the *Project Configuration*. For each column you will see a list which is a set of unique elements from that column in the project DataFrame.

Functions

6.1 Open Sample

Double-click on a Sample in the *SampleID* column to open it in the *Viewer*.

In the viewer you can make changes and then save it by going to File -> Add to Project. You will see a “Save changes (overwrite)” option which will overwrite the data for this project Sample with the current data in the viewer work environment. If you have not changed the image sequence data you can uncheck the “Overwrite image data” checkbox, useful if your image sequences are large.



Note: You can make any changes that you want to the Sample. This may include things such as changing or adding new tags to ROIs, changing stimulus maps, tagging a new custom column etc.

Warning: You can never change the AnimalID or TrialID (i.e. SampleID) since these are partially used as unique identifiers. A workaround is described in the [FAQ for Project Organization](#).

6.2 Filter

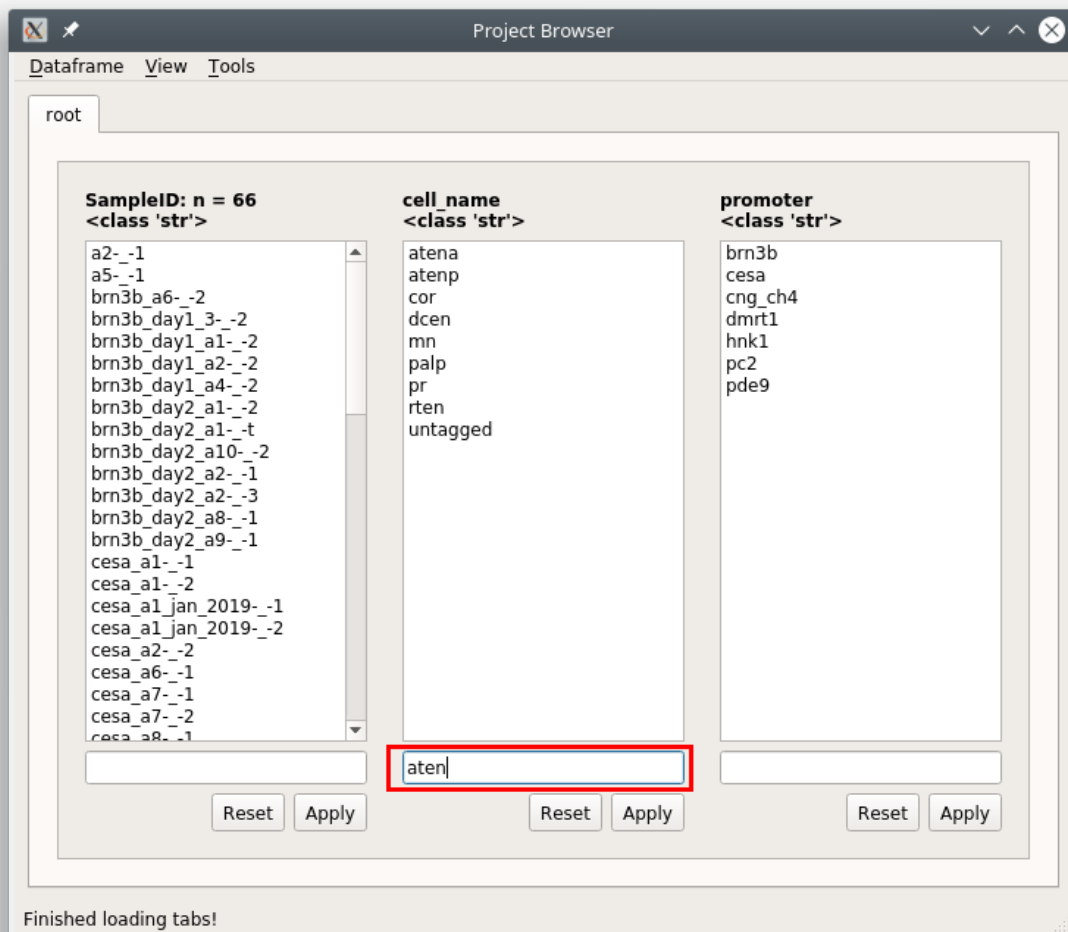
You can sort your Project DataFrame into different groups (such as experimental groups) using text and numerical filters. Type a filter into the text entries that are below the list of elements for a column. You can also click on one or many elements in a column to set those elements as the filters.

If you filter out of the root tab, it will always create a new tab with a name of your choice. If you filter out of any other tab it will apply the filter in-place unless you right click on the “Apply” button and choose “Apply in new tab”

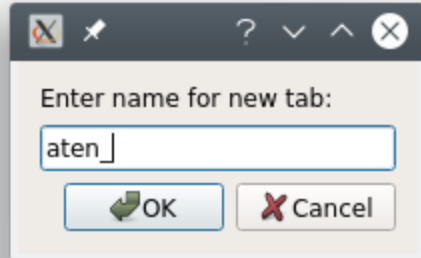
6.2.1 Text filters

6.2.2 Partial match

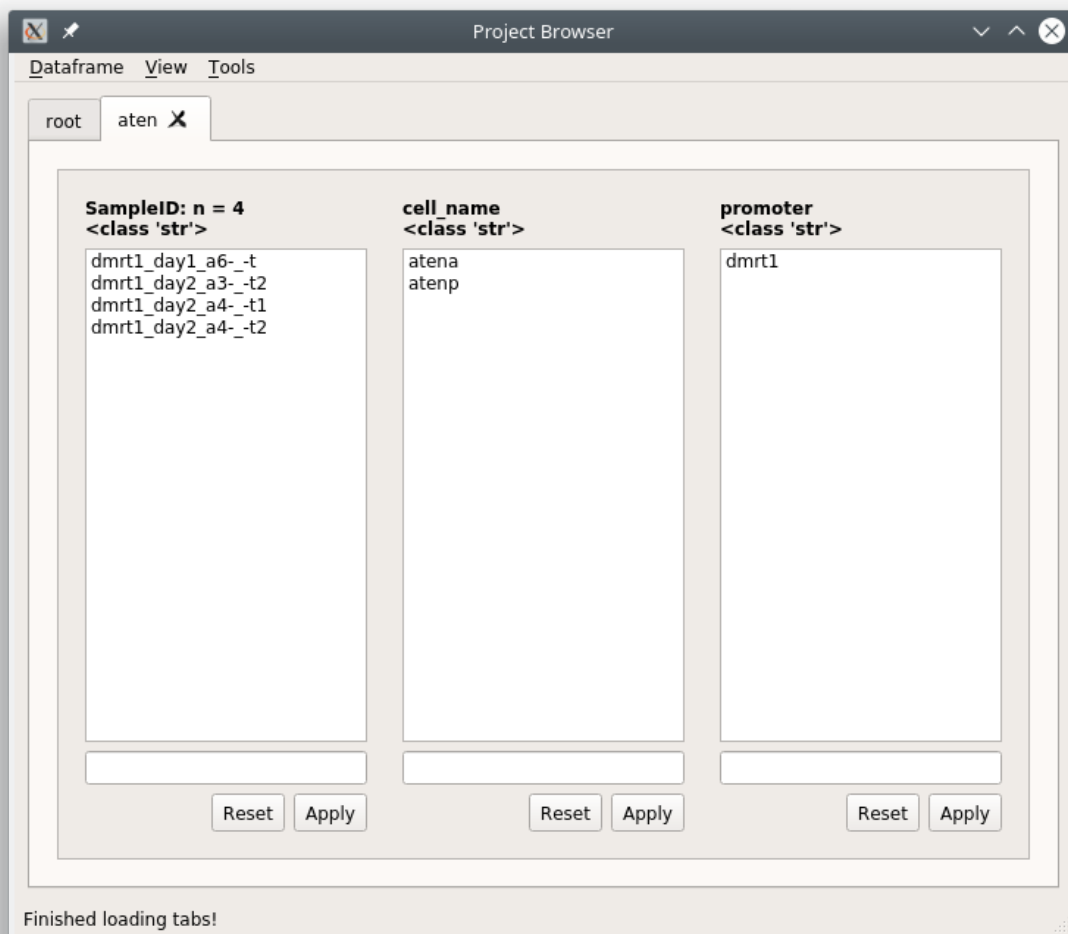
To filter out a group based on partial text matches just enter the text into the filter text entry below the column(s) of interest and click “Apply”



Since this is filtering out of the root tab, you will be prompted to give a name for a new tab that will be created based on the filter you have entered.



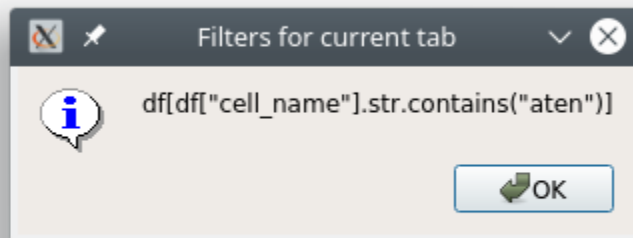
The result is a DataFrame containing all rows where the cell_name contains 'aten'



If you go to View -> Current dataframe you can see the whole dataframe.

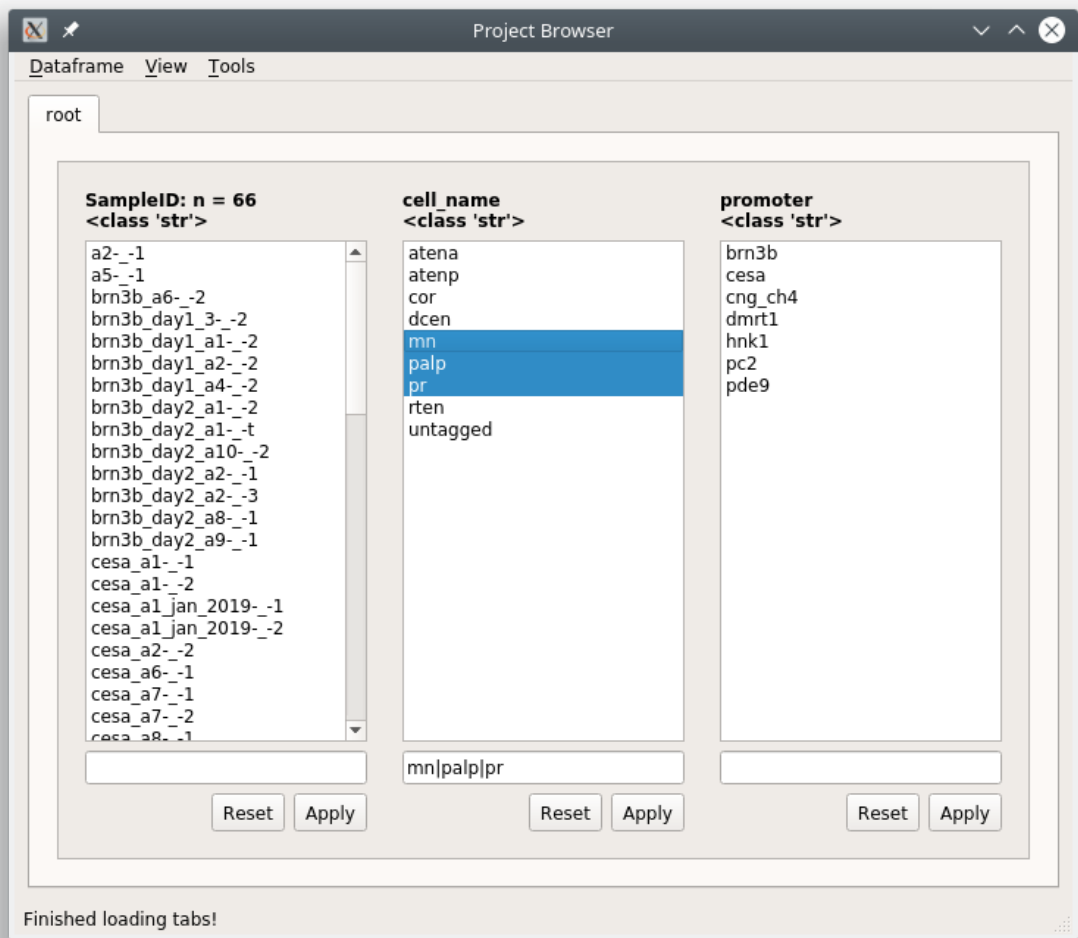
Index	CurvePath	ImgInfoPath	ImgPath	ImgUUID	ROI_State	SampleID	anatomical_locati	cell_name	comments	date	misc	morphology	promoter	stimulus_name	uuid_curve
2	curves/dnr...	images/dnr...	images/dnr...	1b2b1436-0...	{'roi_xs': '...	dnrt1_day2...	sv	atena	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	a58b74ab-d...
3	curves/dnr...	images/dnr...	images/dnr...	1b2b1436-0...	{'roi_xs': '...	dnrt1_day2...	sv	atena	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	fd2a224f-2...
4	curves/dnr...	images/dnr...	images/dnr...	1b2b1436-0...	{'roi_xs': '...	dnrt1_day2...	sv	atena	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	8e861cf8-7...
5	curves/dnr...	images/dnr...	images/dnr...	1b2b1436-0...	{'roi_xs': '...	dnrt1_day2...	sv	atena	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	7b389637-6...
6	curves/dnr...	images/dnr...	images/dnr...	1b2b1436-0...	{'roi_xs': '...	dnrt1_day2...	sv	atena	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	a6c90334-5...
7	curves/dnr...	images/dnr...	images/dnr...	1b2b1436-0...	{'roi_xs': '...	dnrt1_day2...	sv	atenp	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	da3153dd-7...
8	curves/dnr...	images/dnr...	images/dnr...	1b2b1436-0...	{'roi_xs': '...	dnrt1_day2...	sv	atenp	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	c7016d72-9...
9	curves/dnr...	images/dnr...	images/dnr...	1b2b1436-0...	{'roi_xs': '...	dnrt1_day2...	sv	atenp	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	479e6db2-5...
10	curves/dnr...	images/dnr...	images/dnr...	1b2b1436-0...	{'roi_xs': '...	dnrt1_day2...	sv	atenp	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	9dca9b52-6...
165	curves/dnr...	images/dnr...	images/dnr...	f0cbb84e-9...	{'roi_xs': '...	dnrt1_day2...	mg	atenp	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	415a955d-1...
166	curves/dnr...	images/dnr...	images/dnr...	73d208f4-1...	{'roi_xs': '...	dnrt1_day2...	mg	atenp	no if if r...	20190427_0...	()	untagged	dnrt1	['untagged...	5ef77df8-c...
205	curves/dnr...	images/dnr...	images/dnr...	b5f6e926-b...	{'roi_xs': '...	dnrt1_day1...	sv	atena	not sure o...	20190425_1...	()	untagged	dnrt1	['untagged...	4c0f0725-e...

To see how the filter translates to pandas commands go to View -> Current tab filter history

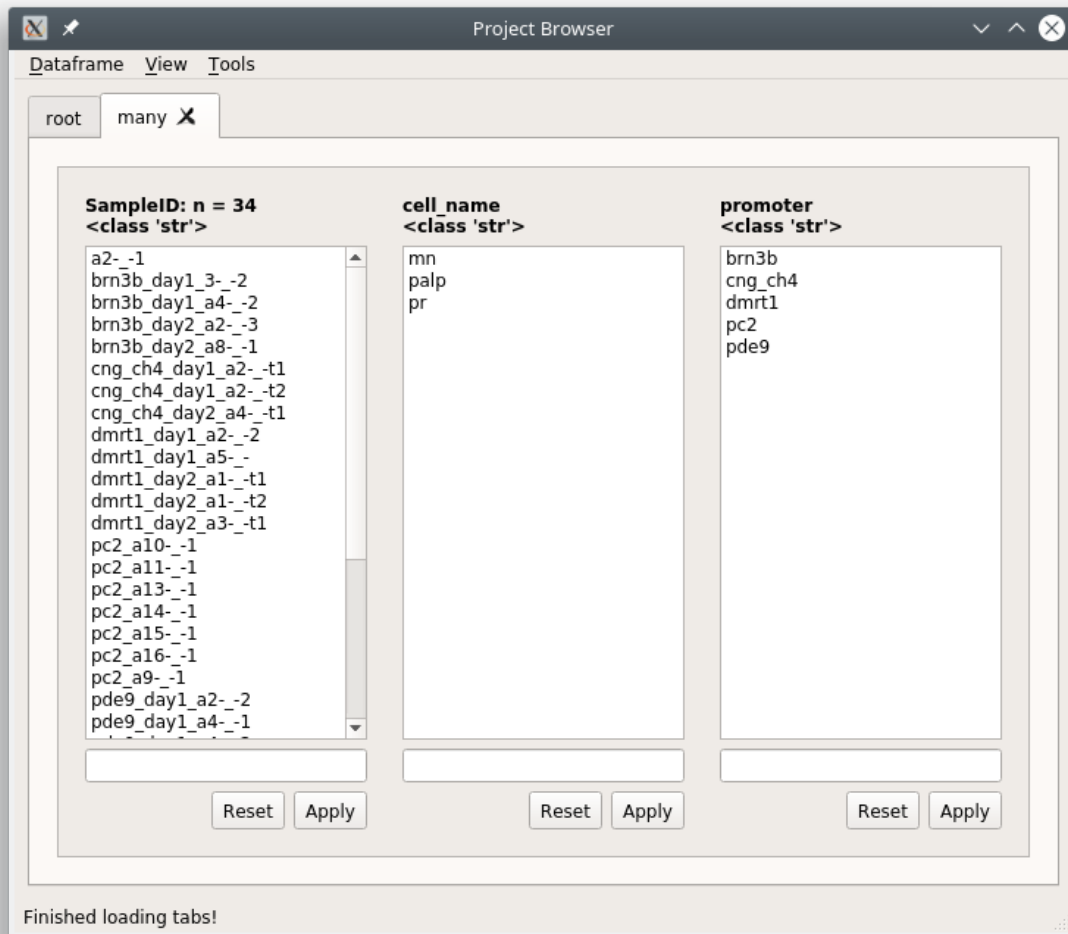


6.2.3 Multiple filters

You can combine filters together by using the | separator. The | acts as an “or” operator.



The result is all rows where mn, palp, or pr are in the cell_name column.

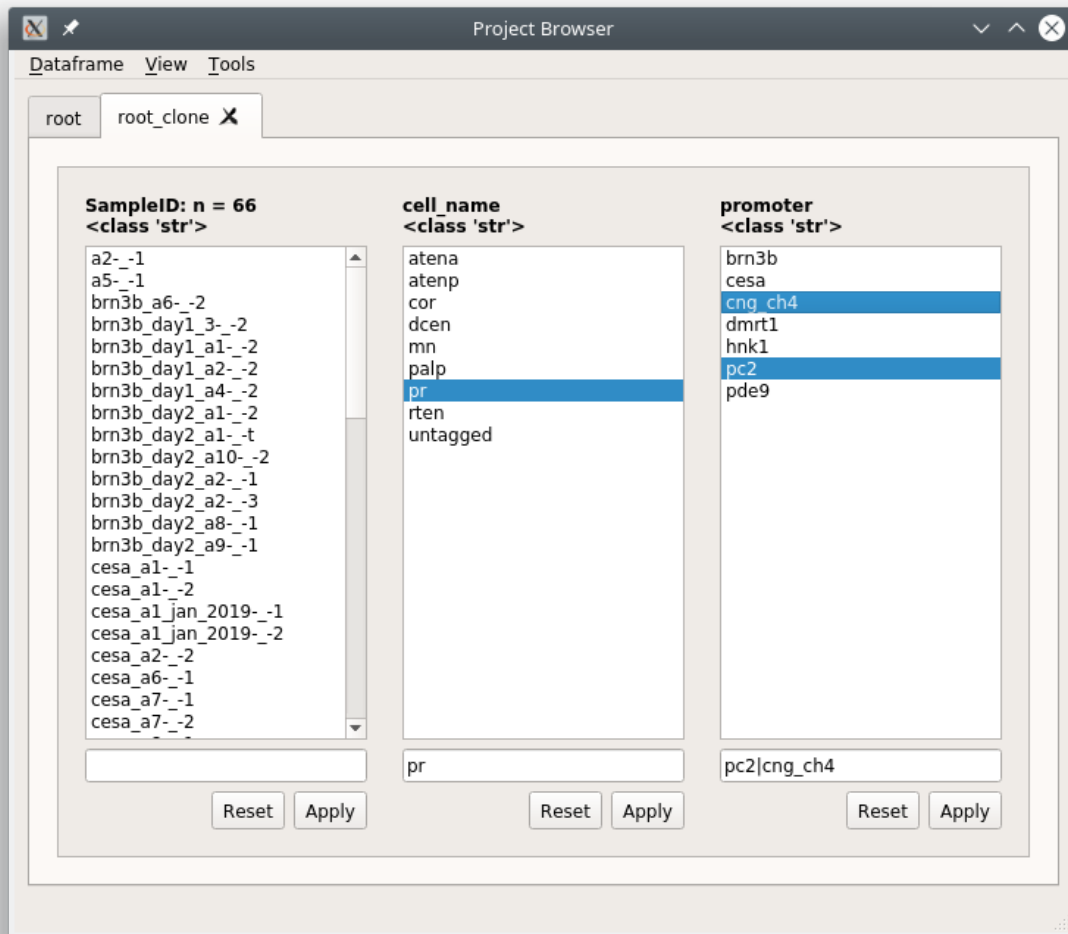


Note: This can be combined with *Modifiers*

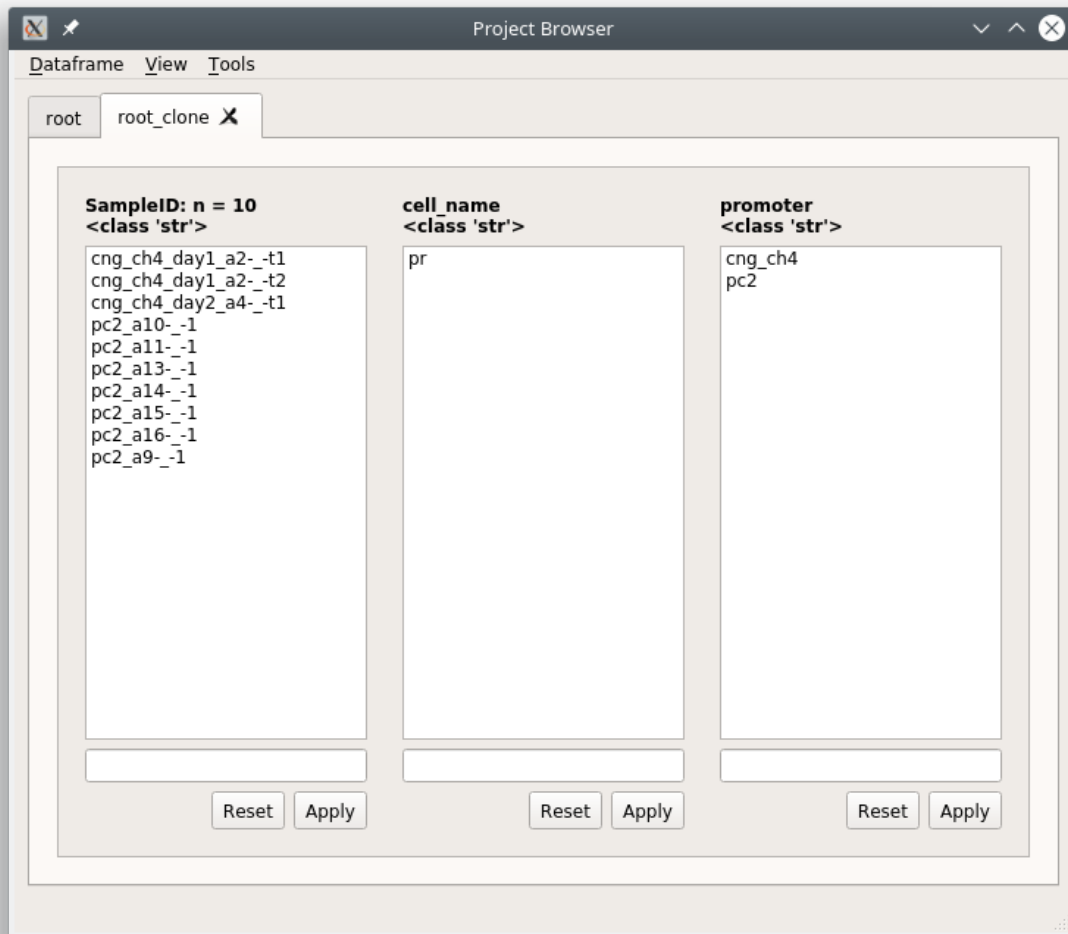
6.2.4 Filter multiple columns

You can filter multiple columns simultaneously if you are not in the root tab. You can create a new tab that is essentially the same as the root by just keeping the filter entries blank and clicking “Apply”.

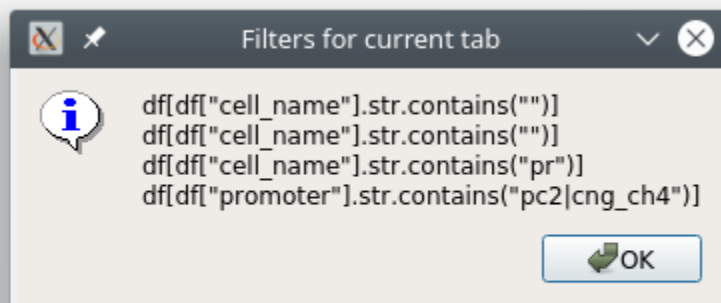
Filter out all rows where the cell_name column contains ‘pr’ and promoter column contains ‘pc2’ or ‘cng_ch4’.



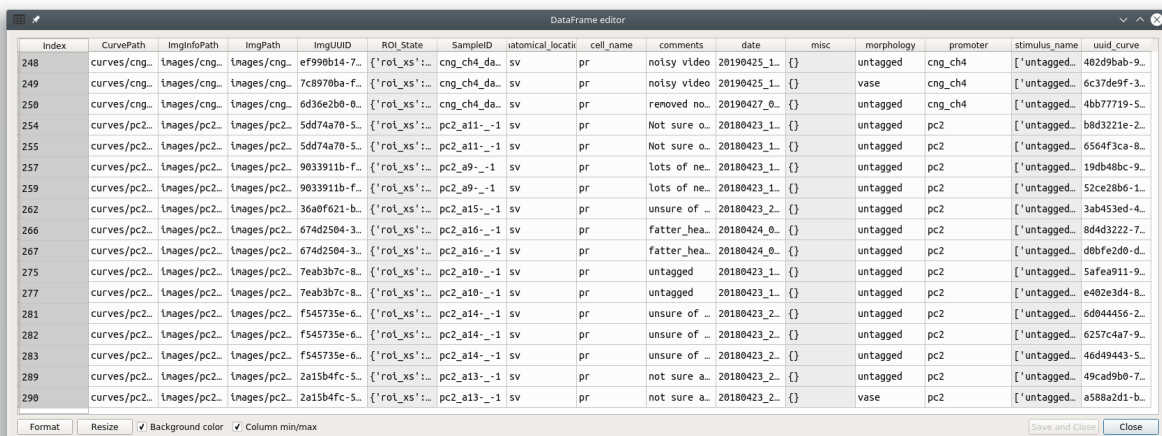
Right click on the “Apply” button and choose “Apply all” or “Apply all in new tab”



If you view the pandas filter history (View -> Current tab filter history) you can see that the filters for each column are simply applied sequentially.



The dataframe



Index	CurvePath	ImgInfoPath	ImgPath	ImgUUID	ROI_State	SampleID	anatomical_locati	cell_name	comments	date	misc	morphology	promoter	stimulus_name	uuid_curve
248	curves/cng...	images/cng...	images/cng...	ef990b14-7...	{'roi_xs': ...}	cng_ch4_da...	sv	pr	noisy video	20190425_1...	{}	untagged	cng_ch4	['untagged...	402d9bab-9...
249	curves/cng...	images/cng...	images/cng...	7c8978ba-f...	{'roi_xs': ...}	cng_ch4_da...	sv	pr	noisy video	20190425_1...	{}	vase	cng_ch4	['untagged...	6c37de9f-3...
250	curves/cng...	images/cng...	images/cng...	6d36e2b0-0...	{'roi_xs': ...}	cng_ch4_da...	sv	pr	removed no...	20190427_0...	{}	untagged	cng_ch4	['untagged...	4bb77719-5...
254	curves/pc2...	images/pc2...	images/pc2...	5dd74a70-5...	{'roi_xs': ...}	pc2_a11-_-1	sv	pr	Not sure o...	20180423_1...	{}	untagged	pc2	['untagged...	b8d3221e-2...
255	curves/pc2...	images/pc2...	images/pc2...	5dd74a70-5...	{'roi_xs': ...}	pc2_a11-_-1	sv	pr	Not sure o...	20180423_1...	{}	untagged	pc2	['untagged...	6564f3ca-8...
257	curves/pc2...	images/pc2...	images/pc2...	9033911b-f...	{'roi_xs': ...}	pc2_a9-_-1	sv	pr	lots of ne...	20180423_1...	{}	untagged	pc2	['untagged...	19db48bc-9...
259	curves/pc2...	images/pc2...	images/pc2...	9033911b-f...	{'roi_xs': ...}	pc2_a9-_-1	sv	pr	lots of ne...	20180423_1...	{}	untagged	pc2	['untagged...	52ce28b6-1...
262	curves/pc2...	images/pc2...	images/pc2...	36a0f621-b...	{'roi_xs': ...}	pc2_a15-_-1	sv	pr	unsure of ...	20180423_2...	{}	untagged	pc2	['untagged...	3ab453ed-4...
266	curves/pc2...	images/pc2...	images/pc2...	674d2504-3...	{'roi_xs': ...}	pc2_a16-_-1	sv	pr	fatter_ha...	20180424_0...	{}	untagged	pc2	['untagged...	8d4d3222-7...
267	curves/pc2...	images/pc2...	images/pc2...	674d2504-3...	{'roi_xs': ...}	pc2_a16-_-1	sv	pr	fatter_ha...	20180424_0...	{}	untagged	pc2	['untagged...	dbbfe2d0-d...
275	curves/pc2...	images/pc2...	images/pc2...	7eab3b7c-8...	{'roi_xs': ...}	pc2_a10-_-1	sv	pr	untagged	20180423_1...	{}	untagged	pc2	['untagged...	5afe9111-9...
277	curves/pc2...	images/pc2...	images/pc2...	7eab3b7c-8...	{'roi_xs': ...}	pc2_a10-_-1	sv	pr	untagged	20180423_1...	{}	untagged	pc2	['untagged...	e482e3d4-8...
281	curves/pc2...	images/pc2...	images/pc2...	f545735e-6...	{'roi_xs': ...}	pc2_a14-_-1	sv	pr	unsure of ...	20180423_2...	{}	untagged	pc2	['untagged...	6d044456-2...
282	curves/pc2...	images/pc2...	images/pc2...	f545735e-6...	{'roi_xs': ...}	pc2_a14-_-1	sv	pr	unsure of ...	20180423_2...	{}	untagged	pc2	['untagged...	6257c4a7-9...
283	curves/pc2...	images/pc2...	images/pc2...	f545735e-6...	{'roi_xs': ...}	pc2_a14-_-1	sv	pr	unsure of ...	20180423_2...	{}	untagged	pc2	['untagged...	46d49443-5...
289	curves/pc2...	images/pc2...	images/pc2...	2a15b4fc-5...	{'roi_xs': ...}	pc2_a13-_-1	sv	pr	not sure a...	20180423_2...	{}	untagged	pc2	['untagged...	49cad9b0-7...
290	curves/pc2...	images/pc2...	images/pc2...	2a15b4fc-5...	{'roi_xs': ...}	pc2_a13-_-1	sv	pr	not sure a...	20180423_2...	{}	vase	pc2	['untagged...	a588a2d1-b...

6.2.5 Modifiers

You can perform other types of matches, such as exact matches, negations, and exact negations. Enter the filter and then right click on the text entry to see available modifiers and choose the desired modifier.

```
***
"Not" modifier $NOT:
"String" modifier $STR:
"String equals" modifier (exact match of text) $STR=:
"String not equals" modifier $STR!=:
```

Modifier	Description
\$NOT:	Results in the negation of partial matches
\$STR:	Treats the filter as a str, same as Partial Match (see above sub-section)
\$STR=:	Exact text match
\$STR!=:	Negation of exact text match

6.2.6 Numerical filters

By default the filters in all entires are treated as text. If your column contains numerical data you have additional options for modifiers. The first four modifiers are the *same as explained above*. The rest are self explanatory.

```
...
"Not" modifier $NOT:
"String" modifier $STR:
"String equals" modifier (exact match of text) $STR=:
"String not equals" modifier $STR!=:
Greater than
Less than
Less than or equal to
Greater than or equal to
```

6.3 Editor

You can view and edit the Project DataFrame directly in a GUI using the DataFrame editor.

DataFrame editor										
Index	CurvePath	ImgInfoPath	ImgPath	MaxProjPath	ROI_State	SampleID	comments	date	promoter	uuid_curve
0	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
1	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
2	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
3	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
4	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
5	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
6	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
7	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
8	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
9	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
10	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
11	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
12	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
13	/curves/a2-...	/images/a2-...	/images/a2-...	/share/data...	{'roi_xs': ...	a2_-t1	untagged	20181215_024112	hnk1	7bc546b2-24...
14	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
15	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
16	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
17	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
18	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
19	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
20	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
21	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
22	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
23	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
24	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
25	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t1	untagged	20181215_025854	hnk1	20daa65b-d1...
26	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t2	untagged	20181215_030417	hnk1	a57ccdc2-0c...
27	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t2	untagged	20181215_030417	hnk1	a57ccdc2-0c...
28	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t2	untagged	20181215_030417	hnk1	a57ccdc2-0c...
29	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t2	untagged	20181215_030417	hnk1	a57ccdc2-0c...
30	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t2	untagged	20181215_030417	hnk1	a57ccdc2-0c...
31	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t2	untagged	20181215_030417	hnk1	a57ccdc2-0c...
32	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t2	untagged	20181215_030417	hnk1	a57ccdc2-0c...
33	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t2	untagged	20181215_030417	hnk1	a57ccdc2-0c...
34	/curves/a3-...	/images/a3-...	/images/a3-...	/share/data...	{'roi_xs': ...	a3_-t2	untagged	20181215_030417	hnk1	a57ccdc2-0c...
35	/curves/a4-...	/images/a4-...	/images/a4-...	/share/data...	{'roi_xs': ...	a4_-t1	untagged	20181215_031107	hnk1	f646df26-db...
36	/curves/a4-...	/images/a4-...	/images/a4-...	/share/data...	{'roi_xs': ...	a4_-t1	untagged	20181215_031107	hnk1	f646df26-db...
37	/curves/a4-...	/images/a4-...	/images/a4-...	/share/data...	{'roi_xs': ...	a4_-t1	untagged	20181215_031107	hnk1	f646df26-db...
38	/curves/a4-...	/images/a4-...	/images/a4-...	/share/data...	{'roi_xs': ...	a4_-t1	untagged	20181215_031107	hnk1	f646df26-db...
39	/curves/a4-...	/images/a4-...	/images/a4-...	/share/data...	{'roi_xs': ...	a4_-t1	untagged	20181215_031107	hnk1	f646df26-db...
40	/curves/a4-...	/images/a4-...	/images/a4-...	/share/data...	{'roi_xs': ...	a4_-t1	untagged	20181215_031107	hnk1	f646df26-db...
41	/curves/a5-...	/images/a5-...	/images/a5-...	/share/data...	{'roi_xs': ...	a5_-t1	untagged	20181215_034156	hnk1	202015b7-28...

Format
Resize
☒ Background color
☒ Column min/max
Save and Close
Close

Warning: Make sure you know what you are doing when you directly modify the Project DataFrame. Changes cannot be undone but you can restore a backup from the project's [dataframe directory](#). For example, do not modify data under the following columns: CurvePath, ImgInfoPath, ImgPath, ROI_State, any uuid column.

See also:

Uses the [Spyder object editor](#)

6.4 Console

If you are familiar with pandas you can interact with the project DataFrame directly. If you are unfamiliar with pandas it's very easy to learn.

See also:

[Pandas documentation](#)

Useful Callables

Callable	Purpose
<code>get_dataframe()</code>	returns dataframe of the current project browser tab
<code>get_root_dataframe()</code>	always returns dataframe of the root tab (entire project DataFrame)
<code>set_root_dataframe()</code>	pass a pandas.DataFrame instance to set it as the project DataFrame

6.4.1 Usage

General usage to modify the project DataFrame would be something like this:

```
# Get a copy the project DataFrame to modify
df = get_root_dataframe().copy()

# Do stuff to df
...

# Set the project DataFrame with the modified one
set_root_dataframe(df)
```

6.4.2 Example

Let's say you have been inconsistent in naming "ATENA" ROI Tags in the "cell_name" column. You can rename all occurrences of 'atena' to 'ATENA'

```
# Get a copy of the project DataFrame
>>> df = get_root_dataframe().copy()

# View all occurrences of 'atena'
>>> df.cell_name[df.cell_name == 'atena']
2      atena
3      atena
4      atena
5      atena
6      atena
205    atena
Name: cell_name, dtype: object

# Rename all occurrences of 'atena' to 'ATENA'
>>> df.cell_name[df.cell_name == 'atena'] = 'ATENA'

# Check that there are more occurrences of 'atena'
>>> df.cell_name[df.cell_name == 'atena']
Series([], Name: cell_name, dtype: object)
```

(continues on next page)

(continued from previous page)

```
# Check that we have renamed the 'atena' occurrences to 'ATENA'
# Indices 2-6 and 205 were named 'atena'
>>> df.cell_name
0      untagged
1      untagged
2        ATENA
3        ATENA
4        ATENA
5        ATENA
6        ATENA
7        atenp
...
Name: cell_name, Length: 311, dtype: object

# Check index 205
>>> df.cell_name.iloc[205]
'ATENA'

# Finally set the changed DataFrame as the root (project) DataFrame
>>> set_root_dataframe(df)
```


VIEWER OVERVIEW

Based on the [pyqtgraph ImageView](#) widget.

The Viewer allows you to do the following things:

- Examine your calcium movies
- Use modules to perform things like motion correction, CNMF(E), ROI labeling, and stimulus mapping. See their respective guides for details.
- You can also make modifications to an existing Sample in your project by opening it in the Viewer. See [Modify Sample](#) and [Overwrite](#) guide.

7.1 Layout

To access Viewer modules choose the module you want to run from the Modules menu at the top. All modules, except the Batch Manager, are small floating windows which you can dock into the Viewer by dragging them to an edge of the viewer.

7.2 Work Environment

Everything in the viewer is stored in a Work Environment object. The main data attributes of the viewer work environment are outlined below.

See also:

ViewerWorkEnv API

Attribute	Description
imgdata	ImgData object containing the Image Sequence and meta data from the imaging source
roi_manager	The back-end ROI Manager that is currently in use
sample_id	SampleID, if opened from a project Sample
stim_maps	Stimulus maps, if any are defined
history_trace	History log, currently used for logging <i>caiman motion correction</i> , <i>CNMF</i> and <i>CNMFE</i> history.
UUID	If opened from a project Sample, it refers to the ImgUUID

You can view everything in the current work environment by going to View -> Work Environment Editor. You cannot edit through this GUI at this time.

7.3 Menubar

7.3.1 File

Add to Project

Add the current *work environment* as a Sample to the project.

Open Work Environment

Deprecated

Save Work Environment

Deprecated

Clear Work Environment

Clear the current *work environment*. Useful for freeing up RAM.

7.3.2 Edit

Deprecated

7.3.3 Image

Reset Scale

Reset the scale of the image ViewBox

Resize

Resize the image sequence using interpolation.

Crop

Crop the image sequence.

Usage

1. When you click this option a square crop region will appear in the top left corner of the image sequence.
2. You can change its shape using the handle in the bottom right corner.
3. To crop to the selection, in the menubar go to Image -> Crop. To cancel cropping right click in the crop region and click "Remove ROI".

Measure

Measure the distance (in pixels) between two points in the image sequence.

Usage

1. After clicking this option in the menubar, click on a point in the image sequence. You will not see anything yet.
2. Click on a second point in the image sequence, a line will appear connecting the first and second points that you clicked.
3. You can use the handles at the endpoints of the line to change the line.
4. To measure the distance of the line go to Image -> Measure. A window will pop up displaying the change in x, y, and length of the line in pixels.

Change dtype

Not implemented yet. You can change the dtype through the console.

Projections

View Mean, Max, and Standard Deviation projections of the current image sequence in the work environment.

7.3.4 Modules

Default Viewer Modules. These are explained in more details in the Viewer Modules chapters.

7.3.5 Plugins

Custom viewer modules.

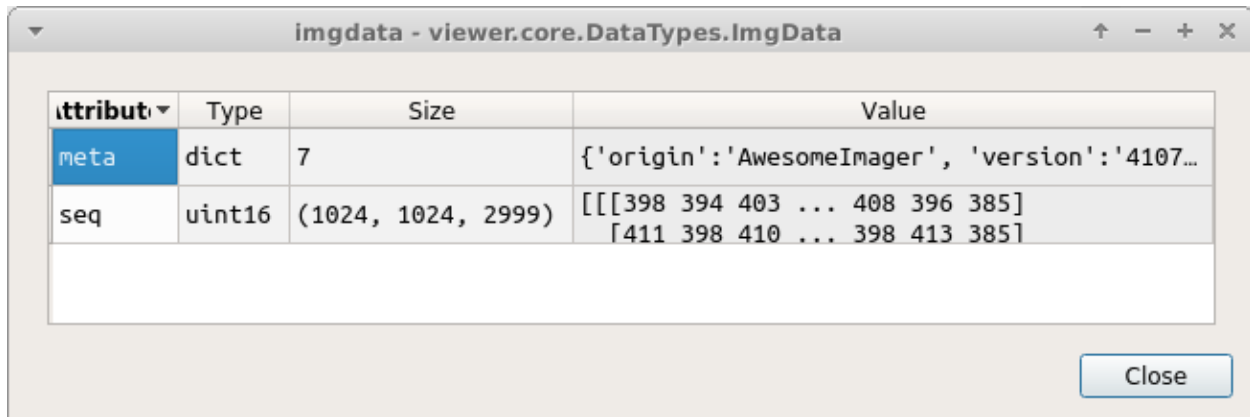
7.3.6 View

Work Environment Editor

Explore the data in your work environment using a GUI.

Note: This is read only, you cannot edit via this GUI.

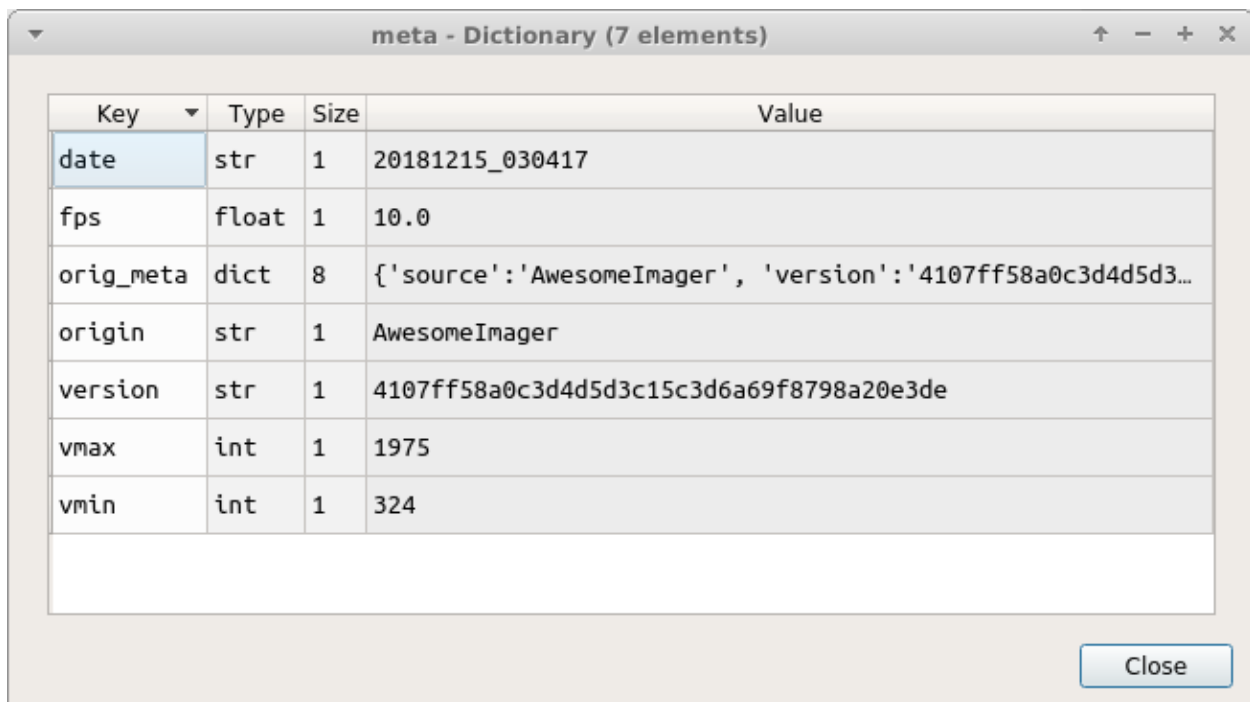
For example if you want to see your meta data, double click on “imgdata” and then you can see that “imgdata” has two things, the image sequence (i.e. your video) and the meta data.



attribut	Type	Size	Value
meta	dict	7	{'origin': 'AwesomeImager', 'version': '4107...
seq	uint16	(1024, 1024, 2999)	[[[398 394 403 ... 408 396 385] [411 398 410 ... 398 413 385]

Close

If you double click on “meta” above you can see your meta data.



Key	Type	Size	Value
date	str	1	20181215_030417
fps	float	1	10.0
orig_meta	dict	8	{'source': 'AwesomeImager', 'version': '4107ff58a0c3d4d5d3...
origin	str	1	AwesomeImager
version	str	1	4107ff58a0c3d4d5d3c15c3d6a69f8798a20e3de
vmax	int	1	1975
vmin	int	1	324

Close

You can view your meta data more quickly using the console.

Open the console by going to View -> Console. You can then call `get_meta()` to print the meta data dict.

Console

View/hide the viewer console

7.3.7 Help

Open docs

Open these docs

7.4 Console

You can interact directly with the *work environment* using the console.

See also:

Viewer Core API, *Overview on consoles*

7.4.1 Namespace

Reference	Description
vi	Instance of ViewerUtils. Use this to interact with the viewer.
all_modules	List all available modules (includes default and any available plugins/custom modules)
ViewerWorkEnv	Use for creating new instances of ViewerWorkEnv
ImgData	Use for creating new instances of ImgData
get_workEnv()	Get the current viewer <i>work environment</i> (instance of ViewerWorkEnv)
get_image()	Get the current image sequence (returns current ViewerWorkEnv.imgdata.seq)
get_meta()	Get the current meta data
get_module(<name>)	Pass the name of a module as a string. Returns that module if it is available.
get_batch_manager()	Get the batch manager.
update_workEnv()	Update the viewer GUI with the viewer work environment (vi.viewer.workEnv)
clear_workEnv()	Clear the current work environment, cleanup the GUI and free the RAM

7.4.2 Examples

View meta data

```
>>> get_meta()

{'origin': 'AwesomeImager', 'version': '4107ff58a0c3d4d5d3c15c3d6a69f8798a20e3de',
↪ 'fps': 10.0, 'date': '20190426_152034', 'vmin': 323, 'vmax': 1529, 'orig_meta': {
↪ 'source': 'AwesomeImager', 'version': '4107ff58a0c3d4d5d3c15c3d6a69f8798a20e3de',
↪ 'level_min': 323, 'stims': {}, 'time': '152034', 'date': '20190426', 'framerate':
↪ 10.0, 'level_max': 1529}}
```

View history trace

```
>>> get_workEnv().history_trace

[{'caiman_motion_correction': {'max_shifts_x': 32, 'max_shifts_y': 32, 'iters_rigid': 1, 'name_rigid': 'Does not matter', 'max_dev': 20, 'strides': 196, 'overlaps': 98, 'upsample': 4, 'name_elas': 'a1_t2', 'output_bit_depth': 'Do not convert', 'bord_px': 5}}, {'cnmfe': {'Input': 'Current Work Environment', 'frate': 10.0, 'gSig': 10, 'bord_px': 5, 'min_corr': 0.9600000000000001, 'min_pnr': 10, 'min_SNR': 1, 'r_values_min': 0.7, 'decay_time': 2, 'rf': 80, 'stride': 40, 'gnb': 8, 'nb_patch': 8, 'k': 8, 'name_corr_pnr': 'a8_t1', 'name_cnmfe': 'a1_t2', 'do_corr_pnr': False, 'do_cnmfe': True}}, {'cnmfe': {'Input': 'Current Work Environment', 'frate': 10.0, 'gSig': 10, 'bord_px': 5, 'min_corr': 0.9600000000000001, 'min_pnr': 14, 'min_SNR': 1, 'r_values_min': 0.7, 'decay_time': 4, 'rf': 80, 'stride': 40, 'gnb': 8, 'nb_patch': 8, 'k': 8, 'name_corr_pnr': '', 'name_cnmfe': 'a1_t2', 'do_corr_pnr': False, 'do_cnmfe': True}}]
```

Open image

```
1  # Get the tiff module
2  tio = get_module('tiff_io')
3
4  # Set the tiff and meta file paths
5  tiff_path = '/path_to_tiff_file.tiff'
6  meta_path = '/path_to_meta_file.json'
7
8  tio.load_tiff(tiff_path, meta_path, method='asarray')
```

See also:

Tiff IO API, Information on the *meta data format*

Use the ViewerCore API to open any arbitrary image

```
1  import tifffile
2  import json
3
4  # Open an image file in whatever way you require
5  seq = tifffile.imread('/path_to_tiff_file')
6
7  meta = ...
8
9  # Create ImgData instance
10 # Image sequence must be numpy array of shape [x, y, t]
11 imgdata = ImgData(seq.T, meta)
12
13 # Create a work environment instance
14 work_env = ViewerWorkEnv(imgdata)
15
16 # Set this new work environment instance as the viewer work environment
17 vi.viewer.workEnv = work_end
18
19 # Update the viewer GUI with the new work environment
20 update_workEnv()
```

Splice img seq

Extract the image sequence between frame 1000 and 2000. Image sequences are simply numpy arrays.

See also:

[Numpy array indexing](#)

```
1  # Get the current image sequence
2  seq = get_image()
3
4  # Trim the image sequence
5  trim = seq[:, :, 1000:2000]
6
7  # Set the viewer work environment image sequence to the trim one
8  vi.viewer.workEnv.imgdata.seq = trim
9
10 # Update the GUI with the new work environment
11 update_workEnv()
```

7.4.3 Running scripts

You can use the *Script Editor* to run scripts in the Viewer console for automating tasks such as batch creation. It basically allows you to use the *viewer console* more conveniently with a text editor.

CONVERT META DATA

In order for you to use your videos with Mesmerize, each of your videos must have meta data organized in the format below.

The simplest way to use your meta data with Mesmerize is to dump all of it into a single json file with the same name as that of the video it is for. You could also add another `elif` block to `mesmerize.viewer.core.ViewerWorkEnv._organize_meta()`.

For example if you have a movie named “**xyz.tiff**” the json meta data file for that movie would be “**xyz.json**” for the Tiff input module to automatically recognize it.

8.1 Minimal Example

```
{
    "source":    "our_recording_program",
    "fps":       20.0,
    "date":      "20190425_114405"
}
```

8.2 Fields and data types

These fields and their types are mandatory

- **source** (*str*): Name of recording program used, for your own record
- **fps** (*float*): Framerate of the video
- **date** (*str*): Recording date & time separated by an underscore

If you have other data that you would like to keep for the purpose of custom modules or other analysis you can organize them into a single dict under the “`orig_meta`” key.

Example with more fields:

```
{
    "source":    "our_recording_program",
    "version":    0.1.2,
    "fps":       20.0,
    "date":      20190425_114405,
    "orig_meta": {
        "scanner_info": [1, 2, 3],
    }
}
```

(continues on next page)

(continued from previous page)

```
        "other_stuff": "val"
    }
}
```

ADD A SAMPLE TO THE PROJECT

When you are happy with the ROIs in the viewer for the current CNMF(E) derived or manually created ROIs, you can add this as a *Sample* to your project.


Each sample in your project contains the following:

- The imaging data from which ROIs were extracted (the video)
- All the ROIs with their spatial location, temporal dynamics, and any tags that you have entered in the ROI Manager.
- Stimulus mappings, if your project is configured for this.
- Meta data (that were associated with the imaging video), the date, video framerate.
- Any further information that you have chosen to add based on your Project Configuration

Note: If your ROIs were obtained through CNMF/CNMFE the following attributes from the final cnm object are stored: cnm.A, cnm.b, cnm.C, cnm.f, cnm.YrA

9.1 How to

To add the current *viewer work environment* (see above) as a sample to your project, go to File -> Add To Project. You will be presented with a window similar to this:



The screenshot shows a window titled "Add to Project" with standard window controls (minimize, maximize, close). Inside the window, there are several input fields: a text box for "*Animal ID", a text box for "*Trial ID", a text box for "drug :str", a text box for "chemogenetic_state :str", a text box for "developmental_stage :int64", and a text box for "gene :str". Below these is a large text area labeled "Comments". At the bottom of the window, there is a radio button labeled "Add to dataframe" which is selected, and a "Proceed" button.

The entries that you are prompted with directly correspond to the custom columns in your Project Configuration.

See also:

Project Configuration

Every Sample in a project has a unique **SampleID** which is the combination of **AnimalID** + **TrialID**.

Warning: You can never change the **AnimalID** or **TrialID** (i.e. **SampleID**) since these are partially used as **unique identifiers**. A workaround is described in the [FAQ for Project Organization](#).

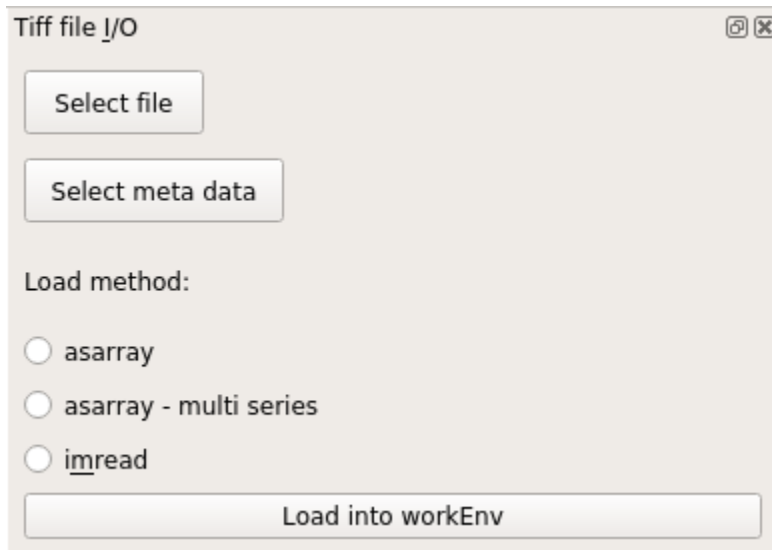
Warning: **AnimalID** and **TrialID** are separated by the `--` character combination when stored as a **SampleID**. Therefore do not use that character combination within your **AnimalID** or **TrialID**.

TIFF FILE MODULE

To open a tiff file go to Modules -> Load Images -> Tiff files.

Note: You can also use this module through the console and scripts. See `API_TiffModule`

To open tiff files just click the “Select file” button and choose your file. You can also drag and drop a tiff file.



Tiff file I/O

Select file

Select meta data

Load method:

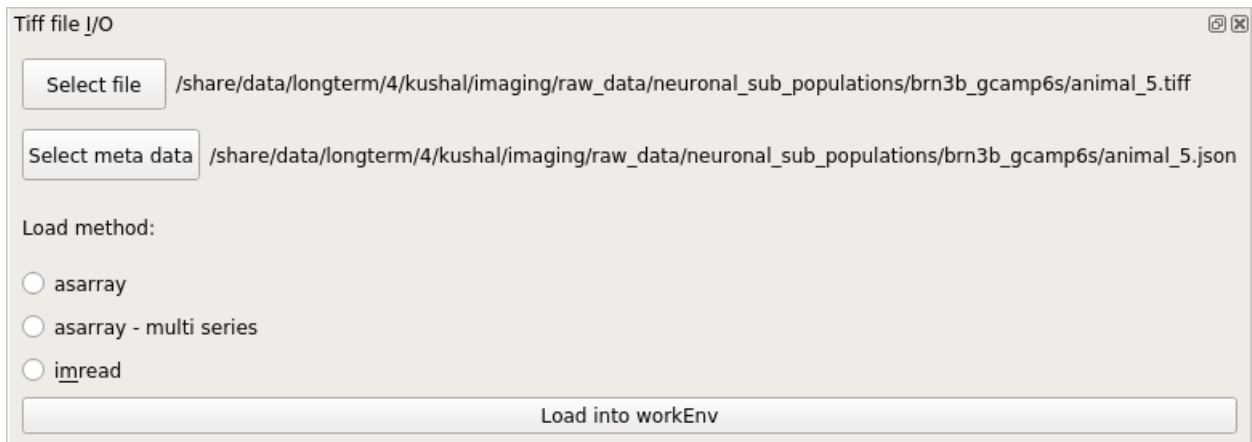
☐ asarray

☐ asarray - multi series

☐ imread

Load into workEnv

When you choose a tiff file it will automatically find the associated .json meta data if it has the same filename.



Tiff file I/O

Select file /share/data/longterm/4/kushal/imaging/raw_data/neuronal_sub_populations/brn3b_gcamp6s/animal_5.tiff

Select meta data /share/data/longterm/4/kushal/imaging/raw_data/neuronal_sub_populations/brn3b_gcamp6s/animal_5.json

Load method:

☐ asarray

☐ asarray - multi series

☐ imread

Load into workEnv

Finally, select an appropriate load method (see next section) and click “Load into workEnv”

Warning: If the name of the tiff file and .json meta data file are different, you must specify the .json meta data file using the *Select meta data* button.

Warning: You cannot perform any analysis without the meta data file since you need the sampling rate of the video and it is specified in the meta data file.

10.1 Load Method

The options for “Load Method” correspond to the `tiff` library method that is used for loading the images.

If you are not sure which method you should use, try all of them and see which one loads your data appropriately. If none of them work, contact us and I may be able to add more methods.

Note: If you have an unsaved work environment open (such as a video with ROIs for example) it will prompt you to confirm that you want to clear the work environment before loading the chosen image into the work environment.

10.1.1 `asarray`

Should work for most tiff files, fast method

10.1.2 `asarray - multi series`

Also fast. Use this if it is a multi-page tiff. For example if the tiff file was created by a program that appends each frame to a file as they are being acquired by the camera.

10.1.3 `imread`

Usually slower, should work for most tiff files.

BATCH MANAGER

Batch process computationally intensive tasks.

See also:

Batch Manager API

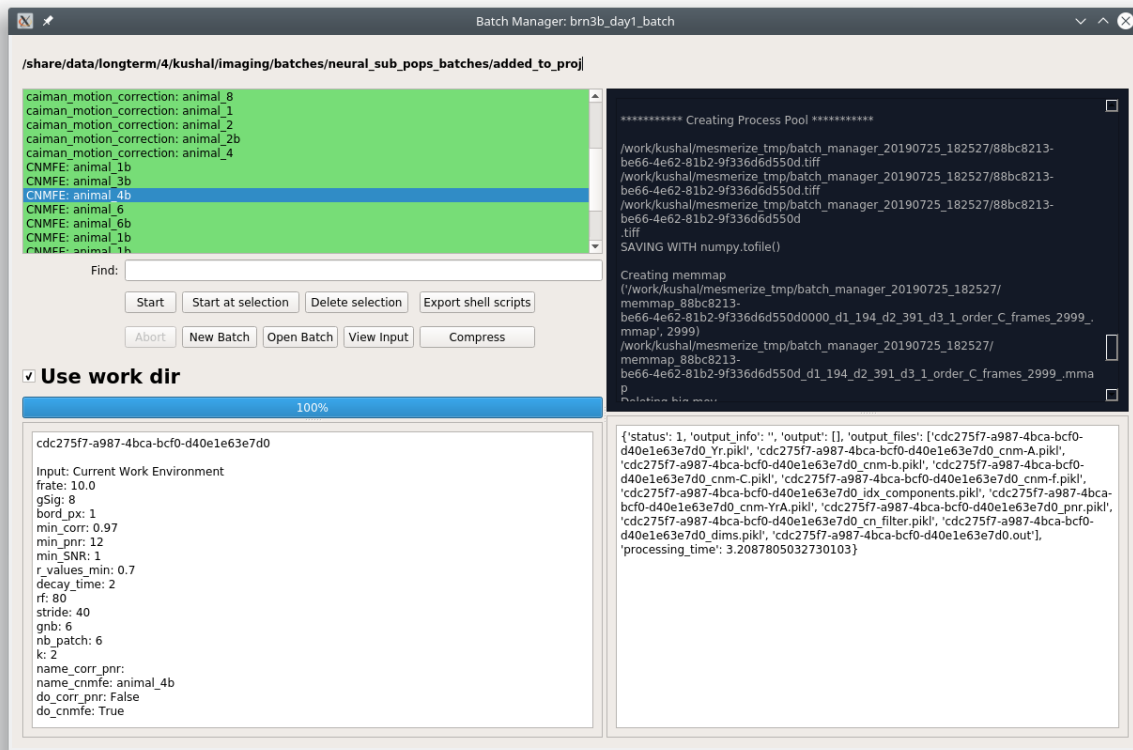
This is currently used for *Caiman Motion Correction*, *CNMF*, and *CNMF-E*.

The Batch Manager can be accessed in the viewer through Modules -> Batch Manager. If you don't have a batch open you will be prompted with a dialog to open a batch or to select a location for a new batch.

Warning: The full path to the batch directory must not contain spaces or special characters, only a-z, A-Z, 0-9 and underscores.

The Batch Manager processes the batch items in external processes, allowing you to add batch items when that batch is being processed.

11.1 Layout



Window title: Name of batch directory

Top: Parent directory of batch directory

Top left: list of batch items and some controls.

Colors	Description
Green	Finished without exceptions
Red	Did not finish, click on the item to see the exceptions in the bottom right information area
Yellow	Currently being processed
Orange	Item aborted by user
Blue	Output data for this item are being moved from the work dir to the batch dir.

Button	Description
Start	Process the batch from the first item.
Start at selection	Process the batch starting from the item that is currently selected in the list.
Delete selection	Delete the item that is currently being selected along with the associated data in the batch dir.
Export shell scripts	Export bash scripts so that the batch items can be run on a computing cluster
Abort	Abort the current batch item
New batch	Create a new batch
Open batch	Open a batch
View Input	Open the input work environment, in the viewer, for the currently selected item
Compress	Not implemented

Use work dir: Check this box to use the work dir that has been set in the *System Configuration*

Top right: Standard out from the external processes that are processing the batch items.

Bottom left: Parameters for the selected batch item. The first line is the UUID of the batch item.

Bottom right: Output information area for the currently selected item.

11.2 Scheduling

You can schedule a batch to run at a later time using the following bash script. Doesn't work for a snap installation yet.

```
mesmerize-scheduler
```

Usage:

```
Usage: mesmerize-scheduler -b <batch> -i <start item> -t <start time>
```

```
-b      full batch path in quotes, no spaces
-i      uuid of the batch item to start from, no quotes
-t      time at which to start the batch, no quotes
```

```
examples of how to specify time:
```

```
23:00 7:30Feb30
```

```
use 24hr time and no spaces
```

```
Full usage example:
```

```
mesmerize-scheduler -b "/share/data/temp/kushal/pc2_batch" -i a80d1923-e490-4eb3-
↪ba4f-7e651d4cf938 -t 2:00
```


STIMULUS MAPPING

API Reference











Map temporal information such as stimulus or behavioral periods.

Stimulus Mapping Module

Stimulus Mapping

Show on timeline: tf

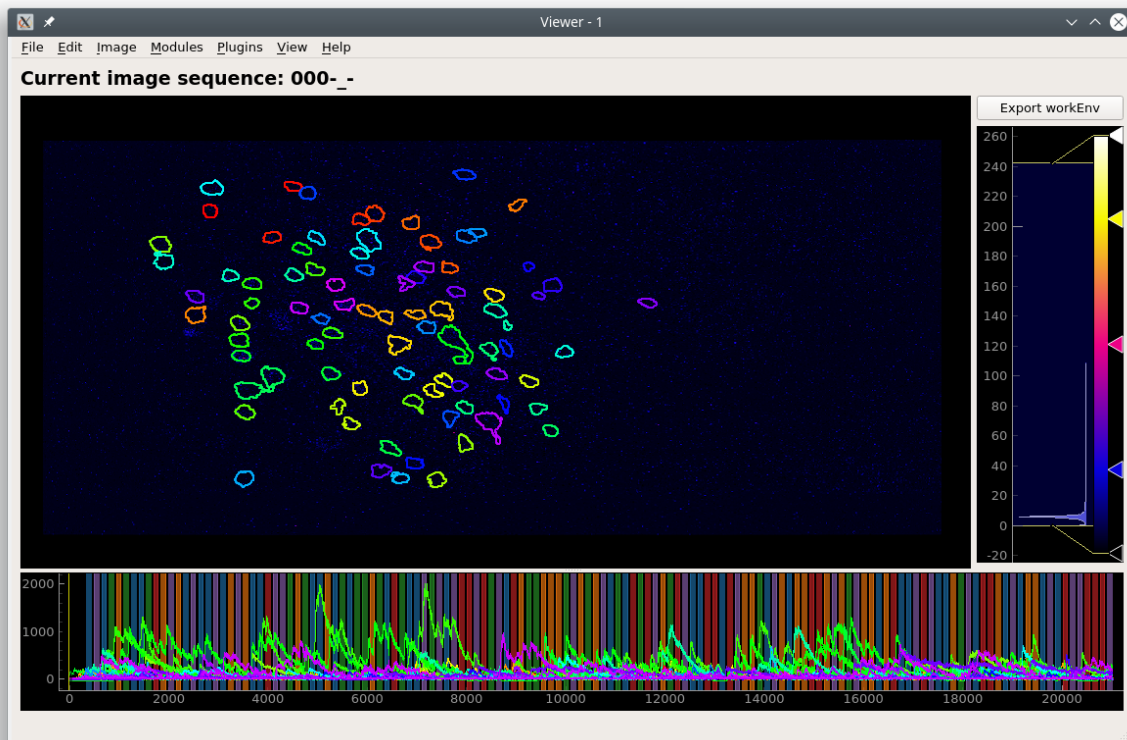
ori sf **tf**

2	6065.0	6155.0		Remove stim
2	15224.0	15314.0		Remove stim
4	9368.0	9458.0		Remove stim
8	11320.0	11410.0		Remove stim
1	359.0	449.0		Remove stim
1	16275.0	16365.0		Remove stim
15	15825.0	15915.0		Remove stim
15	20930.0	21020.0		Remove stim
1	17476.0	17566.0		Remove stim
2	18678.0	18768.0		Remove stim

units: frames Add Row

Import Export Cancel Set all maps

Stimulus periods illustrated on the viewer timeline



The tabs that are available in the stimulus mapping module corresponds to the stimulus types in your *Project Configuration*.

You can add stimulus periods either manually or through a script.

12.1 Manual Annotation

1. To add a stimulus manually click the “Add Row” button. This will add an empty row to the current tab page.
2. Enter a name for the stimulus, start time, end time, and pick a color for illustrating the stimulus periods on the Viewer timeline.
3. To remove a stimulus click the “Remove stim” button. Stimulus periods do not have to be added in chronological order.
4. Click “Set all maps” to set the mappings for all stimulus types. You can then choose to illustrate a stimulus on the viewer timeline by selecting it from “Show on timeline”

Import and Export are not implemented yet.

Warning: At the moment, only “frames” are properly supported for the time units.

12.2 Script

See also:

API Reference

You can also use the *Stimulus Mapping module's API* to set the stimulus mappings from a pandas DataFrame.

This example creates a pandas DataFrame from a csv file to set the stimulus mappings. It uses the csv file from the pvc-7 dataset available on CRCNS: <http://dx.doi.org/10.6080/K0C8276G>

You can also download the csv here: `stimulus_pvc7.csv`

```

1 import pandas as pd
2 from mesmerize.plotting.utils import get_colormap
3
4 # Load dataframe from CSV
5 df = pd.read_csv('/share/data/longterm/4/kushal/crcns_datasets/pvc-7/122008_140124_
  ↳ windowmix/stimulus.csv')
6
7 # Sort according to time (stimulus "start" frame column)
8 df.sort_values(by='start').reset_index(drop=True, inplace=True)
9
10 # Trim off the periods that are not in the current image sequence
11 # This is just because this example doesn't use the whole experiment
12 trim = get_image().shape[2]
13 df = df[df['start'] <= trim]
14
15 # Remove the unused columns
16 df.drop(columns=['sf', 'tf', 'contrast'])
17
18 # Rename the stimulus column of interest to "name"
19 df.rename(columns={'ori': 'name'}, inplace=True)
20
21 # Get the names of the stimulus periods to create a colormap for illustration in the
  ↳ curves plot area
22 oris = df['name'].unique()
23 oris.sort()
24
25 # Create colormap to visualize the stimuli in the viewer's curve plots area
26 oris_cmap = get_colormap(oris, 'tab10', output='pyqt', alpha=0.6)
27
28 # Create a column with colors that correspond to the orientations column values
29 df['color'] = df['name'].map(oris_cmap)
30
31 # name column must be str type for stimulus mapping module
32 # the ori data from the original csv is integers so you must change it
33 df['name'] = df['name'].apply(str)
34
35 # Get the stimulus mapping module
36 smm = get_module('stimulus_mapping')
37
38 # Set the ori colormap
39 smm.maps['ori'].set_data(df)

```


ROI MANAGER

API Reference

Manage and annotate ROIs

The ROI Manager has a manual mode, to draw ROIs manually, and a CNMF(E) mode where ROIs can be imported from CNMF(E) outputs.

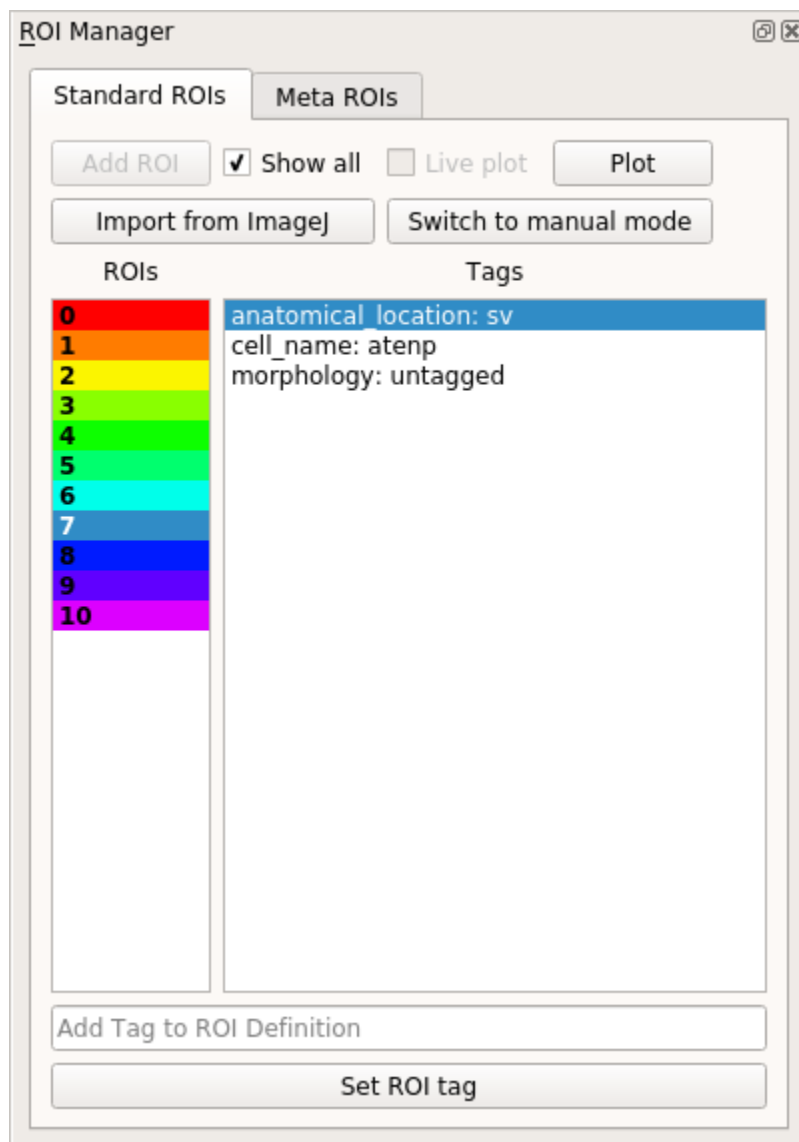
See also:

CNMF Usage and *CNMF(E) Usage*.

Note: You cannot combine manual and CNMF(E) ROIs in the same sample.

The ImageJ ROI import uses the read-roi package by Hadrien Mary <https://pypi.org/project/read-roi/>

13.1 Layout



Controls

UI	Description
Add ROI button	Add Polygon ROI (Manual mode) Right click this button to add an elliptical ROI
Show all	Show all ROIs in the viewer
Live plot	Live update of the curve plot with changes (Manual mode)
Plot	Plot the curves (Manual mode)
Import from ImageJ	Import ROIs from an ImageJ ROIs zip file (Manual mode)
Switch to manual . . .	Switch to Manual mode. Clears CNMF(E) ROIs.
ROIs list	Color-coded list of ROIs. Left click to highlight the ROI in the viewer Right click to show the context menu allowing you to delete the selected ROI
Tags list	List of tags for the selected ROI Correspond to the <i>ROI Type Columns of the Project Configuration</i>
Add Tag to ROI Def. . .	Set the tag for the current selection in the Tags list
Set ROI Tag	Click to set the tag, or just press return in the text entry above

Keyboard shortcuts.

These only work when the ROI manager is docked within the Viewer and while you are typing in the *Add Tag to ROI Definition* text entry.

Key	Description
Page Up	Select previous ROI
Page Down	Select next ROI
Right Arrow	Play the video at high speed
Left Arrow	Play the video backwards at high speed
Home	Go to the beginning of the video
End	Go to the end of the video

13.2 Manual ROI Mode

When you click the “Add ROI” button to add a Manual Polygon ROI, a new rectangular ROI will be add in the top left corner of the image. You can add new vertices to this polygon by clicking on any of its edges. You can drag the vertices to change the shape of the polygon, and you can drag the entire ROI as well by clicking and dragging within the ROI region. Similarly you can reshape elliptical ROIs.

Hovering over the ROI selects it in the ROI list.

13.3 Console

Access the back-end ROI Manager through the viewer console or Script editor to interact with the ROIs.

See also:

Back-end ROI Manager APIs, ROIList API, ROI Type APIs

Get the back-end ROI Manager, see ROI Manager APIs

```
>>> get_workEnv().roi_manager

<mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCNMFE object at 0x7f01b8780668>
```

Get the ROI List, see ROIList API

```
>>> get_workEnv().roi_manager.roi_list

[<mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI object at 0x7f01bc78b278>, <mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI object at 0x7f01bc817630>, <mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI object at 0x7f01bc817668>, <mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI object at 0x7f01bc7c5438>, <mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI object at 0x7f01bc7c5208>]
```

Work with an ROI object, see ROI Type APIs

```
# Get the curve data of an ROI
>>> get_workEnv().roi_manager.roi_list[3].curve_data

(array([ 0, 1, 2, ..., 2995, 2996, 2997]), array([-207.00168389, -161.78229208, -157.62522988, ..., -1017.73174502, -1030.27047731, -1042.26989668]))

# Get the tags of an ROI
>>> get_workEnv().roi_manager.roi_list[2].get_all_tags()

{'anatomical_location': 'tail', 'cell_name': 'dcen', 'morphology': 'untagged'}

# Get a single tag
>>> get_workEnv().roi_manager.roi_list[2].get_tag('cell_name')

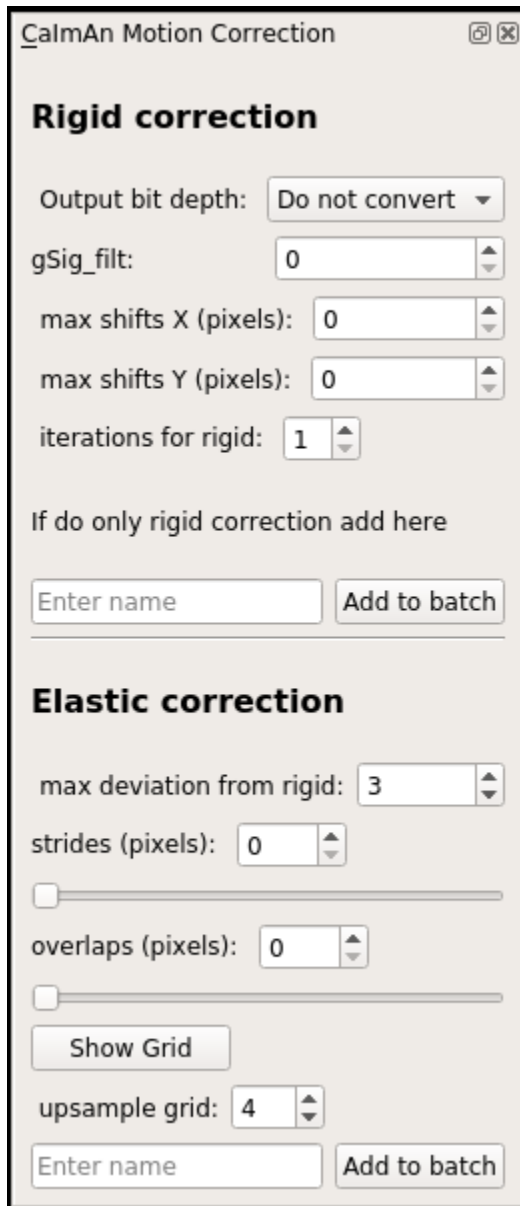
'dcen'
```

CAIMAN MOTION CORRECTION

Perform motion correction using the NoRMCorre implementation in the CaImAn library.

I highly recommend going through the following before using this module

- **NoRMCorre paper** Pnevmatikakis, E. A., & Giovannucci, A. (2017). NoRMCorre: An online algorithm for piecewise rigid motion correction of calcium imaging data. *Journal of Neuroscience Methods*, 291, 83–94.
- **The CaImAn demo notebook, the implementation in Mesmerize is basically from the demo** https://github.com/flatironinstitute/CaImAn/blob/master/demos/notebooks/demo_motion_correction.ipynb

The image shows a software window titled "CalmAn Motion Correction". It contains two main sections: "Rigid correction" and "Elastic correction". The "Rigid correction" section has a dropdown for "Output bit depth" set to "Do not convert", and spinners for "gSig_filt" (0), "max shifts X (pixels)" (0), "max shifts Y (pixels)" (0), and "iterations for rigid" (1). Below this is a text input field "Enter name" and an "Add to batch" button. The "Elastic correction" section has a spinner for "max deviation from rigid" (3), a spinner for "strides (pixels)" (0) with a slider below it, a spinner for "overlaps (pixels)" (0) with a slider below it, a "Show Grid" button, and a spinner for "upsample grid" (4). At the bottom are another "Enter name" field and an "Add to batch" button.

CalmAn Motion Correction

Rigid correction

Output bit depth: Do not convert

gSig_filt: 0

max shifts X (pixels): 0

max shifts Y (pixels): 0

iterations for rigid: 1

If do only rigid correction add here

Enter name Add to batch

Elastic correction

max deviation from rigid: 3

strides (pixels): 0

overlaps (pixels): 0

Show Grid

upsample grid: 4

Enter name Add to batch

Parameters

Output bit depth: The motion corrected image sequences are of float32 type. You can optionally convert the output to 8 or 16 bit uint types to save disk space. **This doesn't always work from my experience, values might get clipped.**

For all other parameters please see the demo notebook mentioned above.

14.1 Usage

This module adds a “caiman motion correction” *item* to the batch. Set the desired parameters (see demo notebook) and then enter a name to add it as an *item* to the batch. After the batch item is processed, double-click the batch item to open the motion corrected image sequence in the viewer. You can then use this motion corrected image sequence for further analysis, such as *CNMF* or *CNMFE*.

See also:

This modules uses the *Batch Manager*.

Note: The parameters used for motion correction are stored in the work environment of the viewer and this log is carried over and saved in the *Project Sample* as well. To see the parameters that were used for motion correction in the viewer, execute `get_workEnv().history_trace` in the viewer console and look for the `caiman_motion_correction` entry.

14.2 Script Usage

A script can be used to add caiman motion correction batch items. This is much faster than using the GUI.

See also:

Script Editor

14.2.1 Add items

This example shows how to add all tiff files (of image sequences) from a directory as batch items with 3 different variants of parameters.

See also:

This example uses the Caiman Motion Correction Module API, ViewerWorkEnv API, and Batch Manager API

```

1  # Import glob so we can get all tiff files in a dir
2  from glob import glob
3  # Import os to get filenames from paths
4  import os
5
6  # Motion correction params.
7  # Params are identical to the GUI ones
8
9  mc_params = {
10     "max_shifts_x":      6,
11     "max_shifts_y":      6,
12     "iters_rigid":       2,
13     "name_rigid":        "Does not matter",
14     "max_dev":           3,
15     "strides":           196,
16     "overlaps":          98,
17     "upsample":          4,
18     "name_elas":         "will set later per file",
19     "output_bit_depth":  "Do not convert",
20     "gSig_filt":         (10, 10)

```

(continues on next page)

(continued from previous page)

```

21         }
22
23     # Path to the dir containing images
24     files = glob("/full_path_to_raw_images/*.tiff")
25     # Sort in alphabetical order (should also work for numbers)
26     files.sort()
27
28     # Open each file, crop, and add to batch with 3 diff mot cor params
29     for i, path in enumerate(files):
30         print("Working on file " + str(i + 1) + " / " + str(len(files)))
31
32         # get json file path for the meta data
33         meta_path = path[:-5] + ".json"
34
35         # Create a new work environment with this image sequence
36         work_env = ViewerWorkEnv.from_tiff(path, "asarray-multi", meta_path)
37
38         # Get caiman motion correction module, hide=False to not show GUI
39         mc_module = get_module("caiman_motion_correction", hide=True)
40
41         # Set name for this video file
42         name = os.path.basename(path)[:-5]
43         mc_params["name_elas"] = name
44
45         # Set the input work environment
46         mc_module.set_input_workEnv(work_env)
47
48         # First variant of params
49         mc_params["strides"] = 196
50         mc_params["overlaps"] = 98
51         # Add one variant of params for this video to the batch
52         mc_module.set_params(mc_params)
53         mc_module.add_to_batch()
54
55         # Try another variant of params
56         mc_params["strides"] = 256
57         mc_params["overlaps"] = 128
58         # Set these params and add to batch
59         mc_module.set_params(mc_params)
60         mc_module.add_to_batch()
61
62         # Try one more variant of params
63         mc_params["strides"] = 296
64         mc_params["overlaps"] = 148
65         # Set these params and add to batch
66         mc_module.set_params(mc_params)
67         mc_module.add_to_batch()
68
69     # If you want to process the batch after adding the items uncomment the following_
    ↪ lines
70     #bm = get_batch_manager()
71     #bm.process_batch(clear_viewers=True)

```

14.2.2 Crop and add items

This example shows how to crop videos prior to adding them as batch items. This is useful if you want to crop-out large unchanging regions of your movides. It uses either simple thresholding or spectral saliency on a standard deviation projection to determine the bounding box for cropping.

See also:

This example uses the Caiman Motion Correction Module API, ViewerWorkEnv API, and Batch Manager API

```

1  # Import glob so we can get all tiff files in a dir
2  from glob import glob
3  # Import os to get filenames from paths
4  import os
5
6  # Just get a shortcut reference to the auto_crop function
7  auto_crop = image_utils.auto_crop
8
9  # Parameters for cropping, these should work for everything
10 # These worked well for various different constructs
11 # If you get non-specific cropping (too much black) try "method" as "spectral_saliency
   ↳" (See below)
12 crop_params = {
13     "projection":      "max+std",
14     "method":          "threshold",
15     "denoise_params":  (32, 32),
16 }
17
18 # Spectral saliency is another method
19 # You can try and play around with the parameters
20 # If the cropping is insufficient, you can set "projection" to just "max" or "std"
21 # If you get too much junk blackness around the animal try increasing denoise_params
22 # or reduce padding. Default padding is 30 (when nothing is specified like above)
23 crop_params_salient = {
24     "projection":      "max+std",
25     "method":          "spectral_saliency",
26     "denoise_params":  (16, 16),
27     "padding":         40
28 }
29
30 # Motion correction params.
31 # Params are identical to the GUI ones
32 mc_params = {
33     "max_shifts_x":    6,
34     "max_shifts_y":    6,
35     "iters_rigid":     2,
36     "name_rigid":      "Does not matter",
37     "max_dev":         3,
38     "strides":         196,
39     "overlaps":        98,
40     "upsample":        4,
41     "name_elas":       "will set later per file",
42     "output_bit_depth": "Do not convert",
43     "gSig_filt":       (10, 10)
44 }
45
46 # Path to the dir containing images
47 files = glob("/full_path_to_raw_images/*.tiff")

```

(continues on next page)

(continued from previous page)

```

48 # Sort in alphabetical order (should also work for numbers)
49 files.sort()
50
51 # Open each file, crop, and add to batch with 3 diff mot cor params
52 for i, path in enumerate(files):
53     print("Working on file " + str(i + 1) + " / " + str(len(files)))
54
55     # get json file path for the meta data
56     meta_path = path[:-5] + ".json"
57
58     # Create a new work environment with this image sequence
59     work_env = ViewerWorkEnv.from_tiff(path, "asarray-multi", meta_path)
60
61     print("Cropping file: " + str(i + 1))
62
63     raw_seq = work_env.imgdata.seq
64     # Auto crop the image sequence
65     cropped = auto_crop.crop(raw_seq, crop_params)
66     # Set work env img seq to the cropped one and update
67     work_env.imgdata.seq = cropped
68
69     # Get caiman motion correction module, hide=False to not show GUI
70     mc_module = get_module("caiman_motion_correction", hide=True)
71
72     # Set name for this video file
73     name = os.path.basename(path)[:-5]
74     mc_params["name_elas"] = name
75
76     # Set the input work environment
77     mc_module.set_input_workEnv(work_env)
78
79     # First variant of params
80     mc_params["strides"] = 196
81     mc_params["overlaps"] = 98
82     # Add one variant of params for this video to the batch
83     mc_module.set_params(mc_params)
84     mc_module.add_to_batch()
85
86     # Try another variant of params
87     mc_params["strides"] = 256
88     mc_params["overlaps"] = 128
89     # Set these params and add to batch
90     mc_module.set_params(mc_params)
91     mc_module.add_to_batch()
92
93     # Try one more variant of params
94     mc_params["strides"] = 296
95     mc_params["overlaps"] = 148
96     # Set these params and add to batch
97     mc_module.set_params(mc_params)
98     mc_module.add_to_batch()
99
100 # If you want to process the batch after adding the items uncomment the following_
    ↪ lines
101 #bm = get_batch_manager()
102 #bm.process_batch(clear_viewers=True)

```


CNMF

Perform CNMFE using the implementation provided by the CaImAn library. This module basically provides a GUI for parameter entry.

I highly recommend going through the following before using this module

- **CNMFE builds upon CNMF** Pnevmatikakis, E. A., Gao, Y., Soudry, D., Pfau, D., Lacefield, C., Poskanzer, K., ... Paninski, L. (2014). A structured matrix factorization framework for large scale calcium imaging data analysis, 1–16.

Pnevmatikakis, E. A., Soudry, D., Gao, Y., Machado, T. A., Merel, J., Pfau, D., ... Paninski, L. (2016). Simultaneous Denoising, Deconvolution, and Demixing of Calcium Imaging Data. *Neuron*, 89(2), 285.

- **CaImAn demo notebook, the implementation in Mesmerize is basically from the demo. The second half of the notebook d**
https://github.com/flatironinstitute/CaImAn/blob/master/demos/notebooks/demo_pipeline.ipynb

CNMF-E

Input: Current Work Environment

Inspect Correlation and PNR

gaussian width of a 2D gaussian kernel, which approximates a neuron
3 times less than the average diamters of a neuron (pixels)

gSig: 14 Must be an even number

Stop here: Enter name Add to batch

CNMF-E

Ain:

Minimum correlation of peak

min_corr: 0.890

Minimum peak to noise ratio

min_pnr: 4

Adaptive way to set threshold on the transient size

min_SNR: 1

Threshold on space consistency
If lower more components will be accepted, potentially with worse quality

r_values_min: 0.70

Average decay time of calcium spikes (seconds)

decay_time: 4.00

Half size of patch

rf: 50

Overlap of patches (at least 4 times the size of a neuron/cell)

overlap: 30

Global number of background components

gnb: 10

Background components per patch

nb_patch: 10

Number of neurons/cell per patch

k: 10

Perform CNMF-E: Enter name Add to batch

Export ParametersImport Parameters

Parameters

Please see the CaImAn demo notebook mentioned above to understand the parameters.

15.1 Console

A script can be used to add CNMFE batch items. This is much faster than using the GUI. This example sets the work environment from the output of a batch item. See the *Caiman Motion Correction script usage examples* for how to load images if you want to add CNMF items from images that are not in a batch.

See also:

Script Editor, :ref:`CNMF module API`

```

1  # CNMF Params that we will use for each item
2  params = {'Input': 'Current Work Environment',
3           'p': 2,
4           'gnb': 1,
5           'merge_thresh': 0.25,
6           'rf': 70,
7           'stride_cnmf': 40,
8           'k': 16,
9           'gSig': 8,
10          'min_SNR': 2.5,
11          'rval_thr': 0.8,
12          'cnn_thr': 0.8,
13          'decay_time': 20,
14          'name_cnmf': 'set_later_per_file',
15          'refit': True
16          }
17
18  # Get the batch manager
19  bm = get_batch_manager()
20  cnmf_mod = get_module('cnmf')
21
22  # Start index if we want to start processing the new items after they have been added
23  start_ix = bm.df.index.size + 1
24
25  for ix, r in bm.df.iterrows():
26      # Use output of items 6 - 12
27      if ix < 6:
28          continue
29      if ix > 12:
30          break
31
32      # Get the name of the mot cor item
33      name = r['name']
34
35      # Set the name for the new cnmf item
36      params['name_cnmf'] = name
37
38      # Load the mot cor output
39      bm.load_item_output(module='caiman_motion_correction', viewers=viewer, UUID=r[
40      ↪ 'uuid'])
41
42      # Set the CNMF params and add to batch
43      cnmf_mod.set_params(params)

```

(continues on next page)

(continued from previous page)

```
43     cnmf_mod.add_to_batch_cnmf()
44
45     # Cleanup the work environment
46     vi._clear_workEnv()
47
48     # Uncomment the last two lines to start the batch as well
49     #bm.process_batch(start_ix, clear_viewers=True)
```

CNMFE

Perform CNMFE using the implementation provided by the CaImAn library.

I highly recommend going through the following before using this module

- **The paper on CNMF-E** Zhou, P., Resendez, S. L., Rodriguez-Romaguera, J., Jimenez, J. C., Neufeld, S. Q., Stuber, G. D., ... Paninski, L. (2016). Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data. *ELife*, 1–37.
- **CNMFE builds upon CNMF** Pnevmatikakis, E. A., Gao, Y., Soudry, D., Pfau, D., Lacefield, C., Poskanzer, K., ... Paninski, L. (2014). A structured matrix factorization framework for large scale calcium imaging data analysis, 1–16.

Pnevmatikakis, E. A., Soudry, D., Gao, Y., Machado, T. A., Merel, J., Pfau, D., ... Paninski, L. (2016). Simultaneous Denoising, Deconvolution, and Demixing of Calcium Imaging Data. *Neuron*, 89(2), 285.
- **CaImAn CNMF-E demo notebook, the implementation in Mesmerize is basically from the demo**
https://github.com/flatironinstitute/CaImAn/blob/master/demos/notebooks/demo_pipeline_cnmfE.ipynb

CNMF-E

Input: Current Work Environment

Inspect Correlation and PNR

gaussian width of a 2D gaussian kernel, which approximates a neuron
3 times less than the average diamters of a neuron (pixels)

gSig: 14 Must be an even number

Stop here: Enter name Add to batch

CNMF-E

Ain:

Minimum correlation of peak

min_corr: 0.890

Minimum peak to noise ratio

min_pnr: 4

Adaptive way to set threshold on the transient size

min_SNR: 1

Threshold on space consistency
If lower more components will be accepted, potentially with worse quality

r_values_min: 0.70

Average decay time of calcium spikes (seconds)

decay_time: 4.00

Half size of patch

rf: 50

Overlap of patches (at least 4 times the size of a neuron/cell)

overlap: 30

Global number of background components

gnb: 10

Background components per patch

nb_patch: 10

Number of neurons/cell per patch

k: 10

Perform CNMF-E: Enter name Add to batch

Export ParametersImport Parameters

Parameters

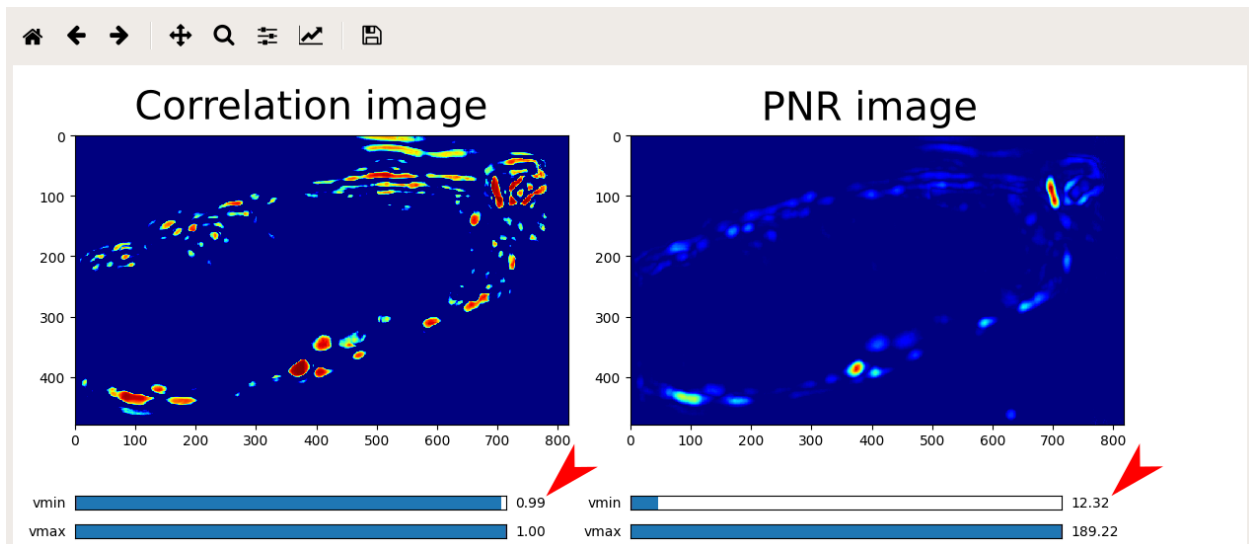
Please see the CaImAn demo notebook mentioned above to understand the parameters.

Ain: Seed spatial components from another CNMFE item by entering its UUID here.

16.1 Usage

This module creates two types of batch items, one where you can inspect the Correlation & PNR images and another that performs CNMFE and extracts components. Here is an outline of typical usage:

- Enter a *gSig* parameter value and a name for “Inspect Correlation and PNR”, the text entry for “Stop here”. Click “Add to batch”. Run the batch item.
- Double-click the batch item, you will be presented with a GUI to help optimize *min_corr* and *min_pnr*. For the correlation image use the *vmin* slider to optimize the separation of cells and set the *min_corr* parameter to this value. Likewise, optimize the value for the PNR until the PNR image mostly contains regions that show real signal and no or few regions that are likely to be just noise and set this *vmin* value as the *min_pnr* parameter. You may need to try slightly different variations to optimize the parameters.



- Enter the rest of the parameters and give a name under “Perform CNMF-E”, click “Add to batch” and run the item.
- Double-click the batch item and you will be presented with 3 options. The first option will display the correlation-pnr images and the second option is currently non-functional (matplotlib Qt issue). The last option will import the components extracted by CNMFE into an open Viewer. The components are managed by the ROI Manager.

See also:

[ROI Manager](#)

See also:

This modules uses the [Batch Manager](#).

Note: The parameters used for CNMFE are stored in the work environment of the viewer and this log is carried over and saved in Project Samples as well. To see the parameters that were used for CNMFE in the viewer, execute

`get_workEnv().history_trace` in the viewer console and look for the 'cnmfe' entry.

16.2 Script Usage

A script can be used to add CNMFE batch items. This is much faster than using the GUI.

See also:

Script Editor.

16.2.1 Add Corr PNR items

Add Corr PNR batch items from a batch that contains motion corrected items. This example add 2 variants of parameters (just gSig) for each motion corrected item.

See also:

This example uses the Caiman CNMFE module API and Batch Manager API

See also:

Caiman Motion Correction script usage examples for how to load images if you want to add Corr PNR items from images that are not in a batch.

```
1  # Get the batch manager
2  bm = get_batch_manager()
3
4  # Get the CNMFE module
5  cnmfe_mod = get_module('cnmfe', hide=True)
6
7  # Start index to start processing the new items after they have been added
8  start_ix = bm.df.index.size + 1
9
10 for ix, r in bm.df.iterrows():
11     if ix == start_ix:
12         break
13
14     # Get the name of the mot cor item
15     name = r['name']
16
17     # Load the output of the motion corrected batch item
18     # The output will load into the viewer that this script
19     # is running in.
20     bm.load_item_output(module='caiman_motion_correction', viewers=viewer, UUID=r[
21 ↪ 'uuid'])
22
23     # Get the currently set params
24     # You just need the dict with all the correct keys
25     # You will just modify the "gSig" and "name_corr_pnr" keys
26     params = cnmfe_mod.get_params()
27
28     # Set the gSig and name params
29     params['gSig'] = 8
30     params['name_corr_pnr'] = name
```

(continues on next page)

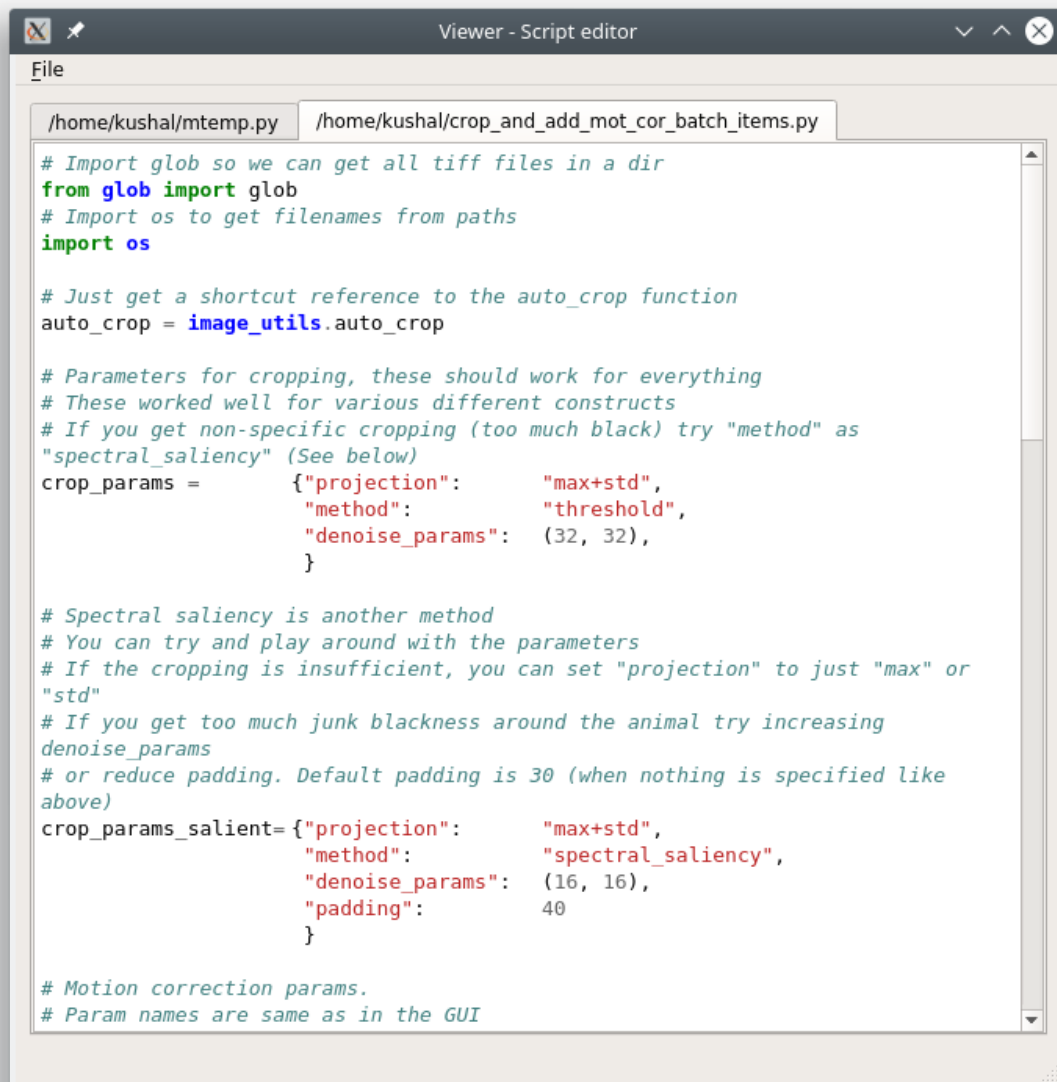
(continued from previous page)

```
31     # Set the params and add to batch
32     cnmfe_mod.set_params(params)
33     cnmfe_mod.add_to_batch_corr_pnr()
34
35     # Another variant of params
36     # Set the gSig and name params
37     params['gSig'] = 10
38     params['name_corr_pnr'] = name
39
40     # Set the params and add to batch
41     cnmfe_mod.set_params(params)
42     cnmfe_mod.add_to_batch_corr_pnr()
43
44     # Cleanup the work environment
45     vi._clear_workEnv()
46
47     # Start the batch from the start_ix
48     bm.process_batch(start_ix, clear_viewers=True)
```


SCRIPT EDITOR

A simple text editor for writing scripts that can be run in the *viewer console*

The scripts are simply ran in the *viewer console* and all output will also be visible in the *viewer console*.



The screenshot shows a window titled "Viewer - Script editor" with a menu bar containing "File". Below the menu bar are two tabs: "/home/kushal/mtemp.py" and "/home/kushal/crop_and_add_mot_cor_batch_items.py". The active tab displays a Python script with the following content:

```
# Import glob so we can get all tiff files in a dir
from glob import glob
# Import os to get filenames from paths
import os

# Just get a shortcut reference to the auto_crop function
auto_crop = image_utils.auto_crop

# Parameters for cropping, these should work for everything
# These worked well for various different constructs
# If you get non-specific cropping (too much black) try "method" as
"spectral_saliency" (See below)
crop_params = {
    "projection": "max+std",
    "method": "threshold",
    "denoise_params": (32, 32),
}

# Spectral saliency is another method
# You can try and play around with the parameters
# If the cropping is insufficient, you can set "projection" to just "max" or
"std"
# If you get too much junk blackness around the animal try increasing
denoise_params
# or reduce padding. Default padding is 30 (when nothing is specified like
above)
crop_params_salient = {
    "projection": "max+std",
    "method": "spectral_saliency",
    "denoise_params": (16, 16),
    "padding": 40
}

# Motion correction params.
# Param names are same as in the GUI
```

See also:

Viewer Core API

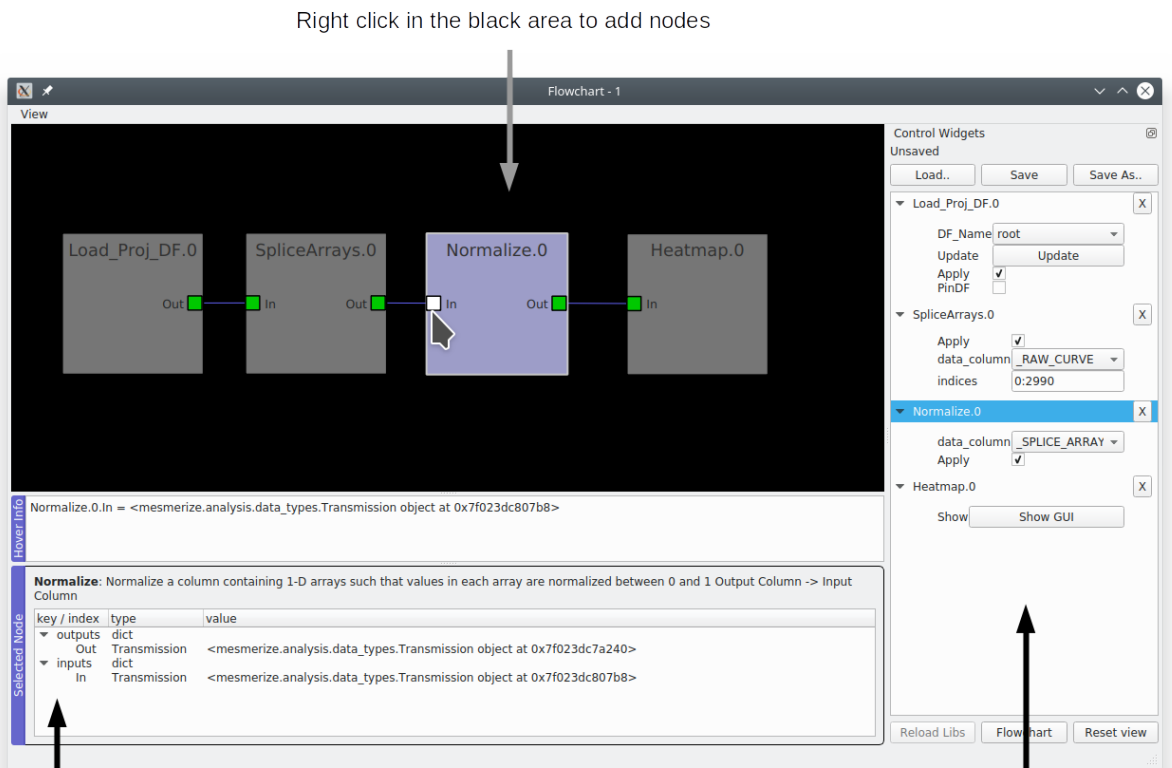
Warning: There is no auto-save function

FLOWCHART OVERVIEW

The flowchart allows you to analyze samples in your project and create plots by arranging analysis nodes. Each node takes an input, performs an operation, and produces an output. For example the *Derivative* node takes use-specified numerical arrays, computes the derivative of these arrays, and then outputs the result.

The Flowchart is based on the [pyqtgraph flowchart widgets](#)

Flowchart Window



Add node: Right click -> Add node -> Choose from selection

Click on a node to highlight the Control Widget

Remove node: Right click -> Remove node

Connecting nodes: Click on a node terminal and drag to another terminal

Save the flowchart layout: Click “Save as...” to save the layout to a new file. You must specify the file extension as “.fc”. If you save this file within the “*flowcharts*” *directory* of your project it will show up in the [Welcome Window](#) when you open your project.

Note: This does not save the data, use the [Save](#) node to save data.

Warning: Due to a weird Qt or pyqtgraph bug certain parameter values (such as those in drop-down menus) can’t be saved. Similarly, parameters values are lost when you save to an existing .fc file. If you’re interested take a look at `pyqtgraphCore.WidgetGroup`. Anyways you shouldn’t be using the flowchart layout to save this information, that’s what the History Trace in Transmission objects is for.

Load an .fc file: Click the “Load” button.

Reset View button: Reset the view, for example if you zoom out or pan too far.

18.1 Transmission

API Reference

Almost every node uses a Transmission object for input and output. A Transmission is basically a DataFrame and a History Trace (analysis log) of the data within the DataFrame.

Transmission DataFrame

The Transmission DataFrame is created from your ProjectDataFrame (or sub-DataFrame) by the `node_Load_Proj_DF` node. This initial DataFrame will contain the same columns as your Project DataFrame, and a new column named **_RAW_CURVE**. Each element (row) in the **_RAW_CURVE** column is a 1-D numerical array representing a single raw curve extracted from an ROI.

A new column named **_BLOCK_** is also added which contains the [UUID](#) for logging the analysis history of this newly created block of DataFrame rows, known as a *data block*. This allows you to merge Transmissions (see [Merge node](#)) and maintain their independent analysis logs prior to the merge.

Naming conventions for DataFrame columns according to the data types

- *numerical data*: single leading underscore (_). All caps if produced by a flowchart node.
- *categorical data*: no leading underscore. All caps if produced by flowhchart node.
- *special cases*: Peak detection data are placed in a column named **peaks_bases** where each element is a DataFrame.
- *uuid data*: has uuid or UUID in the name

Note: **_BLOCK_** is an exception, it contains UUIDs not numerical data.

History Trace

The History Trace of a Transmission is a log containing the discrete analysis steps, known as operations, along with their parameters and any other useful information. When a flowchart node performs an oper-

ation it stores the output(s) data in the Transmission DataFrame and appends the operation parameters to this log. A separate log is kept for each data block present in the Transmission DataFrame.

18.2 Console

You have direct access to the data within the nodes through the console in the flowchart. To show the console go to View -> Console.

See also:

If you are unfamiliar with the console see the overview on [Consoles](#)

Call `get_nodes()` to view a dict of all nodes in the flowchart. You can access the output Transmission in most nodes through the attribute `t`. You can access the transmission dataframe through `t.df`.

See also:

See the Transmission API for more information. Sources for the nodes at `mesmerize/pyqtgraphCore/flowchart/library`.

Example, directly accessing DataFrame elements through the flowchart console

The screenshot shows the Mesmerize Flowchart - 2 interface. The main window displays a flowchart with the following nodes and connections:

- LoadFile.0** (Output) connects to **ButterWorth.0** (Input).
- ButterWorth.0** (Output) connects to **Normalize.0** (Input) and **ScalerMeanVar.0** (Input).
- Normalize.0** (Output) connects to **Heatmap.0** (Input).

The **Control Widgets** panel on the right shows the configuration for the selected **Normalize.0** node:

- LoadFile.0**: load_trn (Open .trn File), fname (jul_7_2019_peaks_dat), proj_path (Project Path), proj_path/cell_types_apr_2019.
- ButterWorth.0**: data_column (_STATIC_DF_C), order (2), freq_divisor (2.00), Apply (checked).
- Normalize.0**: data_column (_BUTTERWOR), Apply (checked).

The **Console** window at the bottom shows the following code and output:

```
Namespaces:
pandas as 'pd'
numpy as 'np'
self as 'this'
'get_nodes()' to get a dict of all nodes

get_nodes()['Normalize.0'].t
<mesmerize.analysis.data_types.Transmission object at 0x7f675e1f21d0>

get_nodes()['Normalize.0'].t.df.iloc[100]._BUTTERWORTH
array([2.84715414, 2.97081577, 3.09908962, ..., 0.42041581, 0.4221815 ,
       0.42370578])
```

18.3 Transmission Files

You can save a Transmission files using the *Save node* and work with the data directly in scripts, jupyter notebooks etc. You can also save them through the flowchart console (and plot consoles) through `Transmission.to_hdf5`.

18.3.1 Load Transmission

Load a saved Transmission instance using `Transmission.from_hdf5`

```

1 >>> from mesmerize import Transmission
2 >>> from uuid import UUID
3
4 >>> t = Transmission.from_hdf5('/share/data/temp/kushal/data.trn')
5 <mesmerize.analysis.data_types.Transmission at 0x7f4d42f386a0>
6
7 # The DataFrame is always the 'df' attribute
8 >>> t.df.head()
9
10                                     CurvePath  ...  FCLUSTER_LABELS
11 0  curves/a2-_-1-_-843c2d43-75f3-421a-9fef-483d1e...  ...              8
12 1  curves/brn3b_a6-_-2-_-21557a64-6868-4ff4-8db1-...  ...              4
13 2  curves/brn3b_a6-_-2-_-21557a64-6868-4ff4-8db1-...  ...              5
14 3  curves/brn3b_day1_3-_-2-_-ff3e95df-0e15-495c-9...  ...              8
15 4  curves/brn3b_day1_3-_-2-_-ff3e95df-0e15-495c-9...  ...              6
16
17 [5 rows x 27 columns]
```

18.3.2 View History Log

Transmissions have a *history_trace* attribute which is an instance of `HistoryTrace`.

Use the `get_data_block_history` and `get_operations_list` methods to view the history log of a data block.

```

1 # To view the history log, first get the block UUID of the dataframe row of which you
2   ↳ want the history log
3
4 # Block UUIDs are stored in the _BLOCK_ column
5 >>> bid = t.df.iloc[10]._BLOCK_
6 >>> bid
7
8 '248a6ece-e60e-4a09-845e-188a5199d262'
9
10 # Get the history log of this data block
11 # HistoryTrace.get_operations_list() returns a list of operations, without parameters
12 # HistoryTrace.get_data_block_history() returns the operations list with the
13   ↳ parameters
14 >>> t.history_trace.get_operations_list(bid)
15
16 ['spawn_transmission',
17  'splice_arrays',
18  'normalize',
19  'rfft',
20  'absolute_value',
```

(continues on next page)

(continued from previous page)

```

19 'log_transform',
20 'splice_arrays',
21 'fcluster']
22
23 # View the entire history log with all params
24 >>> t.history_trace.get_data_block_history(bid)
25
26 [{'spawn_transmission': {'sub_dataframe_name': 'neuronal',
27 'dataframe_filter_history': {'dataframe_filter_history': ['df[~df["promoter"].isin([\
↪ 'cesa\', \'hnk1\'])]',
28 'df[~df["promoter"].isin([\\'cesa\', \'hnk1\'])]',
29 'df[~df["cell_name"].isin([\\'not_a_neuron\', \'non_neuronal\', \'untagged\', \'
↪ 'ependymal\'])]']}]},
30 {'splice_arrays': {'data_column': '_RAW_CURVE',
31 'start_ix': 0,
32 'end_ix': 2990,
33 'units': 'time'}},
34 {'normalize': {'data_column': '_SPLICE_ARRAYS', 'units': 'time'}},
35 {'rfft': {'data_column': '_NORMALIZE',
36 'frequencies': [0.0,
37 0.0033444816053511705,
38 0.0033444816053511705,
39 0.006688963210702341,
40 ...
41
42 # Get the parameters for the 'fcluster' operation
43 >>> fp = t.history_trace.get_operation_params(bid, 'fcluster')
44
45 # remove the linkage matrix first so we can view the other params
46 >>> fp.pop('linkage_matrix');fp
47
48 {'threshold': 8.0,
49 'criterion': 'maxclust',
50 'depth': 1,
51 'linkage_params': {'method': 'complete',
52 'metric': 'wasserstein',
53 'optimal_ordering': True}}

```


NODES

19.1 Data

These nodes are for performing general data related operations

19.1.1 LoadFile

Source

Loads a save Transmission file. If you have a Project open it will automatically set the project path according to the open project. Otherwise you must specify the project path. You can specify a different project path to the project that is currently open (this is untested, weird things could happen). You should not merge Transmissions originating from different projects.

Note: You can also load a saved Transmission file by dragging & dropping it into the Flowchart area. It will create a LoadFile node with the name of the dropped.

Terminal	Description
Out	Transmission loaded from the selected file.

Parameters	Description
load_trn	Button to choose a .trn file (Transmission) to load
proj_trns	Load transmission file located in the project's "trns" directory
proj_path	Button to select the Mesmerize project that corresponds to the chosen .trn file.

Note: The purpose of specifying the Project Path when you load a save Transmission file is so that interactive plots and the *Datapoint Tracer* can find raw data that correspond to datapoints.

19.1.2 LoadProjDF

Source

Load the entire Project DataFrame (root) of the project that is currently open, or a sub-DataFrame that corresponds a tab that you have created in the *Project Browser*.

Output Data Column (*numerical*): `_RAW_CURVE`

Each element in this output column contains a 1-D array representing the trace extracted from an ROI.

Terminal	Description
Out	Transmission created from the Project DataFrame or sub-DataFrame.

Parameters	Description
DF_Name	DataFrame name. List corresponds to <i>Project Browser</i> tabs.
Update	Re-create Transmission from corresponding <i>Project Browser</i> tab.
Apply	Process data through this node

Note: The **DF_Name** options do not update live with the removal or creation of tabs in the *Project Browser*, you must create a new node to reflect these types of changes.

19.1.3 Save

Source

Save the input Transmission to a file so that the Transmission can be used re-loaded in the Flowchart for later use.

Usage: Connect an input Transmission to this node's **In** terminal, click the button to choose a path to save a new file to, and then click the Apply checkbox to save the input Transmission to the chosen file.

Terminal	Description
In	Transmission to be saved to file

Parameters	Description
saveBtn	Button to choose a filepath to save the Transmission to.
Apply	Process data through this node

Note: You must always save a Transmission to a new file (pandas with hdf5 exhibits weird behavior if you overwrite, this is the easiest workaround). If you try to overwrite the file you will be presented with an error saying that the file already exists.

19.1.4 Merge

Source

Merge multiple Transmissions into a single Transmission. The DataFrames of the individual Transmissions are concatenated using `pandas.concat` and History Traces are also merged. The History Trace of each individual input Transmission is kept separately.

Warning: At the moment, if you create two separate data streams that originate from the same Transmission and then merge them at a later point, the analysis log (History Trace) of the individual data streams are not maintained. See the information about data blocks in the *Transmission*.

Terminal	Description
In	Transmissions to be merged
Out	Merged Transmission

19.1.5 ViewTransmission

Source

View the input Transmission object using the spyder Object Editor. For example you can explore the Transmission DataFrame and HistoryTrace.

19.1.6 ViewHistory

Source

View the HistoryTrace of the input Transmission in a nice Tree View GUI.

19.1.7 TextFilter

Source

Include or Exclude Transmission DataFrame rows according to a text filter in a categorical column.

Usage Example: If you want to select all traces that are from photoreceptor cells and you have a categorical column, named `cell_types` for example, containing cell type labels, choose “`cell_type`” as the *Column* parameter and enter “`photoreceptor`” as the *filter* parameter, and select *Include*. If you want to select everything that are not photoreceptors select *Exclude*.

Note: It is recommended to filter and group your data beforehand using the *Project Browser* since it allows much more sophisticated filtering.

Terminal	Description
In	Input Transmission
Out	Transmission its DataFrame filtered accoring parameters

Parameters	Description
Column	Categorical column that contains the text filter to apply
filter	Text filter to apply
Include	Include all rows matching the text filter
Exclude	Exclude all rows matching the text filter
Apply	Process data through this node

HistoryTrace output structure: Dict of all the parameters for this node

19.1.8 SpliceArrays

Source

Splice arrays derived in the specified numerical data column and place the spliced output arrays in the output column.

Output Data Column (*numerical*): `_SPLICE_ARRAYS`

Terminal	Description
In	Input Transmission
Out	Transmission with arrays from the input column spliced and placed in the output column

Parameters	Description
data_column	Numerical data column containing the arrays to be spliced
indices	The splice indices, “start_index:end_index”
Apply	Process data through this node

19.1.9 DropNa

Source

Drop NaNs and Nones (null) from the Transmission DataFrame. Uses `DataFrame.dropna` and `DataFrame.isna` methods.

- If you choose “row” or “column” as axis, entire rows or columns will be dropped if any or all (see params) of the values are NaN/None.
- If you choose to drop NaNs/Nones according to a specific column, it will drop the entire row if that row has a NaN/None value for the chosen column.

Terminal	Description
In	Input Transmission
Out	Transmission NaNs and None’s removed according to the params

Parameters	Description
axis	Choose to rows, columns, or a rows according to a specific column.
how	<p><i>any</i>: Drop if any value in the row/column is NaN/None</p> <p><i>all</i>: Drop only if all values in the row/column are Nan/None</p> <p>ignored if “axis” parameter is set to a specific column</p>
Apply	Process data through this node

19.1.10 NormRaw

Source

Scale the raw data such that the min and max values are set to the min and max values derived from the raw spatial regions of the image sequences they originate from. Only for CNMFE data.

The arrays in the `_RAW_CURVE` column are scaled and the output is placed in a new column named `_NORMRAW`

Terminal	Description
In	Input Transmission
Out	Transmission with the result placed in the output column

Parameter	Description
option	<p>Derive the raw min & max values from one of the following options:</p> <p><i>top_5</i>: Top 5 brightest pixels</p> <p><i>top_10</i>: Top 10 brightest pixels</p> <p><i>top_5p</i>: Top 5% of brightest pixels</p> <p><i>top_10p</i>: Top 10% of brightest pixels</p> <p><i>top_25p</i>: Top 25% of brightest pixels</p> <p><i>full_mean</i>: Full mean of the min and max array</p>
Apply	Process data through this node

Note: If the raw min value is higher than the raw max value the curve will be excluded in the output. You will be presented with a warning box with the number of curves that were excluded due to this.

19.2 Display

These nodes connect input **Transmission(s)** to various plots for visualization

The actual Plot Widget instance that these nodes use can be accessed through the `plot_widget` attribute in the flowchart console.

For example

```
# Get a heatmap node that is named "Heatmap.0"
>>> hn = get_nodes() ['Heatmap.0']

# the plot widget instance
>>> hn.plot_widget

<mesmerize.plotting.widgets.heatmap.widget.HeatmapTracerWidget object at 0x7f26e5d29678>
```

19.2.1 BeeswarmPlots

Source

Based on ppytgraph Beeswarm plots.

Visualize data points as a pseudoscatrer and as corresponding Violin Plots. This is commonly used to visualize peak features and compare different experimental groups.

For information on the plot widget see [Beeswarm Plots](#)

Terminal	Description
In	Input Transmission The DataFrame column(s) of interest must have single numerical values, not arrays

19.2.2 Heatmap

Source

Used for visualizing numerical arrays in the form of a heatmap. Also used for visualizing a hieararchical clustering tree (dendrogram) along with a heatmap with row order corresponding to the order leaves of the dendrogram.

For information on the plot widget see [Heat Plot](#)

Terminal	Description
In	Input Transmission The arrays in the DataFrame column(s) of interest must be of the same length

Note: Arrays in the DataFrame column(s) of interest **must** be of the same length. If they are not, you must splice them using the *SpliceArrays* node.

19.2.3 CrossCorr

Source

Perform Cross-Correlation analysis. For information on the plot widget see *CrossCorrelation Plot*

19.2.4 Plot

Source

For information on the plot widget see <plot_SimplePlot>

A simple plot.

Terminal	Description
In	Input Transmission

Parameters	Description
data_column	Data column to plot, must contain numerical arrays
Show	Show/hide the plot window
Apply	Process data through this node

19.2.5 Proportions

Source

Plot stacked bar chart of one categorical variable vs. another categorical variable.

For information on the plot widget see *Proportions Plot*

19.2.6 ScatterPlot

Source

Create scatter plot of numerical data containing [X, Y] values

For information on the plot widget see *Scatter Plot*

19.2.7 TimeSeries

Plot the means along with confidence intervals or standard deviation of numerical arrays representing time series data.

For more information see `plot_TimeSeries`

19.3 Signal

Routine signal processing functions

I recommend this book by Tom O'Haver if you are unfamiliar with basic signal processing: <https://terpconnect.umd.edu/~toh/spectrum/TOC.html>

19.3.1 Butterworth

Source

Creates a Butterworth filter using `scipy.signal.butter` and applies it using `scipy.signal.filtfilt`.

The `Wn` parameter of `scipy.signal.butter` is calculated by dividing the sampling rate of the data by the `freq_divisor` parameter (see below).

Output Data Column (*numerical*): `_BUTTERWORTH`

Terminal	Description
In	Input Transmission
Out	Transmission with filtered signals in the output data column

Parameters	Description
<code>data_column</code>	Data column containing numerical arrays to be filtered
<code>order</code>	Order of the filter
<code>freq_divisor</code>	Divisor for dividing the sampling frequency of the data to get <code>Wn</code>
Apply	Process data through this node

19.3.2 SavitzkyGolay

Source

Savitzky Golay filter. Uses `scipy.signal.savgol_filter`.

Output Data Column (*numerical*): `_SAVITZKY_GOLAY`

Terminal	Description
In	Input Transmission
Out	Transmission with filtered signals in the output data column

Parameters	Description
data_column	Data column containing numerical arrays to be filtered
win-dow_length	Size of windows for fitting the polynomials. Must be an odd number.
polyorder	Order of polynomials to fit into the windows. Must be less than <i>win-dow_length</i>
Apply	Process data through this node

19.3.3 PowSpecDens

19.3.4 Resample

Source

Resample the data in numerical arrays. Uses [scipy.signal.resample](#).

Output Data Column (*numerical*): _RESAMPLE

Terminal	Description
In	Input Transmission
Out	Transmission with resampled signals in the output data column

Parameters	Description
data_column	Data column containing numerical arrays to be resampled
Rs	New sampling rate in <i>Tu</i> units of time.
Tu	Time unit
Apply	Process data through this node

Note: If $Tu = 1$, then Rs is the new sampling rate in Hertz.

19.3.5 ScalerMeanVariance

Source

Uses [tslearn.preprocessing.TimeSeriesScalerMeanVariance](#)

Output Data Column (*numerical*): _SCALER_MEAN_VARIANCE

Terminal	Description
In	Input Transmission
Out	Transmission with scaled signals in the output column

Parameters	Description
data_column	Data column containing numerical arrays to be scaled
mu	Mean of the output time series
std	Standard Deviation of the output time series
Apply	Process data through this node

Note: if $\mu = 0$ and $\sigma = 1$, the output is the z-score of the signal.

19.3.6 Normalize

Source

Normalize the signal so that all values are between 0 and 1 based on the min and max of the signal.

Output Data Column (*numerical*): `_NORMALIZE`

Terminal	Description
In	Input Transmission
Out	Transmission with scaled signals in the output column

Parameters	Description
<code>data_column</code>	Data column containing numerical arrays to be scaled
Apply	Process data through this node

19.3.7 RFFT

Source

Uses `scipy.fftpack.rfft`. “Discrete Fourier transform of a real sequence”

Output Data Column (*numerical*): `_RFFT`

Terminal	Description
In	Input Transmission
Out	Transmission with the RFT of signals in the output column

Parameters	Description
<code>data_column</code>	Data column containing numerical arrays
Apply	Process data through this node

19.3.8 IRFFT

Source

Uses `scipy.fftpack.irfft`. “inverse discrete Fourier transform of real sequence x”

Output Data Column (*numerical*): `_IRFFT`

19.3.9 PeakDetect

Source

Simple Peak Detection using derivatives. The “Differentiation” chapter of Tom O’Haver’s book has a section on Peak Detection which I recommend reading. <https://terpconnect.umd.edu/~toh/spectrum/TOC.html>

Output Data Column (*DataFrame*): peaks_bases

See also:

Peak Editor GUI

Terminal	Description
Derivative	Transmission with derivatives of signals. Must have _DERIVATIVE column. It’s recommended to use a derivative from a normalized filtered signal.
Normalized	Transmission containing Normalized signals, used for thresholding See <i>Normalize</i> node
Curve	Transmission containing original signals. Usually not filtered to avoid distortions caused by filtering
PB_Input (<i>optional</i>)	Transmission containing peaks & bases data (peaks_bases column). Useful for visualizing a saved Transmission that has peaks & bases data
Out	Transmission with the detected peaks & bases as DataFrames in the output column

Warning: The *PB_Input* terminal overrides all other terminals. Do not connect inputs to *PB_Input* and other terminals simultaneously.

Parameter	Description
data_column	Data column of the input <i>Curve</i> Transmission for placing peaks & bases onto
Fictional_Bases	Add bases to beginning and end of signal if first or last peak is lonely
Edit	Open Peak Editor GUI, see <i>Peak Editor</i>
SlopeThr	Slope threshold
AmplThrAbs	Absolute amplitude threshold
AmplThrRel	Relative amplitude threshold
Apply	Process data through this node

19.3.10 PeakFeatures

Source

Compute peak features. The DataFrame of the output Transmission contains one row for each peak.

Output Data Column	Description
<code>_pf_peak_curve</code>	array representing the peak
<code>_pf_ampl_rel_b_ix_l</code>	peak amplitude relative to its left base
<code>_pf_ampl_rel_b_ix_r</code>	peak amplitude relative to its right base
<code>_pf_ampl_rel_b_mean</code>	peak amplitude relative to the mean of its bases
<code>_pf_ampl_rel_zero</code>	peak amplitude relative to zero
<code>_pf_area_rel_zero</code>	Simpson's Rule Integral of the curve
<code>_pf_area_rel_min</code>	Simpson's Rule Integral relative to the minimum value of the curve Subtracts the minimum values of the peak curve before computing the integral
<code>_pf_rising_slope_avg</code>	slope of the line drawn from the left base to the peak
<code>_pf_falling_slope_avg</code>	slope of the line drawn from the right base to the peak
<code>_pf_duration_base</code>	distance between the left and right base
<code>_pf_p_ix</code>	index of the peak maxima in the parent curve
<code>_pf_uuid</code>	peak UUID
<code>_pf_b_ix_l</code>	index of the left base in the parent curve
<code>_pf_b_ix_r</code>	index of the right base in the parent curve

See also:

`mesmerize/analysis/compute_peak_features` for the code that computes the peak features.

Terminal	Description
In	Input Transmission. Must contain <i>peak_bases</i> column that contains peak_bases DataFrames.
Out	Transmission with peak features in various output columns

Parameter	Description
<code>data_column</code>	Data column containing numerical arrays from which to compute peak features.
Apply	Process data through this node

19.4 Math

Nodes for performing basic Math functions

19.4.1 Derivative

Source

Computes the first derivative.

Output Data Column (*numerical*): `_DERIVATIVE`

Terminal	Description
In	Input Transmission
Out	Transmission with the derivative placed in the output column

Parameter	Description
<code>data_column</code>	Data column containing numerical arrays
Apply	Process data through this node

19.4.2 TVDiff

Source

Based on [Numerical Differentiation of Noisy, Nonsmooth Data](#). Rick Chartrand. (2011)..
Translated to Python by Simone Sturniolo.

19.4.3 XpowerY

Source

Raises each element of the numerical arrays in the `data_column` to the exponent `Y`

Output Data Column (*numerical*): `_X_POWER_Y`

Terminal	Description
In	Input Transmission
Out	Transmission with the result placed in the output column

Parameter	Description
<code>data_column</code>	Data column containing numerical arrays
<code>Y</code>	Exponent
Apply	Process data through this node

19.4.4 AbsoluteValue

Source

Element-wise absolute values of the input arrays. Computes root mean squares if input arrays are complex.

Output Data Column (*numerical*): `_ABSOLUTE_VALUE`

Terminal	Description
In	Input Transmission
Out	Transmission with the result placed in the output column

Parameter	Description
<code>data_column</code>	Data column containing numerical arrays
Apply	Process data through this node

19.4.5 LogTransform

Source

Perform Logarithmic transformation of the data.

Output Data Column (*numerical*): `_LOG_TRANSFORM`

Terminal	Description
In	Input Transmission
Out	Transmission with the result placed in the output column

Parameter	Description
<code>data_column</code>	Data column containing numerical arrays
<code>transform</code>	<i>log10</i> : Base 10 logarithm <i>ln</i> : Natural logarithm <i>modlog10</i> : $\text{sign}(x) * \log_{10}(x + 1)$ <i>modln</i> : $\text{sign}(x) * \ln(x + 1)$
Apply	Process data through this node

19.4.6 ArrayStats

Source

Perform a few basic statistical functions.

Output Data Column (*numerical*): Customizable by user entry

Output data are single numbers, not arrays

Terminal	Description
In	Input Transmission
Out	Transmission with the result placed in the output column

The desired function is applied to each 1D array in the *data_column* and the output is placed in the Output Data Column.

Parameter	Description
<i>data_column</i>	Data column containing numerical arrays
<i>function</i>	<p><i>amin</i>: Return the minimum of the input array</p> <p><i>amax</i>: Return the maximum of the input array</p> <p><i>nanmin</i>: Return the minimum of the input array, ignore NaNs</p> <p><i>nanmax</i>: Return the maximum of the input array, ignore NaNs</p> <p><i>ptp</i>: Return the range (max - min) of the values of the input array</p> <p><i>median</i>: Return the median of the input array</p> <p><i>mean</i>: Return the mean of the input array</p> <p><i>std</i>: Return the standard deviation of the input array</p> <p><i>var</i>: Return the variance of the input array</p> <p><i>nanmedian</i>: Return the median of the input array, ignore NaNs</p> <p><i>nanmean</i>: Return the mean of the input array, ignore NaNs</p> <p><i>nanstd</i>: Return the standard deviation of the input array, ignore NaNs</p> <p><i>nanvar</i>: Return the variance of the input array, ignore NaNs</p>
<i>group_by (Optional)</i>	Group by a categorical variable, for example get the mean array of a group
<i>group_by_sec (Optional)</i>	Group by a secondary categorical variable
<i>output_col</i>	Enter a name for the output column
Apply	Process data through this node

19.4.7 ArgGroupStat

Source

Group by a categorical variable and return the value of any other column based on a statistic. Basically creates sub-dataframes for each group and then returns based on the sub-dataframe.

Group by column “group_by” and return value from column “return_col” where data in *data_column* fits “stat”

Output Data Column (Any): ARG_STAT

Terminal	Description
In	Input Transmission
Out	Transmission with the result placed in the output column

Parameter	Description
data_column	Data column containing single numbers (not arrays for now)
group_by	Group by column (categorical variables)
return_col	Return value from this column (any data)
stat	“max” or “min”
Apply	Process data through this node

19.4.8 ZScore

Source

Compute Z-Scores of the data. Uses `scipy.stats.zscore`. The input data are divided into groups according to the `group_by` parameter. Z-Scores are computed for the data in each group with respect to the data only in that group.

Output Data Column (*numerical*): `_ZSCORE`

Terminal	Description
In	Input Transmission
Out	Transmission with the result placed in the output column

Parameter	Description
data_column	Input data column containing numerical arrays
group_by	Categorical data column to group by.
Apply	Process data through this node

19.4.9 LinRegress

Source

Basically uses `scipy.stats.linregress`

Performs Linear Regression on numerical arrays and returns slope, intercept, r-value, p-value and standard error

Terminal	Description
In	Input Transmission
Out	Transmission with the result placed in the output column

Parameter	Description
data_column	Data column containing 1D numerical arrays. The values are used as the y values and indices as the x values for the regression

Output Columnns: Single numbers, `_SLOPE`, `_INTERCEPT`, `_R-VALUE`, `_P-VALUE`, `_STDERR` as decribed in [scipy.stats.linregress](#)

19.5 Biology

Nodes for some biologically useful things which I couldn't categorize elsewhere

19.5.1 ExtractStim

Source

Extract the portions of a trace corresponding to stimuli that have been temporally mapped onto it. It outputs one row per stimulus period.

Note: Stimulus extraction is currently quite slow, will be optimized after some planned changes in the Transmission object.

Output Data Column	Description
ST_TYPE	Stimulus type, corresponds to your <i>Project Config</i>
ST_NAME	Name of the stimulus
_ST_CURVE	The extracted array based on the parameters
_ST_START_IX	Start index of the stimulus period in the parent curve
_ST_END_IX	End index of the stimulus period in the parent curve
ST_uuid	UUID assigned for the extracted stimulus period

Parameter	Description
data_column	Data column containing the signals to be extracted based on the stimulus maps
Stim_Type	Type of stimulus to extract
start_offset	Offset the start index of the stimulus mapping by a value (in frames)
end_offset	Offset the end index of the stimulus mapping by a value (in frames)
zero_pos	Zero index of the extracted signal <i>start_offset</i> : extraction begins at the <i>start_offset</i> value, stops at the <i>end_offset</i> <i>stim_end</i> : extraction begins at the end of the stimulus, stops at the <i>end_offset</i> . <i>stim_center</i> : extraction begins at the midpoint of the stimulus period plus the <i>start_offset</i> , stops at <i>end_offset</i>

19.5.2 DetrendDFoF

Source

Uses the `detrend_df_f` function from the `CaImAn` library. This node does not use any of the numerical data in a Transmission DataFrame to compute the detrended $\Delta F/F_0$. It directly uses the CNMF output data for the Samples that are present in the Transmission DataFrame.

Output Data Column (*numerical*): `_DETREND_DF_O_F`

19.5.3 StaticDFoFo

Source

Perform $\frac{F-F_0}{F_0}$ without a rolling window. F is an input array and F_0 is the minimum value of the input array.

Output Data Column (*numerical*): `_STATIC_DF_O_F`

Terminal	Description
In	Input Transmission
Out	Transmission with the result placed in the output column

Parameter	Description
<code>data_column</code>	Data column containing numerical arrays
Apply	Process data through this node

19.6 Clustering

19.6.1 KShape

Source

Perform KShape clustering. For more information see [KShape plot](#).

19.6.2 KMeans

Source

Basically `sklearn.cluster.KMeans`.

19.7 Hierarchical

These nodes allow you to perform Hierarchical Clustering using `scipy.cluster.hierarchy`.

If you are unfamiliar with Hierarchical Clustering I recommend going through this chapter from Michael Greenacre: <http://www.econ.upf.edu/~michael/stanford/maeb7.pdf>

Note: Some of these nodes do not use Transmission objects for some inputs/outputs.

19.7.1 Linkage

Source

Compute a linkage matrix which can be used to form flat clusters using the *FCluster* node.

Based on `scipy.cluster.hierarchy.linkage`

Terminal	Description
In	Input Transmission
Out	dict containing the Linkage matrix and parameters, not a Transmission object

Parameters	Description
data_column	Numerical data column used for computing linkage matrix
method	linkage method
metric	metric for computing distance matrix
optimal_order	minimize distance between successive leaves, more intuitive visualization Click here for more info
Apply	Process data through this node

19.7.2 FCluster

Source

“Form flat clusters from the hierarchical clustering defined by the given linkage matrix.”

Based on `scipy.cluster.hierarchy.fcluster`

Output Data Column (*categorical*): FCLUSTER_LABELS

Terminal	Description
Linkage	Linkage matrix, output from <i>Linkage</i> node.
Data	Input Transmission, usually the same input Transmission used for the <i>Linkage</i> node.
IncM (<i>optional</i>)	Inconsistency matrix, output from <i>Inconsistent</i>
Monocrit (<i>optional</i>)	Output from <i>MaxIncStat</i> or <i>MaxInconsistent</i>
Out	Transmission with clustering data that can be visualized using the <i>Heatmap</i>

Parameters: Exactly as described in `scipy.cluster.hierarchy.fcluster`

HistoryTrace output structure: Dict of all the parameters for this node, as well as the parameters used for creating the linkage matrix and the linkage matrix itself from the *Linkage* node.

19.7.3 Inconsistent

19.7.4 MaxIncStat

19.7.5 MaxInconsistent

19.8 Transform

Nodes for transforming data

19.8.1 LDA

Source

Perform Linear Discriminant Analysis. Uses `sklearn.discriminant_analysis.LinearDiscriminantAnalysis`

Terminal	Description
train_data	Input Transmission containing the training data
predict	Input Transmission containing data on which to predict
T	<p>Transmission with Transformed data and decision function. Output columns outlined below:</p> <p>_LDA_TRANSFORM: The transformed data, can be visualized with a <i>Scatter Plot</i> for instance</p> <p>_LDA_DFUNC: Decision function (confidence scores). Can be visualized with a <i>Heatmap</i></p>
coef	<p>Transmission with LDA Coefficients. Output columns outlined below:</p> <p>classes: The categorical labels that were trained against</p> <p>_COEF: LDA Coefficients (weight vectors) for the classes. Can be visualized with a <i>Heatmap</i></p>
means	<p>Transmission with LDA Means. Output columns outlined below:</p> <p>classes: The categorical labels that were trained against</p> <p>_MEANS: LDA means for the classes. Can be visualized with a <i>Heatmap</i></p>
predicted	<p>Transmission containing predicted class labels for the data.</p> <p>The class labels are placed in a column named LDA_PREDICTED_LABELS</p> <p>The names of the class labels correspond to the labels from the training labels</p> <p><i>optional</i></p>

Parameter	Description
train_data	Single or multiple data columns that contain the input features.
labels	Data column containing categorical labels to train to
solver	<i>svd</i> : Singular Value Decomposition <i>lsqr</i> : Least Squares solution <i>eigen</i> : Eigen decomposition
shrinkage	Can be used with <i>lsqr</i> or <i>eigen</i> solvers.
shrinkage_val	shrinkage value if <i>shrinkage</i> is set to “value”
n_components	Number of components to output
tol	Tolerance threshold exponent. The used value is $10^{<tol>}$
score	Displays mean score of the classification (read only)
predict_on	Single or multiple data columns that contain the data that are used for predicting on Usually the same name as the data column(s) used for the training data. <i>optional</i>

HistoryTrace output structure: Dict of all the parameters for this node

EXAMPLES

You can view examples of flowcharts in the demo dataset or one of the other datasets associated with the paper:

Demo dataset: <https://doi.org/10.6084/m9.figshare.11370183>

C. intestinalis dataset: <https://doi.org/10.6084/m9.figshare.10289162>

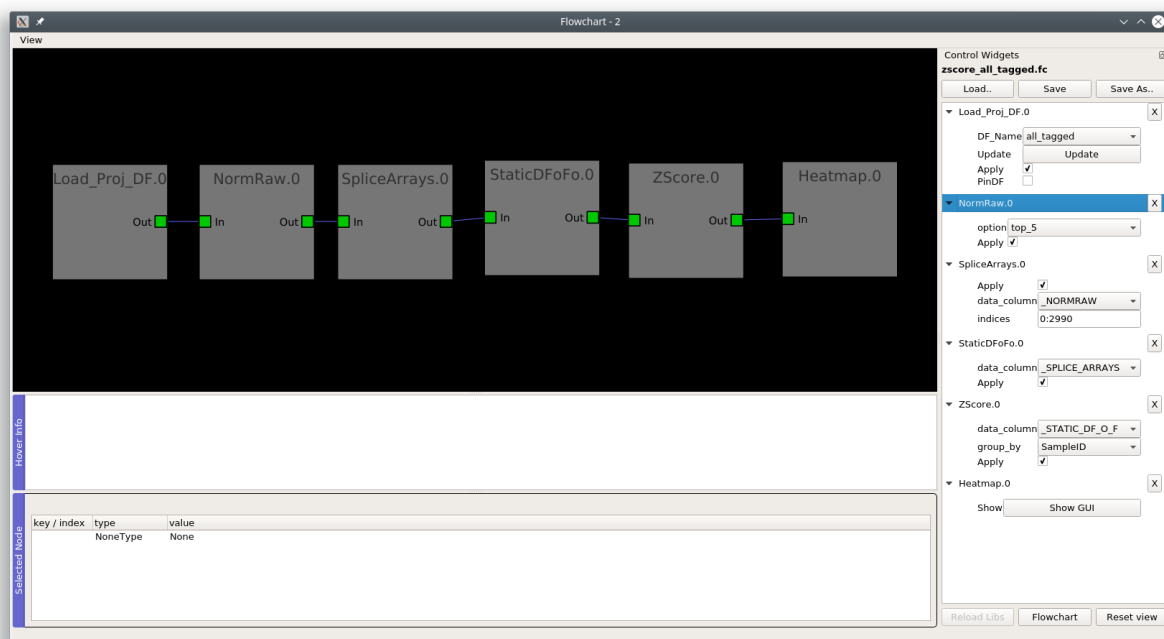
C. elegans dataset: <https://doi.org/10.6084/m9.figshare.10287113>

PVC-7 as a Mesmerize dataset: <https://doi.org/10.6084/m9.figshare.10293041>

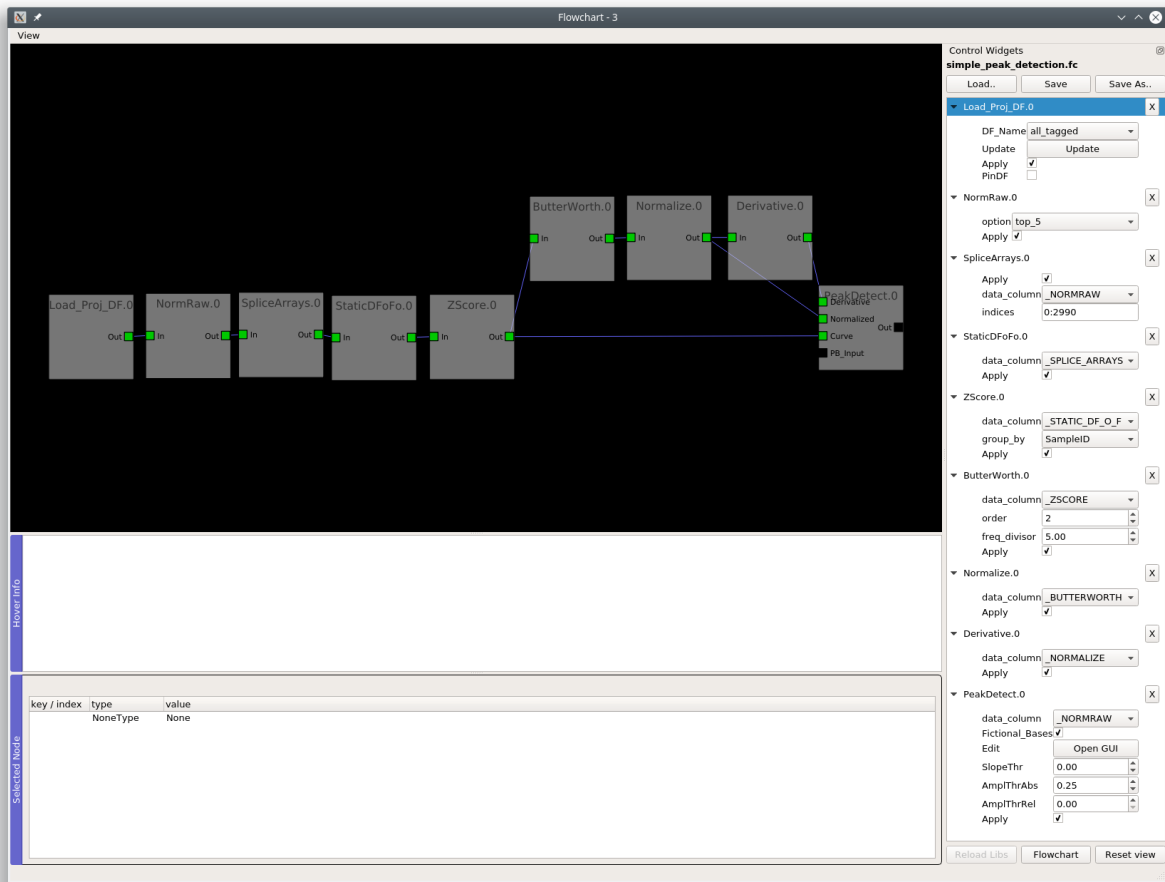
20.1 Screenshots

Flowchart screenshots from the C. intestinalis dataset.

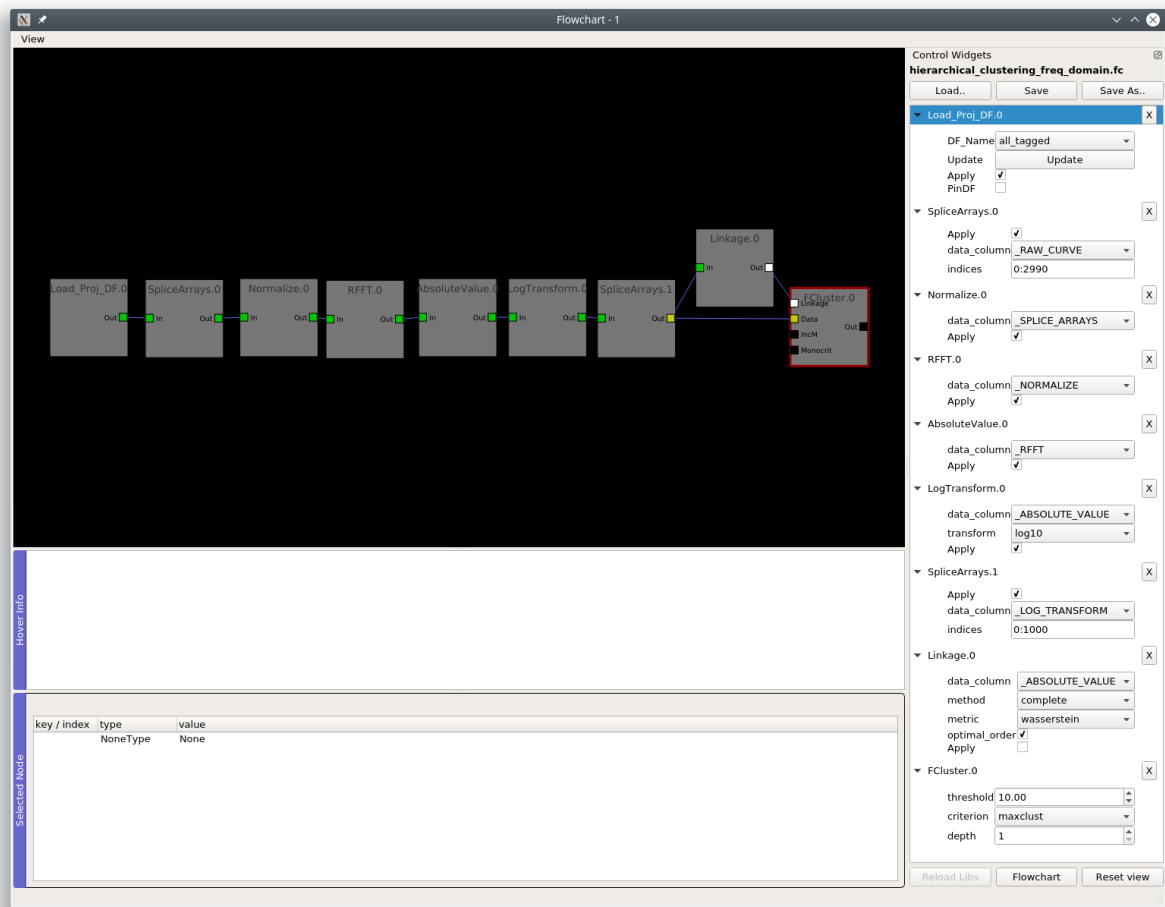
20.1.1 Z-score



20.1.2 Peak detection



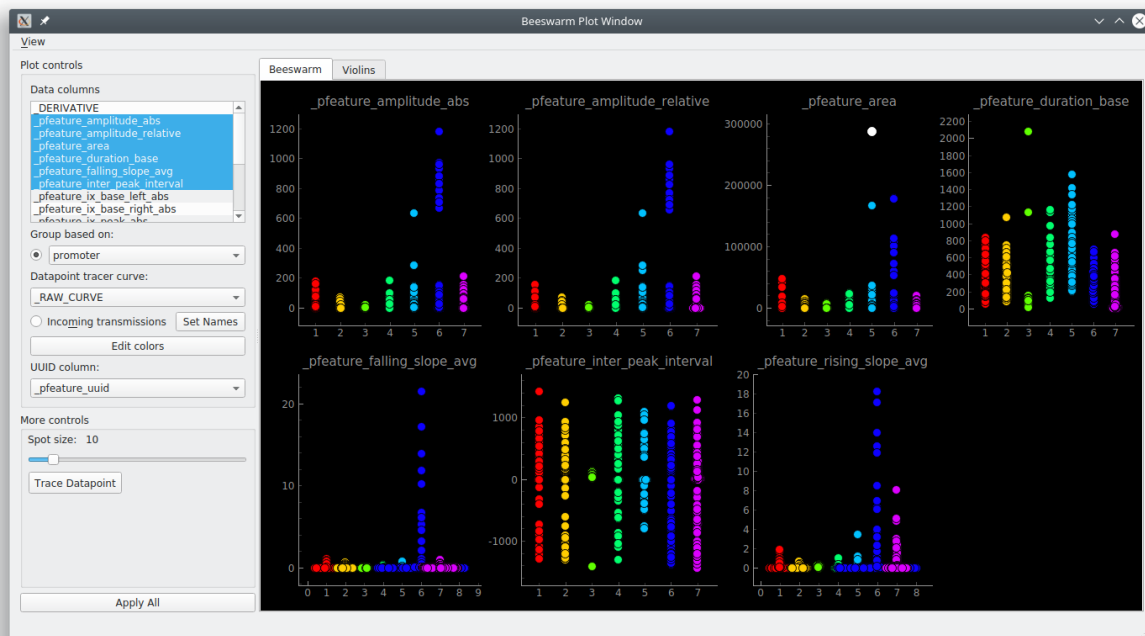
20.1.3 Hierarchical clustering



BEESWARM

Used for visualization of data points using a pseudo-scatter and violin plots.

Layout



You can click on individual datapoints and view the associated data using the *Datapoint Tracer*. To show the Datapoint Tracer, in the menubar go to View -> Live datapoint tracer

21.1 Parameters

Parameter	Description
Data columns	Multi-select data columns to plot They must have single numerical values, not arrays
Group based on	Categorical data column used for grouping the data
Datapoint tracer curve	Data column, containing numerical arrays, that is shown in the <i>Datapoint Tracer</i>
UUID column	Column containing the UUIDs that correspond to the data in the selected data column(s)
Apply all	Apply the plot parameters and draw the plot

CONSOLES

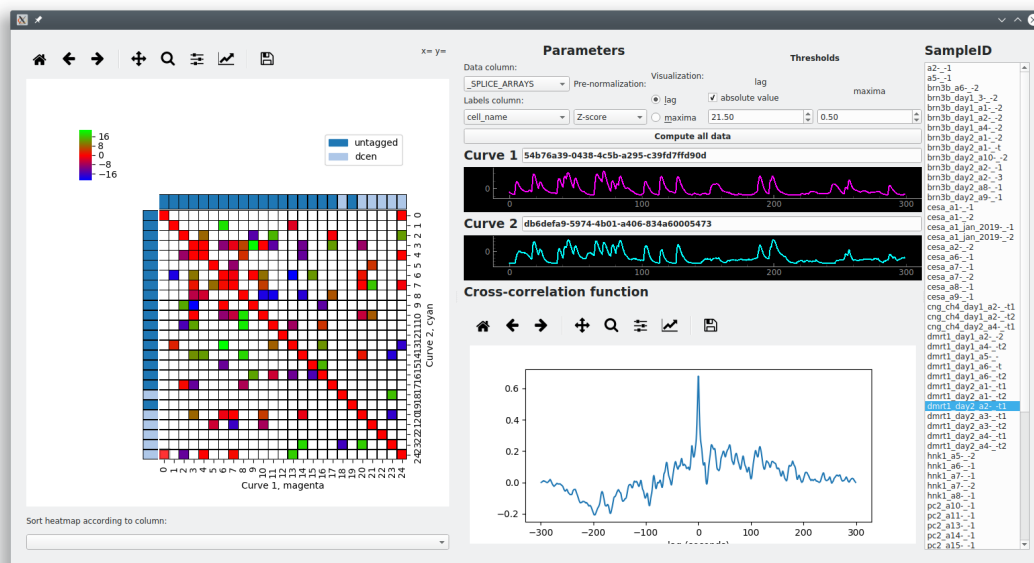
Currently the *Heatmap*, *Scatter*, *Proportions plot* and *KShape* follow a uniform structure allowing internal access to the data and plot axes. Refer to their Base API. For example, through their consoles you can access the *Transmission* containing data for the current plot, manually save the plot, etc.

CROSS CORRELATION

Explore [Cross-correlation functions](#) of all curves from a sample. Normalized cross-correlations are computed using `tslearn.cycc.normalized_cc`

This is an interactive widget. You can click on the individual cells in the heatmap to view the individual curves, the cross-correlation function of the two curves, and the spatial localization of the ROI that they originate from.

23.1 Layout



Left: Lag or Maxima Matrix (see below) with thresholds applied and visualized as a heatmap. When you click on the individual cells it will open/update the *Datapoint Tracer* according to the two curves the cell corresponds to.

Top Center: Parameters.

Center: When you click on a cell in the heatmap you will see Curve 1 (x-axis of heatmap), and Curve 2 (y-axis of heatmap) and their cross-correlation function. **The units are in seconds for all of these**

Right: List of Samples. Click on a Sample to select it as the current sample.

23.2 Lag Matrix

Computed as follows:

1. A 2D array is created where each element is a cross-correlation function (represented by a 1D numerical array).
2. The x-distance (time) between zero and the global maxima of the cross-correlation function (called *lag*) is computed for each of these elements.
3. The 2D array of cross-correlation functions is reduced to a 2D array of these *lag* values.

The result is a matrix where each element is the x-distance between zero and the global maxima of the cross-correlation of the two curves the element represents.

23.3 Maxima Matrix

Similar to computation of the Lag Matrix above, but instead of using the *lag* between zero and the global maxima it uses the y-value of the global maxima.

23.4 Parameters

Data column: The data column, containing *numerical* arrays, that are used as the input curves for computing cross-correlations.

Labels column: The labels column, containing *categorical* labels, that are used for the row and column labels in the heatmaps.

Pre-normalization: Option to perform 0 - 1 Normalization (Same method as the *Normalize*) or *Z-Score* of the input curves prior to computing their cross-correlation functions.

Compute all data: Apply the parameters and compute cross-correlation data for all Samples in the DataFrame of the input transmission.

23.4.1 Thresholds

Apply thresholds for *lag* and the maxima value. The colormap limits of the heatmap are set according to this threshold and all data under are set to white on the heatmap (you can still click and explore them).

Thresholds are applied live onto the heatmap.

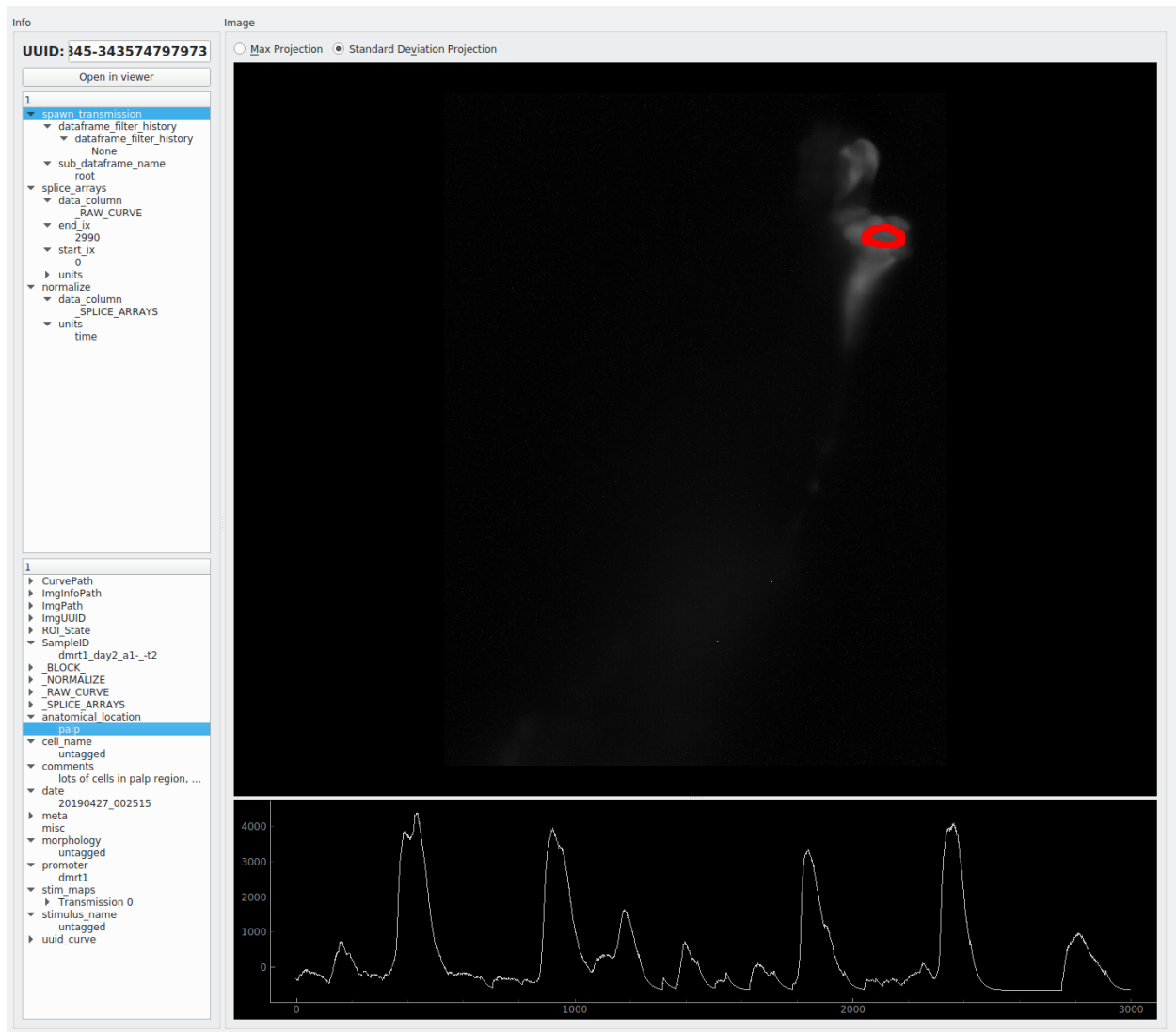
DATAPPOINT TRACER

API Reference

The Datapoint Tracer is attached to many plots, allowing you to interactively explore the data associated to the datapoints. You can explore the analysis history, the spatial localization of the ROI it originates from, associated numerical or categorical data, and view an additional numerical column (such as the raw trace).

The Datapoint Tracer is embedded in some plots, and in others you can open it by going to View -> Live Datapoint Tracer.

Datapoint Tracer Layout



Top right: Max Projection or Standard Deviation Project of the image sequence.

Bottom right: Numerical data, based on the “DPT Curve column” that the user has specified in the plot controls. If exploring peak feature based data the temporal span of the peak will be highlighted.

Top left: Analysis log, a ordered list of operations and their parameters.

Bottom left: All other data associated with this datapoint (the data present in the other columns of the row this datapoint is present in, see [Transmission](#))

Open in viewer button: Open the parent Sample of the current datapoint in the viewer.

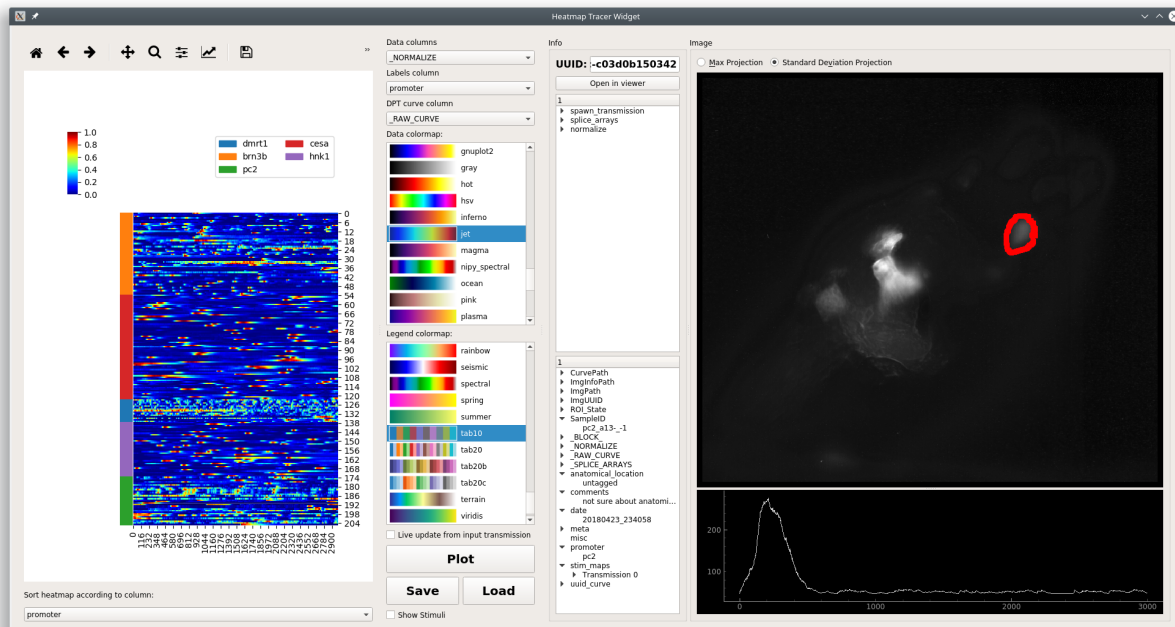
HEATMAP

API Reference

Note: This plot can be saved in an interactive form, see [Saving plots](#)

Visualize numerical arrays in the form of a heatmap. Also used for visualization of Hierarchical clustering dendrograms. *Datapoint Tracer* is embedded.

Layout



Left: The heatmap. Clicking the heatmap highlights the selected row and updates the *Datapoint Tracer*. Right click on the heatmap to clear the selection highlight on the heatmap. You can zoom and pan the heatmap using the tools above the plot area. You can zoom/pan in the legend and heatmap. The up and down keys on your keyboard can be used to move the current row selection.

Bottom left: Set the row order of the heatmap according to a categorical column.

Middle: Plot controls.

Very bottom: Status label - displays any issues that were raised while setting the plot data. Click on the status label to see more information.

25.1 Parameters

Data column: Data column, numerical arrays, that contain the data for the heatmap. Each row of this data column (a 1D array) is represented as a row on the heatmap.

Labels column: Column containing categorical labels that are used to create the row legend for the heatmap.

DPT curve column: Data column, containing numerical arrays, that is shown in the *Datapoint Tracer*.

Data colormap: Colormap used for representing the data in the heatmap. Default is ‘jet’.

Legend colormap: Colormap used for the row legend.

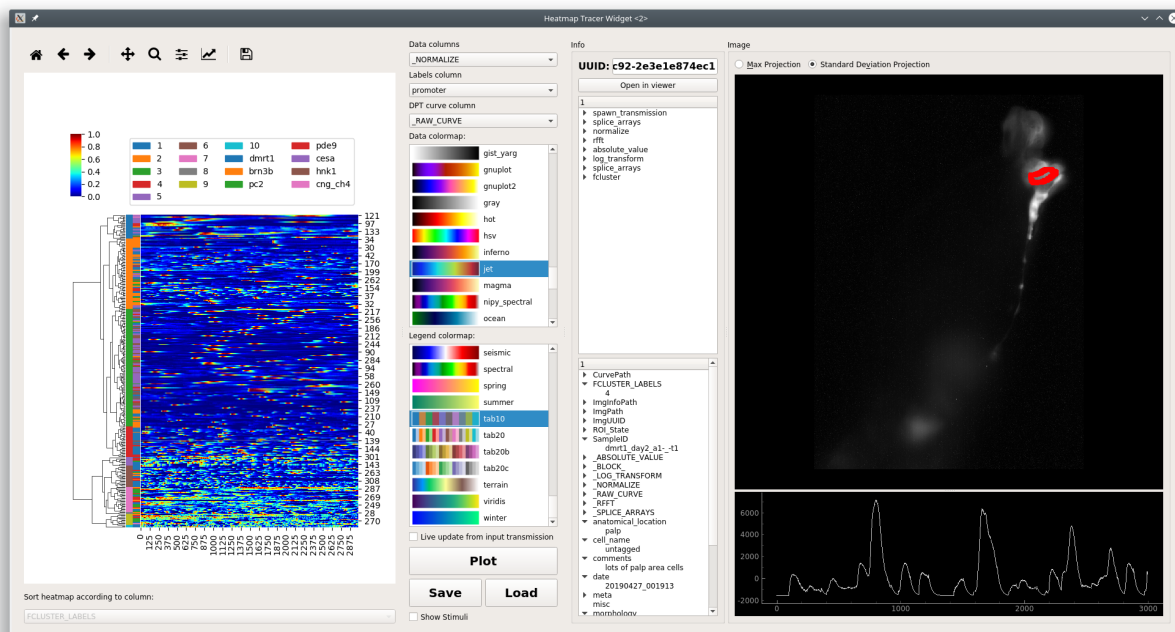
Live update from input transmission: If checked this plots receives live updates from the *flowchart*.

Plot: Updates data input from the flowchart.

Save: Save the plot data and state in an interactive form <save_ptrn>

Load: Load a plot that has been saved as a “.ptrn” file.

Layout to visualize Hierarchical Clustering



This plot widget can also be used to visualize a dendrogram on top of a heatmap of data.

The differences are:

1. There are two legend bars
 - Left: Cluster label
 - Right: Corresponds to Labels column parameter.
2. You can also zoom/pan the dendrogram in addition to the legends and heatmap.
3. Sorting the heatmap rows is disabled because this wouldn't make sense

25.2 Console

You can directly access the heatmap widget through the console. This is useful for plot customization and exporting with specific parameters.

Toggle the console's visibility by clicking on the “Show/Hide Console” button at the bottom of the controls.

See also:

API Reference

25.2.1 Namespace

reference	Description
this	The higher-level HeatmapTracerWidget instance, i.e. the entire widget
this.transmission	Current input <i>Transmission</i>
get_plot_area()	Returns the lower-level Heatmap variant instance, basically the actual plot area
get_plot_area().plot	Returns the seaborn ClusterGrid instance containing the axes
get_plot_area().fig	Returns the matplotlib <i>Figure</i> instance

Attributes of `get_plot_area().plot`

For example, the heatmap axes object can be retrieved through `get_plot_area().plot.ax_heatmap`. See the usage examples.

<code>ax_heatmap</code>	Heatmap axes
<code>ax_row_dendrogram</code>	Row dendrogram axes
<code>ax_col_dendrogram</code>	Used for the legend
<code>cax</code>	Colorbar axes

25.2.2 Examples

Export

See also:

matplotlib API for: `Figure.savefig`, `Figure.set_size_inches`, `Figure.get_size_inches`

```

1  # Desired size (width, height)
2  size = (2.0, 2.5)
3
4  # Get the figure
5  fig = get_plot_area().fig
6
7  # original size to reset the figure after we save it
8  orig_size = fig.get_size_inches()
9
10 #Set the desired size
11 fig.set_size_inches(size)
12
13 # Save the figure as a png file with 1200 dpi
14 fig.savefig('/share/data/temp/kushal/amazing_heatmap.png', dpi=1200, bbox_inches=
    ↳ 'tight', pad_inches=0)

```

(continues on next page)

(continued from previous page)

```
15
16 # Reset the figure size and draw()
17 fig.set_size_inches(orig_size)
18 get_plot_area().draw()
```

Note: The entire plot area might go gray after the figure is reset to the original size. I think this is a Qt-matplotlib issue. Just resize the window a bit and the plot will be visible again!

Warning: From my experience I have not been able to open clustermap SVG files saved with very high DPI (600+). Even with 32 cores & 128GB of RAM both inkscape and illustrator just hang _()_/ . Try png or other formats.

x tick labels

See also:

`matplotlib.axes.Axes.set_xticklabels`, `matplotlib.axes.Axes.set_xticks`.

If the data are in the time domain:

```
1 from mesmerize.analysis import get_sampling_rate
2 import numpy as np
3
4 # Get the sampling rate of the data
5 sampling_rate = get_sampling_rate(this.transmission)
6
7 # Number of frames currently displayed in the heatmap
8 num_frames = get_plot_area().data.shape[1]
9
10 # Set an appropriate interval
11 interval = 30 # This is in seconds, not frames
12
13 # Get the recording time in seconds
14 recording_time = int(num_frames / sampling_rate)
15
16 # Set the new ticks
17 get_plot_area().plot.ax_heatmap.set_xticks(np.arange(0, num_frames, interval *
18     ↳sampling_rate))
19
20 # Set the tick labels
21 # You can change the fontsize here
22 get_plot_area().plot.ax_heatmap.set_xticklabels(np.arange(0, recording_time,
23     ↳interval), fontdict={'fontsize': 4})
24
25 # Set a title for the x axis. You can change the fontsize here
26 get_plot_area().plot.ax_heatmap.set_xlabel('Time (seconds)', fontdict={'fontsize': 6})
27
28 # Draw the plot with these changes
29 get_plot_area().draw()
```

Note: You may need to resize the dock widget that the plot is present in to display the newly drawn plot, this is a

Qt-matplotlib issue.

If the data are in the frequency domain:

```

1  from mesmerize.analysis import get_frequency_linspace
2  import numpy as np
3
4  # Get frequency linspace and Nyquist frequency
5  freqs, nf = get_frequency_linspace(this.transmission)
6
7  # Get the number of frequencies currently shown in the heatmap
8  num_freqs = get_plot_area().data.shape[1]
9
10 # The max frequency currently display in the heatmap
11 max_freq = freqs[num_freqs - 1]
12
13 # Set an appropriate interval
14 interval = 0.25 # This is in Hertz
15
16 # Set the tick labels
17 # Set the new ticks
18 get_plot_area().plot.ax_heatmap.set_xticks(np.arange(0, num_freqs, (num_freqs *
19   ↪ interval) / max_freq))
20
21 # You can change the fontsize here
22 get_plot_area().plot.ax_heatmap.set_xticklabels(np.arange(0, max_freq, interval),
23   ↪ fontdict={'fontsize': 4})
24
25 # Set a title for the x axis. You can change the fontsize here
26 get_plot_area().plot.ax_heatmap.set_xlabel('Frequency (Hertz)', fontdict={'fontsize':
27   ↪ 6})
28
29 # Draw the plot with these changes
30 get_plot_area().draw()

```

Note: You may need to resize the dock widget that the plot is present in to display the newly drawn plot, this is a Qt-matplotlib issue.

Colorbar label

```

get_plot_area().plot.cax.set_title('norm. z-score', x=-0.25, y=0.65, fontdict={
  ↪ 'fontsize': 6}, rotation=90)
get_plot_area().draw()

```

Axes visibility

Hide/show legend

```
get_plot_area().plot.ax_col_dendrogram.set_visible(False)
get_plot_area().draw()
```

Hide/show y axis (similar for x axis)

```
get_plot_area().plot.ax_heatmap.get_yaxis().set_visible(False)
get_plot_area().draw()
```

Hide/show colorbar

```
get_plot_area().plot.cax.set_visible(False)
get_plot_area().draw()
```

Perform KShape clustering.

I recommend reading the paper on it: Paparrizos, John, and Luis Gravano. “k-Shape: Efficient and Accurate Clustering of Time Series.” In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1855-1870. ACM, 2015.

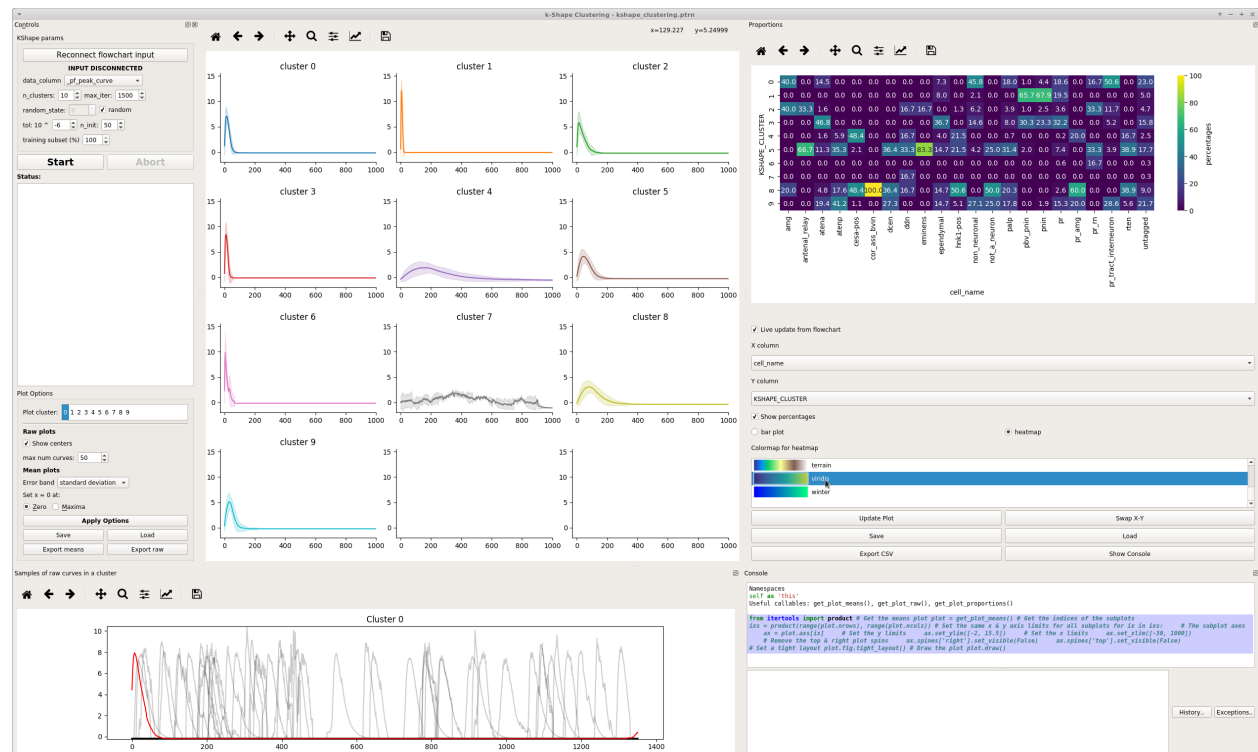
This GUI uses the `tslearn.clustering.KShape` implementation.

See also:

API reference

Note: This plot can be saved in an interactive form, see [Saving plots](#)

Layout



Left: KShape parameters and Plot parameters

Bottom left: Plot of a random sample of input data from a cluster.

Center: Plot of cluster mean and either confidence interval, standard deviation, or neither. Uses `seaborn.lineplot`

Right: Proportions plot. Exactly the same as *Proportions*.

Bottom Right: Console

26.1 KShape Parameters

The parameters and input data are simply fed to `tslearn.clustering.KShape`

Parameters outlined here are simply as they appear in the tslearn docs.

data_column: Input data for clustering.

n_clusters: Number of clusters to form.

max_iter: Maximum number of iterations of the k-Shape algorithm.

tol: Inertia variation threshold. If at some point, inertia varies less than this threshold between two consecutive iterations, the model is considered to have converged and the algorithm stops.

n_init: Number of times the k-Shape algorithm will be run with different centroid seeds. The final results will be the best output of `n_init` consecutive runs in terms of inertia.

random_state: Generator used to initialize the centers. If an integer is given, it fixes the seed. Defaults to the global numpy random number generator.

training_subset: The subset of the input data that are used for used for training. After training, the predictions are fit on all the input data.

26.2 Plot Options

Plot cluster: The cluster from which to plot random samples of input data in the bottom left plot

Show centers: Show the centroids returned by the KShape model

Warning: There's currently an issue where cluster centroids don't appear to be index correctly. See <https://github.com/rtavenar/tslearn/issues/114>

max num curves: Maximum number of input data samples to plot

Error band: The type of data to show for the the error band in the means plots.

set x = 0 at: The zero position of a means plots with respect to the cluster members in the plot.

Save: *Save the plot data and state in an interactive form*

26.3 Console

The console can be useful for formatting plots, inspecting the underlying data etc.

See also:

API reference

26.3.1 Namespace

reference	Description
this	The higher-level KShape widget instance, i.e. the entire widget
this.transmission	Current input <i>Transmission</i>
get_plot_means()	Returns the means plot
get_plot_raw()	Returns the raw plot
get_plot_proportions()	Returns the proportions plot, which is an instance of Proportions Widget

26.3.2 Examples

See also:

matplotlib Axes

Set axis ranges

Set equal x & y axis ranges for the means plots. Also removes the top & right spines.

```

1  from itertools import product
2
3  # Get the means plot
4  plot = get_plot_means()
5
6  # Get the indices of the subplots
7  ix = product(range(plot.nrows), range(plot.ncols))
8
9  # Set the same x & y axis limits for all subplots
10 for ix in ix:
11
12     # The subplot axes
13     ax = plot.axes[ix]
14
15     # Set the y limits
16     ax.set_ylim([-2, 15.5])
17
18     # Set the x limits
19     ax.set_xlim([-30, 1000])
20
21     # Remove the top & right plot spins
22     ax.spines['right'].set_visible(False)
23     ax.spines['top'].set_visible(False)
24
25 # Set a tight layout
26 plot.fig.tight_layout()
```

(continues on next page)

(continued from previous page)

```

27
28 # Draw the plot
29 plot.draw()

```

Note: You may need to resize the dock widget that the plot is present in to display the newly drawn plot, this is a Qt-matplotlib issue.

x tick labels

Set the x tick labels in time units instead of frames

See also:

matplotlib.axes.Axes.set_xticklabels | matplotlib.axes.Axes.set_xticks.

```

1 import numpy as np
2 from itertools import product
3 from mesmerize.analysis import get_sampling_rate
4
5 # Get the sampling rate of the data
6 sampling_rate = get_sampling_rate(this.transmission)
7
8 # Get the padded number of frames that are shown in the plots
9 num_frames = this.cluster_centers.shape[1]
10
11 # Set an appropriate interval
12 interval = 5 # This is in seconds, not frames
13
14 # Convert the padded frame number to time units
15 total_time = int(num_frames / sampling_rate)
16
17 ix = product(range(4), range(3))
18
19 # Set these time units for all the means plots
20 # For the raw plots just remove the loop
21 for ix in ix:
22     # Get the axes
23     ax = get_plot_means().axes[ix]
24
25     # Set the new ticks
26     ax.set_xticks(np.arange(0, num_frames, interval * sampling_rate))
27
28     # Set the tick labels
29     # You can change the fontsize here
30     ax.set_xticklabels(np.arange(0, total_time, interval), fontdict={'fontsize': 4},
31 ↪ rotation=90)
32
33     # Set a title for the x axis. You can change the fontsize here
34     ax.set_xlabel('Time (seconds)', fontdict={'fontsize': 6})
35
36     # Set ylabel as well
37     ax.set_ylabel('z-score', fontdict={'fontsize': 6})
38
39 # Set a tight layout

```

(continues on next page)

(continued from previous page)

```

39 get_plot_means().fig.tight_layout()
40
41 # Draw the plot with these changes
42 get_plot_means().draw()

```

Note: You may need to resize the dock widget that the plot is present in to display the newly drawn plot, this is a Qt-matplotlib issue.

Hide legend

Hide/show legend in the proportions plot

```

get_plot_proportions().ax.legend().set_visible(True)
get_plot_proportions().draw()

```

Export

You can export any of the plots with a specific size & DPI.

Replace the `get_<plot>().fig` on *line 5* with the desired plot.

See also:

matplotlib API for: [Figure.savefig](#), [Figure.set_size_inches](#), [Figure.get_size_inches](#)

```

1  # Desired size (width, height)
2  size = (7.0, 10.0)
3
4  # Get the figure
5  fig = get_<plot>().fig
6
7  # original size to reset the figure after we save it
8  orig_size = fig.get_size_inches()
9
10 #Set the desired size
11 fig.set_size_inches(size)
12
13 # Save the figure as an png file with 600 dpi
14 fig.savefig('/share/data/temp/kushal/amazing_shapes.png', dpi=600, bbox_inches='tight
    ↳', pad_inches=0)
15
16 # Reset the figure size and draw
17 fig.set_size_inches(orig_size)
18 get_<plot>().draw()

```

Note: The entire plot area might go gray after the figure is reset to the original size. I think this is a Qt-matplotlib issue. Just resize the window a bit and the plot will be visible again!

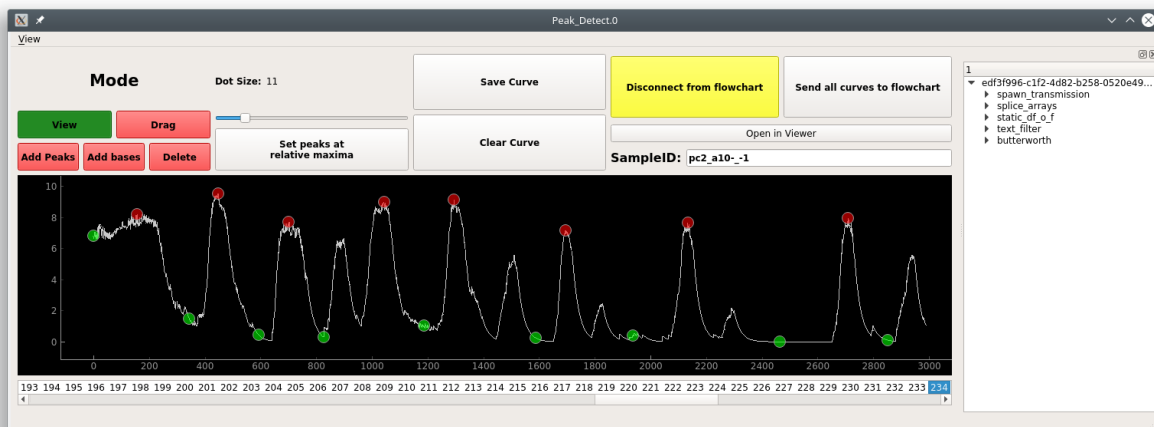
PEAK EDITOR

Visualize and edit detected peaks & bases. This GUI is accessible through the *PeakDetect* node.

27.1 Usage

- Optimize your peaks/bases detection through the datastreams that feed into the *Derivative* and *Normalize* terminals of the parent *PeakDetect* node. For example, play with filtering parameters for the *ButterWorth* node or *SavitzkyGolay* node.
- Optimize amplitude thresholds of the parent *PeakDetect* node.
- Disconnect from the flowchart (see below).
- Edit your peaks/bases
- Click “Send all curves to flowchart” (see below) to set the edited data as the output of the parent *PeakDetect* node.

27.1.1 Layout



Bottom

List of curves from the Transmission inputted to the *Curve* or *PB_Input* terminal. See *PeakDetect node*

Top

Mode buttons: Set the current interactive mode for mouse events.

View: Just view, pan, and zoom the plot.

Drag: Click and drag peaks/bases along the curve.

Add Peak/Base: Click to add a peak/base onto the curve.

Delete: Delete a peak or base.

Dot Size: Move the slider to change the size of the dots representing peaks/bases.

Set Peaks at relative maxima: Not implemented yet.

Save Curve: Save the current curve. A curve auto-saved when you switch to another one.

Clear Curve: Not implemented.

Disconnect from flowchart: Disconnect the GUI from changes in the flowchart. Edits to the peaks/bases will be lost if this GUI is not disconnected while changes occur in the flowchart.

Send all curves to flowchart: Set the edited data as the output of the parent *PeakDetect node*

Open in viewer: Open the parent Sample of this curve in a *Viewer*.

Right

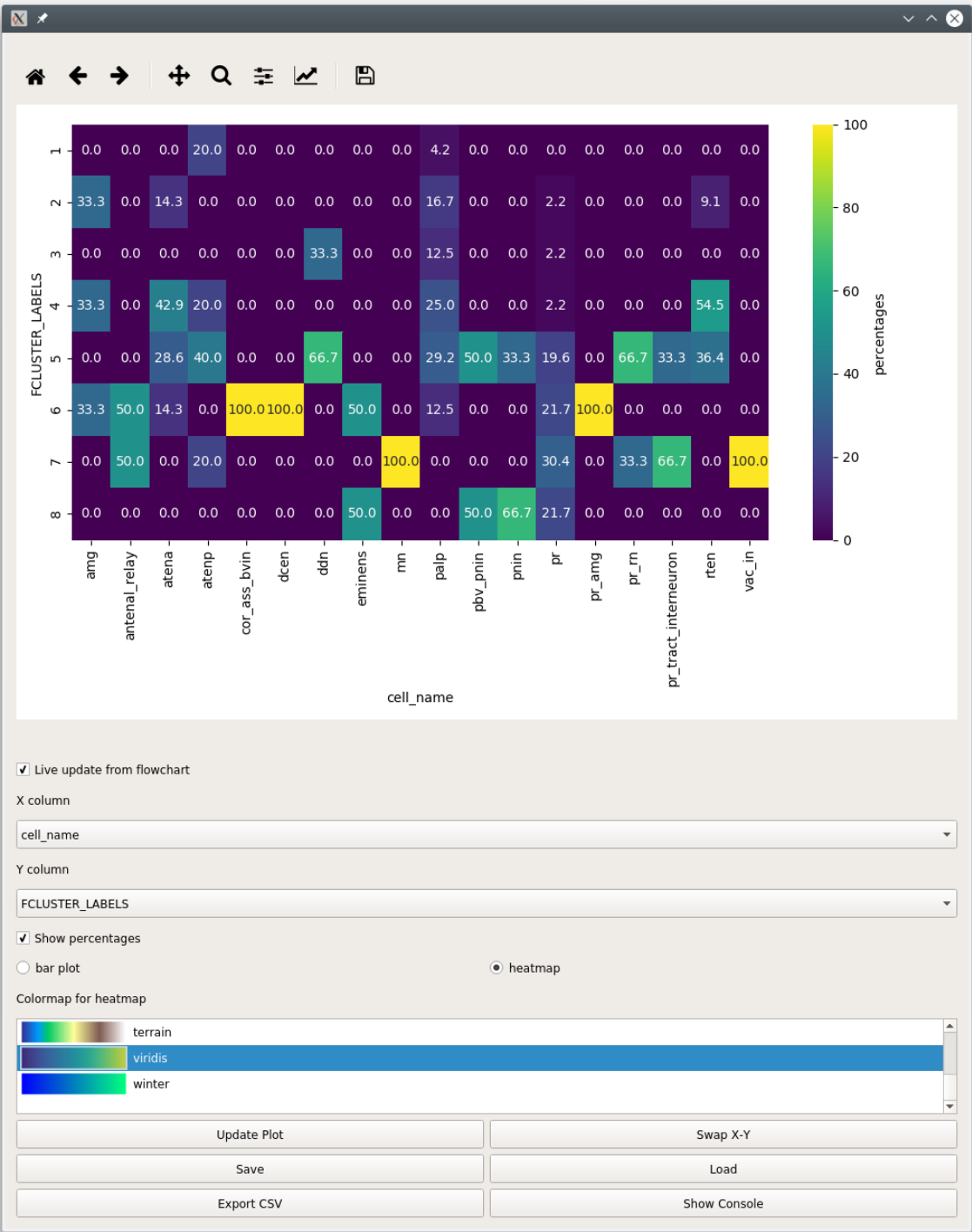
History Tree Widget

PROPORTIONS

API Reference

Note: *This plot can be saved in an interactive form*

Compare proportions of categorical variables between different groups using bar charts.



Parameter	Description
X column	DataFrame column containing the categorical labels used for grouping the data Data in each X column sums to 100% if <i>Show percentages</i> is checked
Y column	DataFrame column containing the categorical labels that are counted for each group
Show percentages	When unchecked shows raw counts
bar plot	Visualize as bar plots
heatmap	Visualize as a heatmap
Update Plot	Update plot
Swap X-Y	Swap X & Y columns
Save	<i>Save this plot as a ptrn file</i>
Load	<i>Load from a ptrn file</i>
Export CSV	Export the data for the current plot as to a csv file.
Show Console	Show/hide the console

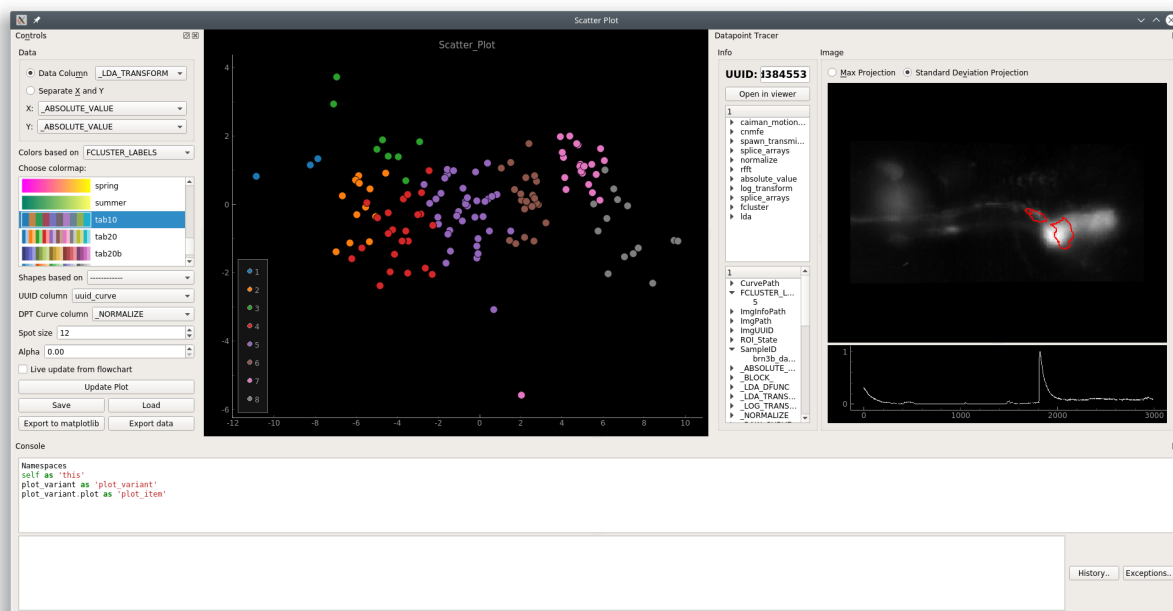
SCATTER

API Reference

Interactive scatter plot

Note: *This plot can be saved in an interactive form*

29.1 Layout



Left: Controls

Control	
Data Column	Data column containing numerical arrays of size 2, X & Y values [x, y]
X	Data column containing only X values
Y	Data column containing only Y values
log x	Use \log_{10} of the X data
log y	Use \log_{10} of the Y data
Colors based on	Set spot colors based on categorical labels in this column
Choose colormap	Colormap for the the spot colors
Shapes based on	Set spot shapes based on categorical labels in this column
UUID Column	Column containing UUIDs that correspond to the plot data
DPT Curve column	Data column containing numerical arrays to show in the <i>Datapoint Tracer</i>
Spot size	Size of the spots
Alpha	Not implemented yet
Live update...	Update the plot with live inputs from the flowchart
Update Plot	Update the plot according to the input data from the flowchart and the parameters
Save	<i>Save the plot as a ptrn file</i>
Load	<i>Load a saved ptrn file</i>
Export to ma...	Not implemented yet
Export data	Not implemented yet

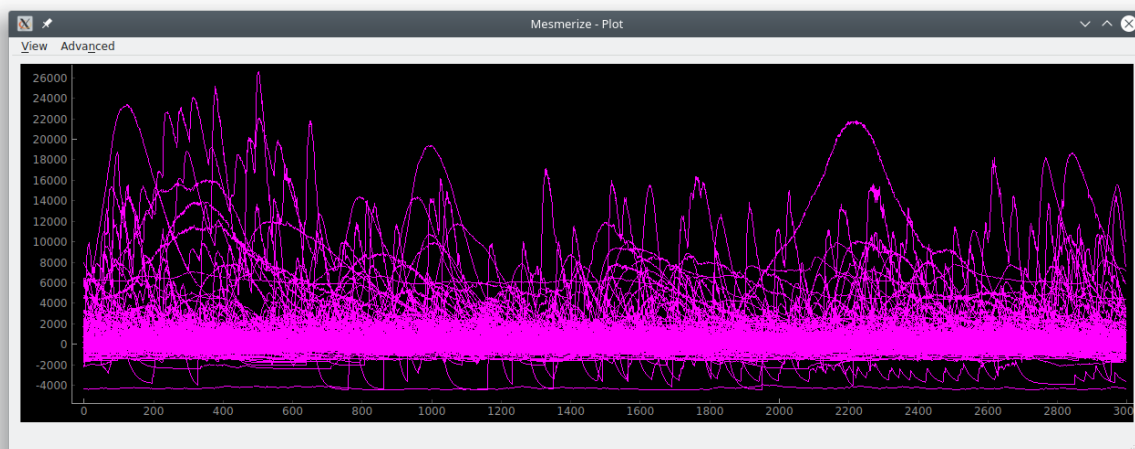
Below the plot: Status label that displays plotting issues. Click the label to see more information.

Right: *Datapoint Tracer*. Click datapoints in the plot to set the Datapoint Tracer.

Bottom: *Console*

SIMPLE PLOT

Just a very basic time-series plot. It will plot all the data in the selected data column.

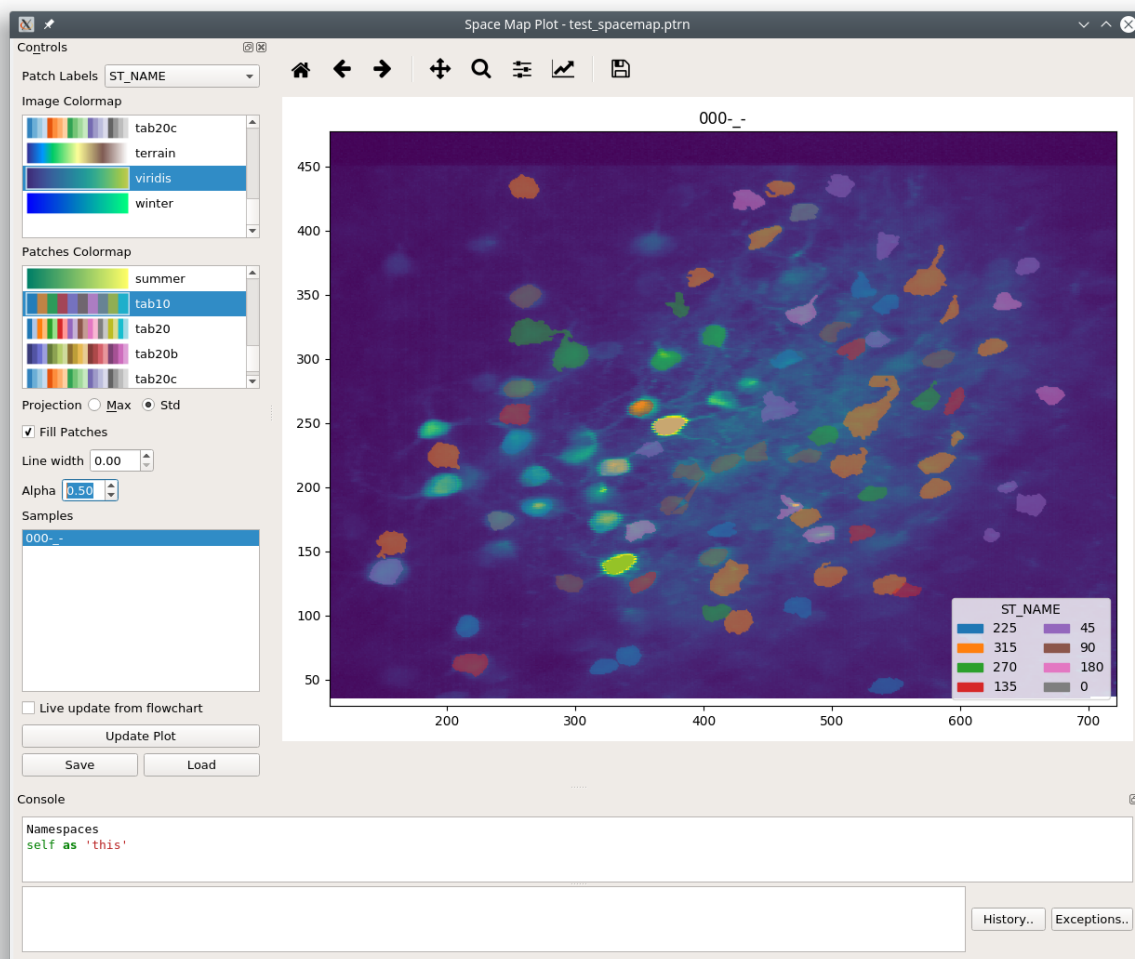


SPACEMAP

API Reference

Note: This plot can be saved in an interactive form, see [Saving plots](#)

Spatially map a categorical variable onto a projection of a Sample's image sequence



Note: Image produced from the following dataset: Garner, Aleena (2014): In vivo calcium imaging of layer 4 cells in the mouse using sinusoidal grating stimuli. CRCNS.org. <http://dx.doi.org/10.6080/K0C8276G>

31.1 Controls

Parameter	Description
Patch labels	Categorical column to use for the patch labels
Image Colormap	Colormap for the image
Patches Colormap	Colormap for the patches
Projection	Show the image as a “Max” or “Standard Deviation” projection
Fill Patches	Fill the patches
Line width	Line width of the patches
Alpha	Alpha level of the patches
Samples	Click on the sample to plot
Save	<i>Save the plot data and state in an interactive form</i>
Load	Load a plot that has been saved as a “.ptrn” file.

31.2 Console

See also:

API Reference

31.2.1 Namespace

reference	Description
this	The SpaceMapWidget instance, i.e. the entire widget
this.transmission	Current input <i>Transmission</i>
get_plot()	Returns the plot area
get_plot().fig	Returns the matplotlib <i>Figure</i> instance
get_plot().ax	Returns the Axes for the current plot <i>matplotlib Axes</i>

31.2.2 Examples

Export

See also:

matplotlib API for: *Figure.savefig*, *Figure.set_size_inches*, *Figure.get_size_inches*

```
1 # Desired size (width, height)
2 size = (6,5)
3
4 # Get the figure
5 fig = get_plot().fig
```

(continues on next page)

(continued from previous page)

```
6
7  # original size to reset the figure after we save it
8  orig_size = fig.get_size_inches()
9
10 #Set the desired size
11 fig.set_size_inches(size)
12
13 # Save the figure as a png file with 600 dpi
14 fig.savefig('/share/data/temp/kushal/spacemap.png', dpi=600, bbox_inches='tight', pad_
    ↪ inches=0)
15
16 # Reset to original size and draw
17 fig.set_size_inches(orig_size)
18 get_plot().draw()
```

Note: The entire plot area might go gray after the figure is reset to the original size. I think this is a Qt-matplotlib issue. Just resize the window a bit and the plot will be visible again!

Legend Title

See also:

matplotlib API for `matplotlib.axes.Axes.get_legend`

```
get_plot().ax.get_legend().set_title('New Title')
get_plot().draw()
```

Hide Axis Borders

See also:

matplotlib API for `matplotlib.axes.Axes.axis`

```
get_plot().ax.axis('off')
get_plot().draw()
```

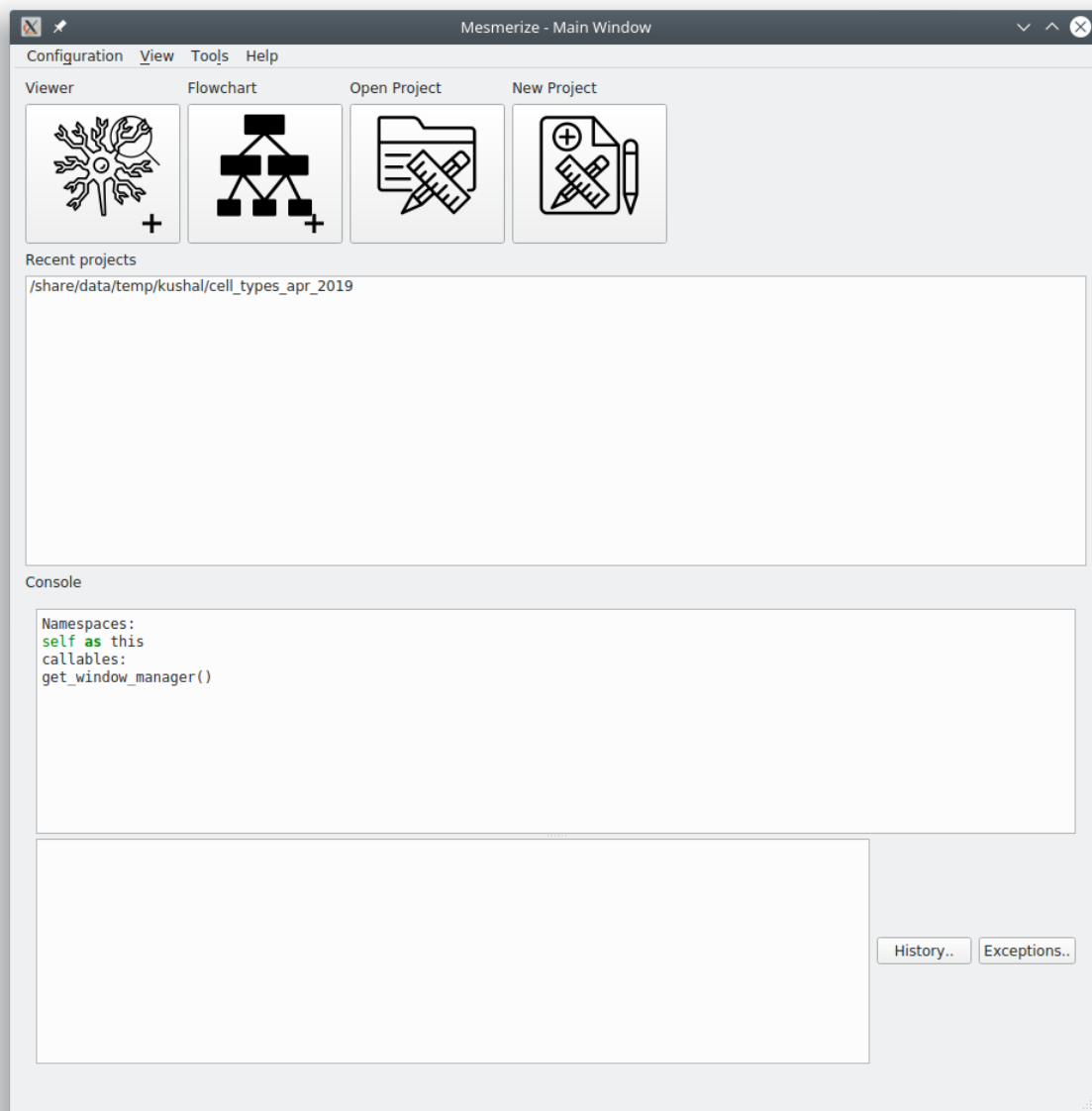

WELCOME WINDOW

The Welcome Window is the first window that you are presented with when you launch Mesmerize.

- Use the large buttons for opening new *Viewer* or *Flowchart* windows.
- Open a project using the button, or double-click a recent project from the list.
- Create a new project using the button.
- You basically have access to all objects in the Mesmerize instance through this console.

See also:

User guide on creating new projects and *Consoles*



PROJECT STRUCTURE

A Mesmerize project is encapsulated within a single directory. It contains the following:

- config file - contains configuration data, such as roi type columns, stimulus type columns, and custom columns with their datatypes.

Warning: Do not manually modify the config file

Directories

Dir	Purpose
dataframe	Contains an file storing the project dataframe, root.dfr, and backups. A new backup is created every time a <i>Sample</i> is added to the project. Restore a backup by renaming it to “root.dfr”.
im-ages	Contains the image sequences and work environment data for all samples in the project
curves	Contains the curves for every sample in the project
batches	Used for storing batches used by the Batch Manager if you wish.
flowcharts	Used for storing .fc flowchart files that save the layout of nodes in a flowchart.
plots	Used for storing .ptm interactive plot files.

See also:

Flowchart Overview and *Saving Plots*

Warning: Do not manually modify the data under the **images** or **curves** directories

PROJECT SAMPLE

CONSOLES

A Python console is embedded in many parts of Mesmerize. You can use it to perform very specific operations, further automate tasks, save an analysis object, format plots, etc.

The console is accessible in many windows through View -> Console. Within the console namespace `this` refers to the window. For example `this` refers to the *Project Browser* Window instance in the Project Browser's console. A list of useful object references and helper functions are listed when you open most consoles.

You can run entire scripts within the console. You can also use import statements to import libraries that you have in your Python environment.

Keyboard controls:

Execute: Shift + Enter

New line: Enter

Scroll up through history: Page Up

Scroll down through history: Page Down

The history is stored in `~/mesmerize`

SAVING PLOTS

Some plots allow you to save them in an interactive form, along with the plot data and the plot state as a “.ptrn” file. If you save the file in the “plots” directory of your project it will be listed in the *Welcome Window* when you open your project.

This is currently possible with the following plots: *Heatmap*, *KShape*, *Proportions*, *Scatter*, and *SpaceMap*

PLOT NAVBAR

Many plots have a navigation toolbar which you can use to zoom, pan, configure plots, and export plots as images.

Official matplotlib docs about the navigation toolbar: https://matplotlib.org/2.1.2/users/navigation_toolbar.html

Home: Reset the plot (not applicable for all plots)

Pan: Pan the plot

Zoom: Zoom in/out a selection using the left/right mouse button respectively

Subplot-configuration: Options to adjust spacing, borders, set tight layout.

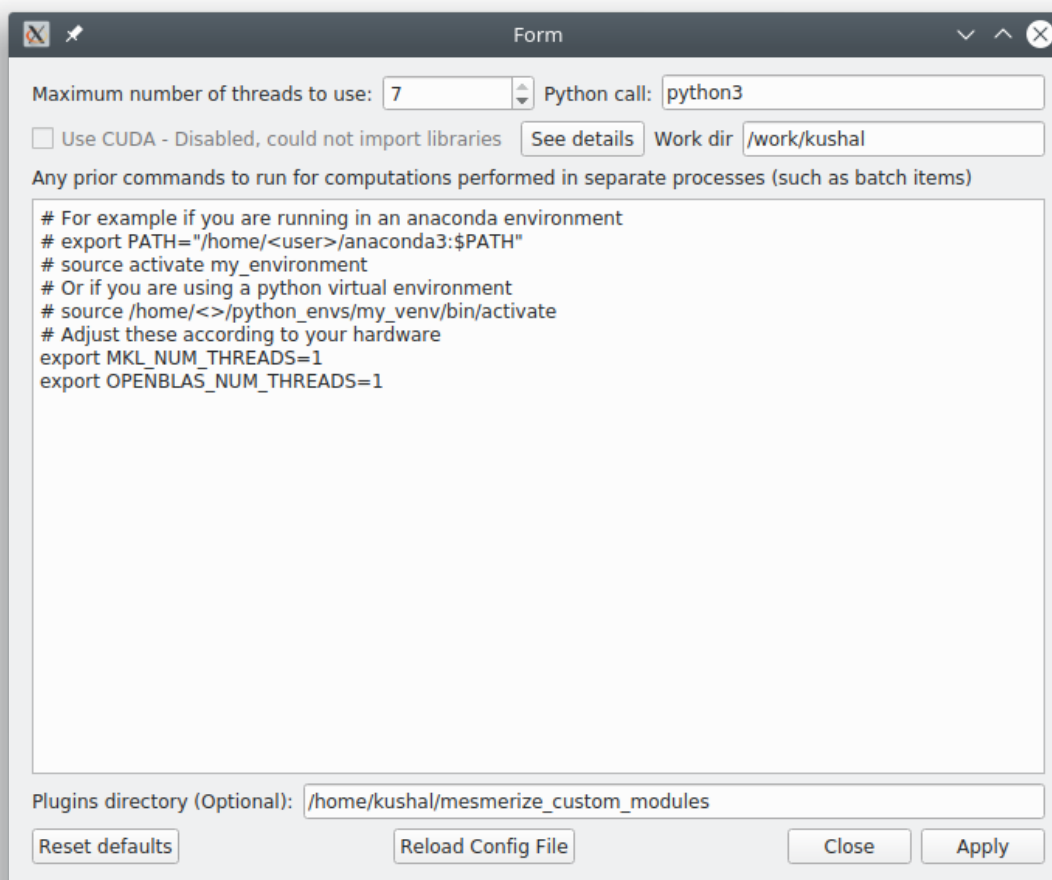
Edit axis, curve...: For some plots. Options for formatting x & y axis limits, labels, select line style, color, etc.

Save: Export the figure as an image. **This is not the same as saving an interactive plot, see “Saving Plots” above.**

SYSTEM CONFIGURATION

Set system configuration options

This window is accessible through the *Welcome Window* menubar at Configuration -> System Configuration.



The screenshot shows a window titled "Form" with the following fields and controls:

- Maximum number of threads to use:** A numeric input field with the value "7" and a small up/down arrow.
- Python call:** A text input field with the value "python3".
- Use CUDA:** A checkbox labeled "Use CUDA - Disabled, could not import libraries".
- See details:** A button.
- Work dir:** A text input field with the value "/work/kushal".
- Any prior commands to run for computations performed in separate processes (such as batch items):** A large text area containing the following text:

```
# For example if you are running in an anaconda environment
# export PATH="/home/<user>/anaconda3:$PATH"
# source activate my_environment
# Or if you are using a python virtual environment
# source /home/<user>/python_envs/my_venv/bin/activate
# Adjust these according to your hardware
export MKL_NUM_THREADS=1
export OPENBLAS_NUM_THREADS=1
```
- Plugins directory (Optional):** A text input field with the value "/home/kushal/mesmerize_custom_modules".
- Buttons:** "Reset defaults", "Reload Config File", "Close", and "Apply".

Maximum number of threads to use: The maximum number of threads that Mesmerize is allowed to use, this includes proprocesses started by the Batch Manager, various analysis processes in the flowchart, and the viewer as well.

Python call: Many parts of Mesmerize, such as the Batch Manager use external processes to run a python script. This setting sets which python call should be used. The default setting of "python3" should work for both *snap* and *pip*

installations unless you have written a custom expansion that uses python2.

Use CUDA: Use CUDA acceleration if you have a GPU with CUDA cores. Only works with the *pip installation*, and you must have pycuda and scikit-cuda (as well as the *nvidia CUDA toolkit*) installed. In Mesmerize CUDA is currently used only by Caiman Motion Correction. We plan to expand CUDA support to computationally intensive tasks that are performed by flowchart nodes.

Work dir: Many parts of Mesmerize use a working directory for temporary files. If you have a fast filesystem you can use that for this purpose.

Pre-run commands (large text entry): Mesmerize runs some computationally intensive tasks in subprocesses. These commands are run prior to the python script that performs the task.

- If you are using Mesmerize in a virtual environment (such as a *pip installed* Mesmerize) you will need activate the environment so you must include the line `source /path_to_venv/bin/activate` to the pre-run commands
- Similarly if you are using Mesmerize in an Anaconda environment you will need include commands to activate the environment.
- If you are using an Intel CPU you should get optimal performance by installing *Math Kernel Library (MKL)* and including `export MKL_NUM_THREADS=1` to the pre-run commands.
- If you are using an AMD CPU make sure you have OpenBLAS installed for optimal performance and include `export OPENBLAS_NUM_THREADS=1` to the pre-run commands. You may better performance by installing the *AMD specific libraries*.

Plugins directory: If you have a plugins dir include enter its path here.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`